→ Actas

JADICC

Jornadas Argentinas de
Didáctica de las Ciencias
de la Computación, 2021.

Fundación SADOSKY
<Program.AR/>
Ministerio de Ciencia, Tecnología e Innovación Argentina
CAF

Organización general

Hernán Czemerinski
Marcos J. Gómez

Gestión del evento

Mara Borchardt
Natalia Iocca
Mariano Benet

Evaluación de talleres

Cecilia Martínez
Gabriel Scarano

Sitio web

Jaqueline Schaab
Alicia Viana

Comunicación

Federico Rey
Facundo Manini
Valeria Saieg
Laura Rivillas

Infraestructura

Alfredo Sanzo
René Izarra
Pablo Carrai

Administración

Mariano Benet
Fundación Sadosky

Comité de programa

Gladys Dapozo (Presidente)
Pablo E. "Fidel" Martínez López (Presidente)
Araceli Acosta
Jorge Rodriguez
Marta Lasso
Claudia Banchoff
Pablo Turjanski
Carmen Leonardi
Francisco Bavera
Marcela Daniele
Laura Garbarini
Claudia Casariego
Hernán Ahumada
Gabriel Scarano
Analía Mendez
Fernando Puricelli
Ilda Flavia Millán
Cristina Werenitzky
Ana María Company
Emanuel Irrazabal
Alice Rambo
Gladis Sequeira
Dante Zanarini
Ana Casali
Mónica Tugnarelli
Cecilia Martínez
Sonia Santana
Noelia Pintos
Verónica Bollati
María Fernanda Golobisky
María de los Milagros Gutiérrez
Lucila Romero
Fernanda Carmona
Fernando Emmanuel Frati
Jacqueline Fernandez
Nora Reyes
Victoria S. Aragón

Editores

Marcos J. Gómez
Hernán Czemerinski

Actas de las Primeras Jornadas Argentinas de Didáctica de Ciencias de la Computación : JADiCC 2021 / Sandra Boari ... [et al.]. - 1a ed. - Ciudad Autónoma de Buenos Aires : Fundación Sadosky, 2022.
Libro digital, PDF

Archivo Digital: descarga
ISBN 978-987-27416-9-3

1. Didáctica. 2. Computación. 3. Formación Docente. I. Boari, Sandra.
CDD 004.0711

Actas de las Primeras Jornadas de Didáctica de Ciencias de la Computación

JADICC

Noviembre, 2021

Índice

1. Introducción **1**

2. Computación y sociedad **4**

Las Ciencias de la Computación en el diseño curricular de la provincia de Neuquén

Jorge Rodríguez, Marcos Cortez y Sandra Boari 5

***With a little help from my friends*: análisis de la comunicación por redes más allá de un curso**

Manuela Cerdeiro, Oscar Filevich, Rafael Grimson y Matías Lopez-Rosenfeld 12

La escuela y la brecha de género en la enseñanza de las Ciencias de la Computación

M. Emilia Echeveste, Marcos J. Gómez, Cecilia Martínez y Luciana Benotti 21

3. Entornos y enfoques **31**

Arduino en la Escuela: una herramienta versátil para la enseñanza de programación y robótica

Gonzalo Pablo Fernández, María Belén Ticona Oquendo y Christian Cossio-Mercado 32

Hacia un estado del arte sobre programación tangible

Leandro Castro, Verónica Artola, Silvia Bast y Gustavo Astudillo 45

CTFs en escuelas: una plataforma para acercar la ciberseguridad a la educación secundaria

Gabriela Suárez, Patricio Bolino, Jeremías Pretto, Paula Venosa y Claudia Queiruga 54

4. Entornos de programación **65**

Entornos programables para la modelización basada en agentes en educación: una revisión

Cristián Rizzi Iribarren 66

Arduino en la Escuela: una herramienta versátil para la enseñanza de programación y robótica

Martín Lobos, Silvia Bast y Gustavo Astudillo 78

Diseño de una arquitectura extensible y escalable para refundar el Proyecto Gobstones

Alan Rodas Bonjour y Federico Agustín Sawady O'Connor 86

¿Scratch, Python, o qué? Criterios para elegir un entorno para enseñar a programar a principiantes

Marcos J. Gómez y Pablo E. "Fidel" Martínez López 101

5. Enseñanza de programación con artefactos computacionales **115**

Uso de domótica como herramienta para la enseñanza de programación

Valentín Basel 116

Simulador de sumo para robótica educativa

Gonzalo Zabala y Ricardo Morán 126

Minirobots App: una aplicación educativa para aprender a programar jugando

Francisco Prieto, Fermín de Fiore, Paula Tristán y Laura Felice 136

AMULEN: robot educativo soberano

Carlos Maximiliano Correa, Pablo Leonel Etcheverry y Ariel Ferreira Szpiniak 147

6. Contenidos y didáctica **161**

Aportes para la construcción de la didáctica de las Ciencias de la Computación: un instrumento para el análisis de secuencias didácticas

Nerina Menchón, Carmen Leonardi y Virginia Mauco 162

El desarrollo de habilidades de resolución de problemas como un saber necesario en el camino del aprendizaje de la programación

Fernando Raúl Alfredo Bordignon y Alejandro Adrián Iglesias 176

Evaluación de producciones de docentes de primaria en formación en Pensamiento Computacional

Francisco Bavera, Teresa Quintero y Marcela Daniele 183

Programar es mucho más que solamente secuenciar comandos

Pablo E. "Fidel" Martínez López 191

7. Estrategias y enseñanza **205**

Prácticas de enseñanza de la Informática: Procesos de resignificación de los saberes pedagógicos y didácticos de profesionales - estudiantes del profesorado de Educación Secundaria en Informática

Silvina Cuello, Verónica Pacheco y Natalia Zalazar 206

Aplicaciones del Pensamiento Computacional en Problemas de Química

Pamela Viale, Claudia Deco y Cristina Bender 218

Reflexiones sobre el diseño e implementación de una especialización docente en enseñanza de la programación

Araceli Acosta, Cristián Rojo, Débero Cingolani y David Araya 227

8. Simposio Lationamericano **238**

Pensamiento computacional en educación primaria: el caso de Uruguay

Víctor Koleszar, Ana Laura Pérez Espagnolo y Emiliano Pereiro 239

El Pensamiento Computacional en el sistema educativo público Costarricense

Andrés Rodríguez y Natalia Zamora 247

Currículo de referência em tecnologia e computação: uma proposta do centro de inovação para a educação Brasileira

Christian Brackmann, André Raabe y Alice Carraturi 256

Trasfondo del Desarrollo de una Propuesta de Formación en Estándares TIC en Paraguay desde el Portal META de Paraguay Educa

Sascha Rosenberger y Carla Fernández 265

Educación a distancia. Reto de la superación a través del curso Algoritmización con Scratch

Rosa María Figueredo Rodríguez y Yor Alex Remond Recio 272

Proyecto IdeoDigital: Implementando ciencias informáticas en el sistema escolar público chileno

Andreas Hein, Claudia Jaña y Catalina Lyon 279

9. Primera sesión de pósters **287**

Aprendiendo a desarrollar un intérprete de un lenguaje de programación. Un software educativo para entender cómo funciona un lenguaje

Francisco Sánchez Guijarro, Lucas Spigariol, Sergio Viera, Juan Bono, Nicolás Ulmete y Adrián Bielsa 288

La participación en competencias: Un modelo de aprendizaje. El caso del Club ATP

Gustavo Cierra 290

Club de Ciencias: Una experiencia de enseñanza y fomento de vocaciones científicas y tecnológicas

Gustavo Cierra 291

Computadoras, Simulaciones y Programas Paralelos en Escuela Primaria

Marcos J. Gómez y Nicolás Wolovick 292

Educación digital, Programación y Robótica en el sistema educativo municipal

Alicia Olmos, Mariana J. Perez, Gabriel Caeiro, Moira Ragona, Ileana N. Gómez, Pablo Cabral, Adrián Vera, Ariel Martín, Erich Kunath y Nicolás Laje 293

Hacia una propuesta didáctica para la enseñanza de la programación

Gustavo Astudillo y Silvia Bast 294

***LudiNEAndo*: Modelo de abordaje ludificado para la capacitación docente en los niveles inicial y primario en CC**

Carina Mellibovsky y Julieta Lombardelli 296

***Misconceptions* de Ciencias de la Computación en niños/as escolarizados/as**

Lucía Parral, Herman Schinca, Fernando Schapachnik y Hernán Czemerinski 298

Pensamiento Computacional con CodyRoby

Marcelo Fabián Páez 300

Primeras experiencias de formación sobre la programación y su didáctica en docentes secundarios de San Juan

Flavia Millán, Manuel Ortega, Marcelo Mondre y Johana Arce 301

Proyectos de Programación en la Universidad: una Experiencia de Evaluación y Trabajo Colaborativo en Pandemia

Natalia Colussi, Pamela Viale y Natalia Monjelat 303

10. Segunda sesión de pósters **305**

Análisis de habilidades de pensamiento computacional en una asignatura de programación inicial en la universidad

Cristina L. Greiner, Gladys Dapozo, Raquel H. Petris, Maria Cecilia Espindola y Ana Maria Company 306

Aprendizaje automático para clasificación de actos de habla ilocutivos en mensajes de foros para educación a distancia

Mariano Piatti 307

Arte algorítmico

Fabio González 308

El abordaje de la alfabetización digital en dos propuestas de formación docente continua

Paola Roldán, Mariana Belluscio y Painé Pintos 309

Experiencia de introducción a la programación en formación inicial docente del Profesorado de Matemática

Silvina Elena Busto y Natalia Daiana Gomez 311

Nivel de satisfacción de los docentes en una capacitación en didáctica de la programación en tipos de pandemia

María Valeria Poliche, Hernán Cesar Ahumada, Daniel Armando Rivas, Nelson Contreras y Oscar Quinteros 312

Prácticas de prevención de phishing: una experiencia en la educación secundaria

Luciana Terreni 313

Taller de degustación. Programación Creativa

Marisa Elena Conde y Andrea Rocca 314

Zamba: una plataforma de aprendizaje como soporte a conceptos computacionales

Gustavo Del Dago, María Virginia Brassesco y Maximiliano Urso 315

Introducción

Los días 4, 5 y 6 de noviembre de 2021 se llevaron a cabo las Jornadas Argentinas de Didáctica de las Ciencias de la Computación (JADiCC) organizadas por la Iniciativa Program.AR de la Fundaciòn Sadosky. Debido a la situación derivada de la pandemia, el evento se desarrolló enteramente en modalidad virtual.

El objetivo de las jornadas fue propiciar el encuentro de investigadores, docentes, estudiantes y funcionarios públicos que trabajan en proyectos de investigación o de intervención educativa en la enseñanza de las Ciencias de la Computación. Las jornadas fueron la continuidad y ampliación temática de las Jornadas de Didáctica de la Programación organizadas en 2018 y 2019 en la Universidad Nacional de Quilmes y la Universidad Nacional de Córdoba.

Desde nuestro punto de vista, la programación es una parte importante de las Ciencias de la Computación, pero no es el único contenido de la disciplina. También lo son los referidos a las arquitecturas de las computadoras, el funcionamiento de las redes informáticas y la inteligencia artificial, por solo mencionar algunos ejemplos. Sin conocimientos sobre estos temas, la comprensión de la realidad se ve limitada y las y los ciudadanos no están en condiciones de participar activamente en los debates actuales sobre las interacciones entre la tecnología informática y la sociedad. El cambio de nombre responde a la importancia de que la incorporación de las Ciencias de la Computación en las distintas instancias educativas se lleve a cabo desde una perspectiva disciplinar amplia.

A las jornadas asistieron docentes argentinos de escuelas de nivel inicial, primaria, secundaria y nivel terciario que se encuentran dictando o tienen intenciones de dictar saberes vinculadas con Ciencias de la Computación; investigadoras, investigadores y docentes universitarios que han realizado reflexiones en el campo de la didáctica de las Ciencias de la Computación; y funcionarias y funcionarios del sector público que tienen a su cargo proyectos de inclusión de Ciencias de la Computación en las escuelas o programas de formación docente en este campo. En total, se registraron más de mil inscripciones y hubo representantes de todas las provincias argentinas, lo cuál es una enorme alegría y evidencia que las temáticas abordadas son de interés a lo largo y ancho del país.

En general, resulta sencillo investigar qué está sucediendo con esta temática en países como Inglaterra, Estados Unidos, Alemania, Israel y Finlandia, por mencionar algunos ejemplos. Sin embargo, no es fácil encontrar información sobre lo que ocurre en nuestros países vecinos. En esta edición, por primera vez abrimos la participación a países de la región. Estuvieron presentes miembros de equipos de países latinoamericanos que están llevando adelante iniciativas para incorporar a las Ciencias de la Computación a la escuela en una escala nacional. Junto a miembros de la **Iniciativa Program.AR**, participaron representantes de **Fundación Kodea** (Chile), **Plan Ceibal** (Uruguay), **Fundación Omar Dengo** (Costa Rica), **Asociación Civil Paraguay Educa** (Paraguay), **Universidad de Oriente** y **Universidad de las Ciencias Informáticas** (Cuba) y **Centro de Inovação para a Educação Brasileira** (Brasil). En el marco de las JADiCC inauguramos el **Simposio de Experiencias Latinoamericanas**, en el que cada institución expuso sobre el estado de situación en sus

respectivos países. El encuentro permitió intercambiar experiencias y reflexiones sobre temas de interés común, como programas de formación docente inicial o continua, propuestas curriculares, producción de material didáctico original y diseño de políticas públicas vinculadas a educación en computación. Además, se generaron lazos para encarar en conjunto futuros proyectos.

Durante el evento hubo tres conferencias principales: una a cargo **Annette Vee**, Profesora Asociada de Inglés y Directora del Programa de Composición de la Universidad de Pittsburgh en Pensilvania (Estados Unidos); una a cargo de **Luciana Benotti**, Profesora Asociada de Ciencias de la Computación de la Universidad Nacional de Córdoba e Investigadora en Inteligencia Artificial del CONICET (Argentina); y otra a cargo de **Ernesto Cuadros-Vargas**, Decano de la Facultad de Ingeniería de la Universidad San Ignacio Loyola de Lima (Perú) y miembro latinoamericano del *Steering Committee* de ACM/IEEE *CS Computing Curricula*. La disertación de Annette Vee giró en torno a las intersecciones entre la computación y la escritura, haciendo especial hincapié en cómo las herramientas teóricas de la alfabetización pueden ayudarnos a comprender la programación informática en sus contextos histórico, social y conceptual. Luciana Benotti abordó distintas problemáticas relativas a la Inteligencia Artificial y la posibilidad de que sean incorporadas en el ámbito escolar, tanto desde una perspectiva disciplinar como desde las implicancias éticas que se ponen en juego si estos saberes se incorporan a las escuelas. Finalmente, Ernesto Cuadros-Vargas brindó diversas perspectivas sobre las Ciencias de la Computación y la articulación que debería existir entre la Escuela y la Universidad para que el aprendizaje resulte eficaz.

También, se llevaron a cabo 3 conversatorios: *(i)* **¿Qué nos enseñó la pandemia? ¿Acaso aprender solo programación no es suficiente?**, a cargo de la Dra. Cecilia Martínez, Mg. Florencia Morado y la Lic. Laura Marés; *(ii)* **Dimensiones claves de la política educativa en 10 casos en el mundo de la inclusión de CC**, a cargo de la Dra. Cecilia Martínez (CONICET-UNC); y *(iii)* **Experiencias de Ciencias de la Computación en provincias argentinas**. A cargo de Tucumán (PLaNEA), CABA (TI) y Neuquén (Nueva Escuela Secundaria). En todos ellos se recuperaron interesantes miradas sobre temas emergentes en relación a la inclusión de las Ciencias de la Computación en el contexto escolar.[1]

Por otro lado, se recibieron un total de 31 artículos de muy buena calidad para su presentación en las jornadas. La selección no resultó nada sencilla. La decisión final de aceptación, a cargo de los presidentes del comité de programa, se hizo en base a una ponderación de múltiples revisiones y una discusión posterior. 21 artículos fueron aceptados para ser publicados y presentados en las jornadas. De los 10 restantes, 4 fueron recomendados para ser presentados como pósters y 6 fueron rechazados. Las presentaciones se realizaron en 6 sesiones distintas, agrupando a los artículos de acuerdo a la temática abordada: *(i)* **Computación y sociedad**; *(ii)* **Entornos y enfoques**; *(iii)* **Entornos de programación**; *(iv)* **Enseñanza de programación con artefactos computacionales**; *(v)* **Contenidos y didáctica**; y *(vi)* **Estrategias y enseñanza**. Por otro lado, se recibieron 20 propuestas de pósters, de los cuales 16 fueron aceptados; sumados a los 4 que cambiaron su

[1]Las conferencias principales y los conversatorios se encuentran grabados y disponibles en la página web de Program.AR.

formato de artículo a póster, en total se presentaron 20 pósters.

Además, en el marco de las jornadas se dictaron talleres destinados a docentes que consistieron en capacitaciones breves sobre temáticas organizadas alrededor de actividades de construcción de una propuesta didáctica. En total se recibieron 21 postulaciones de las cuales se seleccionaron 10: *(i)* **Diseño de actividades personalizadas para el entorno GobstonesWeb para trabajar con estudiantes**, a cargo de Pablo E. "Fidel" Martínez López de la Universidad Nacional de Quilmes; *(ii)* **Introducción práctica al aprendizaje automático en entornos de bloques**, a cargo de Matías Bordone Carranza y Natalia Zalazar del Instituto Superior de Estudios Pedagógicos (ISEP), Córdoba; *(iii)* **Un acercamiento a la ciencia de datos desde la escuela secundaria**, a cargo de Isabel Miyuki Kimura, Ulises Cura Jáuregui, Yessica Borghi, Agustín Leonardo Cao y Antonella Garea de la Universidad Nacional de La Plata; *(iv)* **CARDIAC, una computadora de papel**, a cargo de Nicolás Wolovick y Valentín Base de la Universidad Nacional de Córdoba; *(v)* **SETPOINT, Introducción a los sistemas de control**, a cargo de José Daniel Villagran Lazzarone y Hernán Acosta del Instituto Técnico de la Universidad Nacional de Tucumán; *(vi)* **Wollok, Un entorno de "objetos" para aprender a programar**, a cargo de Lucas Spigariol, Nicolás Passerini y Nahuel Palumbo de la Universidad Tecnológica Nacional - Facultad Regional Buenos Aires y en la Universidad Nacional de San Martín; *(vii)* **Creando música por medio de la programación con Sonic Pi**, a cargo de Cristián Rojo y Eduardo Rodríguez Pesce de la Universidad Nacional de Córdoba; *(viii)* **Introducción a infraestructura para aplicaciones IoT**, a cargo de Iván Sambrana y José Manuel Alonso de la Universidad Nacional del Nordeste; *(ix)* **Escritura de contenidos educativos sobre ciencias de la computación aplicando perspectiva de género**, a cargo de Daniela Villani, Rocío Gonzalez y Franco Bulgarelli del Proyecto Mumuki; y *(x)* **Estructuras de control en programación, implementación con robótica educativa y Arduino utilizando un simulador online**, a cargo de Mariela Burghardt, Fernando Princich y Guillermo Andrés Sampallo de la Universidad Nacional del Nordeste.

Queremos agradecer a todas y todos los que hicieron posible la realización de las jornadas. A Natalia Iocca y Mariano Benet por haberse ocupado de las cuestiones operativas del evento; a Pablo E. "Fidel" Martínez Lopez y Gladys Dapozo por haberse hecho cargo de la enorme tarea de presidir el comité de programa; a Cecilia Martínez y Gabriel Scarano por haber realizado la curaduría de los talleres; a Jacky Schaab y Alicia Viana por haber creado la identidad gráfica del evento; a Federico Rey, Facundo Manini, Valeria Saieg y Laura Ramírez Rivillas que formaron un gran equipo de comunicación; a Alfredo Sanzo, René Izarra y Pablo Carral, que estuvieron a cargo de la infraestructura tecnológica que funcionó a la perfección; y a todo el equipo de administración de la Fundación Sadosky, por todas las gestiones administrativas.

Nos despedimos con el mismo entusiasmo que tuvimos desde el primer día que comenzamos a pensar las jornadas, y esperando verlos a todos de nuevo en JADiCC 2022.

Lic. Mara Borchardt Dr. Marcos Javier Gómez Dr. Hernán Czemerinski

SESIÓN 1:
COMPUTACIÓN Y SOCIEDAD

Moderadora: *Mg. María Carmen Leonardi (UNICEN)*

Las Ciencias de la Computación en el diseño curricular de la provincia de Neuquén
Jorge Rodríguez, Marcos Cortez y Sandra Boari

***With a little help from my friends*: análisis de la comunicación por redes más allá de un curso**
Manuela Cerdeiro, Oscar Filevich, Rafael Grimson y Matías Lopez-Rosenfeld

La escuela y la brecha de género en la enseñanza de las Ciencias de la Computación
M. Emilia Echeveste, Marcos J. Gómez, Cecilia Martínez y Luciana Benotti

Las Ciencias de la Computación en el diseño curricular de la provincia de Neuquén

Jorge Rodríguez*
j.rodrig@fi.uncoma.edu.ar
Universidad Nacional del Comahue

Marcos Cortez†
cortezmarcos@gmail.com
Ministerio de Educación
de la Provincia de Neuquén

Sandra Boari†
sandraboari@gmail.com
Ministerio de Educación
de la Provincia de Neuquén

Resumen

Las iniciativas curriculares tendientes a mejorar la participación de las Ciencias de la Computación en el ámbito de educación obligatoria está captando la atención en todo el mundo. En este contexto en Argentina, las universidades nacionales, la inicativa Program.AR, el Consejo Federal de Educación y los gobiernos provinciales vienen articulando esfuerzos para hacer que la educación en informática resulte accesible y significativa para más estudiantes de la educación inicial, primaria y secundaria en todo el país.

En el período 2016 - 2019, la provincia del Neuquén concreta la redacción de su primer Diseño Curricular del Nivel Secundario que determina la incorporación de la Informática como espacio curricular en todas las modalidades de la enseñanza secundaria adoptando un paradigma de enseñanza anclado en las bases científicas y tecnológicas que proporcionan los campos que conforman las Ciencias de la Computación.

En este trabajo se presenta una revisión sobre el Diseño Curricular Jurisdiccional para las Escuelas Secundarias de la Provincia del Neuquén, en el que se analiza la presencia de las Ciencias de la Computación en la propuesta formativa. En este proceso se definen como dimensiones de análisis las metas y propósitos, la asignación de horas cátedra durante el trayecto de la educación secundaria, la presencia de las áreas de conocimiento de la disciplina y los enfoques y paradigmas curriculares planteados.

Palabras clave: Ciencias de la Computación, Diseño Curricular, Secundaria Neuquén.

1. Introducción

Las iniciativas que buscan ampliar la participación de las Ciencias de la Computación en las propuestas curriculares están captando la atención de gobiernos nacionales, organizaciones sin fines de lucro y centros de investigación

*Grupo de Investigación en Lenguajes e Inteligencia Artificial Departamento de Teoría de la Computación - Facultad de Informática. Universidad Nacional del Comahue. Buenos Aires 1400, Neuquén, Argentina.

†Consejo Provincial de Educación. Ministerio de Educación de la Provincia de Neuquén. Belgrano 1300, Neuquén, Argentina

en todo el mundo. Si bien algunos países tienen una trayectoria extensa en este campo, en muchos la incorporación de la computación de forma rigurosa y sostenida en la educación obligatoria es un proceso en desarrollo (Sadosky, 2013; Royal Society, 2017; K-12 Computer Science Framework Steering Committee, 2016).

En argentina las universidades nacionales, Program.AR, el Consejo Federal de Educación y los gobiernos provinciales vienen articulando esfuerzos para hacer que la educación en informática resulte accesible y significativa para más estudiantes de la educación inicial, primaria y secundaria en todo el país. Sin embargo, la definición de la estructura curricular es potestad de cada jurisdicción, por lo que la posición de la computación en los planes de estudio es dispar y en general está poco representada (Consejo Federal de Educación, 2015 y 2018).

La provincia del Neuquén asume la responsabilidad, en el período 2016 - 2019, de concretar la redacción de su primer Diseño Curricular del Nivel Secundario en concordancia a los acuerdos del Consejo Federal de Educación y en base a la Ley 26.206 - Ley de Educación Nacional (Congreso de la Nación Argentina, 2006). En este devenir es que se determina la incorporación de la Informática como espacio curricular en todas las modalidades de la enseñanza secundaria estableciendo un paradigma de enseñanza anclado en las bases científicas y tecnológicas que proporcionan los campos que conforman las Ciencias de la Computación.

En este trabajo se presenta una revisión sobre el Diseño Curricular Jurisdiccional para las Escuelas Secundarias de la Provincia del Neuquén, en el que se analiza la presencia de las Ciencias de la Computación en la propuesta formativa. En este proceso se definen como dimensiones de análisis las metas y propósitos, la asignación de horas cátedra durante el trayecto de la educación secundaria, la presencia de las áreas de conocimiento de la disciplina y los enfoques y paradigmas curriculares planteados.

El Diseño Curricular del Nivel Secundario (Consejo Provincial de Educación, 2018) define que la forma de organización y estructuración de los conocimientos y saberes se constituyen a través de la conformación de Áreas de Conocimientos. Desde la perspectiva de los paradigmas curriculares para la computación en las escuelas, esta propuesta curricular adopta el de la Enseñanza de las Ciencias de la Computación, proponiendo un recorrido amplio por las áreas de conocimiento de la disciplina a lo largo del trayecto de la educación secundaria en Neuquén.

En adelante, el trabajo se estructura de la siguiente manera. En la sección siguiente, se describe el contexto que da marco a la introducción de las Ciencias de la Computación en la educación obligatoria. A continuación, en la Sección 3, se analiza la posición de la informática en el Diseño Curricular para las Escuelas Secundarias en la provincia del Neuquén. Finalmente, se presentan las conclusiones elaboradas a partir de esta revisión.

2. Contexto

Los argumentos en relación a establecer a la computación como parte de las propuestas curriculares para la educación secundaria resultan convincentes y ganan adhesión en todo el mundo. Las razones que participan de estas argumentaciones son de carácter económico, social, político y cultural, dando lugar al surgimiento de diferentes paradigmas, enfoques y modelos para la integración de la computación en el currículum escolar.

Enfocando la atención sobre los roles y propósitos de la computación en la escena escolar es posible desagregar la presencia de la informática en el currículum en cuatro paradigmas aceptados en el campo de Computing Sciences Education: Mejoramiento de los Aprendizajes, Alfabetización Digital, Desarrollo de Competencias TIC y Enseñanza de las Ciencias de la Computación.

Mejoramiento de la Calidad de los Aprendizajes, se trata de un tipo de inclusión centrada en la producción de

respuestas pedagógicas y didácticas para avanzar en el mejoramiento de los procesos de las enseñanzas en diferentes áreas de conocimiento (Maggio, 2012). Alfabetización digital, plantea la necesidad de desarrollar habilidades básicas para usar dispositivos digitales con fluidez y seguridad. En este paradigma se ubican las capacidades para usar software de oficina, de comunicación y navegadores web, entre otras (Royal Society, 2017). La alfabetización digital, además incluye a las habilidades tecnológicas necesarias para contribuir a reducir las brechas digitales (UNESCO, 2015).

Desarrollo de Competencias TIC, significa desarrollar las capacidades suficientes para manipular sistemas digitales para satisfacer las necesidades relacionadas a campos específicos de la industria, el comercio o el arte (Royal Society, 2017).

Enseñanza de las Ciencias de la Computación, plantea considerar a la informática como una disciplina académica rigurosa que abarca conceptos y prácticas fundamentales de la computación. Las últimas tendencias curriculares procuran superar la preponderancia del área de conocimiento Algoritmos y Programación proponiendo un recorrido amplio por las Ciencias de la Computación (Royal Society, 2017; Tissenbaum y Ottenbreit-Leftwich, 2020).

Inevitablemente, los paradigmas conviven dentro de una propuesta curricular y posiblemente los límites resulten difusos en algunos casos, sin embargo estas demarcaciones resultan útiles para describir y comprender el problema de la incorporación de las Ciencias de la Computación en la educación obligatoria.

A partir de los consensos construidos acerca de la necesidad de establecer a la informática como disciplina escolar, emergen los debates en relación a la posición en el currículum. Los modelos integrados, es decir incorporar contenidos sobre computación en otros espacios curriculares, suelen conducir a la disolución de los contenidos computacionales en los planes de estudio (Webb et al., 2018). En este sentido, toman fuerza los modelos independientes que se sostienen sobre los argumentos que plantean posicionar las Ciencias de la Computación como una asignatura distinta en la escuela secundaria (Webb et al., 2018).

En Argentina, la Ley de Educación Nacional dispone que la revisión y definición de la estructura curricular es responsabilidad de cada jurisdicción. En este contexto se observa, que la integración de la computación en las propuestas curriculares varía en las diferentes provincias, produciendo un alto grado de dispersión en relación a los paradigmas, enfoques y modelos adoptados.

En la mayoría de las jurisdicciones predomina el modelo integrado donde se observa una definición poco clara de la informática como disciplina académica, 19 provincias adoptan total o parcialmente este enfoque, integrando los contenidos informáticos en espacios curriculares destinados al área de Tecnología (Rodríguez y Cortez, 2020).

3. La Computación en el Diseño Curricular para la escuela secundaria en Neuquén

El Diseño Curricular[1] abarca las modalidades del nivel secundario definidas en la Ley 26.206 - Ley de Educación Nacional que se corresponden a: Educación Secundaria Orientada (ESO), Educación Técnica Profesional (ETP), Educación Artística y Educación Permanente de Jóvenes y Adultos (EPJA).

A su vez, el ciclado de cada modalidad se organiza de la siguiente manera:

En la tabla se observa que la organización del ciclado incorpora a los ya existentes Ciclo Básico Común y Ciclo Orientado, en tercer año un “Interciclo” en carácter de Enlace Pedagógico, con el objetivo de combinar Espacios

[1]Resolución N° 1463/2018, p. 67. Consejo Provincial de Educación del Neuquén

Modalidad	Ciclo Básico Común	Interciclo	Ciclo Orientado
Educación Secundaria Orientada	1 y 2 año	3 año	4 y 5 año
Educación Técnica Profesional	1 y 2 año	3 año	4 a 6 año
Educación Artística	1 y 2 año	3 año	4 a 6 año
Educación Permanente de Jóvenes y Adultos (3 años)	1 y 2 año	-	3 año
Educación Permanente de Jóvenes y Adultos (4 años)	1 y 2 año	-	3 y 4 año

Tabla 1: Educación Secundaria - Ciclado por modalidad

Curriculares de formación específica en el marco de la formación general[2].

En cuanto a los campos disciplinares, el Diseño Curricular establece y define que la forma de organización y estructuración de los mismos se constituye a través de Áreas de Conocimientos, entendiéndose a éstas como una concurrencia de disciplinas para la enseñanza de los contenidos en la escuela secundaria que, sin violentar la disciplina, potencia el abordaje de problemas complejos que requieren los aportes interdisciplinares. El área, es pues, un espacio de reunión y aproximación entre docentes que la integran, con el propósito principal de construir un enfoque y una organización interdisciplinaria en la enseñanza de sus respectivas disciplinas.

En este sentido, se contemplan los espacios áulicos correspondientes a los campos disciplinares y se incorporan los Espacios Pedagógicos Articulados (EPA)[3] conformados por las diferentes disciplinas partícipes de un área que requiere del trabajo colectivo de docentes, en pos de construir una propuesta que, incluyendo la especificidad de cada disciplina, las supera y complejiza.

Las áreas definidas en el Diseño Curricular jurisdiccional son las siguientes: Ciencias Sociales, Políticas y Económicas; Ciencias Naturales; Lenguajes y Producción Cultural; Matemática e Informática; Integración Tecnológica; Educación Física Integral; Tecnología y Tecnología Agropecuaria.

Los contenidos relacionados al paradigma Enseñanza de las Ciencias de la Computación se encuentran contemplados en el área Matemática e Informática. El área Integración Tecnológica[4] , presente sólo en la ESO, incluye contenidos de los Núcleos de Aprendizajes Prioritarios de Educación Tecnológica que engloba los paradigmas Alfabetización Digital y Desarrollo de Competencias TIC.

En el Marco Socio Político Pedagógico del Diseño Curricular se establece el paradigma Mejoramiento de la Calidad de los Aprendizajes[5], donde el tema central se refiere a cómo favorecer el uso de las tecnologías con sentido pedagógico. Por tanto, el mismo se presenta de manera transversal a todas las áreas de conocimientos.

[2]Resolución N° 1463/2018, p. 371. Consejo Provincial de Educación del Neuquén.

[3]Resolución N° 1463/2018, p. 371. Consejo Provincial de Educación del Neuquén.

[4]Resolución N° 1673/2019, Anexo II p. 9. Consejo Provincial de Educación del Neuquén.

[5]Resolución N° 1463/2018, p. 31. Consejo Provincial de Educación del Neuquén.

3.1. El lugar de las Ciencias de la Computación

El Diseño Curricular establece la conformación del área Matemática e Informática[6] fundamentando que el Paradigma de Enseñanza de las Ciencias de la Computación, subyacente a la Informática, relaciona ambas disciplinas en su dimensión epistemológica. En efecto esta relación se fundamenta en que, en ambas disciplinas se comparte primero, la forma en que construyen conocimientos partiendo de la resolución de problemas y segundo, en que disponen de diferentes formas de representar los objetos de conocimiento, constituyéndose éstos en los ejes principales con los que concurren en un todo de acuerdo según lo mencionan (Nieto y José, 2005).

El espacio curricular destinado a la Informática se incorpora en el Ciclo Básico Común y en el Interciclo de todas las modalidades, como un espacio disciplinar que adopta el paradigma de la Enseñanza de las Ciencias de la Computación. Se lo concibe como un Campo Disciplinar superador del paradigma utilitario que recupera la dimensión epistemológica de la disciplina[7], reconociendo entonces la existencia de un cuerpo de conocimientos estables que trascienden los cambios tecnológicos que caracterizan a la misma.

La disciplina Informática considera el abordaje de los siguientes campos de conocimientos de las Ciencias de la Computación: Algoritmos y Programación; Arquitectura de Computadoras y Sistemas Operativos; Redes de computadoras e Internet; Seguridad Informática; Bases de Datos; Ingeniería de Software; Inteligencia Artificial y Software Libre.

En cuanto a las horas cátedras asignadas al espacio disciplinar se puede observar la siguiente distribución:

- En la modalidad ESO se totalizan en todo el trayecto de los cinco años de educación secundaria entre 396 y 468 horas cátedra de formación en computación.
- En la modalidad ETP se totalizan en todo el trayecto de los seis años de educación secundaria entre 288 y 720 horas cátedra de formación en computación.
- En la modalidad Artística se totalizan en todo el trayecto de los seis años de educación secundaria 252 horas cátedra de formación en computación.
- En la modalidad EPJA se totalizan en todo el trayecto de los tres años de educación secundaria entre 72 y 108 horas cátedra de formación en computación, mientras que para la totalidad del trayecto de los cuatro años de educación secundaria se totalizan entre 144 y 288 horas cátedra.

4. Discusión

En esta sección se discuten los resultados obtenidos con intención de avanzar en un proceso de análisis de la situación actual. La revisión desarrollada evidencia que, más allá de las iniciativas y políticas públicas concretadas durante los últimos años, aún existe un grado de dispersión importante en referencia al lugar que se le asigna a la disciplina en el currículum escolar.

4.1. Presencia del Paradigma de Enseñanza de Ciencias de la Computación

El caso Neuquén presenta una situación excepcional en el contexto nacional. Mientras que en el país, la computación está poco representada en el nivel secundario, presentando un alto grado de dispersión en relación a la selección

[6]Resolución N° 1463/2018, p. 252. Consejo Provincial de Educación del Neuquén.

[7]Resolución N° 1463/2018, p. 281. Consejo Provincial de Educación del Neuquén.

de contenidos, en Neuquén se asigna un espacio curricular con presencia en todas las modalidades de la escuela secundaria ofreciendo un amplio recorrido por las áreas de conocimiento de las Ciencias de la Computación.

4.2. Convergencia de paradigmas y enfoques

Se observa que no existe un predominio de un paradigma sobre otro, en tanto que éstos se complementan en las propuestas formativas que corresponden a otros espacios disciplinares. Sin embargo, la presencia del paradigma de Enseñanza de la Ciencias de la Computación sostiene su especificidad a través de la asignación de espacios curriculares propios.

4.3. Formación Docente Inicial y Continua

Para acompañar la implementación resulta necesario impulsar dos líneas de acción: por un lado formación a docentes que ya se encuentran en el sistema educativo. En este sentido se llevó a cabo una Especialización Docente en Didáctica de las Ciencias de la Computación, un Encuentro de Docentes de Ciencias de la Computación y se están desarrollando actualmente Encuentros de Coformación. Por otro lado, en el ámbito de la Formación Docente Inicial, la jurisdicción impulsa el Diseño Curricular del Profesorado de Informática que se implementará el próximo año.

5. Conclusiones

En este artículo, se presenta una revisión sobre el lugar de la computación en el Diseño Curricular para la Educación Secundaria en la provincia del Neuquén. Se presta especial atención sobre la presencia del paradigma Enseñanza de las Ciencias de la Computación.

El principal resultado obtenido consiste en identificar la asignación de espacios curriculares destinados específicamente a la enseñanza de la disciplina que proponen un recorrido amplio por las áreas de conocimiento de las Ciencias de la Computación. En esta dirección se identifica a la formación docente como un campo de intervención que mejora las posibilidades de concreción de las iniciativas en desarrollo.

Bibliografía

Congreso de la Nación Argentina (2006).Ley 26.206 - Ley de Educación Nacional.

Consejo Federal de Educación (2015). Resolución N° 263/2015. Resoluciones CFE.

Consejo Federal de Educación (2018). Resolución N° 343/2018. Resoluciones CFE.

Consejo Provincial de Educación del Neuquén (2018). Resolución N° 1463/2018. Establece el Diseño Curricular Jurisdiccional del Nivel Secundario. Ciclo Básico Común e Interciclo.

Consejo Provincial de Educación del Neuquén (2019). Resolución N° 0603/2019. Establece las orientaciones de la Educación Secundaria para las modalidades de Educación Secundaria Orientada y, Educación Técnico Profesional.

Consejo Provincial de Educación del Neuquén (2019). Resolución N° 0809/2019. Establece las orientaciones para la modalidad Educación Permanente de Jóvenes y Adultos.

Consejo Provincial de Educación del Neuquén (2019). Resolución N° 1044/2019. Aprueba los documentos de las Orientaciones de la modalidad Escuela Secundaria Orientada del Diseño Curricular Jurisdiccional del Nivel Secundario.

Consejo Provincial de Educación del Neuquén (2019). Resolución N° 1045/2019. Aprueba los documentos de las Orientaciones de la modalidad Escuela Técnico Profesional del Diseño Curricular Jurisdiccional del Nivel Secundario.

Consejo Provincial de Educación del Neuquén (2019). Resolución N° 1046/2019. Aprueba los documentos de las Orientaciones de la modalidad Enseñanza Permanente de Jóvenes y Adultos del Diseño Curricular Jurisdiccional del Nivel Secundario.

Consejo Provincial de Educación del Neuquén (2019). Resolución N° 1673/2019. Establece la modalidad Artística y sus Orientaciones. Aprueba el espacio de Integración Tecnológica y otras orientaciones de la modalidad Escuela Técnico Profesional.

Consejo Provincial de Educación del Neuquén (2019). Resolución N° 0511/2020. Aprueba el documento de la orientación Técnico/a en Gestión de Proyectos Audiovisuales de la modalidad Artística del Diseño Curricular Jurisdiccional del Nivel Secundario.

Consejo Provincial de Educación del Neuquén (2020) Resolución N° 0636/2020. Aprueba documentos de Orientaciones de la modalidad Escuela Técnico Profesional del Diseño Curricular Jurisdiccional del Nivel Secundario.

Maggio, M. (2018). Habilidades del siglo XXI: cuando el futuro es hoy: documento básico. XIII Foro Latinoamericano de Educación

Nieto S., y José H. (2005). Resolución de problemas, Matemática y Computación. Enl@ce: Revista Venezolana de Información, Tecnología y Conocimiento, 2(2),37-45.[fecha de Consulta 22 de Agosto de 2021]. ISSN: 1690-7515.

K-12 Computer Science Framework Steering Committee (2016). K-12 computer science framework. ACM.

Royal Society (2017). After the reboot: Computing education in UK schools. Policy Report

Rodríguez, J., y Cortez, M. (2020). La posición de las Ciencias de la Computación en el Diseño Curricular para la Escuela Secundaria Argentina: Una Revisión Sistemática. Electronic Journal of SADIO (EJS), 19(2), 136–150.

Sadosky, F. (2013). Una propuesta para refundar la enseñanza de la computación en las escuelas Argentinas. Reporte de la Fundación Sadosky, 23.

Tissenbaum, M., y Ottenbreit-Leftwich, A. (2020). A Vision of K-12 Computer Science Education for 2030. Communications of the ACM,63 (5).

UNESCO (2015). Educación 2030: Hacia una educación inclusiva y equitativa de calidad y un aprendizaje a lo largo de la vida para todos. UNESCO DOC | Biblioteca Digital

Webb M., Bell T., Davis N., Katz Y, Reynolds N., Chambers D, Mori N. (2018). Computer Science in the School Curriculum: Issues and Challenges. En Tomorrow's learning: Involving everyone. learning with and about technologies and computing: 11th ifip tc 3 world conference on computers in education, wcce 2017, dublin, ireland, july 3-6, 2017, revised selected papers (Vol. 515, p. 421). Springer.

With a little help from my friends: análisis de la comunicación por redes más allá de un curso

Manuela Cerdeiro*†
cerdeiro@dm.uba.ar
UNSAM - UBA

Oscar Filevich*‡
ofilevich@unsam.edu.ar
UNSAM - CONICET

Rafael Grimson*‡
rgrimson@unsam.edu.ar
UNSAM - CONICET

Matías Lopez-Rosenfeld*†‡
mlopez@dc.uba.ar
UNSAM - UBA
CONICET

Resumen

Este trabajo propone una exploración del uso de los grupos de mensajería por parte de estudiantes en el marco de un curso de Programación en Python dictado en la Universidad Nacional de San Martín. Éste se realizó en modalidad virtual durante la pandemia de coronavirus y se caracterizó por tener cientos de estudiantes y material principalmente asincrónico.

En primer lugar, analizamos el uso de la mensajería a través del medio oficial de comunicación de la materia –la plataforma Slack–. Ésta fue la plataforma de comunicación más frecuente, donde se observó una gran proporción de miembros activos a lo largo de la cursada. Por esta plataforma se difundió la información oficial de la materia y se crearon canales para consultas, organizados por unidad temática. En estos canales se buscó generar un clima de trabajo colaborativo, donde docentes y estudiantes participaran en responder una pregunta dada, y donde las respuestas consistieran en ideas o propuestas para reflexionar sobre los ejercicios en lugar de brindar solución a los mismos.

Por otro lado, estudiamos el uso de los grupos alternativos, no oficiales, formados exclusivamente por estudiantes. El análisis de redes informales se basa en dos grandes clases: grupos grandes y grupos chicos. Se desprende de nuestro análisis que el tamaño de los grupos condiciona la forma de participar: en los grupos pequeños se observa una mayor distensión mientras que en los grandes se ve un intercambio más específico.

El análisis llevado a cabo en este trabajo se basa en las estadísticas de uso brindadas por la plataforma Slack, así como en encuestas realizadas a los estudiantes sobre el uso de redes, indagando tanto sobre la frecuencia de acceso y el tipo de intervenciones, como sobre la percepción respecto de la utilidad de cada medio. Los resultados indican que el uso de las redes generó un impacto positivo significativo en la experiencia de cursada.

Finalizamos el trabajo con una discusión sobre desafíos de estas redes, formas de incentivarlas y preguntas que surgen de esta experiencia.

*Universidad Nacional de San Martín.

†Universidad de Buenos Aires.

‡Consejo Nacional de Investigaciones Científicas y Técnicas.

Palabras clave: Programación, Asincrónico, Comunicación, Redes, Grupos.

1. Introducción

El año 2020 trajo una gran sorpresa a las instituciones educativas del mundo: una pandemia. Este abrupto e inesperado suceso visibilizó palabras referidas a productos antes prácticamente desconocidas: *Zoom*, *Meet*, *Discord*, mientras que otras cobraron un poco más relevancia como: *asincrónico*, *sincrónico*, *encuentro virtual*, entre otros.

En este contexto y sin experiencia previa ni preparación alguna, todo sistema educativo buscó alguna forma de afrontar el distanciamiento y aislamiento impuesto por la situación sanitaria. Esta situación es novedosa a nivel mundial y está empezando a ser documentada (Ghounane, 2020), pero los efectos y consecuencias de esta realidad serán observables recién cuando se puedan normalizar las dinámicas educativas.

En particular, en este trabajo recuperamos experiencias acontecidas en la materia de Programación en Python de la Escuela de Ciencia y Tecnología de la Universidad Nacional de San Martín en Argentina. Ante la imposibilidad de dar el curso en el aula, el curso del segundo semestre de 2020 se rediseñó completamente basado en una modalidad virtual. Al utilizar una estrategia fuertemente asincrónica se decidió abrir la inscripción más allá de la población de la Universidad (Cerdeiro, Crespo, Grimson, Filevich, y López-Rosenfeld, 2021).

Esta apertura provocó un aluvión de inscripciones, en parte inesperado, y en parte explicable. Inesperado porque se pasó de ser un curso de 60 personas a tener 1200 inscriptos en 2020 y 1500 en la primera edición de 2021. Explicable, por algunas razones como:

- Es un curso de programación, que hoy en día es una herramienta muy útil.
- Es un curso dictado de modo asincrónico, lo que reconoce diversos tiempos y ritmos de estudiantes.
- El curso forma parte de la currícula de una prestigiosa Universidad Nacional.
- Se trata de un curso gratuito, financiado íntegramente por el Estado Argentino. A diferencia de cursos extranjeros en los que es posible solicitar una beca, este curso no requiere ningún tipo de solicitud de beca, ni tarjeta de crédito para cursarlo.

La enorme cantidad de estudiantes tornaba imposible una relación directa docente-estudiante ya que el curso contaba con entre 5 y 8 docentes por edición. El contenido del curso está comprendido en el material escrito y publicado online con texto teórico y ejercicios a realizar y entregar. Además se realizaba un encuentro sincrónico semanal en una sala masiva de Zoom (en la que no entraban todos los estudiantes) y que se transmitía en vivo por YouTube (quedando allí disponible para una cursada asincrónica). Quienes estaban en la sala podían participar activando su cámara y micrófono, pero quienes lo seguían por YouTube lo hacían vía comentarios escritos en el chat de YouTube que eran respondidos por el mismo medio o restransmitidos al docente que estaba dictando la clase para ser respondidos verbalmente. La naturaleza de la infraestructura hace que sea un requisito la aceptación tecnológica (Ramírez, 2021), dejando fuera, lamentablemente, a quienes no se sientan cómodos o tengan rechazo por la virtualidad.

Esta situación dista mucho de ser la ideal para un vínculo fluido, lo que tensionó nuestros modos de enseñar programación y nos forzó a repensar completamente nuestras propias prácticas docentes para abordar la enseñanza. Recuperando el consejo de una de las fundadoras de ExactasPrograma –una iniciativa de la Facultad de Ciencias Exactas y Naturales de la UBA para la enseñanza de la programación desde un abordaje práctico (López-Rosenfeld et al., 2021) – dirigido a los alumnos y refiriéndose a la enseñanza en contexto virtual: *"Si necesitan ayuda avisen, pero sepan que son un montón. Así que mientras llegamos a ustedes hagan como cuando llaman al auxilio del auto,*

vayan viendo si lo pueden solucionar por su cuenta.". En ese sentido, al inicio de cada cursada se recomendaba que se generen grupos de intercambio entre estudiantes por fuera de la estructura del curso. La existencia de redes como parte del curso son ideas ya exploradas en la pre-pandemia (Casey y Evans, 2011; Barnes, 2012; Lego Muñoz y Towner, 2010; Greenhow, 2011), en este trabajo nos concentramos en el contexto de aislamiento desde una universidad en Argentina.

El curso manejaba sus consultas principalmente a través de la red de mensajería Slack (Slack Technologies, 2009). Se trata de una herramienta de comunicación en equipo que se organiza en canales (salas de chat), que fueron organizados por unidades. Permite además el envío de mensajes directos (privados) y cuenta con una herramienta de búsqueda entre todo el historial de mensajes. Esta herramienta es paga, pero fue utilizada en su versión gratuita. En un futuro sería ideal poder acceder a servicios gratuitos y de software libre para estar en línea con la filosofía del curso. Algunas opciones como MatterMost (Mattermost Inc, 2015) fueron consideradas, pero descartadas por requerir de un servidor propio para su instalación y administración del mismo.

Este trabajo propone una exploración del uso y apropiación de las redes de mensajería utilizadas por estudiantes durante el curso. Primeramente se explorará el uso de la mensajería oficial (Slack), luego las redes alternativas separando los casos entre grupos grandes y grupos chicos de estudiantes. Finalizaremos con una discusión sobre desafíos de estas redes, formas de potenciarlas y líneas de investigación futuras a la luz de la exploración de este trabajo.

2. Materiales y métodos

En este trabajo utilizamos las respuestas a un cuestionario realizado al finalizar el curso del primer semestre de 2021. El mismo fue respondido por 186 estudiantes y se realizó de manera voluntaria luego de la devolución de notas utilizando la plataforma *Google Forms*. Además se tomaron los datos que ofrece Slack sobre su uso.

La encuesta estaba separada en tres secciones. En la primera se preguntaba cuál fue la importancia (o el rol) que tomaron distintas herramientas de mensajería (oficiales y no oficiales). En la segunda se indagaba sobre la opinión que tenían del uso del Slack: frecuencia de acceso, si fue útil, el tipo de participación (leyendo, preguntando, respondiendo), si recomiendan su uso y un campo de comentarios libres. En la tercera sección se buscó capturar características sobre las redes no oficiales usadas: cantidad de participantes, frecuencia de acceso, utilidad, y la opción de un campo de comentarios libres.

Para el análisis de los datos se utilizó Python3 (Van Rossum y Drake, 2009) en el entorno Jupy- ter (Community, 2020) utilizando la biblioteca Pandas (Wes McKinney, 2010), y las bibliotecas gráficas MatPlotLib (Hunter, 2007) y Seaborn (Waskom, 2021).

3. Análisis de la redes

La comunicación oficial del curso se realizó utilizando Slack, y los estudiantes fueron invitados a auto-organizarse en grupos (que resultaron más grandes o mas chicos) usando otras plataformas, como whatsapp, discord o telegram.

De los encuestados, 134 (72 %) respondieron que Slack era su medio de comunicación más frecuente, mientras que 42 (23 %) la ubicaron en segundo lugar. Este detalle se puede observar en la Tabla 1.

Por otro lado, al ser consultados sobre la frecuencia de acceso, se obtuvo que el 30 % de los encuestados ingresaba diariamente, y el 60 % ingresaba entre 1 y 3 veces por semana. A su vez, se consultó sobre *cuán útil* les había resultado

Red	Principal	Secundaria	Terciaria
Slack oficial	134	42	10
WhatsApp chico	27	43	12
WhatsApp grande	21	39	19
Otro	1	2	8

Tabla 1: Cantidad de elecciones sobre cada medio de comunicación y su relevancia.

esta herramienta cuando fue consultada. Y se observa que el 65 % opinó que en muchas ocasiones les había resultado útil. Al cruzar estas dos variables, se observa que hay una alta correlación positiva entre la valoración de la herramienta y la frecuencia de acceso a la misma: a mayor frecuencia mejor percepción (ver detalle en Figura 1). Cabe destacar en este punto que estimamos una cantidad promedio de 650 mensajes públicos en Slack por semana. En la Figura 2 se puede observar el número de miembros activos (que entraron a la plataforma) y el número de miembros que además postearon algún mensaje.

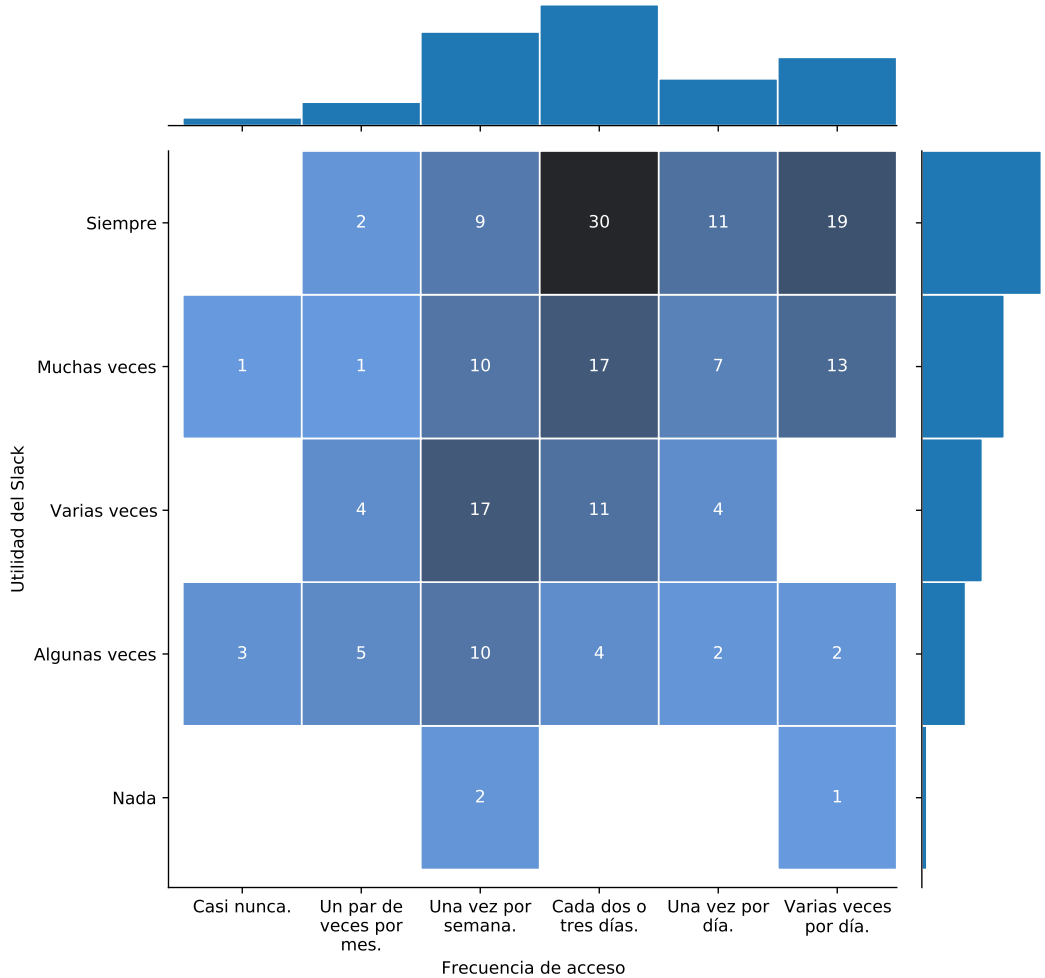

Figura 1: Nivel de utilidad percibido de la participación en el espacio de Slack en función de la frecuencia de acceso. Existe una alta correlación entre la frecuencia y la utilidad.

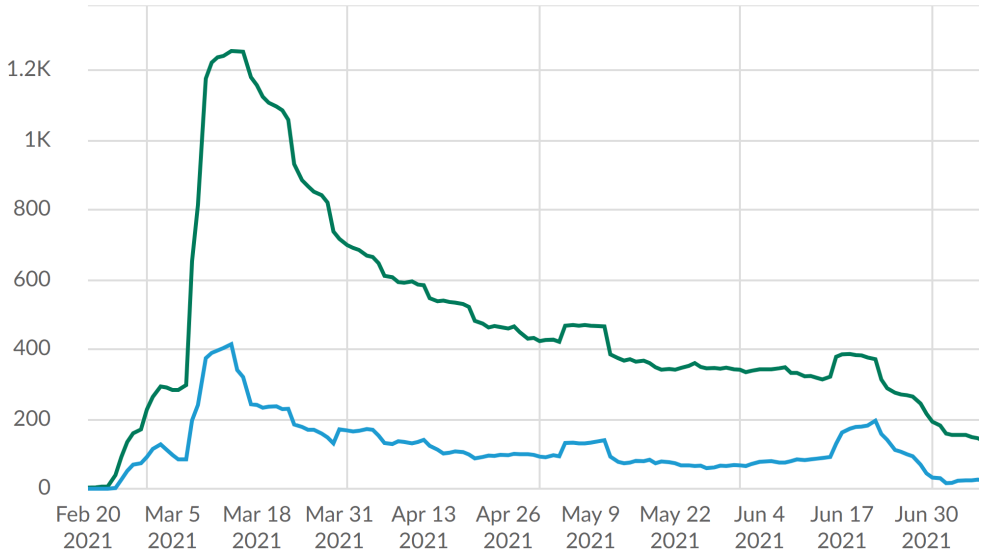

Figura 2: Cantidad de usuarios activos durante el curso (verde) y cantidad de usuarios que escribieron algún post (celeste). El curso oficialmente arrancó el 10 de marzo y terminó el 23 de junio, pero tuvieron acceso desde unas semanas antes para ir instalando el software necesario y semanas después por temas burocráticos (entregas atrasadas, certificados, etc.).

Sobre la forma de participación en las redes, consideramos tres categorías: (1) Leer preguntas y respuesta ajenas (participación pasiva); (2) Escribir consultas propias (participación media); (3) Responder preguntas de otros (participación alta). Estas 3 categorías se ven reflejadas en la Figura 3.

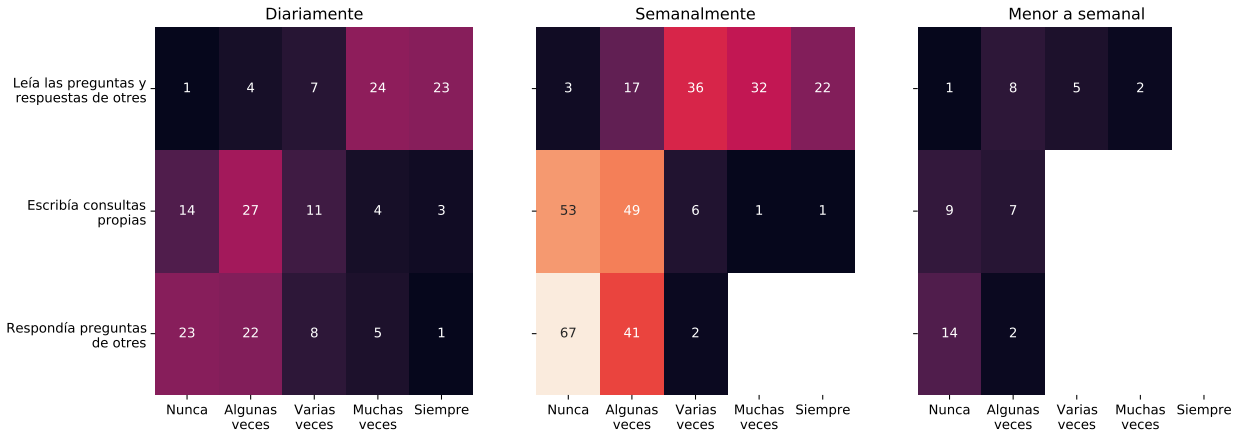

Figura 3: Participación en las redes: frecuencia y tipos de interacción

Entre quienes ingresaban al menos una vez por día ("diariamente"), el $40\,\%$ nunca respondía consultas, mientras que entre quienes ingresaban al menos una vez por semana ("semanalmente") este porcentaje alcanzaba el $60\,\%$. Entre quienes ingresaban diariamente, más del $70\,\%$ escribía consultas, y entre quienes ingresaban semanalmente este porcentaje era del $50\,\%$. Por otra parte, se observa que quienes ingresaban con una frecuencia menor a semanal tenían participación pasiva.

En cuanto al uso de otras redes por fuera del canal oficial, podemos observar 2 categorías de grupos: grupos

Integrantes	Grupo Chico	Grupo Grande
2 a 10	53	0
11 a 20	3	2
21 a 40	6	5
41 a 100	0	8
Más de 100	0	56

Tabla 2: Cantidad de estudiantes que participaron en grupos chicos y grandes de WhatsApp (ó Telegram) y su tamaño.

chicos y grupos grandes (ver Tabla 2). La cantidad de miembros que integran cada uno de estos grupos es variable (y también lo es la percepción de qué es un grupo grande y qué uno chico).

El tamaño de estos grupos condiciona mucho la forma de participar, algunos grupos pueden llegar a ser más de distensión (ver Figura 4) y otros de intercambio más específico.

Figura 4: Captura de pantalla de un mensaje distribuido por un grupo masivo no oficial en respuesta a un pedido de pista sobre un ejercicio que pide calcular el valor de una posición en el Triangulo de Pascal (Pascal, 1654)

El uso que un mismo estudiante daba a distintos grupos se describe en la Figura 5. Los histogramas marginales reflejan que la participación mayoritaria en grupos chicos era de un perfil más activo (en los términos de la clasificación propuesta anteriormente), mientras que para grupos grandes la preponderancia es un perfil más pasivo. Entre quienes participaban en grupos chicos, el 70 % tenía una participación alta, mientras que en el caso de los grandes más del 50 % de los miembros tenía una participación pasiva. A su vez, el 60 % de los encuestados manifestaron participación en ambos tipos de grupos (grandes y chicos), y en estos casos su participación era más activa en los grupos chicos.

> "En mi caso particular, cursé con una amiga. Las pocas veces que no pude resolver por mi cuenta (slack, google, stackoverflow, python docs, etc) lo consulté con ella."

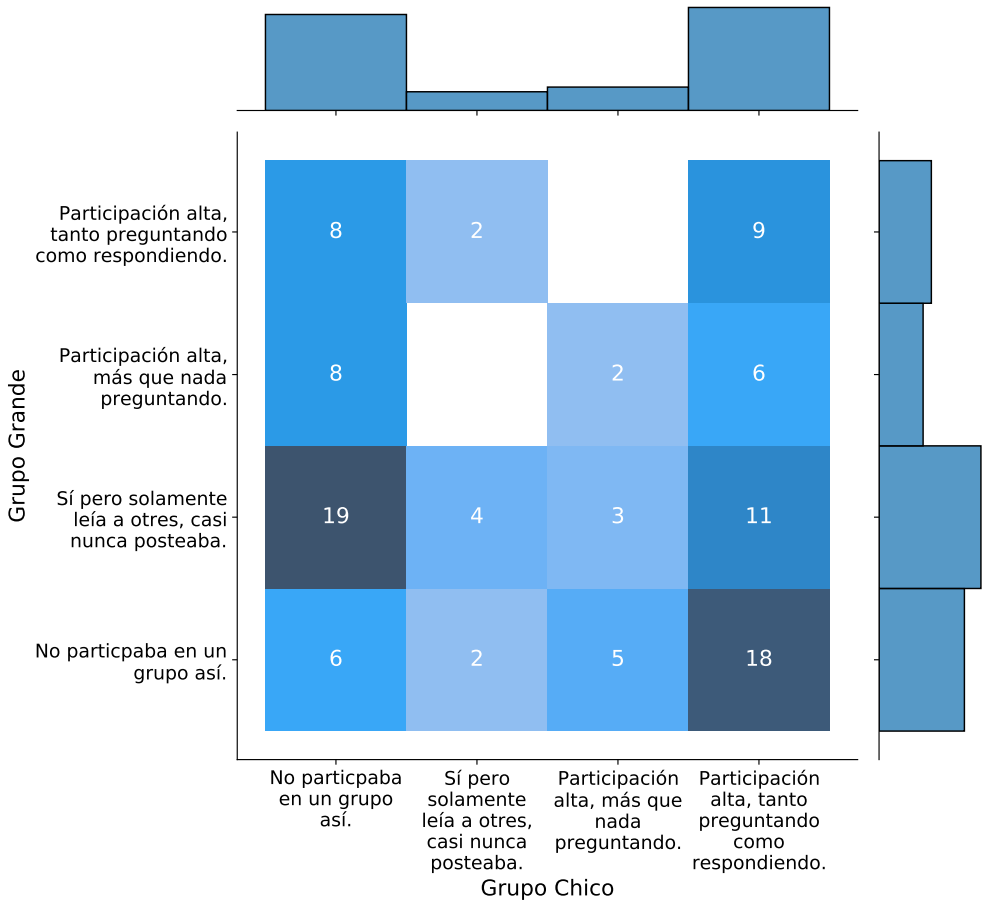

Figura 5: Relación entre el tipo de uso que se le daba al grupo dependiendo el tamaño

Para concluir esta sección incluimos algunos comentarios de los campos abiertos de la encuesta.
Sobre grupos integrados por personas con algún vínculo preexistente:

> "No utilicé otra red además de slack ya que no me gustan los grupos de whatsapp con muchas personas, porque muchas veces se pierden los mensajes, se termina hablando de otros temas, etc. Sin embargo, si tuve mi grupo de estudio reducido con compañeros/amigos que me ayudo muchisimo durante la cursada. Personalmente, esto último lo recomiendo un montón."

Sobre otros lugares donde se realizaron consultas más lejanos aún que los grupos de estudiantes:

> "(...) El grueso de mis consultas se las hacía a unos amigos de world of warcraft (juego online) con quienes comparto ese espacio desde hace años y son todos ingenieros en sistemas que trabajan en programación. Así que mi experiencia más que nada fue esa, la de leer los comentarios en el slack y telegram y consultar el grueso de las cosas a mis compañeros de juego."

Sobre experiencias de estudiantes que no tenían conocidos:

> "Ayuda muchos los grupos de slack masivos ya que de no conocer gente, más aún en virtualidad, uno puede estar contenido."

También se detectaron algunos comportamientos no deseables o negativos:

> "La verdad que yo las redes las aproveche más para mirar que para preguntar. Cuando ingrese a telegram quise preguntar algunas cosas pero me mandaron a leer directamente la teoría del curso (ya que yo no tenía conocimientos de programación) entonces me inhibí por sentirme muy torpe, y ya no participé mucho más en las redes. Así que me dediqué a leer mensajes."

4. Discusión y conclusiones

En este trabajo presentamos un análisis de las redes utilizadas como medio de comunicación en un curso masivo de naturaleza asincrónica dictado en la Universidad Nacional de San Martín.

Hemos observado, entre quienes respondieron la encuesta, que la participación más frecuente muestra una correlación con la percepción de utilidad. En otras palabras estamos viendo que a mayor frecuencia mayor involucramiento y esto podría traer aparejado un mayor aprendizaje (Lopez-Rosenfeld, 2017). Serán necesarios nuevos estudios para poder validar esta hipótesis.

Además, se observó que la existencia de grupos satélites ha sido de gran ayuda para los estudiantes, pero también funcionaron como *buffer* al reducir el volumen de consultas que llegaba al plantel docente. Si bien no tenemos una cuantificación de esta reducción, la estrategia de: 1) intentar resolver el problema con la documentación del curso o buscando en la web, 2) consultar con el grupo cercano, 3) consultar a los docentes, fue reportada por fuera de la encuesta analizada en este trabajo como la estrategia utilizada. Los resultados obtenidos del análisis de los datos son compatibles con la existencia y uso de esta estrategia.

El tamaño y composición de los grupos es un campo que debe ser indagado aún más a futuro. ¿Existe un número mágico de integrantes? ¿Se pueden formar al azar? ¿Es conveniente que exista un vínculo previo entre los integrantes? ¿Pueden ser heterogéneos o es mejor basarlos en afinidades? Si fuera por afinidades, ¿conviene que estas sean geográficas, etáreas, de sincronía, de dedicación, disciplinares?

Nuevos términos se han adoptado en esta nueva realidad que vivimos, en donde se asemeja "el chat de Zoom" con el murmullo de un aula, las salas de Discord con los pasillos de una institución educativa en tiempos pre-pandémicos. Estas analogías están siendo estudiadas y se verán a la luz en los próximos tiempos varios trabajos aportando en esta dirección.

Cabe destacar que esta encuesta no fue respondida por todos los estudiantes, sino únicamente por una fracción de los que terminaron el curso. Por lo que no es posible generalizar desde los datos cómo fue la utilidad de los grupos para el resto (no solo quienes no contestaron, sino quienes no terminaron el curso quizás por falta de un grupo de acompañamiento cercano).

Quedan numerosos interrogantes para estudiar a futuro: ¿la pertenencia a un grupo chico nivela a sus integrantes? ¿Los potencia? ¿O juega en contra? ¿Es posible encontrar alguna relación entre la participación en grupos y las notas obtenidas o la producción durante el curso?

Una última reflexión sobre la comunicación en la virtualidad: el contexto de conexión y masividad y el consiguiente anonimato ha dado espacio a algunos comportamientos no deseados, tales como estudiantes que agreden, desvirtúan el curso y generan un clima hostil que puede influir en la continuidad de otros estudiantes. Por eso, es fundamental fomentar el respeto y el cuidado de los espacios para que puedan ser aprovechados por todos y para todos.

Bibliografía

Barnes, S. B. (2012). Socializing the classroom: Social networks and online learning. Lexington Books.

Casey, G., y Evans, T. (2011). Designing for learning: Online social networks as a classroom environment. International Review of Research in Open and Distributed Learning , 12 (7), 1–26.

Cerdeiro, M., Crespo, J., Grimson, R., Filevich, O., y López-Rosenfeld, M. (2021). Escalando la enseñanza de programación en tiempos de pandemia: desafíos y oportunidades. Cartografías del Sur Revista de Ciencias Artes y Tecnología(13).

Community, E. B. (2020, febrero). Jupyter book. Zenodo. Descargado de https://doi.org/10.5281/zenodo.4539666

Ghounane, N. (2020). Moodle or social networks: What alternative refuge is appropriate to algerian efl students to learn during covid-19 pandemic. Arab World English Journal, 11(3), 21–41.

Greenhow, C. (2011). Online social networks and learning. On the horizon.

Hunter, J. D. (2007). Matplotlib: A 2d graphics environment. Computing in Science & Engineering, 9(3), 90–95. doi: 10.1109/MC-SE.2007.55

Lego Muñoz, C., y Towner, T. L. (2010). Social networks: Facebook's role in the advertising classroom. Journal of Advertising Education, 14(1), 20–27.

Lopez-Rosenfeld, M. (2017). "tell me and i forget, teach me and i may remember, involve me and i learn": changing the approach of teaching computer organization. En 2017 ieee/acm 1st international workshop on software engineering curricula for millennials (secm) (pp. 68–71).

López-Rosenfeld, M., Mocskos, E., Lebrero, M. G., Crespo, J., Arrar, M., Caridi, I., y Sued, M. (2021). Exactas programa: llevando la programación a cada rincón de la ciencia. Electronic Journal of SADIO (EJS), 20(1), 56–76.

Mattermost Inc. (2015). Mattermost. Descargado 2021-08-10, de https://mattermost.com/

Pascal, B. (1654). Triangulo de pascal. Descargado 2021-08-10, de https://es.wikipedia.org/wiki/Triangulo_de_Pascal

Ramírez, J. C. N. (2021). Methodology of learning combined: The use of the social networks in the classroom. En International conference on knowledge management in organizations (pp. 67–71).

Slack Technologies. (2009). Slack. Descargado 2021-08-10, de https://slack.com/

Van Rossum, G., y Drake, F. L. (2009). Python 3 reference manual. Scotts Valley, CA: CreateSpace.

Waskom, M. L. (2021). seaborn: statistical data visualization. Journal of Open Source Software, 6(60), 3021. Descargado de https://doi.org/10.21105/joss.03021

Wes McKinney. (2010). Data Structures for Statistical Computing in Python. En Stéfan van der Walt y Jarrod Millman (Eds.), Proceedings of the 9th Python in Science Conference (p. 56 - 61). doi: 10.25080/Majora-92bf1922-00a

La escuela y la brecha de género en la enseñanza de las Ciencias de la Computación

M. Emilia Echeveste
meecheveste@gmail.com
Universidad Nacional de Córdoba

Marcos J. Gómez
marcos.gomez@unc.edu.ar
Universidad Nacional de Córdoba

Cecilia Martínez
cecimart@gmail.com
Universidad Nacional de Córdoba

Luciana Benotti
luciana.benotti@unc.edu.ar
Universidad Nacional de Córdoba

Resumen

Cerrar la brecha de género en computación es un gran desafío a nivel mundial. Teniendo en cuenta los reportes de los 38 países que integran la Organización para la Cooperación y el Desarrollo Económicos, las mujeres obtienen mayor educación superior que los hombres, sin embargo, en 2016 sólo un 28 % de estudiantes universitarios en Ciencia, Tecnología, Ingeniería y Matemáticas (STEM) son mujeres. En 2010, Argentina reportaba sólo un 18 % de mujeres en el estudiantado en informática.

En este artículo presentamos experiencias de enseñanza de Ciencias de la Computación (CC) en donde analizamos cómo se construye la disposición de las mujeres a los saberes de CC en contextos educativos reales. Las preguntas que guían nuestro trabajo son: ¿Qué lugar ocupan las experiencias escolares y cómo se vinculan con la brecha de género? ¿Cuál es el rol de la escuela en la democratización de los saberes en CC? Para ello recuperamos las indagaciones que forman parte de nuestros trabajos de investigación en diferentes niveles educativos. A través de estudio multi-caso se establecieron similitudes, diferencias y continuidades entre los casos generando evidencia de un mismo fenómeno en diferentes contextos.

Palabras clave: Brecha de Género, Ciencias de la Computación, Escuela primaria, Escuela Técnica, Universidad.

1. Introducción

Experiencias en nivel primario muestran que la enseñanza de computación a las niñas permite achicar las diferencias socio culturales de origen respecto al acercamiento de conceptos de computación. En la escuela donde se enseña computación las estudiantes tuvieron un rendimiento similar o superior a los varones. En cambio, en la escuela en donde no se enseñaba computación el rendimiento académico en computación de los varones superaron estadísticamente a las mujeres.

La escuela secundaria no ofrece en su currícula saberes de CC de manera obligatoria en todas las instituciones sino en aquellas que cuenten con esta orientación específica. La matrícula de varones en secundarias con orientación

en informática triplica a la de mujeres, por tanto sólo aquellas que eligen orientaciones en informática acceden a estos conocimientos. Además de la escasa matrícula femenina en las escuelas con orientación, encontramos que en el aula la oferta de enseñanza varía según el género, ofreciendo diferentes tareas según sean varones o mujeres.

Las pocas estudiantes universitarias que eligen estudiar computación afirman que la escuela secundaria no les ha ofrecido conocimientos que permitan comprender la disciplina. En contraste, los varones manifiestan que el secundario contribuyó a generar interés y que pueden recuperar esos saberes específicos.

La matrícula femenina en carreras de computación no aumentó proporcionalmente en los últimos 10 años aunque sí aumentó la matrícula global. La brecha de género aún sigue siendo desfavorable para las mujeres. La escasez de saberes de computación recibidos previamente es consistente con los procesos analizados en los casos de la escuela primaria y secundaria donde la oferta de contenidos de computación es muy limitada y segmentada por género.

2. Trabajo previo

Como menciona Morgade (2009) en todos los procesos educativos se producen, transmiten y negocian sentidos y saberes respecto de la sexualidad y las relaciones de género. Investigaciones sobre género y CC (Redmond et al., 2013) mencionan que la menor exposición de las mujeres al uso de las computadoras genera menor seguridad en su vínculo con las CC, en especial en culturas en donde el acceso a juguetes y videojuegos están atravesados por prejuicios culturales de género. Según el informe de 2013 sobre mujeres e informática realizado por la Fundación Sadosky, el uso de videojuegos al que denominan complejos[1] es más frecuente en el género masculino. Este informe considera que esta diferencia podría ser uno de los elementos para comprender cómo se produce un vínculo más intenso con la CC y la programación entre los varones que entre las mujeres (Zukerfeld, 2013).

Un equipo de investigadores argentinos liderado por Benitez Larghi (2011) ha realizado reconocidas investigaciones sobre brechas digitales y apropiación desigual. Los y las autoras mencionan que la menor intensidad de uso por parte de las mujeres podría vincularse con una mayor exigencia de ella en lo que concierne a tareas domésticas. Estos primeros acercamientos desiguales van forjando diferentes capitales digitales con los que llegan los y las jóvenes a las escuelas. La investigación realizada por Sanders (2005) en 21 países de América, Europa y Medio Oriente encuentra que las diferencias de género en las actitudes y el comportamiento son relativamente pequeñas a edades más tempranas pero aumentan a medida que los y las estudiantes crecen. Más aún, en muchos contextos, pareciera que las niñas pierden interés en las materias STEM con la edad y en mayor proporción que los niños. Un estudio en el Reino Unido concluyó que a la edad de 10 a 11 años, los niños y las niñas tenían casi el mismo compromiso con STEM en donde un 75 % de niños y 72 % de niñas señalaron que aprendieron cosas interesantes en ciencias. A la edad de 18, esta proporción cayó a 33 % para los varones y 19 % para las niñas, según se midió la participación en los estudios superiores en STEM. Aquí, los niños comenzaron a abandonar las asignaturas STEM a medida que se aproximaban a sus estudios superiores, mientras que las niñas decidieron abandonar mucho antes, en la escuela secundaria.

Las recientes investigaciones (Barker y Aspray, 2006) presentan a la escuela como uno de los factores que interviene como barrera entre el vínculo de las mujeres con las ciencias de la computación. En donde los planes

[1]Refiriéndose a juegos, tales como Counter-Strike, World of Warcraft, God of War, FIFA, PES, etc., que implican la movilización de habilidades de concentración, organización, configuración, análisis, etc. y que podrían constituir bases propicias sobre las cuales erigir algunas habilidades informáticas ulteriores.

de estudios sin conexión relevante, las prácticas de enseñanza que desalientan la colaboración, los estereotipos y los entornos de aprendizaje incómodos desalientan la participación de las niñas, desvaneciendo su interés.

Si tomamos en cuenta los postulados de Vygotsky (1984), los aprendizajes de niños y niñas no se originan en la educación formal sino que tienen un desarrollo preescolar, no sistematizado en el que puede coincidir o no con el que se enseñará una vez que ingresen a la educación formal y obligatoria. Esto significa que los niños y las niñas ingresan a la escuela con un bagaje de saberes previos, que vinculados a los estereotipos de género permite ubicar a la escuela como un espacio que posibilite romper con esta segregación de género o continuar con su reproducción. En este artículo, recuperamos diversas experiencias educativas realizadas en Córdoba que permitirán aportar datos para seguir debatiendo esta situación con el objetivo de construir experiencias educativas que sean más inclusivas.

3. Métodos

Una alternativa para comprender un fenómeno socio-educativo como lo es la brecha de género es un estudio multi-caso que permite establecer similitudes, diferencias y continuidades entre los casos. El valor de este tipo de investigaciones radica en que los datos son interpretados en contextos particulares, pero al mismo tiempo permite analizar y proveer evidencia de un mismo fenómeno en diferentes contextos (Gustaffson, 2017).

Para analizar el rol de las instituciones educativas en la construcción de la brecha de género se seleccionaron tres casos correspondientes a tres niveles del sistema educativo: primaria, secundaria y universitario. De esta manera pudimos observar continuidades de un tema emergente y su despliegue en cada nivel. Los estudios de casos fueron llevados a cabo por diferentes integrantes de un mismo equipo de investigación. La recolección de datos incluyó datos cuantitativos -tales como encuesta y pre y post test- y cualitativos como entrevistas en profundidad, observaciones de clases y revisión documental. El Cuadro 1 resume los participantes, herramientas de recolección de datos y análisis empleados en cada caso.

Caso	Participantes	Recolección de datos	Análisis
Cursado de un módulo de informática en 2 escuelas primarias	86 estudiantes de 10 y 11 años	Estudio exploratorio de cursos de 5hs en cada escuela. Examen múltiple opción analizando conceptos fundamentales	Estadísticas descriptivas
Cursado de materias de programación en 3 Escuelas Técnicas secundarias	38 estudiantes y 9 docentes	Estudio etnográfico 18 Entrevistas, 27 observaciones, 5 grupos focales y revisión documental	Análisis de temas emergentes con triangulación de información
Cursado de introducción a los algoritmos en la carrera de Ciencias de la Computación	180 estudiantes encuestados y 9 estudiantes entrevistados	Estudio exploratorio. Entrevistas y encuestas	Estadísticas descriptivas, análisis de temas emergentes

Cuadro 1: Muestra, recolección de datos y análisis en cada caso.

Para el análisis de los casos se usaron aportes de los enfoques etnográficos y exploratorios. Los casos eran conocidos en profundidad por las y los investigadores por haber hecho estudios previos sobre ellos. Esto permitió interpretar los datos en un devenir histórico e institucional. A lo largo de la descripción y análisis de cada caso se mostrará cómo se construyeron los resultados a partir de los datos recogidos y analizados.

3.1. Caso 1: Experiencias de nivel primario

Martinez et al. (2015) implementaron un estudio exploratorio para comparar cómo los niños de primaria (de 8 a 11 años) incorporan conceptos fundamentales de programación. Al analizar los datos recolectados observaron que no hay diferencia en el desempeño entre los y las estudiantes. A partir de este hallazgo nos preguntamos sobre el rol de la escuela en los aprendizajes de CC, buscando indagar en el desempeño estudiantil. Para indagar sobre el rol de la escuela en la brecha de género diseñamos un estudio en dos escuelas del mismo barrio, con una población socioeconómica equivalente, pero donde en una se enseña CC, y en la otra no. Realizamos la misma intervención en ambas escuelas y luego comparamos los resultados.

Las intervenciones fueron realizadas en dos escuelas primarias públicas de gestión privada, referenciadas como EpriConExp y EpriSinExp. No hay estudiantes debajo de la línea de pobreza y la mayoría forma parte de familias pertenecientes a la clase media. En ambas experiencias enseñamos conceptos fundamentales de programación en Gobstones[2] utilizando Mumuki[3]. Gobstones es un lenguaje pensado y diseñado para la enseñanza introductoria a la programación (Martinez López et al., 2017) . Mumuki es un entorno web para la enseñanza de programación open source que brinda a los usuarios feedback formativo –del inglés formative feedback– (Benotti et al., 2018). Ambos experimentos se llevaron a cabo por el mismo docente, en las dos instituciones durante el horario escolar, con estudiantes de 10 y 11 años de edad. La experiencia completa duró 5 horas en cada escuela. Se pidió al grupo de estudiantes que completaran 23 ejercicios de programación. A través de estos ejercicios, se introdujo a los y las estudiantes a conceptos de programación como secuencia, condicionales, ciclos y uso de argumentos.

En la EpriConExp, el grupo de estudiantes tenían experiencia previa en programación con el lenguajes de bloques Scratch programando videojuegos y animaciones. El estudiantado tuvo 1 hora de programación por semana durante 3 semestres antes de participar de la intervención que presentamos a continuación. La enseñanza de la programación se llevó a cabo en base a proyectos y utilizando la resolución de problemas y la exploración como ejes pedagógicos. En la otra institución, EpriSinExp, el grupo de estudiantes no tenía experiencia previa en programación. El equipo directivo y docente de la institución informaron que la programación no formaba parte de su plan de estudios. Los y las estudiantes confirmaron que no habían participado de experiencias relacionadas a la programación fuera del contexto escolar.

Para poder medir el impacto de la intervención implementamos un examen de múltiple opción. En total 129 estudiantes participaron de las dos experiencias. 63 estudiantes de la EpriConExo y 66 de EpriSinExp. Una vez finalizada la experiencias, los estudiantes, resolvieron el examen el cual no era obligatorio.

El examen estaba compuesto por 6 preguntas múltiple opción. Cada pregunta incluía un programa escrito en Gobstones y un tablero inicial. El estudiantado debía, en base al código y el tablero inicial, elegir uno de cuatro posibles tableros finales, que fuese producto de ejecutar el programa sobre el tablero inicial. El examen fue diseñado para incluir

[2] https://gobstones.github.io/

[3] https://mumuki.io/home/

respuestas que contengan errores conceptuales comunes de programación (Weintrop y Wilensky, 2015). Todas las preguntas tuvieron el mismo peso en la puntuación final. La escala de puntaje del examen fue de 0 a 10. En el Cuadro 2 se informa la cantidad de estudiantes por escuela y género que participaron en la experiencia y que decidieron resolver el examen de múltiple opción.

Escuela	EpriConExp			EpriSinExp		
Género (F/M) y Total	F	M	T	F	M	T
Total de estudiantes participantes	30	33	63	37	29	66
Total de estudiantes evaluados	26	31	57	18	11	29

Cuadro 2: Cantidad de estudiantes por escuela y por género participantes

En cuanto a la brecha de género, en la Figura 1 podemos observar que en la escuela donde los y las estudiantes no tenían conocimientos previos en programación, los estudiantes se desempeñan mejor que las estudiantes. La línea continua representa la nota promedio de los estudiantes, mientras que la línea con trazos representa la nota promedio de las estudiantes. La nota promedio de los estudiantes supera en más de dos puntos a las notas de las estudiantes. En la Figura 1 podemos observar el caso de la escuela cuyos estudiantes tuvieron tres semestres previos de programación utilizando el entorno con lenguaje basado en bloques Scratch. La diferencia entre el desempeño de los estudiantes y las estudiantes no es estadísticamente significativo.

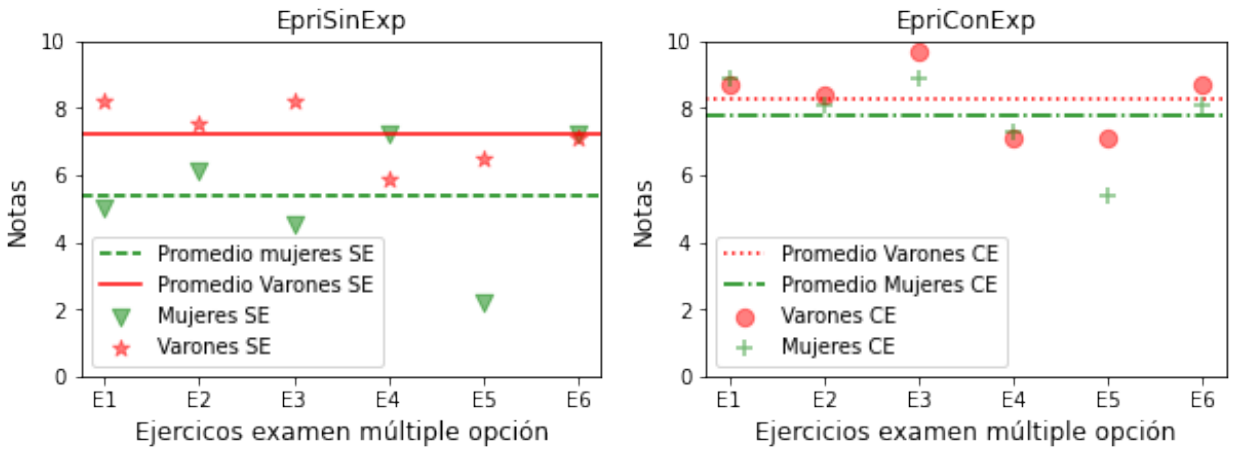

Figura 1: Desempeño de los y las estudiantes en el examen múltiple opción por escuela organizados por escuela. En EpriSinExp se presentan los resultados de la escuela sin experiencia en programación. En EpriConExp se presentan los resultados de la escuela con experiencia en programación.

Estos resultados muestran que en la escuela en donde se ofrecen saberes de computación durante la primaria, el rendimiento de las mujeres no es tan diferente al de los varones. En cambio, en la escuela donde no se ofrecen estos contenidos, se identifica una brecha que bien podría ser de origen respecto al género. Ante estos datos elaboramos una posible hipótesis sobre el potencial que tiene la escuela a partir de ofrecer saberes que contribuyan a achicar las brechas de origen socio-cultural. Tal como veremos más adelante, el dominio de saberes está relacionado positivamente con la autopercepción de eficacia que construyen los y las estudiantes. Esto es, tener dominio de ciertos contenidos ofrece la confianza necesaria para aprender y resolver desafíos cognitivos. De ahí la relevancia también de poder garantizar que

todas las chicas puedan acceder a estos contenidos.

3.2. Caso 2: Experiencias de nivel secundario

Esta experiencia se enmarca en una tesis doctoral que toma como muestra las únicas tres escuelas de la ciudad de Córdoba Capital que ofrecen la orientación programación como formación secundaria técnica. Durante 2016 y 2017 se realizó el seguimiento a un mismo grupo de estudiantes durante los dos últimos años de su escolaridad. Se realizaron 27 observaciones de clases en las materias Programación III, Aplicación de Nuevas Tecnologías (ANT) y Formación, Ambiente y Trabajo (FAT), 5 grupos de discusión y 18 entrevistas en profundidad a estudiantes. Esta investigación, cualitativa y de corte etnográfico, buscó conocer la relación con el conocimiento que establecen los y las jóvenes con la programación, lo que permitió recuperar como se apropian de estos conocimientos, en donde se halló como emergente una distinción de género en las actividades asignadas en las materias de programación. Históricamente el sistema educativo presentó escuelas que admitían oficialmente solo matrículas femeninas o masculinas. Las Escuelas Técnicas han presentado reportes en sus matrículas que evidencian una clasificación de género, que si bien no es oficial, sí presenta una distinción según la especificidad, en donde Óptica o Alimentos se liga a formación de mujeres mientras que electrónica o mecánica se vincula a los varones. El último censo de Escuelas Técnicas de la provincia de Córdoba presenta que 6 de cada 10 alumnos (63,8 %) que se matriculan en el área de la informática son varones. Más específicamente en las Escuelas Técnicas en programación el 64,5 % corresponde a matrícula masculina y un 35,5 % responde a mujeres (Ministerio de Educación de la Provincia de Córdoba, 2018). Estos datos contrastan con los egresados del secundario regular con orientación en informática- pero no técnico- donde egresan varones y mujeres en la misma cantidad.

De las tres instituciones analizadas, sólo una contaba con matrícula femenina. En la escuela privada (EPr.) asistieron 19 estudiantes varones ya que esa institución técnica religiosa no admitía mujeres. En la Escuela Pública N1 (EPuN1) asistieron 14 estudiantes, 5 mujeres y 9 varones, mientras que en la Escuela Pública N2 (EPuN2) eran solo 5 estudiantes masculinos. Los siguientes emergentes surgen de la única escuela con matrícula mixta. A diferencia de las escuelas primarias en donde una ofrecía saberes de programación y la otra no, en estos tres casos las escuelas tienen una orientación específica en programación, es decir que los contenidos aparecen en la formación oficial para todos y todas las estudiantes. Sin embargo, a partir de las observaciones de clases pudimos reconstruir que la oferta no llegaba a todos y todas por igual. Por un lado, la matrícula femenina era escasa, nula o prohibida para mujeres. Por el otro, la oferta al interior de la escuela donde había mujeres era diferente según el género y capital digital de origen. Los profesores dividían las tareas de los proyectos de programación según los saberes específicos que requería la resolución de cada tarea. A los varones con mayor capital digital de origen se les asignaba tareas de programación más complejas mientras que a las mujeres se les delegaba tareas "decorativas" vinculadas a las manualidades o de menor demanda cognitiva.

A continuación se exponen y detallan los emergentes encontrados. Diferencia de actividades entre hombres y mujeres justificado por una falta de interés de las estudiantes o consignaciones de que les cuesta más o son más vagas. En los proyectos de desarrollo de software las estudiantes no lideran las actividades. Se las "invita" a participar pero con actividades anexas o secundarias. Se las vincula con actividades "femeninas" por la delicadeza y la prolijidad asignadas al género como lo fue pintar una maqueta que luego sus compañeros varones automatizaron. Esta diferencia de saberes que mencionan tanto los y las estudiantes como sus docentes se vincula a un mayor dominio de los varones reforzado por experiencias previas autodidactas o de formación extraescolar. Se observó que estudiantes varones demandaban

a sus instituciones escolares conocimientos actualizados, relevantes y con una revisión de las participaciones en los proyectos ya que aquellos que se encargaban del desarrollo de software eran quienes ya tenían un alto conocimiento previo en la disciplina. Esto es reforzado en el caso del género, sin embargo aparece como naturalizado, al considerar que no realizaban una diferencia entre varones y mujeres. Y si esto existía era por responsabilidad de ellas de no querer participar.

Se puede reconocer en el discurso de una de las estudiante que si se les explicara con mayor paciencia seguramente ellas podrían aprender más. De esta manera, las estudiantes muestran una confianza y autopercepción de eficacia que es menor a la de los estudiantes varones, considerando que *‘‘me altero porque no me sale, entonces que se encarguen los chicos que son los que más saben”*. Esto se refuerza con discursos de sus compañeros varones quienes mencionan que *“al grupito de las chicas no les interesa mucho”* o *“las chicas no le dan importancia”*. Esto se intensifica con una propuesta educativa que por más que sea oficial y con posibilidad de acceso para toda la población estudiantil, presenta gran divergencias entre las actividades y las oportunidad ofrecidas a mujer y varones.

Las estudiantes de esta institución no se proyectan trabajando en la programación por más que sean formadas en una escuela de oficio ya que consideran no ser buena en estas tareas. No logran encontrar vínculos con el conocimiento que les permita ser parte de este oficio ni poder integrarse a una comunidad de trabajadoras en el área.

3.3. Caso 3: Experiencias de nivel universitario

Este caso analiza los procesos de apropiación de saberes de computación entre estudiantes de la materia “Introducción a los Algoritmos” (IA) dictada en el primer año del Plan de Estudios de la Licenciatura en Ciencias de la Computación en una Universidad Nacional. Se trata de una carrera que tiene altos niveles de deserción, particularmente durante sus primeros años, con un 19 % de ingresantes mujeres.

Se ha documentado que las primeras experiencias de aprendizaje de algoritmos y de programación suelen ser difíciles para el estudiantado (Guzdial, 2015). Por ello el caso analizado se focaliza en IA al ser la primera materia específica de programación en el Plan de Estudios. De un total de 180 estudiantes, 140 dicen pertenecer al género masculino, 39 al género femenino y 1 a otro. Como parte de este estudio se realizaron 9 entrevistas en profundidad con estudiantes con rendimiento diverso, 3 de ellas eran mujeres. En primer lugar presentamos una clasificación de las respuestas a la pregunta “¿Por qué decidiste estudiar esta carrera?”, luego describimos los principales emergentes de las entrevistas en relación al género.

En el Cuadro 3 clasificamos las respuestas abiertas a la pregunta ¿Por qué elegiste estudiar esta carrera?. Algunas de las razones son similares entre los estudiantes de género femenino y masculino, como aprender a programar, curiosidad, interés en desarrollar aplicaciones útiles. Ellos dicen estar más motivados por la salida laboral y varios mencionan su colegio como el lugar en el que los motivaron. Esto no pasa en el género femenino, no se registraron alusiones a experiencias escolares. Del total de los entrevistados, 5 varones reconocen saberes previos al ingreso universitarios vinculados al área: 4 de ellos hicieron el secundario en escuelas con programación, mientras que otro de sus compañeros menciona que eligió la carrera por su afición a los videojuegos. Del resto, un estudiante varón y las tres mujeres no reconocen vínculos anteriores con la disciplina. En las entrevistas los estudiantes que recibieron saberes de informática en las escuelas comentaron que logran recuperar esos conocimientos para aprender los contenidos de IA. Por ejemplo Cesar menciona que de informática pudo recuperar *“saber qué es un bucle, un condicional, una estructura...tipos de datos, condicionales, bucles y arreglos”*. En contraste, las tres mujeres entrevistadas que no tuvieron experiencias previas con informática en la escuela, mencionaron que hubiera sido importante recibir estos

conocimientos. Inés comentó *"en el colegio es como que ni te imaginas lo que puede llegar a hacer una computadora"*. Para Inés el primer año fue frustrante porque había sido abanderada en su escuela y *"pensé que sabía"*, pero durante el ingreso notó que "no entendía" porque no habían visto derivadas ni integrales. Este mismo sentimiento de frustración lo sintió Ramona, otra estudiante que tampoco había recibido saberes previos en informática en su escuela, *"Si, te daba un poco de frustración en parte decir: Pucha, no entiendo nada, ¿Por qué mis compañeros entienden y yo no? ¿Por qué avanzan? Yo no entiendo..."*.

Los estudiantes varones que recibieron formación en el secundario manifiestan que pudieron desarrollar su interés y ligar esos saberes con IA. Si bien la ausencia de contenidos de informática en la escuela afecta a ambos, mujeres y varones, sabemos que un menor número de mujeres egresan de secundarias que han tenido informática, por tanto menos mujeres tienen posibilidades de generar interés y confianza con saberes de informática.

Con respecto a la salida laboral como motivación para elegir la carrera, ellas mencionan la flexibilidad en horarios de la profesión, ellos no. Ellas mencionan mucho más interés personal, desafío, dicen que es difícil pero que quieren intentar y también algunas mencionan apoyo de la familia. Ellos mencionan una mayor cantidad de temas técnicos como programación, inteligencia artificial, videojuegos, machine learning, investigación científica, y hasta ciberseguridad y blockchain. Para las dos mujeres el interés por la carrera está ligado a las imágenes de sus madres. Una por su relación con la enseñanza de la matemática y la otra por su particular vínculo y perspectiva con los lenguajes al ser su madre sordomuda.

	Femenino	Masculino	Otro	Total
Amigos	0	4	0	4
Aplicaciones	6	7	0	13
Cambio desde otra carrera	5	1	0	6
Colegio	0	9	0	9
Computadora	1	11	0	12
Curiosidad	5	4	0	9
Desafío	4	0	0	4
Familia y Flexibilidad	4	0	0	4
Inteligencia artificial	1	8	0	9
Investigación científica	2	15	0	17
Matemática	2	8	0	10
No sabe	1	1	0	2
Programar	5	53	0	58
Salida laboral	3	18	1	22
Total	39	140	1	180

Cuadro 3: Categorización de las respuestas a la pregunta ¿Por qué decidiste estudiar esta carrera? detallada por género

Las estudiantes expresan más cambios desde otras carreras que los estudiantes. Esto se ve reflejado en que el 51 % de las mujeres cursantes tienen 20 años o más, mientras que los hombres que cubren esa franja etaria son sólo el 36 %. La edad promedio de las mujeres ingresantes es 22 mientras que es 20 para los hombres. Hay varias historias de desconocimiento de la carrera y de "cambio de carrera" como esta: *"Toda mi vida me interesó la computación y todo lo relacionado a software, siempre investigué y experimenté, pero desconocía la existencia de la carrera, conocía Ing.*

en Sistemas pero no me atraía el programa ni la orientación (además de tener amigas estudiando en UTN, pasándola mal por el entorno sumamente machista). Al terminar el secundario en 2013 me dediqué a la comunicación visual (Tec. en Fotografía y Diseño Gráfico), hasta que en noviembre de 2020 se presentó la oportunidad de participar en Argentina Programa, donde a través de charlas con profesionales descubrí la Lic. en Computación."

En resumen, las mujeres no mencionan el colegio como motivación para elegir la carrera, pero los varones sí. Ellos discuten temas técnicos en su motivación y ellas no, lo cual puede ser evidencia de formación previa de ellos y no de ellas. Se registró además que la edad de ellas es mayor y hay varias que vienen de otra carrera. Finalmente, la matrícula femenina no aumentó proporcionalmente en los últimos 10 años aunque sí aumentó la cantidad de estudiantes total que estudia estas carreras. Si bien se ha logrado aumentar el número de ingresantes, la brecha de género no ha disminuido.

4. Conclusiones

El análisis transversal de los casos en tres de los niveles educativos ofrece algunos indicios sobre cómo, de manera sistémica, el sistema educativo contribuye a la brecha de género de las mujeres en computación. Los casos del nivel primario muestran que la enseñanza de computación a las niñas permite achicar las diferencias socio culturales de origen respecto al acercamiento de conceptos de computación. En la escuela donde se enseña computación las chicas tuvieron un rendimiento similar o superior a los varones. En cambio, en la escuela en donde no se enseñaba computación los varones superaron estadísticamente a las mujeres.

La escuela secundaria no ofrece estos saberes de manera obligatoria para todas las instituciones, por tanto solo las mujeres que eligen orientaciones en informática acceden a estos conocimientos. Además de la escasa matrícula femenina en las Escuelas Técnicas con orientación, al interior de las clases encontramos que la oferta de enseñanza varía por género, contribuyendo a reproducir las brechas digitales.

En este contexto, las pocas estudiantes universitarias que eligen comenzar una carrera en computación afirman que la escuela secundaria no les ha ofrecido conocimientos que permitan comprender la disciplina. En contraste, los varones egresados de escuelas con orientación manifiesta que el secundario contribuyó a generar interés y que pueden recuperar esos saberes en la primera materia específica.

Es difícil con estos datos establecer una relación entre la baja matrícula femenina en el nivel superior y el vaciamiento de contenidos en la escolaridad obligatoria. Sin embargo, el análisis multicaso permitió encontrar relaciones entre la ausencia de computación en el nivel secundario y la demanda de las jóvenes universitarias que expresamente manifestaron que no haber recibido esos saberes genera desafíos y dificultades en la carrera.

La falta de representación que afecta a las niñas está profundamente enraizada y frena su progreso hacia el desarrollo sostenible. Esto despliega ampliamente la complejidad de la trama que subyace en los procesos de subjetivación implicados en los modos en que los y las jóvenes se relacionan con los conocimientos y han sido posibilitadoras de preguntas que permiten interpelar esta relación de las instituciones escolares y la división de género. La educación primaria cuenta con la potencialidad de lograr corromper este orden preestablecido por género que se profundiza en la educación secundaria, donde se continúa reproduciendo los capitales digitales que se construyen en sus trayectorias escolares y extraescolares. Lo que genera nulas modificaciones en los porcentajes de acceso de las mujeres a las carreras de Computación. Es importante visibilizar estos procesos para lograr producir un quiebre que posibilite sostener e incentivar un interés de las estudiantes que los estudios en el área evidencian perder con el avance de la escolaridad.

Bibliografía

Barker, L. J., y Aspray, W. (2006). The state of research on girls and IT.

Benítez Larghi, S., Aguerre, C., Calamari M., Fontecoba, A., Moguillansky M. y Ponce de León J. (2011). Juventud, sectores populares y TIC en la Argentina. Revista Versión. Estudios de Comunicación, Política y Cultura, 27, pp. 1–20.

Benotti, L., Aloi, F., Bulgarelli, F., y Gómez, M. J. (2018). The Effect of a Web-based Coding Tool with Automatic Feedback on Students' Performance and Perceptions. Proceeding of the 49th ACM Technical Symposium on Computer Science Education., pp. 2–7.

Guzdial, M. (2015). Learner-centered design of computing education: Research on computing for everyone. Synthesis Lectures on Human-Centered Informatics, 8(6), 1–165.

Gustafsson, J. (2017). Single case studies vs. multiple case studies: A comparative study.

Levis, D. (2007). Enseñar y aprender con informática/enseñar y aprender informática. Medios informáticos en la escuela argentina. En Levis, D y Cabello, R (2007) Medios informáticos en la educación a principios del siglo XXI, pp. 21–50. Prometeo Libros Editorial. Argentina.

Martínez, C., Gomez, M. J., y Benotti, L. (2015). A comparison of preschool and elementary school children learning computer science concepts through a multilanguage robot programming platform. In Proceeding of 2015 ACM Conference on Innovation and Technology in Computer Science Education, pp.159–164.

Martínez López, P. E., Ciolek, D., Arévalo, G., y Pari, D. (2017). The GOBSTONES method for teaching computer programming. XXV Simposio de Educación Superior en Computación (SIESC'17), dentro de la XLIII Conferencia Latinoamericana de Informática (CLEI'17), pp.1–9.

Morgade, G. (2005). Lectura de género y procesos educativos. En: Revista Criterio No 2309. Año 78. www.revistacriterio.com.ar

Redmond, K., Evans, S y M. Sahami, M. (2013). A large-scale quantitative study of women in computer science at stanford university. In Proceeding of the 44th ACM Technical Symposium on Computer Science Education, pp 439–444.

Sanders, J. (2005). Gender and technology in education: What the research tells us. In Proceedings of the international symposium on Women and ICT: creating global transformation, pp. 6–es.

Vigotsky, L. S. (1984). Aprendizaje y desarrollo intelectual en la edad escolar. Infancia y Aprendizaje, 17/28, pp. 105–116.

Weintrop, D., y Wilensky, U. (2015). Using commutative assessments to compare conceptual understanding in blocks-based and text-based programs. In ICER, pp. 101–110.

Zukerfeld, M. (2013). ¿Y las mujeres donde están? Estudio sobre representaciones acerca de la informática en escuelas secundarias del conurbano bonaerense. Informe Final. Program.ar, Fundación Sadosky. Buenos Aires

SESIÓN 2:
ENTORNOS Y ENFOQUES

Moderadora: *Lic. Ana María Company (UNNE)*

Arduino en la Escuela: una herramienta versátil para la enseñanza de programación y robótica
Gonzalo Pablo Fernández, María Belén Ticona Oquendo y Christian Cossio-Mercado

Hacia un estado del arte sobre programación tangible
Leandro Castro, Verónica Artola, Silvia Bast y Gustavo Astudillo

CTFs en escuelas: una plataforma para acercar la ciberseguridad a la educación secundaria
Gabriela Suárez, Patricio Bolino, Jeremías Pretto, Paula Venosa y Claudia Queiruga

Arduino en la Escuela: una herramienta versátil para la enseñanza de programación y robótica

Gonzalo Pablo Fernández* gpfernandez@dc.uba.ar UBA

María Belén Ticona Oquendo* mticona@dc.uba.ar UBA

Christian Cossio-Mercado† ccossio@dc.uba.ar CONICET - UBA

Resumen

En los últimos años cobró gran relevancia la enseñanza del Pensamiento Computacional, la Robótica y la Inteligencia Artificial en la educación de nivel primario y secundario. En este sentido, se han desarrollado varios entornos de programación por bloques para *Arduino*, los cuales se diferencian principalmente por el contexto educativo para el cual fueron pensados. En este trabajo presentamos Arduino en la Escuela, un entorno de programación basada en bloques para Arduino, creado para su utilización en ámbitos educativos. Se trata de un proyecto desarrollado de manera iterativa, fruto de la experiencia del uso de la herramienta en diversos espacios educativos: en la enseñanza primaria y secundaria, en formación y capacitación docente, y en divulgación científica en el nivel universitario, entre otros. Se caracteriza por su portabilidad, ya que no requiere conexión a Internet y tiene soporte para distintos sistemas operativos, por su facilidad de instalación y por haber sido desarrollado desde su origen en español. Todas estas características hacen de Arduino en la Escuela una herramienta útil para la enseñanza de programación, por ejemplo, en contextos de baja disponibilidad de recursos.

Palabras clave: Entorno de programación, Enseñanza de Programación, Arduino, Programación por bloques, Ciencias de la Computación.

1. Introducción

Arduino es una familia de placas programables que ha adquirido gran popularidad tanto en entornos profesionales como educativos por ser simple, barata y de hardware abierto (Kushner, 2011). *Arduino en la Escuela* es un entorno de programación con fines educativos, específico para programar placas *Arduino*. Se caracteriza por usar programación por bloques, paradigma que ha demostrado facilitar la tarea de aprender a programar (Weintrop y Wilensky, 2015; Weintrop, 2015). Al abstraer detalles de implementación del mundo físico, es ideal para aprender los conceptos básicos de la programación de este tipo de placas y, al no estar limitado por las especificaciones de la placa programada, permite explicar conceptos de electrónica aplicada generales. Adicionalmente, su alto nivel de personalización hace que sea

*Universidad de Buenos Aires. Facultad de Ciencias Exactas y Naturales. Departamento de Computación. Buenos Aires, Argentina.

†CONICET-Universidad de Buenos Aires. Instituto de Ciencias de la Computación (ICC). Buenos Aires, Argentina.

adaptable tanto para un público escolar, docente, profesional y hasta para quien simplemente desea armar un proyecto personal en Arduino de forma sencilla y rápida. En la Figura 1 se muestra la interfaz general de la herramienta.

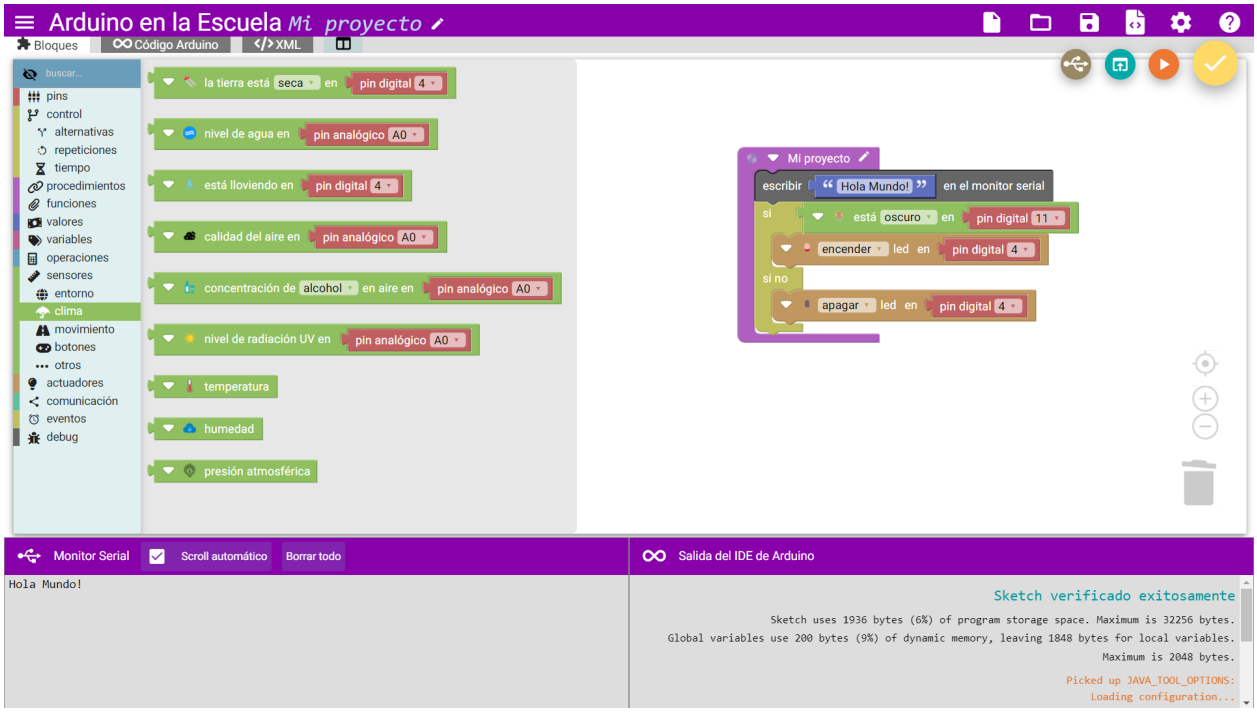

Figura 1: Interfaz gráfica de *Arduino en la Escuela*. Se muestra un programa de ejemplo (derecha), así como la caja de herramientas (izquierda), la salida del IDE de *Arduino* (abajo a la derecha) y la salida del monitor serial (abajo a la izquierda).

Las placas *Arduino* se usan cada vez más para enseñar conceptos de programación, electrónica y robótica. El lenguaje nativo para programarlas es *C++* que, a pesar de ser un lenguaje de alto nivel, presenta complicaciones a estudiantes sin experiencia, sobre todo frente a lenguajes como *Python*, cuya sintaxis es más similar a la del pseudocódigo y presenta mayor legibilidad. Varios estudios advierten acerca de la elevada curva de aprendizaje de *C++* y justifican de esta manera que se implementen entornos de programación por bloques, ya que presentan más similitudes con el lenguaje natural (Su y Lin, 2018). Algunos de los entornos de programación de Arduino más recientes son *ArViz* (Chochiang et al., 2019), *Ardestan* (Nishino, 2019) y *BEESM* (Seraj et al., 2018). Sin embargo, ninguno presenta características destacables respecto a las herramientas en las que están basadas (*Ardublockly*[1] y *Blocklyduino*[2]), sino que son adaptaciones o extensiones de las mismas a contextos específicos.

1.1. Contexto de la enseñanza de Programación y Robótica

El proyecto de *Arduino en la Escuela* se contextualiza en un panorama en el cual las Ciencias de la Computación se han instaurado como una disciplina esencial para el aprendizaje de habilidades lógicas y resolución de problemas. El razonamiento abstracto, el trabajo con problemas y el pensamiento creativo son capacidades que se desarrollan no

[1] https://ardublockly.embeddedlog.com/index.html

[2] https://github.com/BlocklyDuino/BlocklyDuino/wiki

solamente mediante el aprendizaje de las Matemáticas, sino también con la Computación. Si bien estas habilidades son comunes a ambas ciencias, dado que la forma tradicional de abordar la Matemática en la escuela es mediante problemas de abstracción numérica, las Ciencias de la Computación –que incluyen a la Robótica y a la Inteligencia Artificial, entre otras áreas– otorgan un enfoque diferencial que posibilita llevar la abstracción y resolución de problemas a un plano más general (Fundación Sadosky, 2013).

Sin ir más lejos, el Pensamiento Computacional incluye capacidades como el reconocimiento de patrones, la descomposición de problemas complejos, la comprensión y el diseño de sistemas, la búsqueda de heurísticas para encontrar una solución, y el diseño de algoritmos, entre otros (Nardelli, 2019; Wing, 2006). Todas estas habilidades resultan importantes en la educación media para la comprensión de fenómenos y procesos complejos estudiados en ciencias, de forma de ayudar a comprender el mundo en el cual vivimos, además de su aprovechamiento para el seguimiento de estudios superiores.

Por todas estas razones la enseñanza de las Ciencias de la Computación se promueve en distintos niveles escolares para que forme parte de la currícula general y no solamente como conocimiento específico relacionado a la Informática. Su incorporación desde los niveles básicos de la escolarización está tomando gran importancia en todo el mundo, por lo que hay un interés generalizado en llevarlo a cabo de manera efectiva en el corto y mediano plazo. En particular, en Argentina hace varios años que se trabaja en cómo introducir a la programación y el Pensamiento Computacional como parte de la enseñanza obligatoria. La resolución N° 263/15 del Consejo Federal de Educación (CFE) de la Argentina remarca que es de "importancia estratégica para el sistema educativo argentino la enseñanza y el aprendizaje de la programación durante la escolaridad obligatoria, para fortalecer el desarrollo económico-social de la Nación"[3]. En este sentido, otra resolución más reciente del CFE, bajo número 343/18[4], aprobó los Núcleos de Aprendizaje Prioritarios (NAP) para Educación Digital, Programación y Robótica, estableciendo la obligatoriedad de la enseñanza de estos saberes en todos los niveles de la educación obligatoria. En esa misma resolución se había definido un plazo de 2 años para la adecuación de la currícula de las distintas jurisdicciones del país, de forma de cumplir con las pautas de los NAPs definidos.

No obstante, a pesar de la intención y su promoción institucional, aún no existen programas educativos, recursos preestablecidos ni metodologías estándares para llevar las Ciencias de la Computación a todas las escuelas del país, ya que la implementación de la resolución del CFE citada es dispar, con resultados concretos recién en el nivel medio, y con baja cobertura en inicial y primario. Por otro lado, el campo de la Didáctica de la Computación se encuentra en una etapa de desarrollo inicial, donde queda mucho por investigar para conocer qué tipo de recursos son propicios a emplear de acuerdo a cada contexto educativo, lo que dificulta el cumplimiento de los objetivos planteados. Sin embargo, se espera un gran desarrollo del área en los próximos años, debido no sólo a las demandas a nivel gubernamental, sino también por los requerimientos de los sectores productivos, que requieren cada vez más personas formadas en programación y otros temas relacionados.

La Robótica viene con una rica historia en el campo de la investigación y, desde hace tiempo, también, con su utilización para la enseñanza de programación en diferentes niveles educativos. Esta elección radica en su naturaleza tangible la cual facilita la comprensión y el dimensionamiento del comportamiento lógico en estudio en un sentido físico, haciendo que su aprendizaje sea más atractivo e interactivo (Horn et al., 2009).

[3] https://cfe.educacion.gob.ar/resoluciones/res15/263-15.pdf

[4] https://www.argentina.gob.ar/sites/default/files/res_cfe_343_18_0.pdf

El hecho de que se haya popularizado tanto la enseñanza de la Computación a través de la Robótica dio lugar al desarrollo de múltiples herramientas didácticas para facilitar la tarea de enseñanza, siendo los entornos de programación por bloques los predominantes (Melo et al., 2020). Así, la gran variedad de entornos disponibles genera la duda de qué tanto se adecua cada uno a los objetivos didáctico-pedagógicos que definieron al momento de su creación, por lo que es necesario determinar qué entorno debería elegirse de acuerdo a cada contexto de uso. Esta clase de interrogantes resulta clave analizar ante la necesidad de incorporar las Ciencias de la Computación a la currícula escolar, en este caso, por medio de la robótica utilizando entornos de programación por bloques.

Debido a que estos entornos son los más usados para abordar la temática y dado el carácter disruptivo e innovador de la robótica como recurso pedagógico, en los últimos años gran parte de ellos se desarrollaron para trabajar con un hardware en particular, ya que en general forman parte de un proyecto que incluye el diseño de un robot específico. Si bien este enfoque es muy utilizado, tiene la desventaja de acotar el campo de aplicación de los conceptos aprendidos y, por lo tanto, limita los posibles contenidos a desarrollar a aquellos que se adaptan a lo permitido por el sistema. De esta forma se reduce la versatilidad de la herramienta utilizada, no permitiendo encarar problemas más generales ni profundizar su desarrollo de acuerdo a las necesidades de cada institución.

En contraposición a estos entornos dedicados, creados para un modelo particular de robot, existen otros más generales en los que no se programa un sistema particular sino que permiten escribir programas genéricos para una placa *Arduino*.

En este trabajo mencionaremos distintos aspectos de la herramienta *Arduino en la Escuela*, en tanto permitiría resolver los desafíos planteados anteriormente. Así, en la Sección 2 se eligen y se evalúan distintas características para los sistemas de programación de placas *Arduino*. En la Sección 3 se hace una descripción de *Arduino en la Escuela*, mientras que en la Sección 4 se resumen algunas experiencias y posibles escenarios de uso para la herramienta. Por último, en la Sección 5 se realiza un análisis final de la herramienta, y se marcan posibles líneas de trabajo a futuro.

2. Revisión de herramientas de programación para *Arduino*

Resulta difícil determinar qué aspectos considerar al realizar una comparativa entre diferentes entornos de programación por bloques para *Arduino*. Las funcionalidades básicas son similares, pero difieren en ciertos aspectos importantes al momento de decidir cuál es el recurso más adecuado para el contexto deseado. Aunque muchas de las iniciativas apuntan a crear la herramienta más versátil, aplicable a cualquier contexto y a cualquier público, lo cierto es que cada una está diseñada con propósitos específicos, y con determinado modelo de *público objetivo* en mente, que no siempre es fácil de generalizar.

En este sentido, Melo y otros (Melo et al., 2020) afirman que no hay acuerdo en cuál es la mejor herramienta para enseñar a programar con *Arduino* debido a la necesidad de soportar múltiples contextos educativos. De esta forma, queda claro que no hay ni puede haber una herramienta que sea la más adecuada para cualquier contexto posible. Cada una de las existentes tiene ventajas y desventajas, así como situaciones en donde sería beneficioso usarla y otras en las cuales no es así.

2.1. Aspectos a analizar

A continuación resumimos cuatro aspectos seleccionados para la evaluación de las herramientas disponibles.

Facilidad de instalación

Para poder ser utilizado en las escuelas, el entorno debería instalarse en cada equipo que disponga cada institución. La tarea de instalar la aplicación en varios dispositivos puede ser extremadamente compleja y larga cuando los pasos a seguir para realizar la instalación no son lo más simples posibles, a lo que se le suma cualquier configuración adicional e instalación de dependencias que la aplicación pudiera requerir. Además, las computadoras en las escuelas suelen tener deshabilitados los permisos de administrador con lo cual una característica deseable es que se pueda usar como una aplicación ejecutable sin requerir instalación.

Multiplataforma

Si bien Windows sigue siendo el sistema operativo más común para usuarios finales, cada vez más se utilizan sistemas de código abierto, basados en Linux. En forma más general, es importante el uso de software libre, más aún en espacios educativos de gestión estatal, en tanto permite el acceso irrestricto al conocimiento, y hace su aporte para alcanzar la soberanía tecnológica.

Así, las computadoras otorgadas por el Estado Argentino tienen ambos tipos de sistemas operativos y se espera que en los siguientes programas se distribuyan computadoras con *Huayra*[5], distribución de Linux desarrollada por el Estado en el marco del Programa Conectar Igualdad. Si bien esto no significa que necesariamente se deje de usar Windows, la tendencia es adoptar cada vez más los sistemas y programas de código abierto. De allí que sea deseable que los recursos para enseñar no se encuentren limitados a un sistema operativo en particular.

Independencia de internet

Muchas escuelas aún no tienen conexión a internet estable, y en algunas localidades tampoco tienen en los hogares. De esta manera, es importante garantizar la igualdad de condiciones para trabajar en el hogar –donde también se espera que ocurra el acto educativo– con las mismas herramientas usadas en el aula. Así, esperamos utilizar una herramienta en las escuelas que no dependa de la conexión a Internet para funcionar.

Disponibilidad en español

El uso del lenguaje nativo resulta primordial en los recursos didácticos a emplear en la etapa infantil, donde aún se encuentran en pleno desarrollo las habilidades de comunicación. La elección del vocabulario a emplear tanto para explicar conceptos así como también para que cada estudiante se apropie de saberes, debe hacerse considerando la riqueza expresiva del idioma. De allí que sea indispensable que el lenguaje usado por la herramienta y el que conozcan quienes la usan sea común, más aún si se quiere llevar una herramienta a cualquier escuela.

2.2. Análisis preliminar

Las herramientas más conocidas y usadas tienen como principal desventaja el idioma, ya que, si bien la mayoría tiene una versión en español, las traducciones suelen ser vagas o estar incompletas, y esto es problemático para alguien que está en la etapa inicial del aprendizaje de Robótica y Programación. A esto se le suman ciertas incoherencias en

[5] https://huayra.educar.gob.ar/

cuanto a términos utilizados en la propuesta didáctica que se quiere seguir, con respecto a los términos arbitrarios elegidos por la persona que realizó la traducción.

Como se mencionó anteriormente, es importante tener en cuenta la dependencia de la conexión a internet. La mayoría de las herramientas tienen versiones *offline*, pero algunas sólo funcionan en línea, lo que las descarta totalmente para su uso en escuelas sin acceso a internet. Además, al tener que ejecutarse desde un navegador, se complejiza técnicamente la forma en que la aplicación se conecta con la placa *Arduino*, ya que algunas de las herramientas exigen instalar un complemento para el navegador que no necesariamente funciona en cualquier sistema operativo. Las versiones *offline* también suelen tener la desventaja de ser específicas para determinadas plataformas, acotando también los espacios en los que pueden ser utilizadas.

Finalmente, se considera la disponibilidad de bloques específicos como un factor limitante tanto para la facilidad en el aprendizaje como para el abanico de posibles proyectos a encarar utilizando la herramienta en cuestión. Con muy pocos bloques se dificulta planificar y realizar proyectos medianamente interesantes, y la falta de bloques específicos puede hacer que ciertos proyectos sean demasiado complejos o, en ocasiones, imposibles. Tener demasiados bloques, en contraposición, dificulta el aprendizaje autoasistido y la exploración, ya que genera incertidumbre a la hora de elegir qué bloques usar. También hay que tener en cuenta qué tan expresivos son estos bloques: pocos bloques facilitan el aprendizaje, pero si estos son de muy bajo nivel tampoco se alcanza el efecto deseado.

Nombre	Fácil de instalar	Multiplataforma	Funciona sin internet	Está en español
TinkerCAD	**Sí**	**Sí**	**No**	**Sí**
SenseBlock	**Sí**	**Sí**	**No**	**No**
Scratch 4 Arduino	**No**	**Sí**	**Sí**	**Sí**
Minibloq	**No**	**No**	**Sí**	**Sí**
mBlock	**No**	**No**	**Sí**	**En parte**
Blocklyduino	**Sí**	**Sí**	**Sí**	**No**
Blocklyrduino	**Sí**	**Sí**	**No**	**Sí**
Blocklino	**Sí**	**No**	**Sí**	**No**
Arduino blocks	**Sí**	**Sí**	**No**	**Sí**
Arduoblockly	**Sí**	**Sí**	**Si**	**En parte**
Ardublock	**Sí**	**Sí**	**Sí**	**No**
Arduino en la Escuela	**Sí**	**Sí**	**Sí**	**Sí**

Tabla 1: Comparación de las principales características buscadas en la herramienta ideal para enseñar a programar Arduino en las escuelas. La columna “Fácil de instalar” hace referencia a si puede usarse sin necesidad de permisos de administrador (ya sea como aplicación ejecutable o como programa instalado en el sistema operativo). La columna “Multiplataforma” indica si además de funcionar en Windows, funciona en sistemas basados en Linux.

Los aspectos considerados son aquellos que consideramos relevantes para determinar qué herramienta utilizar. Como ya se ha mencionado, reflexionar sobre el contexto educativo en el cual se quiere aplicar la herramienta comparado con aquel para el que fue pensada resulta trascendental. A continuación profundizaremos en el entendimiento de las características del contexto para el cual fue pensado *Arduino en la Escuela*.

En la Tabla 1 se presenta una comparación de las herramientas más conocidas de programación por bloques de

Arduino a partir de las cuatro características mencionadas anteriormente. Como se observa, ninguna logra cumplir con todas satisfactoriamente.

2.3. Otras características evaluadas

En la Tabla 1 se incluyen las principales características analizadas en este trabajo, pero también se incluyeron otras como parte del análisis.

Una de ellas es la **disponibilidad**. Si bien todas las herramientas analizadas son gratuitas, *Blocklyrduino*[6] requiere un *plugin* pago para poder bajar el código a la placa. Por otro lado, *TinkerCad*[7] y *Arduino blocks*[8] requieren registrarse para poder utilizar la herramienta.

También consideramos los **requerimientos técnicos de uso e instalación**. La mayoría de las herramientas analizadas tiene una versión *online*, con lo cual no sería necesaria instalación alguna, aunque a veces requieren de *plugins* adicionales para poder bajar el código a la placa. De aquellas que tienen una versión *offline*, *Ardublock*[9] necesita *Java* y *Blocklyduino*, *Blocklino*[10] y *Ardublockly* se pueden usar sin necesidad de instalar (i.e., no requieren permisos de administrador).

Al analizar la **documentación de usuario**, muy pocas tienen y sólo Arduino blocks tiene en español. Adicionalmente, algunas herramientas que formaron parte del análisis no tienen **mantenimiento reciente**, como es el caso de *Minibloq*[11], *Blocklyduino*, *Ardublockly* y *Ardublock*.

De todas las herramientas analizadas sólo *Blocklyrduino* y *Blocklino* permiten **personalizar el *toolbox***. Por otro lado, *Minibloq, Blocklyrduino, Blocklino, Arduino blocks* y *Ardublock* tienen acceso al **monitor serial incorporado**.

En cuanto a la **disponibilidad de bloques**, la mayoría cuenta con los básicos y más usados, aunque sólo *Blocklyrduino* y *Arduino blocks* proveen un extenso catálogo de bloques específicos. Con respecto a *Scratch 4 Arduino*[12], no se lo analiza como un entorno de propósito general para *Arduino*, sino que es sólo un intérprete de *Scratch* en la placa, por lo que no hay forma de ver el código *Arduino* generado por la herramienta.

3. Características destacadas de Arduino en la Escuela

3.1. Funcionalidades

Un aspecto interesante que casi ninguna de las herramientas analizadas tiene en consideración es el control de ciertas restricciones en el código dependientes de la arquitectura de la placa. Un ejemplo muy simple es la asignación de roles para los pines. Al programar una placa *Arduino*, un pin puede ser configurado como de entrada o de salida, pero no ambos. Son muy pocas las herramientas que impiden utilizar un mismo *pin* para entrada y para salida, lo cual

[6] https://github.com/technologiescollege/Blockly-at-rduino

[7] https://www.tinkercad.com/dashboard

[8] http://www.arduinoblocks.com/

[9] http://blog.ardublock.com/

[10] https://github.com/fontainejp/blocklino

[11] http://blog.minibloq.org/

[12] http://s4a.cat/index_es.html

es muy importante a tener en cuenta ya que un usuario podría pensar que su programa es correcto sólo por el hecho de que el entorno de programación se lo permitió generar. Sin embargo, al intentar compilar el código obtendrá un error que probablemente no pueda interpretar.

Pensando en un público objetivo sin mucho conocimiento previo de programación, sería deseable que sea la herramienta y no el compilador quien pueda detectar estas fallas y ayude a quien la está usando con mensajes claros. Un buen ejemplo de esto es el sistema de advertencias de *Ardublockly*. Este sistema, heredado de *Blockly*, permite guiar a la persona que está usando la herramienta al momento de determinar por qué cierta combinación de bloques no es válida (véase Figura 2a). Los mensajes claros son vitales para entender el problema y buscar cómo solucionarlo. *Arduino en la Escuela* utiliza el mismo sistema, pero, además, distingue entre advertencias y errores, de manera análoga a como lo hace la herramienta de programación de *Apps* de celular *AppInventor*[13].

Uno de los mecanismos que implementa *Blockly*[14] para facilitar el aprendizaje es impedir la conexión de bloques que visualmente parecieran encajar, para evitar un error de tipos (véase Figura 2b). Esto se puede ver por ejemplo al intentar usar un bloque numérico como condición del bloque de alternativa condicional. Sin embargo, notamos que esto tiene algunas desventajas didácticas. Por ejemplo, a quienes están empezando a programar y no tienen ninguna capacitación respecto a teoría de tipos, les es difícil entender por qué un bloque numérico no puede usarse como condición del bloque de alternativa condicional. Durante el dictado de talleres por parte de nuestro equipo, se notó que los usuarios tratan de unirlos varias veces pensando que fue un error propio, hasta que se dan cuenta que es la herramienta la que no les deja hacerlo. Incluso después de haberse dado cuenta, no queda claro que entiendan la razón. Es por eso que *Arduino en la Escuela* en lugar de impedir que estos bloques se conecten, lo permite, y genera un mensaje de error indicando que la combinación es inválida (véase Figura 2c).

Una de las grandes falencias en la mayoría de las herramientas es que al estar basadas en *Blockly*, no permiten hacer chequeo estático de tipos. *Blockly* fue desarrollado para lenguajes de tipado dinámico, con lo cual, no hay nada que impida, por ejemplo, asignar a una misma variable dos valores de tipos distintos. Esto es un problema al intentar generar código *Arduino*, ya que permite generar código que no se puede compilar. En cambio, *Arduino en la Escuela* implementa un sistema avanzado de chequeo de tipos[15] que le permite detectar muchas combinaciones inválidas, como, por ejemplo, asignación de una misma variable con valores de tipos distintos, argumentos de tipos inválidos en invocaciones a funciones o procedimientos, entre otros (véase Figura 2d).

Otra funcionalidad relevante de *Arduino en la Escuela* es que permite tratar a los pines de la placa como un tipo más de datos. Es por esto que provee bloques específicos para los pines de forma de poder nombrarlos y reutilizarlos en varios lugares, a diferencia de la mayoría de las herramientas, en las que el *pin* debe seleccionarse de un menú desplegable (véase Figura 2e).

Finalmente, se destaca el hecho de que *Arduino en la Escuela* permite trabajar en un modelo orientado a eventos gracias a su sistema de manejo de eventos, el cual se implementa como una capa de abstracción por encima de las interrupciones de hardware de *Arduino*(véase Figura 2f).

[13] https://appinventor.mit.edu/

[14] https://developers.google.com/blockly

[15] Se puede ver una demostración en https://gpfernandezflorio.github.io/blockly-inferencia/

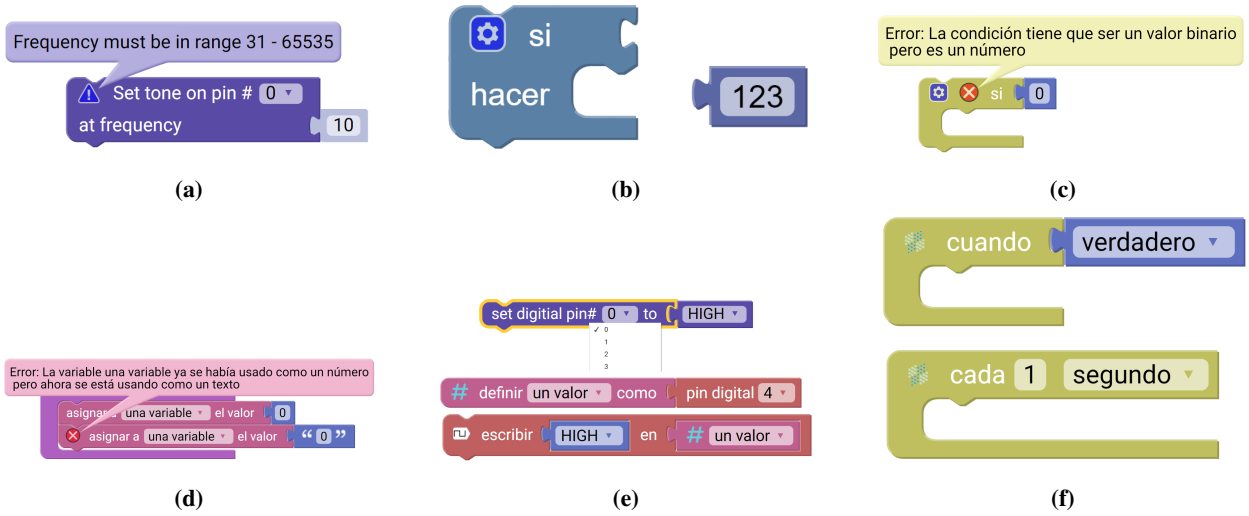

Figura 2: Funcionalidades a partir de los bloques que las implementan. (a) Advertencia de Ardublockly para indicar una combinación de valores inválidos. (b y c) Comparación entre Blockly (b) y Arduino en la Escuela (c) al intentar realizar una conexión inválida. (d) Detección de una asignación de variables inválida por error de tipado. (e) Comparación entre Ardublockly (arriba) y Arduino en la Escuela (abajo) en la forma de referirse a los pines de la placa. (f) Bloques ejemplos de Arduino en la Escuela para manejar eventos.

3.2. Personalización

Arduino en la Escuela fue pensado para introducir a las personas a la programación y la electrónica aplicada. Con esos objetivos en mente, se desarrolló de forma que sea lo más simple posible y que abstraiga la mayor cantidad de información que pudiera entorpecer el aprendizaje. Sin embargo, también permite planificar proyectos más complejos habilitando configuraciones avanzadas. Un ejemplo de esto es la posibilidad de trabajar sólo con variables locales, sólo con variables globales, o permitir ambas. Como una opción por defecto en un contexto de aprendizaje inicial, sólo se pueda trabajar con variables locales, pero puede permitirse trabajar con variables globales con sólo unos clics.

Algo similar sucede con las advertencias que se muestran. Muchas veces estas son sólo sugerencias para ayudar en el proceso de aprendizaje, pero podrían ser molestas para alguien en una etapa más avanzada. Así, *Arduino en la Escuela* permite configurar qué tipo de advertencias se muestran y cuáles se tratan como errores, de la misma forma que lo hace un compilador.

Otro elemento del sistema que se puede configurar es el conjunto de bloques disponible (o *toolbox*), ya que se puede elegir entre varios disponibles. Además, se permite crear otros personalizados, lo cual es especialmente útil para docentes que quieren planificar ejercicios específicos para los cuales necesitan sólo algunos bloques.

3.3. Usabilidad

Algunas características de *Arduino en la Escuela* permiten simplificar su uso y promover el aprendizaje. Uno de los diferenciales importantes es la incorporación del monitor serial a la interfaz web. Muchas de las herramientas existentes requieren abrir la interfaz del *IDE de Arduino* para poder acceder a esta información, mientras que en *Arduino en la escuela* todo forma parte de la misma aplicación. Adicionalmente, esto permite subir el código a la placa mientras el monitor serial está abierto, algo que siempre ocasiona problemas a quienes están empezando.

Algunas herramientas sólo proveen los bloques básicos de entrada y salida, lo que hace que sea difícil trabajar con proyectos grandes, mientras que otras tienen muchas variantes de los mismos bloques. Por ejemplo, un bloque para encender un led y otro para encenderlo durante una determinada cantidad de tiempo. Esto aumenta la capacidad expresiva del lenguaje, aunque tener demasiados bloques también puede ser un problema a la hora de encontrar el bloque adecuado para una determinada tarea. Algunas herramientas solucionan este inconveniente permitiendo personalizar el conjunto de bloques disponibles, y *Arduino en la Escuela* también posee esta característica, además de que permite encontrar bloques a través de un buscador, lo cual posibilita encontrar bloques específicos a partir del ingreso de texto. Esto también es muy útil para aquellas personas que se inician en el tema, y aún no han tenido tiempo para familiarizarse con la interfaz o, más específicamente, con los bloques disponibles y su ubicación dentro del *toolbox*. Sumado a eso, implementa un sistema de opciones avanzadas para muchos de sus bloques –principalmente los relacionados a sensores y actuadores– que permite que un bloque sea lo suficientemente simple como para que sea fácil de entender, y que a la vez se pueda complejizar para obtener comportamientos más específicos (véase Figura 3a). Otro ejemplo de esta propuesta es que el bloque principal también tiene un modo avanzado para configurar su funcionamiento, ya sea como un ciclo infinito o como una secuencia de instrucciones que se ejecuta una única vez (véase Figura 3b). Entre estas opciones se encuentra también la posibilidad de implementar los procedimientos nativos de Arduino *setup* y *loop* para quien ya está en una etapa más avanzada del aprendizaje.

(a) **(b)**

Figura 3: Algunos de los bloques de Arduino en la Escuela que admiten personalización avanzada. Mediante la flecha blanca se expande el bloque y se muestran opciones avanzadas de configuración. (a) El bloque de led colapsado (arriba) y expandido (abajo). (b) El bloque principal expandido que permite agregar código para ejecutar en la inicialización del Arduino y definir si el programa principal se va a ejecutar una vez o repetirse indefinidamente.

Recientemente se le agregó a *Arduino en la Escuela* un nuevo modo que permite que se lo ejecute en un servidor web, para que esté disponible para accederse a través de internet. En este modo están disponibles todas las características excepto aquellas que requieren el *IDE de Arduino* para funcionar –esto es, verificar el código, bajarlo a la placa y abrir el monitor serial–. Desde este modo se puede construir un programa y luego descargar el código *Arduino* generado para seguir editándolo en el editor de *Arduino*.

Finalmente se destaca la documentación de usuario. Aunque aún está en proceso de mejora, se está trabajando para que sea lo más descriptiva posible y que esté completamente en español. Un dato interesante es que desde la documentación se pueden observar cada uno de los bloques disponibles y hasta interactuar con ellos. Posteriormente

se espera agregar tutoriales y propuestas de ejercicios.

Respecto a la interfaz de usuario, algunas de las herramientas existentes están demasiado sobrecargadas, dificultando la tarea de organización para desarrollar la actividad. La gran disponibilidad de características de *Arduino* motiva a tenerlas todas accesibles a la vez, aunque esto podría abrumar a la persona que está comenzando a utilizar una herramienta. En este sentido, es preferible una interfaz simple, con pocas funcionalidades accesibles a simple vista. Las funcionalidades adicionales más específicas no deberían estar demasiado ocultas tampoco sino que deberían ser fáciles de encontrar en el momento que sean necesarias.

Muchos de los entornos analizados siguen este enfoque, y eligieron una interfaz simple y cómoda. Sin embargo, no siempre son fáciles de encontrar las características adicionales por lo que nunca llegan a utilizarse. Otro aspecto notable respecto a los entornos que prefieren este tipo de interfaz es que no suelen aprovechar el espacio de la pantalla, dejando mucho margen sin usar. *Arduino en la Escuela* se inclina por una interfaz simple, destinando la mayor parte del espacio visible en la pantalla inicial al sector de bloques. Todas las configuraciones y menús adicionales están sobre los bordes de la pantalla y pueden investigarse en el momento en que sean necesarios.

4. Algunas experiencias y posibles escenarios de uso de la herramienta

Por todas las características mencionadas en las secciones anteriores creemos que *Arduino en la Escuela* tiene un gran potencial para ser utilizado en múltiples contextos educativos. Puede utilizarse para enseñar conceptos generales de Programación y Ciencias de la Computación a personas que no tuvieron ninguna educación previa relacionada, ya que es ideal para introducir los conceptos a través de ejemplos del mundo real (“si está oscuro, apagar la luz”, “mientras no haya obstáculo, avanzar”) ignorando detalles sobre la electrónica. También puede ser utilizado para enseñar conceptos de electrónica aplicada, más allá de la programación, haciendo uso de los bloques de lectura de los sensores y su posterior observación en el monitor serial.

En particular, destacamos la experiencia de un taller dictado a principios de 2019, para estudiantes de 11 a 17 años, en el que se combinaron ambos enfoques para enseñar programación y electrónica aplicada a la vez. Incluso cuando la herramienta estaba aún en etapa de desarrollo, quienes participaron del taller expresaron su conformidad al utilizarla y apreciaron su versatilidad y facilidad de uso (Fernandez-Florio et al., 2019). Durante el resto del año, mientras la herramienta seguía evolucionando, se continuó utilizando en distintos talleres, tanto para estudiantes como para docentes de distintos niveles educativos, actividades de divulgación y hasta en el curso docente *La Programación y su Didáctica*[16]. Sin embargo, *Arduino en la Escuela* no se limita a ser sólo una herramienta educativa, ya que la gran variedad de bloques y sus configuraciones avanzadas la convierten también en una excelente herramienta para encarar proyectos personales interesantes. Por ejemplo, en uno de los talleres realizados se lo utilizó para programar el funcionamiento de una estación meteorológica con estudiantes de escuelas secundarias, como parte de la actividad *Científics por un Día de Exactas-UBA*[17].

Otro escenario de uso podría ser como un paso intermedio para introducir a alguien que ya sabe programar y quiere aprender *Arduino* propiamente dicho, antes de pasar a programar en su *IDE*. Así, las dificultades que surgen de

[16] https://www.dc.uba.ar/el-dc-apuesta-fuertemente-a-la-didactica-de-la-programacion/

[17] https://www.dc.uba.ar/conociendo-y-midiendo-las-inundaciones-urbanas-mediante-el-uso-de-tecnologias/

ir de un lenguaje a otro se pueden resolver de a poco, en lugar de tener que lidiar con todas juntas.

Finalmente, podemos pensar en una situación de alguien que no necesariamente quiere aprender a programar, aunque le gustaría poder desarrollar un proyecto usando *Arduino*. Todo esto hace de *Arduino en la Escuela* una herramienta altamente versátil.

5. Discusión y conclusiones

En este trabajo se resumieron las características principales del entorno de programación por bloques para placas *Arduino* llamado *Arduino en la Escuela*, que aunque guarda similitudes con otras herramientas ya existentes, posee características que lo hacen más adecuado para ciertos contextos. Nos concentramos en aquellas características que permiten utilizarla en la escuela, prestando particular atención a las necesidades de aquellas con baja disponibilidad de recursos. Las experiencias que hemos tenido con estudiantes han sido satisfactorias, con una buena recepción por parte de los usuarios. Cada taller, a su vez, sirvió para encontrar fallas y recibir sugerencias o pensar ideas sobre mejoras que luego fueron implementadas. De esta forma, concluimos que Arduino en la Escuela satisface los objetivos definidos al comienzo del proyecto.

Sin embargo, debemos analizar una limitación que surge como consecuencia de la virtualización de las clases debido a la pandemia. *Arduino en la Escuela* se planteó desde el principio como una herramienta para usarse en clases presenciales, de forma de aprovechar la riqueza de la tangibilidad de *Arduino* (es decir, que puedan interactuar físicamente con las placas). Si bien previamente al desarrollo observamos herramientas que proveen un simulador para probar el código generado sin necesidad de tener una placa física, como *TinkerCAD*, nunca lo consideramos una prioridad. La imposibilidad de probar el código sin tener una placa se ha vuelto una dificultad considerable en esquemas de clases virtuales, lo que queda como trabajo a futuro, de forma de incorporar un simulador de circuitos con *Arduino* para cubrir esta necesidad.

Bibliografía

Chochiang, K., Chaowanawatee, K., Silanon, K., y Kliangsuwan, T. (2019). Arduino Visual Programming. En 23rd International Computer Science and Engineering Conference (ICSEC).

Fernandez-Florio, G., Cossio-Mercado, C., Macario-Cabral, D. y Reartes, C. (2019) Electrónica aplicada para Ciencias y Tecnología con Arduino en la Escuela. Póster presentado en Segundas Jornadas Argentinas de Didáctica de la Programación. Disponible en https://jadipro.unc.edu.ar/wp-content/blogs.dir/34/files/sites/34/2019/07/JADIPRO2019_paper_24.pdf

Fundación Sadosky (2013). CC-2016: Una propuesta para refundar la enseñanza de la computación en las escuelas Argentinas. Disponible en http://www.fundacionsadosky.org.ar/wp-content/uploads/2014/06/cc-2016.pdf

Horn, M.S., Solovey, E.T., Crouser, J.C., y Jacob, R.J.K. (2009). Comparing the Use of Tangible and Graphical Programming Languages for Informal Science Education. En Proceedings of the SIGCHI Conference on Human Factors in Computing Systems, April 2009, pp. 975–984, doi: 10.1145/1518701.1518851

Kushner, D. (2011). The making of Arduino. En IEEE Spectrum, 26-11-2011. Disponible en https://spectrum.ieee.org/the-making-of-arduino

Melo, J., Fidelis, M., Alves, S., Freitas, U., y Dantas, R. (2020). A comprehensive review of Visual Programming Tools for Arduino. En Proceedings of Latin American Robotics Symposium (LARS), 2020 Brazilian Symposium on Robotics (SBR) and 2020 Workshop on Robotics in Education (WRE), doi: 10.1109/LARS/SBR/WRE51543.2020.9307023.

Nardelli, E. (2019). Do we really need computational thinking?. En Communications of the ACM, Vol. 62, No. 2, pp. 32–35, doi: 10.1145/3231587.

Nishino, H. (2019). Ardestan: A Visual Programming Language for Arduino. En The Adjunct Publication of the 32nd Annual ACM Symposium on User Interface Software and Technology, pp 93–95, doi: 10.1145/3332167.3357126.

Seraj, M., Autexier, S., y Janssen, J. (2018). BEESM, a Block-Based Educational Programming Tool for End Users. En Proceedings of the 10th Nordic Conference on Human-Computer Interaction, pp. 886–891, doi: 10.1145/3240167.3240239.

Su, J., y Lin., T. (2018). Building a Simulated Blockly-Arduino-Based Programming Learning Tool: A Preliminary Study. En 7th International Congress on Advanced Applied Informatics (IIAI-AAI), pp. 378–381, doi: 10.1109/IIAI-AAI.2018.00082.

Weintrop, D., y Wilensky, U. (2015). To Block or Not to Block, That is the Question: Students' Perceptions of Blocks-Based Programming. En Proceedings of the 14th International Conference on Interaction Design and Children, pp. 199–208, doi: 10.1145/2771839.2771860.

Weintrop, D. (2015). Minding the gap between blocks-based and text-based programming. En Proceedings of the 46th ACM Technical Symposium on Computer Science Education (2015), ACM, p. 720, doi: 10.1145/2676723.2693622.

Wing, J. M. (2006). Computational thinking. En Communications of the ACM, Vol. 49, No. 3, March 2006, pp. 33–36, doi: 10.1145/1118178.1118215.

Hacia un estado del arte sobre programación tangible

Leandro Castro*
castro.leandro@exactas.unlpam.edu.ar
UNLPam

Verónica Artola
vartola@lidi.info.unlp.edu.ar
III-LIDI UNLP

Gustavo Astudillo*
astudillo@exactas.unlpam.edu.ar
UNLPam

Silvia Bast*
bast@exactas.unlpam.edu.ar
UNLPam

Resumen

Los saberes sobre programación comenzarán a ser parte de la currícula escolar, en Argentina, a partir del año 2021 en educación inicial, primaria y secundaria. En este contexto, los lenguajes de programación tienen un rol fundamental para abordar los saberes propuestos en los Núcleos de Aprendizajes Prioritarios aceptados en 2018. Existen lenguajes de programación textuales que derivan muchas veces en errores de sintaxis y requieren un esfuerzo extra por quien debe aprender a utilizarlo. Así mismo, también existen en menor medida, lenguajes de programación del tipo icónico (mediante el uso de bloques) como Blockly (y los derivados de este) o el popular Scratch. Independientemente de si es a través de escritura o mediante bloques, para poder hacer uso de estas herramientas, es necesario poseer conocimientos mínimos sobre informática, ya que para ello se requieren un conjunto de comandos aprendidos, como palabras claves que se deben ingresar o teclas de función que se deben presionar. Aquí es donde la incorporación de las Interfaces de Usuario Tangibles refleja diversos beneficios al permitir una forma de interacción diferente. En este artículo se presentan los avances de investigación acerca del estado del arte sobre la programación tangible en el contexto educativo.

Palabras clave: educación, programación tangible,interfaces de usuario tangible, estado del arte.

1. Introducción

En Argentina, a partir de la Resolución CFE N° 343/18[1] los saberes: **educación digital, programación y robótica** comenzarán a ser obligatorios en todo el país. En el año 2021 se deberán comenzar a implementar estos saberes dentro

*GrIDIE, Departamento de Matemática, FCEyN, UNLPam

[1]Los Núcleos de Aprendizajes Prioritarios para la Educación Inicial, Primaria y Secundaria fueron elaborados mediante un proceso que incluyó trabajo técnico, consultas regionales, discusiones y acuerdos federales. A partir de esta resolución la educación digital, la programación y la robótica comenzarán a ser obligatorios en todos los establecimientos del país. Disponible en https://www.educ.ar/recursos/150123/nap-de-educacion-digital-programacion-y-robotica?from=150199.

de las áreas correspondientes. Podemos observar distintas situaciones de enseñanza que se esperan promover en las y los estudiantes.

Para **Educación Inicial** el entorno del individuo es un elemento esencial en el cual a través de la exploración y observación busca soluciones ante problemas del mundo real mediante materiales concretos y/o digitales. En cuanto a **Educación Primaria - primer y segundo ciclo** - la creatividad y el error son parte del proceso en la elaboración de estrategias de resolución. Así mismo se hace un fuerte hincapié en el uso de la robótica como parte de la construcción de conocimientos relacionados con la programación. Para **Educación Secundaria - ciclo básico y orientado -** aparece la utilización de entornos de programación textuales e icónicos como herramientas para la elaboración de resoluciones de problemas aplicando estrategias y fragmentación de las soluciones. Al igual que en Educación Primaria se hace hincapié en la aplicación de la robótica como parte del proceso para la elaboración de soluciones vinculadas con la sociedad (entorno social, económico, ambiental y cultural) (Resolución CFE N. 343/18, 2018).

Se puede observar que recién en los niveles educativos de **Educación Secundaria** se comienzan a realizar menciones de entornos de programación tanto textuales como icónicos. En **Educación Inicial y Educación Primaria - primer y segundo ciclo -** los saberes sobre programación no se encuentran condicionados por la utilización de un entorno de programación en particular.

Los lenguajes de programación tienen un rol fundamental para abordar algunos de los saberes propuestos. Existen distintos tipos de lenguajes de programación de acuerdo a la expresión de sus instrucciones. Por un lado, los lenguajes de programación textuales que requieren que el programador escriba las instrucciones. En estos lenguajes, el código debe ser escrito siguiendo reglas de sintaxis que derivan muchas veces en errores y requieren un esfuerzo extra por quien debe aprender a utilizarlo. En este punto es importante mencionar que la gran mayoría de los lenguajes textuales utilizados - Python[2], C++[3], Java[4], entre otros - no han sido pensados para enseñar conceptos sobre programación. Por otro lado, existen en menor medida, lenguajes de programación del tipo icónico como Blockly[5] (y los derivados de este) o el popular Scratch[6], en los cuales las instrucciones se expresan a través de representaciones gráficas del estilo de bloques. Independientemente de si es a través de escritura o mediante bloques, el o la estudiante debe poseer conocimientos mínimos sobre informática para poder hacer uso de estas herramientas, ya que para ello se requieren un conjunto de comandos aprendidos, como palabras claves que se deben ingresar o teclas de función que se deben presionar. En este escenario, la incorporación de Interfaces de Usuario Tangibles (TUI por sus siglas en Inglés: Tangible User Interfaces) proporciona diversos beneficios al permitir una forma de interacción diferente (Bern y Horn, 2010).

En este artículo se presentan los avances de investigación sobre el estado del arte en el uso de interfaces de programación tangible en el contexto educativo. Dicha investigación se encuentra enmarcada dentro del trabajo final de Especialización en Tecnología Informática Aplicada en Educación (UNLP) de uno de los autores. La motivación de esta investigación radica en describir un estado del arte de la utilización de la programación tangible en el contexto

[2]Python es un lenguaje de programación que permite trabajar rápidamente e integrar sistemas de manera más efectiva. Disponible en: https://www.python.org/

[3]C++ es una variante del veterano C. Es un lenguaje orientado a objetos y multiplataforma.

[4]Java es un lenguaje de programación muy versátil utilizado para crear programas de computadoras, App de Android, etc. Disponible en: https://go.java/

[5]Blockly es una biblioteca que agrega un editor de código visual a aplicaciones web y móviles, utilizado en MIT App Inventor, Micro:bit, entre otros. Disponible en: https://developers.google.com/blockly

[6]Con Scratch es posible programar historias interactivas, juegos y animaciones. Disponible en: https://scratch.mit.edu/

educativo. Para ello se pretende:

- Identificar las bases teóricas y/o enfoques que sustentan la programación tangible.
- Identificar distintas experiencias que utilizan programación tangible en contextos educativos en los últimos diez años.
- Reconocer características de la utilización de la programación tangible en contextos educativos.
- Definir un conjunto de criterios que permitan el análisis de las experiencias seleccionadas.
- Analizar las experiencias educativas que hacen uso de programación tangible.

Este artículo se organiza como sigue: se aborda el marco teórico donde se presentan aportes significativos que han dado sustento a las interfaces de usuarios tangibles y a la programación tangible. Se presenta la metodología empleada para llevar adelante la investigación y se exponen los resultados obtenidos y las conclusiones alcanzadas.

2. Marco teórico

Un primer acercamiento a las Interfaces de Usuario Tangibles está dada por los autores Ishii y Ullmer (1997). A partir de sus postulados las TUI se podrían pensar como un puente entre el ciberespacio[7] y el medio ambiente. Estas "aumentan el mundo físico real al acoplar información digital a objetos y entornos físicos cotidianos" (Ishii y Ullmer, 1997, p. 234 - 241). Proponen tres conceptos claves para pensar las TUI:

- **Superficies interactivas**, compuestas por paredes, escritorios, puertas, etc.
- **Acoplamiento de bits y átomos**, estos serían objetos cotidianos manipulables como tarjetas, libros, etc.
- **Medio ambiente** compuesto por sonido, aire, luz y agua para la percepción del ciberespacio.

A lo largo de los años, con el avance en el estudio de las interfaces tangibles se han encontrado diversos beneficios al relacionarlas con los procesos de enseñanza y aprendizaje (Marshall et al., 2003; Marshall, Price, y Rogers, 2003; Price, 2008).

La incorporación de las interfaces tangibles en lenguajes de programación ha dado lugar a la programación tangible. La programación tangible permite construir programas organizando y/o conectando objetos físicos que representan elementos y comandos del lenguaje. En palabras de Horn y Jacob (2007) "un lenguaje de programación tangible es similar a un lenguaje de programación visual o basado en texto. Sin embargo, en lugar de usar imágenes y palabras en una pantalla de computadora, los lenguajes tangibles usan objetos físicos para representar varios elementos de programación, comandos y estructuras de flujo de control" (p. 159). Estos lenguajes pueden aprovechar las características físicas de los objetos tangibles (forma, tamaño, color, textura, etc) para expresar la sintaxis. En este sentido, se encuentran beneficiadas las actividades exploratorias, expresivas y experimentales (Julià et al., 2009). Cabe aclarar que, como mencionan Marshall et al.(2007), para poder realizar estas afirmaciones se necesitan pruebas empíricas y un mayor desarrollo teórico.

Basados en *What is the next generation of human-computer interaction?* (Jacob, 2006), Horn y Jacob (2007) expresan dos principios fundamentales de los lenguajes de programación tangibles:

1. **La interacción tiene lugar en el mundo real.** En los entornos de programación tangible no se programa a través de pantallas de computadoras donde encuentran distractores como juegos, chat en línea, etc. La programación

[7]El término ciberespacio fue acuñado por William Gibson en su novela Neuromancer y describe el mundo de los ordenadores y la sociedad creada en base a ellos. Otras definiciones y visiones surgen de los autores Michael Benedikt (desde la arquitectura), Javier Echeverría (desde la filosofía) y Manuel Castells (desde la sociología).

se desarrolla en un entorno más natural, como pueden ser sus bancos de clase o el suelo. Así mismo para el equipo docente esta es una característica que suma en flexibilidad para determinar la estructura y el tiempo de las actividades de programación en clase.

2. **La interacción se comporta más como el mundo real.** Aquí se hace uso de los conocimientos cotidianos del/la estudiante (no informáticos). Por ejemplo, los bloques a utilizar tienen terminaciones similares a las de un rompecabezas que imposibilita construcciones no válidas del lenguaje.

Existen varios desarrollos previos que han servido como sustento y han dado lugar a nuevas propuestas, a modo de ejemplo y con el fin de clarificar se describe aquí **AlgoBlocks**. Este entorno fue desarrollado por Suzuki y Kato (1993). El lenguaje está formado por una colección de cubos de aluminio que entrelazados representan un lenguaje similar a **Logo**[8]. Fue pensado para estudiantes de nivel primario y secundario con el objetivo de crear procedimientos (programas) a través de actividades que promuevan el aprendizaje colaborativo. Cada bloque (cubo) corresponde a un comando. El objetivo consiste en guiar a un submarino que está visible en la pantalla de una computadora. Luego de su implementación y posterior análisis **AlgoBlocks** demostró estar calificado como un entorno conversacional fomentando el aprendizaje colaborativo entre estudiantes.

Figura 1: Estudiantes utilizando AlgoBlocks Imagen extraída de **Project Bloks** (Suzuki Y Kato, 1993)

[8]"Logo es una variante del lenguaje de programación LISP, popular entre los investigadores de inteligencia artificial. Es un lenguaje poderoso pero simple para explorar la resolución de problemas lingüísticos y matemáticos" (McNerney, 2004)

3. Metodología

Para el desarrollo de esta investigación se emplea la revisión sistemática de literatura, metodología definida por Kitchenham et al. (2009). En esta metodología, la autora plantea: (a) definir preguntas de investigación, (b) trazar una estrategia de búsqueda (dónde buscar, con qué palabras claves), (c) establecer criterios de inclusión y exclusión, que serán aplicados, tanto para la selección inicial, como para la selección final.

Hasta la fecha de publicación de este artículo se han realizado búsquedas en **IEEE Xplore**[9] y **ACM Digital Library**[10]. Los filtros aplicados han sido por palabras claves y artículos publicados en los últimos diez años. Las palabras claves utilizadas son:

- tangible programming; education
- programación tangible; educación

Para el primer conjunto de palabras mencionadas se han obtenido 10 (IEEE Xplore) y 25 (ACM Digital Library) resultados. Mientras que para el segundo conjunto de palabras no surgieron artículos. A partir de estos resultados se ha realizado una primera lectura de cada uno de sus resúmenes. En base a ello se ha descartado un artículo obtenido de IEEE Xplore y tres de ACM Digital Library. La supresión de estos radica en que su contenido no aporta información significativa a los objetivos propuestos, por ejemplo: realidad aumentada aplicada en educación o la utilización de una herramienta que permite automatizar el hogar.

En adelante, el documento se centra en los artículos obtenidos y seleccionados de IEEE Xplore, dado que los artículos de ACM Digital Library están en proceso de análisis.

Para los artículos localizados en IEEE Xplore se ha realizado una lectura y análisis que ha derivado en la exclusión de cinco de ellos. Los motivos fueron:

- Doble publicación. Artículos con contenido similar, mismos autores y autoras. Se seleccionó el más actual.
- Artículo que presenta una propuesta de interfaz de usuario tangible para la creación de juego. No está pensado para el aprendizaje de programación.
- Artículo que realiza una revisión sistemática y no permite analizar las características de las propuestas definidas en el presente artículo.

Durante la lectura se ha considerado para el análisis:

- País donde se ha elaborado la publicación
- Características de los destinatarios (rango etario, alguna discapacidad, etc)
- Recursos físicos utilizados
- Recursos lógicos implementados
- El artículo presenta experiencias con el grupo destinatario

[9] **IEEE Xplore** ofrece acceso a textos completos de literatura técnica en ingeniería y tecnología. Disponible: https://ieeexplore.ieee.org/Xplore/home.jsp.

[10] **ACM** proporciona la principal biblioteca digital del campo de la informática y ofrece a sus miembros y a la profesión informática publicaciones, conferencias y recursos profesionales de vanguardia. Disponible en: https://dl.acm.org/.

4. Resultados y discusión

A partir de los resultados (Tabla 1) obtenidos se identifican dos artículos que están centrados en la comparación de interfaces de programación tangibles con interfaces gráficas. Un artículo que presenta una propuesta de programación tangible y otro que va en la misma línea, pero con el agregado de que la interfaz está pensada para personas con disminución visual.

Id	Autor	Año	Título
Sapounidis19	Sapounidis et al.	2019	Latent Class Modeling of Children's Preference Profiles on Tangible and Graphical Robot Programming
Kwon12	Kwon et al.	2012	Algorithmic Bricks: A Tangible Robot Programming Tool for Elementary School Students
Caceres18	Caceres et al.	2018	Tangible programming mechatronic interface for basic induction in programming
Kakehashi14	Kakehashi et al.	2014	Improvement of P-CUBE: Algorithm education tool for visually impaired persons

Tabla 1: Lista de artículos seleccionados para el análisis

Sapounidis19 y Kwon12 realizan las comparaciones de aplicaciones de programación con interfaces tangibles y con interfaces gráficas. El primero de los artículos, involucró estudiantes de seis a trece años que utilizaron ambas interfaces para programar el comportamiento de un robot Lego NXT. Se crearon parejas para las actividades propuestas y luego se les permitió interactuar libremente. Las aplicaciones utilizadas fueron V-ProRob (gráfica) y T-ProRob (tangible). T-ProBob está compuesta de cubos que representan parámetros y comandos y un robot Lego NXT. Los cubos se pueden conectar entre sí para generar secuencias de programación. Algunos de los comandos disponibles son girar a la izquierda/derecha, mover un paso hacia atrás/adelante, entre otros. Los cubos de parámetros son más pequeños y existen dos tipos: números y sensores. Además, es posible realizar condiciones, repeticiones y bucles anidados. Todos estos cubos se deben colocar en la "caja maestra" que al presionar el botón de ejecución lee los comandos y realiza la comunicación con una computadora en la cual se almacena el programa y posteriormente esta realiza la comunicación con el robot Lego NXT. V-ProRob se desarrolló de tal manera que cumpliera con las mismas prestaciones, pero a través de una pantalla en la cual se deben arrastrar y soltar los bloques de instrucciones (cubos) para crear y ejecutar secuencias de programación. A partir de este experimento se obtuvo que la mayoría de las y los participantes expresaron su preferencia por la herramienta con TUI. Además, se observó que los estudiantes más pequeños prefirieron las TUI sobre las gráficas en todos los casos, mientras que en estudiantes mayores las preferencias se dividieron y pudo notarse una relación con el género. En el segundo de los artículos, Kwon12, el estudio involucró estudiantes de primaria de primer grado que fueron divididos en dos grupos: experimental y de control. El grupo experimental recibió A-bricks (tangible) y el grupo de control recibió Scratch (gráfica) como herramientas de programación. A-bricks está compuesto por un robot que tiene dos ruedas que funcionan con motores paso a paso y un sensor led que permite detectar una línea negra. El robot puede avanzar, girar hacia la izquierda/derecha, entre otras funciones. Así mismo su programación se realiza a través de ladrillos que pueden ser conectados horizontalmente o apilados entre sí. Cuenta con ladrillos que permiten indicar el inicio/final de una repetición. A partir de las experiencias realizadas se obtuvo que A-bricks resultó

ser eficiente para permitir que las y los estudiantes puedan manifestar habilidades de pensamiento lógico y resuelvan problemas por su cuenta. Además, se observó que A-bricks es una herramienta eficaz para las primeras etapas del aprendizaje de la programación.

Caceres18 presenta una interfaz de programación tangible denominada FYO (Follow Your Objective - Siga su objetivo) que está compuesto por un tablero programable, bloques basados en rompecabezas y un robot móvil como intérprete. Las y los estudiantes pueden experimentar la programación básica, sintaxis y depuración al mismo tiempo que juegan durante el proceso. La sintaxis se encuentra definida por bloques de comando (avanzar, girar izquierda/derecha e ir a la función), bloques de parámetros (uno, dos, tres, F1, entre otros) y bloques de definición. El tablero cuenta con un botón "Play" en la parte superior izquierda que al pulsarlo comienza a interpretar el código y lo envía al robot mediante una comunicación inalámbrica. FYO fue implementado con estudiantes de entre cinco y ocho años de edad para realizar un estudio exploratorio de su usabilidad como una plataforma de programación tangible. Los resultados obtenidos demostraron que los juegos conocidos como rompecabezas son importantes porque pueden interactuar intuitivamente con la plataforma y jugar mientras aprenden. Los autores mencionan que, si bien algunos participantes necesitaron más tiempo que el resto para programar el comportamiento del robot pudo notarse que con la práctica lograron mejorar dicho tiempo. En esta misma línea, Kakehashi14 presenta P-CUBE que está pensado para que personas con discapacidad visual puedan contar con una herramienta en el inicio de la programación. P-CUBE está compuesto por un robot móvil, una caja de programación, bloques de programación y una computadora. Los bloques que se deben colocar dentro de la caja se identifican por etiquetas de radiofrecuencia, están pensados para ser reconocidos a través de información táctil. Hay cuatro tipos de bloques: movimiento, temporizador, condicional y repetición. Así mismo el robot puede avanzar o retroceder y emite sonidos diferentes cuando gira hacia la derecha o izquierda. El programa es transmitido hacia la computadora y posteriormente se transmite al robot. P-CUBE se puso a prueba en un taller para personas con discapacidad visual de secundario y bachillerato. En el grupo, tres de los integrantes tenían disminución visual grave y uno de ellos era ciego. Además, ninguno tenía conocimientos previos sobre programación. Quienes participaron del taller pudieron completar las tareas propuestas: realizar un programa secuencial y un rastreador de línea. El tiempo implementado para desarrollar las tareas fue de 90 min. A partir de esta experiencia se han realizado mejoras en los cubos dado que algunos eran difíciles de identificar y además se modificó la transmisión del programa para permitir que se ejecute en el robot directamente, ya que quienes participaron no querían depender de una persona para completar la ejecución de los programas. El entorno permite una mayor atención al momento de desarrollar los algoritmos al no tener que realizar operaciones sobre una computadora que les pueda ocasionar problemas. Así mismo se cree que será fácil de usar por personas principiantes en programación.

A partir de la información recabada se obtiene que 3 de los entornos de programación han sido pensados para estudiantes de primaria (5 a 13 años). Todos hacen uso de un robot móvil que recibe las instrucciones y en base a ello actúa. Así mismo tres de ellos hacen uso de una "caja maestra o tapete de programación". En Kwon12 y Kakehashi14 se concluye que los entornos propuestos pueden ser fáciles de utilizar para principiantes en programación. El lenguaje A-bricks resultó ser eficiente para que puedan manifestar habilidades de pensamiento lógico y resuelvan problemas por su cuenta. Esta idea se refuerza con la aplicación FYO al estar sus bloques basados en rompecabezas y resultar intuitivo y con P-CUBE que por no tener que realizar operaciones sobre una computadora logran una mejor atención de los participantes (ver Tabla 2).

Id	País	Destinatario	Recurso físico	Recurso lógico	Experiencia
Sapounidis19	Grecia	6 a 13 años	Bloques; PC; Robot Lego NXTt; Caja maestra; Bluetooth	Base de datos	Comparación de V-ProRob (gráfica) con T-ProRob (tangible)
Kwon12	Corea del Sur	5 a 7 años	Robot; Sensor led; Ladrillos	Scratch	Comparación de A-bricks (tangible) con Scratch
Caceres18	Perú	5 a 8 años	Robot; Tapete de programación; Bloques basados en rompecabezas; Bluetooth	No se especifica	Estudio exploratorio de su usabilidad como una plataforma de programación tangible, FYO
Kakehashi14	Japón	Estudiantes de secundaria y bachillerato	Robot móvil; Caja de programación; Bloques; PC: RFID; Arduino	No se especifica	El entorno de programación tangible se puso a prueba en un taller para personas con discapacidad visual

Tabla 2: Resumen de los criterios de análisis.

5. Conclusiones y trabajo futuro

En este artículo se presentaron los avances de investigación sobre el estado del arte en el uso de interfaces de programación tangible en el contexto educativo. A partir de la información obtenida la programación tangible se presenta como una alternativa de aprendizaje para la programación. En el estudio se observó que en 3 de los trabajos encontrados los destinatarios fueron niños de 5 a 13 años, mientras que en 1 de ellos los destinatarios eran adolescentes de secundaria y bachillerato. Además, 2 de los trabajos realizaron comparaciones entre aplicaciones de programación gráfica y tangible, obteniendo en ambos casos resultados que evidencian preferencias y aspectos de interés de cada una. En cuanto a las características de la programación tangible mencionadas en las experiencias analizadas se destacan: promover que el foco de atención se encuentre en la actividad a resolver, puesto que permite que el interés de los participantes no se encuentre afectado por el uso de la computadora; y posibilitar el desarrollo de propuestas de aspecto lúdico para jugar mientras se aprende. Otro aspecto que se observa de las aplicaciones de programación tangible es que al utilizar objetos físicos conocidos, pueden recurrir a que el usuario interactúe de manera intuitiva con la aplicación, como en el caso las piezas de rompecabezas en la plataforma de programación FYO. Es de importancia aclarar que para que estas consideraciones pasen a ser afirmaciones son necesarias más pruebas empíricas. Se puede notar que con la mayoría de los lenguajes analizados en este estudio solo se han realizado pruebas piloto para ver su desempeño y usabilidad. Se destaca P-CUBE por ser pensado para que personas con discapacidad visual puedan contar con una herramienta en el inicio de la programación.

Los resultados de este estudio se limitan al análisis de dos bases de datos. Se planea a futuro continuar con esta

investigación incluyendo otras fuentes bibliográficas con el fin de abarcar una mayor cantidad de estudios.

Bibliografía

Bers, M. U., y Horn, M. S. (2010). Tangible programming in early childhood. High-tech tots: Childhood in a digital world, 49, 49–70. https://doi.org/10.1109/TE.2012.2190071.

Caceres, P. C., Venero, R. P., y Cordova, F. C. (2018). Tangible programming mechatronic interface for basic induction in programming. 2018 IEEE Global Engineering Education Conference (EDUCON), 183–190. https://doi.org/10.1109/EDUCON.2018.8363226.

Resolución CFE N. 343/18, Pub. L. No. 343 (2018).https://www.argentina.gob.ar/sites/default/files/res_cfe_343_18_0.pdf.

Horn, M. S., y Jacob, R. J. K. (2007). Designing tangible programming languages for classroom use. Proceedings of the 1st international conference on Tangible and embedded interaction, 159–162. https://doi.org/10.1145/1226969.1227003.

Ishii, H., y Ullmer, B. (1997). Tangible bits: Towards seamless interfaces between people, bits and atoms. Proceedings of the ACM SIGCHI Conference on Human factors in computing systems, 234–241. https://doi.org/10.1145/258549.258715.

Jacob, R. J. K. (s.f.). What is the next generation of human-computer interaction? | CHI '06 Extended Abstracts on Human Factors in Computing Systems (world). Recuperado 17 de diciembre de 2020, de https://dl.acm.org/doi/abs/10.1145/1125451.1125768.

Julià, C. F., Gallardo, D., y Jordà, S. (2009). TurTan: Un Lenguaje de Programación Tangible. 10.

Kakehashi, S., Motoyoshi, T., Koyanagi, K., Oshima, T., Masuta, H., y Kawakami, H. (2014). Improvement of P-CUBE: Algorithm education tool for visually impaired persons. 2014 IEEE Symposium on Robotic Intelligence in Informationally Structured Space (RiiSS), 1-6.https://doi.org/10.1109/RIISS.2014.7009180.

Kitchenham, B., Pearl Brereton, O., Budgen, D., Turner, M., Bailey, J., y Linkman, S. (2009). Systematic literature reviews in software engineering - A systematic literature review. Information and Software Technology, 51(1), 7–15. https://doi.org/10.1016/j.infsof.2008.09.009.

Kwon, D.-Y., Kim, H.-S., Shim, J.-K., y Lee, W.-G. (2012). Algorithmic Bricks: A Tangible Robot Programming Tool for Elementary School Students. IEEE Transactions on Education, 55(4), 474–479. https://doi.org/10.1109/TE.2012.2190071.

Marshall, P., Price, S., y Rogers, Y. (2003). Conceptualising tangibles to support learning. Proceedings of the 2003 conference on Interaction design and children, 101–109. https://doi.org/10.1145/953536.953551.

Marshall, P., Rogers, Y., y Hornecker, E. (2007). Are tangible interfaces really any better than other kinds of interfaces? CHI'07 workshop on Tangible User Interfaces in Context & Theory, San Jose, California, USA. http://www.cl.cam.ac.uk/conference/tangibleinterfaces/.

McNerney, T. S. (2004). From turtles to Tangible Programming Bricks: Explorations in physical language design. Personal and Ubiquitous Computing, 8(5), 326–337. https://doi.org/10.1007/s00779-004-0295-6.

Price, S. (2008). A representation approach to conceptualizing tangible learning environments. Proceedings of the 2nd international conference on Tangible and embedded interaction, 151–158.https://doi.org/10.1145/1347390.1347425.

Sapounidis, T., Stamovlasis, D., y Demetriadis, S. (2019). Latent Class Modeling of Children's Preference Profiles on Tangible and Graphical Robot Programming. IEEE Transactions on Education, 62(2), 127–133. https://doi.org/10.1109/TE.2018.2876363.

Suzuki, H., y Kato, H. (1993, August). AlgoBlock: a tangible programming language, a tool for collaborative learning. In Proceedings of 4th European Logo Conference (pp. 297–303).

CTFs en escuelas: una plataforma para acercar la ciberseguridad a la educación secundaria

Gabriela Suárez*
gabisuarez04@gmail.com
UNLP

Patricio Bolino*
patriciobolino@gmail.com
UNLP

Jeremías Pretto*
jpretto@linti.unlp.edu.ar
UNLP

Paula Venosa*
pvenosa@info.unlp.edu.ar
UNLP

Claudia Queiruga*
claudiaq@info.unlp.edu.ar
UNLP

Resumen

Actualmente, la **ciberseguridad** se identifica como un área de conocimiento dentro de la Informática estrechamente vinculada a la protección de la información, de los servicios y las comunicaciones. A su vez, se relaciona con amenazas que ocurren en Internet como el robo de información, la suplantación de identidad, el fraude digital, entre otras; como así también con problemas sociales que afectan particularmente a los jóvenes como el grooming y el sexting cuando usan redes sociales y juegos en red. El trabajo aquí presentado se inscribe en las actividades desarrolladas en el proyecto de extensión "Extensión en vínculo con escuelas secundarias" de la Facultad de Informática de la UNLP, acreditado desde 2015. El objetivo de este proyecto es **dar a conocer la Informática** como campo de conocimiento fuertemente vinculado al desarrollo socio-productivo y al campo laboral. A su vez desde el proyecto se reconoce que actualmente, en general, el abordaje de la Informática en la escuela secundaria es instrumental, ligado al uso de programas y herramientas informáticas y no a la construcción de conocimiento digital. En este sentido, podemos afirmar que la construcción de una ciudadanía digital es claramente un área de vacancia en la formación de nuestros jóvenes y es en este sentido que el abordaje de la **ciberseguridad** cobra relevancia en la escuela secundaria. El propósito de este trabajo es compartir las experiencias de realización de competencias de ciberseguridad, conocidos como Capture The Flag (CTF), desarrollados con la plataforma "CTF en escuelas secundarias". De las mismas participaron estudiantes y docentes de escuelas secundarias de gestión pública de La Plata y Berisso, con la intención de acercar a los jóvenes a la ciberseguridad e interpelar sus prácticas en las redes sociales y en Internet en general.

Palabras clave: Ciberseguridad, ciudadanía digital, gamificación, aprendizaje basado en problemas, escuela secundaria.

*LINTI, Facultad de Informática, UNLP, Argentina

1. Introducción

Varios autores y reportes internacionales y nacionales (Bell, Andreae y Robins, 2012; Bocconi, Chioccariello, Dettori, Ferrari y Engelhardt, 2016; Bonello y Schapachnik, 2020; K–12 Computer Science Framework, 2016; The Royal Society, 2012; Fundación Sadosky, 2016) señalan la relevancia de incorporar la enseñanza de la Informática, como campo del saber, en los sistemas de educación obligatoria. De esta manera se atiende a la formación de ciudadanos que comprendan e intervengan activamente en la llamada "Sociedad del conocimiento"[1], caracterizada por el acelerado avance científico-tecnológico y su impacto en los sistemas productivos. En esta sociedad las tecnologías digitales cobran un rol central, transformando la mayoría de las actividades de creación humana. En nuestro país, múltiples políticas públicas se han desplegado a lo largo de las últimas décadas con la intencionalidad de dar respuesta a la formación de ciudadanos para esta nueva sociedad. El dominio de las tecnologías digitales se vincula fuertemente con dinámicas de inclusión/exclusión social relacionadas con el acceso a la información y al conocimiento, y la capacidad de saber usarlas, comprenderlas e intervenirlas se torna cada vez más relevante para la participación en la vida social, económica y política. Ejemplo de ello son el programa "Conectar Igualdad", el "Plan Nacional de Telecomunicaciones Argentina Conectada" y más recientemente el proyecto "Program.AR" (Program.ar, sf), el programa "Aprender Conectados" (Decreto Nacional 386/2018 Creación del Plan Aprender Conectados, sf) y el "Plan Federal Juana Manso" (Plan Federal Juana Manso, sf). En el plano normativo, el Consejo Federal de Educación (CFE) mediante las resoluciones de 2015 y 2018 (CFE Nº 263/15 y CFE 343/18) puso el acento en la educación digital, en la programación y en el pensamiento computacional, en la escolaridad obligatoria.

En este contexto, el equipo del proyecto de extensión "Extensión en vínculo con escuelas secundarias" se pregunta **cómo acercar la informática a la escuela secundaria, con qué didáctica, con qué materiales, y cómo aproximar a los jóvenes a la ciberseguridad.**

La **gamificación** es un tipo de aprendizaje que ha ganado terreno en las metodologías de formación dado que facilita la interiorización de conocimientos de una forma entretenida, genera una experiencia de aprendizaje positiva, e involucra la utilización de juegos para motivar a los estudiantes. Esto se puede llevar a cabo mediante diferentes actividades, como la resolución de problemas, el alcance de objetivos, el trabajo en equipo, entre otras. Leune y Salvatore Petrilli (2017) señalan que uno de los motivos por los cuales los estudiantes no se involucran en el aprendizaje es el distanciamiento entre las actividades propuestas en el aula y su aplicación en la vida real. La gamificación enfoca a los estudiantes en el contenido relevante, proporciona retroalimentación, mejora la retención y soporta múltiples estilos de aprendizaje.

Howard Barrows (1986) define el **aprendizaje basado en problemas** como un método de aprendizaje basado en el principio de usar problemas auténticos y complejos como punto de partida para la adquisición e integración de los nuevos conocimientos. Dicho autor señala de esta manera las características fundamentales del aprendizaje basado en problemas: *"El aprendizaje está centrado en el alumno; El aprendizaje se produce en pequeños grupos; Los profesores son facilitadores o guías de este proceso; Los problemas son el foco y estímulo para el aprendizaje; Los problemas son un vehículo para el desarrollo de habilidades de resolución de problemas; La nueva información se adquiere a*

[1]La etapa de la sociedad que estamos transitando es llamada de diferente manera, Sociedad del Conocimiento, Sociedad Postindustrial, Sociedades del Conocimiento (UNESCO), Sociedad Informacional (Manuel Castells), centrada en modelo productivo basado en el conocimiento científico-tecnológico y el papel preponderante de lo social, político y cultural en la construcción de dicho conocimiento.

través del aprendizaje autodirigido".

Existen experiencias de **metodologías alternativas de enseñanza** que se están aplicando a nivel global, entre ellas podemos mencionar a la Red Global de Aprendizajes (Red Global de Aprendizajes, s.f.) que es una iniciativa de colaboración internacional que integra nuevas pedagogías de aprendizaje a través de un marco común de acciones e investigación entre los países participantes. Esta metodología busca que los estudiantes aprendan en torno a proyectos vinculados a experiencias de la vida real en diferentes ambientes, no solo en las aulas, sino también en el patio, el barrio y en espacios virtuales de acceso al conocimiento. La red tiene por finalidad que los estudiantes desarrollen la creatividad, el pensamiento crítico, el carácter, la comunicación, la colaboración y la ciudadanía. Uruguay, a través de la Administración Nacional de Educación Pública y el Plan Ceibal, participa de esta red junto con Australia, Canadá, Estados Unidos, Finlandia, Países Bajos, Nueva Zelanda, Hong Kong, Japón y Taiwán.

Los antecedentes de aplicación de metodologías alternativas de enseñanza, los principios del aprendizaje basado en problemas y las técnicas de gamificación como enfoques de la enseñanza y aprendizaje que favorecen la apropiación de conocimientos, sirvieron de motivación para proponer la realización de torneos de ciberseguridad del estilo CTF. Estos torneos son implementados utilizando la plataforma on-line "CTF en la escuela secundaria", con la intención de que sea utilizada como recurso educativo en las aulas de las escuelas para acercar conocimientos sobre ciberseguridad.

2. CTF en escuelas secundarias: ¿qué es? ¿por qué?

El aumento del teletrabajo durante el aislamiento obligatorio ha conducido de manera inadvertida al incremento del ciberdelito (Pwc Argentina, 6 de Agosto del 2020). Tanto las organizaciones como la ciudadanía se encuentran expuestas a ataques dirigidos a sus sistemas digitales que pueden impactar fuertemente en su economía, reputación y en la integridad de la información que dominan. Ejemplo de ello es el hackeo al servicio de correo electrónico de Microsoft donde se han robado datos de aproximadamente 60.000 organizaciones (BBC News Mundo, 9 de Marzo del 2021). Como consecuencia de esta situación se ha incrementado la demanda de profesionales especializados en seguridad informática (Pwc Argentina, 6 de Agosto del 2020). Esta problemática interpela a toda la sociedad en su conjunto y dentro de ella al campo educativo, en particular a la escuela y a la formación de jóvenes informados en temas de ciberseguridad que puedan participar activamente en debates públicos sobre estos temas y que a la vez se transformen en usuarios críticos.

Se conoce como CTF o "Capture The Flag" a las competencias de seguridad informática cuya finalidad es fomentar el desarrollo y formación profesional en esta área de conocimiento, en un entorno estrechamente vinculado con lo lúdico, con el aprendizaje basado en la resolución de problemas de ciberseguridad y con el trabajo en equipo. Los premios, el esfuerzo y el trabajo colectivo, caracterizan a las competencias CTF. Actualmente se desarrollan eventos a nivel nacional e internacional, como la Ekoparty[2] y la DEFCON[3], en los cuáles es común el desarrollo de CTF como técnica de formación profesional en ciberseguridad y la promoción de comunidades en torno a este tema.

Existen dos modalidades de desarrollo competencias CTF (CTFtime team, s.f.): (i) los CTFs del tipo jeopardy y (ii) los CTFs de ataque-defensa. Los CTFs de tipo jeopardy se basan en desafíos que al resolverlos entregan un texto

[2]**Ekoparty** es una conferencia anual de seguridad informática que se realiza desde 2001 en la Ciudad Autónoma de Buenos Aires que, por sus características únicas y su particular estilo, se ha convertido en un referente para toda Latinoamérica.

[3]**DEFCON** es una de las convenciones de hackers más antiguas que se desarrolla en la ciudad de Las Vegas en Estados Unidos desde el año 1993.

secreto que se denomina bandera o flag. Este flag no es más que un texto en un formato específico, y el descubrirlo implica la resolución del desafío. Los desafíos son sobre distintas temáticas de ciberseguridad como pueden ser criptografía, esteganografía, seguridad web, etcétera (Pablos, 2014). Cuando un participante resuelve un desafío se incrementa el puntaje total del equipo al cual pertenece. El juego tiene un horario de comienzo y uno de finalización y al llegar al final el equipo ganador será el que acumule la mayor cantidad de puntos. Mientras que en los CTFs de ataque-defensa cada equipo debe defender los servicios propios (puntos de defensa) y atacar los servicios de los otros equipos (puntos de ataque). Muchos de los desafíos están directamente relacionados con problemas de seguridad actuales o contemporáneos al momento en que se desarrolla la competencia y los patrones de resolución de problemas son similares a los que se aplican en la vida real (Díaz, Venosa, Macia, Lanfranco, Sabolansky, Durante, Rubio, y Pretto, s. f.).

En Países Bajos, Estados Unidos y la India se ha empezado a utilizar una modalidad de enseñanza práctica sobre ciberseguridad en las escuelas mediante la incorporación competencias de CTFs orientadas a niños y jóvenes. Sin embargo, no se han encontrado experiencias de aplicación de CTFs en escuelas en Argentina ni en los países de América Latina más comprometidos con la ciberseguridad (International Telecommunication Union, 2021). Los CTFs foráneos encontrados poseen las siguientes limitaciones para su replicación directa en el contexto regional (Suárez y Bolino, 2020: 13-16): la mayoría de las competencias están en idioma inglés, lo cual es excluyente para aquel que no domina la lengua, los niveles de dificultad de los desafíos se incrementan de manera abrupta y exigen conocimientos técnicos, algunas competencias exigen su resolución de manera presencial o se presenta contenido de la cultura propia de cada país dificultando la resolución.

En consecuencia, los autores del presente artículo han diseñado e implementado una plataforma CTF online, "CTF en escuelas secundarias", basada en software libre, que usa la modalidad jeopardy. La misma está destinada a jóvenes, adaptada al contexto regional y puede ser instanciada para el desarrollo de diferentes competencias CTF, enfocadas en temas diversos. "CTF en escuelas secundarias" aborda las limitaciones previamente descritas y se ha puesto en práctica en el proyecto de extensión "Extensión en vínculo con escuelas secundarias".

3. La plataforma "CTF en escuelas secundarias": diseño e implementación

Luego de hacer un análisis exhaustivo de las herramientas disponibles para el desarrollo de plataformas CTF, se optó por CTFd[4] que es un proyecto desarrollado en Python, que utiliza como base de datos SQLite y Redis, usa Nginx como servidor web y cuenta con una imagen de Docker. Se ha seleccionado esta herramienta dado que ofrece la posibilidad de crear competencias de manera eficaz, está basada en software libre y código fuente abierto y en la actualidad es adoptado ampliamente por la comunidad de ciberseguridad. Entre los criterios de evaluación usados se encuentran la usabilidad, seguridad y portabilidad de la plataforma y la sencillez de instalación (Suárez y Bolino, 2020: 30-54). Otra característica que hizo a la elección de CTFd fue su capacidad de generar estadísticas tanto generales de la competencia como específicas de cada usuario o equipo participante. Estos datos cuantitativos son de mucha utilidad a la hora de evaluar el desempeño de los estudiantes ya sea en el momento en que se desarrolla la competencia brindando un monitoreo en tiempo real, como al finalizar la misma para hacer un análisis posterior.

En relación a la puesta en acción de la plataforma "CTF en escuelas secundarias", se diseñaron competencias

[4] https://ctfd.io/, https://github.com/CTFd/CTFd

orientadas a estudiantes sin conocimientos previos en computación y sin experiencia previa en competencias de seguridad informática. Como consecuencia, el contenido (que se traduce en desafíos en el vocabulario de CTF) abordó temáticas que no requieren poseer conocimientos técnicos previos para resolverlos: (1) **Open Source Intelligence (OSINT):** la inteligencia de fuentes abiertas consiste en la recolección de datos de una persona o entidad que se encuentran disponibles de forma pública, junto con su posterior análisis y uso. Con ejercicios que abarcan esta temática se pretendió que los estudiantes realicen búsquedas avanzadas para obtener resultados más precisos y que comprendan la importancia de la privacidad de la información; (2) **Ingeniería Social:** consiste en engaños que realizan personas malintencionadas para hacer que su víctima entregue datos confidenciales, como por ejemplo información de la tarjeta bancaria. Los desafíos de ingeniería social tuvieron por objetivo que los estudiantes tomen noción de las amenazas a las que se enfrentan en la vida cotidiana al momento de utilizar los dispositivos tecnológicos; (3) **Criptografía:** se basa en tomar un mensaje y pasarlo por un proceso de manera que quede expresado de forma enigmática y así garantizar que sólo el destinatario legítimo pueda comprenderlo. Se ha introducido esta temática dado que la criptografía se usa en el día a día, por ejemplo las páginas web que usan el protocolo HTTPS cifran las comunicaciones entre el cliente y el servidor de manera que si un tercero intercepta esa comunicación no puede comprenderla.

Dentro de la misma competencia se brinda material didáctico en formato de diapositiva que incluye una breve explicación introductoria a las temáticas para que los estudiantes puedan orientarse en la resolución de los desafíos. La idea de utilizar este material adicional surgió como un aporte a los CTFs convencionales que sólo ofrecen desafíos sin ningún tipo de orientación. En nuestro caso consideramos que el público objetivo son jóvenes sin formación previa y es necesario aportar guías. Otro de los lineamientos de diseño de las competencias CTF fue concebir los desafíos de manera que puedan ser resueltos con herramientas online, evitando así que los estudiantes tengan que instalarlas en sus computadoras.

Figura 1: Desafío de Criptografía - a) arriba izquierda: desafío; b) arriba derecha: acceso a las pistas; c) abajo derecha: filmina explicativa

La competencia se desarrolla en modalidad grupal y se configura con un horario de comienzo y de finalización. Una vez iniciada la competencia, los estudiantes pueden acceder a los desafíos. Todos los desafíos poseen un puntaje. Si uno de los participantes responde correctamente un desafío, el mismo se resuelve para todo su equipo, sumándole puntos al mismo. Si la respuesta es incorrecta, sólo se alerta al usuario que su respuesta no es válida, no se decrementan puntos ni se penaliza al equipo de ninguna manera. La cantidad de intentos para responder es ilimitada.

A medida que un equipo resuelve desafíos, comienzan a habilitarse otros. Los desafíos se ordenan de acuerdo a un criterio de dificultad gradual, con la intención que los estudiantes puedan acceder a los conocimientos de manera progresiva. Dentro de los desafíos es posible acceder a ayudas, que en algunos casos se canjean por puntos, este es un elemento comúnmente usado en la gamificación. El equipo ganador es aquel que posee el puntaje más alto al finalizar la competencia, y en caso de empate, el ganador es el que resolvió los ejercicios más rápidamente.

Las Figuras 1, 2 y 3 muestran algunos ejemplos de desafíos que formaron parte de las experiencias realizadas.

Figura 2: Desafío de OSINT - a) izquierda: desafío; b) derecha: filmina explicativa

Figura 3: Desafío de Ingeniería Social - a) izquierda: desafío; b) derecha: filmina explicativa

4. Las experiencias: puesta en práctica de las competencias CTF

A continuación se describen y analizan las experiencias desarrolladas de competencias de ciberseguridad con la plataforma "CTF en escuelas secundarias". Aunque inicialmente estas competencias se concibieron en modalidad presencial, la emergencia de la pandemia COVID-19 resignificó la propuesta en modalidad virtual.

4.1. Modalidad presencial: primeras experiencias

Las primeras dos competencias CTF se realizaron en el año 2019 en la Facultad de Informática de la UNLP, en modalidad presencial, y contó con la participación de estudiantes y docentes de escuelas secundarias de gestión pública de La Plata y alrededores. La Tabla 1 sistematiza las principales características de ambos encuentros.

En ambas experiencias los estudiantes comprendieron rápidamente la modalidad del juego y se encontraron motivados con la competencia. Algunas de las estrategias de motivación que se utilizaron fue proyectar en la pared

CARACTERÍSTICAS	PRIMER CTF	SEGUNDO CTF
Escuelas participantes	ES N° 14 de La Plata EEST N°5 de Villa Elvira	Liceo Víctor Mercante ES N° 14 de La Plata ES N° 50 de Tolosa
Público objetivo	Jóvenes de 15 y 16 años.	Jóvenes de 14 a 17 años.
Duración	1 hora y 20 minutos.	3 horas y un recreo de 30 minutos
Cantidad total de participantes	6 estudiantes y 1 profesora	18 estudiantes
Cantidad de equipos	3	7
Cantidad de desafíos	16	25
Rol de los desarrolladores	Mentores	Mentores

Tabla 1: Características de los CTFs de modalidad presencial

la tabla de posiciones y ofrecer premios para el primer puesto. Los desarrolladores de los CTFs adoptaron el rol de mentores durante la competencia con la intención de orientar a los estudiantes en la resolución de los desafíos. Entre las singularidades observadas, se destacan:

Primer encuentro: los estudiantes nunca habían participado en una competencia de seguridad y el tiempo para comprender los ejercicios y resolverlos, les resultó escaso. Durante el transcurso de la competencia se pudo observar que los estudiantes no lograban interpretar con facilidad los objetivos de los retos, es por eso que el rol de acompañamiento de los mentores fue central.

Segundo encuentro: se incluyeron presentaciones en los desafíos para favorecer la comprensión de los retos y la autonomía para resolverlos. Los mentores focalizaron las ayudas en los equipos que tuvieron más dificultades e iban últimos en el podio: (i) la Escuela N° 50 de Tolosa se caracteriza por estar enclavada en un barrio con vulnerabilidades sociales y varios de los estudiantes se encontraban re-escolarizándose. El rol de los mentores fue de acompañamiento durante todo el proceso para que puedan resolver algunos desafíos y no resulte en una actividad frustrante y no-inclusiva. Se pudo observar que prestaron mucha atención a las explicaciones brindadas por los mentores y que pudieron comprender los conceptos relacionándolos con situaciones de la vida diaria; (ii) uno de los equipos participantes estaba integrado por un estudiante que había participado en la primera instancia, por lo que contaba con un aprendizaje previo de la competencia aunque los desafíos a los que debía enfrentarse fueron diferentes. A este equipo se le brindó menos ayuda para lograr equidad entre los grupos.

En el segundo CTF el equipo que terminó en el primer puesto fue el que estaba integrado por el estudiante que contaba con la experiencia previa. Se pudo corroborar que para este estudiante fue mucho más fácil la resolución de los desafíos porque respondió correctamente 16 retos de los 21 resueltos por el equipo. A partir de estas observaciones, se pudo concluir que el estudiante logró incorporar el conocimiento de las temáticas y el manejo de la plataforma en su primera experiencia con el CTF.

Un aspecto interesante de la actividad fue el acompañamiento de sus docentes, quiénes colaboraron y motivaron a sus estudiantes en la resolución de los desafíos.

Para evaluar las experiencias se administraron encuestas para conocer por un lado la experiencia en el uso de la plataforma y por otro la experiencia con los contenidos. De los datos procesados se puede concluir que la plataforma fue sencilla de usar, en cuanto a la complejidad de los desafíos para el 64 % fue intermedia, y para el 36 % resultó difícil,

los temas abordados resultaron desconocidos para casi la mitad de los estudiantes y se mostraron muy entusiasmados con la posibilidad de volver a participar. Del desarrollo de estos primeros CTFs se observó mucho interés por parte de los estudiantes ya que se involucraron rápidamente en la competencia y la dificultad de los desafíos propuestos no los desanimó en ningún momento.

Podemos concluir que independientemente del resultado final de cada equipo, el rendimiento de los estudiantes fue muy bueno en ambas experiencias: en el primer encuentro el equipo que terminó en el puesto uno resolvió 13 desafíos y el que terminó en el último resolvió 9, mientras que en el segundo encuentro el equipo que terminó en el puesto uno resolvió 21 desafíos y el que terminó último resolvió 11.

4.2. Modalidad Pandemia: adaptación al contexto

En el año 2020, el cual tuvo la particularidad de la pandemia por COVID-19 y el consecuente aislamiento social preventivo y obligatorio, se llevaron a cabo 2 encuentros de forma virtual. Si bien la idea original fue implementar la actividad en el formato presencial y se pensó la posibilidad de adaptarla a la virtualidad en un futuro como ocurre con las competencias CTF orientadas a profesionales de la seguridad, el contexto del 2020 obligó a resignificar la propuesta para llevar adelante las competencias y sostener el trabajo con las escuelas. Entre las adaptaciones realizadas se encuentran: (i) la incorporación de imágenes explicativas o pistas que fueran más precisas a la hora de servir de ayuda para los estudiantes, dado que los mentores participaron desde sus hogares y el apoyo a los estudiantes sería diferente que dentro del aula y (ii) la definición de formas alternativas de comunicación para saldar las dudas de los estudiantes. En la primera competencia se les dió más protagonismo a los docentes para que puedan acompañar a sus estudiantes y los desarrolladores del CTF únicamente brindaron soporte técnico. En la segunda competencia cada equipo tuvo asignado una sala de videollamada para que los estudiantes puedan organizarse y conocerse, un profesor para darles un acompañamiento pedagógico en el contexto de virtualidad y un mentor para evacuar todo tipo dudas que tuvieran. La Tabla 2 sistematiza las principales características de ambos encuentros.

CARACTERÍSTICAS	PRIMER CTF: MUJERES EN LAS TICS	SEGUNDO CTF
Escuelas participantes	EEST N°1 “Gral San Martin”de Berisso Escuela de Educación Técnica N°5 Secundario Nuestra Señora del Valle.	ES N° 50 de Tolosa Estudiantes becados por la embajada de EEUU
Público objetivo	Jóvenes de 15 a 18 años.	Jóvenes de 13 a 17 años.
Duración	2 horas	2 horas y 30 minutos
Cantidad total de participantes	21 estudiantes	20 estudiantes y 5 profesores
Cantidad de equipos	7	4
Cantidad de desafíos	23	28
Rol de los desarrolladores	Soporte técnico	Mentores remotos

Tabla 2: Características de los CTFs de modalidad pandemia

La primera competencia en modo pandemia se desarrolló en abril de 2020 en el marco del día mundial de las

Mujeres en las TIC. Para esta ocasión, al tratarse de este evento tan particular, se decidió adaptar los desafíos con contenido relacionado a las mujeres que formaron parte de la historia de la informática y de aquellas que actualmente se desempeñan en el campo de la ciberseguridad. En la búsqueda de contenido para armar estos desafíos se encontró una barrera que tiene que ver tanto con la escasa participación de mujeres en el área de las ciberseguridad cómo por la poca difusión de información de mujeres destacadas en el campo de la informática en décadas previas a la expansión de las redes sociales y el activismo en las mismas. Pese a estos obstáculos, se pudieron desarrollar varios desafíos interesantes cuyas respuestas se relacionaron con datos relevantes a las actividades de las mujeres que forman parte de la comunidad de ciberseguridad por ejemplo nombres de libros, programas, proyectos o conferencias. Esta competencia fue difundida por redes sociales y la participación se encontró abierta a cualquier estudiante sin importar su género.

El segundo CTF virtual se desarrolló en octubre del 2020 y contó con una dificultad extra para el equipo organizador: los jóvenes participantes provenían de dos mundos completamente diferentes. Por un lado participaron estudiantes becados del "Programa Space Camp" de la Embajada de Estados Unidos que concurren a escuelas de nuestro país distribuidas en diferentes lugares y por el otro estudiantes de la Escuela N° 50 de La Plata, que describimos previamente y está caracterizada por ser una comunidad socialmente vulnerable. Esta diversidad nos planteó re-pensar la organización de los equipos y decidimos organizar grupos integrados por estudiantes de ambas instituciones y promover la colaboración.

El resultado final de ambas experiencias virtuales fue muy bueno y gratificante, se pudo percibir que los estudiantes disfrutaron al participar e interactuaron muy bien entre ellos para poder resolver los desafíos.

A continuación, en la Tabla 3, se sistematiza la performance de los equipos participantes del segundo encuentro virtual que desde el punto de vista de los autores fue la experiencia más enriquecedora dado que fue la competencia más concurrida y donde la plataforma se encontró en su mayor grado de madurez.

Porcentaje de Equipos que respondieron correctamente	Cantidad de ejercicios resueltos	Grado de dificultad de los desafíos
100 % (4 equipos)	15 (53.57 % del total)	Baja
75 % (3 equipos)	8 (28.57 % del total)	Media
50 % (2 equipos)	5 (17.85 % del total)	Alta

Tabla 3: Performance segundo encuentro virtual

Como se puede observar en la tabla, los desafíos de baja dificultad introductorios en las distintas categorías pudieron ser resueltos por el total de los equipos participantes y a medida que la dificultad de los desafíos se incrementaba, la cantidad de equipos que pudieron resolverlos disminuyó.

5. Conclusiones

A partir de las experiencias de uso de la plataforma "CTF en escuelas secundarias" y los resultados obtenidos, se puede concluir que la incorporación de conocimientos técnicos del campo de la seguridad informática mediante metodologías de enseñanza vinculadas con la gamificación, el aprendizaje basado en problemas y el trabajo colaborativo favorece la apropiación de conocimientos nuevos entre estudiantes de escuelas secundarias. La realización de las competencias de ciberseguridad con dicha plataforma nos permitieron observar la receptividad e interés de los jóvenes

al interiorizarse sobre estos saberes técnicos, así como también sobre saberes vinculados con la comunicación, la organización y el trabajo colaborativo para poder alcanzar una meta común.

Estas experiencias nos permiten afirmar que este tipo de torneos pueden realizarse tanto en modalidad presencial como virtual. La adaptación a la virtualidad extendió las fronteras de las competencias posibilitando que estudiantes de diferentes provincias de la Argentina pudieran participar. De esta manera, se puede empezar a pensar en una red de aprendizaje en torno a la ciberseguridad en la escuela secundaria que aliente la participación de jóvenes, promueva vocaciones tecnológicas específicamente en seguridad informática y consolide la articulación entre la escuela y la universidad pública.

En el marco de las iniciativas y políticas educativas nacionales que promueven una alfabetización científica y tecnológica desde edades tempranas, la enseñanza de la Informática como disciplina en la escuela cobra relevancia en la construcción de una ciudadanía digital que favorece una relación estrecha, activa y fructífera con las tecnológicas digitales. Para ello, consideramos que contar con materiales educativos adecuados y pertinentes, como la plataforma "CTF en escuelas secundarias" presentada en este trabajo, puede resultar un recurso didáctico útil a la hora de incluir a la ciberseguridad. La posibilidad de adquirir conocimientos nuevos, en forma activa, a través de la propia experimentación, realizando conjeturas, investigando y trabajando colaborativamente, resulta por un lado motivador para los estudiantes y por otro, fundamentalmente favorece la apropiación de los mismos dado que se adquieren en contextos y situaciones reales y cotidianas.

Bibliografía

Ámbito Financiero. (16 de Noviembre del 2020). Hackearon Cencosud y piden millones de dólares para no revelar la información privada. Ámbito. https://www.ambito.com/informacion-general/hackers/hackeron-cencosud-y-piden-millones-dolares-no-revelar-la-informacion-privada-n5148493.. Recuperado 15/8/2021.

BBC News Mundo. (9 de Marzo del 2021). El "inusualmente agresivo" ciberataque del que Microsoft acusa a China (y por qué no es simplemente una nueva crisis de ciberseguridad) - BBC News Mundo. Bbc. Recuperado 15/8/2021. https://www.bbc.com/mundo/noticias-56299627

Bell, T., Andreae, P. y Robins A. (2012). Computer science in NZ high schools: the first year of the new standards. SIGCSE '12: Proceedings of the 43rd ACM technical symposium on Computer Science Education. February 2012. pp. 343—348.

Bocconi, S., Chioccariello, A., Dettori, G., Ferrari, A. y Engelhardt, K. (2016). Developing computational thinking in compulsory education – Implications for policy and practice; EUR 28295 EN; doi:10.2791/792158.

Bonello, M. y Schapachnik, F. (2020). Diez preguntas frecuentes (y urgentes) sobre pensamiento computacional. Virtualidad, Educación y Ciencia, 20 (11), pp. 156–167.

Barrows, H.S. (1986). Taxonomy of problem-based learning methods. Medical education: 20(6).

CTFtime team. (s.f.). What is Capture The Flag?. Ctftime. https://ctftime.org/ctf-wtf/.

Díaz, J., Venosa, P., Macia, N., Lanfranco, E., Sabolansky, A., Durante, M., Rubio, D. Pretto, J. (s. f.). Participación y despliegue de CTFs como herramienta para fortalecer la formación en ciberseguridad. Sedici.http://sedici.unlp.edu.ar/handle/10915/104050

Decreto Nacional 386/2018 Creación del Plan Aprender Conectados (sf). https://siteal.iiep.unesco.org/bdnp/2279/decreto-nacional-3862018-creacion-plan-aprender-conectados

Echeveste, M. y Martínez, C. (2016). Desafíos en la enseñanza de Ciencias de la Computación. Virtualidad, Educación y Ciencia, 12 (7), pp. 34–48.

Fundación Sadosky (2016). Una propuesta para refundar la enseñanza de la computación en las escuelas Argentinas.

International Telecommunication Union. (2021). Global Cybersecurity Index 2020 Measuring commitment to cybersecurity. Ginebra, Suiza, 2021. https://www.itu.int/dms_pub/itu-d/opb/str/D-STR-GCI.01-2021-PDF-E.pdf

K–12 Computer Science Framework. (2016). Retrieved from http://www.k12cs.org.

Leune K. y Petrilli S. (2017). Using Capture-the-Flag to Enhance the Effectiveness of Cybersecurity Education. SIGITE '17: Proceedings of the 18th Annual Conference on Information Technology Education. September 2017. pp. 47-–52.

Pablos, R. (26 de Febrero de 2014). CTF: Entrenamiento en seguridad informática. Incibe-cert. https://www.incibe-cert.es/blog/ctf-entrenamiento-seguridad-informatica

Plan Federal Juana Manso (sf). https://www.argentina.gob.ar/educacion/juana-manso.

Pwc Argentina. (6 de Agosto del 2020). Al ritmo de la pandemia, aumenta la demanda de profesionales y capacitaciones en ciberseguridad. pwc. lhttps://www.pwc.com.ar/es/prensa/al-ritmo-de-la-pandemia-aumenta-demanda-de-profesionales-en-ciberseguridad.html. Recuperado 20/8/2021.

Red Global de Aprendizajes. (s.f.). Red Global de Aprendizajes. Administración Nacional de Educación Pública y Plan Ceibal. https://redglobal.edu.uy/. Recuperado 15/8/2021.

Suárez, G. y Bolino, P. (2020). "Capture the Flag" aplicada a la enseñanza de ciberseguridad en escuelas. Sedici. http://sedici.unlp.edu.ar/handle/10915/118187.

The Royal Society. (13 January 2012). Shut down or restart? The way forward for computing in UK schools. Issued: January 2012 DES2448.

SESIÓN 3:
ENTORNOS DE PROGRAMACIÓN

Moderador: *Dr. Pablo G. Turjanski (UBA, CONICET)*

Entornos programables para la modelización basada en agentes en educación: una revisión
Cristián Rizzi Iribarren

Arduino en la Escuela: una herramienta versátil para la enseñanza de programación y robótica
Martín Lobos, Silvia Bast y Gustavo Astudillo

Diseño de una arquitectura extensible y escalable para refundar el Proyecto Gobstones
Alan Rodas Bonjour y Federico Agustín Sawady O'Connor

¿Scratch, Python, o qué? Criterios para elegir un entorno para enseñar a programar a principiantes
Marcos J. Gómez y Pablo E. "Fidel" Martínez López

Entornos programables para la modelización basada en agentes en educación: una revisión

Cristián Rizzi Iribarren*
crizzi@udesa.edu.ar
Universidad de San Andrés

Resumen

El LOGO, creado por Seymour Papert en la década de 1960, fue el primer lenguaje de programación de computadoras para niños y niñas en edad escolar. Si hoy podemos hablar de tecnología[1] en la educación, en gran parte es gracias al LOGO. Pero el LOGO no fue solamente un lenguaje de programación, sino que fue parte de un marco más amplio de ideas pedagógicas, tecnológicas y educativas. Aún después de medio siglo, con tantos cambios tecnológicos (comenzando por la aparición de Internet), el espíritu del LOGO sigue presente en varios de sus herederos. StarLogo y NetLogo, son dos de los descendientes directos del LOGO de Papert, en su estructura y también en su filosofía, pero con una particularidad: son entornos programables de **modelización[2] basada en agentes** (MBA de aquí en adelante).

En este artículo se resume la historia de estos entornos programables de modelización basada en agentes (EPMBA); se detallan sus características más salientes; y se esbozan sus potencialidades en la educación en ciencias.

Se parte desde los inicios del lenguaje LOGO y la evolución de sus descendientes, tanto en sus capacidades como en cuanto a la interfaz de usuario. Se introduce la modelización basada en agentes como un abordaje nuevo de estos entornos y se esbozan algunas de las características que los hacen interesantes para la integración curricular en el área de las disciplinas STEM/CTIM en el marco del pensamiento computacional aplicado.

Palabras clave: Modelización, Simulación, Programación, Agentes, LOGO.

*Universidad de San Andrés, Buenos Aires, Argentina.

[1] Se utiliza aquí el término genérico tecnología para abarcar a todos los dispositivos basados en computadoras, desde las computadoras mismas hasta dispositivos móviles como netbooks, celulares o tabletas, o incluso placas robóticas, internet, etc.

[2] A pesar de que muchas veces son utilizados como sinónimos, se utilizará aquí el término modelización y no modelado, para distinguirlos. Al hablar de modelización, nos referimos a un término más relacionado con el mundo lógico, mientras que la noción de modelado la reservamos para el mundo físico. Por ejemplo, al referirnos a la tarea de esculpir una cabeza humana en arcilla, lo haríamos en términos de modelado, y no de modelización.

1. La génesis del lenguaje LOGO

En el año 1968, Seymour Papert, co-director del Laboratorio de Inteligencia Artificial del Instituto de Tecnología de Massachusetts (M.I.T) en ese entonces, creó el primer lenguaje de programación de computadoras pensado específicamente para la educación: el LOGO.

El lenguaje LOGO utilizó como metáfora una tortuga en honor al neurofisiólogo británico William Grey Walter, quien había construido unas tortugas cibernéticas (Elsie y Elmer), las cuales exhibían patrones de comportamiento compatibles con un organismo vivo.

Luego de 10 años de trabajo con el LOGO, Papert (2001) escribió su famoso libro *"Desafío a la mente"*, en el cual habla en extenso sobre diferentes maneras en que una computadora puede ser utilizada para aprender.

En cierta forma, *Desafío a la mente*, puede entenderse como una amplificación de un artículo que escribió Papert junto a Cynthia Solomon en 1971 titulado *Twenty things to do with a computer* (Veinte cosas para hacer con una computadora).

Si pensamos en cómo era la tecnología computacional en 1971, y leemos el artículo de Papert y Solomon, es imposible no sorprendernos de la claridad que tenían sobre la potencialidad de estas tecnologías, imaginando proyectos, actividades, desafíos, que siguen vigentes, aún después de transcurrido medio siglo.

Entre las veinte cosas que describe el artículo, la número 3 se denomina *Turtle Biology* (Biología de la Tortuga). En los primeros párrafos del artículo se lee lo siguiente: "Para hacer a nuestra tortuga más parecida a una criatura viviente, debemos dotarla de patrones de comportamiento. Esto implica usar órganos sensoriales." (Papert y Solomon, 1971, p.9).

Esos órganos sensoriales se traducían en sensores, para que la tortuga reconociera líneas u objetos, y en base a esa detección realizara determinadas acciones. Este punto, de alguna manera, es la base de la modelización basada en agentes, tal la naturaleza de los entornos programables que aquí se presentan.

El lenguaje LOGO de los 70 tuvo un gran protagonismo en las aulas[3] en los 80' y parte de los 90', y luego fue decayendo en su interés a manos de nuevas herramientas, principalmente las ofimáticas y algunos "juegos didácticos". De todos modos, el LOGO siempre conservó su grupo de cultores, hasta el día de hoy inclusive.

Esa pérdida de interés en el LOGO como recurso didáctico en los 90', creó una suerte de brecha pedagógico-digital, donde una parte creía que no todos los chicos y chicas debían saber programar. Parte de esto tuvo que ver con que las "ideas poderosas" que Papert había vislumbrado para el LOGO como herramienta integrada fueron interpretadas, en algunos casos, como enseñanza de la programación como fin último, resultando en actividades que no tenían mucho sentido para niños y niñas, como por ejemplo generar una lista de números primos o dibujar una casa con líneas, además de que la sintaxis de los lenguajes era difícil para algunos (Resnick, 2009).

También sucedió algo que Papert (1981) señala en Desafío a la mente como la paradoja de "(...) la utilización de tecnologías nuevas para reforzar métodos educativos cuya existencia misma es un reflejo de las limitaciones del período pre-computacional" (p. 162–163).

[3]El término "aulas" está empleado acá de manera general para referir el entorno escolar-curricular, y no como ámbito físico, ya que en la mayoría de las escuelas, el trabajo con las computadoras se realizaba en un espacio específico, comúnmente llamado "laboratorio de informática".

2. De la tortuga al gato

Desde su invención hasta nuestros días, la tortuga del lenguaje LOGO fue mutando al ritmo de los cambios tecnológicos, incorporando colores, modo multi-tortuga y multimedia, entre otras características.

Uno de los saltos evolutivos de la tortuga de LOGO se produjo en Mayo de 2007, cuando Mitch Resnick, discípulo de Papert, creó Scratch, una especie de LOGO multimedia con énfasis en el desarrollo de la creatividad y la colaboración, con una particularidad: el salto evolutivo fue tan brusco que la tortuga se convirtió en un gato, tal el protagonista de Scratch. Esta metáfora viene de cuando los DJs rasguñaban los discos para crear sonidos nuevos y fue utilizada para resaltar las características de Scratch de crear a partir de las creaciones de otros (remixar).

En Scratch, a diferencia del LOGO, la programación no se hace en modo texto sino a través de bloques de diferentes colores, que exhiben muescas, las cuales permiten anticipar ciertas estructuras de programación: si dos bloques no pueden engancharse entre sí, eso significa que dicha estructura no es correcta.

Esta característica de Scratch, de programar de modo visual utilizando bloques al estilo LEGO, (la metáfora no es casual, pero no se abordará el tema en este artículo), marcó un antes y un después en los lenguajes/entornos de programación pensados para el ámbito educativo. Hoy en día, la mayoría de estas herramientas utiliza la metáfora de bloques multicolor como interfaz de programación.

3. La programación visual por bloques

Los bloques vienen a resolver algunos problemas que se dan frecuentemente en la programación, principalmente en cuanto a la **sintaxis** y a la **gramática**.

Esto es especialmente importante para el proceso de depuración[4] de los programas de computadora: en cuanto a la sintaxis, evitan que el programador escriba mal la instrucción ya que la misma está impresa en el bloque: en el LOGO original, para que la tortuga avance 10 pasos había que escribir “AV 10”, mientras que en Scratch, para que el gato avance 10 pasos, hay que escribir el número 10 en el bloque que dice “mover” (en realidad, el 10 ya viene impreso en el bloque):

Figura 1: LOGO y Scratch. Cómo lograr que el personaje (tortuga o gato) se mueva 10 pasos hacia adelante. Elaboración del autor en base a capturas de pantalla de Scratch 3.0. online. https://scratch.mit.edu

[4]Se conoce como depuración al proceso que llevan adelante los programadores cuando prueban el código que escribieron para detectar posibles fallas.

En cuanto a la gramática del lenguaje de programación, por un lado se agrupan los comandos con diferentes colores según categorías, así por ejemplo, los comandos de **movimiento** (avanzar, girar, etc.) son de color azul, mientras que los de **apariencia** (tamaño, color, ubicación, etc.) son de color violeta:

Figura 2: Algunos bloques de programación en Scratch. Nótese la diferencia en los colores. Elaboración del autor en base a capturas de pantalla de Scratch v3.0 online.https://scratch.mit.edu

Y por otro lado, algo que se mencionó anteriormente: si dos bloques no enganchan uno debajo del otro o uno dentro del otro, eso significa que la estructura es incorrecta. Incluso, hay bloques que solo tienen una muesca (en vez de dos como los de la Figura 3), y esos bloques solo aceptan bloques debajo de ellos y se los conoce como bloques de **eventos**:

Figura 3: Algunos bloques de programación en Scratch de la categoría “Eventos”. Elaboración del autor en base a capturas de pantalla de Scratch v3.0 online. https://scratch.mit.edu

El año 2012 significó un gran cambio en el status de la programación como herramienta pedagógica, ya que en paralelo con la aparición de Scratch, nació CODE.ORG, una iniciativa de dos hermanos (Hadi y Ali Partovi) para enseñar a programar a niños y niñas.

La iniciativa contó (y cuenta) con el apoyo de casi todos los gigantes del mundo de la tecnología digital, incluyendo, entre otros, a Amazon, Google, Microsoft, Facebook y Apple.

Pero no solo las empresas del sector tecnológico apoyaron esta iniciativa, sino también sus fundadores a título personal, comenzando por Mark Zuckerberg y Bill Gates, hasta el entonces presidente de EEUU, Barack Obama. Fue él mismo quien en 2014, en su discurso de la Unión, dijo que todos los niños y las niñas debían aprender a programar.

Al día de hoy, CODE.ORG ofrece en sus sitio web, de manera gratuita, numerosos recursos para aprender a programar, para niños y niñas de diferentes edades.

A partir de Scratch y CODE.ORG, la programación, como parte del pensamiento computacional, retomó su protagonismo en la arena educativa, y hoy hay cientos, quizás miles, de recursos similares que promueven la enseñanza

de las ciencias de la computación, el pensamiento computacional y la programación utilizando interfaces visuales basadas en bloques.

4. De lenguajes a entornos

El lenguaje LOGO, y otros lenguajes de programación, nacieron mucho antes de que se inventara Internet, o se fundara Google. Cuando la web 2.0 permitió que las personas colaboraran a través de Internet, los lenguajes (y muchas otras herramientas de software) se transformaron en **entornos programables**, donde se creaba una cuenta, se podía guardar sus creaciones y eventualmente poner dichas creaciones a disposición para que otras personas pudieran utilizarlas como inspiración, o para colaborar en el proceso de mejora.

Scratch nació de esta manera y es así hasta el día de hoy: una plataforma que cuenta con 59 millones de usuarios[5], donde un usuario crea una cuenta y si lo desea, puede poner sus creaciones a disposición de otros.

La plataforma Scratch permite ver cómo están programadas estas creaciones, y además ofrece la posibilidad de "remixarlas", es decir, modificarlas para dar vida a nuevas creaciones en lo que el propio Resnick llama "la espiral de la creatividad" (Resnick, 2007).

5. StarLogo, las múltiples tortugas, la computación en paralelo y los sistemas descentralizados

StarLogo es un descendiente del LOGO original, y fue creado también por Mitch Resnick, quien como ya señalamos es también el creador de Scratch, miembro del Laboratorio de Medios del MIT y actual director del proyecto Lifelong Kindergarten (LLK).

El StarLogo de Resnick se diferenciaba del LOGO original no solamente en el modo multi-tortuga, sino que además, las tortugas de StarLogo podían sensar mejor su entorno que la tortuga primigenia, algo que Papert y Solomon habían contemplado ya en su artículo de 1971 como una característica importante deseable para incorporar al LOGO en futuras versiones.

Al principio de este artículo se destacó la importancia de los sensores de la tortuga que Papert y Solomon mencionan en su artículo.

Las tortugas de StarLogo pueden sensar su entorno, detectar la presencia de otras tortugas cercanas o de otros elementos, en definitiva tienen la capacidad de "olfatear", tal como lo expresa Resnick en "Tortugas..." (p. 61).

La descentralización, el caos, la auto-organización, son conceptos muy vinculados entre sí en el estudio de los sistemas, y a Resnick le preocupaba especialmente que la gente tendiera siempre a atribuir los fenómenos a una causa primera o única, haciendo uso de lo que él llamaba "la mentalidad centralizada" (centralized mindset). Resnick lo ilustra en la obra citada (2001) con cómo la gente explica que una bandada de aves exhiba un patrón de vuelo tan armónico y bello: debe haber un líder de la bandada que oficia de líder y ordena a las demás aves cómo hacerlo.

Esta preocupación, en parte, fue uno de los grandes motores que impulsaron la idea de Resnick de crear StarLogo en el año 1991 en una Connection Machine diseñada por Dany Hillis dentro del M.I.T.

[5]Información obtenida de la wiki oficial de Scratch el 18 de octubre de 2020. Disponible en: https://en.wikipedia.org/wiki/Scratch_(programming_language)

En 1985, Hillis, graduado del M.I.T y discípulo de Marvin Minsky, en ese momento co-director del Laboratorio de Inteligencia Artificial de la prestigiosa universidad, considerado uno de los padres de esa disciplina, diseñó una computadora que rompió con el paradigma reinante en cuanto a la arquitectura de las máquinas. Hasta ese momento, las computadoras estaban construidas en base a la arquitectura de Von Neumann, donde las operaciones se realizaban en serie, es decir, una detrás de otra a la mayor velocidad posible (Brand, 1987).

La computadora que diseñó Hillis, en cambio, podía ejecutar tareas en paralelo gracias a su arquitectura de hipercubo, con 65.536 procesadores trabajando al mismo tiempo en forma de red, con cada procesador conectado con otros 16. Esta máquina, que le costó al M.I.T cuatro millones de dólares, se llamó "Connection Machine", y se la utilizó para procesos de alta demanda de datos, desde el análisis de operaciones con tarjetas de crédito (Markoff, 1994, como se cita en Wikipedia, 2020) hasta exploraciones de la industria del petróleo.

Para programar la Connection Machine se crearon versiones para lenguajes ya existentes, como LISP o C, y para distinguirlas se les añadió el prefijo "Star", por estrella en inglés. Así nacieron el C-STAR y el StarLISP, y debido a esto fue que Resnick llamó a su criatura StarLogo, como una forma de continuar con esa tradición (Resnick, 2001).

El hecho de que la Connection Machine permitiera disponer de múltiples tortugas que pudieran ejecutar tareas en paralelo, era una gran oportunidad para modelizar sistemas descentralizados, idea en la que Resnick venía trabajando desde hacía un tiempo.

Mitch Resnick creó **StarLogo** como una herramienta de MBA, para la cual alcanza con tener conocimientos de matemática básicos para modelizar sistemas complejos. Esta característica de StarLogo es la que lo hace una herramienta interesante para utilizar en el nivel escolar o incluso terciario/universitario, como una lente diferente para mirar ciertos fenómenos, como por ejemplo los ecosistemas o la propagación de epidemias.

El mundo, hoy, es más complejo que nunca (vaya si lo hemos aprendido a partir del 2020), y herramientas como StarLogo pueden ofrecer una gran oportunidad a estudiantes en el nivel escolar para explorar, analizar y quizás también predecir estos fenómenos en un mundo cada vez más complejo.

6. El pensamiento computacional

Tal como se señalara al principio de este artículo, el entusiasmo para que niños y niñas aprendan programación, dentro y fuera de la escuela, resurgió con fuerza en 2012 a partir de la aparición de Scratch, de CODE.ORG y del apoyo que figuras de renombre mundial le dieron al proyecto.

Sin embargo, hubo un hito anterior, que marcó la agenda mundial en el tema, y le dio un barniz nuevo al interés por la programación como parte del acervo cultural de una persona para desempeñarse en el siglo XXI: un artículo de tres páginas escrito por Jeanette Wing en 2006 llamado "Pensamiento computacional".

En ese momento, Wing era Profesora y Jefa del Departamento de Ciencias de la Computación en la Universidad Carnegie Mellon, Pittsburgh, PA, Estados Unidos. A partir de la publicación de ese artículo, el impulso de CODE.ORG, la aparición de Scratch y la creciente influencia de lo digital en todos los aspectos de la sociedad (de la mano de la ciencia de datos, la inteligencia artificial, la robótica y los dispositivos móviles principalmente) el **pensamiento computacional** se ha ganado un lugar de privilegio en la agenda educativa.

Este creciente interés se ha visto reflejado en la aparición de numerosos programas de capacitación, plataformas, planes de estudio y ofertas extracurriculares donde el pensamiento computacional toma diversas formas y es interpretado de múltiples maneras.

En su artículo, Wing destaca el avance de lo computacional en las disciplinas, pero mucho más allá que como meras herramientas:

> La contribución de la informática a la biología va más allá de la capacidad de buscar a través de grandes cantidades de datos de secuencias en busca de patrones. La esperanza es que las estructuras de datos y los algoritmos, nuestras abstracciones y métodos computacionales, puedan representar la estructura de las proteínas de manera que aclaren su función. La biología computacional está cambiando la forma de pensar de los biólogos.(p.33-34)

En el marco de lo que ya señalaba Wing hace quince años, es que adquiere relevancia presentar recursos didácticos que permitan a docentes y estudiantes abordar estos escenarios de manera crítica y reflexiva.

En este caso se trata de herramientas basadas en la filosofía del LOGO de Papert para promover la integración del pensamiento computacional y la programación en la enseñanza de las ciencias naturales.

En un trabajo realizado por David Weintrop (2016) junto a un grupo de colegas de la Universidad de Northwestern (entre los cuales está Uri Wilensky, creador de NetLogo), se presenta una taxonomía de prácticas de pensamiento computacional para la enseñanza de las disciplinas STEM[6], entre las cuales están **el diseño y la construcción de modelos computacionales** (entre otras).

StarLogo y NetLogo son dos entornos programables que permiten a estudiantes y docentes integrar el pensamiento computacional, al proveer una manera visual y amigable para diseñar y construir modelos computacionales (Resnick, 2001).

7. StarLogo y NetLogo: la historia de dos hermanos

StarLogo comenzó siendo un lenguaje donde la programación se realizaba en modo texto, igual que su antecesor. Ya se ha mencionado en este artículo que para indicarle a una tortuga que avanzara 10 pasos se debía escribir “forward 10” (avance 10).

StarLogo siguió evolucionando: en 1997 se convirtió en StarLogoT con la colaboración de Uri Wilensky, graduado del M.I.T y también discípulo de Papert, quien en 1999 creó, a partir de StarLogoT, otra versión: el **NetLogo**, en el *Center for Connected Learning and Computer-Based Modeling*, primero en la Universidad de Tufts y luego en la Universidad de Northwestern, en el área de Chicago. Wilensky continúa al mando del proyecto NetLogo con una actualización constante.

La primera versión de NetLogo, la 1.0, apareció en el año 2002. Al momento de escribir este artículo, la última versión es la 6.2, NetLogo se ha robustecido y modernizado, pero su interfaz sigue siendo primordialmente en dos dimensiones (2D), y la programación se realiza en modo texto y solamente en inglés. NetLogo se sigue pareciendo mucho al StarLogo original, y goza de muy buena salud, cumpliendo el precepto papertiano de “Low threshold, high ceiling” (umbral bajo, techo alto) queriendo significar que es un lenguaje sencillo, pero al mismo tiempo potente.

StarLogo continuó su desarrollo dentro del M.I.T., ahora en el *Scheller Teacher Education Program* (STEP) bajo la dirección de Eric Klopfer.

[6]STEM: Sigla de origen anglosajón que engloba las ciencias, la tecnología, la ingeniería y la matemática.

En 2001, Resnick, Klopfer y Vanesa Collella, escribieron un bellísimo libro, lleno de ilustraciones a todo color, llamado *Adventures in Modeling: Exploring Complex, Dynamic Systems with StarLogo*, compuesto por desafíos y actividades de modelización basada en agentes con StarLogo.

Durante esos años, Mitch Resnick, en ese entonces -y hasta ahora- Director del *Lifelong Kindergarten Group* (LLK), dentro del Laboratorio de Medios del M.I.T., dirigió sus esfuerzos a crear Scratch, el cual ya ha sido presentado en este artículo. El derrotero del StarLogo iba a continuar con Eric Klopfer al mando del proyecto, e iba a dar un salto cualitativo muy importante en 2008: StarLogo TNG *(The Next Generation)*, donde se abandonó la interfaz de programación en modo texto para pasar al modo de programación visual por bloques, y además a que el escenario donde ocurren las simulaciones sea en tres dimensiones (3D).

En 2012 se produjo otro salto importante en la vida de StarLogo con la aparición de la primera versión basada en la web: StarLogo NOVA con tecnología Adobe Flash[7], para luego en 2016, dar a luz a la versión actual: StarLogo NOVA 2.0 con tecnología Javascript[8].

Tanto StarLogo TNG como StarLogo NOVA cuentan con una versión en español. Si bien StarLogo TNG todavía puede bajarse desde el sitio oficial del STEP (`education.mit.edu`), esta versión no tiene más soporte y ha sido reemplazada por StarLogo NOVA.

A continuación, se muestran cuatro capturas de pantalla para ilustrar las diferentes versiones de StarLogo, desde su nacimiento, hasta nuestros días y la actual de NetLogo (la cual se parece mucho al StarLogo original). Entre paréntesis se consigna el año al cual corresponde la captura de pantalla de dicha versión:

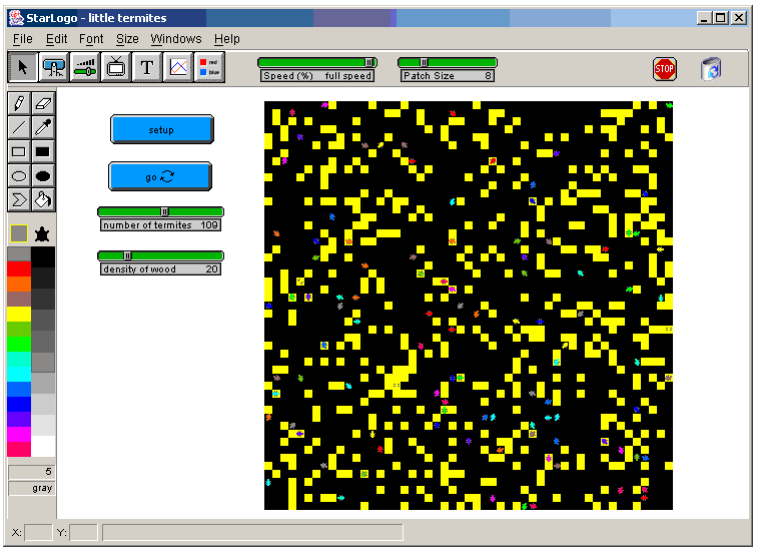

Figura 4: StarLogo original: modelo de simulación de termitas construyendo su nido (1999). MIT STEP LAB, Cambridge, MA, Estados Unidos. Recuperado de: `http://web.mit.edu/mitstep/starlogo/gettingstarted/getting_started.html`

[7] Adobe Flash (antes Macromedia Flash) fue una de las primeras tecnologías para hacer de la web un espacio multimedia e interactivo. En la actualidad, por cuestiones de seguridad, ya no es más soportado por los navegadores convencionales e incluso la propia Adobe lo ha sacado de su sitio web.

[8] Javascript es el lenguaje de la web, es la tecnología dominante en la actualidad por su seguridad y universalidad.

Figura 5: StarLogo TNG: modelo de simulación de termitas similar al anterior (2009)

Figura 6: StarLogo NOVA: modelo de simulación de un ecosistema formado por conejos, pasto y lobos (2021). Elaboración del autor. Captura de pantalla del modelo "consumidor con depredador y crecimiento de pasto". Modelo disponible en https://beta.slnova.org/rizzic/projects/349835/

Tanto **StarLogo NOVA** como **NetLogo** son dos entornos de MBA que pueden utilizarse con gran potencial en el nivel escolar, por su sencillez de uso y su versatilidad. Vale recalcar que ambos son gratuitos y que NetLogo, es además, software libre.

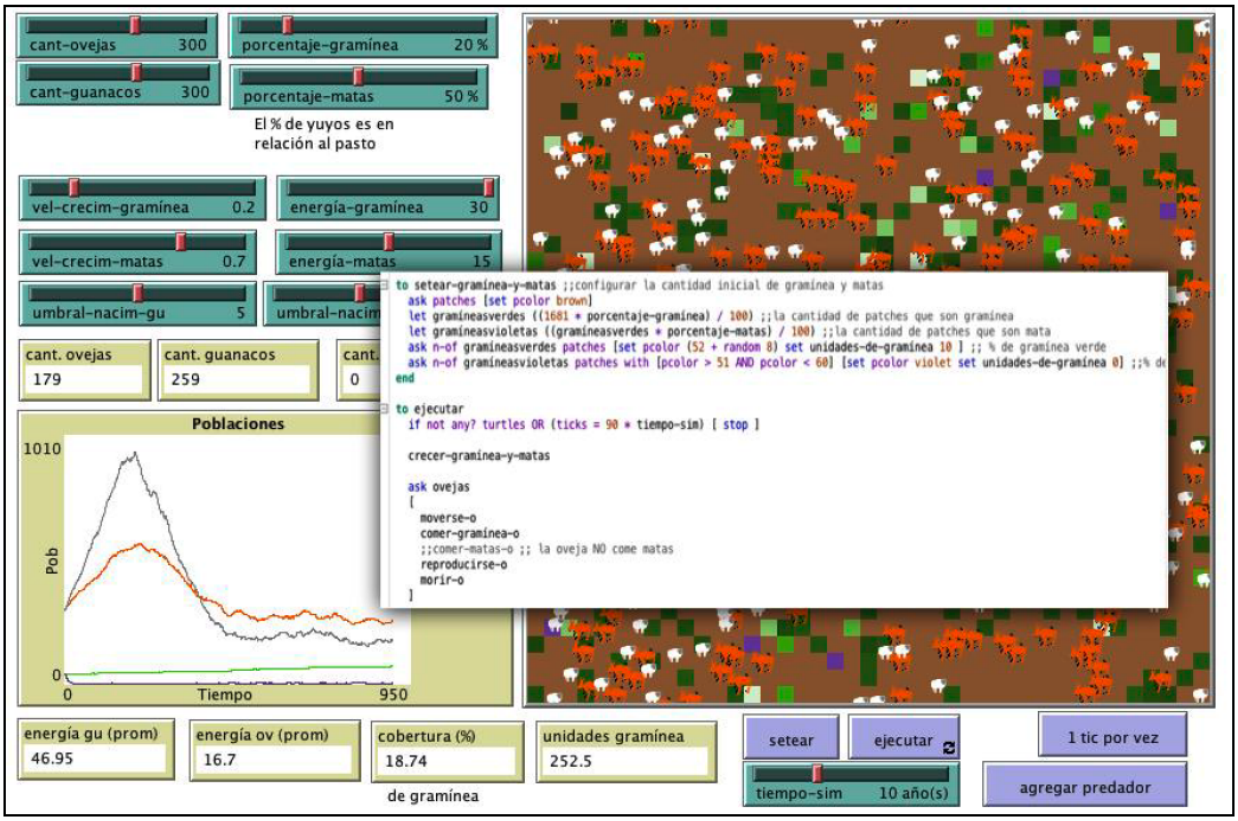

Figura 7: NetLogo: modelo de simulación de un ecosistema de guanacos y ovejas (2021). Elaboración del autor. Captura de pantalla del modelo "Guanacos y ovejas". Modelo disponible en: https://cienciascontic.github.io/simuladores/sim-guanacos-y-ovejas.html

StarLogo NOVA está traducido al español en su versión beta (https://beta.slnova.org) y NetLogo tiene su interfaz traducida al español pero los comandos siguen siendo en idioma inglés.

8. De cafeteras y multiprocesadoras: la cocina de la programación

Nicholas Burbules y Thomas Callister (2006) en su obra Riesgos y promesas de las nuevas tecnologías, hablan de la concepción instrumental de las TIC[9] que muchas personas tienen, como instrumentos que cumplen una función determinada, como por ejemplo una cafetera que solamente sirve para una función determinada: hacer café.

Si bien la metáfora de estos autores está utilizada en su libro para defender la idea de que no somos las personas quienes usamos las tecnologías sino que son ellas las que nos usan a nosotros, moldeandonos de maneras a veces imperceptibles, tomaremos prestado el concepto para contraponer a la cafetera con un electrodoméstico bastante en boga en estos tiempos: la Thermomix, que no es ni más ni menos que una multiprocesadora computarizada. Mientras que la cafetera solo hace café, la Thermomix puede hornear pan, cocinar guiso o amasar pasta.

Las cuatro herramientas de programación pensadas para ámbitos educativos de las cuales se ha hablado en este artículo (LOGO, Scratch, NetLogo y StarLogo NOVA) tienen una característica general que es la de permitir crear el

[9]TIC: Tecnologías de la Información y las Comunicaciones

modelo que uno quiera. Se podría decir que son el análogo a multiprocesadoras computarizadas.

Yendo a lo más específico, tanto NetLogo como StarLogo NOVA son entornos de programables de modelización basada en agentes que permiten crear el modelo que uno quiera, siempre obviamente dentro de las limitaciones estructurales que imponen tanto cada herramienta como el abordaje de MBA.

NetLogo y StarLogo NOVA son entornos de programación abiertos, en el sentido de que permiten a la persona en rol de modelizadora, crear el modelo que quiera. Esta libertad de creación, por supuesto, implica el conocer y ensayar la gramática y la sintaxis de ambos entornos, las posibilidades que ofrecen y la lógica algorítmica detrás de ellas.

9. Usar, modificar y crear modelos de simulación computacionales en el aula

Tanto en NetLogo como en StarLogo NOVA pueden **crearse** modelos de simulación nuevos, **usar** modelos ya creados, y/o **modificar** modelos existentes.

La mayoría de los simuladores con que trabajan los docentes de disciplinas STEM en el aula son una caja negra: no se sabe cómo están construidos por dentro.

En la Figura 6 se pueden observar algunos de los bloques de comandos que constituyen un modelo de simulación en StarLogo NOVA. Por su parte, en la Figura 7 puede verse una parte del código de NetLogo que corresponde al modelo en pantalla.

StarLogo NOVA y NetLogo son dos herramientas que permiten abrir la caja negra de los simuladores, posibilitan ver lo que hay debajo del capó. Como docente, se puede utilizar un modelo ya creado en StarLogo NOVA o NetLogo para que los estudiantes exploren un fenómeno determinado al igual que lo harían con un simulador tradicional, pero estas herramientas permiten además modificar un simulador ya existente, y/o crear uno nuevo.

StarLogo NOVA contiene un conjunto determinado de comandos/instrucciones que componen a este entorno programable, pero además, permite crear bloques propios que engloben varias instrucciones a la vez. Esta capacidad de StarLogo NOVA de crear bloques propios, permite crear pequeños micromundos[10] para desarrollar y hacer crecer ideas propias de ese dominio modelizado y modelizable.

NetLogo por sí solo no tiene esta capacidad, pero recientemente, un grupo de investigadores ligados al proyecto, desarrollaron una herramienta llamada NetTango (Horn, 2014), que permite también crear estos micromundos con bloques específicos para un dominio particular. “Laguna de las ranas (LdlR)” (Horn et. al, 2014) es un micromundo programable para trabajar los conceptos de evolución y selección natural.

En LdlR, se programa a los individuos rana a través de muy pocos bloques y muy específicos, como croar, girar, saltar, cazar, y apenas algunos más.

Con estos pocos bloques se programa a cada rana para lograr ciertos objetivos como por ejemplo una población estable de ranas de un tamaño determinado.

Esto es un ejemplo del concepto de programación dentro de un dominio: un micromundo específico con un conjunto acotado de instrucciones, donde la programación es muy sencilla integrando de esta manera el pensamiento computacional a la enseñanza de esa disciplina.

[10]En este caso se utiliza el concepto de micromundo en el más puro sentido papertiano: como un habitat donde alumbrar y hacer crecer ideas poderosas de ese dominio particular (Papert en Desafío de la Mente, p.146)

10. Conclusiones

El legado del LOGO como pionero de las tecnologías digitales en la educación llega hasta nuestros días a través de entornos programables como Scratch, StarLogo NOVA, NetLogo y NetTango, entre otros.

Algunos de estos entornos programables, tienen un enfoque específico de modelos basados en agentes, e igual que su predecesor, fueron pensados específicamente para el ámbito educativo; nacieron en el seno de la familia del LOGO, dentro del MIT; y de la mano de los discípulos de Papert.

Estos entornos programables nos permiten integrar el pensamiento computacional en la enseñanza de las disciplinas STEM, para ayudar a nuestros estudiantes a comprender mejor el impacto que estas tecnologías están teniendo en todos los ámbitos de la ciencia.

En un artículo complementario a éste, con el foco puesto en la creación de modelos de simulación computacionales en estos entornos programables, se profundizará sobre el entorno LdlR y se abordará específicamente el **cómo** crear estos micromundos en StarLogo NOVA y en NetLogo (a través de NetTango).

Bibliografía

Brand, S. (1988). El Laboratorio de Medio: inventando el futuro en el M.I.T. (Trad. L.Espinosa de Matheu). Ediciones Galápago, Argentina. (Trabajo original publicado en 1987)

Burbules, N. y Callister, T. (2006). Riesgos y promesas de las nuevas tecnologías de la información. Granica, Buenos Aires, Argentina.

Horn, M., Brady, C., Hjorth, A., Wagh, A., y Wilensky, U. (2014). Frog Pond: A code first learning environment on natural selection and evolution [Laguna de las Ranas: Un entorno de programación para el aprendizaje sobre la selección natural y la evolución]. Actas del IDC 2014.

Markoff, J. (1994). Thinking Machines to file for bankruptcy. New York Times. Citado en Wikipedia contributors. (2020, July 11). Danny Hillis. En Wikipedia, The Free Encyclopedia. Recuperado el 24 de octubre 24 de 2020, de https://en.wikipedia.org/w/index.php?title=Danny_Hillis&oldid=967092034

Papert, S. (1981). Desafío a la mente. Ediciones Galápago, Argentina.

Papert, S. y Solomon, C. (1971) Twenty things to do with a computer [Veinte cosas para hacer con una computadora]. M.I.T

Resnick, M., Klopfer, E. y Colella, V. (2001). Adventures in modeling: Exploring complex, dynamic systems with StarLogo [Aventuras en la modelización: Explorando sistemas dinámicos y complejos con StarLogo]. Teachers College Press, Estados Unidos de Norteamérica.

Resnick, M. (2001). Tortugas, termitas y atascos de tráfico. (Trad. J. Álvarez). Gedisa, España. (Trabajo original publicado en 1994)

Resnick, M. (2007). Sembrando las semillas para una sociedad más creativa. (Trad. C. Rizzi Iribarren). Learning & Leading with Technology, ISTE, Oregon, Estados Unidos.

Resnick et al. (2009) Scratch: Programming for All [Scratch: Programación para Todos], Communications of the ACM, Vol. 52, No. 11. Recuperado de https://cacm.acm.org/magazines/2009/11/48421-scratch-programming-for-all/fulltext

Weintrop, D. et. al (2020). El pensamiento computacional en las aulas de ciencias naturales y matemática (Trad. C. Rizzi Iribarren). [Archivo PDF] Recuperado de: http://www.terpconnect.umd.edu/~weintrop/papers/WeintropEtAl_2016_DefiningCT_esp.pdf (Trabajo original publicado en 2016).

Wing, J. (2006). Computational Thinking [Pensamiento Computacional]. Communications of the ACM March 2006/Vol. 49, No. 3.

Hacia una evaluación de calidad de Entornos de Desarrollo Integrado de Robótica Educativa

Martín Lobos* lobmar146@gmail.com UNLPam

Silvia Bast* silviabast@exactas.unlpam.edu.ar UNLPam

Gustavo Astudillo* astudillo@exactas.unlpam.edu.ar UNLPam

Resumen

Actualmente se puede encontrar innumerables propuestas didácticas para que estudiantes de distintos niveles educativos aprendan a programar. Todas estas propuestas involucran la utilización de software, particularmente entornos de desarrollo. Pero cuál es el más apropiado, cómo identificar el que mejor se adapta a la propuesta. En este artículo se detalla parte del proceso de medición y evaluación de calidad de entornos de desarrollo integrado para el aprendizaje de las nociones de programación usando robótica educativa. Para la evaluación de los productos de software se usará la estrategia denominada *Goal oriented context aware measurement and evaluation*. Se detalla el proceso de desarrollo del modelo de calidad, que toma como base el estándar ISO 25010:2011, y la definición de métricas para cada uno de los atributos del modelo de calidad.

Palabras clave: Entorno de desarrollo integrado, Medición, Evaluación, GOCAME, ISO 25010, Robótica Educativa, Programación.

1. Introducción

Actualmente es posible encontrar innumerables propuestas didácticas para que estudiantes de distintos niveles educativos aprendan a programar. Desde la cátedra Introducción a la Computación (FCEyN-UNLPam), en conjunto con el grupo GrIDIE, se viene implementando una propuesta de enseñanza de nociones básicas de programación para ingresantes a la carrera Profesorado en Computación. Al incluir la robótica educativa como parte del Taller de Introducción a la Programación (Astudillo et al. , 2019), surge la pregunta que motiva la presente investigación ¿Qué Entornos de Desarrollo Integrado (EDI), para robótica educativa, cuentan con la mayor flexibilidad para adaptarse a la propuesta didáctica del mencionado taller? Se inició, entonces, con la búsqueda que dio como resultado, la definición de la pregunta que guía la investigación ¿Cómo medir la calidad de los EDI?

Frente una amplia oferta de entornos basados en Google Blockly[1] se comenzó con el análisis para seleccionar el EDI más adecuado a través de metodologías de evaluación de calidad para productos de software.

*GrIDIE, Departamento de Matemática, FCEyN, UNLPam

[1]Blockly es un cliente de librerías para el lenguaje de programación Javascript, con el objetivo de crear lenguajes de programación visuales y editores basados en bloques.

La calidad del producto de software se puede interpretar como el grado en que dicho producto satisface los requisitos de sus usuarios. Por lo tanto, la importancia de la medición de calidad radica en que permite determinar si un producto de software es mejor a otro según las necesidades de sus usuarios.

2. Marco teórico

2.1. Medición de calidad

Dado que los EDI son desarrollos de software, debe usarse una metodología de evaluación de calidad asociada a estos productos, aplicando criterios establecidos por estándares internacionales, como la ISO 25010:2011.

Según David Garvin (1984) la calidad es un concepto complejo y multifacético. En la ISO 25010, se define la calidad de un sistema como "el grado en que el producto satisface los requisitos declarados e implícitos de los diversos interesados y, por lo tanto, proporciona valor".

Tal como afirman (Pfleeger y Kitchenham, 1996) para poder medir la calidad habitualmente se construyen modelos que relacionan las características de la calidad, descomponiendo cada una de ellas en diversos factores o relacionándolas de forma jerárquica. Será necesario, entonces, generar un modelo de calidad que dé cuenta de esta forma de descomposición. En este artículo se toman como base las características de calidad de producto de software que se encuentran especificadas en la ISO 25010:2011 y se hace uso de la estrategia de medición y evaluación denominada *Goal Oriented Context Aware Measurement and Evaluation* (GOCAME).

2.2. ISO 25010

El modelo de calidad del producto de software definido por la ISO/IEC 25010:2011 tiene ocho (8) características y cada una de ellas presenta un conjunto de subcaracterísticas (Figura 1) que se encuentran definidas dentro de la norma:

Figura 1: Modelo de calidad del producto software de ISO/IEC 25010. Imagen extraída de `iso.25000.com` .

2.2.1. Estrategia GOCAME

La estrategia GOCAME (Olsina et al., 2008) es un framework de medición y evaluación. Uno de sus principales aportes es el proceso para la medición y evaluación, que se especifica a continuación.

Para comenzar, se establece la necesidad de información, y a partir de la misma se definen los requerimientos, generando el modelo de calidad jerárquico que, en este caso, toma sus bases del estándar ISO 25010:2011, al que se

agregan un conjunto de subconceptos y atributos que surgen de los requerimientos discutidos y establecidos por el grupo de investigación.

Seguidamente, para cada uno de los atributos del modelo se definen las métricas, que se aplican posteriormente en el proceso de medición. Luego, se definen los indicadores, y se aplican en el proceso de evaluación. Los mismos ofrecen información contextualizada para la toma de decisiones.

Finalmente, se analizan los resultados y se realizan las recomendaciones de acuerdo a la necesidad de información establecida al inicio del proceso.

2.3. El TIP y la secuencia didáctica

Introducción a la Computación (IC), es la primera asignatura específica de la carrera Profesorado en Computación (PUC). Siendo, en la mayoría de los casos, el primer encuentro de los estudiantes con los conceptos de programación. Debido a esto, la cátedra IC y el grupo GrIDIE[2] vienen desarrollando distintas estrategias de enseñanza para favorecer el aprendizaje de conceptos básicos de programación.

El TIP fue diseñado con el objetivo de ofrecer un recorrido previo a dichos conceptos, haciendo uso de la programación en bloques, para luego poder transferir los mismos al contexto de la cursada de IC donde se usa el lenguaje de programación Pascal. Sus actividades fueron diseñadas para desarrollarse durante el período de ambientación universitaria que se lleva a cabo, las semanas previas al inicio del primer cuatrimestre de forma semipresencial (Astudillo et al., 2016).

Desde 2019 se trabajan los conceptos de alternativa condicional, repetición y nociones de variables a partir de una propuesta diseñada desde el grupo de investigación que hace uso de la robótica educativa para los ingresantes del PUC (Astudillo et al, 2019) y se sustenta en los principios del construccionismo (Papert, 1990), el buen aprendizaje (Pozo, 2008) y aprendizaje basado en problemas (Barrows, 1986).

La presente investigación comienza a partir de la decisión de incluir robótica educativa en el TIP, que fue tomada teniendo en cuenta que los robots resultan la conexión ideal entre la programación con una impronta lúdica y la representación de las instrucciones en un contexto real. Para Monsalves González (2011) y Ruiz-Velasco (2007) se trata de una disciplina que tiene como objetivo generar entornos de aprendizaje heurístico poniendo el foco en la participación activa, donde los aprendizajes se construyen a partir de la experiencia del estudiante durante el proceso de armado y programación de los robots.

Para la especificación de la secuencia fue necesaria la definición de kit (sensores y actuadores), la selección de un EDI, un simulador (para tareas extra-clase) y de un conjunto de ejercicios (actividades guiadas por el docente donde los estudiantes obtienen nuevos conceptos) y problemas (actividades sin guía del docente donde los estudiantes ponen en práctica dichos conceptos) que hacen uso de estos. La secuencia abarca los conceptos de estructuras de control (secuencia, repetición y selección) y la noción de variable. Asimismo, se desarrollaron un conjunto de *concept cards*[3] que complementan la secuencia en busca de la transferencia de los aprendizajes.

[2]Grupo de Investigación y Desarrollo en Innovación Educativa https://gridie.exactas.unlpam.edu.ar/

[3]Tarjetas las cuales plantean un problema a resolver a los estudiantes con determinado hardware (sensores y actuadores) disponible.

3. Resultados y discusión

Para iniciar el proceso de medición y evaluación de calidad, se definió el objetivo o necesidad de información: evaluar EDI de robótica educativa para posteriormente recomendar el que presente las condiciones adecuadas para implementar la secuencia didáctica propuesta proveyendo las mejores condiciones de usabilidad.

Luego, para comenzar con la construcción del modelo de calidad, se tomó como base el estándar ISO 25010:2011 y se seleccionaron 3 características (Figura 2) que, a criterio del grupo de investigación, sirven a modo de insumo para alcanzar el objetivo planteado:

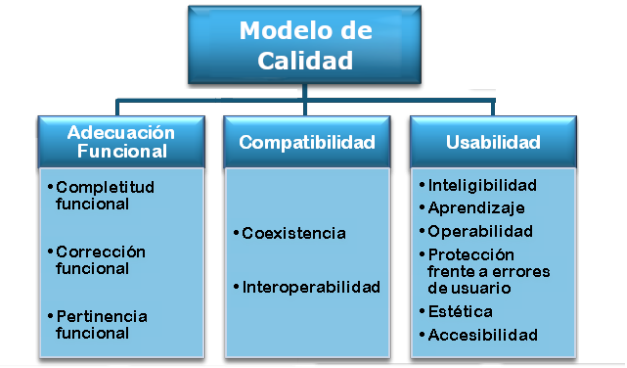

Figura 2: Ilustración del modelo de calidad generado con sus características y subcaracterísticas

A continuación, se definen cada una de las características y la motivación para su inclusión en el presente estudio:

Adecuación Funcional: *"Representa la capacidad del producto software para proporcionar funciones que satisfacen las necesidades declaradas e implícitas, cuando el producto se usa en las condiciones especificadas"* (ISO/IEC, 2011).

La inclusión de esta característica en el modelo de calidad, se basa en el vínculo estrecho que existe entre las funcionalidades ofrecidas por los EDI y las necesidades específicas que presentan las actividades que se proponen en la secuencia didáctica.

Focalizando así en las funcionalidades esenciales: ¿Autocompleta huecos en los bloques?, ¿Qué sentencias y tipo de datos tiene disponibles?, ¿Presenta uso en línea?, ¿Qué sensores y actuadores presentes en la secuencia nos permite utilizar?, etc.

Usabilidad: *"Capacidad del producto software para ser entendido, aprendido, usado y resultar atractivo para el usuario, cuando se usa bajo determinadas condiciones"* (ISO/IEC, 2011).

Su inclusión pone foco específicamente en evaluar características de los IDE que faciliten la comprensión del funcionamiento y resulten atractivos y sencillos para los estudiantes. También se observan características relacionadas a las licencias de uso y distribución que son de interés al cuerpo docente a cargo del TIP. Se evalúa entonces, la presencia de materiales educativos disponibles, elementos gráficos de la interfaz y la forma de interactuar con ellos, licencias, documentación en español, etc.

Compatibilidad: *"Capacidad de dos o más sistemas o componentes para intercambiar información y/o llevar a cabo sus funciones requeridas cuando comparten el mismo entorno hardware o software"*.

Finalmente se seleccionó Compatibilidad para evaluar el hardware y software compatible con los EDI en torno al material tecnológico disponible para el dictado del taller: ¿Qué placas son compatible con EDI?, ¿Cómo sube los

programas a las mismas?, etc.

Continuando con el proceso propuesto por la estrategia GOCAME, el siguiente paso fue definir los atributos, que son propiedades que se desprenden de cada una de las características y subcaracterísticas y resultan relevantes en relación con la necesidad de información. En la Tabla 1 puede observarse la definición de algunos atributos del modelo de calidad[4].

1. Adecuación Funcional	**2. Usabilidad**	**3. Compatibilidad**
1.1 Completitud Funcional *1.1.1 ¿Presenta aplicación móvil?* *1.1.2 ¿Se puede usar en línea?* ... 1.2 Correctitud Funcional *1.2.1 ¿Presenta ayuda experta?* *1.2.1.1 ¿Autocompleta huecos en los bloques que necesitan la inserción de valores/datos/expresiones?* ... 1.3 Pertinencia Funcional *1.3.1 ¿Permite la integración de extensiones?* *1.3.2 ¿Presenta gestión de proyectos y formato de aula en la nube?* *1.3.3 ¿Qué cantidad de bloques tiene para Sentencias Condicionales?* *1.3.3.1 Presenta el bloque If* *1.3.3.2 Presenta el bloque If-Else* ...	2.1 Accesibilidad *2.1.1 ¿Provee herramientas para adaptar el tamaño de los bloques?* *2.1.2 ¿Posee soporte para la utilización de lectores de pantalla?* ... 2.2 Estética e Interfaz de Usuario *2.2.1 ¿Presenta categorías de bloques distinguidas con imágenes?* *2.2.2 ¿Los botones representan correctamente su funcionalidad?* ... 2.3 Operabilidad *2.3.1 ¿Permite guardar el código fuente en extensión Arduino (.ino)?* *2.3.2 ¿Permite guardar el código fuente en bloques?* ...	3.1 Coexistencia *3.1.1 ¿Trabaja en conjunto con otro IDE?* 3.2 Interoperabilidad *3.2.1 ¿Para cuántas plataformas está disponible?* ...

Tabla 1: Una definición parcial del modelo de calidad

El siguiente paso fue establecer métricas, que representan un mapeo de un atributo a una variable que puede tomar valores categóricos o numéricos (Olsina, 2008). Para la definición de las mismas se debe indicar el método de medición (Objetivo/Subjetivo) y la escala (nominal, ordinal, intervalo, proporción y absoluta). Las métricas pueden ser directas (no depende de otra métrica) o indirectas (dependen de otra métrica). A continuación se presenta un ejemplo de métrica directa y otro de métrica indirecta[5].

Ejemplo de métrica directa:

1.1.1 **¿Presenta aplicación móvil?**

- **Atributo:** ¿Presenta aplicación móvil?
  - **Código:** 1.1.1.
  - **Definición:** Indica la existencia de una versión compilada para Smartphones con sistema operativo Android o iOS

[4]El modelo de calidad completo se encuentra disponible en: https://gridie.exactas.unlpam.edu.ar/?page_id=115

[5]El documento completo con todas las métricas está disponible en: https://gridie.exactas.unlpam.edu.ar/?page_id=115

- **Nombre de la métrica directa:** Existencia de una aplicación móvil para Smartphones
  - **Objetivo:** Determinar la existencia de una aplicación móvil para Smartphones
- **Método de medición:**
  - **Tipo:** Objetivo
  - **Especificación:** Se observa si existe una aplicación móvil para Smartphone.
- **Representación:** Discreta.
- **Tipo de valor:** (Android, iOS, Android y iOS, No presenta aplicación móvil)
- **Escala:** Nominal

Ejemplo de métrica indirecta:

2.2.5 **¿Presenta un adecuado contraste entre la letra y el fondo?**

- **Atributo:** ¿Presenta un adecuado contraste entre la letra y el fondo??
  - **Código:** 2.2.5
  - **Definición:** Grado en el que los colores de los bloques presentan adecuado contraste entre letra y fondo
  - **Nombre de la métrica Indirecta:** Porcentaje de colores de bloques con adecuado contraste entre letra y fondo
    - **Objetivo:** determinar el grado de colores de bloques con adecuado contraste entre letra y fondo
  - **Método de cálculo:**
    - **Especificación de la función:**

$$(\#CCFL/\#CCIDE) * 100$$

    - **Escala numérica:**
      - Tipo: Objetivo
      - Representación: Continua
      - Tipo de valor: Real
      - Tipo de Escala: Razón (*ratio*)
    - Donde:

      #CCLF: Cantidad de colores de bloques con adecuado contraste entre letra y fondo de acuerdo con la norma **WCA6 2.1 AA SC 1.4.3**

      #CCIDE: Cantidad de colores de bloques del IDE.

Se especifican a continuación #CCLF y #CCIDE que son las métricas directas de cuyo cálculo depende la métrica indirecta.

- **Nombre de la métrica directa:** Cantidad de colores de bloques con adecuado contraste entre letra y fondo de acuerdo con la norma **WCAG 2.1 AA SC 1.4.3**.(#CCLF)
- **Objetivo:** contar los colores de bloques que al menos presentan una relación 4.5:1 qué indica adecuado contraste entre letra y fondo de acuerdo con la norma **WCAG 2.1 AA SC 1.4.3.**
- **Método de medición:**
  - Tipo: Objetivo
  - Especificación: Para cada color de bloques del IDE. Si la el color del bloque presenta al menos una relación 4.5:1 que indica adecuado contraste entre letra y fondo de acuerdo con la norma **WCAG 2.1 AA SC 1.4.3**, entonces: $\#CCLF = \#CCLF + 1$

- **Escala Numérica:**
  - Representación: Discreta.
  - Tipo de Valor: Entero.
  - Tipo de Escala: Absoluta.
- **Nombre de la métrica directa:** Cantidad total de colores de bloques del IDE (#CCIDE)
- **Objetivo:** Contar todas los colores de bloques que presenta el IDE.
- **Método de medición:**
  - Tipo: Objetivo
  - Especificación: Para cada color de bloque del IDE.

$$\#CCIDE = \#CCIDE + 1$$

- **Escala Numérica:**
  - Representación: Discreta.
  - Tipo de Valor: Entero.
  - Tipo de Escala: Absoluta.

4. Conclusiones y rabajos Futuros

Dada la necesidad de información establecida, se definió un modelo de calidad, para ello, se seleccionaron las características de la ISO que resultan relevantes para la investigación, y se discutieron y especificaron cada uno de los atributos que permiten modelar la propuesta didáctica. Con base en lo anterior, para cada uno de los atributos se desarrollaron las métricas que se aplicarán en el proceso de medición.

Como trabajo futuro resta realizar el proceso de medición, que consiste en tomar la definición de cada métrica para producir un valor que será la medida (número o categoría asignada a un atributo de una entidad a través de la medición). Luego, tomando como insumo las medidas obtenidas queda realizar el proceso de evaluación, que consiste en tomar como insumo las medidas para generar información contextualizada. Para este proceso se realizará la definición de indicadores con el objetivo de proporcionar una estimación o evaluación de un concepto calculable con respecto a la necesidad de información definida.

Finalmente, como resultado de la evaluación se espera poder recomendar el EDI que mejor se adapte a las necesidades del cuerpo docente del TIP.

Bibliografía

Astudillo, G. J., Bast, S. G., y Willging, P. A. (2016). Enfoque basado en gamificación para el aprendizaje de un lenguaje de programación. Virtualidad, Educación y Ciencia, 7(12), 125–142.

Astudillo, G. J., Bast, S. G., Willging, P., Segovia, D., Castro, L., Lucero, P., Lobos, M., y Distel, J. M. (2019). Estrategias innovadoras en los procesos de enseñanza y de aprendizaje de la programación. XXI Workshop de Investigadores en Ciencias de la Computación, San Juan, Argentina. http://sedici.unlp.edu.ar/handle/10915/77161

Barrows, H. S. (1986). A taxonomy of problem-based learning methods. Medical education, 20(6), 481–486.

ISO/IEC (2011). ISO/IEC 25010 - Systems and Software Engineering - Systems and Software Quality Requirements and Evaluation (SQuaRE) - System and Software Quality Models. Technical report.

Garvin, D. A. What does Product Quality really mean? (1984). Sloan management review, 25, 25–43.

Kitchenham, B., y Pfleeger, S. (1996). Software Quality: The Elusive Target. IEEE Softw., 13, 12–21.

Monsalves González, S. (2011). Estudio sobre la utilidad de la robótica educativa desde la perspectiva del docente. Revista de Pedagogía, 32(90), 81–117.

Olsina, L., Papa, F., y Molina, H. (2008). How to measure and evaluate web applications in a consistent way. In Web Engineering: Modelling and Implementing Web Applications (pp. 385–420). Springer, London.

Papert, S. (1990). A critique of technocentrism in thinking about the school of the future. Epistemology and Learning Group, MIT Media Laboratory. Recuperado de http://cursa.ihmc.us/rid=1N5PWKQGN-H47Q0R-37ZW/Papert%20critique%20of%20technocentrism.pdf

Pozo, J. I. (2008). Capítulo 4. Los rasgos de un buen aprendizaje. En Aprendices y maestros: la psicología cognitiva del aprendizaje (pp. 159–175). Madrid, España: Alianza.

Prensky, M. (2001). Digital Natives, Digital Immigrants. On the Horizon, 9(5).

Ruiz-Velasco, E. (2007). Educatrónica: Innovación en el aprendizaje de las ciencias y la tecnología. Madrid: Díaz de Santos.

Vaillant, D. (2013). Integración de TIC en los sistemas de formación docente inicial y continua para la Educación Básica en América Latina. Argentina: UNICEF Argentina. Recuperado de https://www.unicef.org/argentina/spanish/educacion_Integracion_TIC_sistemas_formacion_docente.pdf

Diseño de una arquitectura extensible y escalable para refundar el Proyecto Gobstones

Alan Rodas Bonjour
alanrodas@unq.edu.ar
Universidad Nacional de Quilmes

Federico Agustín Sawady O'Connor
federico.oconnor@unq.edu.ar
Universidad Nacional de Quilmes

Resumen

La enseñanza inicial de la programación se presenta actualmente como un desafío de especial relevancia a nivel mundial. Entre los desafíos principales se encuentra el contar con herramientas y plataformas capaces de asistir a los docentes en sus propuestas pedagógicas, así como acompañar a los alumnos en su formación. Gobstones es una de dichas plataformas que ha presentado una amplia adopción y visibilidad a nivel nacional e internacional en los últimos años. Esto ha traído aparejado diversas falencias en su base de código, documentación y materiales didácticos.

En este trabajo se plantea entonces una refundación integral del proyecto Gobstones, generando una nueva base de código más robusta y escalable, que permita un desarrollo más rápido y dinámico, logrando adaptarse a las necesidades actuales, pero también a futuras, permitiendo que el proyecto se adapte a nuevas necesidades de estudiantes y docentes, así como a diferentes escenarios de aprendizaje de la programación.

Adicionalmente se generará documentación precisa y accesible, tanto para usuarios de la plataforma como para desarrolladores de la misma, congregando todo el material disperso en una única ubicación. Eso sumado a nuevas propuestas de gobernanza en el manejo de este proyecto de software libre generará mayor visibilidad y acercará nuevas oportunidades de colaboración.

En su conjunto, se espera que esta refundación logre posicionar a Gobstones como una de las primeras elecciones al momento de diseñar propuestas de enseñanza de la programación inicial.

Palabras clave: Gobstones, Entornos de Aprendizaje, Secuencia Didáctica, Enseñanza Inicial.

1. Introducción

La programación está en todas partes y su aprendizaje se vuelve menester en la sociedad moderna para poder comprender el mundo que nos rodea. La enseñanza inicial de la programación es un desafío, pues requiere no solo de docentes capacitados a tal fin, sino también de herramientas que apoyen a la didáctica.

Gobstones es un proyecto que nació en la Universidad Nacional de Quilmes en el año 2009, para usarse como herramienta en el dictado de la materia Introducción a la Programación (Martínez López et al., 2012). Sus creadores, el Dr. Pablo Ernesto Martínez López y el Dr. Eduardo Bonelli, querían un lenguaje que invirtiera el orden en el que se

encaraban los conceptos de enseñanza con respecto a los cursos tradicionales (donde se comienza enseñando conceptos abstractos para arribar a soluciones concretas, optando por ir desde lo concreto a lo abstracto en su lugar). Gobstones consiste de una serie de desarrollos que integran una ambiciosa plataforma, compuestas de múltiples proyectos, tanto de software con utilidades para estudiantes y docentes, como de materiales didácticos enfocados en estos (Martínez López et al., 2017). Entre los primeros destacan:

- **El lenguaje Gobstones**: Un lenguaje de programación ad-hoc para la enseñanza de la programación inicial. El lenguaje tiene como características distintivas eliminar elementos innecesarios encontrados en los lenguajes industriales, como entrada y salida, importaciones, construcciones complejas como los bucles de repetición mediante actualización de variables, y otras, simplificando los elementos del lenguaje a aprender (Martínez López, 2013). A su vez, provee un universo de discurso que los estudiantes pueden comprender y hasta visualizar, incluso si no cuentan con conceptos matemáticos subyacentes o grandes niveles de abstracción. Así, el lenguaje aísla a los estudiantes del sistema de cómputo subyacente, para enmarcarlos en un universo en donde un cabezal se mueve por un tablero rectangular cuadriculado, dejando y quitando bolitas de colores de las celdas del mismo.
- **Un entorno de desarrollo integrado**: Enfocado en el mencionado lenguaje, y a partir de la segunda versión del mismo, se incorporó a Gobstones un entorno de desarrollo integrado. El entorno no solo provee los clásicos elementos de un editor de texto, como resaltado de sintaxis y *code folding*, sino que además permite editar elementos propios del lenguaje, como el tablero, cambiando su estado inicial y visualizando el estado final.
- **Un sistema de programación mediante bloques encastrables**: En la última versión del entorno se agregó la posibilidad de programar mediante bloques encastrables en lugar de programación mediante texto, volviendo la plataforma más adecuada para estadios iniciales de la enseñanza. Las ventajas de este tipo de plataformas han demostrado superar ampliamente a sus desventajas (Weintrop y Wilensky, 2015). Además el sistema mediante bloques de Gobstones permite ir incrementando gradualmente la cantidad de herramientas disponibles.

Entre los segundos podemos mencionar:

- **Cursos integrados a la plataforma**: El entorno cuenta con la posibilidad de cargar ejercicios. Un ejercicio modifica el entorno permitiendo presentar por ej. un enunciado, código inicial incompleto con espacios claros donde se espera que el estudiante complete un escenario inicial a resolver, entre otros. Además es posible limitar la cantidad de opciones o herramientas disponibles al estudiante, permitiendo que él deba hacer foco en aspectos específicos del problema.[1]
- **Manual docente**: A través de la Fundación Sadosky y en colaboración con la Universidad Nacional de Quilmes, se desarrolló un manual para el primer ciclo de secundaria (Martínez López et al., 2019). El manual presenta múltiples fichas de ejercicios utilizando Gobstones.
- **Decenas de ejercicios adicionales presentados en PDF**: A lo largo del dictado de cursos y materias en distintos lugares se han desarrollado cientos de ejercicios, que no se encuentran disponibles al público. En ocasiones, sus autores distribuyen el contenido mediante licencias libres, pero no se cuenta con un lugar claro para centralizar y encontrar los mismos.
- **Videos didácticos**: Pueden encontrarse videos de aprendizaje de la plataforma, el lenguaje y la didáctica a través

[1] Puede verse el entorno con un curso con actividades en https://gobstones.github.io/gobstones-sr/?course=gobstones/curso-LPYSD1.

de *YouTube*[2].

Gobstones ha experimentado un rápido crecimiento en los últimos años, pasando de ser utilizado únicamente en la Universidad Nacional de Quilmes, lugar que vio su nacimiento, para pasar a ser utilizado en múltiples instituciones, tanto universitarias como de educación media y proyectos privados, incluso a nivel internacional. Entre las que tenemos registro podemos mencionar: Universidad Nacional de Quilmes, Universidad Nacional de Hurlingham, capacitaciones docentes por la Fundación Sadosky dentro del marco de Program.ar, Plan Argentina Programa, Mumuki (que a su vez es utilizado en distintas provincias argentinas y Brasil), Plan Ceibal (Uruguay), ESET UNQ y Escuela Florentino Ameghino. Únicamente entre el nivel universitario hablamos de un promedio de 1500 estudiantes/usuarios por año, y a eso debemos sumar que en Argentina Programa se inscribieron más de 157.000 personas[3]. A las mencionadas hay que sumar una gran cantidad de instituciones que utilizan Gobstones para sus propuestas didácticas pero de las cuales no tenemos actualmente registro certero. Actualmente tenemos registro de más de 10 cursos o propuestas didácticas en la plataforma, pero podrían haber muchas más debido a la característica descentralizada de los mismos.

Este crecimiento dio como resultado una demanda amplia por parte de los usuarios de diversas características en el lenguaje y en el entorno asociado, pero la falta de financiamiento del proyecto, la ausencia de un equipo de desarrollo estable y malas decisiones tecnológicas iniciales llevaron a un desfasaje entre la cantidad de usuarios y las posibilidades de crecimiento de la plataforma. Entre las problemáticas podemos incluir la rápida obsolescencia del código (la mayoría de las tecnologías usadas se encuentran deprecadas y con problemas de seguridad), la imposibilidad de traducir el software a distintos idiomas (solo se soportan algunos idiomas de forma ad-hoc), la falta de comunidad y pautas de contribución al proyecto y la disgregación de los materiales de apoyo.

A continuación vamos a enumerar los componentes más relevantes que actualmente integran la plataforma Gobstones y que se encuentran activos y con mantenimiento mínimo. Adicionalmente a éstos, hay diversos componentes en forma de bibliotecas que brindan funcionalidad a los aquí mencionados. Describiremos además algunas problemáticas asociadas a los mismos:

- **Intérprete**: El intérprete es el componente de la plataforma encargado de ejecutar programas Gobstones y dar los resultados de tal ejecución. Este se compone, a su vez, de un *tokenizador* y *parser* tradicionales, escritos manualmente con técnicas estándar, un compilador que traduce el código a una serie de instrucciones ad-hoc y una máquina virtual que interpreta dichas instrucciones. Seguimos utilizando la denominación de intérprete, a pesar de que el código del mismo haya evolucionado hasta ser un compilador con una máquina virtual ad-hoc. El motivo de que el lenguaje deba ser compilado y se utilice una máquina virtual en lugar de un simple interprete responde a la necesidad de soportar actividades interactivas.

  Una característica interesante del lenguaje, soportado por el intérprete, es la capacidad de poder utilizar identificadores que estén en castellano. Es decir que, a diferencia de muchos lenguajes industriales, símbolos como “ñ”, o letras acentuadas (“á”, “é”, “í”, “ó”, “ú”) están permitidas. Esto elimina la barrera del idioma en el proceso de aprendizaje de la programación, permitiendo escribir una función bajo el nombre “díaDeMañana” en lugar de “diaDeManana” o “diaDeManhana”, estos últimos mecanismos estándar en cursos en dónde se enseña sin lenguaje con estas características.

  La forma en la que se maneja el soporte para letras acentuadas es frágil y se encuentra fuertemente atada al

[2] https://www.youtube.com/watch?v=JWc58qgRh_A&list=PLfrTDBDj636aRNJQA19LB7paPuQX_QEpm

[3] https://www.argentina.gob.ar/noticias/argentina-programa-mas-de-157-mil-inscriptos-en-7-dias

lenguaje en el que se implementó (se delega a *JavaScript* determinar si una letra es un símbolo válido o no). Más aún, esto hace muy difícil formalizar el lenguaje Gobstones, y brindar soporte a idiomas con alfabetos más complejos, como el árabe, chino o japonés.

Algunas de las tecnologías empleadas en el componente se encuentran anticuadas y/o obsoletas, incluyendo la forma en la que se distribuye el componente (Vía *Bower*[4], elemento ya deprecado en favor de *NPM*[5], estándar de la industria). Adicionalmente, el hecho de estar escrito en un lenguaje con tipado dinámico hace imposible detectar errores en tiempo de compilación (y como veremos más adelante, el código sufría de problemas por esta carencia). La documentación está escrita como comentarios dentro del código, o como archivos de *LaTeX*[6] que deben ser compilados para leerse cómodamente, colocando una barrera a los usuarios del componente, así como a posibles contribuyentes.

- **Componente de Bloques**: Gobstones soporta la programación mediante bloques encastrables, de forma similar a otras plataformas pensadas para la enseñanza inicial de la programación, como Scratch (Resnick et al., 2009), que se ha vuelto muy popular en las instituciones de educación primaria y secundaria. A diferencia de Scratch, Gobstones brinda la posibilidad de mostrar un panel de herramientas reducido, adaptado a la actividad en concreto y limitando la cantidad de decisiones que el estudiante/usuario debe tomar. En ese sentido, se parece más a PilasBloques (Sanzo et al., 2017) que usa una aproximación similar, con la ventaja para los docentes de poder diseñar sus propias actividades. Tras bambalinas, Gobstones utiliza *Blockly* para proveer esta funcionalidad, un *framework* desarrollado por *Google* que provee mecanismos necesarios para definir bloques y espacio de trabajo con los mismos y generar código a partir de éstos. La versión del *framework* utilizada es una versión beta, ni siquiera publicada oficialmente, atada a un commit del proyecto blockly en su momento.

  Este componente se distribuye como un componente de *Polymer*[7], tecnología en la cual está parcialmente desarrollado, junto con *JavaScript*, también utilizando *Bower* que, como ya hemos mencionado, es una tecnología obsoleta. También se exporta como un *gem* de *Ruby*, como una necesidad surgida de un usuario de la plataforma, pero el proceso de actualización y distribución en este formato es separado de la primera.

  A diferencia de otros componentes de la plataforma, el soporte multilenguaje es prácticamente nulo, y lo poco que hay se encuentra realizado como un agregado ad-hoc para el soporte únicamente de inglés estadounidense y castellano.

  En cuanto a la base de código, podemos mencionar que es bastante pobre, pues no incluye documentación, ni para los usuarios ni para los desarrolladores, y los comentarios en el código son escasos y poco útiles. El código se compone principalmente de un script de 1700 líneas de código, escritas de forma imperativa, dónde partes del código se encuentran duplicadas o incluso triplicadas y no hay reutilización o generalización de ningún tipo.
- **IDE**: El *IDE* es el componente visual más importante de la plataforma, pues integra los diversos componentes, proveyendo una experiencia coherente de desarrollo, amena para los estudiantes. El *IDE* integra un editor de texto con soporte para el lenguaje, el entorno de bloques, un componente para visualizar el tablero y sus bolitas (parte fundamental del lenguaje) y una serie de menús y botones para interactuar con todos los diversos

[4] https://bower.io

[5] https://www.npmjs.com

[6] https://latex.org

[7] https://polymer-library.polymer-project.org

elementos.

Este *IDE* consiste en una aplicación, capaz de ser ejecutada tanto vía web (y *webapp*) como de forma offline como aplicación de escritorio (mediante el uso de *Electron*[8]). La aplicación se integra además con elementos didácticos, como las guías de ejercicios, presentando en su interfaz la posibilidad de navegar y elegir entre los diversos ejercicios, leer los enunciados y cargar su código asociado. La herramienta en general, con todos los componentes integrados, es denominada GobstonesWeb.

No es de extrañar que la base del código del *IDE* también se encuentre obsoleta, pues fue desarrollado junto con los componentes antes mencionados. El uso de *Polymer* en su versión 1.7, ya no mantenida, junto con *Bower* y otros, se presentan como algo difícil de mantener. A esto cabría sumar la problemática de encontrar desarrolladores para tal tecnología, pues *Polymer* nunca logró gran popularidad.

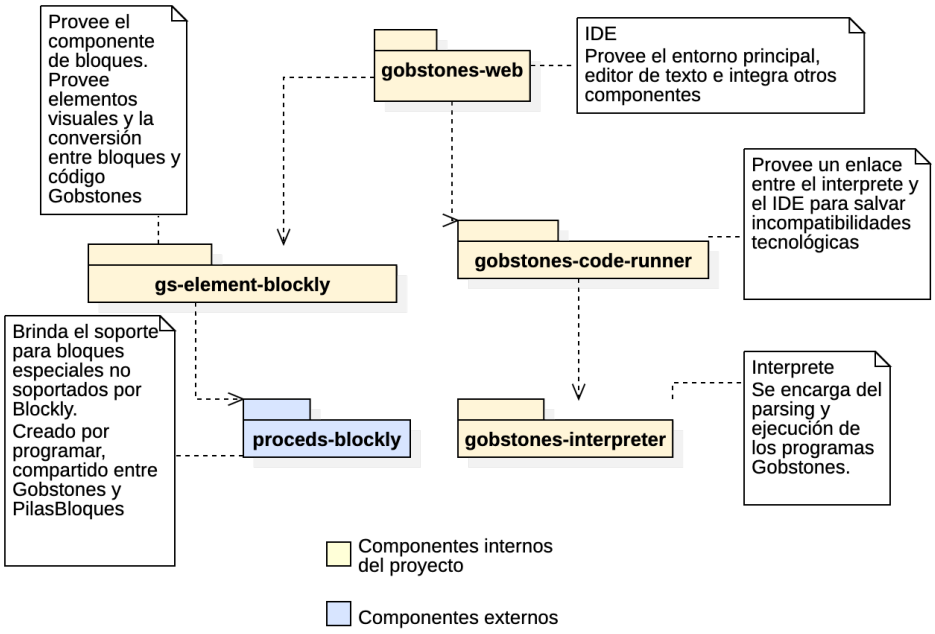

Figura 1: En este gráfico se aprecia la arquitectura actual de componentes de la plataforma Gobstones para la versión actualmente en uso, GobstonesWeb.

GobstonesWeb, la versión actual de Gobstones no solo sufre de problemas de obsolescencia de sus dependencias o ausencia de documentación, sino que su arquitectura es endeble, pues depende en exceso de servicios de terceros. Por ejemplo, los proyectos de actividades se encuentran alojados en la plataforma GitHub, y se utiliza la API de dicho servicio para acceder a los archivos allí ubicados, pero hay un número máximo de peticiones por hora que pueden solicitarse. Así, cuando múltiples personas se encuentran conectadas al mismo tiempo, no se puede acceder a las actividades y se presenta un error insalvable. Ninguna solución es viable para este problema sin modificar sustancialmente la arquitectura de la plataforma. Además cabe destacar que estos problemas arquitecturales surgieron de un equipo de trabajo que hoy no forma parte del proyecto, y que no tiene interés en participar en el desarrollo y mantenimiento del mismo, haciendo para el equipo actual la tarea de mantener la base de código mucho más lenta y

[8] https://www.electronjs.org

compleja de lo que debería.

En este trabajo se presenta la realización del proceso de refundación del proyecto Gobstones, incluyendo la arquitectura inicial de todos sus componentes, y una plataforma inicial funcional. Esta refundación implica una nueva arquitectura, escalable y robusta, utilizando tecnologías modernas y permitiendo un amplio desarrollo a futuro y apostando a que desarrolladores de cualquier parte del mundo puedan contribuir al proyecto de forma clara y sencilla. A su vez, se realizará y actualizará la documentación, tanto para desarrolladores como para usuarios (estudiantes y docentes), colocando la totalidad del material existente y nuevo material, al alcance de los mismos, y generando espacios de intercambio y difusión para las personas que quieran sumarse al uso de la plataforma.

2. Desarrollo de Software de la plataforma

En esta sección analizaremos el desarrollo llevado a cabo en términos del software del proyecto, así como diversas decisiones de diseño y arquitecturales, y las motivaciones subyacentes para las mismas. Comenzaremos por describir las tecnologías empleadas y sus decisiones de diseño. Luego pasaremos a discutir sobre cada uno de los componentes elaborados. Finalmente charlaremos sobre el estado general de los nuevos desarrollos.

El foco de este proyecto consiste en el desarrollo de una arquitectura que sea lo suficientemente robusta y escalable. Así, entre los principales desafíos se encontraban el elegir el lenguaje y las tecnologías (frameworks y bibliotecas) a utilizar, así como determinar ciertas características no funcionales adicionales para el proyecto.

Dividiremos esta sección en decisiones generales sobre la totalidad de la plataforma y desarrollos puntuales de nuevos componentes o reescrituras de los actuales.

2.1. Decisiones generales

Cabe destacar que la Universidad Nacional de Quilmes se presenta actualmente como el sponsor más importante del proyecto Gobstones, cuidando su continuidad y desarrollo. Así, el actual equipo de trabajo se compone principalmente de voluntarios de dicha institución, docentes, estudiantes doctorales, de grado y pregrado que realizan diversos trabajos en el marco de sus tesis o trabajos finales de carrera suelen ser los principales contribuyentes al código. Sin embargo, los desarrollos suelen estar acotados en el marco de un trabajo puntual, por lo que es alta la rotación de personas.

Así, mantener un código prolijo, altamente documentado y con un arquitectura sólida en tecnologías fuertemente conocidas, que permita a las personas rotar activamente, acercándose al proyecto con bajas barreras de entrada, es la primera consideración que tuvo el equipo previo a la refundación.

Así, se tuvieron en cuenta los aspectos no tecnológicos, sino conceptuales y humanos, prestando especial importancia a la visión a futuro de la plataforma en su conjunto. La alta rotación de gente nos llevó a pensar en una arquitectura de pequeños módulos bajamente acoplados, pero con integraciones clave. Cada modulo se plantea provea una API clara para su integración desde otros, que no requiera sistemas de adaptación intermedios, y que usen una misma tecnología y lenguaje.

Diseñamos y pensamos módulos de los cuales solo planteamos una idea inicial, un *README* con una propuesta de API, y que podrán ser creados a futuro en el marco de nuevos desarrollos, y rediseñamos los módulos principales de la plataforma que brindarán soporte clave a la arquitectura en general.

Adicionalmente pensamos en que una de las necesidades más importantes está en el poder formar una comunidad, uniendo desarrolladores con usuarios (ya sean docentes o estudiantes) y pudiendo enriquecer la plataforma compartiendo experiencias y actividades. Así surgió la idea de tener un espacio centralizado para cursos y actividades que puedan ser fácilmente compartibles por sus creadores, espacios de encuentro y un nuevo y mejorado sitio web (Troilo, 2021).

Todos estos aspectos llevaron a decisiones desde lo tecnológico que se exploran en las siguientes subsecciones.

2.2. Lenguaje de programación

Una de las decisiones de diseño importantes tiene que ver con la elección de un único lenguaje de programación a utilizar a lo largo de la totalidad de la plataforma. Previamente cada componente utilizaba el lenguaje que el desarrollador hubiera elegido en su momento, y hay componentes escritos en *CoffeeScript*[9], otros en *JavaScript*, e incluso partes en *Ruby*. El determinar el uso de un único lenguaje responde a que nuevos desarrolladores que quieran contribuir a la plataforma deban contar con conocimientos en un único lenguaje en lugar de varios, para poder comprender y participar en cualquiera de los componentes, reduciendo el costo de entrada. Esto además permite utilizar el mismo conjunto de herramientas de desarrollo para todos los componentes, simplificando los procesos de configuración previos a comenzar a trabajar.

El lenguaje de programación elegido para utilizar a lo largo de toda la plataforma fue *TypeScript*[10]. *TypeScript* es un *superset* de *JavaScript* que agrega el soporte de análisis estático de tipos, brindando un nivel de confianza superior en el código producido. La elección del lenguaje se basa en múltiples factores que se detallan a continuación.

En primer lugar, se requería un lenguaje que pudiera ser *transpilado*[11] a **JavaScript**, pues la idea de mantener un entorno web, que reduzca las fricciones de acceso de los usuarios a la herramienta, sigue presente. Así, las opciones se reducen sensiblemente, aunque no tanto, pues la cantidad de lenguajes que apuntan a cubrir tal nicho son amplias, destacando el mismo *JavaScript* (nativo o vía herramientas como *Babel*[12]), *Elm*[13], *ClojureScript*[14] y *Dart*[15]. Elegir un lenguaje que se encuentra en sus inicios, o que no cuenta con una fuerte comunidad es una apuesta arriesgada que decidimos no realizar, pues podría acarrear la obsolescencia del código en plazos muy cortos, sin mencionar que dificulta la contribución al proyecto (habiendo menos desarrolladores para esos lenguajes).

Por otro lado, se buscaba un lenguaje que tuviera la característica de tipado estático. Esto minimiza en gran medida la cantidad de posibles errores al momento del despliegue de los componentes, y permite tener un nivel de confianza superior en cuanto a la interfaz que cada componente exporta. No solo eso, sino que también permite a los desarrolladores contar con herramientas avanzadas (integrados al entorno de trabajo), como puede ser la detección inmediata de errores (al momento de escribir el código, mediante compilación en background), auto completado inteligente o refactoring

[9] https://coffeescript.org

[10] https://www.typescriptlang.org

[11] La transpilación es un proceso equivalente a la compilación, pero en donde el lenguaje objetivo es uno con características de abstracción similares a aquel del lenguaje fuente

[12] https://babeljs.io

[13] https://elm-lang.org

[14] https://clojurescript.org

[15] https://dart.dev

avanzado, entre otros. Esto mejora sensiblemente la experiencia de desarrollo de los programadores.

Así, *TypeScript* se presenta como un lenguaje que posee la característica de tipado estático deseada, y se ha posicionado en los últimos años como un estándar de la industria, con soporte para su sistema de tipado por parte de herramientas como *Rollup*[16] o *Webpack*[17], y la elección de este por empresas de importante envergadura, como *Microsoft*, *Google*, *Facebook* y otras. Esto hace que *TypeScript* no sea solo una moda pasajera, sino que se posicione como un entorno estable que tendrá gran soporte y crecimiento en los próximos años. El hecho de su gran popularidad, así como de que sea un superset de *JavaScript* (uno de los lenguajes más populares actualmente) permite asegurarse que habrá capacidad de conseguir desarrolladores que contribuyan al proyecto en futuro, sin necesidad de grandes capacitaciones.

2.3. Frameworks

En cuanto a *frameworks*, intentamos mantener una idea minimalista, eligiendo utilizar *frameworks* o toolkits solo en aquellos lugares donde el beneficio de su uso compense el tiempo de aprendizaje del mismo por parte de un desarrollador ajeno al mismo.

Así, podemos mencionar que una gran cantidad de los componentes desarrollados, se exportan como bibliotecas *JavaScript* en formato *ESM* (*ECMA module*). En ese sentido, no hay *frameworks* involucrados en la mayoría de los casos, a excepción de los componentes que incluyen parsers, en donde se eligió utilizar un *framework* que genera el parser a partir de la definición de la gramática, eligiendo el uso de Nearley parser por su versatilidad y simpleza de uso.

En cuanto a los componentes visuales, se optó por *React*[18]. La motivación nuevamente viene de la mano de la posibilidad de contar con mayor cantidad de desarrolladores con conocimientos previos a la tecnología, así como apostar a un *framework* que sea estable y cuente con el apoyo de algunos de los grandes jugadores de la industria y una amplia comunidad, permitiendo proyectar un gran soporte y crecimiento a futuro de la herramienta. *React* se presenta como tal framework, siendo hoy una de las alternativas más populares. Otras opciones analizadas fueron *Angular*[19], *Ember*[20] y *Vue.js*[21], pero finalmente la popularidad, gran comunidad y excelente documentación, inclinaron la balanza a favor de *React*.

Otros *frameworks* fueron utilizados en lugares puntuales, como Blockly, que brinda soporte a todo el entorno de bloques de la plataforma, siendo la alternativa más estable y mejor documentada que lo haga, y siendo su competidor Scratch Blocks una de las pocas alternativas disponibles.

Por último podemos mencionar a *Jest*[22] como el *framework* de testing, que se utiliza a lo largo de los diversos componentes para realizar pruebas unitarias y de integración.

[16] https://rollupjs.org

[17] https://webpack.js.org

[18] https://es.reactjs.org

[19] https://angular.io

[20] https://emberjs.com

[21] https://vuejs.org

[22] https://jestjs.io

2.3.1. Otras herramientas

Adicionalmente a los *frameworks* que se utilizan en tiempo de ejecución, hay múltiples herramientas de software que son utilizadas en los diversos proyectos, ya sea brindando asistencia al programador en tiempo de desarrollo, empaquetando los componentes, permitiendo automatizar tareas, etc.

Git[23] es usado como herramienta de control de versiones, y *GitHub*[24] es la plataforma elegida para mantener el proyecto. Esto trae también acarreado el uso de los diversos elementos que provee *GitHub* para el mantenimiento de los proyectos, como su sistema de *Issues*, *Pull Requests* y *Wikis*. También se hace uso de *GitHub Actions*, como herramienta para la realización de *Integración Continua (CI)*, y *Deploy Continuo (CD)* ante determinados eventos en el repositorio.

A esto se suma el uso de *Husky*[25], una herramienta que se integra con *git* en la máquina local del desarrollador, y que se encarga de instalar y ejecutar *hooks de git*. Los proyectos tienen entonces configurados *hooks* que corren los *tests* del proyecto previo a la realización de un *commit* o un *push*, evitando que un desarrollador suba código que pueda fallar de forma no intencional.

Todo proyecto incluye *ESLint*[26] y *Prettier*[27] como una forma de garantizar un estilo de programación uniforme a lo largo del proyecto, así como detectar posibles errores lógicos en el código. Adicionalmente se incluyen una serie de archivos de configuración para el editor, siendo *Visual Studio Code*[28] el editor elegido como plataforma de desarrollo (Aunque pudiendo desarrollar en cualquier otro entorno si así se desea).

NPM es el registro (y herramienta) elegida para ejecutar, mantener y publicar los diversos componentes de software. La alternativa era el uso de *Yarn*[29], pero optamos por por una línea estándar clásica.

2.4. Componentes específicos

La división de componentes del proyecto actual resultaba insuficiente desde las experiencias previas. Nuevos desarrollos se presentaban como sumamente complejos y requerían gran cantidad de horas por parte de los interesados solo para comprender y conocer las tecnologías utilizadas (en sus versiones sin soporte y a veces sin documentación online), conocer en profundidad el código fuente y todos los pequeños “trucos” usados en el mismo para solucionar distintas problemáticas. Esto llevaba a que las personas optaran por no colaborar debido a su complejidad.

Así, decidimos crear una nueva serie de componentes bajamente acoplados, que permitan a los futuros desarrolladores ocuparse únicamente en aprender el o los componentes relevantes para su desarrollo.

El diagrama de componentes que se muestra a continuación es la base del diseño actual de la nueva plataforma. Notar que no todos los componentes están siendo actualmente desarrollados, o que no todos se encuentran publicados. Sin embargo, la concepción de dichos componentes y su integración en el proyecto general permite tener una visión clara del trabajo a realizar, brindando una clara hoja de ruta en el progreso de la plataforma.

[23] https://git-scm.com

[24] https://github.com

[25] https://typicode.github.io/husky

[26] https://eslint.org

[27] https://prettier.io

[28] https://code.visualstudio.com

[29] https://yarnpkg.com

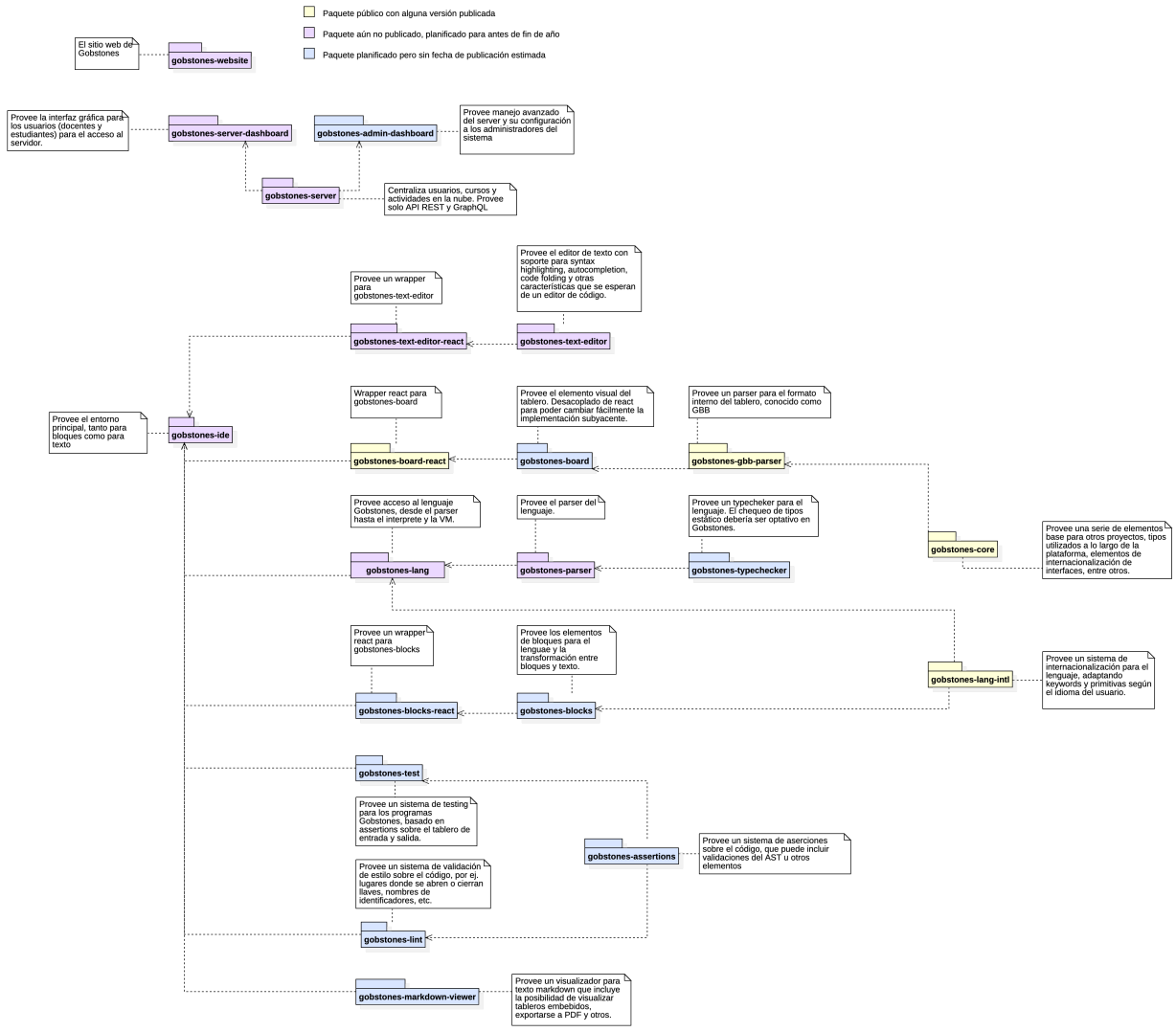

Figura 2: La proyección de la arquitectura tras la refundación.

2.4.1. Nuevos componentes y elementos base

A raíz de la refundación, se observó que múltiples componentes requerían hacer uso constante de los mismos elementos conceptuales. Por ejemplo, el concepto de tablero era transversal al visualizador de tableros, el intérprete, el parser, etc. Cada componente definía su propia implementación de ese concepto, y el paso de un elemento a otro requería entonces la conversión entre formatos. El tablero era solo uno de los elementos en esa situación.

Así, surgió la necesidad de tener algún tipo de componente base, capaz de ser utilizado por todos los otros componentes, y que incluyera una serie de elementos generales que se usan de forma transversal.

Este componente no solo incluyó un modelo de tablero lo suficientemente eficiente y escalable para poder ser utilizado en cualquier situación, sino también herramientas de validación de datos mediante reglas, elementos de internacionalización y otras herramientas necesarias de forma transversal. A nivel arquitectural, ante la detección de duplicación conceptual en diversos componentes, se cuenta ahora con un lugar específico donde poder desacoplar el elemento y evitar duplicación de código.

También se separaron elementos anteriormente integrados en un único componente. Por ejemplo, el parser y el intérprete y máquina virtual son ahora componentes separados. Esto da lugar al uso individual de los mismos para diferentes fines. Por ejemplo, el parser podría ser utilizado de forma independiente al intérprete en sistemas de validación de estructura del código, como forma de corrección automatizada.

Adicionalmente, el editor de código se separó del *IDE*, el editor de bloques se hizo agnóstico de *framework* y se crearon componentes adicionales que los engloban y permiten su integración sencilla al *IDE*. Esto no solo desacopla el componente, sino que permite en el futuro cambiarlos rápidamente por una nueva versión, incluso sí se utilizan bibliotecas o *frameworks* completamente distintos.

En su conjunto, todos estos elementos contribuyen a una arquitectura más escalable, en donde cada componente tiene una, y solo una, responsabilidad específica. Luego, el *IDE* integra esos componentes para lograr una experiencia coherente y cohesiva.

2.4.2. Reescritura de componentes

Los componentes actualmente disponibles fueron reescritos en su totalidad o en gran parte.

El sistema de bloques fue reescrito en su totalidad, utilizando las últimas herramientas disponibles (Blockly en sus últimas versiones). Esto trae aparejado no solo mejoras en la seguridad, sino nuevas funcionalidades, que permitirán a futuro crear nuevos elementos en Gobstones. No solo eso, sino que el uso de *TypeScript*, el manejo de procesos de integración continua y testing brinda mayor confiabilidad ante cambios en este componente. El uso de NPM permite su incorporación en nuevos desarrollos de forma rápida y sencilla.

También se reescribió el *IDE*, pasándolo a *React*, dando lugar a mayor cantidad de desarrolladores a trabajar sobre la plataforma. No solo eso, sino que el uso de otras bibliotecas dan lugar a cosas como el poder realizar *hooks* ante acciones del estudiante u otros elementos, algo imposible con la arquitectura actual.

El intérprete fue portado completamente a *TypeScript*, proceso en el cual se encontraron errores en el mismo en casos específicos que se usan poco, y que no habían sido detectados. Las ventajas de la elección de un lenguaje estático resultaron evidentes en este escenario.

2.4.3. Componentes proyectados

Dentro del diseño de la arquitectura también se decidió marcar un rumbo a futuro, para lo cual se proyectaron múltiples componentes que no fueron completamente desarrollados. La mayoría de estos cuenta solamente con un repositorio que incluye un *README* sobre la idea a desarrollar, e incluso algunos *tasks* en un *backlog*. La idea de los mismos es que cualquier desarrollador podría ir planteando diversos elementos de estos componentes, dando lugar a procesos de mejora claros y estructurados, que llevarán a futuros hitos en la plataforma, aunque sin fechas límites.

Entre estos elementos se encuentran el *testing* de programas Gobstones y otros componentes que contribuyen de forma general al desarrollo. El diseño arquitectural nuevo de Gobstones da lugar a explorar y explotar la idea de *"plugins"* del lenguaje, expandiendo las capacidades del mismo. Ahondaremos en algunos de estos elementos en la sección de trabajo a futuro.

2.5. Estado general de los componentes

Al momento de la redacción del presente, algunos de los componentes arriba mencionados se encuentran ya publicados mediante versiones estables y son públicos en el repositorio de la organización Gobstones. Otros aún se encuentran en estado alfa y están en proceso de ser pulidos previo a su liberación final. No se cuenta de momento con proyectos en beta pública, pero la nueva versión de Gobstones se liberará como tal previo a un *release* formal que reemplace a la actual plataforma.

3. Generación de documentación, comunidad y gobernanza

En este capítulo discutiremos los desarrollos realizados en términos de la construcción de comunidad, políticas de gobernanza y documentación. Dividiremos este capítulo en varias secciones según el desarrollo realizado.

3.1. Sitio web

La web de Gobstones siempre fue pobre. A raíz de la creciente demanda por información del proyecto, los accesos a la misma crecieron de forma exponencial. Sin embargo, solo podía encontrarse en esta un enlace a la herramienta, nada más.

La creación de un nuevo sitio web resultó entonces completamente necesario. Se creó un sitio realizado en Wordpress, con la intención de poder ser manejado y escalado fácilmente a futuro. Allí, no solo se encuentran enlaces a la herramienta, sino que se aglutina información sobre la propuesta didáctica y sus conceptos filosóficos, su historia, etc. Se dejan pautas de contribución, se documenta claramente el lenguaje, se proveen guías de uso del mismo, etc.

La web aún no ha hecho su debut público, pero su lanzamiento formal es inminente.

3.2. Discord

Discord[30] es una plataforma de comunicación que ha ganado popularidad en los últimos años por su versatilidad, en particular entre los jóvenes. Puede pensarse como un nuevo *IRC*. Las ventajas de *Discord* sopesan sus desventajas (el hecho de ser cerrado y privado, con servidor centralizado).

Se creó entonces un espacio en *Discord* para generar comunidad. Sí bien aún no ha hecho su debut público ya se encuentra en funcionamiento, y el equipo de trabajo interno ha encontrado en este espacio una muy valiosa forma de comunicación y discusión.

3.3. GitHub

El uso que se daba a *GitHub* en el proyecto era exclusivamente como repositorio de código. La refundación ha decidido abrazar *GitHub* y apoyarse en la plataforma en virtud de generar una comunidad y una mejor política de gobernanza del proyecto. Así, se explotaron los *GitHub* Sites, para contar con la documentación de todo proyecto siempre publicada, las *GitHub* Actions, para contar con testing e integración continua, así como publicación automatizada de los componentes, y el uso de los proyectos e issues para manejar adecuadamente el desarrollo del proyecto y las expectativas a futuro.

[30] https://discord.com

3.4. Server de manejo de la plataforma

La creación de un servidor centralizado donde el usuario pueda conectarse y cargar sus actividades (con enunciados, estado inicial, etc.) es también una parte didáctica. Para los docentes, posibilita el seguimiento claro de sus actividades, la generación de cursos completos, la asociación de sus estudiantes y su seguimiento, entre otros elementos.

Para los estudiantes permite visualizar sus cursos, pero también revisar cosas previamente realizadas sin estar llevando consigo las soluciones en archivos, con la posibilidad de acceder a las mismas desde cualquier lado.

La oportunidad de compartir elementos es algo que esta plataforma brinda de forma cómoda para los docentes, dando lugar a un repositorio con licencias libres de actividades y propuestas didácticas diversas.

El servidor aún no ha hecho su debut, pero se espera que lo haga a fin de año, luego de tener publicada una beta inicial de la nueva plataforma.

4. Trabajo Futuro

En primer lugar cabe destacar que aún queda pendiente la puesta en producción y la culminación del proyecto, realizando el despliegue de los elementos realizados. Sí bien esto es parte de lo que compone el presente trabajo, al momento de publicación del presente, aún no todos los nuevos componentes se encuentran desplegados.

Además, si bien este trabajo incluye una gran cantidad de software generado, aún queda gran cantidad de software por realizar. Muchos repositorios presentan únicamente una propuesta de API con lineamientos generales para su implementación, pero el código debe aún ser desarrollado. Estos trabajos ampliarán la plataforma con nuevas funcionalidades, que podrán ser desarrollados por el mismo equipo, o por programadores independientes.

Se abren las puertas entonces para las siguientes líneas de trabajo.

- **Testing de aplicaciones y validaciones de código**: La capacidad de realizar pruebas sobre el código de diferentes formas. Por un lado, en el sentido de pruebas de unidad. La propuesta incluye la posibilidad de plantear una serie de escenarios (tableros iniciales) y sus resultados finales esperados (un tablero final asociado que debe obtenerse luego de ejecutar el programa). Esto permitirá validar que el código cumple con lo planteado en el enunciado en diferentes casos. Sin embargo, la filosofía Gobstones no está centrada en las características funcionales del programa únicamente, pues son precisamente varias de las características no funcionales las que aportan valor al código realizado y al proceso de aprendizaje. Así, también se busca que los tests puedan incluir aseveraciones sobre la estructura del código (Ejemplo, el programa principal llama a un único procedimiento, ese procedimiento incluye una repetición condicional, etc.).
- **Formalizaciones**: Gran parte de los componentes pueden conllevar una futura formalización. Entre estas podemos destacar la gramática, el sistema de tipos, la semántica operacional y denotacional del lenguaje, etc.
- **Minería de datos y recomendaciones de IA**: La minería de datos o exploración de datos puede aplicarse a la base de datos del servidor, pudiendo extraer estadísticas y hasta descubrir patrones, dado el volumen de conjuntos de datos recolectados a lo largo de los cursos en todo el país. Por otra parte, los datos obtenidos pueden ser utilizados para entrenar sistemas con técnicas como machine learning o redes neuronales. Estos sistemas pueden aportar sugerencias para alumnos y docentes, sobre qué errores un alumno podría estar cometiendo en diferentes ejercicios, y así sugerir material de repaso de ciertos conceptos.

5. Conclusiones

La nueva arquitectura de la plataforma, así como las nuevas políticas de gobernanza del proyecto hace que Gobstones se presente más escalable y robusto, permitiendo un amplio desarrollo a futuro que dará lugar a que nuevos desarrolladores puedan contribuir al proyecto de forma clara y sencilla.

A su vez, la nueva documentación, incluido su nuevo sitio web, centralización de materiales, guías, foros y otros, darán lugar a que tanto los estudiantes que estén aprendiendo con esta plataforma, como aquellos docentes que quieran utilizarla como su herramienta didáctica, puedan contar con documentación verdaderamente útil, mejorando significativamente la experiencia general.

Es temprano para sacar conclusiones en este momento debido a que no todos los componentes se encuentran desplegados. Sin embargo se espera que una vez puesta en producción se observen beneficios tangibles para estudiantes y docentes.

Entre los beneficios para estudiantes podemos destacar una mayor facilidad para acceder a sus cursos y actividades, registro de su progreso y la posibilidad de continuar actividades desde el último punto, independientemente de la máquina utilizada. Con el despliegue final y los diversos proyectos en mente se espera cuenten con mejor *feedback* automatizado, que hoy está solo disponible a través del *feedback* docente (por ejemplo, saber si la actividad cumple ciertos criterios de estilos en términos de nombres de identificadores, estructura de código, etc.).

Para los docentes se espera puedan realizar cursos a partir de otros ya existentes, compartir actividades con otros docentes y realizar *remixes* de forma similar a lo que provee Scratch. A eso se suma la capacidad de visualizar las actividades realizadas por los estudiantes de sus cursos y analizar su estado y progreso. Asimismo, las devoluciones automatizadas permitirían reducir la carga de trabajo docente al momento de la corrección.

Por supuesto, estos beneficios aún quedan por observarse de forma empírica.

Bibliografía

Martínez López, P. E., Bonelli, E. y Sawady O'Connor, F. A. (2012). El nombre verdadero de la programación: Una concepción de introducción a la programación y sus bases conceptuales. Anales del 10mo Simposio de la Sociedad de la Información (SSI'12) para las 42as Jornadas Argentinas de Informática (JAIIO'12), 1–23.

Martínez López, P. E. (2013). Las bases conceptuales de la programación: una nueva forma de aprender a programar (1ra ed.) http://www.gobstones.org/bibliografia/Libros/BasesConceptualesProg.pdf

Martínez López, P. E., Ciolek, D., Arévalo, G. y Pari, D. (2017). The GOBSTONES method for teaching computer programming. SIESC'17 dentro del 2017 XLIII Latin American Computer Conference (CLEI), 1-9.

Martínez López, P. E., Aloi, F., Ciolek, D., Pari, D. y Tobía, P. (2019). Ciencias de la computación para el aula: 1er ciclo de secundaria (V. Klinkovich y H. Czemerinski, Eds.; 3ra ed). Fundación Sadosky. http://bit.ly/CCau1S

Resnick, M., Maloney, J., Monroy-Hernández, A., Rusk, N., Eastmond, E., Brennan, K., Millner, A., Rosenbaum, E., Silver, J., Silverman, B. y Kafai, Y. (2009). Scratch: Programming for All. Communications of the ACM, 52(11), 60–67. https://doi.org/10.1145/1592761.1592779

Sanzo, A., Schapachnik, F., Factorovich, P. y Sawady O'Connor, F. (2017). Pilas bloques: A scenario-based children learning platform. 2017 Twelfth Latin American Conference on Learning Technologies (LACLO), 1–6.

Troilo, J. (2021). Aspectos organizacionales de proyectos de Software Libre. Análisis comparativo y propuesta de aplicación para proyectos jóvenes (Reporte del Trabajo de Inserción Profesional). Tecnicatura Universitaria en Programación Informática

del Departamento de Ciencia y Tecnología de la Universidad Nacional de Quilmes. Director del Trabajo Ing. Alfredo Sanzo. UNQ.

Weintrop, D. y Wilensky, U. (2015). To block or not to block, that is the question: students' perceptions of blocks-based programming. Procs. of 14th international conference on interaction design and children, 199–208.

¿Scratch, Python, o qué?

Criterios para elegir un entorno para enseñar a programar a principiantes

Marcos J. Gómez
marcos.gomez@unc.edu.ar
Universidad Nacional de Córdoba
Fundación Dr. Manuel Sadosky

Pablo E. "Fidel" Martínez López
fidel@unq.edu.ar
Universidad Nacional de Quilmes
Fundación Dr. Manuel Sadosky

Resumen

A la hora de dictar un curso de programación inicial, una decisión importante, que influye en las características didácticas del curso, es la elección del lenguaje de programación y el entorno de trabajo asociado. Existen numerosos lenguajes y entornos para el dictado de un curso inicial de programación. Sin embargo, muchas veces la decisión se toma en forma azarosa, o guiada por modas, y no se basa realmente en una consideración criteriosa de las características relevantes del lenguaje y/o entorno, dando como resultado cursos que no siguen totalmente las características que el docente buscaba.

En este trabajo nos proponemos analizar de forma amplia las características de los lenguajes y entornos existentes que impactan en la didáctica del curso que se busca dictar, y de esa forma ofrecer una serie de criterios que permitan tomar la decisión de manera informada, conociendo las consecuencias que la misma tendrá sobre las características del curso. Para ello proponemos y describimos espacios de dimensiones para clasificar los entornos y lenguajes, los cuales impactan al momento de elegir para enseñar a programar. Estas dimensiones fueron elegidas a partir de nuestro trabajo con diversos entornos y lenguajes, y abarcan de manera amplia la mayoría de las características relevantes que afectan las características del curso a dictar. Estas características a su vez se pueden agrupar en diferentes facetas que nos ayudan a ordenar las ideas presentadas. Las dimensiones están vinculadas con el propósito, la expresividad, la flexibilidad y otras características de los lenguajes de programación en sí, a las las formas de construcción y expresión de programas, y a las posibilidades de las herramientas asociadas, así como con el grado de influencia que cada una de las dimensiones tiene en la presentación de los conceptos buscados. Teniendo en cuenta las dimensiones y facetas definidas definiremos algunas consideraciones para elegir entorno o lenguaje en base al contexto de enseñanza.

Palabras clave: Entornos para enseñar a programar, Lenguajes de bloques, Lenguajes de texto, Nivel de expresividad.

1. Introducción

En la actualidad la enseñanza de la programación en niveles primario y secundario ha cobrado una relevancia destacada como herramienta para adquirir los elementos necesarios para comprender partes fundamentales de la

sociedad y la cultura modernas. Los cursos tradicionales de programación se focalizan en la enseñanza de programación desde el ángulo profesional, sin tener en cuenta el tipo de formación buscada, y usualmente resultan demasiado complejos. Por esa razón han surgido numerosos enfoques de cursos que buscan acercar la enseñanza de la programación a estos niveles, muchos con la intención de transmitirlo como conocimientos de cultura general, sin aspiraciones a priori de desarrollo profesional posterior, y otros contemplando desde cero la enseñanza profesional de la disciplina.

Un elemento fundamental de estos cursos, tanto los que no tienen aspiraciones profesionales como los que sí buscan esta formación, es la elección del lenguaje de programación a utilizar, usualmente acompañado de un entorno de trabajo que facilite diversos aspectos de la tarea de desarrollar un programa. Cuando se decide dar un curso inicial de programación, la variedad de enfoques y características hace que, a la hora de elegir el lenguaje o entorno, los docentes nos vemos abrumados de opciones, y muchas veces no contamos con los elementos necesarios para tomar una decisión informada. Esto lleva a que se caiga en una elección azarosa, por criterios basados en modas o por simple desconocimiento de otras opciones, antes que tener en cuenta las implicaciones que tal decisión tendrá en el éxito de nuestro curso.

La selección de los entornos de trabajo, y consecuentemente de los lenguajes de programación que se trabajarán en el curso a dictar implica una paradoja. No es tan importante usar o no un lenguaje o entorno determinado, como transmitir una serie de conceptos relevantes de manera clara y efectiva. El tema es que para cumplir ese objetivo (transmitir conceptos importantes de manera clara y efectiva), la elección del lenguaje y el entorno resulta importante. Y así, llegamos a la ***paradoja del lenguaje***, presentada por Martínez López et al. (Martínez López et al., 2012): el lenguaje y el entorno a utilizar no son importantes, pero sin embargo son importantes. Por un lado no es importante qué lenguaje o entorno se elige, porque lo importante son los conceptos. Pero por otra parte, dado que el lenguaje, experimentado a través del entorno, es la única forma de expresar adecuadamente esos conceptos, es importante ver de qué manera este entorno afecta la forma de transmitir los conceptos deseados. Eso hace que decidir bien por el entorno no sea una tarea trivial, pero tampoco lleva a que haya una única opción y ni siquiera "la mejor" opción: cada uno debe decidir en función de una serie de cuestiones. El punto clave es conocer esas cuestiones, y decidir teniéndolas en cuenta. Gómez clasifica a los entornos y lenguajes de enseñanza de programación en base a diferentes aspectos que impactan en la incorporación de un lenguaje al aprender a programar (Gómez, 2020): la expresividad del lenguaje, la interactividad formativa y la evaluación automática. Nosotros hemos decidido ampliar ese aporte con una organización sistemática de los criterios, a través de diferentes **dimensiones**, cada una de ella contemplando diversas **facetas**.

En este trabajo nos proponemos analizar de forma amplia las características de los lenguajes y entornos existentes que impactan en la didáctica del curso que se busca dictar, y así ofrecer una serie de criterios que nos permitan tomar la decisión de manera informada, conociendo las consecuencias que la misma tendrá sobre las características de nuestro curso.

2. El espacio de dimensiones a considerar

Con el objetivo de analizar los lenguajes y entornos de programación que podrían elegirse a la hora de dictar un curso, comenzaremos por describir diferentes ***dimensiones*** que nos permiten clasificar los diferentes entornos de enseñanza de programación existentes y así analizar sus características. Estas dimensiones fueron elegidas a partir de nuestro trabajo con diversos entornos y lenguajes, y abarcan de manera amplia la mayoría de las características relevantes que afectan las características del curso a dictar. A su vez las características de cada dimensión se pueden

agrupar en diferentes ***facetas*** que nos ayudan a ordenar las ideas presentadas. Las dimensiones están vinculadas con el propósito, la expresividad, la flexibilidad y otras características de los lenguajes de programación en sí, a las las formas de construcción y expresión de programas, y a las posibilidades de las herramientas asociadas, así como con el grado de influencia que cada una de las dimensiones tiene en la presentación de los conceptos buscados.

La primera de las dimensiones tiene que ver con el ***lenguaje en sí, su propósito y expresividad***, y otras características del mismo. En esta dimensión están la distinción de comenzar a enseñar directamente con lenguaje de propósitos generales o utilizar un lenguaje diseñado con el objetivo específico de aprender programación y también otras consideraciones. Esta dimensión se discutirá en la Sección 2.1.

La segunda de las dimensiones tiene que ver con la ***forma de construcción*** de los programas, y en esta dimensión, en un extremo están los lenguajes textuales, donde el código se construye a través de la confección de textos que deben seguir una sintaxis rígida, y en el otro, entornos basados en bloques, donde el código se construye a través de bloques con formas de piezas de rompecabezas. Esta dimensión se discutirá en la Sección 2.2.

Finalmente, la tercera dimensión analizada se vincula con las ***herramientas de soporte*** asociadas a los entornos de trabajo, más que al lenguaje. Esta dimensión trata acerca de la capacidad del entorno para permitir la confección de cualquier programa sin restricciones en un extremo (conocidos como "entornos de mundo abierto"), a la capacidad de controlar totalmente las características de las actividades a desarrollar, guiando a los estudiantes en una secuencia didáctica específica, en el otro. Esta dimensión se discutirá en la Sección 2.3.

2.1. Propósito y expresividad: lenguaje de propósitos generales, o lenguaje específico para aprender

Una dimensión importante a la hora de analizar los lenguajes y entornos utilizados para enseñar a programar es desde el punto de vista del lenguaje en sí, entendiendo por lenguaje al conjunto de reglas sintácticas que definen cuáles son programas válidos del lenguaje de programación y los propósitos y posibilidades del lenguaje. En esta dimensión aparece una primera faceta con respecto al ***propósito***: en un extremo los lenguajes de propósitos generales, mayormente utilizados a nivel industrial y comercial y sin consideraciones de didáctica o enseñanza, y en el otro los lenguajes diseñados específicamente para aprender a programar. También incluimos en esta dimensión tres facetas más: por un lado los elementos fundamentales que cada lenguaje ofrece como ***universo de discurso***, por otro las consideraciones asociadas al ***paradigma de programación***, y por último, la faceta de la ***explicitación de la definición*** del lenguaje: muchos de los entornos basados en bloques no cuentan con una definición del lenguaje de programación que sea independiente del entorno en sí, sino que se basan en un conjunto de bloques comunes a muchos lenguajes del paradigma imperativo, con algunas incorporaciones ocasionales vinculadas al paradigma de objetos.

Veamos en primer lugar la faceta del **propósito del lenguaje**. Los lenguajes de programación históricamente más comunes, utilizados en los ámbitos comercial e industrial, son lenguajes de propósitos generales, pues permiten solucionar una amplitud grande de problemas y son adecuados para producir productos de software complejos. Tradicionalmente estos lenguajes se utilizaban también a la hora de enseñar. Sin embargo, y a pesar de que la denominación "propósitos generales" incluye también su uso educativo, los lenguajes de propósitos generales suelen tener gran cantidad de herramientas pensadas para atacar problemas de índole diversa, y una sintaxis y estructura que se concentra más en la productividad que en la claridad. Por esa razón resultan poco eficaces para usarlos como primer lenguaje a la hora de enseñar.

Entre 1969 y 1976, Papert (Solomon and Papert, 1976) realiza un estudio sobre pedagogía y enseñanza de programación, y propone la creación del lenguaje Logo con el propósito exclusivo de servir como lenguaje para la enseñanza de la programación. Este trabajo y este lenguaje dieron origen a toda una línea de lenguajes diseñados con el exclusivo propósito de aprender a programar. A lo largo del tiempo surgieron numerosas propuestas de lenguajes y entornos pensados con este propósito, pero es recién en la última década cuando estas propuestas cobran una nueva relevancia, a partir del trabajo de muchos gobiernos e instituciones. Lenguajes como Alice (Cooper et al., 2000), Scratch (Resnick et al., 2009), App Inventor (Wolber, 2011), Gobstones (Martínez López et al., 2012; Martínez López, 2013; Martínez López et al., 2017), Pilas-Bloques (Sanzo et al., 2017) y Chatbot (Benotti et al., 2017) son lenguajes/entornos definidos con el propósito principal de ser utilizados para la enseñanza básica de la programación.

Otra faceta interesante a considerar a la hora de evaluar un lenguaje radica en el conjunto de elementos primitivos que cada uno tiene para expresar los problemas, su **universo de discurso**. Los lenguajes diseñados para enseñar a programar se distinguen pues ofrecen un universo de discurso que resulta concreto a los estudiantes. El objetivo es que pueda comenzarse el abordaje de la programación desde elementos que resulten familiares, en lugar de encontrarse con herramientas y elementos abstractos, como una memoria compuesta por números, estructuras de datos como arrays o punteros, u otras. Así, Logo tiene la *tortuga y sus dibujos*, Alice, Scratch y Pilas-Bloques tienen *personajes y sus acciones* y Gobstones tiene un *tablero y sus bolitas*. Con respecto a estos elementos, hay dos aspectos a considerar: por un lado qué tan bien son entendidos por los estudiantes y qué tanto sirven para expresar conceptos de programación, cuáles conceptos y con qué flexibilidad, y por el otro qué tan generalizables son para luego enseñar elementos más abstractos de los lenguajes de propósitos generales. Estos aspectos de la faceta del universo de discurso suelen referenciarse como “piso bajo, techo alto”, y por lo general tienen menos impacto que otros durante la elección de un lenguaje para enseñar.

La tercera faceta que elegimos considerar está relacionada con los **paradigmas de programación** en los que se basan los lenguajes, y en la influencia de la misma a lo largo de la formación posterior. La gran mayoría de los lenguajes populares para utilizar en primeros cursos se basa en el paradigma imperativo, especialmente en el enfoque de la programación estructurada (Dijkstra, 1968; Watt and Findlay, 2014) dado que esos cursos no pretenden ir más allá de los conceptos más básicos de dicho paradigma. Sin embargo, si se tiene en cuenta objetivos formativos de más largo plazo, debe considerarse que en programación existen otros paradigmas, como el paradigma funcional[1] (Hughes, 1989; Bird and Wadler 1990) o el paradigma de objetos (Black and Palsberg, 1994; Abadi and Cardelli, 2012) para mencionar los más difundidos. La pregunta es, entonces, si para nuestro curso alcanza con enseñar un subconjunto de la programación imperativa/estructurada (o funcional u orientada a objetos), o si buscamos una formación más amplia, más general y de largo plazo. Y en ese caso, considerar las características que cada lenguaje ofrece, qué tan bien expresa los conceptos que se busca transmitir de un paradigma, o que permitan luego el aprendizaje de otros paradigmas es muy importante. Como dijimos, la gran mayoría de los lenguajes y entornos más difundidos se concentran en aspectos de la programación imperativa, con algunas características de otros paradigmas, como el de objetos o de funcional. Entendemos que es así por dos razones: la primera es que históricamente la programación imperativa es anterior a los otros paradigmas, y la segunda es que la programación imperativa tiene elementos que son más visibles desde lo concreto que las abstracciones que proponen los otros paradigmas, y por lo tanto más complejos para ser utilizados como primer abordaje. Queda por considerar cómo seguir con la formación que se busque.

[1]Un definición concisa del paradigma funcional en los primeros 52 segundos del siguiente vídeo: https://www.youtube.com/watch?v=LnX3B9oaKzw&ab_channel=Computerphile.

La última de las facetas a considerar en esta dimensión hace referencia a la definición explícita del lenguaje mediante reglas que permitan determinar si un programa será válido o no respecto de ellas. En muchos de los lenguajes cuyo propósito es enseñar, usualmente definidos mediante formas visuales de construcción (ver la Sección 2.2), no existe una definición *a priori* de cuáles son las combinaciones válidas de herramientas del lenguaje, quedando las mismas implícitas en los bloques que existen y en sus formas. Esto no es un problema al comienzo, pero si posteriormente se quiere avanzar hacia herramientas más complejas, o seguir haciendo una transición hacia lenguajes textuales, esta falta de definición explícita puede constituirse en una traba o dificultad nueva a superar. En estos lenguajes sin definición a priori normalmente las construcciones más complejas no están definidas o siguen lógicas que son difíciles de generalizar a lenguajes de propósitos generales. Esto es así porque dichos lenguajes fueron pensados solamente como iniciales, sin considerar la transición posterior a otros lenguajes. Otros lenguajes, como Logo, Gobstones o Racket (Felleisen et al., 2015), sí poseen una definición del lenguaje que es independiente de cualquier entorno o forma de construcción. Nuevamente, vale la pena considerar esta faceta para comprender cómo afectará el uso posterior del conocimiento que nuestro curso imparte.

Esto completa nuestra discusión sobre la primera de las dimensiones.

2.2. Formas de construcción: lenguaje textual, o entorno de bloques

Otra dimensión para analizar lenguajes de programación está vinculada a la forma de construir los programas. Considerando que un programa es siempre una descripción de una solución a un problema computacional ejecutable en forma mecánica (Martínez López et al., 2012), podemos entender que un programa está estructurado en base a una serie de partes que componen esa descripción, y cabe preguntarse de qué manera el programador expresa y ensambla esas partes. En la programación tradicional la manera de expresar y ensamblar las partes que conforman un programa es simplemente mediante cadenas de texto (secuencias de caracteres). En la programación con bloques, en cambio, la forma de expresar y ensamblar las partes es mediante *bloques*, similares a las piezas de un rompecabezas. Mientras que en la programación basada en texto existen infinidad de cadenas de caracteres que resultan inválidas y que solamente pueden ser descubiertas como tales luego de un análisis pormenorizado, en la programación basada en bloques la validez de una combinación está dada por la forma misma de los bloques ya que se establece de manera visual y concreta las posibilidades de combinación sintáctica, eliminando la necesidad de presentar en forma explícita estas reglas sin las cuales el programa sería inválido. La consecuencia de esto es que resulta mucho más fácil para una persona con mínimos conocimientos o experiencia construir combinaciones válidas usando bloques que usando texto. Dicho de forma más técnica, al usar bloques no es necesario conocer las reglas de sintaxis que permiten establecer cuáles combinaciones son válidas, pues es la forma de los bloques la que impone esa validez; y al ser la forma un elemento más concreto que reglas sintácticas abstractas, resulta mucho más intuitivo construir programas válidos usando bloques que texto. Esta faceta la denominaremos **mecanismos de expresión de programas**.

El primer entorno de bloques fue Logo Blocks (Begel, 1996). El mismo fue desarrollado en el año 1996 por el MIT Media Lab. A partir de este momento, los entornos de enseñanza de programación comenzaron a incorporar bloques como forma de construir los programas, con el fin de permitir la construcción de programas sin tener que aprender necesariamente las reglas sintácticas antes de encarar la construcción. Entre los entornos actuales basados en bloques podemos encontrar Scratch (Resnick et al., 2009), Alice (Cooper et al., 2000), App Inventor (Wolber, 2011), Pilas-Bloques (Sanzo et al., 2017) y Gobstones en su versión GobstonesJr (Martínez López et al., 2017). Los entornos basados en bloques son utilizados para principiantes de todas las edades, principalmente en escuela primaria

y secundaria, pero también en nivel universitario (Weintrop and Wilensky, 2017). Duncan et.al. (Duncan et al., 2014) realizaron una revisión de los lenguajes de enseñanza de programación usados en secundaria en Estados Unidos. En total analizaron 47 entornos, de los cuales 31 de ellos fueron lanzados después del 2010, y 28 tienen entornos basados en bloques.

Actualmente es tema de debate si es mejor enseñar a programar usando entornos basados en bloques o basados en lenguajes en texto, y si se debe hacer una transición y en qué momento. Weintrop y Wilensky (Weintrop and Wilensky, 2015) se preguntan: *"to block or not to block..."*[2]. Podemos identificar tanto ventajas como desventajas de ambos lados de esta dimensión.

Entre las ventajas, la primera y más relevante a considerar es que la expresión visual de los bloques brinda ayudas concretas directas iniciales de en qué construcciones y de qué manera es posible utilizarlos; además los colores con los que se grafica a los bloques pueden ser usados con diferentes propósitos didácticos y no solamente como forma de brindar apariencia estética. Esta expresión visual también se propaga a las categorías de bloques, que se expresan en la forma de una *caja de herramientas* (*toolbox*, en inglés) en los entornos de trabajo, evitando la necesidad de memorizar las diferentes categorías *a priori* del trabajo, y facilitando de esta forma la indagación y la apropiación de las categorías y de los diferentes bloques dentro de cada una de manera gradual. Estas dos ventajas son el principal argumento de los defensores de los entornos basados en bloques, pues los frecuentes errores de sintaxis que sufren los programadores principiantes en los lenguajes textuales los desmotivan y generan abandono. Así, esta ventaja de los entornos de bloques es la principal desventaja de los lenguajes textuales.

Sin embargo, los bloques no son la solución a todos los problemas. Su desventaja principal es su capacidad de expresión: los bloques tienen formas fijas, que ocupan espacio tanto horizontal como verticalmente, y las combinaciones que se pueden obtener son rígidas. Además su utilización requiere de la utilización de entornos de trabajo digitales, ya que el aspecto visual de los bloques es parte constitutiva de los mismos. Si bien pueden pensarse formas de trabajo con bloques sin utilizar entornos, en forma desenchufada (*unplugged*, en inglés), se pierde parte la ventaja pues se requiere que sean los docentes, o incluso los propios estudiantes, los que mantengan la forma de los bloques y sus propiedades de combinación, y esto requiere más trabajo, y puede dar lugar a errores o imprecisiones que disminuyen las ventajas mientras aumentan su costo. En este aspecto, los lenguajes textuales permiten tanto las combinaciones de elementos del lenguaje de formas más variadas (e.g. en bloques las secuencias de comandos solamente pueden aparecer en formato vertical y todas alineadas, mientras que en comandos las secuencias pueden combinarse también en forma horizontal, uno detrás de otro, y también pueden usarse diferentes indentaciones para destacar diferentes conceptos), y si bien esto puede ser complejo para principiantes, es una herramienta útil cuando ya comienzan a trascenderse los aspectos básicos. Además, los lenguajes textuales no requieren más trabajo a la hora de utilizarlos con papel y lápiz, pues las reglas sintácticas aprendidas (a mayor costo, es cierto) son las que guían la construcción. Finalmente, los errores de sintaxis, que los entornos de bloques evitan al comienzo, aparecerán invariablemente en el momento de comenzar a utilizar lenguajes textuales, por lo que puede considerarse que solamente se retrasó el problema de manejar errores de sintaxis.

Así, aparece una nueva faceta en esta dimensión: si se espera aprender programación de una forma más allá de las ideas básicas, se requiere aprender a trabajar con lenguajes textuales, y la nueva pregunta es en qué momento y de qué manera. Entonces, la faceta de la **transición de bloques** a texto cobra sentido dentro de esta dimensión cuando se

[2]"Usar bloques o no usar bloques..."; traducción libre de los autores al juego de palabras vinculado a la famosa frase de Hamlet, "ser o no ser..." (*"to be or not to be..."*).

elige comenzar con un entorno basado en bloques. En la Sección 2.3 describimos esta faceta.

Como se puede apreciar, ya en la decisión de si utilizar o no bloques, hasta qué momento, y cómo hacer la transición aparecen una multitud de pequeños aspectos que tienen incidencia didáctica, influyendo directamente en las decisiones que el docente debe tomar para organizar sus objetivos de aprendizaje y sus clases. Posponemos la discusión con nuestras opiniones para la Sección 3.

2.3. Herramientas de soporte: entorno de mundo abierto, o controlado

La tercera de las dimensiones que consideraremos está asociada a los entornos de trabajo, más que al lenguaje, pero consideraremos sus facetas pues tienen impacto en las decisiones didácticas y por lo tanto en el logro de los objetivos del curso diseñado. Cuestiones como la posibilidad de confeccionar cualquier programa sin restricciones o solamente tener disponibles actividades controladas para guiar a los estudiantes en una secuencia didáctica específica, o las herramientas asociadas para trabajar determinados conceptos o para colaborar con la evaluación del desempeño, y otras cuestiones asociadas se incluyen en esta dimensión. Por eso denominamos a esta dimensión herramientas de soporte.

En esta dimensión las facetas que se trabajarán, siguiendo a Gómez (Gómez, 2020), son la **interactividad formativa**, la **evaluación automática** de actividades, el **pasaje de bloques a texto** y el **grado de control** que el docente tiene sobre el entorno, para que el mismo pueda establecer límites al acceso que tiene el estudiante sobre las herramientas y otros aspectos a presentar en cada actividad, permitiéndole mayor flexibilidad y poder de decisión sobre los aspectos didácticos de cómo trabajar cada concepto.

La primera faceta tiene que ver con la **interactividad formativa** que brinda el entorno. Los lenguajes de programación no se aprenden de modo espontáneo sino que se adquieren y evolucionan a través de la interacción. El aprendizaje del lenguaje se ve afectado por distintas condiciones de interactividad (Gómez, 2020). Teniendo en cuenta el impacto de la interactividad formativa podemos clasificar a los entornos en dos grupos: entornos de mundo abierto o entornos de mundo controlado. Una diferencia entre los entornos abiertos y controlados es que en los entornos controlados los ejercicios de programación están predefinidos dentro del entorno mientras que en los entornos abiertos es el estudiante o el docente quien diseña (o saca de un libro) el ejercicio a programar. Los entornos abiertos están diseñados para que los usuarios puedan diseñar y programar distintos tipos de aplicaciones, como animaciones, videojuegos, apps, chatbots, etc. Los entornos controlados, a diferencia de los entornos abiertos, cuentan con un conjunto de ejercicios creados con una secuenciación predefinida en los contenidos y conceptos. Los entornos controlados suelen incluir herramientas de evaluación automática, las cuales permiten generar una interacción formativa con los estudiantes. Esto es posible ya que los programas corresponden a un problema predefinido. Este tipo de entornos pueden ser efectivos para cursos con muchos estudiantes. Ciertos entornos controlados son capaces de almacenar las soluciones enviadas, la evaluación de las mismas y el *feedback* formativo por cada estudiante.

Otra faceta importante, ya mencionada en relación a la primera, es la posibilidad del entorno de realizar **evaluación automática** de los ejercicios planificados en el curso. Esta forma de evaluación forma parte de las herramientas que son utilizadas al momento de enseñar a programar. Cuando los cursos son multitudinarios o los docentes no tienen formación previa en programación pueden ser herramientas útiles para ayudar al docente a enseñar los contenidos de manera más efectiva (Pears et al., 2007). Sin embargo, hay que prestar atención al tipo de evaluación que se realiza, ya que automatizar evaluaciones significativas es complejo. Algunas evaluaciones consisten simplemente en verificar que se cumplió el propósito del programa, sin verificar ninguna características del mismo, y podría ser que el código no siga

los conceptos que se quieren trabajar en el curso, por ejemplo. O quizás puede mirar detalles menos relevantes, pues son fáciles de automatizar. Como describimos en la sección anterior los entornos controlados tienen como característica poder contar con herramientas de evaluación automática. Pilas Bloques (Sanzo et al., 2017) y Mumuki.io (Benotti et al., 2018), por ejemplo, cuentan con herramientas de evaluación automática en mayor o menor grado. En el caso de los entornos controlados, al ser ejercicios concretos con un problema definido es posible diseñar distintas herramientas de evaluación automática. Para los entornos abiertos existen herramientas que pueden realizar evaluaciones automáticas de las habilidades en programadores principiantes que son independientes del ejercicio particular, usualmente detección código duplicado y código muerto (es decir, que nunca se ejecuta), pero como dijimos, estas características pueden ser de relevancia baja con respecto a los conceptos que se buscan impartir. Dr. Scratch (Moreno-León and Robles, 2015) es un ejemplo de este tipo de herramientas, y de lo superficial y ambigua que es la evaluación propuesta. Puntúan los proyectos según un conjunto de habilidades definidas en base a el uso de ciertas ideas de programación, sin importar que puedan contar con errores conceptuales. Dada la dificultad de poder realizar evaluación automática en entornos abiertos, las herramientas desarrolladas para intentar resolver esta tarea terminan detectando errores que no impactan directamente en la incorporación de los aspectos conceptuales. A pesar de que la evaluación automática en este tipo de entornos sea una herramienta interesante para los docentes y estudiantes, la habilidad de los docentes de poder observar el proceso de sus estudiantes parece perderse en la evaluación automática. Poder entender el proceso y progreso de un estudiante es invalorable. Por lo que es necesario pensar las herramientas de evaluación automática para que no se encarguen de la evaluación total del estudiante, sino para que faciliten y sinteticen información para el docente al momento de realizar la evaluación de cada uno de sus estudiantes (Gómez, 2020).

La tercera faceta en esta dimensión se vincula como estrategia didáctica para facilitar la **transición de bloques a texto**. Como describimos en la Sección 2.2 los entornos de bloques son utilizados mayoritariamente al momento de enseñar a programar a principiantes. De esta manera los estudiantes pueden centrarse en aspectos conceptuales dejando de lado la complejidad sintáctica de los lenguajes de texto. Para facilitar la transición de bloques a texto existen entornos que cuentan con herramientas de transición. Weintrop y Wilensky (Weintrop and Wilensky 2017) identifican dos enfoques principales para intentar lograr esta transición. Un enfoque es el pedagógico, dejando la responsabilidad de la transición a cargo del docente. El otro enfoque propuesto es diseñar entornos de enseñanza de programación que propicien y faciliten la transición de un entorno basado en bloques a un lenguaje de texto. A partir de este enfoque, la clasificación de los entornos de enseñanza de programación deja de ser binaria en cuanto a si usar entornos basados en bloques o lenguajes textuales. Estos nuevos entornos diseñados para facilitar la transición de bloques a texto se pueden clasificar en entornos de doble modalidad y entornos que traducen de bloque a texto (Weintrop and Wilensky 2017). Esta faceta se vincula con la existencia de una definición explícita del lenguaje analizada en la Sección 2.1, ya que en aquellos lenguajes con definición explícita la transición de bloques a texto resulta, usualmente, más sencilla. Los entornos de *doble modalidad* permiten al estudiante elegir si desea programar en bloques o en texto y traduce automáticamente los cambios de un lenguaje a otro. Es decir, permiten realizar una traducción bidireccional, de bloques a texto y de texto a bloques. Esto es posible porque los lenguajes de texto y de bloques tienen el mismo nivel de expresividad, a diferencia de los entornos que traducen de bloques a texto que describimos a continuación. Pencil Code (Bau et al., 2015), entre otros, es un ejemplo de entorno de doble modalidad. UNCDuino (Martínez et al., 2015), Gobstones en su modo GobstonesJr (Martínez López et al., 2017) son ejemplos de entornos que *traducen de bloques a texto*. Blockly permite traducir los bloques a diferentes lenguajes textuales como JavaScript, Python, C++ entre otros; en el caso de Gobstones, los bloques son solamente una facilidad para construir el código, pero no constituyen un

lenguaje por sí mismo, por lo que la traducción permite ir hacia la versión en texto del mismo lenguaje. Sin embargo, en estos entornos no es posible traducir texto a bloques. Esto se debe a que, en este tipo de entornos, los bloques tienen un menor nivel de expresividad que los lenguajes textuales, por lo que existen muchos más programas en forma textual que los que se pueden construir con los bloques. Por ello, al momento de querer agregar funcionalidades más complejas o tener mayor control sobre el programa es necesario utilizar los lenguajes textuales. La necesidad práctica de traducir entre lenguajes de texto y bloques pone en evidencia una característica que diferencia a distintos lenguajes de enseñanza de programación que no es evidente a simple vista: el nivel de expresividad del lenguaje. Los lenguajes basados en texto permiten al estudiante escribir libremente y por lo tanto dan mayor flexibilidad, aunque el léxico permitido en dichos lenguajes está restringido por la gramática formal del lenguaje de programación lo que dificulta su aprendizaje inicial. Generalmente los lenguajes basados en bloques son menos expresivos que los basados en texto pero esto no es necesariamente así. Es una situación causada porque los lenguajes de bloques son generalmente diseñados para aprendices más jóvenes y los lenguajes con menor nivel de expresividad también, usualmente sin haber definido explícitamente el lenguaje, como se mencionó en la dimensión del propósito y expresividad.

La última de las facetas que consideramos en esta dimensión está vinculada al **grado de control** que el docente tiene sobre el entorno, para que el mismo pueda establecer límites al acceso que tiene el estudiante sobre las herramientas y otros aspectos a presentar en cada actividad, permitiéndole mayor flexibilidad y poder de decisión sobre los aspectos didácticos de cómo trabajar cada concepto. En los entornos de mundo abierto lo usual es ofrecer la totalidad de las herramientas disponibles, sin dar ningún tipo de restricción a cuáles herramientas se ofrecerán a los estudiantes. En el otro extremo, los entornos de mundo cerrado ofrecen un conjunto restringido de actividades predeterminadas, y cada uno ofrece solamente aquellas herramientas que fueron consideradas desde el enfoque didáctico propuesto por los creadores del entorno. Sin embargo existe otra opción, que es darle al docente algún grado de control tanto sobre las actividades a ofrecer como sobre las herramientas que cada actividad ofrecerá y la configuración de las características de interfaz que tendrá el estudiante al interactuar con dicha actividad en el entorno. Esta última opción está poco explorada, y la gran mayoría de los entornos, tanto de mundo abierto como de mundo cerrado quedan cortos en este punto, enfrentando al docente con la decisión de, o bien de usar un entorno de mundo abierto y tratar de trabajar en clase la confusión que genera en los estudiantes iniciales tener todas las herramientas desde el comienzo, sin conocimientos para comprender la gran mayoría de ellas, o bien usar un mundo cerrado, con todas las decisiones didácticas tomadas, y acomodar su curso a tales decisiones. Una alternativa para esto es usar más de un entorno durante el curso, comenzando con uno de mundo cerrado y más tarde pasar a uno de mundo abierto. Existe una opción más: usar un entorno que ofrezca cierto control, y aunque tiene el costo adicional de tener que diseñar actividades para el curso, puede ser beneficioso en el hecho de que se ajuste el curso a decisiones didácticas propias y planificadas, y también tiene el atenuante de que tal tarea se hace una primera vez y luego se puede aprovechar en varios dictados.

3. Discusión: ¿cómo elegir?

Al momento de elegir un entorno para dictar un curso inicial de programación, la propuesta es tener en cuenta las dimensiones trabajadas en este artículo, y contrastarlas contra los objetivos del curso, las intenciones de cómo (y si) continuar aprendiendo programación de forma profesional, y las características de los estudiantes que tomen el curso. Cada docente deberá realizar estas consideraciones para elegir de la manera más adecuada, dependiendo de sus objetivos y del nivel en el que desea impartir el curso, sin necesidad de copiar modas o elegir por descarte o

desconocimiento, sin imponer al curso características no buscadas o que no son consecuentes con los objetivos del mismo. Por ejemplo, si el objetivo es enseñar los primeros pasos de programación en nivel inicial o primeros años de primaria, al considerar bloques debería refinarse para tener en cuenta si la naturaleza de los mismos se basa en nombres textuales o íconos, ya que en ese nivel no está afianzada la lectoescritura; sin embargo, este artículo invita a reflexionar sobre dimensiones generales, que se explicitan para facilitar el análisis, y así tal especificidad está más allá del alcance de este trabajo.

Para facilitar el análisis de diferentes entornos diseñamos una tabla de doble entrada, que tiene a los entornos en las columnas y a las dimensiones en las filas, y en cada caso indica si el entorno tiene o no la dimensión considerada. Por cuestiones de espacio elegimos solamente los entornos de mayor difusión y uso a nivel nacional, pero esta tabla podría extenderse para otros entornos. La tabla se presenta en la Tabla 1.

Como ejemplo de aplicación podemos mencionar las elecciones de algunos cursos actuales, y contrastar cuáles fueron los criterios utilizados para elegir los entornos que en ellos se utilizan. Estos criterios existen desde antes de la sistematización en dimensiones que proponemos, y los utilizamos como base para construir esta propuesta, por lo que sirven como ejemplos de la aplicación de la misma.

El primer ejemplo a considerar son los dos cursos de la iniciativa Program.AR llamados “La Programación y su Didáctica”. Estos cursos se dictan desde la Fundación Sadosky para capacitar a grupos universitarios de Universidades de todo el país, que luego capacitan a docentes de secundaria, quienes finalmente los aplicarán con sus estudiantes. Los criterios establecidos para los entornos en estos cursos son los siguientes:

- deben proveerse entornos amigables para el estudiante inicial, y en lo posible no abrumarlo desde el comienzo con multitud de opciones, sino que le permitan al docente ir incrementando gradualmente las opciones y herramientas disponibles,
- debe utilizarse más de un entorno, para favorecer la centralidad de los conceptos y prácticas desarrollados, al mostrar que las ideas pueden transponerse de uno a otro con gran simplicidad (más allá de las diferencias en ubicación o denominación que cada concepto pueda tener en cada entorno diferente) y combinar dimensiones de entornos diferentes para conseguir propósitos a veces antagónicos,
- deben proveer cierto grado de control sobre las actividades iniciales sin limitar las posibilidades de exploración de otras características, y
- deben permitir tanto la indagación en actividades iniciales guiadas como la producción de aplicaciones propias sencillas pero interesantes.

En base a estos criterios, para el curso “La Programación y su Didáctica 1” se eligieron tres entornos: Pilas Bloques, Scratch y Alice. Para el curso “La Programación y su Didáctica 2” se eligieron dos entornos de programación: GobstonesWeb (en su modo GobstonesJr y GobstonesTeacher) y App Inventor. Puede contrastarse, a partir de las dimensiones analizadas, que estos entornos en conjunto cubren de forma amplia los requerimientos establecidos.

Otro ejemplo para analizar es el caso de cursos iniciales de programación a nivel de formación universitaria. En ellos el criterio a privilegiar está asociado al pasaje a lenguajes y contenidos más avanzados en cursos posteriores. Por tal razón se suele comenzar con entornos que o bien trabajan directamente con texto o bien permiten una modalidad dual. Además, también se privilegia la facilidad para pasar a la utilización de elementos abstractos y la trasposición de conceptos a otros paradigmas. En base a esto, en el curso inicial de la Universidad Nacional de Rosario se utiliza Racket, y en los cursos iniciales de las Universidades de Quilmes y de Hurlingham se utiliza Gobstones, comenzando con una fase de indagación en bloques, y luego incorporando la programación en modo textual, y también incluyendo

Dimensiones y Facetas			Entornos									
			Alice	App Inventor	Chatbot	Gobstones	Pencil Code	Pilas Bloques	Python	Racket	Scratch	UNC Duino
Propósito y expresividad	Propósito	¿Específico para enseñar a programar?	Sí	Sí	Sí	Sí	Sí	Sí	No	No	Sí	Sí
	Universo de discurso	¿Universo con elementos concretos?	Sí	Sí	Sí	Sí	Sí	Sí	No	No	Sí	Sí
		¿Favorece el pasaje a elementos abstractos?	No	No	No	Sí	No	No	Sí	Sí	No	No
	Paradigmas	¿Habilita el pasaje simple a otros paradigmas	No	No	No	Sí	No	No	Sí	Sí	No	No
Formas de Construcción		¿Se puede programar con bloques?	Sí	Sí	No	Sí	Sí	Sí	No	No	Sí	Sí
		¿Se puede programar con texto?	No	No	Sí	Sí	Sí	No	Sí	Sí	No	Sí
Herramientas de soporte	Abiertos o controlados	Permite actividades controladas	No	No	Sí	Sí	No	Sí	No	No	No	No
		Permite actividades libres	Sí	Sí	Sí	Sí	Sí	No	Sí	Sí	Sí	Sí
		¿Evaluación automática?	No	No	Sí	No	No	Sí	No	No	No	No
	Transición de bloques a texto	¿Tiene versión textual del mismo lenguaje?	No	No	Sí	Sí	No	No	Sí	Sí	No	No
		¿Doble modalidad?	No	No	No	Sí [1]	Sí [2]	No	No	No	No	No
		¿Compila de bloques a algún lenguaje textual?	Sí	No	No	No	No	Sí [3]	No	No	No	Sí

1. En Gobstones se puede programar tanto en un entorno de bloques, como en un entorno de texto, pero no existe una comunicación directa entre ambos.

2. La doble modalidad se da con un subconjunto de un lenguaje de programación diferente (Python).

3. Pilas Bloques tiene la capacidad de generar código Javascript que implementa el mismo comportamiento, pero no está actualmente disponible para su uso desde el entorno.

Tabla 1: Tabla de análisis comparativo entre entornos

sus características extendidas de trabajo con estructuras de datos simples.

En algunas escuelas secundarias de la provincia de Córdoba, a partir de los cursos de formación docente

implementados por el grupo de extensión universitaria UNC++ (Martínez et al., 2016), se implementaron secuencias didácticas donde se combinaron el uso de Alice, UNCDuino y Chatbot. La propuesta combinó programar animaciones en Alice utilizando bloques, programar kits de robótica con UNCDuino, utilizando la transición de bloques a lenguajes profesionales, y programar bots conversacionales en Chatbot utilizando lenguajes textuales. Explorar la programación en mundos abiertos como los que ofrece Alice o UNCDuino o en contextos más controlados con evaluación automática como permite Chatbot. La articulación de los entornos fue pensada para abordar los diferentes contenidos en base a los elementos concretos que ofrecen ambos entornos, pero aprovechando las características que los diferencian, y como estrategia didáctica para poder dar cuenta que los conceptos a desarrollar son independientes de los entornos.

Como se puede ver, la elección de los entornos de programación no se limita solamente a elegir entre algunas opciones de moda, y si se quiere realizar un trabajo acorde con los objetivos del curso, debe considerarse con cuidado. En este sentido, nuestro aporte en la clasificación de dimensiones puede ser de gran utilidad.

4. Conclusiones

Al diseñar un curso inicial de programación debe recordarse la paradoja discutida al inicio sobre la elección del lenguaje/entorno de programación: el lenguaje/entorno no es lo verdaderamente importante, pero sin embargo es muy importante. Por esa razón, conocer adecuadamente las características de las diferentes propuestas y las implicaciones y consecuencias que tiene elegir una u otra para facilitar el logro de los objetivos propuestos para el curso que deseamos impartir resulta fundamental.

Este trabajo contribuye a tal conocimiento, al proponer una clasificación de características según un conjunto de dimensiones y facetas. Las dimensiones propuestas sirven para tomar decisiones de qué lenguaje/entorno utilizar para diferentes niveles, tanto primaria, como secundaria, como universidad, y tanto cuando el objetivo es dar computación como cultura general, o como formación profesional; permiten, así, juzgar mejor un entorno con respecto a los objetivos de un curso específico. Esto evidencia la generalidad del espacio de dimensiones ofrecido, contribuyendo a la toma de decisiones a la hora de diseñar cursos de computación, aportan a lo que hace falta tener en cuenta para mejorar el logro de los objetivos buscados.

Bibliografía

Abadi, M. and Cardelli, L. (2012). A Theory of Objects. Springer Science & Business Media, 396 págs. ISBN 978-1-4612-6445-3.

Bau, D., Bau, D.A., Dawson, M. and Pickens, C.S. (2015). Pencil code: block code for a text world. In Procs. 14 Int. Conf. Interaction Design and Children, pp.445–448. ACM.

Begel, A. (1996). LogoBlocks: A graphical programming language for interacting with the world. Reporte técnico. Electrical Eng. and Computer Science Department, MIT, Boston, MA. Disponible en: https://andrewbegel.com/mit/begel-aup.pdf.

Benotti, L., Aloi, F., Bulgarelli, F., and Gómez, M. J.(2018). The Effect of a Web-based Coding Tool with Automatic Feedback on Students' Performance and Perceptions. In Procs. of 49th ACM Tech. Symp. on Computer Science Ed., pp.2–7. ACM.

Benotti, L., Martínez, M. C., and Schapachnik, F. (2017). A tool for introducing computer science with automatic formative assessment. IEEE Transactions on Learning Technologies, 12(2):179–192.

Bird, R.J. and Wadler, W. (1988). An introduction to Functional Programming. CAR Hoare Series, Prentice Hall. ISBN 0-13-484189-1.

Black, A. and Palsberg, J. (1994). Foundations of Object-Oriented languages. ACM SIGPLAN Notices 29(3):3–11. ACM.

Cooper, S., Dann, W., and Pausch, R. (2000). Alice: a 3-D tool for introductory programming concepts. Journal of Computing Sciences in Colleges, 15(4):107–116.

Dijkstra, E. (1968). Go To statement considered harmful. Communications of the ACM, 11(3):147–148. ACM.

Duncan, C., Bell, T. and Tanimoto, S. (2014). Should your 8-year-old learn coding? Procs.of 9th Workshop in Primary and Secondary Comp. Ed., pp.60–66. ACM.

Felleisen, M., Findler, R. B., Flatt, M., Krishnamurthi, S., Barzilay, E., McCarthy, J., and Tobin-Hochstadt, S. (2015). The Racket manifesto. In 1st Summit on Advances in Prog. Langs., p.113–128. Schloss Dagstuhl-Leibniz-Zentrum für Informatik.

Gómez, M. J. (2020). Aspectos de adquisición de lenguaje en la enseñanza de programación. Tesis de doctorado. FAMAF, UNC, Córdoba, Argentina.

Hughes, J. (1989). Why Functional Programming matters? The Computer Journal, 32(2):90–107. ACM.

Martínez, C., Gomez, M. J., and Benotti, L. (2015). A comparison of preschool and elementary school children learning computer science concepts through a multilanguage robot programming platform. In Procs. of 2015 ACM Conference on Innovation and Technology in Computer Science Education, pp.159–164. ACM.

Martinez, M. C., Gomez, M. J., Moresi, M., and Benotti, L. (2016). Lessons learned on computer science teachers professional development. In Procs. of 2016 ACM Conf. on Innovation and Technology in Computer Science Education, pp. 77–82. ACM.

Martínez López, P. E., Bonelli, E. A., y Sawady, F. A. (2012). El nombre verdadero de la programación. Una concepción de la enseñanza de la programación para la sociedad de la información. En Anales del 10mo Simposio de la Sociedad de la Información (SSI'12), dentro de las 41ras Jornadas Argentinas de Informática (JAIIO '12), pp.1–23, septiembre 2012. ISSN 1850-2830.

Martínez López, P. E. (2013). Las bases conceptuales de la programación. Una nueva forma de aprender a programar. EBook (1era ed.). La Plata, el autor. ISBN: 978-987-33-4081-9. http://www.gobstones.org/bibliografia/Libros/BasesConceptualesProg.pdf.

Martínez López, P. E., Ciolek, D., Arévalo, G., and Pari, D. (2017). The GOBSTONES method for teaching computer programming. XXV Simposio de Educación Superior en Computación (SIESC'17), dentro de la XLIII Conferencia Latinoamericana de Informática (CLEI'17), pp.1–9. IEEE, ISBN 978-1-5386-3057-0.

Martínez López, P.E., Sanzo A., y Schapachnik, F. (2019). Hacia una didáctica de la programación para la secundaria argentina. Actas II Jornadas Argentinas de Didáctica de la Programación (JADiPro II). EBook, Acosta et.al. (editores) pp.88–99. UNRC.

Moreno-León, J., and Robles, G. (2015). Dr. Scratch: A web tool to automatically evaluate Scratch projects. In Procs. of the workshop in primary and secondary computing education, pp.132–133. ACM.

Pears, A., Seidman, S., Malmi, L., Mannila, L., Adams, E., Bennedsen, J., and Paterson, J. (2007). A survey of literature on the teaching of introductory programming. In Working group reports on ITiCSE on Innovation and technology in computer science education, pp.204–223. ACM.

Resnick,M., Maloney, J., Monroy-Hernández, A., Rusk, N., Eastmond, E., Brennan, K., Millner, A., Rosenbaum, E., Silver, J., Silverman, B. and Kafai, Y. (2009). Scratch: programming for all. Communications of the ACM, 52(11):60–67. ACM.

Sanzo, A., Schapachnik, F., Factorovich, P., and Sawady O'Connor, F. A. (2017). Pilas Bloques: A scenario-based children learning platform. 2017 Twelfth Latin American Conference on Learning Technologies (LACLO), pp.1–6. IEEE.

Solomon, C. J. and Papert, S. (1976). A case study of a young child doing turtle graphics in LOGO. In Procs. of June 7-10, 1976, National Computer Conference and Exposition (AFIPS '76), pp.1049–1056. ACM.

Watt, D.A. and Findlay, W. (2004). Programming language design concepts. John Wiley & Sons. ISBN 978-0-470-85320-7.

Weintrop, D., and Wilensky, U. (2015). To block or not to block, that is the question: students' perceptions of blocks-based programming. Procs. of 14th international conference on interaction design and children, pp.199–208. ACM.

Weintrop, D., and Wilensky, U. (2017). Between a block and a typeface: Designing and evaluating hybrid programming environments. Procs. of 2017 conference on interaction design and children, pp.183–192. ACM.

Wolber, D. (2011). App Inventor and real-world motivation. Procs. of 42nd ACM technical symposium on Computer science education, pp.601–606. ACM.

SESIÓN 4:

ENSEÑANZA DE PROGRAMACIÓN CON ARTEFACTOS COMPUTACIONALES

Moderador: *Dr. Francisco Bavera (UNRC)*

Uso de domótica como herramienta para la enseñanza de programación

Valentín Basel

Simulador de sumo para robótica educativa

Gonzalo Zabala y Ricardo Morán

Minirobots App: una aplicación educativa para aprender a programar jugando

Francisco Prieto, Fermín de Fiore, Paula Tristán y Laura Felice

AMULEN: robot educativo soberano

Carlos Maximiliano Correa, Pablo Leonel Etcheverry y Ariel Ferreira Szpiniak

Uso de domótica como herramienta para la enseñanza de programación

Valentín Basel
valentinbasel@gmail.com
Universidad Nacional de Córdoba

Resumen

La emergencia del Núcleo de Aprendizaje Prioritario (NAP) de programación y robótica, aprobado mediante la resolución del Consejo Federal de Educación RCFE N° 343-18 en el año 2018 dio pie a profundizar las discusiones con respecto al uso de robots educativos para la enseñanza, y en general en enseñanza de programación. De las herramientas tecnológicas que se están adquiriendo en las escuelas públicas y privadas, podemos ver una gran proliferación de equipos pensados para construir robots tipo "carrito" o tribots (robots de tres ruedas), así como kits de tipo constructivos. Sin embargo la proliferación de dispositivos electrónicos baratos, con capacidad de conexión WIFI y diseñados dentro del paradigma de IoT (Internet of Things), nos permite pensar en sistemas de domótica educativa de bajo costo. La domótica educativa permite la enseñanza de programación a través de experiencias significativas porque involucra dispositivos que interactúan con el medio ambiente real de los estudiantes y no solamente simulaciones de situaciones con robots de juguetes.

En este artículo presentamos Domoticaro, un proyecto de hardware de bajo costo y especificaciones libres bajo licencia GNU GPL V3.0, que incluye software libre distribuido bajo licencia GNU GPL V3.0. Fue diseñado para ser fabricado con herramientas caseras y así permitir su construcción a pequeña escala en escuelas o espacios no formales de aprendizaje como clubes de ciencia, o bibliotecas populares. Consta de una interfaz de hardware basado en la placa de desarrollo Weemos d1 mini con electrónica de potencia capaz de conectarse a internet mediante WIFI, así como las librerías necesarias para su programación mediante lenguaje Python y tiene como finalidad ser una herramienta para la enseñanza de programación.

Palabras clave: domótica educativa, software libre, hardware libre, constructivismo, programación.

1. Introducción

El uso de robots educativos como herramientas para enseñanza de programación, tiene sus orígenes en el trabajo de Seymourd Papert con el grupo de inteligencia artificial del M.I.T y la creación del lenguaje LOGO, así como del robot "tortuga" que servía como plataforma de trabajo con dicho lenguaje. Las limitaciones de costos y capacidades del hardware de la época, llevaron al desarrollo de una versión de LOGO que aprovechara las posibilidades gráficas

de las microcomputadoras de la década de los ochenta, y así evitar el uso de los costosos robots físicos que se habían diseñado en el MIT.

Sin embargo, a partir de la aparición en el año 2003 del proyecto ARDUINO, hubo una proliferación de equipos de robótica educativa de bajo costo basados en los microcontroladores AVR de la empresa Atmel usando las librerías ARDUINO, lo que permitió retomar algunas de las premisas que se usaron originalmente con LOGO durante los ochenta.

En la actualidad, los robots educativos son generalmente diseñados como plataformas móviles capaces de interactuar con el entorno mediante sensores y actuadores, y son usados para diversas actividades como pueden ser las competencias de sumo robot, seguidores de línea o evasores de obstáculos. No obstante esta práctica de enseñanza de programación, si bien muy extendida y de la cual hay hasta competencias nacionales como la roboliga, está limitada a simular situaciones que puedan generar una respuesta que estos equipos de robótica puedan afrontar, teniendo en cuenta las limitaciones de costos y complejidades inherentes a todo sistema físico-electrónico.

Asumir una perspectiva desde el enfoque constructivista como la herramienta teórica para tratar de comprender los procesos de aprendizajes, y sobre todo los procesos concernientes a la enseñanza de un lenguaje de programación, implica la capacidad de producir un aprendizaje significativo en los estudiantes al trabajar sobre entorno reales que tengan algún sentido para ellos y su comunidad, entendiendo que para la teoría sociohistórica los procesos psicológicos superiores se originan en la vida social, en otras palabras, la participación de los sujetos en las actividades con otros (Baquero, 1996, p. 34).

2. El constructivismo de Vigotsky

Todo proyecto de enseñanza debe estar soportado sobre un marco teórico pedagógico que lo fundamente y sirva de sostén para les elecciones metodológicas que se implementen en la creación de una propuesta curricular específica, en ese sentido, el uso de robots como herramientas de mediación en el proceso de enseñanza de programación requiere una atención especial. Tomar como marco de referencia al constructivismo vigotskiano implica entender que todo proceso de aprendizaje debe ser abordado desde una perspectiva social que tenga en cuenta las realidades propias de la comunidad donde se esté trabajando, así como generar situaciones didácticas significativas para los estudiantes (Panizza, 2003). Podemos decir que las teorías socioculturales acuden a la participación para explicar el aprendizaje, donde la idea fundamental radica en que los individuos aprenden en el marco de prácticas sociales. Abandonando con una intención explicita las concepciones individualistas de la psicología y filosofía, en las que el individuo es la unidad de análisis y el foco de investigación (Radford, 2017). En ese sentido, tomamos conceptos de la teoría de la objetivación planteados por Luis Radford donde postula que:

> La educación en general y la enseñanza y aprendizaje en particular no tratan de saberes únicamente. La educación en general y la enseñanza y aprendizaje en particular tratan de saberes y de seres. En términos más específicos, en la enseñanza y aprendizaje deben estudiarse tanto los conocimientos en juego (es decir, el conociendo o "knowing" de los alumnos), como la formación del alumno en tanto que sujeto humano (es decir, el volviéndose o "becoming", esto es, la transformación perpetua del sujeto). (Radford, 2014, p. 135)

Esta perspectiva plantea el objetivo de la educación (en el caso de Radford la enseñanza de las matemáticas)

como un esfuerzo político, social, histórico y cultural cuyo fin es la creación de individuos éticos y reflexivos que se posicionan de manera crítica en prácticas matemáticas constituidas histórica y culturalmente.

3. Domótica como herramienta constructivista

Asumimos que la tecnología no es neutral (Letzen et al., 2019), las elecciones técnicas sobre el hardware y software que usemos en una propuesta educativa no es trivial, por el contrario, implican un posicionamiento político sobre su uso y apropiación por parte la comunidad educativa de las herramientas tecnológicas sugeridas. En ese sentido, el desarrollo de una propuestas basada en hardware y software libre (Basel, 2020; Chavarría, 2011) tiene especial importancia para pensar desde la soberanía tecnológica (Haché, 2014).

La domótica se trata de automatizar y controlar mediante sistemas de electrónica digital una casa o cualquier ambiente (un jardín, un invernadero, etc.) y así lograr mejorar las condiciones de habitabilidad del mismo (de la Colina, 2004). Si bien estos sistemas pueden ser simulados para fines didácticos, gracias al abaratamiento de dispositivos electrónicos específicos para IoT como el chip ESP8266 (hardware integrado con conexión WiFi y compatible con el protocolo TCP/IP) es factible diseñar y utilizar equipos que permitan crear situaciones reales en los ambientes físicos donde se desenvuelven los estudiantes.

El hardware propuesto para su uso como herramienta de enseñanza de programación es un desarrollo propio, basado en la plataforma IoT de código abierto Weemos d1 mini y que tiene algunas de las siguientes características:

- Velocidad: 80MHz/160MHz
- Memoria Flash: 4M bytes
- Tensión funcionamiento: 3.3V
- Entradas y salidas digitales: 11
- Entradas analógicas: 1

Sin embargo esta placa de desarrollo de bajo costo no posee electrónica de control capaz de manipular los sistemas eléctricos ni sensores para poder usar en domótica, así que fue necesario desarrollar una placa de circuito integrado específica para poder usar las placas Weemos D1 mini en domótica.

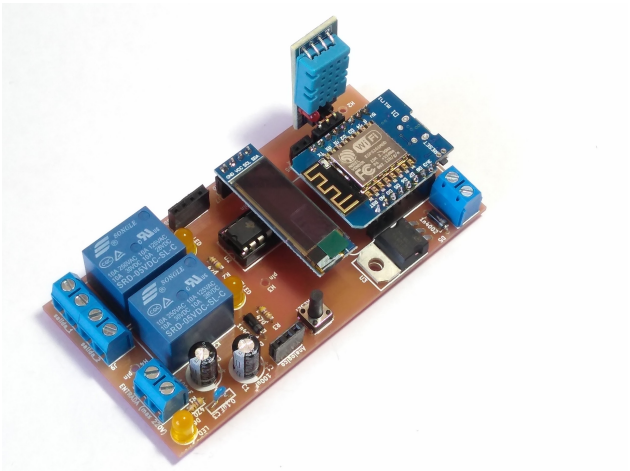

Figura 1: Placa Domoticaro basada en el hardware Weemos d1 mini

El desarrollo del circuito integrado específico para domótica, que de cuenta de alguna prerrogativas planteadas

por las comunidades de desarrollo de hardware libre (González et al., 2003), implicó tomar decisiones con respecto a los actuadores y sensores que se usarán, así como pensar en la facilidad de construcción con procesos semi-industriales (como la técnica de serigrafía) o directamente caseros, para facilitar su adquisición y construcción en las comunidades. En ese sentido se eligió los siguientes actuadores y sensores para la placa de domótica educativa:

Actuadores:

- 2 salidas a relés con capacidad de manejo de hasta 220V y 10A.
- 1 salida para led RGB.
- conexión USART para puerto serie (Rx, Tx).
- una pantalla OLED de 128x32 pixeles.

Sensores:

- 1 sensor de temperatura DTH11.
- 1 sensor analógico.

Los proyectos de IoT aprovechan las ventajas de los sistemas de WIFI para poder tener pequeños dispositivos que controlen un ambiente en particular, a diferencia de sistemas centrales que controlan todos los dispositivos de una casa, y de esta forma tener un ecosistema de dispositivos más o menos independientes que interactúan entre sí y con el usuario. Si bien es factible usar equipos de altas prestaciones como las computadoras Raspberry pi (Realinho, 2015) que permiten tener las ventajas de un microcontrolador gracias a su GPIO sumada a la potencia de una computadora completamente funcional, su coste superior puede ser un inconveniente para trabajar con sistemas baratos para desarrollar en una escuela o espacio no formal de aprendizaje.

El hardware de domótica propuesto, está diseñado para controlar un ambiente, que puede ser la habitación de un estudiante, el espacio de cursado o un pequeño invernadero, y poder interactuar con otros dispositivos mediante WIFI y usando Python como lenguaje de programación.

Al ser un proyecto de software y hardware libre, toda la documentación y planos de fabricación, así como librerías y firmware de control, se pueden encontrar en el siguiente repositorio:

https://gitlab.com/valentinbasel/doctorado

4. El lenguaje Python como herramienta de enseñanza e instrumento mediador

Python es un lenguaje de programación interpretado cuya filosofía está orientada a la legibilidad de su código y la facilidad de su escritura al ser un lenguaje de alto nivel (van Rossum et al., 2001), se trata de un lenguaje de programación multiparadigma, interpretado, dinámico y multiplataforma con una gran cantidad de módulos que le permiten ser uno de los lenguajes más populares para múltiples usos, entre ellos como primer lenguaje de programación textual en la Enseñanza Secundario (Acosta et al., 2019; Estévez et al., 2014; García Monsálvez, 2017).

La elección de Python como lenguaje para enseñanza de programación también llevó a la elección de Micropython para el desarrollo del firmware que se usa en el hardware Weemos D1 mini, con el fin de unificar los tipos de lenguajes que se usan del lado de la placa de domótica y del lado de la computadora. De esta forma, las placas de domótica pueden ser programadas de dos maneras, como un hardware independiente, mediante la programación de firmware específico con Micropython, y por otro lado como hardware controlado mediante un pequeño servidor web que implementa

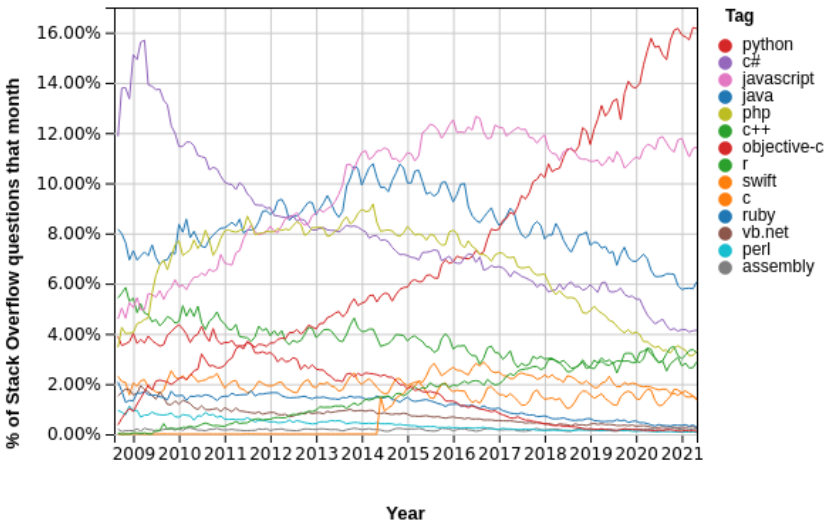

Figura 2: Crecimiento de las consultas con respecto a Python en el popular sitio stack overflow

peticiones GET y una API específica para poder interactuar con una computadora.

Este esquema de trabajo, donde podemos controlar las placas de domótica para prender o apagar dispositivos eléctricos (lámparas, ventiladores, calefactores etc.) así como recibir notificaciones de los sensores de temperatura y humedad, permite integrar las potencialidades de Python con las capacidades de interactuar con un medio físico real y concreto, como puede ser la habitación de un estudiantes o el aula de la escuela (Revilla, 2018), además de permitir usar Chatbots computacionales (Echeveste et al., 2012; Janarthanam, 2017) como interface para el hardware, creando un entorno desde donde los estudiantes pueden desarrollar software de control para un ambiente dado, usando sus celulares con la app de Telegram, y así poder controlar dispositivos eléctricos o recibir reportes de estado de los sensores de las placas de domótica.

El uso de chatbots basados en Telegram, permite simplificar la conexión a internet del dispositivo de domótica, dado que la API de Telegram para chatbots solamente necesita habilitación en el puerto 80 de una red doméstica, sin tener que administrar ni configurar un router WIFI (puertos de conexión, ip pública/privada) , siendo una ventaja cuando no se tiene acceso a permisos de administración en la red desde donde se quiere trabajar. Por otro lado, los chatbots computacionales son una herramienta interesante para poder dar cuenta de conceptos avanzados sobre procesamiento de lenguaje natural (Benotti et al., 2012) así como crear instancias de desarrollo más complejas que los ejercicios tradicionales que suelen hacerse con robots móviles (Barrera Lombana, 2015), dando la posibilidad de trabajar conceptos como variables, saltos condicionales, repeticiones y funciones en un entorno escalable en complejidad.

Si bien el hardware de domótica puede ser programado de forma autónoma para usar sus actuadores y sensores sin necesidad de interactuar con una computadora, es en la integración con chatbots donde más actividades se pueden hacer con los estudiantes, permitiendo la creación de situaciones didácticas genuinas aprovechando la interacción entre celulares, hardware de domótica y computadoras.

Mediante un chatbot los estudiantes pueden explorar las posibilidades de interactuar con una entidad artificial y reflexionar sobre sus propios procesos de pensamiento, o en palabras de Papert: “al enseñarle a pensar a la computadora, los chicos se embarcan en una exploración de modo que ellos mismos piensan. La experiencia puede ser embriagadora:

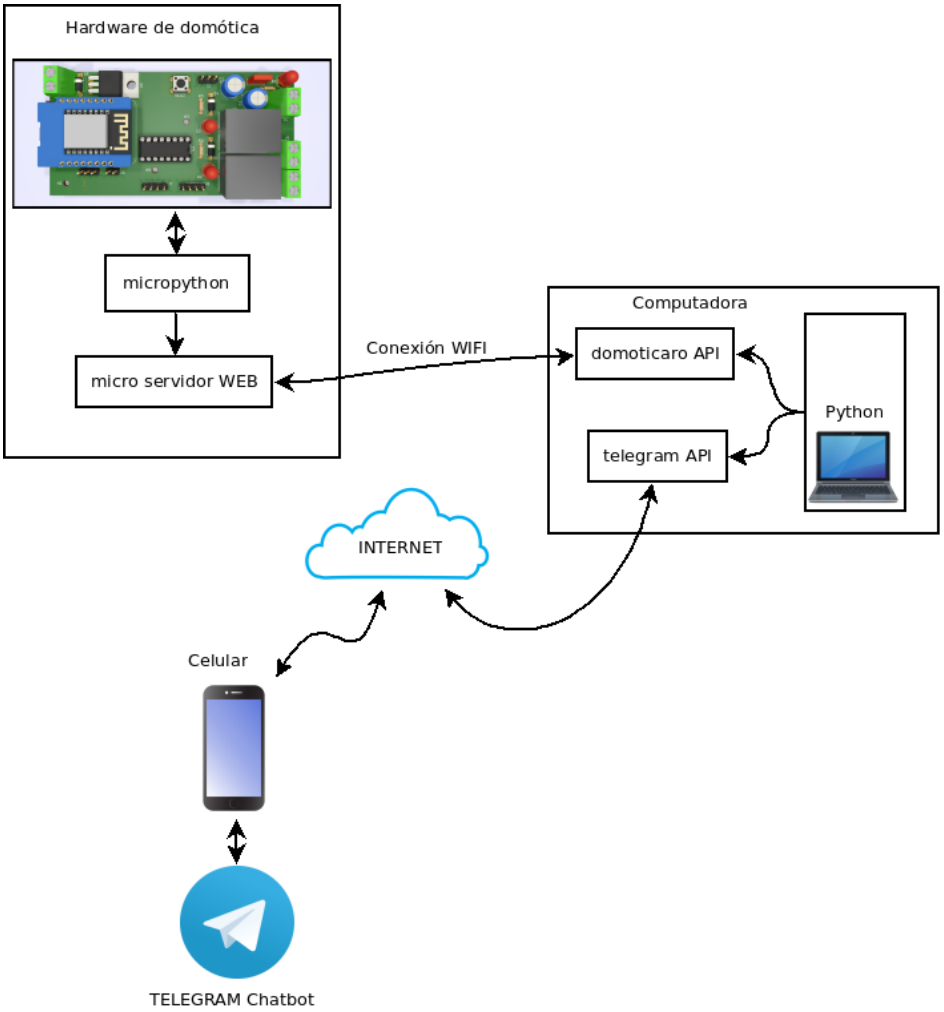

Figura 3: Esquema de conexión del hardware con chatbots basados en Telegram

```python
#!/usr/bin/env python3
# -*- coding: utf-8 -*-

import domoticaro

habitacion = domoticaro.iniciar("weemos")
habitacion.iniciar("http://192.168.0.141/")


for t in range(4):
    print("el valor del sensor analogico es: ", habitacion.Analogico.leer())
    print("el valor del sensor humedad es: ", habitacion.Humedad.leer())
    print("el valor del sensor temp es: ", habitacion.temperatura.leer())
    print("estado del RELE_1: ", habitacion.Rele_1.conmutar())
    print("estado del RELE_2: ", habitacion.Rele_2.conmutar())
habitacion.cerrar()
```

Figura 4: Ejemplo de código fuente Python para controlar una placa Domoticaro

pensar sobre el pensamiento convierte al niño en epistemólogo" (Papert, 1987, p. 33).

5. Una experiencia en bibliotecas populares

Domoticaro es un hardware experimental, y desarrollado en concreto como herramienta de investigación en el marco del Doctorado en educación en ciencia básica y tecnología de la UNC, diseñado para ser usado en pequeña escala, en espacios comunitarios de aprendizaje (Martín, 2017) como las bibliotecas populares de la ciudad de Córdoba.

En ese sentido, durante septiembre del 2021 se llevo a cabo 3 experiencias de uso del hardware, trabajando con las bibliotecas populares **la Brújula Barrial, Biblioteca Popular de Alberdi y Biblioteca Juana Manuela Gorriti**, con cursos reducidos de cinco estudiantes por comisión para poder mantener un ambiente con distanciamiento social, a causa de las regulaciones del *Distanciamiento Social, Preventivo y Obligatorio* (DISPO).

Durante los cursos se trabajó nociones básicas de programación con el lenguaje Python (repeticiones, variables, saltos condicionales), así como soldar componentes en las placas con estudiantes que no tenían formación previa en programación ni en electrónica.

Figura 5: Mesa de trabajo en la biblioteca La brújula barrial

Si bien fue una experiencia limitada a causa de las restricciones sanitarias, la época del año, el presupuesto y tiempo destinado a los talleres, la experiencia mostró algunos resultados interesantes para poder retro alimentar el proceso de desarrollo del hardware y software.

En principio los estudiantes trabajaron con la versión de LOGO desarrollada para Python (modulo Turtle) para luego diseñar un chatbot computacional sencillo y controlar luces navideñas y de colores conectadas al hardware Domoticaro. Trabajar directamente con dispositivos eléctricos reales permitió a los estudiantes entender que cuando escribían una orden en Python y se activaba un relé en la placa Domoticaro, esta prendía luces de la habitación, concepto sencillo que parecía no ser comprendido en su totalidad cuando solamente se activaba el led indicador en la placa y que sin embargo genero un gran interés cuando se vio en la practica como la sala se iluminaba por completo con las luces de colores. Por otro lado, el la integración de los celulares a la actividad, mediante los chatbots de Telegram, pareciera

ser una actividad que potencia la apropiación de los contenidos, al ser un dispositivo muy conocido por los estudiantes. Sin embargo, también agrego complejidad a la logística de trabajo en clases, dado que no todos los estudiantes tenían un dispositivo en condiciones de trabajar (falta de espacio para instalar Telegram por ejemplo) o no la llevaban a clases y al ser trabajo individual a causa del DISPO, no podían continuar, lo que llevo a usar un emulador de chat para poder seguir con la propuesta de la clase.

A nivel de armado de las placas, se noto algunas consideraciones a tener en cuenta para el desarrollo de un nuevo modelo, dado que algunos pads de soldadura son demasiado chicos para un estudiante con poco dominio de un soldador de tipo cautín, lo que dio como resultado algunos problemas con soldadura fría, poco o demasiado estaño y pistas levantadas por el exceso de calor al aplicar mucho tiempo la punta del cautín.

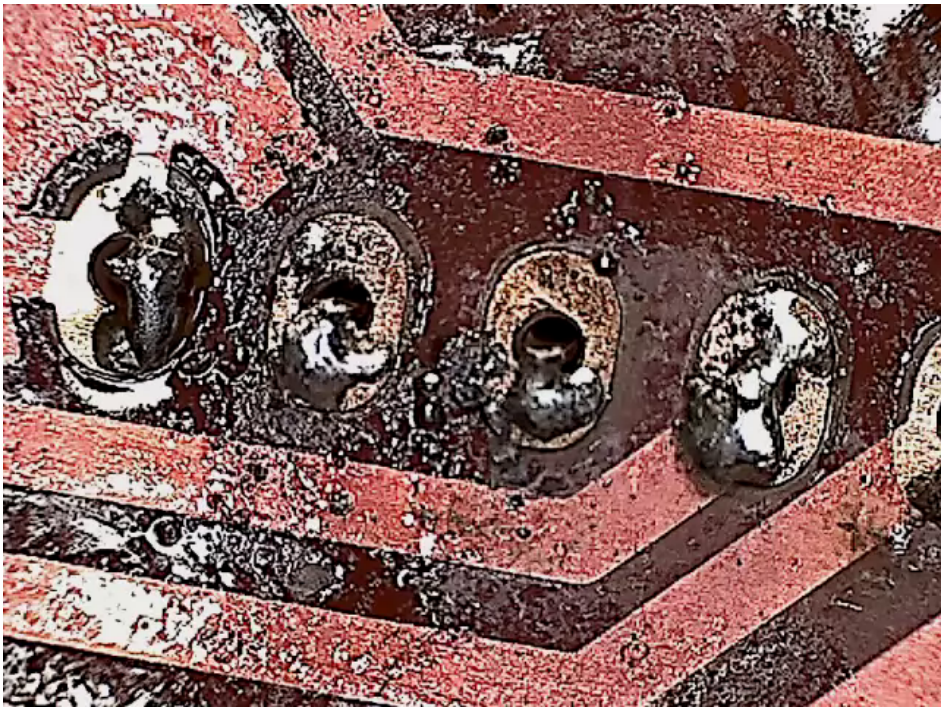

Figura 6: Poco estaño y “escorias” al rededor de los pads donde esta montado el integrado UNL2803

6. Consideraciones finales

El uso de dispositivos electrónicos como robots o electrónica de propósito general (por ejemplo ARDUINO) requiere no solo de decisiones didácticas sobre la pertinencia de su uso, si no también implica un compromiso con respecto su costo de adquisición, la capacidad de reparación y mantenimiento, así como las comunidades que puedan dar soporte técnico a los docentes (Adell y Bernabé, 2007) y la infraestructura logística necesaria para poder trabajar con esos dispositivos en el aula. Gracias al abaratamiento del hardware específico para sistemas de IoT y la gran proliferación de comunidades que crean proyectos con estos componentes, es factible pensar en el uso de domótica como herramienta didáctica para la enseñanza de programación.

La posibilidad de poder fabricar en pequeña escala, con herramientas caseras y componentes de bajo costo, así como un hardware lo suficientemente robusto para soportar el “mal trato“ de manos inexpertas, permite “abrir la caja

negra" que es el hardware y crear situaciones de empoderamiento.

La domótica permite generar situaciones de interés para los estudiantes, dado que se está trabajando directamente sobre el mundo físico que los rodea, pudiendo ser su propia casa o su escuela un ámbito de trabajo desde donde aplicar sus conocimientos de programación. Además la elección de soluciones técnicas de software y hardware libre para ser usadas en el aula (Linarez, 2013), permite dar cuenta de las discusiones sobre el rol social de las tecnologías digitales y su impacto en la soberanía tecnológicas de los estudiantes, entendiendo a estos como futuros ciudadanos que deberán enfrentarse a un mundo cada vez más atravesado por herramientas digitales que tomaran decisiones sobre sus vidas diarias y por consiguiente, la necesidad de formarlos con una mirada crítica y fundamentada respecto a las mismas.

Bibliografía

Acosta, A., Rojo, C., y Martínez, M. C. (2019). Enseñando Python en una propuesta de formación docente en enseñanza de la programación. XXV Congreso Argentino de Ciencias de la Computación (CACIC) (Universidad Nacional de Río Cuarto, Córdoba, 14 al 18 de octubre de 2019).

Adell, J., y Bernabé, Y. (2007). Software libre en educación. Tecnología educativa. Madrid: McGraw-Hill, 173–195.

Baquero, R. (1996). Vigotsky y el aprendizaje escolar (Vol. 4). Aique Buenos Aires.

Barrera Lombana, N. (2015). Use of Educational Robotics as a Teaching Strategy in the Classroom. Praxis &; Saber, 6(11), 215–234.

Basel, V. (2020). Hardware libre en el aula: Una experiencia de capacitación en el uso de recursos educativos abiertos en escuelas técnicas en Tucumán, Argentina.

Benotti, L., Echeveste, M. E., y Schapachnik, F. (2012). Despertando Vocaciones en Computación mediante el uso de autómatas de chat. 10.

Chavarría, J. V. (2011). Software libre, alternativa tecnológica para la educación. Revista Actualidades Investigativas en Educación, 5(2). https://doi.org/10.15517/aie.v5i2.9150.

de la Colina, M. F. (2004). Hacia una definición de la domótica. Informes de la Construcción, 56(494), 11–17.

Echeveste, M. E., Benotti, L., y Martinez, M. C. (2012). "Chatbot": Un software para incentivar el estudio en carreras en Tecnologías de Información y Comunicación. ExT: Revista de Extensión de la UNC, 3(2).

Estévez, R. M., Rosa, M. P., y Fernández, R. G. (2014). Viabilidad de Python en la enseñanza de la programación. Mendive. Revista de Educación, 12(2), 179–186.

García Monsálvez, J. C. (2017). Python como primer lenguaje de programación textual en la Enseñanza Secundaria.

González, I., González, J., y Gómez-Arribas, F. (2003). Hardware libre: Clasificación y desarrollo de hardware reconfigurable en entornos GNU/Linux. VI Congreso de Hispalinux, Universidad Rey Juan Carlos I. http://ftp1.nluug.nl/ftp/pub/ftp/os/Linux/doc/LuCaS/Presentaciones/200309hispalinux/8/8.pdf.

Haché, A. (2014). Soberanía tecnológica. Dossier sobre Soberanía Tecnológica, 9–18.

Janarthanam, S. (2017). Hands-on chatbots and conversational UI development: Build chatbots and voice user interfaces with Chatfuel, Dialogflow, Microsoft Bot Framework, Twilio, and Alexa Skills. Packt.

Letzen, D., Basel, V., Massolo, A., y Ferrero, F. (2019). Robótica educativa con software libre y hardware de especificaciones abiertas para enseñanza de programación. II JORNADAS ARGENTINAS DE DIDÁCTICA DE LA PROGRAMACIÓN, 47.

Linarez, G. (2013). La implementación del software libre en la educación. 65–76.

Martín, R. B. (2017). Contextos de Aprendizaje: Formales, no formales e informales.

Panizza, M. (2003). II Conceptos básicos de la teoría de situaciones didácticas. Recuperado el, 7.

Papert, S. (1987). Desafío a la mente: Computadoras y educación. Galápago.

Radford, L. (2014). De la teoría de la objetivación. Revista latinoamericana de etnomatemática, 7(2), 132–150.

Radford, L. (2017). Aprendizaje desde la perspectiva de la Teoría de la Objetivación. Enseñanza y aprendizaje de las matemáticas: problemas semióticos, epistemológicos y prácticos, 115–136.

Realinho, V. (2015). Low Cost Domotic System based on Open Hardware and Software. The Eighth International Conference on Advances in Human-oriented and Personalized Mechanisms, Technologies, and Services.

Revilla, J. O. (2018). ¿Cómo diseñar un prototipo de iluminación para mi sala de estudio? Proyectos STEAM para la Educación Primaria: fundamentos y aplicaciones prácticas, 163–194.

van Rossum, G., Warsaw, B., y Coghlan, N. (2001). PEP 8: Style guide for Python code. Python. org.

Simulador de sumo para robótica educativa

Gonzalo Zabala*
gonzalo.zabala@uai.edu.ar
Universidad Abierta Interamericana

Ricardo Morán†
ricardo.moran@uai.edu.ar
Comisión de Investigaciones Científicas

Resumen

Desde el año 2000 se realizan en Argentina las Olimpíadas Argentinas de Robótica y Feria de Proyectos de Robótica y Control Automatizado, conocida como la Roboliga, destinada a estudiantes de primaria y secundaria de todo el país[1]. En 2020, la pandemia impidió casi en su totalidad la actividad escolar presencial, en particular los talleres donde se comparten materiales, como son habitualmente los de robótica. Por otro lado, estuvo prohibido el encuentro masivo de personas. Por este motivo, tanto la Roboliga como las actividades previas de robótica en cada escuela no se pudieron llevar a cabo. Como solución a ambos problemas, desde nuestro grupo de investigación desarrollamos un simulador de sumo sobre un motor de física, que presenta características y comportamientos muy similares a la actividad en el mundo real. De esta manera, junto al simulador de rescate realizado por la Robocup, se pudo llevar adelante la Roboliga a fines del 2020. A partir de los comentarios positivos de la experiencia en el uso del simulador, se organizó una competencia nueva durante Junio y Julio de 2021, abierta a todas las edades y para todos los países de Latinoamérica (virtual.roboliga.ar). Este tipo de competencia es inédita, dado que la duración fue de seis semanas, permitiendo a los equipos mejorar sus robots de ronda en ronda. A continuación, presentamos los motivos didácticos para el desarrollo del simulador, la solución técnica y los resultados de las experiencias realizadas durante 2020 y 2021.

Palabras clave: Educación Tecnológica, Robótica, Programación, Simulación.

1. Introducción

El uso de robots en las aulas provee a los estudiantes un espacio de aprendizaje reflexivo y experimental. Cuando en equipo diseñan, construyen, programan y documentan el diseño de un robot autónomo, no sólo aprenden de tecnología, sino que también ponen en práctica un conjunto de habilidades fundamentales para el mundo que encontrarán cuando salgan de su educación formal (Eguchi, 2017). Por otra parte, el mercado laboral requiere cada día más de recursos humanos vinculados al desarrollo de tecnología, en campos como la programación, el diseño y las ingenierías. El

*Universidad Abierta Interamericana, Centro de Altos Estudios en Tecnología Informática, CABA, Argentina

†Comisión de Investigaciones Científicas, La Plata, Buenos Aires, Argentina

[1] www.roboliga.edu.ar

estudio de la robótica puede ser un recurso para despertar vocaciones en estas áreas, además de aumentar la retención de estudiantes mujeres que están subrepresentadas en el campo de la tecnología (Miller y Nourbakhsh, 2016). Por estos motivos, los robots se han convertido en un recurso didáctico valorado por los docentes de nivel primario (Khanlari, 2016) y secundario (Kandlhofer y Steinbauer, 2016).

Desafortunadamente, en el 2020 y 2021 la pandemia imposibilitó en gran medida la educación presencial. Inclusive, en algunos países donde se retornó paulatinamente a las aulas, las actividades de laboratorio que implican uso compartido de materiales no pudieron restablecerse. Por lo tanto, fue necesario reconvertir los talleres presenciales en general, y de robótica en particular en actividades remotas en línea (Birk et al., 2021; Younis et al., 2021). Entre otras propuestas, el uso de simuladores de robótica fue una de las soluciones implementadas en diversas instituciones (Gomes et al., 2020; Holowka, 2020; Salas, 2021).

Además de los talleres, existen diversas competencias de robótica que sirven como motor y faro de las actividades cotidianas de los estudiantes en las aulas. Estas competencias también se vieron afectadas por la pandemia. En particular, en Argentina se organiza desde el año 2000 la Roboliga, Olimpíada Argentina de Robótica y Feria de Proyectos y Control Automatizado, donde participan anualmente alrededor de 500 estudiantes de todo el país. La Roboliga además es la instancia nacional que realiza la selección del equipo representante de nuestro país en la Robocup Junior Internacional[2]. El COVID impidió la realización presencial de ambas, y planteó el desafío de proponer una instancia remota que se acercara a la versión física de las pruebas que se llevan a cabo. En el caso de Robocup, un equipo de desarrolladores crearon un espacio de simulación de rescate sobre la plataforma Webots (Michel, 2004). Por este motivo, en la Roboliga se implementó el mismo simulador como prueba de rescate. Pero la competencia más popular, la lucha de Sumo, carecía de un simulador con características realistas que funcionara como reemplazo de la competencia física.

Es por este motivo que se comenzó el estudio de diferentes plataformas de simulación con motor de física para la creación de un simulador de Sumo para la Roboliga. En el presente trabajo se detalla su proceso de desarrollo, y los resultados obtenidos tanto en la Roboliga 2020, como en una competencia especial virtual que se llevó a cabo en forma abierta para toda Latinoamérica en el año 2021.

2. El uso de simuladores en robótica educativa

En los últimos años, y a partir del aumento de poder de cómputo de las placas gráficas, han surgido con fuerza diferentes ambientes de simulación de robots: CoderZ (CoderZ Homepage - CoderZ, 2019), Webots (Michel, 2004), VRT (Simulate FIRST LEGO League y WRO | Virtual Robotics Toolkit, 2020), Gazebo (Gazebo, 2014), CoppeliaSim (Robot simulator CoppeliaSim: create, compose, simulate, any robot - Coppelia Robotics, 2020), entre otros. Más allá del impacto que genera la robótica en el aprendizaje de los estudiantes en diversas áreas (Benitti y Spolaôr, 2017; Eguchi, 2017; Fagin y Merkle, 2002; García-Peñalvo et al., 2019; Lepuschitz et al., 2018; Merdan et al., 2016; Miller y Nourbakhsh, 2016; Theofanellis et al., 2020), el mundo de la simulación presenta otras características que obliga a realizar estudios propios, no sólo con respecto a la mejora en el aprendizaje, sino también en comparación con los laboratorios físicos (Eguchi y Shen, 2012; Fernandes, 2013; Guyot et al., 2011; Hughes, 2016; Infante Jiménez, 2014; Major, 2014).

Con relación al estudio del impacto del uso de simuladores y laboratorios virtuales en las aulas, (Major, 2014)

[2] https://junior.robocup.org/

analiza su efectividad para el aprendizaje en programación. En (Hughes, 2016) se presenta una experiencia realizada con CoSpace en 500 escuelas durante 3 años. Basados en la misma herramienta, en (Eguchi y Shen, 2012) se realizó un estudio entre 78 participantes de la Robocup 2011. En (Casañ Núñez y Cervera, 2018) se analiza el impacto de una herramienta creada en el marco de la iniciativa "The Robot Programming Network", aunque no es un ambiente físico en 3D como el propuesto en el presente artículo. Junto con el desarrollo del simulador, en (Fernandes, 2013) se realiza un experimento con 60 usuarios entre profesores, estudiantes y adultos sin conocimientos en robótica. Finalmente, en (Guyot et al., 2011) se desarrolla una propuesta curricular sobre Webots, que es evaluada en 64 estudiantes de escuela media y en la universidad donde se llevó a cabo la investigación.

A partir de los resultados de estos estudios, se puede concluir que, sin reemplazar el uso de materiales físicos para la enseñanza de la robótica, el uso de simuladores permite un primer acceso más sencillo a la temática; es más económico en su implementación y mantenimiento; genera menos temor a los errores y su depuración y solución es menos problemática.

Por lo tanto, más allá de ser una solución indispensable para la situación provocada por la pandemia, la utilidad de un simulador no finaliza ante la superación del COVID, sino que es un recurso didáctico que puede ofrecer otras variantes complementarias al material de robótica concreto. Para ellos, deben presentar las siguientes características:

- Estar basados en un motor de física que permita una recreación lo más realista posible del comportamiento en el mundo real, considerando colisiones, rozamientos, ruido en los sensores, comportamientos no deterministas entre otros.
- No necesitar de un equipamiento muy sofisticado para su ejecución, dado que no es habitual contar con equipos de última generación en los ámbitos educativos.
- Ser multiplataforma.
- Ser sencillos de instalar y operar.

Luego de realizar un profundo estudio de las plataformas que cumplían con estas características, se eligió a Webots (Michel, 2004) por su fidelidad en la representación del mundo físico, su portabilidad, su versatilidad para la construcción de nuevos mundos y porque también fue seleccionado por la Robocup para el simulador de la categoría de rescate, presente en la Roboliga.

3. Características de Webots

Webots es una plataforma de simulación de robots basada en un motor de física (ODE) para la detección de colisiones y la simulación de la dinámica del cuerpo rígido. Permite desarrollar prototipos rápidamente mediante espacios virtuales en tres dimensiones, que poseen propiedades físicas como masa, coeficientes de fricción, articulaciones y otros. Los robots construidos en el sistema pueden tener diferentes tipos de locomoción, y estar equipados con sensores y actuadores de distinto tipo como sensores de distancia, lidars, cámaras, sistemas de comunicación, etc. La programación de los robots se puede realizar en diferentes lenguajes como C, C++, Python, Java o MATLAB.

En su librería posee muchos modelos con características diversas, como vehículos autónomos, humanoides, aspiradoras, robots de laboratorio, brazos y otros. Mediante el lenguaje VRML97 y un conjunto de primitivas pueden construirse nuevos robots donde se definen los elementos mecánicos y electrónicos que posee. Para su ejecución se requiere una computadora PC o Mac con un procesador de al menos doble núcleo de 2Ghz. Debe contar con 2 GB de RAM y una placa gráfica NVIDIA o AMD OpenGL (versión 3.3 al menos) con 512 Mb de RAM o más. Tiene

versiones para Linux (64 bits), Windows 8.1 en adelante (64 bits) y macOS 10.14 en adelante.

La arquitectura de una simulación está compuesta por:

a) World: descripción 3D de los robots y el entorno en el que actúan. Describe cada elemento mediante un árbol jerárquico de composición. Por ejemplo, un robot puede estar compuesto por un módulo de locomoción y otro de sensado, donde podemos encontrar en el primero dos ruedas motorizadas, una rueda loca y una estructura de soporte; y en el segundo dos sensores de distancia y un conjunto de leds, también montados en su estructura. Para cada objeto se describen propiedades (según el tipo de objeto) como la posición, rotación, dimensiones, geometría visible, espacio de colisión, propiedades físicas, tipo de sensado, ruido de sensado, etc.

b) Controllers: programa que posee cada robot que determina su comportamiento. Como se mencionó previamente, puede estar en C, C++, Python, Java o MATLAB.

c) Supervisor controller: es un programa que realiza un control general de todo el ambiente. En general, realiza acciones similares a las que haría el humano que manipula el ambiente, como control de todo el mundo, movimiento de los robots, arbitraje y otros.

4. Acerca de la competencia de Sumo Robótico

El sumo robótico es una de las competencias que se realiza habitualmente en la Roboliga. Es un desafío interesante dado que no representa una complejidad excesiva, lo cual facilita la participación de alumnos pequeños; permite una gran variedad de estrategias tanto en la construcción del robot como en su programación; y es divertida de ver y de programar, con resultados a veces sorpresivos para los propios participantes.

Los encuentros de sumo robótico constan de dos robots situados en una plataforma circular negra (usualmente denominada “dohyo”) delimitada por una línea blanca. Cada robot tiene como objetivo sacar al contendiente del dohyo antes de que transcurra un tiempo determinado. Para más información, el reglamento completo se puede consultar online[3].

Normalmente los robots están construidos de forma que puedan, además de moverse, detectar la proximidad del oponente y los límites del dohyo. En el ambiente de simulación desarrollado para la Roboliga Virtual se decidió que ambos equipos utilicen un robot de iguales características. Estos robots simulados cuentan con dos sensores de distancia ubicados en la parte frontal, un sensor de color mirando hacia el piso (para detectar la línea blanca), dos sensores de tacto en la parte trasera, y dos motores ubicados uno a cada lado.

Considerando que el sumo robótico es una disciplina en la cual los robots están en frecuente contacto uno con el otro es imprescindible que el simulador pueda modelar interacciones físicas con gran precisión para que la simulación se vea realista. Al mismo tiempo, es deseable que la resolución de estas interacciones no sea completamente determinística y que exista algún grado de aleatoriedad que dificulte predecir el resultado de un encuentro. Por esta razón se agregaron al ambiente de simulación: una cuota de ruido en el funcionamiento de los sensores; y una rutina que modifica aleatoriamente la posición y orientación inicial de cada robot.

[3] http://virtual.roboliga.ar/1_5_reglamento.html

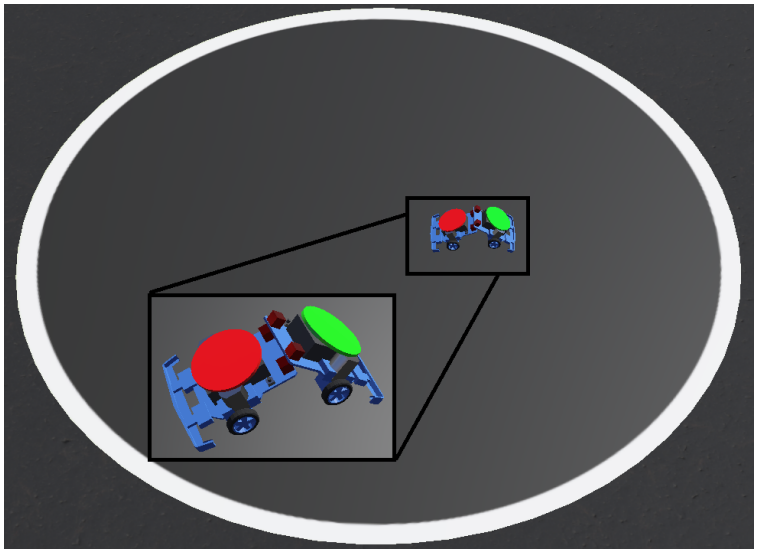

Figura 1: Los robots simulados enfrentándose en el dohyo

5. Programación en Python/bloques

Dada su facilidad de uso se decidió utilizar Python como lenguaje oficial para la programación de los robots durante la competencia. Para simplificar la tarea y homogeneizar los programas de los participantes, se desarrolló una clase utilitaria que implementa la necesaria interacción con el simulador. Esta clase, llamada “RobotRL”, incluye métodos para acceder a los valores de los sensores del robot, así como para modificar las velocidades de los motores. De esta forma, los participantes pueden concentrarse específicamente en definir su estrategia y delegar en una instancia de “RobotRL” los detalles de comunicación con el simulador.

A pesar de que Python es un lenguaje usualmente recomendado para el aprendizaje de la programación, se decidió desarrollar también un editor visual basado en bloques para poder incluir en la competencia a quienes no tienen la suficiente experiencia en programación y se sienten más cómodos con un lenguaje de bloques (particularmente los niños más pequeños). Este editor fue desarrollado específicamente para esta competencia, por lo cual además de los bloques tradicionales para estructuras de control u operaciones lógico-matemáticas, incluye bloques específicos para la programación del robot de sumo.

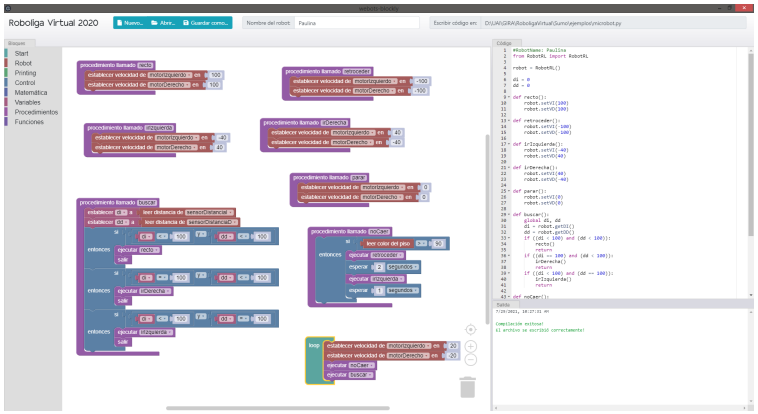

Figura 2: Herramienta webots-blockly para la programación por bloques

A medida que el usuario va desarrollando su programa, el editor de bloques va generando el código Python correspondiente que luego el simulador utilizará para controlar al robot. La interfaz gráfica fue diseñada de tal forma que el código generado a partir de los bloques sea siempre visible a los usuarios, en un intento de fomentar la adopción de Python.

6. Supervisor/Automatización

Para simplificar la operación del simulador se incluyó un controlador "supervisor" que se encarga de gestionar la simulación en su totalidad. Este supervisor está compuesto, por un lado, por un programa en Python encargado de orquestar el progreso de la simulación y, por otro lado, por una interfaz HTML que se embebe dentro del simulador y permite al usuario un mínimo de interacción (esencialmente seleccionar los programas controladores de cada robot, iniciar/pausar la simulación, y visualizar información asociada a la competencia).

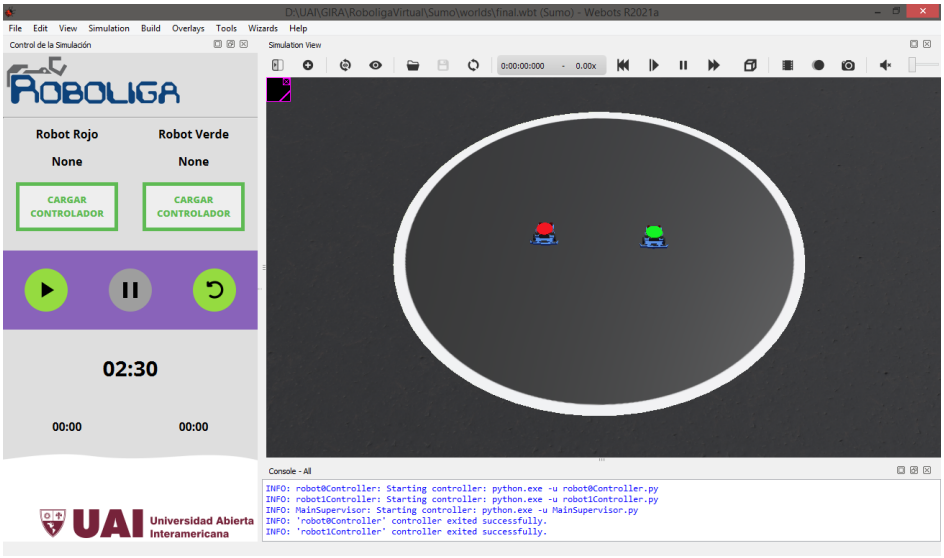

Figura 3: Interfaz del supervisor

Además de controlar el correcto funcionamiento de la simulación, el supervisor es el encargado de verificar las reglas de la competencia, lo cual lo convierte en un árbitro 100 % automatizado. Es capaz de detectar cuando un robot sale de los límites del dohyo, declarar al ganador y al perdedor del encuentro, y reubicar a los robots si superan un tiempo determinado sin moverse, entre otras cosas. En una segunda iteración, se desarrolló un supervisor para uso interno que además de controlar la correcta ejecución de cada encuentro es capaz de armar el fixture completo de la competencia, ejecutar automáticamente todas las rondas, y comunicarse con OBS (software de edición de videos en tiempo real) para grabar videos de cada encuentro (los cuales pueden verse en https://www.youtube.com/channel/UC-0P2S8oH9xz2GwEWj160dw).

7. Resultados

La Roboliga Virtual 2020 contó con la participación de 96 estudiantes de todo el país agrupados en 49 equipos. La competencia se organizó en 2 categorías, diferenciadas principalmente en el tipo de herramienta utilizada para

programar los robots. Los participantes de la categoría "inicial" debían utilizar los bloques (que finalmente generaban código Python) y los de "avanzado" podían utilizar Python directamente. Esta separación se hizo considerando el limitado poder expresivo de los bloques, en un intento por igualar oportunidades. De los 96 estudiantes totales, 85 participaron en la categoría "inicial" (44 equipos) y 11 en la categoría "avanzado" (5 equipos). La competencia se realizó en dos etapas: una clasificatoria y una ronda final. En la etapa clasificatoria se dividió a los equipos en zonas, donde compitieron en la modalidad todos contra todos, clasificando los equipos con mayor puntaje de cada zona. La ronda final utilizó un esquema de eliminación directa y se transmitió en vivo a través de YouTube.

La Roboliga Virtual 2021 tuvo diferencias significativas con su antecesor. En primer lugar, la competencia se abrió a todo Latinoamérica en lugar de sólo estudiantes argentinos. Asimismo, se modificaron las categorías, ya no en función de la herramienta de programación sino en la edad de los participantes (Infantil: de 9 a 12 años, Juvenil: de 12 a 18 años, y Libre: cualquier edad). En segundo lugar, la competencia se organizó de forma que la etapa clasificatoria se extendiera durante varias semanas, permitiendo a los participantes analizar el rendimiento tanto de su robot como el de sus contrincantes y modificar el programa en consecuencia antes de la siguiente ronda.

Por país	# equipos	# participantes	Partic. por equipo
Argentina	129	239	1.85
Panamá	31	80	2.58
Cuba	1	1	1.00
México	1	1	1.00
Colombia	3	3	1.00
Por categoría	**# equipos**	**# participantes**	**Partic. por equipo**
Infantil (9 a 12 años)	59	99	1.68
Juvenil (12 a 18 años)	82	181	2.21
Libre (19 años o +)	24	44	1.83
Total	165	324	1.96

Tabla 1

Luego de la etapa de rondas quedaron clasificados 8 equipos de cada categoría, y el 17 de julio de 2021 se realizaron en vivo los cuartos, semifinales y finales. Para este encuentro final también pudieron enviar un nuevo código, sabiendo de antemano cómo estaban diagramadas las llaves.

Posteriormente al evento se realizó una encuesta a los participantes, donde se obtuvieron 33 respuestas con los siguientes resultados:

- ¿Cuál es tu opinión sobre el simulador y la dinámica de la lucha de sumo? (1 malo - 10 excelente): promedio 8.33
- ¿Cuál es tu opinión sobre la organización del evento? (1 malo - 10 excelente): promedio 8.91
- ¿Qué opinás sobre la comunicación? (Mails, lista de correo, respuesta a consultas, etc) (1 malo - 10 excelente): promedio 8.64
- ¿Qué te pareció la página web? (1 malo - 10 excelente): promedio 8.97
- ¿Qué usaste para programar el código?: Python 22 - Bloques 11
- ¿Te gustaría hacerlo en otro lenguaje de programación? ¿Cuál?: 10 participantes propusieron otros lenguajes

como Arduino, C++, Scratch, Javascript y HTML.

- ¿Cuántas veces modificaste el código durante el transcurso de la competencia?: promedio 3.76
- ¿Participarías nuevamente?: Si 27 - Tal vez 6.
- ¿Te gustaría poder cambiar características del robot como cantidad de sensores, posición de estos, etc?: Sí 25 - No 2 - Tal vez 6.
- ¿Te gustaría participar de una competencia de fútbol de robots virtual?: Sí 25 - No 2 - Tal vez 6 (los números coinciden con el ítem anterior pero no fueron las mismas respuestas)

8. Lecciones aprendidas

En la encuesta mencionada se ofreció un apartado para realizar comentarios libres. Si bien las respuestas fueron mayoritariamente positivas, algunos participantes realizaron propuestas que podrían servir para mejorar la competencia a futuro.

Una de las mayores críticas fue la limitada variedad en la simulación. Dado que todos los robots son exactamente iguales en lo que respecta a su estructura física, la única posibilidad de personalización que se le ofrece a los participantes es mediante la programación. Una de las ideas que se está poniendo en práctica en este sentido es la utilización de un editor que permita modificar la estructura del robot, agregar o quitar sensores, moverlos de lugar, etc. Este editor ya está siendo desarrollado tomando como base el realizado para la Robocup en el contexto del simulador de rescate Erebus 2021, y permitiría un alto grado de personalización de los robots en futuras competencias.

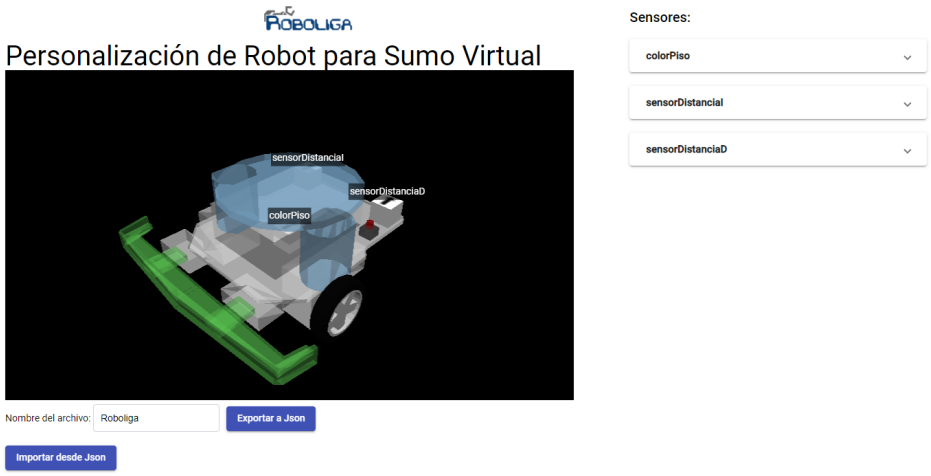

Figura 4: Interfaz de personalización del robot (en desarrollo)

Finalmente, algunos participantes plantearon también que sería interesante permitir enfrentamientos amistosos entre equipos distintos sin la necesidad de compartir el código. Para ello, se están analizando soluciones que permitan ejecutar simulaciones a pedido en un servidor centralizado.

9. Conclusiones

La Roboliga Virtual se organizó como una respuesta a la virtualidad obligada por el COVID y los resultados obtenidos son alentadores. Una de las consecuencias positivas del uso de simulación es que la competencia pudo expandirse geográficamente a una escala que sería inviable de forma presencial. Si bien los encuentros virtuales no reemplazan completamente a la presencialidad, consideramos que una modalidad mixta podría ser una opción atractiva para futuros eventos. Otro aspecto positivo es la posibilidad de expandir en el tiempo el evento, lo que permite el rediseño de los robots entre las diferentes rondas. Ante estos resultados, se ha comenzado a analizar el desarrollo de otros simuladores para ser utilizados en el ámbito educativo.

Bibliografía

Benitti, F. B. V., y Spolaôr, N. (2017). How Have Robots Supported STEM Teaching? En M. S. Khine (Ed.), Robotics in STEM Education: Redesigning the Learning Experience (pp. 103–129). Springer International Publishing. https://doi.org/10.1007/978-3-319-57786-9_5

Birk, A., Dineva, E., Maurelli, F., y Nabor, A. (2021). A Robotics Course during COVID-19: Lessons Learned and Best Practices for Online Teaching beyond the Pandemic. Robotics, 10(1), 5.

Casañ Núñez, G. A., y Cervera, E. (2018). The Experience of the Robot Programming Network Initiative.

CoderZ Homepage—CoderZ. (2019). https://gocoderz.com/

Eguchi, A. (2017). Bringing robotics in classrooms. En Robotics in STEM education (pp. 3–31). Springer.

Eguchi, A., y Shen, J. (2012). Student learning experience through CoSpace educational robotics. Society for Information Technology & Teacher Education International Conference, 19–24.

Fagin, B. S., y Merkle, L. (2002). Quantitative analysis of the effects of robots on introductory Computer Science education. Journal on Educational Resources in Computing, 2(4), 2-es. https://doi.org/10.1145/949257.949259

Fernandes, C. da C. (2013). S-Educ: Um Simulador de Ambiente de Robótica Educacional em Plataforma Virtual. https://repositorio.ufrn.br/jspui/handle/123456789/15464

García-Peñalvo, F. J., Conde, M. Á., Gonçalves, J., y Lima, J. (2019). Computational thinking and robotics in education. Proceedings of the Seventh International Conference on Technological Ecosystems for Enhancing Multiculturality, 2–5.

Gazebo. (2014). http://gazebosim.org/

Gomes, A. S. A., Da Silva, J. F., y Teixeira, L. R. D. L. (2020). Educational robotics in times of pandemic: Challenges and possibilities. 2020 Latin American Robotics Symposium (LARS), 2020 Brazilian Symposium on Robotics (SBR) and 2020 Workshop on Robotics in Education (WRE), 1–5.

Guyot, L., Heiniger, N., Michel, O., y Rohrer, F. (2011). Teaching robotics with an open curriculum based on the e-puck robot, simulations and competitions. Proceedings of the 2nd International Conference on Robotics in Education. Vienna, Austria.

Holowka, P. (2020). Teaching robotics during COVID-19: Machine learning, simulation, and aws deepracer. 17th International Conference on Cognition and Exploratory Learning in Digital Age, CELDA 2020.

Hughes, J. (2016). Robotic rescue simulation for computing teaching in the UK: A case study. 2016 IEEE Global Engineering Education Conference (EDUCON), 1051–1055.

Infante Jiménez, C. (2014). Propuesta pedagógica para el uso de laboratorios virtuales como actividad complementaria en las asignaturas teórico-prácticas. Revista mexicana de investigación educativa, 19(62), 917–937.

Kandlhofer, M., y Steinbauer, G. (2016). Evaluating the impact of educational robotics on pupils' technical-and social-skills and science related attitudes. Robotics and Autonomous Systems, 75, 679–685.

Khanlari, A. (2016). Teachers' perceptions of the benefits and the challenges of integrating educational robots into primary/elementary curricula. European Journal of Engineering Education, 41(3), 320–330.

Lepuschitz, W., Merdan, M., Koppensteiner, G., Balogh, R., y Obdržálek, D. (2018). Robotics in Education: Methods and Applications for Teaching and Learning (Vol. 829). Springer.

Major, L. (2014). An empirical investigation into the effectiveness of a robot simulator as a tool to support the learning of introductory programming. [PhD Thesis]. Keele University.

Merdan, M., Lepuschitz, W., Koppensteiner, G., y Balogh, R. (2016). Robotics in education: Research and practices for robotics in STEM education (Vol. 457). Springer.

Michel, O. (2004). Cyberbotics Ltd. WebotsTM: Professional mobile robot simulation. International Journal of Advanced Robotic Systems, 1(1), 5.

Miller, D. P., y Nourbakhsh, I. (2016). Robotics for education. En Springer handbook of robotics (pp. 2115–2134). Springer.

Robot simulator CoppeliaSim: Create, compose, simulate, any robot—Coppelia Robotics. (2020). https://www.coppeliarobotics.com/

Salas, R. P. (2021). Teaching Continuity in Robotics Labs in the Age of Covid and Beyond. arXiv preprint arXiv:2105.08839.

Simulate FIRST LEGO League y WRO | Virtual Robotics Toolkit. (2020). https://www.virtualroboticstoolkit.com/

Theofanellis, T., Voulgari, E., y Tsolakis, S. (2020). Educational Robotics and Computational Thinking Development. En Handbook of Research on Tools for Teaching Computational Thinking in P-12 Education (pp. 306–334). IGI Global.

Younis, H. A., Mohamed, A. S. A., Jamaludin, R., y Ab Wahab, M. N. (2021). Survey of robotics in education, taxonomy, applications, and platforms during COVID-19. Computers, Materials and Continua, 67(1).

Minirobots App: una aplicación educativa para aprender a programar jugando

Francisco Prieto*
priettt@gmail.com
UNCPBA

Fermín de Fiore*
fer.defiore@outlook.com.ar
UNCPBA

Paula Tristán*
ptristan@exa.unicen.edu.ar
UNCPBA

Laura Felice*
lfelice@exa.unicen.edu.ar
UNCPBA

Resumen

Se ha demostrado que los primeros años de vida de un niño son un período clave en su desarrollo intelectual, e influyen significativamente en su trayectoria educativa. Estos son los motivos principales en los que se ha basado la comunidad educativa al introducir a la alfabetización digital en la educación inicial. En este trabajo se presenta una aplicación móvil denominada Minirobots App. Esta app forma parte del proyecto Minirobots, simplificando el mecanismo de conexión con el robot y aportando el diseño de tarjetas físicas como piezas de un rompecabezas que representan las instrucciones que el dispositivo puede ejecutar. El uso de las tarjetas por parte de los alumnos incorpora el concepto lúdico durante el proceso de aprendizaje. La combinación de las tarjetas permite crear programas formando secuencias que el robot reconocerá y ejecutará. La app permite, tomando una fotografía con un dispositivo móvil, reconocer las secuencias de tarjetas construidas por el alumno, asistido por el docente, y transformarla en un programa que hará que el robot responda según la secuencia creada.

Palabras clave: Robótica educativa, Minirobots, Nivel Inicial, Machine Learning, Text Recognition.

1. Introducción

Durante mucho tiempo, la alfabetización digital de los niños desde los niveles iniciales de la educación, ha estado orientada principalmente a convertirlos en usuarios de las aplicaciones tecnológicas. Sin embargo, en los últimos años ha crecido el interés por ampliar esta formación, educando en la comprensión de cómo funcionan las tecnologías, con el objetivo de convertirlos en usuarios creativos del mundo digital y no solamente en simples consumidores (Hepp y Jara, 2016). Estas habilidades genéricas forman parte de lo que se denomina el pensamiento computacional (Fundación Sadosky, 2021), y fomentan un pensamiento altamente analítico que puede ser aprovechado en un sinnúmero de áreas.

*INTIA. Facultad de Ciencias Exactas. Universidad Nacional del Centro de la Pcia. de Buenos Aires.

En este sentido, los mini robots se han convertido en un elemento clave para la educación inicial. Particularmente, en Argentina, Minirobots (MiniRobots, 2021) es un proyecto que utiliza tecnologías modernas, de bajo costo y código abierto enfocado en el ámbito educativo inicial. Los dispositivos de Minirobots constan de un robot con ruedas y un lápiz, con la capacidad de dibujar figuras mediante instrucciones simples a través de una aplicación web. Tanto el robot como el software para comandar son de libre acceso para quienes quieran conocerlo e incluso modificarlo. La Figura 1 ilustra el dispositivo y sus funciones principales.

(a) **(b)** **(c)**

Figura 1: MiniRobots y algunas de sus funcionalidades.

Para que el MiniRobot pueda ejecutar instrucciones, se requiere que tanto el robot como el dispositivo desde donde se edita el programa estén conectados a la misma red. El editor permite escribir bloques de programa (Figura 2) combinando las diferentes instrucciones disponibles, que luego serán enviados al robot para su posterior ejecución.

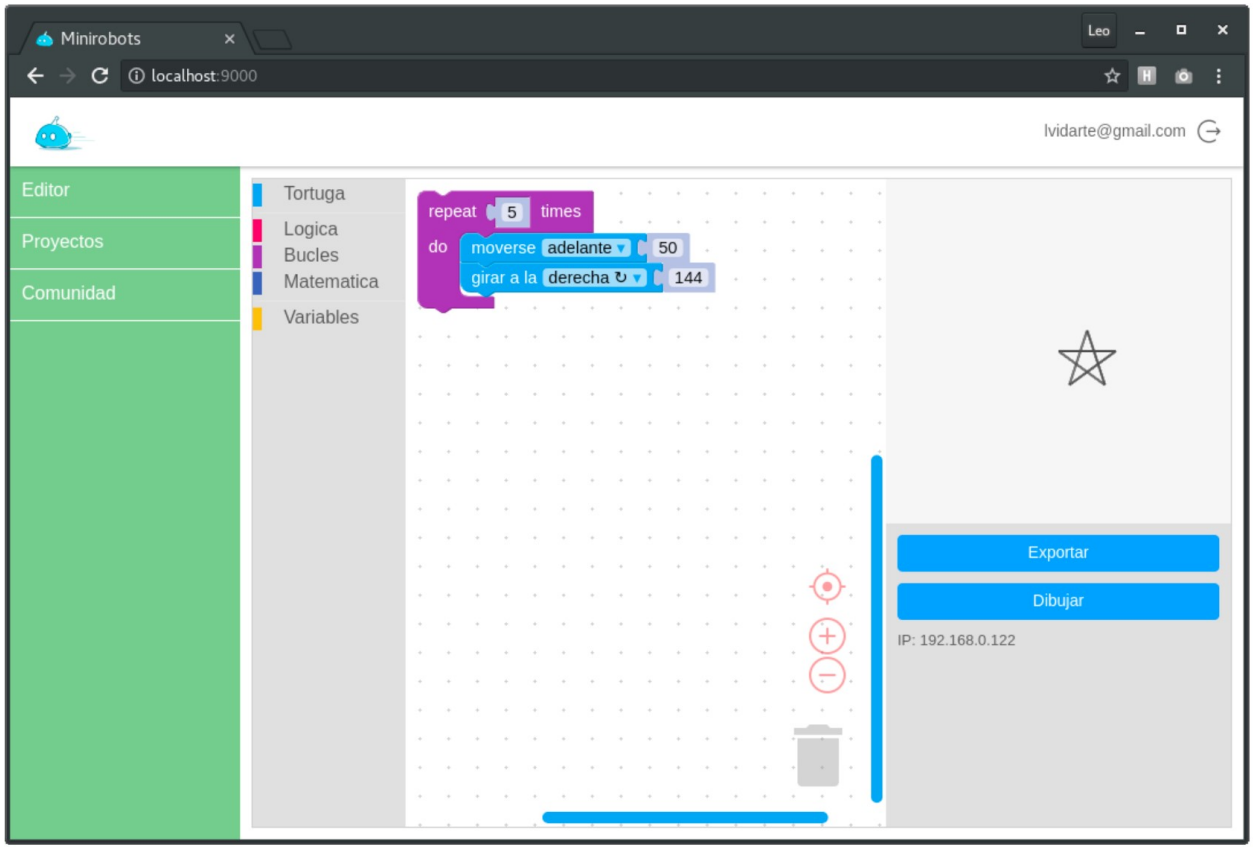

Figura 2: Editor web disponible para escribir el programa, enviarlo al robot y ejecutarlo.

En este trabajo se presenta Minirobots App, una aplicación móvil que propone no solo un nuevo paradigma de interacción con los alumnos utilizando tarjetas físicas para construir programas, sino que simplifica los requisitos de comunicación con el robot.

Las tarjetas físicas incorporan el concepto lúdico al proceso de aprendizaje de los niños ya que los programas se construyen encastrando instrucciones secuencialmente.

La aplicación permite, mediante técnicas de procesamiento de imágenes y Machine Learning (Chollet, 2018), reconocer las instrucciones que conforman el programa para luego enviarlo directamente al robot para su ejecución.

Este artículo está organizado de la siguiente forma: en la Sección 2, se describen los productos más utilizados en robótica educativa en nivel inicial. Posteriormente, en la Sección 3 se presenta la mejora al proyecto descrito. Más adelante, en la Sección 4, se presenta la aplicación propuesta, detallando las tecnologías utilizadas para su implementación. Finalmente, la Sección 5 presenta conclusiones y trabajo futuro.

2. Estado del arte

Actualmente, la robótica educativa en todos los niveles, ofrece ambientes favorables para el desarrollo de proyectos de programación cuyos productos salen de las pantallas para dar vida a artefactos concretos que se mueven y toman datos del ambiente para realizar ciertas acciones.

En nuestro país, el programa Aprender Conectados (Recursos Inicial, 2021) presenta actividades, proyectos y una amplia variedad de recursos educativos para orientar la alfabetización digital del Nivel Inicial. El impacto que genera en un niño ver cómo un robot sortea obstáculos hasta llegar a su meta interactuando con los docentes a través de un dispositivo, tiene mucho más valor que ver cómo un personaje u objeto deambula en un monitor, esquivando obstáculos virtuales hasta llegar a su meta (Educación digital Inicial, 2021).

En el mundo, algunos de los productos de robótica educativa más relevantes son: LEGO Mindstorms (Aprende a programar, 2021), Bee-Bot, Fischertechnik Learning fischertechnik (2021), VEX EDR kit (Home - VEX Robotics , 2021) y robot Moway. La mayoría de estos productos fueron creados por compañías en países desarrollados con niveles y condiciones de vida diferentes a los de las economías de países de Latinoamérica. A su vez, en el campo técnico, las desventajas más comunes de estos kits son su alto costo, el número de piezas definido y el comportamiento de los sensores que componen el sistema de percepción de los robots, los cuales en muchos casos son complejos en el uso y en la programación. En Argentina, por ejemplo, se destacan alternativas comerciales como el Proyecto Icaro (Icaro Robótica Educativa, 2021), y otros.

Las universidades argentinas, por su parte, a través de sus investigadores, desarrollan diversos proyectos en el área, se pueden mencionar: “Enseñando a programar con Robots” de la UNLP (Programando con Robots, 2021), el proyecto “UNC++” de la UNC[1] y FrankLab (De la Fuente et al., 2018).

Minirobots nace de la visión de llevar la programación a todos los hogares, creando robots que deban ser controlados mediante algoritmos. Minirobots es un proyecto de Software Libre, premiado con el Fondo Semilla del Ministerio de la Producción y ganador del Concurso Clarín-Zurich, Premio a la Educación 2018. El prototipo inicial de Minirobots es un robotito que se programa visualmente en Logo, un lenguaje que fue creado para enseñar los conceptos básicos de la programación y con el cual se pueden repasar y reforzar conceptos de geometría, ya que se

[1]Grupo de extensión de la Universidad Nacional de Córdoba (2021, 13 de Octubre). UNC http://umm.famaf.unc.edu.ar

trata de programar la tortuga (así se llama el cursor que dibuja en Logo) para que dibuje distintas figuras geométricas.

Es importante destacar que en otros campos de la educación inicial, en áreas como la Música, se ha desarrollado y se utiliza un modelo que opera sobre la estimulación del desarrollo cognitivo a través de distintos canales de percepción, utilizando recursos lúdicos visuales, auditivos y táctiles (Sauber, 1998). Este tipo de experiencias con material basado en imágenes ha sido una referencia importante para abordar el desarrollo del proyecto relacionado con los mini robots.

3. Propuestas de mejora

Entre las experiencias desarrolladas con Minirobots, en instituciones educativas tanto públicas como privadas de la ciudad de Tandil, se puede destacar la realizada en la Escuela Primaria Nro. 53 “Cte. Eduardo Olivero”, con alumnos de entre 8 y 12 años durante los años 2018 y 2019. Durante dicha experiencia, se pudieron detectar varias características que podían ser optimizadas, así como la necesidad de incorporar nuevas capacidades y funcionalidad al robot.

Por lo antes expuesto, en este trabajo se proponen una serie de alternativas que pretenden mejorar el proyecto Minirobots y sus posibilidades de uso. Estas mejoras están relacionadas fundamentalmente con la conectividad, la usabilidad, y la implementación de nuevas funcionalidades. A continuación, se describen las mejoras desarrolladas en este trabajo.

3.1. Conectividad y comunicación

Para mejorar la usabilidad de los Minirobots y hacer más amigable la experiencia de usuario, se realizaron una serie de cambios que modificaron el flujo de conexión entre el mismo y los dispositivos que lo comandan. Antes del desarrollo de esta app, para poder enviarle instrucciones era necesario seguir paso a paso las siguientes tareas:

- Encender el robot, el cual se levantaba como Access Point,
- Conectarse a la red provista por el robot,
- Enviar al robot los datos de la red wifi a la cual se debía conectar,
- Una vez recibidos, el robot se reiniciaba y se conectaba a la red,
- Averiguar la IP del robot dentro de la red,
- Enviar las instrucciones dentro de un dispositivo conectado al mismo AP que el robot. Por ejemplo el editor web de la Figura 3.

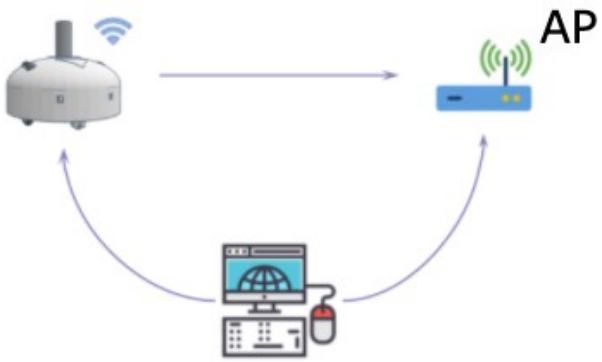

Figura 3: Modelo de conexión y uso de MiniRobots previo a las mejoras implementadas

Luego de evaluar alternativas para mejorar el flujo anteriormente descrito, y con la premisa de simplificar el proceso de comunicación y minimizar los requisitos de hardware e infraestructura para su uso, se procedió a modificar parte del código del robot que se encarga del manejo de la red y la comunicación.

Con el fin de mantener la funcionalidad que el robot ya proveía, se añadió la posibilidad de *setear* dos modos distintos de red en el mismo. Siendo el primero el original, y el segundo el utilizado por la App. En este caso el flujo para el envío de instrucciones (ilustrado en la Figura 4) es el siguiente:

- Encender el robot, el cual se levanta como Access Point,
- Conectarse a la red provista por el robot,
- Enviar las instrucciones a la IP fija del robot mediante la API provista.

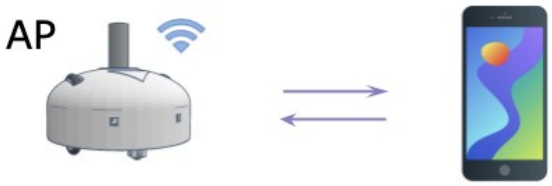

Figura 4: Nuevo modelo de conexión y uso de MiniRobots

Comparando ambos modelos, se puede observar la mejora sustancial de la interacción entre el robot y la aplicación.

La simplicidad del nuevo proceso de comunicación garantiza la usabilidad en ámbitos en donde los recursos e infraestructura de red son escasos. No obstante, esta propuesta impide la conexión directa desde un dispositivo a múltiples robots, opción provista en la versión anterior.

3.2. Las tarjetas físicas como elemento lúdico

Los Minirobots pueden ejecutar un conjunto finito de instrucciones individuales, que también pueden ser combinadas para generar programas o bloques de instrucciones contiguas.

Con el fin de poder crear secuencias didácticas que ayuden al aprendizaje basado en elementos lúdicos, se diseñó y creó un conjunto de tarjetas físicas que se correlacionan con las instrucciones que el robot puede ejecutar.

Para permitir generar distintas secuencias válidas de instrucciones, las tarjetas son ensambladas al estilo rompecabezas, unas con otras dependiendo de la compatibilidad entre las mismas. De esta manera, se busca que el uso de Minirobots sea más atractivo para el usuario (los alumnos) y que los mismos puedan aprender jugando. A continuación, se detalla la funcionalidad de cada tarjeta y la imagen de cada una.

3.2.1. Inicio y Fin de bloque

Las tarjetas *INICIO* y *FIN* son utilizadas para denotar el comienzo y el final de los bloques de instrucciones.

Figura 5: Tarjetas diseñadas para las instrucciones de Inicio y Fin de bloque

3.2.2. Inicio y Fin de Función

De manera similar, pero para la declaración de bloque de funciones, se diseñaron las tarjetas *INICIO FN* y *FIN FN*. El propósito de estas tarjetas es permitir la declaración, mediante las primeras dos, y la ejecución de bloques de instrucciones mediante la tarjeta FUNCIÓN.

Figura 6: Tarjetas diseñadas para las instrucciones de Inicio y Fin de Función

3.2.3. Desplazamientos

En cuanto a los movimientos que el robot puede realizar, hay dos tipos de instrucciones, desplazamiento hacia *ADELANTE* y desplazamiento hacia *ATRAS*.

Figura 7: Tarjetas diseñadas para las instrucciones de mover adelante y atrás

3.2.4. Repetición

Los desplazamientos pueden combinarse con un número de veces a repetir ese movimiento, representados por tarjetas numéricas.

Figura 8: Tarjetas diseñadas para las instrucciones de cantidad de repeticiones

3.2.5. Giros

El segundo tipo de tarjetas corresponde a los giros permitidos: girar hacia *DERECHA* y girar hacia *IZQUIERDA*.

Figura 9: Tarjetas diseñadas para las instrucciones de giro a Izquierda y Derecha

3.2.6. Giros por Grado

La instrucción de giro puede combinarse con un número de grados para realizar el giro. La tarjeta ángulo elige un ángulo aleatorio entre los dispuestos.

Figura 10: Tarjetas diseñadas para las instrucciones de giro en grados específicos

3.2.7. Luces

Además, el robot posee luces led RGB. Para esto se tiene un set de instrucciones que permiten manejar su estado. La tarjeta que hace referencia a ellos es *LEDS* y podrá combinarse con una serie de colores predefinidos. La tarjeta color elige un color aleatorio entre los dispuestos.

Figura 11: Tarjetas diseñadas para las instrucciones de colores de leds

3.2.8. Dibujar

En el centro del robot hay un espacio dedicado para colocar un fibrón. Esto permite que a través de los movimientos, se realicen dibujos. Para ello, se dispone de dos instrucciones que permiten subir y bajar el fibrón: *ARRIBA* y *ABAJO*.

Figura 12: Tarjetas diseñadas para las instrucciones de bajar o subir el lápiz

3.2.9. Música

El último recurso del robot es la posibilidad de emitir sonidos. Se cuenta con 3 tipos de instrucciones: corchea, negra y melodía.

Con una instrucción de melodía, se reproducen melodías según el modificador que se elija. Por ejemplo, si se pone la tarjeta junto a la nota A, sonará la canción de Super Mario Bros.

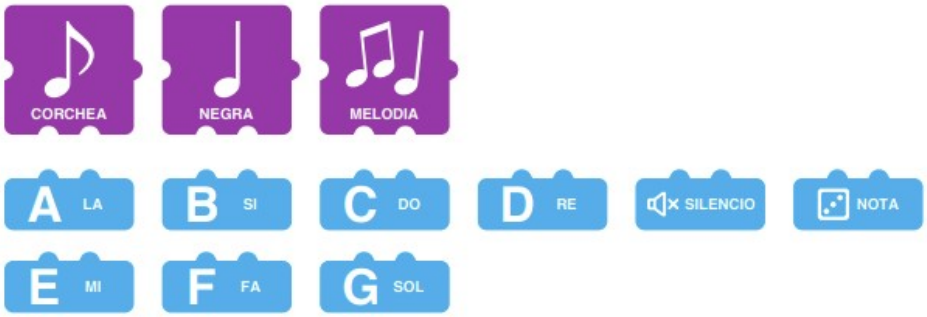

Figura 13: Tarjetas diseñadas para las instrucciones de notas musicales

Con las instrucciones negra y corchea, se reproduce el tono indicado en el modificador, con una duración de 60ms para la corchea, y 120ms para la negra. Por ejemplo, una corchea combinada con la nota LA, reproducirá un tono a una frecuencia de 440hz durante 60ms.

3.2.10. Iteración

Para trabajar la noción de ciclos, se cuenta con dos tarjetas que permiten repetir un bloque de instrucciones, *REPETIR* y *FIN REPETIR*. Son tarjetas combinables horizontalmente con las instrucciones de movimiento, las relacionadas a leds, al fibrón para escritura y a las musicales. A su vez, también permiten combinarse con las instrucciones numéricas, para poder repetir el bloque dado X cantidad de veces.

Figura 14: Tarjetas diseñadas para las instrucciones de ciclos de interacción

El diseño de las tarjetas se realizó en varias iteraciones, mejorando la usabilidad que brindaban las mismas, como también la efectividad de su reconocimiento en la aplicación.

4. La Aplicación

Como se mencionó anteriormente, la aplicación incorpora un conjunto de funcionalidades que tiene como objetivo facilitar la comunicación y envío de instrucciones al robot. El flujo de uso de la app podría dividirse en dos partes fundamentales: el reconocimiento de las instrucciones en base a una foto tomada por el usuario, y el envío de las instrucciones reconocidas al robot para su ejecución. En esta sección, se presenta la metodología aplicada para la construcción de la app.

A continuación se describen las etapas involucradas en el procesamiento de la figura.

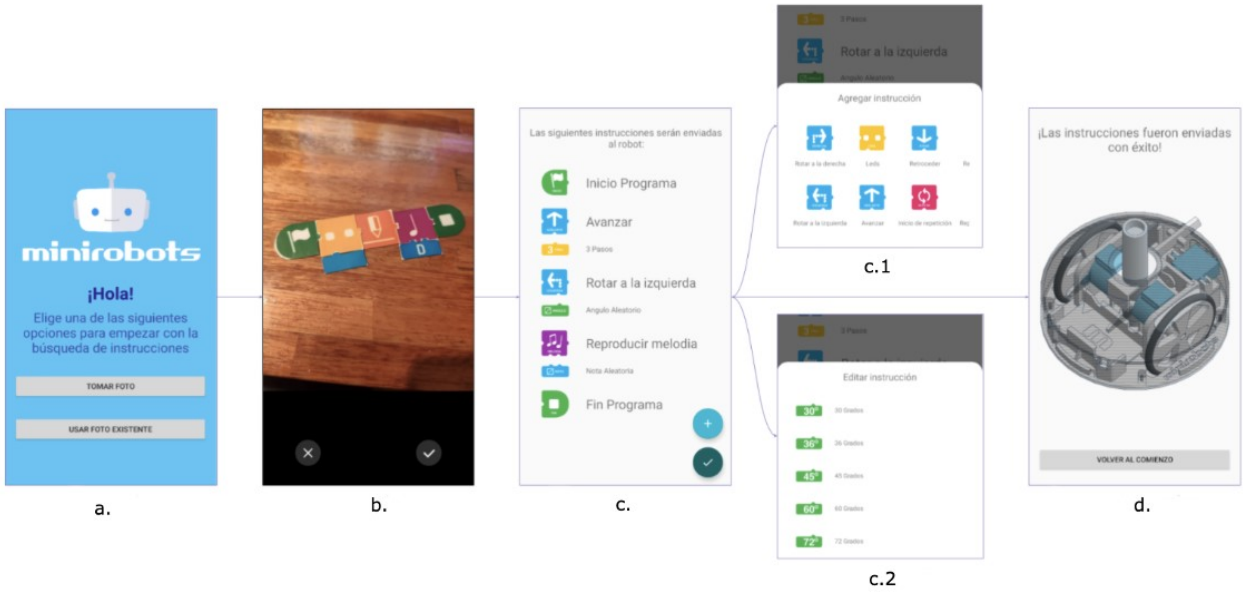

Figura 15: Flujo general de la aplicación. (a) Pantalla de Bienvenida (b) Programa a ser reconocido (c) Traducción del programa reconocido a instrucciones de MiniRobot (c.1 y c.2) Opciones de edición (f) Envío de programa exitoso

4.1. Imagen de entrada

Tal como se observa en en la Figura 15.a, la aplicación permite dos alternativas diferentes a la hora de obtener la captura que contenga la secuencia de instrucciones que se desean ejecute el Minirobot.

La primera alternativa se basa en obtener la imagen mediante el uso de la cámara nativa de Android. Esta opción permite aprovechar las prestaciones de cualquier tipo de dispositivo, sin reparar en la implementación que puedan tener los diversos fabricantes.

Adicionalmente, la aplicación permite utilizar una fotografía desde el sistema de archivos del dispositivo, permitiendo efectuar la búsqueda y reconocimiento de tarjetas en imágenes que ya han sido tomadas previamente.

4.2. Reconocimiento de instrucciones

Para reconocer las instrucciones se utiliza la técnica de Text Recognition (reconocimiento de texto) mediante la librería ML Kit[2] (ML-KIT)8, de Google que utiliza Optical Character Recognition (Mithe et al., 2013), una técnica conocida como reconocimiento de caracteres, la cual realiza reconocimiento de textos mediante un proceso en donde se consigue extraer y analizar la información que se encuentra en documentos de formato imagen. Estas imágenes pueden contener tanto texto impreso como escritura a mano, el cual luego del análisis se transforma y retorna la información en cadenas de caracteres para su posterior procesamiento.

La tecnología de reconocimiento de texto de Google es una de las más avanzadas en su tipo, diseñada especialmente para funcionar en dispositivos móviles. Los modelos de reconocimiento son actualizados periódicamente cuando la app se conecta a internet, pudiendo utilizar la aplicación sin conexión. El resultado del reconocimiento de texto es un bloque, que contiene líneas, que a su vez contienen elementos.

[2] https://developers.google.com/ml-kit

En base a estas estructuras, un parser compara cada palabra encontrada con todas las posibles instrucciones, y elige la más parecida, usando la distancia de Levenshtein. Esto permite que, si por un error, se lee "VANZAR" en lugar de "AVANZAR", se reconozca la instrucción sin problemas. Este algoritmo fue desarrollado manualmente y sin hacer uso de librerías de terceros.

4.3. Previsualización del programa

Una vez que se obtienen todas las instrucciones reconocidas, como se muestra en la Figura 15.c, estas son presentadas al usuario en una lista visual, en la que se permite agregar nuevas, eliminar, y modificar las existentes. Gracias a esto, y aunque el reconocimiento de imágenes tenga una tasa de aciertos muy elevada, se pueden rectificar fácilmente los posibles errores de reconocimiento que puedan surgir.

4.4. Transmisión del programa al minirobot

Luego de obtener la lista de instrucciones, se requiere enviarlas al robot. Para que la conexión sea más sencilla, se modificó el código del microcontrolador, logrando que el robot cree un punto de acceso WiFi, en lugar de depender de un router externo. Así, el usuario deberá conectarse al punto de acceso WiFi desde su celular, lo que será facilitado por la aplicación. Una vez conectado, las instrucciones pasan por un proceso de parsing, para ser traducidas desde las que fueron reconocidas a código entendible por el robot. Este código es enviado mediante solicitudes HTTP, y una vez recibido, es ejecutado.

4.5. Características de la app

La aplicación fue programada enteramente en Kotlin, siguiendo las prácticas y estándares usados en el desarrollo Android moderno. Su código es abierto, y se encuentra disponible en un repositorio de GitHub (`https://github.com/prietttt/Minirobots`).

Se utiliza una arquitectura separada en capas, en la que cada funcionalidad tiene su propio paquete, que a su vez se divide en presentación, dominio e infraestructura. La arquitectura de presentación elegida es MVVM, por lo que todas las interfaces de usuario cuentan con un ViewModel que se encarga de la lógica de presentación, y de comunicarse con los casos de uso.

5. Conclusiones y trabajos futuros

Minirobots app nace como un proyecto que intenta ofrecer una dinámica innovadora en el trabajo docente-alumno en nivel inicial. De hecho, con las pruebas realizadas hasta el momento en las instituciones, se observa una buena respuesta y buena retroalimentación por parte de los docentes hacia los desarrolladores.

Con respecto a la conectividad, hoy día en muchas escuelas y jardines de infantes un principal obstáculo a la hora de iniciar la implementación de un proyecto de enseñanza-aprendizaje con robótica, es la conexión a Internet. Si bien las instituciones cuentan con el servicio, los edificios no están completamente adecuados para tener el servicio de Wifi en todas las salas, lo que implica un impedimento sobre todo en los jardines de infantes, entorpeciendo el desarrollo de las actividades propuestas por los docentes. La app permite, en este aspecto, una dinámica que resuelve estos impedimentos, no distrayendo la didáctica y las actividades que formula el docente.

Como trabajo en desarrollo y futuro muy cercano, se encuentra la elaboración de secuencias didácticas junto con docentes de nivel inicial. Para ello, se trabaja en conjunto con docentes del nivel e investigadores del área del Profesorado de Nivel Inicial y Ciencias de la Computación de la Universidad Nacional del Centro (UNCPBA).

Se prevé para un futuro cercano empezar a trabajar con los alumnos y registrar experiencias en la tarea para la retroalimentación continua de esta aplicación.

Bibliografía

Aprende a programar (2021, 13 de Octubre). Lego. https://abc.gob.ar/inicial/sites/default/files/educacion_digital_inicial.pdf.

Chollet, F (2018) Deep Learning with Python. Manning Publications Co.

De la Fuente, J., Picucci, M., Bonet Peinado, D., Zurita R., Parra, G; Rodriguez, J. y Cecchi, L. (2018) "Construyendo FrankLab: Una Plataforma Web de Robótica Educativa". U Nacional del Comahue. XXIV Congreso Argentino en Ciencias de la Computación: VII Workshop de Innovación en Educación en Informática. Tandil, Buenos Aires, Argentina.

Educación digital Inicial (2021, 13 de Octubre). ABC. https://abc.gob.ar/inicial/sites/default/files/educacion_digital_inicial.pdf.

Fundación Sadosky (2021, 13 de Octubre). Program.AR|Vocaciones en TIC (manuales para docentes) http://fundacionsadosky.org.ar/publicaciones/.

Grupo de extensión de la Universidad Nacional de Córdoba (2021, 13 de Octubre). UNC http://umm.famaf.unc.edu.ar.

Hepp, P. y Jara, I (2016). (2021, 13 de Octubre) "Enseñar Ciencias de la Computación: Creando oportunidades para los jóvenes de América Latina". https://www.researchgate.net/publication/309761093_Ensenar_Ciencias_de_la_Computacion_Creando_oportunidades_para_los_jovenes_de_America_Latina.

Home - VEX Robotics (2021, 13 de Octubre). VexRobotics https://www.vexrobotics.com/.

Icaro Robótica Educativa (2021, 13 de Octubre). Roboticaro. http://roboticaro.org/.

Learning fischertechnik (2021, 13 de Octubre). Fischertechnik https://www.fischertechnik.de/en/teaching.

MiniRobots (2021, 13 de Octubre). http://minirobots.com.ar/.

Mithe, R., Indalkar, S. y Divekar, N. (2013) "Optical Character Recognition". International Journal of Recent Technology and Engineering (IJRTE) ISSN: 2277-3878, Volume-2, Issue-1.

ML-Kit (2021, 13 de Octubre). Google. https://developers.google.com/ml-kit.

Programando con Robots (2021, 13 de Octubre). UNLP. http://robots.linti.unlp.edu.ar/.

Proyecto ganador Clarín Zurich (2021, 13 de Octubre). Clarin https://premioalaeducacion.clarin.com/files/10ma-edicion/proyecto-ganador.pdf.

Recursos Inicial (2021, 13 de Octubre). Argentina. https://www.argentina.gob.ar/educacion/aprender-conectados/material-pedagogico/recursos-inicial.

Sauber, M. (1998). (2021, 13 de Octubre). "Un modelo interactivo de estimulación musical temprana". Música y Educación. Revista trimestral de pedagogía musical. Centro de Investigación en Educación Musical. ISSN: 0328-9316, Año 5. Nro 15.

AMULEN: robot educativo soberano

Carlos Maximiliano Correa
maximiliano.c.correa@gmail.com
Universidad Nacional de Río Cuarto (UNRC)

Pablo Leonel Etcheverry
etcheverrypablol@gmail.com
Universidad Nacional de Río Cuarto (UNRC)

Ariel Ferreira Szpiniak
aferreira@exa.unrc.edu.ar
Universidad Nacional de Río Cuarto (UNRC)

Resumen

A partir del Programa Conectar Igualdad, desde 2010 existen fuertes políticas públicas para reducir la brecha digital de 1er orden y 2do orden. Desde 2014, se viene dando un proceso para reducir la brecha digital de 3er orden, es decir, comprender cómo funcionan las computadoras y el mundo digital. En 2015, el Consejo Federal de Educación resolvió que la enseñanza de la programación es estratégica. A fines de 2018 se aprobaron los Núcleos de Aprendizaje Prioritarios (NAP) de Educación Digital, Programación y Robótica que abarcan a todo el territorio nacional. Varios programas se han puesto en marcha, con menor o mayor impacto. Sin embargo, la tecnología utilizada por las escuelas es diversa, generalmente con hardware (robots) y software no libres y/o importados, en muchos casos. Por tal motivo, entendemos que es necesario desarrollar íntegramente un nuevo equipamiento tecnológico, que sea soberano y pensado en las necesidades de formación y los conocimientos previos que dispone la comunidad educativa local y regional de las distintas escuelas primarias y secundarias.

AMULEN significa caminar o progresar en lengua rankülche. Su nombre nos inspira al desarrollo de un robot educativo adaptado a nuestro medio y contexto social, que satisfaga las necesidades de los distintos niveles educativos, basado en raíces locales y soberanía tecnológica. Por tal motivo, en una primera etapa del proyecto se pretende diseñar y construir AMULEN, un prototipo de robot educativo basado en Arduino. AMULEN permitirá disponer de una gran capacidad de expansión, ya sea para dar los primeros pasos en el abordaje de conceptos vinculados a la programación y robótica, como también para ser una adecuada plataforma educativa y de investigación para las escuelas. En una segunda parte, se adaptará un entorno de programación por bloques libre, se desarrollarán nuevos bloques que permitirán interactuar de manera muy simple con AMULEN, y se diseñarán guías didácticas para el aula, con desafíos y sus respectivas soluciones.

AMULEN forma parte de los Proyectos de Estímulo a la Vocación Emprendedora y del Proyecto de Investigación “Pensamiento Computacional y los NAP de Educación Digital, Programación y Robótica desde un paradigma inclusivo y con compromiso social. Aportes para la implementación de una propuesta educativa digital concreta en contexto de territorio”, ambos de la Universidad Nacional de Río Cuarto.

1. Pensamiento computacional y los Núcleos de Aprendizaje Prioritarios

Las Tecnologías de la Información y la comunicación (TIC) han traído al mundo del conocimiento nuevos conceptos y, por lo tanto, nuevas prácticas que pueden ayudar a pensar esta problemática más específicamente, dando la posibilidad de redefinirlos acorde a planteos que tengan una visión holística que permita organizar un mundo más humano e inclusivo a la luz de estas nuevas formas de abordar y crear la realidad.

En este marco, si bien insuficientemente definido, resulta útil la categoría de "pensamiento computacional". Según Wing, el pensamiento computacional se refiere a los procesos de pensamiento implicados en la formulación de problemas y sus soluciones para que éstas últimas estén representadas de forma que puedan llevarse a cabo de una manera efectiva por un procesador de información (Wing, 2011). Otros autores también giran la definición en torno a "resolución de problemas", "diseño de soluciones a través de una computadoras", etc.

De estas definiciones es necesario destacar el objetivo del pensamiento computacional en términos de "resolver problemas", "dar soluciones". Este objetivo debería constituir el eje de toda propuesta porque de suyo involucra todo el contexto al que se refieran las prácticas computacionales.

El punto es identificar el problema y analizar el alcance de su resolución. Si por ejemplo abarca un universo de personas cuyo "mundo de la vida" no tiene acceso a la digitalización y la práctica se reduce al espacio áulico, habría que redefinir la propuesta educativa. En un caso semejante no se sugiere descartar la propuesta innovadora, sino en definir el proceso por el cual se puede llegar a incorporarlas y que resulten de impacto en la vida cotidiana con sentido.

Todo conocimiento, si es socialmente relevante, debe dar soluciones, debe resolver problemas. Sin embargo, el alcance de la solución debería ser tan amplio como la inclusión que genere. Al mismo tiempo, el pensamiento computacional estimula habilidades tales como el pensamiento crítico y creativo que también se pueden corresponder a otros campos, donde originalmente la computadora no estaba pensada como una herramienta que pudiera asistir como elemento de apoyo. Todas estas capacidades deberían potenciarse en vistas a la obtención de resultados no solo eficientes sino comprometidos.

A partir del Programa Conectar Igualdad, desde 2010 existen fuertes políticas públicas para reducir la brecha digital de 1_{er} orden y 2_{do} orden. Desde 2014, se viene dando un proceso para reducir la brecha digital de 3_{er} orden, es decir, comprender cómo funcionan las computadoras y el mundo digital. En 2015, el Consejo Federal de Educación resolvió que la enseñanza de la programación es estratégica. A fines de 2018 se aprobaron los Núcleos de Aprendizaje Prioritarios de Educación Digital, Programación y Robótica (NAPEDPyR) que abarcan a todo el territorio nacional. Varios programas se han puesto en marcha, con menor o mayor impacto.

Es el propósito de los NAPEDPyR responder justamente al imperativo de la época, la inclusión socio-educativa en medio de un veloz desarrollo científico-tecnológico atendiendo a las condiciones sociales en

medio del territorio al cual se aplica.

Los NAPEDPyR involucran estrictamente “una propuesta integral de innovación pedagógica y tecnológica que comprende como núcleos centrales el desarrollo de contenidos, el equipamiento tecnológico, la conectividad y la formación docente, que ayude tanto al desarrollo de las competencias de educación digital, como de las capacidades y saberes fundamentales” (Resolución CFE Nº 343/18).

Lo interesante de la fundamentación es la referencia a los cambios y desafíos constantes que las nuevas tecnologías imponen a la sociedad y la necesidad de adecuar estas nuevas formas de praxis social a las condiciones concretas de vida de los niños y niñas en edad escolar.

Uno de los aspectos sobresalientes de los NAPEDPyR es precisamente que las nuevas tecnologías sirvan para “la integración plena a la sociedad y al mundo del trabajo” explicitando que una vez comprendido cómo se construyen los sistemas digitales y cómo se crean, puedan hacer uso crítico y creativo de las tecnologías (Resolución CFE Nº 343/18). A esto se le agrega que dichos “saberes se constituyen en referentes y organizadores de la tarea cotidiana de enseñanza, en la medida en que los maestros y profesores los deconstruyen y reconstruyen, resignificándolos en función de atender a la heterogeneidad de las trayectorias escolares de sus estudiantes” (Resolución CFE Nº 343/18).

La problemática del territorio aparece también, entonces, en la fundamentación de los NAPEDPyR: “la identificación colectiva de ese núcleo de aprendizajes prioritarios sitúa a cada una de ellas, sobre la base de sus particularidades locales en sus respectivos marcos regionales, en oportunidad de poner el acento en aquellos saberes considerados comunes “entre” jurisdicciones e ineludibles desde una perspectiva de conjunto. [...] Desde esa perspectiva, las acciones que se orienten al trabajo con un núcleo de aprendizajes prioritarios deben fortalecer al mismo tiempo lo particular y los elementos definitorios de una cultura común, abriendo una profunda reflexión crítica desde la escuela sobre las relaciones entre ambas dimensiones y una permanente reconceptualización de lo curricular.” (Resolución CFCyE 225/04, Anexo, p. 5.)

Así, si el pensamiento computacional se impone como nueva forma de abordar y crear realidad y dada la existencia de una formulación adecuada de los NAPEDPyR, nos queda generar fuentes de aplicación real y concreta que generen una transformación en términos éticos, inclusivos y abiertos.

2. El problema

Dadas las consideraciones precedentes, es preciso generar tanto prácticas concretas en espacios áulicos, así también como formaciones, capacitaciones y producciones teóricas que garanticen el vínculo “conocimiento-sociedad” (en términos de inclusión socio-educativa), evitando de esta manera que la educación digital se limite a equipar escuelas, es especial con tecnologías que responden a otras realidades socio-culturales y podrían no resultar aptas para recrearlas o adaptarlas a nuestros contextos, como los dispositivos cerrados provenientes de empresas internacionales que proveen un mismo paquete tecnológico a todos los países que deciden adoptarlo, sumado a software propietario sin posibilidades de adaptación.

En síntesis, la tarea debería abarcar todas las dimensiones descriptas, en prácticas concretas, tanto en los contextos áulicos como en la producción de material didáctico y teórico que se referencien como soportes de una integración real, auténtica y efectiva. El desafío es entonces generar estos dispositivos, ejecutarlos y evaluarlos en función de un constante proceso de contrastación con los NAPEDPyR.

El objetivo general del proyecto de investigación es analizar críticamente la concepción de "pensamiento computacional" y su puesta en práctica, en relación a los Núcleos de Aprendizaje Prioritarios de Educación Digital, Programación y Robótica con el fin de evaluar si tienen un impacto positivo en los ámbitos socio-educativos en los que se aplican y proponer nuevas prácticas áulicas y nuevos dispositivos acordes a una integración socio-educativa en un sentido humano e inclusivo. Dentro de los objetivos específicos nos planteamos desarrollar dispositivos (materiales educativos, actividades de capacitación, objetos digitales interactivos, software) para estudiantes relacionados con los NAPEDRyP de acuerdo al diagnóstico realizado. En ese contexto nace AMULEN, un proyecto anclado en los NAPEDRyP.

3. Desarrollo de AMULEN

AMULEN significa *caminar* o *progresar* en lengua *rankülche* o *ragkülche*. Los Rankülches, más conocidos como Ranqueles, son un pueblo originario del suelo que habitamos en el sur de la provincia de Córdoba, La Pampa, sur de San Luis y oeste de Buenos Aires. *Rangkül* significa *caña* o *carrizo*, y *che* quiere decir *persona* o *gente*, es decir *gente de los cañaverales*. Su nombre nos inspira al desarrollo de un robot educativo adaptado a nuestro medio y contexto social, que satisfaga las necesidades de los distintos niveles educativos, basado en raíces locales y soberanía tecnológica.

Por tal motivo, en una primera etapa del proyecto se diseñó y construyó el primer prototipo de AMULEN, un robot educativo basado en Arduino. AMULEN permite disponer de capacidad de expansión, ya sea para dar los primeros pasos en el abordaje de conceptos vinculados a la programación y robótica, como también para ser una adecuada plataforma educativa y de investigación para las escuelas. En una segunda etapa, se adaptó un entorno de programación por bloques libre y se desarrollaron nuevos bloques que permiten interactuar de manera muy simple con AMULEN.

AMULEN forma parte de los Proyectos de Estímulo a la Vocación Emprendedora (segunda y tercera convocatoria) y del Proyecto de Investigación "Pensamiento Computacional y los NAP de Educación Digital, Programación y Robótica desde un paradigma inclusivo y con compromiso social. Aportes para la implementación de una propuesta educativa digital concreta en contexto de territorio" (2020-2022), ambos aprobados y financiados por la Universidad Nacional de Río Cuarto.

En la actualidad existe, tanto a nivel nacional como internacional, una creciente demanda insatisfecha de profesionales en las áreas de las ciencias exactas e ingenierías. Es por eso que, con el fin de generar vocaciones por carreras relacionadas con las denominadas STEM (acrónimo en inglés de Ciencia, Tecnología, Ingeniería y Matemática), se definió un marco de trabajo dentro de este proyecto AMULEN lo más similar a un proceso

productivo vigente en las Software Factory del País, ya que se consideró, desde la organización del proyecto, sumamente importante la incorporación, en la currícula de los participantes (estudiantes becados), un proyecto que fortalezca tanto las habilidades blandas como duras de los mismos.

En relación a este proceso, se definieron ciertos pasos previos, por lo que los participantes debieron indagar e investigar sobre la transformación digital por la que han atravesado las instituciones educativas en relación a los contextos sociopolíticos de la última década, como para poder identificar un producto disruptivo de los tradicionales vistos hasta el momento, como también comenzar a indagar en la factibilidad de producirlo dentro del país, sin depender de la importación de partes inherentes al desarrollo.

Como primera instancia, se decidió investigar diversas empresas destinadas a brindar hardware y software orientados a la robótica educativa. Abarcando una lista reducida de organizaciones existentes, entre las cuales podemos mencionar IT10, RobotGroup, LEGO, Robobloq, Mis Ladrillos y Rasti. A cada una de estas empresas se las analizó, teniendo en cuenta: catálogo de productos ofrecidos, estrategias de marketing, estrategias de asistencia al cliente, puntos de distribución, centralización de las actualizaciones, accesibilidad de los materiales de repuesto, experiencia del cliente, reutilización y reparación de los elementos utilizados. También se decidió analizar el catálogo de productos de las mismas, vigentes en el mercado, con la intención de conocer las bondades como puntos de mejora a la hora de trabajar con ellos como material educativo. Analizando casos de aplicación a diferentes edades educativas, documentación que acompaña al equipamiento, accesibilidad a los materiales de repuesto, autonomía del equipamiento (fuente de alimentación), robustez en cuanto a lo mecánico.

Como última instancia de investigación, se realizó una encuesta con el objetivo de recolectar información sobre el contexto actual de las escuelas primarias y secundarias de nuestra región en relación a las TIC utilizadas.

Se logró obtener información que fué agrupada en 3 grande categorías:

- *Capacitación*

  Uno de los principales problemas de incorporar la robótica y programación en las escuelas es la escasa capacitación de los docentes, esto se vió reflejado en las encuestas, siendo un gran porcentaje de las respuestas.

- *Equipamiento*

  Se consultó a los colegios por la disponibilidad de salas de computación o salas digitales móviles; la mayoría indicó que disponen de tales y que el sistema operativo en las máquinas es Windows el más utilizado. Un poco menos de la mitad de las respuestas indicaron que no poseen Kits o Tecnofactos, mientras que las escuelas que sí disponen de dichas tecnologías facilitadoras, los más utilizados son los kits de robótica, los juguetes STREAM / robots y las pizarras digitales.

- *Experiencia del cliente*

  La experiencia de cliente incluye muchos aspectos, tanto la atracción y el interés que genera el

marketing, el descubrimiento, la compra, el uso, la asistencia al cliente y hasta la baja de las campañas comerciales.

Se consultó acerca de cómo se enteraron de la existencia de estos productos y en su gran mayoría ha sido gracias a las autoridades educativas. En cuanto a la compra, se les preguntó si los productos llegan listos para usarse y la mayoría de las respuestas fueron negativas; han tenido que ensamblar o configurar el producto. Finalmente, en la etapa de atención al cliente, la mayoría de las empresas, no hacen un seguimiento según las respuestas obtenidas en las encuestas.

En relación a la etapas de investigación, se detectó que los principales desafíos tienen que ver principalmente con la educación, ya sea con el mejoramiento de la absorción de conocimientos por parte del estudiante, o facilitando la tarea de capacitación y enseñanza a los docentes, asi como también el mejoramiento de la accesibilidad para personas de distintas edades.

Destacando de esto, tres ramas principales que el desarrollo deberá satisfacer:

- *Orientado a estudiantes*

  Lo principal en este campo es intentar utilizar las tecnologías disponibles, la impresión 3D y los sistemas electromecánicos (físicos) bajo licencias de software libre, para maximizar la absorción de conocimientos dentro del aula de una manera lúdica y recreativa.

- *Orientado a educadores*

  De esto desprenden dos aristas: una es la educación hacia el profesor por parte del proyecto y la otra es la educación del estudiante por parte del docente. A su vez, de éstas surge la propuesta de realizar contenidos/materiales online, tanto del uso como del mantenimiento. Todo esto acompañado con documentación escrita y descriptiva de cada componente y sus funciones, para su aplicación en el día a día en las aulas.

- *Accesibilidad*

  En este caso, se destaca la importancia de intentar definir un producto amigable y con facilidades para personas de distintas edades. Asumiendo que, para cierto rango etario, las partes pequeñas de los kits diseñados dentro del aula, podría generar algún accidente.

Finalizada la etapa investigación, con el objetivo de hacer viable y accesible nuestro proyecto y, como una alternativa al uso de los kits robóticos comerciales vigentes, este equipo de trabajo plantea desarrollar, bajo la modalidad hardware y software libre, un kit de bajo costo que responda a las siguientes características: (i) modular, (ii) polimórfico basado en tecnología 3D y (iii) programable mediante bloques.

La intención es que pueda ser empleado por estudiantes de escuelas primarias y secundarias, e incluso, que pueda ser utilizado en cursos de Introducción a la Programación a nivel universitario, posibilitando de esta manera, la implementación y puesta en práctica de conceptos tales como:

- construir su propio conocimiento, pues los estudiantes desempeñan un papel activo durante la programación del robot;

- acercarse a la ciencia y la tecnología de una manera práctica, lúdica, motivadora y significativa;
- trabajar en equipo, es decir, realizar un trabajo colaborativo y asumiendo responsabilidades;
- compartir sus conocimientos con el resto de sus compañeros; y
- desarrollar la creatividad y el interés por la investigación y la innovación, al fomentar la observación, la medición y la comparación sistemática.

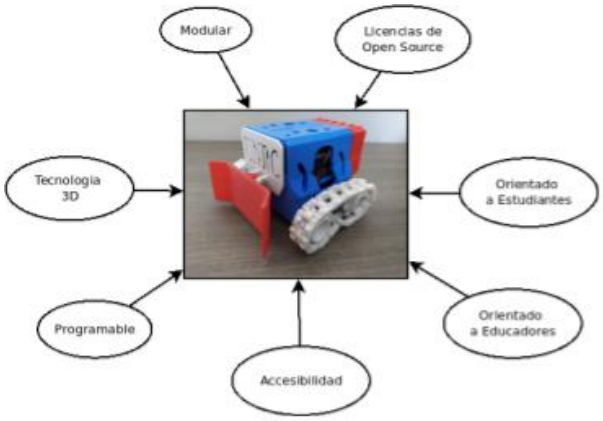

Figura 1: Características de AMULEN

3.1. Proceso productivo y metodología de desarrollo

Con el fin de trabajar siguiendo el ritmo actual de las empresas de investigación y desarrollo el proyecto AMULEN fué desarrollado mediante la metodología de trabajo Scrum. Poniendo foco en las personas, haciendo que se sientan apoyadas y motivadas, fomentando la responsabilidad y autonomía del equipo mediante la fluidez en la comunicación y la participación equitativa, facilitando la toma de decisiones mediante revisiones continuas, al mismo tiempo que se va mejorando la experiencia del cliente final (escuelas) teniendo feedbacks con frecuencia. Éstos, son algunos de los beneficios de la cultura agile, que vemos necesario mencionar, ya que revela desde la formación, el pilar fundamental del proyecto y lo que se ha trabajado durante los meses del ciclo de vida del mismo.

Esta metodología se basa en la teoría de control de procesos empírica. El empirismo asegura que el conocimiento procede de la experiencia y de la toma de decisiones basándose en lo que se conoce.

Para esto emplea un enfoque iterativo e incremental, el cual permite optimizar la predictibilidad y el control del riesgo, al dividir el trabajo en etapas; de modo que se pueda mejorar continuamente el producto, el equipo y el entorno de trabajo. La metodología consiste en Equipos Scrum y roles, eventos, artefactos y reglas asociadas.

Al considerarse al equipo y sus integrantes como uno de los elementos principales de dicha metodología, el primer paso para el desarrollo de AMULEN consistió en identificar los roles y competencias de cada uno de los miembros. Entre los roles principales se identificaron al cliente o product-owner (el cual solicita el

producto), el scrum master (quien se encarga de guiar y organizar al resto del equipo), el technical leader (quien se ocupa de guiar y organizar al equipo en cuestiones técnicas de desarrollo de hardware y software) y los developers (encargados del desarrollo del producto).

Rol	**Responsabilidad**
Gestor de configuración	Confeccionar el plan de gestión de configuración. Generar y actualizar las líneas base de acuerdo a lo planificado. Generar integración de código, versiones y releases de acuerdo a la estrategia planteada por el proyecto. Generar reporte de estado de ítems de configuración.
Líder de proyecto / Scrum master	Solicitar la generación de líneas base y releases cuando sea apropiado. Asegurar la integridad y trazabilidad de los productos desarrollados por el equipo de proyecto.
Miembros del equipo de proyector	Colaborar en la generación de líneas base, versiones y releases cuando sea necesario. Mantener la gestión de configuración en los repositorios definidos para el proyecto.
Responsable de asegura-miento de calidad	Auditar el proceso de gestión de configuración definido en la adaptación del proceso al tipo de proyecto. Verificar la integridad y trazabilidad de los productos desarrollados por el equipo de proyecto.

Tabla 1

Una vez identificados los roles y competencias de cada individuo, se determinó la duración de cada una de las etapas de trabajo. Estas etapas se dividieron en 20 Sprints con una duración de 15 días (estos Sprints consisten en períodos de trabajos en los cuales el equipo decide resolver una serie de tareas). Dentro de cada uno de estos se identificaron diversos eventos, entre los que se incluyen una planing (en la cual se deciden las tareas a realizar durante el sprint), una demo o presentacion del avance realizado (la cual será presentada al cliente por todos los miembros del equipo Scrum) y una retrospectiva en la que los miembros expresan el proceso durante el sprint (lo bueno, lo malo y lo que hay que mejorar del mismo).

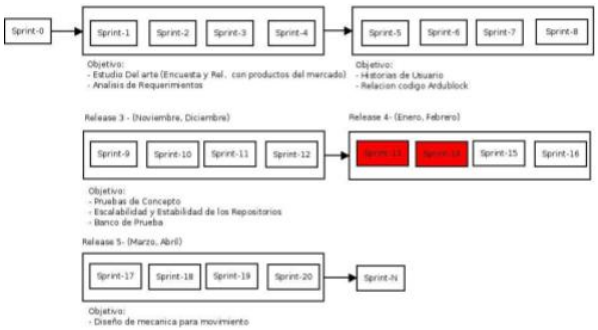

Figura 2: Sprints para el desarrollo de AMULEN

Además cada una cierta cantidad de Sprints, cuatro en el caso de AMULEN, se realizó una versión release, en la que debía entregarse una instancia finalizada del producto final (beta).

3.2. Programación mediante bloques

Existe un consenso generalizado sobre la importancia de la enseñanza de las ciencias de la computación en las escuelas, y en particular la programación, con el objetivo de difundir el pensamiento computacional.

Recientemente ha surgido un creciente interés por los lenguajes de programación visual (LPV). Esta clase de programación se refiere al desarrollo del software que emplea y manipula dinámicamente elementos visuales (como bloques, íconos, gráficos, etc.) de manera parcial o total. De esta manera, los programas son creados con objetos visuales que representan distintas instrucciones del programa, mediante la combinación de los mismos como si fueran piezas de un rompecabezas.

De esta manera, se evita la frustración que producen los errores de sintaxis que suelen presentarse en los lenguajes de programación de alto nivel, ya que cualquier pequeño error en el código (como la falta de una coma, por ejemplo), puede hacer que el programa realizado no funcione correctamente. También permite, debido a que los bloques mencionados se encuentran agrupados por categorías de acuerdo a su función, que los programadores puedan encontrar el bloque que necesitan en lugar de tener que recordar el nombre de la instrucción, eliminando de esa forma, otro problema que se presenta con la programación por código.

Partiendo de esta base, el proyecto AMULEN decidió expandir el desarrollo provisto por Ardublock, definido como un entorno de programación visual (construido como un plugin de java), el cual se usa para programar placas Arduino, que permite una conexión con dispositivos físicos, de una manera más fácil e intuitiva sin la necesidad que escribir código.

Existen múltiples métodos y plataformas por medio de los cuales es posible generar programación en bloques, sin embargo (si bien en la actualidad hay un auge principalmente en las aplicaciones para dispositivos móviles) el equipo de AMULEN utilizó ArduBlock ya que en su mayoría las instituciones educativas cuentan con computadoras, mientras que no ha sido posible asegurar que los estudiantes tengan un dispositivo apto para una aplicación de este tipo.

Además, otra razón para preferir la utilización de las computadoras, es el hecho de que Arduino ha sido diseñado para funcionar mediante este método; porque si bien existen aplicaciones compatibles con dispositivos móviles, la utilización de placas y la programación de las mismas, resulta mucho más intuitiva y sencilla desde una computadora.

Existen diversas razones por las que se determinó la utilización de ArduBlock:

- es multiplataforma: ArduBlock, al ser un plugin de java, se ha desarrollado para ser utilizado en cualquier sistema operativo de computadoras;
- diseño intuitivo: debido al diseño simple de sus bloques y a la forma en la que encastran unos con otros, se hace sencillo identificar cuáles son compatibles entre sí y como deben utilizarse para generar

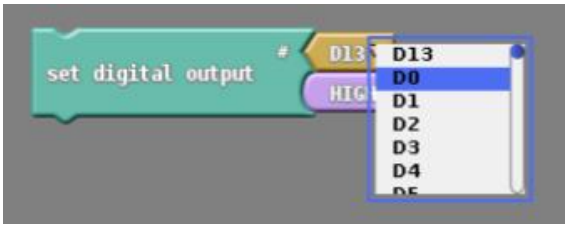

Figura 3: Bloque típico de ArduBlock

un código funcional;

- es software libre: no es necesario pagar por licencias para su uso y permite a los desarrolladores acceder a su código con el fin de agregar nuevas funcionalidades y/o mejorar las existentes;
- utiliza el IDE de Arduino: a diferencia de otros programas de programación visual, Ardublock no genera el código de sus bloques en sí mismo, sino que lo hace dentro del mismo IDE de Arduino, siendo más claro para el usuario; y
- portabilidad: ArduBlock cuenta con una versión portable, es decir, no se necesita realizar un proceso de instalación complejo, sino que basta con copiar el contenido de la carpeta descargada en la ruta correcta.

Las funciones y posibilidades que nos ha brindado Ardublock, son las mismas que nos ha ofrecido Arduino IDE, es decir, nos fue posible conectar Ardublock a una placa Arduino y, de esa forma, enviar el código creado en Ardublock gracias a los bloques y testear nuestros proyectos de una forma rápida y sencilla, ya que una vez finalizado el programa, la información guardada no deja de ser código escrito; código que ha creado Ardublock con nuestros bloques.

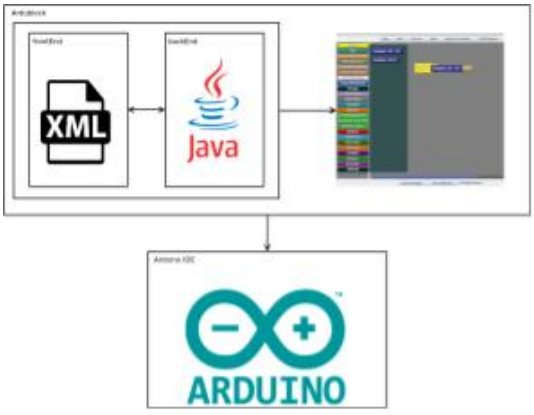

Figura 4: Forma de funcionamiento de ArduBlock

3.3. AMULEN V1.0

El proyecto AMULEN, en su primera versión, ha reflejado una arquitectura, tanto de un sistema de programación y automatización de bajo coste, como también una metodología de trabajo similar a la cadena productiva vigente en las Software Factory.

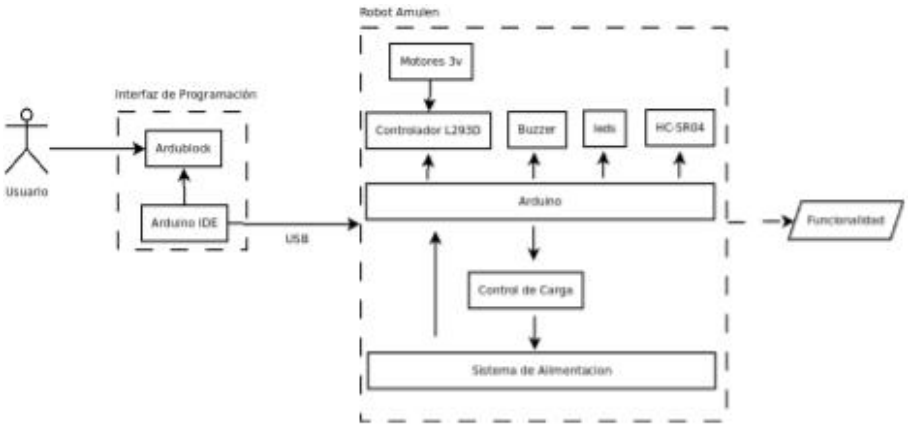

Figura 5: Sistema electromecánico (placa y elementos arduino-compatibles)

De tal modo que la fusión de ambas aristas dió como resultado un sistema modular que permite la enseñanza mediante programación asistida por bloques. Como se muestra en la figura anterior, el sistema electromecánico (placa y elementos arduino-compatibles) responde a cada "caso de uso" definido en las instancias de planificación e investigación.

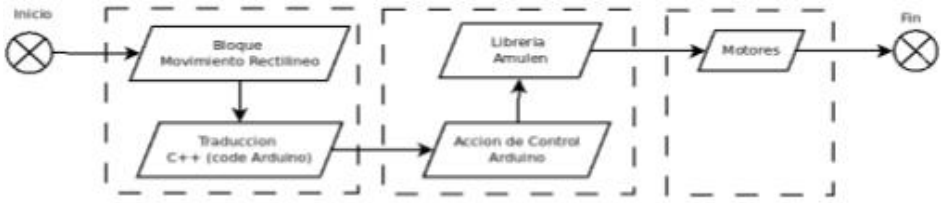

Figura 6: Lazo abierto de control de movimiento rectilíneo

En esencia, cada bloque definido en Ardublock, tiene una relación directa con el lazo abierto de control definido en Arduino para con los elementos que se comunican con él, dando como resultado un automatismo físico tan simple como un movimiento rectilíneo uniforme.

Durante la etapa de investigación se identificaron varios "casos de uso" dentro de los cuales es posible destacar el de Gestión de Movimiento, compuesto por diferentes subprocesos que permiten en su totalidad interactuar con el usuario final en diferentes actividades prácticas, tal como se muestra en la figura 7.

Esto dió como resultado un prototipo tecnofacto ampliamente escalable, programado de inicio a fin, bajo licencias libres tanto en su programación como en su electrónica.

Figura 7: Interfaz en ArduBlock para la gestión de movimiento

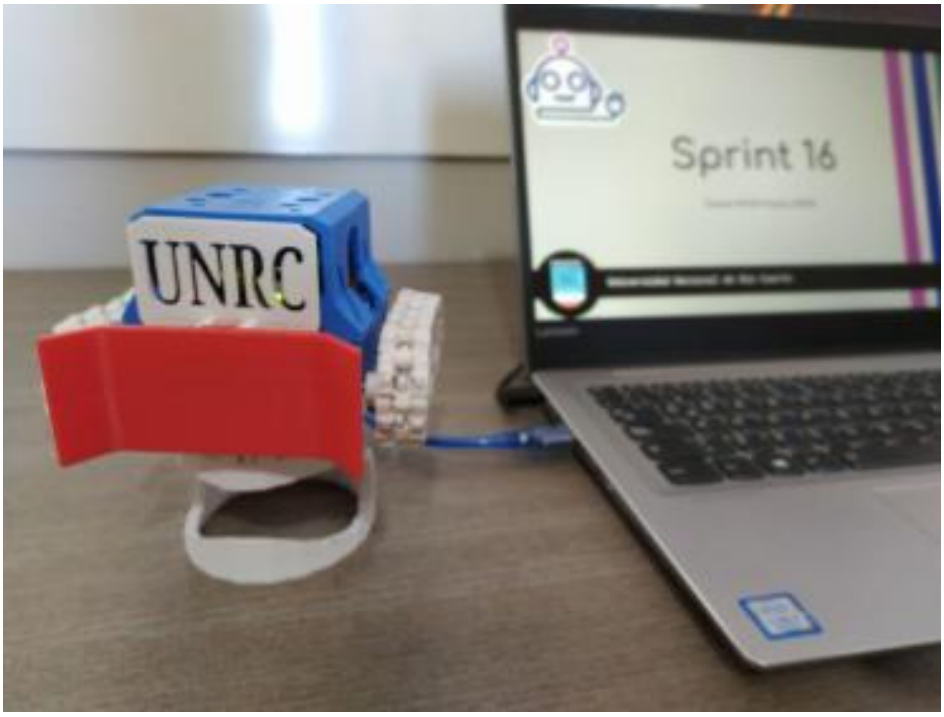

Figura 8: Prototipo de AMULEN analizado en el Sprint 16

4. Conclusiones y trabajo futuro

El trabajo realizado por estudiantes de grado, docentes-investigadores y graduados de la Universidad Nacional de Río Cuarto posibilitó diseñar y construir AMULEN, un prototipo de robot educativo libre

basado en Arduino, que posibilita abordar conceptos vinculados al pensamiento computacional y elaborar programas de manera sencilla mediante bloques propios en el entorno de programación Ardublock.

Existen varias líneas de trabajo futuro en relacion al prototipo y al diseño propuesto. Si bien el análisis de requerimientos fue exhaustivo, visitando a varias entidades educativas, aún resta un análisis más detallado al interior de cada nivel de la educación inicial, primaria (primer y segundo ciclo) y secundaria (ciclo básico y ciclo orientado). Por otro lado, es necesario realizar mejoras tanto en la funcionalidad como en el diseño del prototipo, de las cuales, en relación al software y el hardware podemos mencionar:

- nuevos casos de uso;
- seguidores de línea y control de servomotores;
- módulo de recarga de batería;
- gestión de mantenimiento de cada robot;
- migración a un entorno web y aplicación para dispositivo móvil;
- programación mediante protocolos inalámbricos;
- fusión de todos los elementos de control en una placa única central; y
- mayor robustez, autonomía y resistencia a golpes y caídas.

Para la próxima etapa de desarrollo se plantea diseñar el prototipo final, analizar la factibilidad de pasar del prototipo a un producto, estimar el costo de producción en serie, garantizar que cumpla con los requisitos de ser libre, de bajo costo, industria nacional, pensado en los NAPEDPyR y en las necesidades de formación y conocimientos previos que dispone la comunidad educativa local y regional. Por otro lado, diseñar materiales de aprendizaje, con desafíos y soluciones comentadas, que propongan utilizar AMULEN en el nivel inicial, primario y secundario para abordar los conceptos de pensamiento computacional desde un paradigma inclusivo, en contexto de territorio, con compromiso social y en concordancia con los NAPEDPyR. Finalmente, continuar la capacitación del equipo en desarrollo emprendedor, programación y robótica y analizar las diferentes posibilidades de constitución jurídica del equipo para formar parte de la incubadora de empresas de la universidad.

Bibliografía

Ananiadou, K. y Claro, M. (2009). 21st Century Skills and Competences for New Millennium Learners in OECD Countries, OECD Education Working Papers, No. 41, OECD Publishing. http://dx.doi.org/10.1787/218525261154

Asimov, I. (2007). Yo, Robot. ISBN 978-84-350-3482-1. Edit. EDHASA

Astudillo, G. J., Bast, S. G., y Willging, P. A. (2016). Enfoque basado en gamificación para el aprendizaje de un lenguaje de programación. Virtualidad, Educación y Ciencia, 7(12), 125–142. https://revistas.unc.edu.ar/index.php/vesc/article/view/14739

Astudillo, G. J., Bast, S. G., Willging, P., Segovia, D. , Castro, L. y Distel, J. M. (2019) Estrategias innovadoras en los procesos de enseñanza y de aprendizajes de informática. XXI Workshop de Investigadores en Ciencias de la Computación. WICC19 (pp.600–604) . San Juan. ISBN 978-987-3984-85-3

Astudillo, G.J., Bast, S.G., Willging, P., Segovia, D.; Castro, L., Distel, J., Lucero P. y Lobos, M. (2019) Póster Propuestas de Enseñanza y de Aprendizaje de la Programación. Hacia un estado del Arte. Segundas JADiPro. Córdoba. 7 y 8 de junio 2019

Bordignon, F., y Alejandro, I. (2015). Diseño y construcción de objetos interactivos digitales. Edit. UNIPE Editorial Universitaria. https://libros.unlp.edu.ar/index.php/unlp/catalog/book/478

Burbules, N. (2011). Entrevista a Nicholas Burbules. Educación y tecnologías: las voces de los expertos. Conectar Igualdad (pág. 196). CABA, Argentina: ANSES. https://www.academia.edu/

Capek, K. (1921) RUR. ROBOTS UNIVERSALES ROSSUM. Karel Capek. 1921. Traducción por Consuelo Vázquez de Parga. 2004. ISBN 978-84-672-0808-5. https://www.ciencia-ficcion.com/opinion/op02125.htm

Compañ-Rosique, P., Satorre-Cuerda, R., Llorens-Largo, F., y Molina-Carmona, R. (2015). Enseñando a programar: un camino directo para desarrollar el pensamiento computacional. RED- Revista de Educación a Distancia, 46 (11). https://revistas.um.es/red/article/view/240191/182931

Cucuzza, G. (2019). Sobre los NAP de Educación Digital, Programación y Robótica: ¿Otra vez sopa? https://paraquesepan.blogspot.com/2019/01/sobre-los-nap-de-educacion-digital.html

Ferreira Szpiniak, A., y Etcheverry, P. (2019) Herramientas para descubrir vocaciones científicas y vincular la universidad con la escuela secundaria: talleres, olimpiadas y festivales de robótica y programación. Segundas JADiPro. Córdoba. 7 y 8 de junio 2019.

Ferreira Szpiniak, A., y Locati, M. (2019) ¿Para qué la robótica en la escuela? Una mirada nacional situada en la Provincia de Córdoba. Segundas JADiPro. Córdoba. 7 y 8 de junio 2019.

Fernández Enguita, M. (2017). La brecha digital es ya una brecha escolar. EcoAula (El Economista). https://blog.enguita.info/2017/02/la-brecha-digital-es-ya-una-brecha.html

Maggio, M. (2012). Enriquecer la enseñanza. Los ambientes con alta disposición tecnológica como oportunidad. Buenos Aires. Editorial Paidós.

Monsalves González, S. (2011). Estudio sobre la utilidad de la robótica educativa desde la perspectiva del docente. Revista de Pedagogía, 32(90), 81–117. https://www.redalyc.org/articulo.oa?id=65920055004

Müller, G (2004). El vínculo tecno-científico con el mundo. El proyecto moderno y los desafíos éticos contemporáneos. Ediciones de ICALA. ISBN 987-20969-5-3.

Serrano A., y Martínez, E. (2003): La Brecha Digital: Mitos y Realidades, Editorial UABC, México, en https://www.labrechadigital.org

Willging, P. A., Astudillo, G. J, Bast, S., Occelli, M., Castro, L., y Distel. J. (2017). Educación con Tecnologías: la Robótica Educativa Aplicada para el Aprendizaje de la Programación, Memorias WICC 2017, XIX Workshop de Investigadores en Ciencias de la Computación, RedUNCI, 1174–1178. (ISBN: 978-987-42-5143-5).

Wing, J. M. (2006). Computational Thinking. Viewpoint, Communications of the ACM, Vol. 49, 33–35. https://www.cs.cmu.edu/~./15110-s13/Wing06-ct.pdf

Wing, J. M. (2011). Computational Thinking: What and Why? The Link Magazine, Spring.

Zabala, G. (2012). Robots o el sueño eterno de las máquinas inteligentes. ISBN 978-987-629-222-1. http://www.sigloxxieditores.com.ar

SESIÓN 5:
CONTENIDOS Y DIDÁCTICA

Moderador: *Dr. Marcos J. Gómez (UNC, Fundación Sadosky)*

Aportes para la construcción de la didáctica de las Ciencias de la Computación: un instrumento para el análisis de secuencias didácticas
Nerina Menchón, Carmen Leonardi y Virginia Mauco

El desarrollo de habilidades de resolución de problemas como un saber necesario en el camino del aprendizaje de la programación
Fernando Raúl Alfredo Bordignon y Alejandro Adrián Iglesias

Evaluación de producciones de docentes de primaria en formación en Pensamiento Computacional
Francisco Bavera, Teresa Quintero y Marcela Daniele

Programar es mucho más que solamente secuenciar comandos
Pablo E. "Fidel" Martínez López

Aportes para la construcción de la didáctica de las Ciencias de la Computación: un instrumento para el análisis de secuencias didácticas

Nerina Menchón*
nmenchon@fch.unicen.edu.ar
UNCPBA

Carmen Leonardi†
cleonard@exa.unicen.edu.ar
UNCPBA

Virginia Mauco†
vmauco@exa.unicen.edu.ar
UNCPBA

Resumen

El presente artículo se enmarca en la experiencia del Taller de Práctica Docente de una Especialización en Didáctica de las Ciencias de la Computación para el nivel primario. Como Trabajo Final de este taller, los cursantes diseñaron una secuencia didáctica para abordar algunos de los contenidos desarrollados a lo largo del postítulo. Precisamente, este trabajo presenta el instrumento que se diseñó para el análisis de esas secuencias didácticas, constituido de cinco dimensiones relativas al qué, cómo, para quiénes, por qué y para qué y a través de qué enseñar, definiendo para cada una de ellas diversos interrogantes que orientan el abordaje del estudio. Si bien este instrumento fue diseñado con la finalidad de poder llevar a cabo el análisis de un caso en particular, se destaca por su potencialidad y versatilidad para adaptarse y así ser reutilizado en otros casos o investigaciones acerca de las prácticas educativas en el campo de las Ciencias de la Computación. Este instrumento fue usado para el análisis de la selección de un conjunto de diez secuencias didácticas, desarrolladas por los/as cursantes del Taller de Práctica Docente, que abordaron contenidos de Ciencias de la Computación tales como: algoritmo, lenguaje simbólico, programa y repetición simple.

Por otro lado, se realizó un estudio exploratorio más amplio sobre la experiencia vivenciada en el Taller, desde un enfoque integrado que combina tanto metodología cuantitativa como cualitativa. Para eso, se recuperaron los resultados obtenidos de la implementación del instrumento propuesto, en complementación con otras técnicas de recolección de datos (como la observación participante) y el análisis de otras fuentes como encuestas, registros de las clases y demás producciones realizadas por los/as cursantes. De esta forma, se intenta en este trabajo reconocer potencialidades, desafíos, problemáticas y aspectos significativos que puedan aportar a la construcción de la didáctica de las Ciencias de la Computación para el nivel primario.

Palabras clave: Didáctica de las Ciencias de la Computación, Instrumento de análisis, Secuencias didácticas, Nivel Primario.

*Facultad de Ciencias Humanas, UNCPBA

†INTIA, Facultad de Ciencias Exactas, UNCPBA

1. Introducción

En las últimas décadas, se ha evidenciado el avance de la programación, la robótica y el pensamiento computacional (Cuny, Snyder y Wing, 2010) en el campo educativo. A nivel mundial, varios países han incorporado la enseñanza de las Ciencias de la Computación (CC) en sus programas y diseños curriculares (Martinez y Echeveste, 2020). En Argentina, también hace varios años que se ha planteado la importancia de enseñar CC y el desarrollo del Pensamiento Computacional en la escuela (Fundación Sadosky, 2016, p. 12). Finalmente, los contenidos de las CC fueron integrados de manera transversal, a partir de los Núcleos de Aprendizajes Prioritarios para la Educación Digital, Programación y Robótica, que fueron aprobados por el Consejo Federal de Educación (CFE, 2018). Este hecho es un gran logro, pero aún queda mucho camino por recorrer porque el campo de la **didáctica de las CC** aún se encuentra en construcción en nuestro país, "lo cual hace necesario seguir pensando la formación docente sobre la base de procesos críticos y reflexivos y sin separarse de la formación de ciudadanos comprometidos con el contexto tecnológico actual" (Leonardi, Mauco, Felice y Menchón, 2021).

Para poder crear propuestas de actividades y secuencias didácticas abordando o vinculando temas de CC, es necesario que los docentes piensen en múltiples cuestiones relativas a ¿qué contenidos de las CC pueden ser enseñados acorde al nivel y grupo de clase? ¿qué metodologías pedagógico-didácticas y modalidades de evaluación pueden ser las más adecuadas para la enseñanza de las CC? ¿por qué y para qué enseñar estos contenidos? ¿cómo se puede favorecer el desarrollo del pensamiento computacional? ¿a través de qué recursos educativos o medios didácticos pueden proponerse distintos tipos de actividades? En ese sentido, es fundamental que los docentes no caigan en una visión simplista y reduccionista acerca de la enseñanza de estos contenidos que sólo quede ligada a saber usar programas informáticos o conocer los componentes de una computadora y reconocer el potencial de la enseñanza de las CC y del desarrollo del pensamiento computacional (Fundación Sadosky, 2016).

Desde esta perspectiva y fundamentos se planteó el desarrollo del Taller de Docencia (TD) de la *Especialización Docente de Nivel Superior en Didáctica de las Ciencias de la Computación para la Educación Primaria*. El TD, desarrollado en 2020, forma parte del diseño curricular de la Especialización (DGCyE, 2018). Tuvo por propósito que los/as cursantes pudiesen recuperar y resignificar los temas y contenidos abordados en las distintas prácticas integradoras y asignaturas, siendo capaces de poder diseñar una secuencia didáctica sobre CC de manera crítica y reflexiva, a partir de concebir al planeamiento didáctico como herramienta del trabajo docente útil para abordar la complejidad propia del proceso de enseñanza y de aprendizaje al llevarse a cabo en situaciones caracterizadas por la multiplicidad de variables y factores intervinientes. Es importante tener en cuenta que al ser el TD la última asignatura de la Especialización, al momento de cursarlo, los/as cursantes ya tenían un trayecto formativo previo y un bagaje teórico en lo que respecta a los saberes propios del campo de las CC y algunas cuestiones relativas a su didáctica. Por esta razón, se optó por analizar esta experiencia en particular.

A continuación, en la segunda sección, se describe el marco metodológico utilizado para el estudio de la experiencia del TD. En la tercera, se presenta el instrumento diseñado para el análisis de las secuencias didácticas desarrolladas por los/as cursantes y se muestra un ejemplo concreto de su implementación. En la cuarta se da cuenta de algunas reflexiones que podrían constituirse en posibles aportes para la construcción de la didáctica de las CC y en la quinta se presentan algunas potencialidades y desafíos resultantes de la puesta en práctica del instrumento.

2. Marco metodológico

Para el análisis de esta experiencia se llevó a cabo un estudio de tipo **exploratorio**, desde un **enfoque integrado** (Hernández, Fernández y Baptista, 2004) que combina tanto la **metodología cuantitativa** como **cualitativa**. Específicamente, fue realizado desde el modelo de *enfoque dominante* debido a que prevaleció la perspectiva de la metodología cualitativa, pero se complementa con algunas técnicas cuantitativas. La justificación de la elección del enfoque es por su potencial para enriquecer tanto la recolección de datos como su análisis (Grinnell, 1997 en Hernández et al., 2004).

En relación a las **técnicas de recolección e instrumentos para el análisis de datos**, la diversidad de procedimientos abarca desde observación participante[1], registros de las clases, encuestas y el estudio de producciones -secuencias didácticas e informes de coevaluación- realizadas por los cursantes de la Especialización durante el cursado del TD. El análisis de la información recolectada por estas técnicas puede encontrarse en Menchón, Elizalde, Mauco y Leonardi (2021). En lo que respecta a las secuencias didácticas de los/as cursantes se destaca que, del total de las 22 propuestas presentadas, se consideraron las 10 que abordaron contenidos básicos de CC, como los conceptos de algoritmo, programa, secuencia y repetición simple. Por último, se destaca la utilización del software ATLAS.ti[2], desarrollado para el análisis de datos en investigaciones con metodologías cualitativas.

3. Qué, para quiénes, por qué/para qué, cómo y a través de qué enseñar: las cinco dimensiones del instrumento para analizar secuencias didácticas

Para poder llevar a cabo el análisis de las secuencias didácticas presentadas por las/os cursantes del TD, se construyó un instrumento particular a fin de poder contemplar y abarcar todas aquellas cuestiones relativas a la especificidad de la didáctica de las CC. El mismo consta de 5 dimensiones -referentes a aspectos básicos de la planificación didáctica- y para cada una de las cuales se definieron distintos interrogantes orientadores para el estudio de las mismas, como se puede visualizar en la Tabla 1.

Dimensión	Interrogantes
D1: Dimensión referente al para quiénes enseñar (estudiantes destinatarios)	1.1. ¿Para qué año y ciclo del nivel primario estaba destinada la secuencia didáctica? 1.2. ¿Qué características pueden destacar del grupo de estudiantes destinatarios de la secuencia?

[1] En la observación participante "el observador es parte de la escena del aula, observando el desarrollo de una clase donde se presenta una determinada propuesta pedagógica con TIC, así como las situaciones que se presentan a partir de ella, asumiendo un rol de participación e interacción" (Cappellacci, Ros y González, 2015, p. 39).

[2] https://atlasti.com/es/

Dimensión	Interrogantes
D2: Dimensión respecto al qué enseñar (contenido)	2.1. ¿Cuáles fueron los conceptos teóricos seleccionados? 2.2. ¿Son correctos los saberes previos presentados como necesarios para el desarrollo de la secuencia didáctica? 2.3. ¿De qué manera se jerarquizaron y/o relacionaron los conceptos y por qué? 2.4. ¿Los conceptos teóricos abordados y las actividades propuestas fueron acordes con respecto al grupo etario para el cual está diseñada la secuencia? 2.5. ¿Se presentaron definiciones de los conceptos trabajados? En caso afirmativo ¿Cuáles y de qué manera se plantearon? 2.6. ¿De qué modo se organizaron los contenidos en la secuencia didáctica? (transversal/integrado, como disciplina propia o extracurricular) En el caso de ser transversal/integrado ¿cómo se planteó la articulación y con qué materias?
D3: Dimensión relativa al por qué y para qué enseñar (finalidad y objetivos de aprendizaje)	3.1. ¿Cómo se planteó la coherencia entre la selección de contenidos, finalidad/es de la secuencia didáctica y objetivos de aprendizaje? 3.2. ¿De qué manera la secuencia didáctica intentó favorecer el desarrollo de las distintas habilidades de la programación (programar, comprender, modificar y depurar programas) y por tanto el pensamiento computacional?
D4: Dimensión en relación al cómo enseñar (metodología pedagógico-didáctica utilizada, diseño de actividades y evaluación)	4.1. ¿Cuál o cuáles metodología/s pedagógico-didáctica/s se seleccionaron para el desarrollo de la secuencia didáctica? ¿Cuál fue la fundamentación de su elección? 4.2. ¿Qué aspectos dan cuenta de la coherencia -o no- entre la metodología pedagógico-didáctica elegida y la metodología utilizada en las actividades? 4.3. ¿Adaptaron propuestas existentes o desarrollaron una nueva? Si adaptaron una propuesta existente, ¿cuál y por qué? 4.4. ¿Las actividades propuestas promueven el uso del concepto que quiere enseñarse o pueden resolverse sin usarlo? 4.5. ¿Qué tipo de actividades diseñaron? ¿Qué cantidad de cada tipo? ¿Por qué se optó por esa elección?

Dimensión	Interrogantes
D4: Dimensión en relación al cómo enseñar (metodología pedagógico-didáctica utilizada, diseño de actividades y evaluación)	4.6. ¿Qué aspectos pueden destacarse en relación a la claridad de los enunciados de las consignas para las actividades propuestas? 4.7. La modalidad de evaluación planteada, ¿Qué características tiene? ¿Es coherente con la metodología pedagógico-didáctica propuesta y contempla aspectos propios de la disciplina de las CC? 4.8. ¿Qué potencialidades y limitaciones pueden reconocerse en relación a la viabilidad de la puesta en práctica de la secuencia didáctica?
D5: Dimensión acerca del a través de qué enseñar (recursos educativos y medios didácticos)	5.1. ¿Qué recursos educativos y medios didácticos se integran para el desarrollo de actividades? ¿Por qué y para qué los eligieron? 5.2. ¿Qué otros recursos y medios podrían integrarse para mejorar la mediación en el proceso de enseñanza y de aprendizaje?

Tabla 1: Instrumento para analizar secuencias didácticas de CC. Fuente: Elaboración propia.

3.1. Ejemplo de aplicación del instrumento

A partir de este instrumento, se analizó cada una de las secuencias didácticas y se elaboró una tabla con un registro de las respuestas a las preguntas planteadas, dando cuenta así no solo de datos cuantitativos (ejemplo: cantidad de actividades conectadas, desconectadas o de ambos tipos) sino también de carácter cualitativo (tales como las modalidades de evaluación por las que se optaron en cada caso). A fines de presentar una ejemplificación de la utilización del instrumento, se explicita a continuación el análisis de una de las secuencias que abordaron el concepto de repetición simple para primer ciclo del nivel primario.

D.	Ítem	Secuencia didáctica
D1	1.1.	El grupo destinatario de la secuencia pertenece al 1er año del 1er ciclo del nivel primario.
D1	1.2.	El grupo se caracteriza por ser de 6 estudiantes, lo que posibilita un seguimiento personalizado. Tienen conocimientos previos del concepto de algoritmo, noción de secuencia de instrucciones y el reconocimiento de símbolos utilizados dentro de nuestra sociedad como códigos comunes.
D2	2.1.	El concepto teórico seleccionado fue Repetición simple.
D2	2.2.	Los saberes previos que se precisaron en la secuencia didáctica (algoritmo con instrucciones secuenciales y lenguaje simbólico) son los necesarios para el desarrollo de la misma.
D2	2.3.	Hubo una adecuada selección y jerarquización de los contenidos a abordar de acuerdo al grupo etario. A partir de la recuperación de los saberes previos de los/as estudiantes, se plantea la enseñanza de estructura de repetición. Se reconoce el trabajo en las actividades de patrón, sin embargo, no se presenta una referencia explícita del mismo.

D.	Ítem	Secuencia didáctica
D2	2.4.	Los conceptos teóricos abordados y las actividades propuestas fueron acordes con respecto al grupo etario para el cual está diseñada la secuencia. Teniendo en cuenta la franja etaria, resulta fundamental que los/as estudiantes construyan todo concepto propuesto partiendo, en primera instancia, desde lo concreto donde con el propio físico pueden explorar distancias, movimientos y proponer formas de resolución a distintos planteos con las herramientas que naturalmente se manejan (espacio físico). Como segunda instancia y nivel de profundidad para el abordaje de los contenidos se propone volcar en el plano las resoluciones vividas y experimentadas, logrando un primer grado de abstracción para más adelante llevarlo a una tercera instancia (próximas secuencias) donde construyan y reconozcan lo aprendido a través de actividades "enchufadas".
D2	2.5.	Se presentan definiciones de todos los contenidos abordados. Luego del desarrollo de las actividades, aparecen como una conclusión (construcción realizada colectivamente) y anotada en un portfolio.
D2	2.6.	El modo en que se organizaron los contenidos fue como materia propia. Se pensó con una carga horaria semanal de 2hs (de 80 minutos una vez por semana). A la vez se propone trabajar de manera articulada con otras disciplinas en paralelo tales como: Educación Física, Artística y Música.
D3	3.1.	Existe coherencia entre la selección de contenidos, finalidad/es de la secuencia didáctica y objetivos de aprendizaje. Todos los contenidos se ven reflejados no solo en las finalidades de la secuencia sino también en objetivos generales que fueron abordados en cada actividad.
D3	3.2.	La secuencia didáctica intentó favorecer el desarrollo de las habilidades de programar y comprender. La manera en la cual se desarrollaron fue: en las primeras, una vez introducido el concepto de Repetición, se trabajó sobre la habilidad de programar en papel (usando la noción de algoritmo), incluso definiendo un lenguaje de símbolo. Algunos de estos algoritmos eran intercambiados para que los/as estudiantes los ejecuten, con lo cual desarrollaron la habilidad de interpretar. Finalmente, las últimas actividades desarrollaron la comprensión ya que se daban algoritmos incompletos o donde los/as estudiantes debían expresar qué resultado se obtenía.
D4	4.1.	La metodología pedagógico-didáctica seleccionada para el desarrollo de la secuencia didáctica fue el aprendizaje basado en problemas. La fundamentación de su elección fue debido a que, a partir de la resolución de problemas, se posibilitan espacios de intercambio de opiniones entre los/as estudiantes acerca de los diferentes caminos resueltos. En esos debates, se debería concebir al error como parte del proceso de aprendizaje, procediendo a la interpretación de las resoluciones erróneas y proponiendo desafíos para lograr mayor abstracción a la hora de desarrollar algoritmos con simbologías acordadas.

D.	Ítem	Secuencia didáctica
D4	4.2.	El primer aspecto que da cuenta de la coherencia entre la metodología pedagógico-didáctica elegida y la utilizada en las actividades es el planteo de una serie de problemas que las/os estudiantes deben resolver tales como: el descubrir cuáles son los patrones de repetición en diversas fotos de la vida cotidiana que se les presentan. El segundo aspecto es el abordaje de lo concreto/espacial a lo abstracto/plano, correlativo al tipo de actividades que van desde unplugged a plugged (aunque estas últimas no se plantean en la secuencia sino como alternativa para una próxima). Los/as estudiantes (6-8 años) se encuentran entre la etapa pre-operacional (desde los 2 años hasta los 7 años aproximadamente) y la etapa de operaciones concretas (de 7 a 11 años aproximadamente) por lo que su capacidad de abstracción se encuentra en desarrollo.
D4	4.3.	Las actividades desarrolladas en la secuencia fueron propuestas nuevas. La única excepción fue una actividad alternativa, pensada para una articulación con el área de música, que fue recuperada del Manual para Primer Ciclo (Fundación Sadosky, 2018).
D4	4.4.	Las actividades propuestas promueven el uso del concepto de repetición al haberse trabajado alrededor de muchos ejemplos con repeticiones. El uso de ese concepto aparece como alternativa para la correcta resolución de los problemas planteados.
D4	4.5.	Los tipos de actividades que se diseñaron fueron individuales (3) y grupales (3). Las mismas fueron desde la elaboración de conclusiones en grupo, espacios de invención de algoritmos en forma individual y decodificación de otros de manera grupal. Se optó por actividades desconectadas (unplugged) (6) y combinando ambos tipos no solo por las características del grupo etario sino también para favorecer el aprendizaje colaborativo.
D4	4.6.	Los aspectos que pueden destacarse en relación a la claridad de los enunciados de las consignas para las actividades propuestas son: su explicitación completa en todos los casos, la utilización de un lenguaje adecuado para el grupo etario y la presentación y explicación de las mismas previamente a su desarrollo.
D4	4.7.	Las modalidades de evaluación planteadas son coherentes con la metodología pedagógico-didáctica propuesta. Las mismas fueron de tres tipos: heteroevaluación, autoevaluación y coevaluación. La primera se presenta en forma reflexiva después de la resolución de ejercicios, la segunda a partir del propio análisis del/a estudiante acerca los saberes aprendidos (autoevaluación del/a estudiante) y la evaluación de la propia propuesta de enseñanza (autoevaluación del docente) y la tercera con la puesta en común y elaboración de conclusiones. Se presenta la distinción entre criterios de evaluación y de acreditación presentados. Se contemplan aspectos propios de la disciplina de las CC, a partir de la recuperación de la concepción de Pensamiento Computacional de Brennan & Resnick (2012) del cual se parte para la definición de los criterios y la profundidad en la evaluación de la propia práctica docente.

D.	Ítem	Secuencia didáctica
D4	4.8.	Las potencialidades que pueden reconocerse sobre la viabilidad de la puesta en práctica de la secuencia didáctica tienen que ver con: la accesibilidad de los materiales y recursos elegidos, el tipo de actividades propuestas (desconectadas) y los propósitos que busca. La limitación encontrada es que la secuencia es adecuada para su desarrollo en un grupo pequeño, sin embargo, puede resultar difícil ponerla en práctica con un grupo numeroso, por el seguimiento personalizado que implican las actividades.
D5	5.1.	Los recursos educativos y medios didácticos para el desarrollo de actividades fueron: portfolio (afiche), juego Simón, fotos, fotocopias de las actividades, tiza y pizarrón. Su elección se justifica por el tipo de actividades que se desarrollaron (unplugged) lo cual no requirió del uso de más recursos tecnológicos ni entornos de programación. Para una actividad complementaria desde la materia de Música, se propone la recuperación de un desafío presente en el Manual de Primer ciclo (Fundación Sadosky, 2018, p. 140), sumándose así el manual como otro medio didáctico.
D5	5.2.	Ninguno. Los recursos y medios presentados fueron adecuados para el desarrollo de la secuencia didáctica.

Tabla 2: Ejemplo aplicación instrumento para analizar secuencias didácticas de CC. Fuente: Elaboración propia.

Como resultado del análisis con el software de ATLAS.ti, se presenta a continuación una nube con los conceptos predominantes en la secuencia didáctica ejemplificada anteriormente.

Figura 1: Nube conceptual secuencia N° 1. Fuente: Elaboración propia.

4. Algunas reflexiones para la construcción de una didáctica para las CC a partir del análisis del TD

Se realizó el estudio exploratorio presentado en la Sección 2 (Marco metodológico) sobre la experiencia vivenciada en el TD. Para el mismo, se consideraron las 10 secuencias didácticas que abordaron contenidos básicos de CC. A

partir de este análisis, se pudieron observar algunos aspectos generales a tener en cuenta para la construcción de una didáctica de las CC.

4.1. Resultados del análisis acerca del TD

En relación al ***para quiénes enseñar***, el 80 % de las secuencias didácticas fueron elaboradas para el primer ciclo y el 20 % para segundo (Figura 2 y la Figura 3).

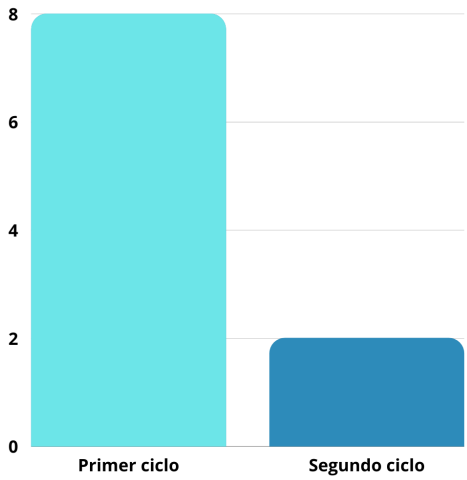

Figura 2: Cantidad de secuencias para cada ciclo

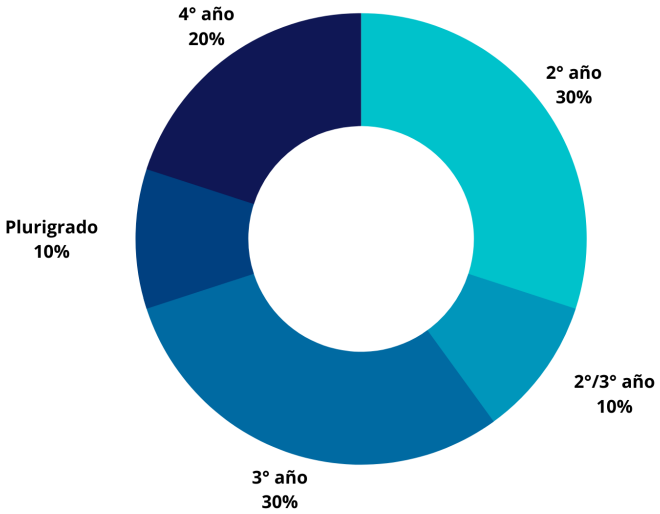

Figura 3: Porcentaje de secuencias por año

Respecto a la dimensión del ***qué enseñar***, los contenidos elegidos para ser abordados en primer ciclo y primer año del segundo ciclo fueron: algoritmo y lenguaje simbólico (3 secuencias), programa (4 secuencias) y repetición simple (3 secuencias). En la mayoría de las secuencias, los conceptos se construyeron de manera grupal durante el desarrollo de la actividad o cierre y se plasmaron en algún medio gráfico (por ejemplo, afiche para que quede en el

aula, carpeta/cuaderno de cada estudiante).

Acerca del ***por qué y para qué enseñar*** se reconoció -en el apartado de la fundamentación de las secuencias didácticas- la recuperación de los argumentos sobre la importancia de la enseñanza de los contenidos de las CC expresados en los Núcleos de Aprendizaje Prioritarios de Educación digital, Programación y Robótica (CFE, 2018). Asimismo, se logró identificar como una finalidad común de las secuencias el hecho de favorecer el desarrollo del pensamiento computacional, aunque no en todos los casos se correspondió luego con las propuestas de las actividades.

En vinculación al ***cómo enseñar***, las 10 secuencias didácticas presentaron la elección de distintas metodologías -en base a la clasificación presentada por Torp y Sage (2004), tales como: Aprendizaje Basado en Problemas, Aprendizaje centrado en un problema, Aprendizaje Basado en Proyectos, Aprendizaje por indagación, Aprendizaje reflexivo, Gamificación y Aprendizaje Basado en Juegos. En algunos casos, se optó por la elección de una única metodología y en 3 secuencias se combinaron dos de ellas: 1) Gamificación y Aprendizaje Basado en Problemas, 2) Gamificación y el Aprendizaje por Indagación y 3) Aprendizaje por Indagación y Aprendizaje Basado en Juegos.

En referencia a las actividades, se puede reconocer la predominancia de las desconectadas (7) para el abordaje de los contenidos en el nivel primario. Las 3 restantes combinaron actividades de este tipo con conectadas. En relación a la evaluación, en general, fue un componente poco desarrollado en las secuencias.

Acerca del a ***través de qué*** enseñar, se observó una predominancia de actividades desconectadas, por lo que se planteó en pocos casos la integración de entornos de programación. De las 3 secuencias que plantearon actividades conectadas, en una se eligió el Code.org y en las otras Pilas Bloques Junior, justificando su elección, por la simplicidad del uso y la disponibilidad de acceso a internet. En relación a la integración de medios didácticos como manuales/material didáctico con actividades prediseñadas, algunas de las secuencias que optaron por usarlas lo hicieron adaptándolas.

4.2. Reflexiones a partir de los resultados

Acerca de la dimensión del ***para quiénes***, se pudo destacar que los contenidos fueron adecuados para los grupos etarios debido a que son los primeros conceptos y los estudiantes poseen las habilidades cognitivas para su abordaje, ya que no implican mayores niveles de abstracción. La importancia de tener en cuenta esta dimensión es debido a que las características e intereses del grupo destinatario definirán la toma de decisiones del resto de los componentes de la secuencia didáctica.

Acerca del ***qué enseñar***, se pudo reconocer que:

- Es necesario, al momento de realizar el recorte del contenido a enseñar, contemplar los saberes previos de los/as estudiantes del grupo destinatario para poder partir desde ellos y así diseñar actividades que favorezcan su articulación con los nuevos contenidos a ser enseñados. Más allá que los NAP (CFE, 2018) proponen un conjunto de los mismos para el nivel, sería importante tener en cuenta la trayectoria escolar del grupo destinatario para poder avanzar progresivamente o, si se requiere, repasar o introducir los contenidos básicos para luego proceder con los nuevos.
- Por la especificidad de los contenidos, la mejor manera de abordarlos es como objeto de estudio propio. En este sentido, se considera que es necesario que los/as docentes tengan una formación sólida en los contenidos fundamentales de la disciplina. En esta experiencia, habiendo pasado casi dos años de formación, se observó que todas las secuencias abordaban los contenidos de CC como objeto de estudio, a diferencia con otras secuencias planteadas en los comienzos de la Especialización que si bien proponían abordar un determinado concepto de CC quedaba diluido en una problemática de otra área, usado como herramienta

para resolverla (Leonardi, Mauco, Felice y Menchón, 2021). Estas observaciones coinciden con el posicionamiento expresado por Bonello y Schapachnik (2020) acerca de la importancia del reconocimiento de las CC como una asignatura con su propio espacio curricular.

- La jerarquización de los contenidos es fundamental para el planteo de las actividades. Para una mejor comprensión, es necesario tener en cuenta la organización de los contenidos de manera tal que faciliten posteriormente la relación entre los conceptos. Nuevamente, se destaca la importancia de la formación disciplinar del docente para poder reconocer qué contenidos pueden desprenderse de otros y/o no pueden ser abordados sin conocer otros previamente.

Respecto al ***por qué y para qué enseñar***:

- Es necesario que el docente reconozca la importancia de la enseñanza de los contenidos de CC en la escuela. Más allá del planteo de los NAP (CFE, 2018) es significativo que el docente comprenda el fundamento desde un punto de vista crítico y reflexivo, no por mera imposición normativa o reduciendo el argumento al despertar vocaciones científicas para cubrir la demanda del mercado laboral sino pensar su enseñanza como un derecho, donde mediante la alfabetización en estos temas se colaborará para la formación de una ciudadanía más crítica, comprendiendo más ampliamente el mundo digital. Este posicionamiento va en la misma línea de "Quienes argumentan que este cambio curricular es necesario, sostienen que la enseñanza de las CC, específicamente a través de la programación, contribuiría a cerrar la brecha digital, mejorar la fluidez tecnológica entre la ciudadanía" (Benitez Larghi, 2014 citado en Martinez y Gomez, 2018).
- Debería existir coherencia entre el por qué y para qué enseñar y los contenidos o habilidades que se pretenden desarrollar. Muchas veces aparece como una finalidad de la secuencia el hecho de "favorecer el desarrollo del pensamiento computacional" (Secuencia didáctica analizada, 2020) y sin embargo luego las actividades propuestas no potencian las distintas habilidades que forman parte del mismo (descomposición en subproblemas, generalización, entre otras), limitándose a propuestas en las que el objetivo es diseñar un programa para resolver un problema dado. En otras actividades, se encontraron puestas en común de las soluciones desarrolladas por los estudiantes para promover un trabajo colaborativo. En estos casos, se podrían aprovechar estas instancias de discusión para favorecer las distintas habilidades del pensamiento computacional como abstracción, generalización, quizás sin ser totalmente conscientes.

En vinculación al ***cómo enseñar***, algunos aportes respecto a esta dimensión:

- Es muy importante la elección de una adecuada metodología que posibilite el desarrollo del pensamiento computacional en los/as estudiantes. La elección de la metodología impacta directamente en la formación, debido a que hay algunas de ellas (como el Aprendizaje Basado en Problemas) donde ya desde la forma de enseñar se está favoreciendo la puesta en práctica de esos contenidos y el desarrollo de algunas de las habilidades propias del pensamiento computacional. En base a los fundamentos de Edwars (1993) el contenido no es independiente de la forma en la cual es presentado. Por ejemplo, difícilmente se pueda enseñar al/a estudiante a desarrollar la habilidad de descomponer en subproblemas un problema más complejo con una mera clase tradicional, sin propuesta de actividades prácticas que involucren su participación activa.
- Se sugiere el desarrollo de actividades desconectadas en los primeros años del nivel primario debido a las características propias del desarrollo cognitivo en esa edad. Como aún la capacidad de abstracción se está desarrollando, es necesario comenzar a trabajar desde lo concreto a lo abstracto. Las secuencias que

plantearon actividades conectadas, lo hicieron progresivamente y en articulación con previas actividades desconectadas.

- Es fundamental que el docente se posicione concibiendo al error como una oportunidad para el aprendizaje y sin connotación punitiva para evitar la desmotivación y afectar negativamente la autoestima y confianza del/a estudiante (Giraldez, 2018), especialmente porque durante el desarrollo de las actividades es común que aparezca el error en las distintas resoluciones a los problemas o desafíos planteados.
- Respecto a la evaluación, es importante la coherencia entre la modalidad de evaluación elegida y la utilizada para el desarrollo de las actividades. Es decir, si se opta por una metodología basada en aprendizaje colaborativo, sería incongruente luego presentar como evaluación un test individual o una evaluación que no contemple las habilidades vinculadas al trabajo colectivo y al pensamiento computacional. En ese sentido, se propone por ejemplo la combinación de múltiples modalidades de evaluación (heteroevaluación, autoevaluación y co-evaluación), a partir de la concepción de la evaluación como una oportunidad para el aprendizaje (Anijovich y Cappelletti, 2017). Asimismo, para poder identificar esas cuestiones, se propone como instrumento de evaluación a la rúbrica, diseñadas a partir de la concepción del pensamiento computacional como la planteada por Brennan y Resnick (2012) y por tanto debería contemplar: a) los conceptos computacionales, b) prácticas computacionales y c) perspectivas computacionales (expresar, preguntar y conectar). De esa forma, podrían evaluarse las competencias propias de la disciplina.

Por último, sobre la dimensión de ***a través de qué enseñar*** se pudo reflexionar:

- Al momento de elegir el entorno a integrar, se recomienda tener en cuenta: a) que el grupo esté alfabetizado -y/o proceder a la lectura tanto de la consigna como de las instrucciones y comandos-, b) probar y analizar las limitaciones y potencialidades del entorno a fin de poder optar por el que sea lo más versátil para el abordaje de los objetivos propuestos en la secuencia didáctica -reconociendo que algunos no permiten la edición de sus templates o la creación de desafíos personalizados-, c) elegir no solo aquel al que pueda garantizarse su acceso sino también considerando los que funcionen en los dispositivos que los/as estudiantes cotidianamente utilizan -por ejemplo, si no se dispone de computadoras hay juegos como Lightbot que sirven para aprender a programar y se puede acceder desde el celular-, d) más allá que algunos/as estudiantes tienen acceso, siempre ofrecer explicaciones y espacios de práctica para el manejo de esas herramientas porque no todos/as cuentan con los mismos recursos.
- Cuando se eligen actividades prediseñadas, es importante que el docente realice un análisis para determinar si se pueden usar tal como están o es necesario realizar adaptaciones de acuerdo a las características, necesidades e intereses del grupo de estudiantes, ya que el material didáctico disponible es relativamente nuevo y hay poca documentación sobre la experiencia de su uso en las aulas.

5. Conclusiones y futuros desafíos

A partir de la implementación del instrumento presentado, se puede reconocer su potencial para el abordaje analítico de una gran cantidad de propuestas de enseñanza, contemplando las múltiples dimensiones que componen los procesos de enseñanza y de aprendizaje (qué, para quiénes, por qué y para qué, cómo y a través de qué enseñar), posibilitando la recuperación de aspectos tanto cualitativos como cuantitativos. Asimismo, los interrogantes de cada dimensión –al ser planteados en términos generales– pueden utilizarse para el estudio de secuencias didácticas de otros

niveles del sistema educativo e inclusive incorporar otras cuestiones relativas a su encuadre en el diseño curricular –por ejemplo, en el caso del Nivel Inicial donde las CC parecen dentro del Área de *Enseñanza de Educación Digital, Programación y Robótica*– (DGCyE, 2019).

Al momento de utilizarlo, la única dificultad que surgió fue que solo algunas secuencias didácticas no contaban con toda la información necesaria para el análisis de cada una de las dimensiones propuestas. Sin embargo, más que una limitación propia del instrumento, la razón de este inconveniente puede haber resultado de las dificultades de los/as cursantes de la Especialización para diseñar de la secuencia, contemplando todos los aspectos básicos de la planificación de una propuesta de enseñanza –por ejemplo, la evaluación–. Se espera que la difusión de este instrumento sirva para que pueda implementarse en otros análisis y así seguir aportando al reconocimiento de aspectos significativos para la construcción colectiva y colaborativa de la didáctica de la CC.

Bibliografía

Anijovich, R. y Cappelletti, G. (2017) La evaluación como oportunidad. Paidós.

Bonello, M. B. y Schapachnik, F. (2020) "Diez preguntas frecuentes (y urgentes) sobre pensamiento computacional" En Virtualidad, Educación y Ciencia. Año 11 (20).

Brennan, K. y Resnick, M. (2012). "Nuevos marcos de referencia para estudiar y evaluar el desarrollo del pensamiento computacional". Encuentro anual de la "American Educational Research Association", AERA 2012, Canada.

Cappellacci, I., Ros, C. y González, D. (2015) Estrategias de Producción y Análisis de Información en la Investigación Educativa. Argentina, Instituto Nacional de Formación Docente, Dirección Nacional de Formación e Investigación, Área de Investigación Educativa.

Consejo Federal de Educación (2018). Núcleos de Aprendizajes Prioritarios para Educación Digital, Programación y Robótica. Resolución 343/18. Argentina.

Cuny, J., Snyder, L. y Wing, J. M. (2010) "Demystifying Computational Thinking for Non-Computer Scientists". www.cs.cmu.edu/~CompThink/resources/TheLinkWing.pdf.

Dirección General de Cultura y Educación de la Provincia de Buenos Aires (2018). Resolución N° 929/18. Especialización Docente de Nivel Superior en Didáctica de las Ciencias de la Computación para la Educación Primaria. La Plata: Dirección General de Cultura y Educación de la Provincia de Buenos Aires.

Dirección General de Cultura y Educación de la Provincia de Buenos Aires (2019) Diseño Curricular para la Educación Inicial. Área de Educación Digital, Programación y Robótica. La Plata.

Echeveste, M. E., y Martinez, M. C. (2020) Principales tendencias de los programas que enseñan Pensamiento Computacional. Material de circulación interna. Universidad Nacional de Córdoba.

Edwars, V. (1993) "La relación de los sujetos con el conocimiento" Revista Colombiana de Educación (27) DOI: https://doi.org/10.17227/01203916.5304.

Fundación Sadosky (2013) CC 2016. Una propuesta para refundar la enseñanza de la computación en las escuelas argentinas. Buenos Aires.

Giraldez, A. (2018) El error como oportunidad de aprendizaje. ¿Y si dejamos de castigar los errores? Educación 3.0. www.educaciontrespuntocero.com/noticias/dejamos-castigar-los-errores/.

Hernández, S. R., Fernández, C. C. y Baptista, L. P. (2004) Metodología de la investigación. México, D.F: McGraw Hill.

Leonardi, C., Mauco, V., Felice, L. y Menchón, N. (2021) Pensando la enseñanza de las Ciencias de la Computación en el nivel primario: una experiencia de la Especialización Superior en la ciudad de Tandil. Revista Espacios en Blanco 2 (31), 227–242. https://doi.org/https://doi.org/10.37177/UNICEN/EB31-298.

Martinez, M. C. y Gomez, M. J. (2018) Programar computadoras en educación infantil. EDUTEC. Revista Electrónica de Tecnología Educativa (65).

Menchón, N., Elizalde, A., Mauco, V. y Leonardi, C. (2021) Formación docente para la enseñanza de las Ciencias de la Computación en el nivel primario: la experiencia del Taller de Práctica en el marco de la pandemia. Aceptado para su presentación en SAEI 2021. 50° JAIIO 18-29 octubre 2021.

Torp, L. y Sage, S. (2004) El Aprendizaje Basado En Problemas. Desde el jardín de infantes hasta la escuela secundaria. United States: Amorrortu.

El desarrollo de habilidades de resolución de problemas como un saber necesario en el camino del aprendizaje de la programación

Fernando Raúl Alfredo Bordignon
fernando.bordignon@unipe.edu.ar
Universidad Pedagógica Nacional (UNIPE)

Alejandro Adrián Iglesias
alejandro.iglesias@unipe.edu.ar
Universidad Pedagógica Nacional (UNIPE)

Resumen

La incorporación de la enseñanza de las Ciencias de la Computación en la educación básica obligatoria en la República Argentina es un proceso que se desarrolla día a día y que representa un importante desafío educativo: la construcción de una didáctica asociada a tales niveles. En particular, a través de un relevamiento bibliográfico, hemos identificado problemas educativos recurrentes que suceden en los primeros cursos de enseñanza de la programación. La falta de capacidades en torno a la resolución de problemas, se ha manifestado como uno de los principales obstáculos para lograr un buen desempeño de los estudiantes. El objetivo de este trabajo es, en una primera parte, presentar evidencias en torno a la situación planteada con la finalidad de visibilizar y poner en discusión el tema, y en una segunda parte, plantear la necesidad de desarrollar las habilidades del pensamiento computacional como un saber inicial en el camino del aprendizaje de la programación.

Palabras clave: Resolución de problemas, aprendizaje de la programación, pensamiento computacional.

1. Introducción

Diferentes países del mundo, consideran importante y estratégico el desarrollo de conocimientos que permitan a sus ciudadanos entender, crear y resolver problemas[1] utilizando tecnologías digitales (Grover, 2018). Esto se puede ver reflejado en cómo diferentes conceptos, contenidos y habilidades relacionadas con la computación empiezan a tener lugar dentro de los planes de estudio en los sistemas educativos de diversos países[2]. El desarrollo de estos saberes se configura como parte de las nuevas alfabetizaciones que permiten a los jóvenes integrarse de una mejor forma a la sociedad, entendiendo cómo funciona el mundo digital y colaborando con sus oportunidades laborales.

[1]En este trabajo vamos a utilizar la definición de resolución de problemas propuesta por Mayer (1990), quien indica que consiste en un proceso cognitivo dirigido a transformar una situación dada en una situación meta u objetivo cuando no se dispone de un método obvio de solución.

[2]Países que han incluido la programación en el currículo escolar: Austria, Bulgaria, República Checa, Dinamarca, Estonia, Francia, Hungría, Irlanda, Israel, Lituania, Malta, España, Polonia, Portugal, Eslovaquia y Reino Unido (Uzunboylu et al., 2017).

Si bien en un principio la enseñanza de temas relacionados con las tecnologías digitales se asoció a trabajar con contenidos de ofimática y con otros cercanos a la alfabetización en medios, hoy se ha decidido cambiar el enfoque y centrarlo en la computación. Aunque puede parecer un concepto amplio, en esencia, la computación se refiere a las tareas de procesamiento automático de información que involucran computadoras, ya sea usándolas o creándolas. En este sentido, esta área incluye temas como: el diseño y la construcción de sistemas de hardware y software para múltiples propósitos; el procesamiento de datos, la estructuración y la gestión de diversos tipos de información; la realización de estudios científicos utilizando computadoras y el comportamiento inteligente de los sistemas informáticos entre los principales (ACM/AIS/IEEE, 2005).

Este cambio en las políticas curriculares presenta una serie de desafíos al sistema educativo. Si bien existe un camino recorrido en el nivel superior, con experiencias valiosas en relación a la enseñanza y aprendizaje de la computación, no es posible transferir estos saberes de manera directa a los niveles educativos que conforman la enseñanza obligatoria; se trata en este caso de otras edades, otros desarrollos cognitivos, otras subjetividades y diferentes objetivos de enseñanza. Por eso entendemos que la computación necesita de una nueva didáctica asociada a tales niveles y ese es el desafío principal que nos convoca.

La programación, como uno de los temas que resultan transversales a toda la computación, es uno de los elegidos para ser incorporado en las escuelas argentinas. En este trabajo presentamos un problema educativo, relacionado con las capacidades de resolución de problemas de los estudiantes, que resulta recurrente dentro de la enseñanza de la programación en un primer curso. Este artículo es parte de los primeros resultados de una investigación, llevada a cabo en la Universidad Pedagógica Nacional, denominada "Sistematización de problemas y desafíos asociados a la enseñanza y aprendizaje del pensamiento computacional y la programación en el primer ciclo de la escuela secundaria"; cuyo objetivo principal es presentar un estado del arte acerca de los problemas más comunes de enseñanza y de aprendizaje que se producen en un primer curso de programación, como así también orientaciones docentes que permitan mitigarlos y/o superarlos.

Luego de esta introducción, presentamos evidencias surgidas de una serie de trabajos de investigación que hablan acerca del bajo desarrollo de habilidades para resolución de problemas en los estudiantes en un primer curso de iniciación a la programación; entendiendo a esta situación como un factor asociado al abandono y bajo rendimiento de los estudiantes. Luego, como una forma de visibilizar esta situación y comenzar el camino hacia una posible solución, proponemos el desarrollo del pensamiento computacional como un elemento de inicio en la enseñanza y aprendizaje de la programación.

2. La capacidad de resolución de problemas y la enseñanza de la programación

El aprendizaje de la programación es un tema que representa cierta dificultad para algunos estudiantes (Gomes y Mendes, 2007; Insuasti, 2016)[3] y que se percibe como una materia compleja que requiere de un esfuerzo importante para llevarla adelante. Más allá de las apreciaciones, en los hechos, se registra que los cursos iniciales tienen altas tasas de deserción y desaprobación (Baldwin y Kuljis, 2001; Luxton-Reilly, 2016). Aissa y otros (2020) afirman que

[3]Aclaración: la mayoría de los trabajos de investigación presentados están relacionados con un primer curso de programación en el nivel educativo superior ya que, debido al desarrollo del tema, hay escasa evidencia que se refiera a la educación básica.

el desafío principal en este aprendizaje, está en relación a que los estudiantes deben adquirir al mismo tiempo un conjunto de habilidades que van más allá de solo aprender a escribir código. Sin embargo, los autores mencionan que los enfoques tradicionales de enseñanza de la programación hacen más hincapié en la sintaxis y la semántica del lenguaje que en las estrategias de resolución de problemas. En este sentido, "*... el aprendizaje demanda habilidades cognitivas complejas tales como la planificación, razonamiento y resolución de problemas*" (Baldwin y Kuljis, 2001).

En un trabajo de revisión de literatura científica relacionada con el aprendizaje de la programación, Susanti y otros identifican que el fracaso en los cursos iniciales de programación tiene que ver con la dificultad en la comprensión de: conceptos básicos sobre algoritmos, habilidades de resolución de problemas, identificación de problemas, desarrollo de algoritmos y escritura de programas en lenguajes de programación (Susanti y otros, 2021). En otro relevamiento bibliográfico sobre problemas relacionados, Insuasti (2016) indica que ciertas habilidades cognitivas son importantes cuando se aprenden los fundamentos de la programación, específicamente se refiere a: capacidad de abstracción, aptitud lógico-matemática desarrollada y la facilidad para la resolución de problemas de orden algorítmico. De forma más precisa, en relación a los requisitos de aprendizaje, indica que "*(l)a falta de experiencia previa en programación de computadoras no parece ser un problema; sin embargo, sí es un problema el bajo desarrollo de habilidades para la resolución de problemas.*" (Insuasti, 2016). La situación descrita constituye un problema educativo, ya que el aprender a programar requiere tener desarrolladas habilidades de pensamiento lógico y de resolución de problemas.

Diversos trabajos sostienen que los estudiantes que desaprueban o tienen bajo desempeño en los primeros cursos de programación en el nivel superior, no tienen suficientemente desarrolladas capacidades para la resolución de problemas (Lister y otros; 2004, Chin Soon, 2020; Grover y otros, 2016; Qian y Lehman, 2016; Babori y otros, 2016; Luza, 2017) . También se hace referencia a la falta de consolidación de los conocimientos asociados al razonamiento lógico (Savage y Piwek, 2019; Bosse y Gerosa, 2017) y evidencia de niveles de abstracción débiles (Gomes y otros, 2006).

Gomes y Mendes son más directos en relación a la situación planteada y afirman que:

> "*Los alumnos no saben resolver problemas. Creemos que la causa más importante de las dificultades que muchos estudiantes de primer año para aprender a programar es su falta de habilidades genéricas de resolución de problemas. Los estudiantes no saben cómo crear algoritmos, principalmente porque no saben cómo resolver problemas. La resolución de problemas requiere múltiples habilidades que los alumnos no suelen tener*" (Gomes y Mendes, 2007).

En tal sentido, han identificado en esta situación, varios problemas que deben superarse en los primeros cursos de programación, tales como: a) deficiencias en la comprensión del problema (muchas veces los estudiantes intentan resolver un problema sin entenderlo de manera completa); b) falta de capacidad para relacionar conocimientos, lo cual se evidencia cuando no pueden establecer analogías correctas con problemas anteriores; c) carencia de reflexión sobre el problema y la solución, ya que se evidencia una tendencia a escribir una respuesta sin una reflexión previa.

En relación a las investigaciones mencionadas, entendemos que estamos frente a un desafío educativo asociado a la enseñanza de la computación y, en particular, a la programación en el nivel secundario. En su forma central este nos habla acerca de cómo construimos una propuesta didáctica para la enseñanza de la programación donde el primer paso sea el desarrollo de capacidades en torno a la resolución de problemas. En este sentido, en el siguiente apartado vamos a proponer y poner a discusión una posible solución.

3. Pensamiento computacional: un camino necesario para avanzar en nuestras prácticas de enseñanza de la programación

Como es sabido, programar es mucho más que solo escribir código para computadoras. En pocas palabras, entendemos que la programación es un proceso mediante el cual, una persona desarrolla una solución a un problema y la expresa en la forma de un algoritmo utilizando un lenguaje que pueda ser procesado por una computadora. A partir de las evidencias presentadas en el apartado anterior, se recomienda que antes de trabajar con los lenguajes de programación, es necesario desarrollar una serie de habilidades de pensamiento en torno a la resolución de problemas. En particular, algunas de estas son: la capacidad de abstracción, el análisis y la descomposición en partes y la destreza para la síntesis o recomposición (Insuasti, 2016).

En los últimos años, el término "pensamiento computacional" ha emergido con fuerza asociado a discusiones educativas que se han dado en la educación básica. Si bien es un término de debate actual, polisémico y aún no sedimentado (Adell y otros, 2019), en este trabajo vamos a tomar la definición aportada por la Sociedad Internacional para la Tecnología en la Educación (ISTE) y la Asociación de Profesores de Ciencias de la Computación (CSTA) quienes han desarrollado una definición operativa del pensamiento computacional:

> "*El pensamiento computacional (PC) es un proceso de resolución de problemas que incluye (pero no se limita a) las siguientes características: Formulación de problemas de manera que nos permita utilizar un ordenador y otras herramientas para ayudar a resolverlos. Organizar y analizar lógicamente los datos. Representar los datos a través de abstracciones como modelos y simulaciones. Automatizar las soluciones a través del pensamiento algorítmico (una serie de pasos ordenados). Identificar, analizar y poner en práctica posibles soluciones con el objetivo de lograr la combinación más eficiente y eficaz de pasos y recursos. Generalizar y transferir este proceso de resolución de problemas a una amplia variedad de problemas*" (ISTE, 2011).

De manera complementaria a la definición anterior, ISTE asocia una serie de disposiciones o actitudes que constituyen dimensiones esenciales para el desarrollo del PC. Siendo estas: Confianza en el manejo de la complejidad; persistencia en el trabajo con problemas difíciles; tolerancia a la ambigüedad; capacidad de tratar problemas abiertos y capacidad de comunicar y trabajar con otros para lograr un objetivo o solución común.

Cuando analizamos los problemas encontrados en el relevamiento bibliográfico para la enseñanza de la programación, y tras presentar la definición del PC, resulta evidente que se encuentran coincidencias entre ellos. En primer lugar y desde la misma definición, este pensamiento está orientado a desarrollar habilidades y estrategias para la resolución de problemas, carencia que es destacada por las investigaciones referenciadas. Por otro lado, y con mayor precisión, tales investigaciones hacen referencia al bajo desarrollo de las mismas capacidades que ISTE señala como propias del PC: pensamiento algorítmico y lógico, capacidad de abstracción y descomposición de problemas.

Si bien el concepto pensamiento computacional hoy está en debate y no hay acuerdos firmes sobre sus alcances, entendemos que de manera central se relaciona con el problema educativo que estamos tratando y por lo tanto amerita ser parte de nuestra discusión. En este sentido, percibimos que podría colaborar en la concientización y visibilización, por parte de la comunidad docente, de alguno de los problemas centrales asociados a un primer curso de programación; haciendo hincapié en que antes de trabajar con aspectos más técnicos de la programación es necesario desarrollar capacidades en nuestros estudiantes en relación a la comprensión y la resolución de problemas.

En la educación argentina, este problema educativo recién empieza a ser tenido en cuenta en el currículum de la escuela básica, a partir de la aprobación de los Núcleos de Aprendizaje Prioritarios de Educación Digital, Programación y Robótica.[4] Los mismos incluyen, dentro de su propuesta de alfabetización digital, objetivos de enseñanza y de aprendizaje relacionados con la resolución de problemas y la programación de computadoras. En este momento las distintas jurisdicciones educativas están discutiendo su modo de incorporación a los distintos diseños curriculares. Por otro lado, en agosto de 2021, desde el Ministerio de Educación se ha anunciado el Programa Nacional de Ciencia y Tecnología en la Escuela el cual, en su punto dos, indica que es importante la "potenciación y actualización de la educación tecnológica con contenidos de Ciencias de la Computación".[5]

Puede ser que en un futuro cercano las habilidades de resolución de problemas se desarrollen con mayor intensidad en la educación básica obligatoria y, de alguna forma, se naturalice el trabajo de estos temas en distintos espacios curriculares. Seguramente, en ese momento, ya no se necesitará hablar tanto de pensamiento computacional, no obstante, mientras esto no suceda entendemos que tal concepto nos ayuda a visibilizar y comenzar a abordar el mencionado problema educativo.

4. Consideraciones finales

En el presente, donde el paradigma informacional (Castells, 2002) está sustituyendo en una buena parte al industrialismo y la aceleración de los desarrollos científicos y las aplicaciones tecnológicas favorecen un entorno de cambios constantes, está cada vez más definido que los estudiantes deben ser capaces de pensar críticamente y resolver problemas complejos. En este contexto la informática debe verse centralmente como una disciplina de resolución de problemas con aplicaciones en muchas áreas del mundo real (Grover, y otros, 2016) . En particular, enseñar a programar, se configura como uno de los caminos posibles para que las personas se apropien de la tecnología y la puedan adaptar según sus necesidades e intereses.

Por otro lado, según hemos analizado, aprender a programar implica algo que va más allá de solo memorizar la sintaxis y conocer la semántica de un lenguaje de programación. Desarrollar habilidades en esta área es un asunto complejo que requiere de un conjunto de capacidades y conocimientos que están en relación con analizar, entender y proponer diferentes soluciones algorítmicas a problemas. Más aún, entendemos que cada problema (o conjunto de problemas, estrictamente hablando) se soluciona de una manera distinta y cada programador tiene su propia forma de hacerlo.

Ante la importancia y necesidad de desarrollo de estas capacidades, entendemos que la resolución de problemas es una habilidad inicial que se debe abordar en los primeros cursos de programación y así, hablar de pensamiento computacional es una forma de ayudar a darle nombre y forma a un problema educativo.

Las nuevas directivas curriculares que incorporan la enseñanza de la computación a la educación básica obligatoria nos presentan varios desafíos educativos, pero también la posibilidad de ser coautores en la construcción de una nueva didáctica disciplinar. No podemos simplemente transferir la experiencia acumulada en el nivel superior de enseñanza (la cual efectivamente es mucha), por lo cual se hace necesario revisar los por qué, los para qué y también

[4]Resolución Consejo Federal de Cultura y Educación N° 343/18.

[5]Registro audiovisual del lanzamiento del Programa Nacional de Ciencia y Tecnología en la Escuela (20 de agosto de 2021) https://www.youtube.com/watch?v=gJ_3kMlyWmc.

los cómo, a la luz de las nuevas normativas directivas como así también de las edades de los estudiantes y las realidades de los actores involucrados.

Bibliografía

ACM/AIS/IEEE (2005). Computing Currícula 2005. The Joint Task Force for Computing Curricula 2005. Disponible en https://bit.ly/3kRRbbE.

Aissa, M., Al-Kalbani, M., Al-Hatali, S. y Touq Ahmad A. (2020). Novice Learning Programming Languages in Omani Higher Education Institution (Nizwa University) Issues, Challenges and Solutions. En A. Al-Masri y Y. Al-Assaf (eds.) Sustainable Development and Social Responsibility−Volume 2. Springer

Adell, J., Llopis, M., Esteve, M., y Valdeolivas, N. (2019). El debate sobre el pensamiento computacional en educación. RIED. Revista Iberoamericana de Educación a Distancia, 22(1), pp. 171–186. doi: http://dx.doi.org/10.5944/ried.22.1.22303.

Babori, A., Fihri, H., Hariri, A. y Bideq, M. (2016). An e-Learning environment for algorithmic: Toward an active construction of skills. World. En Journal on Educational Technology: Current Issues, 8(2), pp. 82-90

Baldwin, L. y Kuljis, J. (2001). Learning programming using program visualization techniques. En Proceedings of the 34th Hawaii International Conference on System Sciences.

Bosse, Y. y Gerosa, M. (2017). Why is programming so difficult to learn? Patterns of Difficulties Related to Programming Learning Mid-Stage. ACM SIGSOFT Software Engineering Notes, 41(6), pp. 1–6, https://doi.org/10.1145/3011286.3011301.

Castells, M. (2002). Epílogo, en P. Himanen, La ética del hacker y el espíritu de la era de la información. Barcelona: Destino.

Chin Soon, C. (2020). Factors Contributing to the Difficulties in Teaching and Learning of Computer Programming: A Literature Review. Contemporary Educational Technology, v.12, n.2

Gomes, S. y Mendes, A. (2007). Learning to program-difficulties and solutions. En International Conference on Engineering Education − ICEE 2007, Coimbra, Portugal.

Gomes, A., Carmo, L., Bigotte, E. y Mendes, A. (2006). Mathematics and programming problem solving. 3rd E-Learning Conference−Computer Science Education.

Grover, S. (2018). The 5th 'C' of 21st century skills? Try computational thinking (not coding). EdSurge News. Disponible en https://bit.ly/3x1PSfc.

Grover, S., Pea, R., y Cooper, S. (2016). Factors influencing computer science learning in middle school. En Proceedings of the 47th ACM technical symposium on computing science education, https://doi.org/10.1145/2839509.2844564.

Insuasti, J. (2016) Problemas de enseñanza y aprendizaje de los fundamentos de programación. En Revista educación y desarrollo social, 10 (2), pp. 234–246. https://revistas.unimilitar.edu.co/index.php/reds/article/view/1966.

International Society for Technology in Education (ISTE) y Computer Science Teachers Association (CSTA). (2011). Operational Definition of Computational Thinking for K-12 Education. Disponible en https://cdn.iste.org/www-root/Computational_Thinking_Operational_Definition_ISTE.pdf.

Lister, R., Adams, E., Fitzgerald, S., Fone, W., Hamer, J., Lindholm, M., McCartney, R., Moström, J., Sanders, K., Seppälä, O., Simon, B. y Thomas, L. (2004). A multi-national study of reading and tracing skill. SIGCSE Bulletin, V. 36, I. 4, pp. 119–150.

Luxton-Reilly, A. (2016). Learning to program is easy. En Proceedings of the 2016 ACM Conference on Innovation and Technology in Computer Science Education. https://doi.org/10.1145/2899415.2899432.

Luza, C. (2017). La computación y solución de problemas computacionales. Perspectiv@S, 13(12), 23-27. Recuperado de http://revistas.uigv.edu.pe/index.php/perspectiva/article/view/208.

Mayer, R. (1990). Problem solving. En M. Eysenck (ed.), The Blackwell Dictionary of Cognitive Psychology, Oxford: Basil Blackwell, pp. 284–288.

OECD. (2013). Problem solving framework, in PISA 2012. OECD Publishing.

Qian, Y. y Lehman, J. (2016). Correlates of success in introductory programming: A study with middle school students. Journal of Education and Learning, 5(2), 73. https://doi.org/10.5539/jel.v5n2p73.

Savage, S. y Piwek, P. (2019). Full report on challenges with learning to program and problem solve: an analysis of first year undergraduate Open University distance learning students? online discussions. Disponible en http://oro.open.ac.uk/68073/1/IOC_TM112_Python_Help_Forum_Research.pdf.

Susanti, W., Jama, J., Krismadinata., Ramadhani, D. y Nasution, T. (2021). An Overview Of The Teaching And Learning Process Basic Programming In Algorithm And Programming Courses. En Turkish Journal of Computer and Mathematics Education, Vol.12, N.2, pp. 2934–2944.

Uzunboylu, H., Kinik, E., y Kanbul, S. (2017). An Analysis of Countries Which Have Integrated Coding into Their Curricula and the Content Analysis of Academic Studies on Coding Training in Turkey. TEM J. 6, pp. 783–791.

Evaluación de producciones de docentes de primaria en formación en Pensamiento Computacional

Francisco Bavera*† pancho@dc.exa.unrc.edu.ar UNRC - UNViMe

Teresa Quintero‡ tquintero@exa.unrc.edu.ar UNRC

Marcela Daniele* marcela@dc.unrc.edu.ar UNRC

Resumen

Aún no existe un estricto consenso de la definición del Pensamiento Computacional, pero la mayoría de los autores acuerdan y coinciden con la apropiación de ciertas habilidades, las que se identifican como: abstracción, pensamiento algorítmico, reconocimiento de patrones, descomposición, generalización, análisis lógico y evaluación. Desde distintos ámbitos se promueve la incorporación del Pensamiento computacional en los diseños curriculares de la educación primaria y la indiscutible necesidad de la formación de los educadores. También se pone la atención en las mediciones y evaluaciones requeridas para validar los resultados y mejorar las propuestas de incorporación. En este trabajo se detectan y analizan habilidades asociadas al Pensamiento Computacional que se encuentran presentes en los trabajos finales realizados por docentes de nivel primario que cursaron una Especialización Docente en Didáctica de las Ciencias de la Computación. Con este análisis se pretende avanzar en estudios científicos que aporten a la evaluación, análisis y comprensión de la manera en la que este grupo de docentes de primaria, con formación en didáctica de las ciencias de la computación y Pensamiento Computacional, desarrollaron estas habilidades y cómo las trasladan al aula.

Para realizar este trabajo, se analizaron y evaluaron muestras de distintas producciones de los docentes en diferentes instancias de la formación recibida. Se consideran producciones en modalidad posters, trabajos finales de los distintos módulos, trabajo final integrador, observaciones realizadas a las prácticas docentes en las escuelas, encuestas a docentes de primaria, y presentaciones orales en el marco del cierre de la especialidad. Los resultados obtenidos aportan al desarrollo de conocimientos sobre la formación continua de docentes de Educación Primaria en Didáctica de las Ciencias de la Computación.

Palabras clave: Pensamiento Computacional, Habilidades Cognitivas, Integración, Formación docente continua.

*Departamento de Computación, FCEFQyN, UNRC.

†Escuela de Ingeniería, UNViMe.

‡Departamento de Física, FCEFQyN, UNRC.

1. Introducción

La enseñanza de las Ciencias de la Computación permite promover la construcción del Pensamiento Computacional (PC), es decir, de capacidades y competencias útiles para la búsqueda de soluciones a diversos tipos de problemas, aportando además a la formación de ciudadanos críticos, participativos y autónomos en el uso de las tecnologías (Valverde Berrocoso et al., 2015). Diversos estudios a nivel mundial dan cuenta de la importancia de introducir y desarrollar el PC en el sistema educativo obligatorio. Cabe remarcar que, el pensamiento computacional, como un modelo de pensamiento, es importante no solo en ciencias de la computación y matemáticas, sino también en la educación en ciencia, tecnología, ingeniería y matemáticas (STEM) (Li et al., 2020).

Por otra parte, numerosas investigaciones, concluyen en que la mayoría de los docentes que en la actualidad enseñan en los primeros años de educación básica (en especial en países emergentes), presentan marcadas falencias en el desarrollo de habilidades asociadas al PC, como abstracción, generalización, definición de patrones y resolución de problemas. Esto genera deficiencias significativas cuando propuestas educativas en torno al PC son implementadas en aulas de primaria, lo que conduce a la necesidad de comenzar por la formación de docentes. Desde 1980, Papert (Papert, 1978) abogó por que los niños pudieran aprender a pensar de manera diferente, fomentando el pensamiento procesual a través de la programación. Pero fue recién en el año 2006, cuando Wing (Wing, 2006) comienza a hablar de Pensamiento Computacional y propone una definición que sentó las bases a profundas y prometedoras discusiones en torno, a la definición del término, a la incorporación del PC en los diseños curriculares desde la educación primaria, a la indiscutible necesidad de la formación de los educadores, a las mediciones y evaluaciones requeridas para validar los resultados y mejorar las propuestas.

En una amplia revisión de artículos (Catlin y Woollard, 2014; Wing, 2006; Wing, 2008; Wing, 2011; Brennan y Resnick, 2012; Bers, 2018; ISTE y CSTA, 2011; Csizmadia et al., 2015), se puede ver una plena coincidencia entre todos los autores que abordan el estudio del PC, de que aún existen importantes controversias para lograr una definición consensuada y acabada de Pensamiento Computacional. No obstante, existen términos que claramente forman parte de dicha definición como son: resolución de problemas, descomposición, abstracción, generalización, patrones y algoritmos o solución algorítmica. De la misma manera, existen propuestas que avanzan en las mediciones de las habilidades de pensamiento computacional construidas tanto por niños, niñas y adolescentes, como por los educadores que están recibiendo distintas formaciones en la temática.

Estudios realizados permiten señalar que el desempeño de los docentes en el ejercicio de su profesión está más íntimamente ligado a su capacidad comunicativa y sus habilidades didácticas que, por ejemplo, a tener títulos de posgrado (Elacqua et al., 2018). Además, hay evidencia de que los resultados de aprendizaje tienen como techo la calidad del docente (Darling-Hammond, 2000; Darling-Hammond et al., 2005; Barber y Mourshed, 2008). Existe un importante y constante aumento de investigaciones que demuestran el rol fundamental que juegan los docentes en el logro de una educación de calidad (Secretaria de Evaluación Educativa, 2018). Es decir, el factor docente debería considerarse como la variable, a nivel escuela, que mayor impacto tiene en el aprendizaje del estudiante (OREALC-UNESCO, 2013).

La construcción del PC requiere mayor análisis e investigación, explorando sus dimensiones, importancia y beneficios (Brennan y Resnick, 2012). En nuestro país, y en particular en nuestra provincia, se hace necesario profundizar en trabajos de investigación que reflejen los resultados obtenidos con educadores, como así también, con alumnos de la escuela primaria, que hayan recibido formación relativa a la construcción del pensamiento computacional.

En la actualidad, el desafío es evaluar en profundidad el impacto de estas iniciativas formativas en PC en la escuela (Bocconi et al., 2016), en particular, estudiar si a partir de experiencias de formación los docentes construyen dialécticamente su propio PC y se apropian de maneras/modos de construcción que se constituyen en nuevas prácticas educativas que determinan la posibilidad del desarrollo de dicho pensamiento en sus estudiantes. Los autores de este trabajo forman parte de un equipo de investigación que ha comenzado un proceso de estudio (Daniele et al., 2019; Bavera et al., 2019; Bavera et al., 2019; Bavera et al., 2020) intentando atrapar parte de la complejidad inherente al proceso de construcción del PC en la escuela, que se identifica a partir de los ejemplos particulares que representa cada grupo –docentes y sus estudiantes de la escuela– en los que se focaliza la investigación. La producción de conocimiento de esta investigación será uno de los insumos para el desarrollo de propuestas de enseñanza situadas para la formación y el desarrollo del PC en docentes y estudiantes.

En el año 2016, se conformó un equipo de trabajo multidisciplinar integrado por informáticos, físicos, químicos, matemáticos y pedagogos, algunos con sólida formación en didáctica de las ciencias para realizar el diseño de una Especialización Docente de Nivel Superior en Didáctica de las Ciencias de la Computación (EDNSDCC) dirigida a docentes de educación primaria. En el año 2018 comenzó el dictado de la EDNSDCC, iniciativa conjunta de la Fundación Sadosky, el Instituto Superior de Formación Docente Ramón Menéndez Pidal y la Facultad de Ciencias Exactas Físico-Químicas y Naturales de la Universidad Nacional de Río Cuarto. Esta instancia de formación destinada a docentes de Nivel Primario tenía como objetivo introducir a los docentes en las Ciencias de la Computación y la programación, en el PC, con el fin de ofrecerles herramientas y experiencias de aprendizaje, para que puedan introducirlas y replicarlas en sus aulas.

En este trabajo, se presenta el análisis realizado a partir de observaciones de clase y de las producciones personales y grupales de los docentes de primaria cursantes. Entre el material analizado podemos mencionar: problemas resueltos, programas entregados, planificaciones realizadas y actividades de prácticas docentes.

2. Habilidades Asociadas al Pensamiento Computacional

Si bien no hay un estricto consenso en cuanto a la definición del Pensamiento Computacional existen habilidades comunes en que coinciden la mayoría de los autores (Catlin y Woollard, 2014; Wing, 2006; Wing, 2008; Wing, 2011; Brennan y Resnick, 2012; Bers, 2018; ISTE y CSTA, 2011; Csizmadia et al., 2015). Estas habilidades son: *abstracción, pensamiento algorítmico, reconocimiento de patrones, descomposición, generalización, análisis lógico y evaluación*.

La **abstracción** es el proceso de hacer un artefacto más comprensible a través de la reducción de los detalles innecesarios. La habilidad en la abstracción reside en la elección del detalle a ocultar de manera que el problema se vuelva más fácil, sin perder todo lo que es importante. Una parte fundamental de la misma es la elección de una buena representación de un sistema. Diferentes representaciones hacen diferentes cosas fáciles de hacer (Csizmadia et al., 2015).

El **Pensamiento Algorítmico** es una forma de llegar a una solución a través de una definición clara de los pasos (Csizmadia et al., 2015).

El **Reconocimiento de Patrones** consiste en extraer información que permita establecer propiedades/relaciones entre objetos. Estos objetos pueden ser tanto físicos como abstractos.

La **Descomposición** es una manera de pensar acerca de los artefactos en términos de sus partes y componentes. Cada pieza debe entenderse, solucionarse, desarrollarse y evaluarse por separado. Esto hace más fácil resolver problemas

complejos, y grandes sistemas son más fáciles de diseñar (Csizmadia et al., 2015).

La **Generalización** se asocia con la identificación de patrones, similitudes y conexiones, y la explotación de estas características para transferir el proceso de solución de problemas a una gran diversidad de contextos. Es una forma de resolver rápidamente los nuevos problemas sobre la base de las soluciones en los problemas anteriores, y la construcción en la experiencia previa. Haciendo preguntas tales como *"¿Esto es similar a un problema que ya he solucionado?" y "¿Cómo es diferente?"* Algoritmos que resuelven algunos problemas específicos se pueden adaptar para resolver toda una clase de problemas similares (Csizmadia et al., 2015).

El **Análisis Lógico** consiste en aplicar e interpretar la lógica (booleana) de manera correcta.

La **Evaluación** consiste en sistematizar (a través de distintos criterios) y hacer un juicio de valor. Se incluyen dentro de esta habilidad la depuración, validación y verificación de las soluciones a problemas. Involucra el análisis lógico para predecir y verificar los resultados.

3. La Especialidad en Didáctica de la Ciencias de la Computación

La primera cohorte de la Especialización Docente de Nivel Superior en Didáctica de las Ciencias de la Computación comenzó el cursado en marzo de 2018. Esta formación se trató de una iniciativa conjunta del Instituto de Formación Docente Ramón Menéndez Pidal conjuntamente con la Facultad de Ciencias Exactas Físico-Químicas y Naturales de la Universidad Nacional de Río Cuarto y la Fundación Sadosky.

La Especialización tuvo una duración de 400 horas, divididas en 8 módulos, a lo largo de dos años. Los temas desarrollados en estos módulos fueron: huella digital, ciudadanía digital responsable, derechos de autor y propiedad intelectual. Estrategias para la resolución de problemas, abstracción, reconocimiento de patrones y generalización. Instrucciones, algoritmos, lenguajes de programación. Prueba y depuración de algoritmos. Verificación y validación de algoritmos. Relación entre software y hardware. Conceptos básicos de sistemas operativos y redes. Programación de robots educativos. Representación y codificación de la información y criptografía. A lo largo del dictado se utilizaron actividades enchufadas y desenchufadas. Entre las herramientas utilizadas en las actividades enchufadas se pueden mencionar: Lightbot, code.org, Pilas-Bloques, Tortugarte y MBot.

Culminaron el cursado cuarenta y tres (43) Profesores de Educación Primaria (40 maestras y 3 maestros), estudiantes de la especialidad, que forman parte de veintiocho (28) escuelas primarias de la ciudad y la región. El 98 % de estos docentes de nivel primario son maestras/os de grado y sólo una maestra del área Informática. La media de edad es 42 años, en un rango que va desde los 23 a los 54 años. Con una mediana y una moda de 43 y 44 años, respectivamente. La desviación media es de 4,5. El 50 % son docentes titulares y los restantes son interinos y suplentes, que se desempeñan en los diferentes grados de la escuela primaria. Además, dos docentes no están relacionados con institución alguna y un docente desempeña actividades en forma ad-honorem. Los docentes pertenecen a 28 escuelas, donde el 80 % son instituciones públicas de gestión estatal y el restante 20 % instituciones públicas de gestión privada. Un 75 % de estas escuelas están localizadas en la ciudad de Río Cuarto, y el restante 25 % en localidades de su región de influencia (en un radio de 30 a 150 km de distancia de Río Cuarto).

El 98 % de los docentes informaron, al comienzo de la especialidad, que tenían conocimientos de informática. De los cuales, el 30 % proviene de capacitaciones formales, otro 30 % recibió capacitaciones informales y el restante 38 % se considera autodidacta. Ninguno participó previamente en capacitaciones relacionadas con las Ciencias de la Computación, la programación o la robótica. El 2 % restante no tiene conocimiento alguno de informática.

Al indagar sobre las experiencias previas que tuvieron con la informática en el ejercicio docente, se pudo recabar que, 21 docentes emplearon utilitarios de oficina, 7 aplicaciones de celulares y 9 docentes otras aplicaciones. Mientras que 12 maestras nunca realizaron experiencias relacionadas con la informática en su aula. Sólo 5 de los docentes tuvieron alguna experiencia con lenguajes de programación visuales y 2 con robótica educativa.

4. Metodología

Se planteó una investigación exploratoria, descriptiva desde un enfoque cualitativo. Para el análisis de las producciones se siguió una metodología cualitativa, basada en el análisis del contenido y en la Teoría Fundamentada. Los datos obtenidos fueron triangulados entre sí, por al menos dos investigadores. Los objetos de estudio fueron las planificaciones de actividades propuestas por los docentes, junto con, las observaciones de sus prácticas docentes al implementar dichas propuestas en sus aulas. Todos son docentes de educación primaria y estudiantes de la especialidad. Se analizaron las propuestas presentadas y se consideraron las observaciones y notas realizadas por los investigadores en las prácticas docentes, en las entrevistas con los docentes y en la revisión de las producciones. Los mismos se analizaron en función de las habilidades del Pensamiento Computacional mencionadas anteriormente.

5. Resultados

Las producciones representaron diferentes dimensiones de la formación docente: cómo habitaron la formación y capacitación sobre un contenido disciplinar, cómo construyeron conocimientos y cómo resolvieron la transposición didáctica en sus aulas de la escuela primaria.

Los docentes lograron trabajar contenidos y habilidades abordados en los módulos de la Especialización en distintas áreas de la enseñanza primaria, con aplicación en las diferentes áreas escolares. Las áreas de aplicación de las actividades planificadas: 33 % Lengua; 16 % Lengua y Matemática, 15 % Ciencias Naturales y el resto fueron interdisciplinarias (Matemática, Ciencias Sociales, Tecnología y Lengua; Ciencias Sociales y Naturales, Ciudadanía y Computación). En la mayoría de los trabajos se observa, la transferencia al aula y el trabajo con sus estudiantes de algunas de las propuestas desarrolladas en la especialización.

El 75 % de las actividades se dirigieron a alumnos de quinto y sexto grado y el resto se repartió por partes iguales entre alumnos de segundo grado y cuarto grado.

Las producciones de los grupos docentes giraron principalmente sobre: situaciones problemáticas, trabajo colaborativo y desarrollo del pensamiento computacional Las actividades implementadas en el aula se establecieron en torno a la búsqueda y explicitación de estrategias en la resolución de los problemas con posteriores debates y análisis entre sus alumnos sobre las distintas soluciones. Hicieron hincapié en la validación de las soluciones, la detección y solución de los errores. Trabajaron claramente sobre la necesidad de contar con instrucciones precisas y no ambiguas en la definición de secuencias de instrucciones (algoritmos) para expresar las soluciones a los problemas.

El 50 % de las actividades planificadas trabajaron Abstracción. Todos los trabajos se centraron en Pensamiento Algorítmico y Evaluación (depuración, testing y evaluación fueron los aspectos trabajados). Reconocimiento de Patrones, Descomposición en Subproblemas y Análisis Lógico se encontró en el 70 %-80 % de las planificaciones y actividades realizadas por los docentes. Generalización solo fue trabajada por el 15 % de los docentes.

Con respecto al nivel de reconocimiento y manejo de de los conceptos básicos asociados a la enseñanza de la programación, el 60 % se encuentra en un nivel básico y el resto se reparte casi en partes iguales dentro de un nivel incipiente y suficiente.

En el aula realizaron actividades enchufadas y desenchufadas. El 82 % de los docentes trabajaron ambos tipos de actividades con sus estudiantes, el 3 % solo realizaron actividades enchufadas y el restante 15 % solo realizó actividades desenchufadas.

Sin embargo, aproximadamente el 80 % de los docentes cursantes tienen un mayor grado de predisposición a desarrollar actividades desenchufadas. Aunque, es similar el porcentaje que considera que podría introducir a la programación y/o enseñar a programar animaciones o simulaciones simples.

Por otra parte, una cantidad elevada de docentes demuestran que pueden identificar las habilidades del PC involucradas en la resolución de diferentes problemas, siendo el *Reconocimiento de Patrones* la habilidad que pudieron reconocer con mayor destreza.

Algunas habilidades que se encontraron en las producciones y que se identifican con las del pensamiento en la era digital son: dar y seguir instrucciones, secuenciar, ordenar, probar, ejecutar, verificar, analizar, validar, codificar, validar distintas estrategias de solución, identificar situaciones, descomponer en subproblemas, construir algoritmos, reconocer patrones, usar herramientas de colaboración y comunicación, aplicar estrategias, trabajar colaborativamente, buscar y validar información, abordaje y resolución de problemas, programación en acción, robótica, producciones creativas, fundamentación de procedimientos.

Otro aspecto destacable es que un alto porcentaje afirma que esta formación les permitió adquirir y/o afianzar ciertas habilidades relacionadas con el pensamiento computacional:

- Abstracción: 61 % de los docentes cursantes.
- Descomposición: 61 % de los docentes cursantes.
- Reconocer Patrones: 45 % de los docentes cursantes.
- Solución Algorítmica: 45 % de los docentes cursantes
- Modelado y simulación: 11 % de los docentes cursantes.

De los docentes cursantes el 55 % considera que el pensamiento crítico y computacional les permitió definitivamente transformar y mejorar sus prácticas docentes. Mientras que el 45 % respondió que cambió y mejoró sólo en parte sus prácticas. Además, es importante resaltar que el 72 % asegura que modificó algunas de sus actividades docentes teniendo en cuenta: la programación, el diseño de algoritmos, la abstracción, la descomposición del problema, la definición de modelos y de patrones. Mientras que el 22 % afirma que transformó todas sus actividades docentes. Sólo el 6 % afirma que no lo hizo, pero que lo realizará en el futuro. Ningún docente respondió que ya lo realizaba antes de cursar el postítulo o que considera que no podría hacerlo.

6. Conclusiones y Trabajos Futuros

Del análisis realizado se obtuvieron una serie de resultados que aportan al desarrollo de conocimientos sobre la formación continua de docentes de Educación Primaria en Didáctica de las Ciencias de la Computación. Se encontró que los docentes pudieron planificar y llevar al aula actividades tendientes a desarrollar el Pensamiento Computacional en sus alumnos. Si bien lograron enfocar las actividades en los objetivos (relacionados al PC) por ellos planteados, necesitaron un fuerte acompañamiento y asistencia del equipo que dictó la especialidad. Por ejemplo, en actividades

que involucran habilidades de ordenamiento o búsqueda, necesitaron un fuerte trabajo para comprender que el resultado buscado no consistía solamente en que sus alumnos lograran ordenar/buscar correctamente, sino que también era muy importante que enfocarán la actividad en favorecer el razonamiento sobre los procesos realizados al buscar/ordenar. Que también, lograran hacer foco en generalizar los procesos involucrados o en favorecer procesos de abstracción. En relación a la representación de la información, es importante trabajar en el aula diferentes maneras de representación, lo que conduce a la necesidad de que los alumnos analicen y reflexionen sobre las ventajas y desventajas de esas diferentes representaciones en distintos contextos/problemas. Es decir, para garantizar una transposición didáctica significativa, se requiere del acompañamiento y supervisión de esas experiencias por especialistas (Profesores y Licenciados en Ciencias de la Computación) para dar continuidad y profundizar la construcción de conocimientos disciplinares y didácticos específicos, hasta que los docentes en formación profundicen sus comprensiones.

Es importante señalar que en la mayoría de los casos las propuestas presentadas por los docentes consistieron en réplicas de las actividades desarrolladas en la especialidad.

La recepción, por parte de sus estudiantes, de las actividades de introducción a la programación, pensamiento computacional y Ciencias de la Computación (enchufadas y desenchufadas) los sorprendió gratamente. Ya que los resultados y las respuestas que obtuvieron superaron sus expectativas. Mencionaron los términos motivación, interés, entusiasmo, logros, experimentación y curiosidad de sus alumnos para expresar la respuesta que tuvieron en el aula.

Ninguna de las actividades planificadas por los docentes se enfocó en automatizar soluciones, creemos que esto se debe tanto al nivel educativo al que están dirigidas dichas actividades, como así también, a la propia especialidad.

Se pretende continuar valorando las habilidades de Pensamiento Computacional, que poseen los docentes participantes de la Especialización, con otras metodologías. También se está trabajando en la evaluación de la inclusión del PC en la Formación Docente Inicial y en la colaboración con los docentes que cursaron la Especialización en el diseño e implementación de propuestas didácticas en aulas de las Escuelas Primarias.

Por otra parte, se planifica incursionar en el análisis de las percepciones de los docentes de Educación Primaria con respecto a las Ciencias de la Computación, la programación y la robótica. Este análisis también posibilitó vislumbrar el aporte, en cuanto al cambio de las percepciones de los docentes, de distintas formaciones en Ciencias de la Computación, Pensamiento Computacional, Programación y Robótica desarrolladas. La evaluación de las producciones docentes y su trabajo áulico, brindó nuevos conocimientos contextuales que son un recurso importante para pensar la formación continua en estas temáticas y el acompañamiento a los docentes de la Especialización.

Bibliografía

Barber, M. y Mourshed M. (2008). Cómo hicieron los sistemas educativos con mejor des-empeño del mundo para alcanzar sus objetivos. DC: PREAL.

Bavera, F., Quintero, T., Daniele, M., Buffarini, F. (2019). Análisis de prácticas de docentes de educación primaria en el marco de una formación en pensamiento computacional. Aprobado para las 48 Jornadas Argentinas de Informática, SAEI-JAIIO, Universidad Nacional de Salta, 16 al 20 de septiembre de 2019.

Bavera, F., Quintero, T., Daniele, M., Buffarini, F. (2019). Habilidades de Pensamiento Computacional en docentes de primaria: evaluación usando Bebras. Publicado en las actas del XXV Congreso Argentino de Ciencias de la Computación (CACIC 2019). Universidad Nacional de Río Cuarto.

Bavera, F., Quintero, T., Daniele, M., Buffarini, F. (2020). Computational Thinking Skills in Primary Teachers: Evaluation Using Bebras. Communications in Computer and Information Science book series (CCIS, volume 1184). Springer International

Publishing. Computer Science – CACIC 2019. 25th Argentine Congress of Computer Science, CACIC 2019, Revised Selected Papers.

Bers, M. U. (2018). Codings as a Playground: Programming and Computational Thinking in the Early Childhood Classroom. New York: Routledge. https://doi.org/10.4324/9781315398945

Bocconi, S., Chioccariello, A., Dettori, G., Ferrari, A., Engelhardt, K. (2016), Developing computational thinking in compulsory education – Implications for policy and practice; EUR 28295 EN; doi: 10.2791/792158

Brennan, K., y Resnick, M. (2012). Entrevistas basadas en artefactos para estudiar el desarrollo del Pensamiento Computacional (Pensamiento Computacional) en el diseño de medios interactivos. Vancouver, BC, Canada: American Educational Research Association.

Catlin, D., y Woollard, J. (2014). Educational robots and computational thinking. En and others (Ed.), 4th TRtWR & RIE 2014. International workshop: Teaching robotics & teaching with robotics.

Csizmadia, A., Dorling, M., Ng, T. y Selby, C.. (2015) Computational thinking-a guide for teachers. Computing at School.

Daniele, M., Quintero, T., Bavera, F., Buffarini, F., Solivellas, D., y De Dominici, C. (2019). Análisis de producciones de docentes de educación primaria con formación en didáctica de las ciencias de la computación. Aprobado y presentado en las Segundas Jornadas de Didáctica de la Programación, FAMAF, Universidad Nacional de Córdoba, 7 y 8 junio 2019.

Darling-Hammond, L. (2000). Teacher Quality and Student Achievement. A review of state policy evidence. En Education policy analysis archives, 8 (1), pp. 1–44.

Darling-Hammond, L., Holtzman, D. J., Gatlin S. J. y Vasquez Heilig, J. (2005). Does Teacher Preparation Matter? Evidence about Teacher Certification, Teach for America, and Teacher Effectiveness. En Education Policy Analysis Archives, 13 (42), pp. 1–51

Elacqua, G., Hincapié, D., Vegas, E., y Alfonso, M. (2018). Profesión: Profesor en América Latina. ¿Por qué se perdió el prestigio docente y cómo recuperarlo? Washington, DC, Estados Unidos: Banco Interamericano de Desarrollo.

ISTE, y CSTA. (2011). Pensamiento computacional: caja de herramientas para líderes. Descargado de http://eduteka.icesi.edu.co/pdfdir/PensamientoComputacional_Definicion.pdf

Li, Y., Schoenfeld, A. H., diSessa, A. A., Graesser, A. C., Benson, L. C., English, L. D., y R. A. Duschl (2020). Computational Thinking Is More About Thinking than Computing. Journal for STEM Education Research. Springer Nature Switzerland AG. https://doi.org/10.1007/s41979-020-00030-2

OREALC-UNESCO. (2013). Antecedentes y Criterios para la Elaboración de Políticas Docentes en América Latina y el Caribe. Centro de Estudios de Políticas en Educación (CEPPE). París, Francia: UNESCO.

Papert, S. (1978). Personal Computing and its Impact on Education. En Harris, D. (Ed) Proceedings of the P. Wegg Memorial Conference. Iowa City: University of Iowa.

Secretaria de Evaluación Educativa. (2018) Evaluación diagnóstica 2017. Estudiantes avanzados de carreras docentes. Ministerio de Educación, Cultura, Ciencia y Tecnología. Argentina.

Valverde Berrocoso, J., Fernández Sánchez, M. R., y Garrido Arroyo, M. C. (2015). El pensamiento computacional y las nuevas ecologías del aprendizaje. RED. Revista de Educación a Distancia, Vol. , núm.46, pp.1–18 [Consultado: 22 de febrero de 2021]. Disponible en : https://www.redalyc.org/articulo.oa?id=54741184003

Wing, J. (2006). Computational thinking. CACM Viewpoin. https://doi.org/10.1145/1118178.1118215

Wing, J. (2008). Computational thinking and thinking about computing. Philosophical Transactions Of The Royal Society, 366, 3717–3725. https://doi.org/10.1098/rsta.2008.0118

Wing, J. (2011). Computational thinking: What and Why. Descargado de https://www.cs.cmu.edu/link/research-notebook-computational-thinking-what-and-why

Programar es mucho más que solamente secuenciar comandos

Pablo E. "Fidel" Martínez López
fidel@unq.edu.ar
Dpto. CyT, Universidad Nacional de Quilmes
Fundación Dr. Manuel Sadosky

Resumen

En el ámbito del dictado de cursos de programación, muchos de ellos comienzan estableciendo una definición de algoritmo como secuencia de pasos, pero tal término luego juega un rol menor durante el curso. Esta definición dada no es completamente precisa y no se aprovecha después de ninguna forma razonable; por otra parte, la definición misma evidencia una forma de concebir la programación que no incluye a todas las formas de programar que existen hoy en día, y que han mostrado ser mucho más eficaces.

En este artículo buscamos analizar la concepción de la programación que es dominante en los cursos iniciales, y proponer ampliarla de forma de abarcar todas las formas modernas de programación, sin que ello implique grandes cambios a nivel de las actividades propuestas, aunque sí extenderlas y ampliar la forma en que las integramos en un todo. Esto podría redundar en cursos iniciales con mayor proyección para poder continuar luego el estudio de la programación desde ángulos diversos sin grandes saltos disruptivos desde la construcción del conocimiento y en una mayor comprensión sobre la fascinante disciplina de la computación.

Palabras clave: Programación, Algoritmo, Enseñanza, Didáctica, Expresiones, Representación de información.

1. Introducción

Un lugar común de la gran mayoría de los cursos de programación es comenzar por una definición que establece la idea de algoritmo como secuencia de pasos. Notablemente, el término algoritmo juega luego un rol menor durante el resto del curso –si es que aparece nuevamente–, siendo reemplazado rápidamente por el de programa, definido como la concreción en un lenguaje específico de programación de la secuencia de pasos dada por el algoritmo.

¿Por qué comenzar en este lugar si no se aprovechará después? ¿Cuál es la concepción de programación que implica esta definición tan arraigada? ¿Es realmente útil desde el punto de vista didáctico arrancar ahí, o abrazar la concepción implicada por ello? ¿Qué puede ganarse al cuestionar este punto de partida y reemplazarlo por otro más meditado? Estas preguntas nos guían en este artículo para estudiar la concepción de la programación que es dominante en los cursos iniciales, y proponer ampliarla de forma de abarcar todas las formas modernas de programación, sin

que ello implique grandes cambios a nivel de las actividades propuestas, aunque sí extenderlas y ampliar la forma en que las integramos en un todo. Esto podría redundar en cursos iniciales con mayor proyección para poder continuar luego el estudio de la programación desde ángulos diversos sin grandes saltos disruptivos desde la construcción del conocimiento y en una mayor comprensión sobre la fascinante disciplina de la computación.

2. El rey "algoritmo, secuencia de pasos" está desnudo

Es realmente una situación que sorprende que se comience un curso de enseñanza de la programación con una definición que no será luego aprovechada durante el mismo, y que además es incompleta.

Podemos rastrear el término algoritmo hasta los trabajos de Church sobre lambda cálculo en la década de 1930, puesto que lo que se buscaba en ese momento era una forma mecanizable de hacer matemáticas, en palabras de Church, "definir la noción de calculabilidad efectiva en términos de un algoritmo" (Church, 1936). Sin embargo, el lenguaje definido por Church no se basaba en la idea de instrucciones o de secuencia, sino en la de funciones, entendidas como transformaciones de información. Lo más notable es que en esa época aún no existían las computadoras digitales, y tampoco se había desarrollado la noción de programa.

Fue Turing, alumno de Church, quién definió por primera vez la noción de máquina programable con sus 'máquinas de computar' –que Church bautizó como máquinas de Turing, nombre con el que las conocemos hoy– y estudió las posibilidades y los límites teóricos de la noción de programa y de computabilidad (Turing, 1936; Turing, 1938). Posteriormente se desarrollaron las computadoras digitales, basadas en la arquitectura von Neumann (unidad central de proceso, memoria y canales –*buses*– de comunicación entre ambas) (von Neumann, 1945; Godfrey and Hendry, 1993); los lenguajes utilizados para programarlas consistían en un conjunto de instrucciones muy simples para modificar dicha memoria. Así, los primeros programas que efectivamente vieron la luz en computadoras digitales estaban concebidos como secuencias de instrucciones, en lo que hoy conocemos como *assemblers*, o *lenguajes de bajo nivel*. Esta denominación hace referencia a que los lenguajes expresan mejor la idea del funcionamiento de la máquina –considerada como lo que está debajo, el soporte físico de las soluciones programadas– antes que la forma en que piensan las personas que programan –considerados como las ideas que están por arriba de las computadoras que sustentan estos programas, y que son las utilizadas para solucionar los problemas computacionales que se abordan–, siendo los lenguajes que expresan mejor esas formas de pensar los llamados, por lo dicho, *lenguajes de alto nivel*.

La historia de la programación nos muestra que una línea de progreso en el desarrollo de los lenguajes de programación consistió, justamente, en dejar atrás los lenguajes de bajo nivel, donde los programas eran simples secuencias de instrucciones, a lenguajes de alto nivel, expresando de otras formas las nociones de cómputo y transformación de la información. Así, primero surgieron los lenguajes imperativos, que muy posteriormente se refinaron en lenguajes de programación estructurada (Dijkstra, 1968), y rápidamente aparecieron otras formas de lenguajes como los lenguajes funcionales (McCarthy, 1960; Gordon et al., 1978; Milner, 1978), lenguajes basados en objetos (Dahl, 1963; Dahl and Nygaard, 1966) y lenguajes lógicos (McCarthy, 1958).

En los *lenguajes imperativos* la unidad principal para expresar ideas son los *comandos*, formas elaboradas de instrucciones usadas para expresar comportamiento, usando la *memoria* para expresar la noción de estado. En los *lenguajes funcionales*, en cambio, la unidad principal para expresar las ideas son las *expresiones*, y dentro de ellas se destacan las *funciones* –de ahí la denominación que reciben estos lenguajes– entendidas como descripciones de transformaciones de información. Por su parte, en los *lenguajes orientados a objetos* la unidad principal es la de

los así llamados *objetos*, abstracciones de datos que expresan en forma conjunta la noción de estado y de posibles comportamientos. Finalmente, en los *lenguajes lógicos* la unidad principal es las llamadas *cláusulas*, afirmaciones sobre propiedades de la información, las cuales se vinculan a través de relaciones de consecuencia lógica.

Como vemos, los programas en realidad no son únicamente secuencias de comandos, sino que existe una amplísima gama de formas de expresar soluciones computacionales a problemas. Y sin embargo, seguimos insistiendo en comenzar nuestros cursos con la, obviamente incompleta, definición de *algoritmo como secuencia de instrucciones*. Hasta la RAE lo define así, aunque no debería sorprender esto, ya que la Real Academia Española solamente se encarga de recopilar las cuestiones del lenguaje en tanto el mismo es utilizado por la mayoría de las personas. En la discusión posterior (Sección 6), veremos una propuesta de cómo se puede reemplazar de forma simple esta noción.

Por otra parte, otro problema asociado al uso de ejemplos de índole cotidiana para presentar la noción de algoritmo, como ser el "algoritmo para cebar mate", o el "algoritmo para cambiar la rueda de un auto", adolecen de las características adecuadas para poder transmitir adecuadamente las nociones buscadas. En particular, como se intenta trabajar sobre ejemplos cotidianos, se omite el hecho fundamental de que para programar debe existir un conjunto *a priori* de comandos y/o expresiones primitivas que deben ser conocidas o descubiertas para poder establecer las soluciones buscadas en forma precisa. Así, es usual ver que muchos estudiantes iniciales tienen dificultades en entender por qué no pueden simplemente decirle a la computadora "cebá un mate", y deben imaginarse ellos mismos un conjunto impreciso de tales instrucciones, lo que produce confusión. Este problema es de fácil solución si se establece un lenguaje *a priori*, tal como se propone en los cursos de la Iniciativa Program.AR (Factorovich y Sawady O'Connor, 2015), o en aquellos que contemplan la utilización de un lenguaje previamente definido (Martínez López, 2013; Colussi y Viale, 2019; Martínez López et.al. 2019; Colussi y Viale, 2021); en ese caso estaremos más lejos de la noción de algoritmo, y más cerca de la noción de programa, algo deseable, como veremos en la discusión posterior (Sección 6).

Esta insistencia en definir algoritmo y programa como secuencia de pasos se explica en parte por la supremacía de la secuencia de instrucciones y los lenguajes imperativos en la historia de la programación, debido a que las computadoras modernas siguen aún utilizando el modelo von Neumann y también por la inercia en nuestros sistemas educativos para trascender los modelos con los que aprendimos. Pero también hemos visto que una mirada más detallada sobre las formas de expresar los programas muestran que la secuencia es apenas una de las formas de combinación y expresión de cómputo utilizadas. Incluso en los lenguajes imperativos más elementales existen formas de alterar el orden secuencial, dando lugar a bifurcaciones y ciclos en el bajo nivel, que en las versiones de alto nivel que proveen los lenguajes estructurados se vuelven alternativas y repeticiones. E incluso si solamente pensamos en la secuencia, ¿en qué nivel de abstracción estamos pensando? ¿Pensamos los comandos simplemente como comandos primitivos? ¿O también incluimos en ellos los "comandos definidos por el usuario" mediante procedimientos? Y en este último caso, ¿cómo damos cuenta de esos procedimientos? En el Apéndice A se ofrece un ejemplo de la dificultad de entender una solución simple a un problema no muy complejo, si se la piensa solamente como secuencia de instrucciones primitivas (dando por sentado que estas instrucciones son las de la máquina que queremos programar).

¿Qué consecuencias tiene esta definición básica en la didáctica de la programación? ¿Qué implica esto acerca de la concepción de la programación? Al enseñar que programar es simplemente secuenciar instrucciones estamos recortando fuertemente las posibilidades de lo que podemos enseñar sin forzar las ideas, y sesgando la educación hacia la programación imperativa. E incluso en ese caso, la secuencia de pasos no alcanza para explicar las alternativas y repeticiones que rápidamente le dan riqueza expresiva a los programas. Se pasa de un concepto incompleto a otro que hace foco en el lado menos deseable de la naturaleza de los programas: el flujo de control. Es consecuencia de

pensar los programas en términos de la arquitectura von Neumann y se vincula con concentrarse más en la secuencia de instrucciones, generalizada a flujo de control de instrucciones, en lugar de poner el foco en la manera en que el programa describe la solución en términos de ideas inteligibles para las personas. El modelo de cómputo que rige nuestra concepción de la programación es, así, el mencionado modelo von Neumann, aunque existen, como mencionamos, numerosos otros modelos de cómputo con características muy diversas, todos equivalentes[1] (Kleene, 1952), y de los que sería deseable explorar y conocer al menos algunos para entender realmente las nociones de programación con mayor profundidad. Se hace, por tanto, necesario cuestionar el enfoque de flujo de control tanto como el de cuestionar la secuencia de pasos. Otras consecuencias menos obvias, más específicas, se discuten en el Apéndice A.

Como si esto fuera poco, en la comunidad de computación cuando se habla de algoritmos no nos referimos a la secuencia de pasos. ¿Cuál es la secuencia de pasos en el algoritmo para dar sugerencias de contenido en los tanques de streaming de vídeo o audio, o en redes sociales? ¿Y un algoritmo paralelo, acaso no es un algoritmo por no establecer la secuencia precisa de pasos?

Del "algoritmo para cebar mate" a los algoritmos de aprendizaje automático o los algoritmos de tratamiento paralelo de información hay un mundo silenciado, que estamos dejando afuera a la hora de enseñar computación desde este ángulo particular, limitado.

3. Las expresiones: el gemelo subvalorado del universo de los comandos

Como vimos, al programar, lo más importante es la capacidad de describir la solución de un problema, y esta descripción no siempre se focaliza en la secuencia de pasos. Todos los lenguajes imperativos poseen elementos que van más allá de esta secuencia, aunque es usual que los aprendamos y usemos sin reflexionar sobre ellos. Nos referimos por supuesto a las *expresiones*, que se utilizan para describir datos en los programas. Salvo los primeros programas muy simples, todos los programas manejan alguna forma de expresiones para describir información. Usualmente esta información toma la forma de Números, por ser la más común y difundida, pero también pueden encontrarse otros tipos simples, como Direcciones (en Pilas Bloques y Gobstones), Colores (en Scratch, App Inventor y Gobstones), Booleanos (en casi todos los entornos), Strings (en Scratch, Gobstones, App Inventor, Python), etc. También aparecen formas simples de hablar de información no conocida o que varía según el entorno, tales como sensores, parámetros y variables, y con estas formas aparece la necesidad de tener operadores para combinar diferentes expresiones en descripciones más complejas de información (tales como sumas y otras operaciones numéricas, operaciones lógicas y otras). Tales expresiones no es necesario explicarlas en base a un flujo de control o una secuencia de pasos, sino que es mejor pensarlas en tanto la combinación de elementos y la (transformación de) información que expresan.

El mundo de las expresiones está presente casi desde el principio del aprendizaje de programación, y sin embargo no le damos el lugar que merece. Este mundo es tan rico y complejo como, e incluso más aún que, el de los comandos. Así como en el mundo de los comandos encontramos secuencias, alternativas, repeticiones y capacidad de definir

[1]Existe una hipótesis en la teoría de computabilidad, conocida como la **Tesis de Church-Turing** (Kleene, 1952), que sostiene que todo modelo de cómputo que pueda ser considerado como "computación efectiva" es equivalente a algunos de los modelos clásicos definidos en las décadas de 1930 y posteriores (máquinas de Turing, lambda-cálculo, funciones recursivas generales, etc.). Esta idea indemostrable, y su generalización, es que todos los modelos de cómputo suficientemente generales tienen el mismo poder expresivo, y por lo tanto cualquiera de ellos puede usarse para estudiar la noción de "computación". Al día de hoy existe un consenso extremadamente difundido sobre la validez de esta "tesis", y todos los modelos adecuados que se han propuesto, han demostrado no contradecirla. Para más detalles sobre la historia y naturaleza de esta tesis, ver el sitio de Wikipedia en inglés (la versión castellana carece de suficientes detalles): https://en.wikipedia.org/wiki/Church%E2%80%93Turing_thesis.

nuestros propios comandos a través de procedimientos, en el mundo de las expresiones existen formas que podrían homologarse a secuencias (las tuplas, registros, o tipos producto), a alternativas (los poco conocidos variantes, o tipos suma), a repeticiones (las listas o colecciones), y también la capacidad de definir nuestras propias expresiones (las funciones).

En la programación de índole más profesional normalmente las expresiones aparecen entremezcladas con los comandos (por ejemplo, las "funciones" en C son a la vez expresiones y comandos), o desdibujadas por las formas de representación de bajo nivel utilizadas (ver la Sección 4). Sin embargo, a la hora de enseñarlas, resulta importante identificarlas en forma pura, para que los conceptos a transmitir aparezcan en forma clara y directa, y así los estudiantes puedan identificarlos e incorporarlos a su concepción de la programación (Reynolds, 1998).

Resulta coherente con esta invisibilización de las expresiones que muy pocos cursos introductorios ofrezcan caminos para continuar profundizando en las expresiones a través de las formas más complejas de las mismas y estructuras de datos simples. Y en muchos de los casos en los que aparece alguna forma de expresiones más complejas, se evidencia una falta de reflexión sobre la verdadera naturaleza de los conceptos trabajados. Tal falta se comprueba por ejemplo en el intento de distinguir diferentes tipos de datos a través de sus formas (en Scratch), mecanismo que no es escalable y generalizable para tipos y estructuras de datos más complejas.

Otra evidencia sobre la falta de reflexión sobre la relevancia e importancia del mundo de las expresiones se puede ver en las formas clásicas de presentar un concepto complejo: la recursión. En la gran mayoría de los cursos iniciales que abordan este concepto, el mismo se aborda desde su manifestación en el mundo de los comandos en lugar de comenzar con su presentación en el mundo de las expresiones. Sin embargo, es notablemente más difícil de comprender la recursión cuando se la combina con la noción de efecto y con la secuencia de instrucciones, pues esta forma compleja de repetición tiene una naturaleza de muchísimo más alto nivel, generando interacciones que resulta complicado de comprender sin haber comprendido primero el concepto en forma independiente. En el mundo de las expresiones puras, donde las expresiones no producen efectos como los comandos, la recursión es una forma clara y efectiva, de alto nivel, para expresar y transformar datos complejos. Es por tanto necesario explicitar estas nociones si queremos refinar nuestras prácticas didácticas de forma tal de mejorar nuestro trabajo como docentes.

Hemos visto una de las caras de la programación que necesitamos explicitar a la hora de enseñar a programar. Pero aún debemos considerar otras más.

4. La representación de información: protagonista escondida de todo programa

Un concepto central en computación es el concepto de representación de la información. Toda, *absolutamente toda* la información que maneja un programa está representada de una forma u otra. Sorprendentemente, en los cursos iniciales de programación este concepto central no aparece en absoluto. Y en cursos más avanzados suele abordarse mediante ejemplos específicos, sin explicitarlo y darle un lugar central.

Otro sesgo usual en el tratamiento de representación de información es que cuando se aborda, se lo hace como si fuese un tema separado de las nociones básicas de programación, y usualmente a través de casos específicos, tomados de usos del mundo real, y por ende extremadamente complejos. Por ejemplo, en cursos de organización de computadoras se trabaja la manera de representar números en notación binaria y hexadecimal, sin motivar adecuadamente la necesidad de

estas formas específicas de representación, ni trabajar su vinculación con los temas dados en los cursos de programación. En cursos de estructuras de datos se trabaja la representación de estructuras abstractas –como diccionarios, heaps y otras– a través de elementos de más bajo nivel –como arreglos, memoria dinámica y punteros u otras. Pero rara vez se explica que estos temas son casos particulares de un concepto profundo y central en computación, que es el tópico de esta sección: la *representación de información*.

En la mayoría de los cursos modernos utilizados para enseñar a programar se utilizan elementos de apariencia concreta, como personajes antropomorfizados para ejercer ciertas acciones (un gato, un perro u otros animales, un robot, un marciano, personajes de historias clásicas y otros), puesto que resultan más cercanos y concretos a los estudiantes iniciales, y resulta adecuado comenzar por ahí. Sin embargo, estos elementos solamente sirven como inicio, ya que resulta complejo luego utilizarlos para representar otros elementos, y así trabajar las nociones de representación de información a través de ellos. Algo similar sucede cuando los elementos utilizados son realmente concretos, como sucede al utilizar robots u otros elementos de *hardware*. En forma consecuente con el hecho de que la representación de información no se explicita, estos cursos no realizan ninguna propuesta hacia el reemplazo de los personajes iniciales o elementos concretos por otros de mayor nivel de abstracción, como estructuras de datos, números u otras, y así el salto cognitivo necesario para pasar de dichos cursos iniciales a cursos más avanzados es grande, y puede ser negativo para muchos estudiantes.

Desde el punto de vista didáctico es conveniente que la curva de aprendizaje sea suave, y para lograrlo, debe comenzarse con ejemplos sencillos, donde tanto los elementos a representar como los de representación sean de naturaleza (aparentemente) concreta, y trabajar el concepto de representación de información para explicitar sus características. Por ejemplo, puede utilizarse un palo como elemento de representación, y utilizarlo para representar una trompeta, un bastón o una espada como elementos representados. Es claro que el palo *no es* una trompeta, pero cualquier niño puede entender esta forma de representación como un juego. Luego se pueden utilizar elementos de apariencia concreta en la computadora, y representar una manzana o un globo rojo a través de una bolita de ese color. Un ejemplo más complejo, pero de índole más cotidiana puede venir de mostrar que un vídeo está representado a través de una secuencia de imágenes, y desde allí comenzar a trabajar en formas más complejas, tales como la representación de una imagen a través de puntos de colores, luego de los colores a través de grupos de números, y finalmente llegar a la representación de los números como secuencias de bits. De esta forma se puede vincular la idea de representar un vídeo con la de una secuencia de bits, pero no antes de haber trabajado el concepto en forma independiente a la representación compleja.

Otra forma en la que se manifiesta la representación de información es en las implementaciones de conceptos de alto nivel en términos de conceptos de bajo nivel. Si bien la implementación NO es el concepto implementado, sino una forma de representarlo, muchas veces las ideas de alto nivel terminan siendo explicadas a través de su implementación, en lugar de en base a su naturaleza original de alto nivel. A los estudiantes les suele resultar complejo distinguir con claridad que no están trabajando con el bajo nivel *per se*, sino con una representación de otra información, y por lo tanto suelen "aprovechar" operaciones del bajo nivel, desdibujando así los conceptos representados. Un ejemplo de esto puede encontrarse en la manera de explicar una función a través de la idea de modificación del flujo de control combinada con alguna convención para comunicar los datos a describir (posiciones de memoria específica, por ejemplo). Esta forma de entender las funciones en base a la ejecución en una máquina von Neumann complejiza de forma seria la capacidad de comprender a la función en tanto *transformación de información*, y como resultado muchos programadores profesionales tienen serias dificultades para comprender lenguajes funcionales puros, por

ejemplo. En el caso del ejemplo simple, esta idea sería mezclar la "trompeta" con el palo, y utilizar el palo para hacer palanca, ignorando que en realidad la "trompeta" no podría ser utilizada para tal fin. Cuando alguien no sabe lo que es exactamente una trompeta, y solamente percibe al palo como elemento, no logra comprender las características reales de la trompeta mirando solamente al palo. Es nuestra convicción que un curso de introducción a la programación debe abordar este tema tan importante de forma temprana. Ampliamos esta idea en la discusión posterior (Sección 6).

Creemos que es importante que un tema tan central como la representación de información debe dejar de estar escondido e invisibilizado en nuestros currículos, y pasar a integrar una posición central en las mismas, junto con los demás temas centrales, ya desarrollados.

5. Otros temas avanzados: una cuestión de estado

Existen algunos otros temas de mayor complejidad técnica que también podrían considerarse fundamentales al enseñar programación. En general, son temas vinculados al manejo de memoria y a las nociones de estado –e identidad, en el caso de la programación orientada a objetos. Podemos mencionar la utilización de variables globales, estructuras de datos mutables (como arreglos o estructuras dinámicas), y nociones de memoria estática o dinámica, entre otros. Asociados a esos temas también puede considerarse el análisis de complejidad, y las ventajas y problemas que trae aparejado el manejo de la memoria –usualmente mayor eficiencia, a costa de la expresividad y de mayor cantidad de errores sutiles, de difícil detección.

Sin embargo, estos temas pueden considerarse lo suficientemente avanzados para soslayarlos en cursos iniciales, al menos en toda su complejidad técnica. Es usual que los cursos iniciales contengan nociones de estado en el entorno de trabajo propuesto, pero esas nociones son elementales y usualmente no involucran mayores problemas didácticos, pues se asocian con los elementos concretos que integran el universo de discurso de los entornos, y resultan naturales para los estudiantes (considerar, por ejemplo, actividades de Pilas Bloques tales como las de Lita haciendo su ensalada –no puede hacer la ensalada si primero no cambia su estado recolectando los ingredientes). Es nuestra convicción también que no es necesario comenzar con estos temas, y que de hacerlo, solamente incorporamos una complejidad que dificultará los pasos iniciales de los estudiantes.

En el caso de considerar el tratamiento inicial de estructuras de datos, posiblemente la única excepción a lo establecido en el párrafo anterior, existen formas de enseñarlas sin recurrir a modificaciones de estado, a través de formas inmutables, pudiendo dejar los temas de mutabilidad y las formas de tratarla, y de complejidad para cursos más avanzados. Existen muchas variaciones posibles para el dictado de tales cursos, pero su consideración excede el propósito de este artículo.

6. Discusión: el balance nos hará capaces

Como vimos, la programación consiste en muchas formas distintas de expresar soluciones a problemas y representar información. Y también vimos que creemos que es importante que estas distintas formas estén presentes, o al menos conceptualmente cercanas, desde el comienzo de un curso de programación inicial. ¿Cómo hacerlo de forma integrada con las prácticas actuales que tenemos? ¿Es necesario descartar todo y comenzar de nuevo? ¿O podemos adaptar nuestros contenidos para ampliarlos e incluir todos los temas que consideramos importantes?

Los cursos actuales a nivel secundario comienzan con elementos concretos en el universo de discurso, porque los mismos resultan más accesibles para el estudiante inicial, sin demasiada formación matemática, que las formas más abstractas utilizadas en la programación profesional. De la misma forma, los comandos resultan más concretos que las expresiones, pues sus efectos son apreciables sobre los elementos del universo de discurso, a diferencia de las modificaciones sobre datos de naturaleza más abstracta, que resulta difícil de visualizar. Sin embargo, es importante entender que toda la programación es de naturaleza abstracta, y por tanto *estas formas concretas deben ser utilizadas solamente como punto de partida, y servir como base para la construcción del conocimiento necesario para realizar un pasaje gradual a los elementos abstractos*. Citando a un gran matemático y científico de la computación, un personaje de los más influyentes en el desarrollo de las Ciencias de la Computación, Edsger W. Dijkstra (Dijkstra, 1988):

> *"La moraleja de la historia es: para tratar con todos los elementos de un conjunto debe ignorárselos, y trabajar con la definición del conjunto."*

Para lograr este pasaje, además de la propuesta actual de cursos como los de la Fundación Sadosky (Factorovich et.al., 2015; Klinkovich y Czemerinski eds, 2018) o los basados en Gobstones (Martínez López et.al. 2012; Martínez López et.al., 2017), que comienzan con el uso de procedimientos para expresar acciones de alto nivel, creemos que es necesario comenzar a incluir el manejo de expresiones básicas en forma explícita desde temprano, e ir complejizándolas al mismo tiempo que las formas de combinar comandos, balanceando los conceptos que se aprenden en ambas formas de expresar ideas, y comparando los diferentes mecanismos de ambos para reforzar sus similitudes, como se discutió en la Sección 3.

Asimismo, resulta importante comenzar también en forma temprana con nociones elementales de representación de información, proveyendo una curva suave de aprendizaje. Para eso es necesario proveer alguna forma de transición desde los elementos concretos iniciales hacia otras formas más abstractas. Un posible abordaje al tema de representar información es comenzar con elementos concretos que no estén antropomorfizados ni sean realmente concretos, y utilizarlos como vehículos para representar otros elementos. Así, podría inicialmente soslayarse la presencia de la representación de información, pero la misma podría hacerse presente en cualquier momento del curso sin representar un salto disruptivo en complejidad. Y luego podrían irse trasladando gradualmente estas nociones de representar elementos concretos antropomorfizados mediante otros también concretos pero más maleables, a representaciones que utilicen elementos más abstractos, tales como estructuras de datos elementales (e.g. registros, listas, variantes), y así habilitar un pasaje suave a temas más avanzados tales como los que se abordan en materias como estructuras de datos, o modelización mediante objetos, nociones de compresión de información, encriptación y otros relacionados. Estos conceptos pueden integrarse con pocas modificaciones a los cursos basados en comandos, aunque para el tema del tratamiento de la representación de información, tal como se dijo, es necesario contar con elementos no antropomorfizados en el universo de discurso para poder representar con ellos otros elementos.

La propuesta que acercamos, por tanto, incluye *descartar* la definición de *algoritmo como secuencia de pasos* como elemento introductorio, y focalizarse en la noción de ***programa***, en tanto *descripción de la solución a un problema computacional que es ejecutable por algún mecanismo de cómputo automático* (Martínez López, 2013 ; Martínez López et.al., 2019). Esta forma de concebir a los programas como descripciones, en lugar de como secuencia de pasos, habilita a la incorporación de diversas otras formas y paradigmas, tales como las discutidas. Por otra parte, la transición de la noción más abstracta y elusiva de algoritmo a la noción más concreta y definida de programa va en concordancia con la búsqueda de comenzar la didáctica con elementos concretos para construir la visión abstracta

deseada. La búsqueda de trascender la secuencia de comandos como forma básica de concebir la programación se ha explorado desde otros ángulos también. Por ejemplo, existen cursos iniciales de programación basados en lenguajes funcionales, tanto a nivel universitario (Felleisen et.al., 2001; Bloch, 2010; Felleisen et.al., 2015; Colussi y Viale, 2019; Colussi y Viale, 2021) como a nivel secundario (Felleisen et.al., 2013) y estos cursos reportan diversos grados de éxito. Si bien son una propuesta interesante, son bastante diferentes en su presentación y herramientas con respecto a los que se vienen usando de manera masiva a nivel nacional. Por eso creemos que es valioso establecer otros enfoques que permitan ampliar la búsqueda de una forma más rica de transmitir las nociones iniciales de programación.

En nuestro país, el enfoque que proponemos viene usándose con éxito a nivel universitario (Martínez López, 2013) en los cursos iniciales de programación de la Universidad Nacional de Quilmes y la Universidad Nacional de Hurlingham, y a nivel secundario, a través del manual del 1er ciclo de secundaria de la Fundación Sadosky (Martínez López et.al., 2019) de reciente publicación, y en cursos de la Iniciativa Program.AR (Martínez López et.al., 2019b), pero todavía no contamos con experiencias suficientes de su implementación como para poder llegar a conclusiones sobre el mismo. Sin embargo, creemos que la discusión derivada de esta propuesta es extremadamente necesaria.

7. Conclusiones

La generalización en la enseñanza de la programación y las ciencias de la computación son un gran avance, del que nuestro país no es ajeno. El trabajo realizado desde los equipos nacionales es de una envergadura y calidad que ponen a la Argentina en una posición destacada en temas de didáctica de Ciencias de la Computación. Por otra parte, el espacio de discusión y difusión dado por las Jornadas de Didáctica de Ciencias de la Computación (antes, Jornadas de Didáctica de la Programación) es un ámbito ideal para la mejora de las prácticas y las ideas sobre las que se funda la práctica cotidiana. En ese sentido, este trabajo se integra a esa discusión en pos de la mejora de ideas y prácticas cotidianas, con el fin de ampliar la concepción imperante en didáctica de la programación, y de esa forma lograr una experiencia completa y adecuada para dar las bases necesarias tanto para la comprensión del mundo que nos rodea, como para habilitar el estudio posterior de la programación con índole profesional.

La discusión sobre los posibles enfoques para trascender la secuencia de comandos y abarcar las formas más diversas y ricas de la programación en general es algo que debemos promover para avanzar en la mejor formación para nuestros ciudadanos. En esta reflexión continua sobre lo que hacemos, y en el cuestionamiento de lo que hacemos para fortalecerlo y mejorarlo en pos de conseguir mejor los objetivos planteados está la semilla del éxito que podemos tener en esta tarea tan gratificante que es educar.

Agradecimientos

No puedo dejar de agradecerle a Gabriel Baum por inspirarme, empujarme y motivarme en este apasionante tema que es aprender a enseñar nuestra disciplina. También debo agradecer a todas las personas con las que he tenido discusiones a lo largo de los años y que me han ayudado a clarificar, enriquecer y profundizar mi comprensión y mi didáctica, demasiados para nombrarlos en forma individual. Solamente voy a mencionar en especial a Alfredo Sanzo y a Marcos Gómez, que fueron los puntos de apoyo finales que me ayudaron a poner estas ideas en un texto accesible. Finalmente, debo mi gratitud a la Fundación Sadosky, a la iniciativa Program.AR y a su principal artífice, Fernando Schapachnik, por proveer un marco significativo donde estas ideas pueden florecer y dar verdaderos frutos

para contribuir a la educación nacional y a la construcción de soberanía tecnológica. Es un orgullo y un aliciente saber que uno no está solo en esta tarea de hacer ciencia útil desde y para nuestro país.

Bibliografía

Bloch, S. (2010). Picturing Programs. An Introduction to Computer Programming. College Publications.

Church, A. (1936). An Unsolvable Problem of Elementary Number Theory. The American Journal of Mathematics 58(2):345-363. JSTOR 2371045.

Colussi, N. y Viale, P. (2019). Actividades de Programación Grupales para Primer año de la Licenciatura en Ciencias de la Computación. Experiencias Didácticas en el Aula. Actas de XIII Jornadas de Ciencia y Técnica, Pairoba, C., Cricco, J. y Rius, S. (comp), pp.196. Universidad Nacional de Rosario.

Colussi, N. y Viale, P. (2021). Prácticas de Programación Grupales en el Aula. Estrategias Didácticas para el desarrollo del Pensamiento Computacional en los Primeros Cursos de Programación del Ciclo Inicial Universitario. Actas de XXIII Workshop de Investigadores en Ciencias de la Computación, Fratti, F. y Carmona, F. (comp), pp.377–381. Universidad Nacional de Chilecito.

Dahl, O.-J. (1963). The SIMULA storage allocation scheme. Technical Report of the Royal Norwegian Council for Scientific and Industrial Research, NCC Doc. 162.

Dahl, O.-J. and Nygaard, K. (1966). SIMULA: an ALGOL-based simulation language. Communications of ACM, 9(9):671-678. ACM.

Dijkstra, E.W. (1968). Go to statement considered harmful. Communications of the ACM. 11(3):147–148. ACM.

Dijkstra, E.W. (1988). On the cruelty of really teaching computing science. E.W.Dijkstra Archive. Center for American History, University of Texas at Austin.

Factorovich, P.M. y Sawady O'Connor, F.A. (2015). Actividades para aprender a Program.AR. Fundación Sadosky. ISBN: 978-987-27-4161-7.

Felleisen, M., Findler, R.B., Flatt, M., y Krishnamurti, S. (2001). How to Design Programs. An Introduction to Computing and Programming. The MIT Press, Cambridge, Massachusetts, London, England.

Felleisen, M., Van Horn, D., Barski, C. y eight students of Northeastern University (2013). Learn to Program, One Game at a Time! No Starch Press, p.312. ISBN: 9781593274917.

Felleisen, M., Findler, R. B., Flatt, M., Krishnamurthi, S., Barzilay, E., McCarthy, J., y Tobin-Hochstadt, S. (2015). The Racket manifesto. In 1st Summit on Advances in Prog. Langs., p.113–128. Schloss Dagstuhl-Leibniz-Zentrum für Informatik.

Godfrey, M.D. and Hendry, D. F. (1993). The computer as von Neumann Planned it. IEEE Annals of the History of Computing,15(1). IEEE.

Gordon, M.J.C., Milner, R., Morris, L., Newey, M.C., Wadsworth, P. (1978). A Metalanguage for Interactive Proof in LCF. POPL '78: Procs. of 5th ACM SIGACT-SIGPLAN Symp. on Principles of prog. langs., pp.119–130. ACM.

Kleene, S.C. (1952). Introduction to Metamathematics. North-Holland. OCLC 523942.

Klinkovich, V. y Czemerinski, H. (2018). Ciclo de manuales: Ciencias de la Computación para el Aula.: Manual para docentes. Fundación Sadosky.

Martínez López, P. E., Bonelli, E. A., Sawady, F. A. (2012). El nombre verdadero de la programación. Una concepción de la enseñanza de la programación para la sociedad de la información. En Anales de SSI'12, dentro de las JAIIO'12, pp.1–23, septiembre 2012. ISSN 1850-2830.

Martínez López, P. E. (2013). Las bases conceptuales de la programación. Una nueva forma de aprender a programar. EBook (1era ed.). La Plata, el autor. ISBN: 978-987-33-4081-9. http://www.gobstones.org/bibliografia/Libros/BasesConceptualesProg.pdf.

Martínez López, P. E., Ciolek, D., Arévalo, G. y Pari, D. (2017). The GOBSTONES method for teaching computer programming. SIESC'17 dentro del CLEI'17, pp.1–9. IEEE, ISBN 978-1-5386-3057-0.

Martínez López, P.E., Aloi, F., Ciolek, D.A., Martínez, F., Pari, D., y Tobia, P. (2019). Ciencias de la Computación para el aula. Manual para docentes. 1er Ciclo de Secundaria (Klinkovich, V. y Czemerinski, H. eds. 3). Fundación Sadosky. ISBN: 978-987-27416-7-9.

Martínez López, P.E., Sanzo A., Schapachnik, F. (2019b). Hacia una didáctica de la programación para la secundaria argentina. Actas II Jornadas Argentinas de Didáctica de la Programación (JADiPro II). EBook, Acosta et.al. (editores) pp.88–99. UNRC.

McCarthy, J. (1958). Programs with common sense. Procs. of the Symposium on the Mechanization of Thought Processes, National Physiology Lab, England.

McCarthy, J. (1960). Recursive Functions of Symbolic Expressions and their Computation by Machine, part I. CACM, 3(4):184–195, ACM.

Milner, R. (1978). A theory of type polymorphism in programming. Journal of Computer and System Sciences, 17(3):348–375. Reynolds, J. (1998). Theories of Programming Languages, Cambridge University Press. ISBN 0-521-59414-6.

Sanzo, A., Schapachnik, F., Factorovich, P., y Sawady O'Connor, F. A. (2017). Pilas Bloques: A scenario-based children learning platform. 2017 Twelfth Latin American Conference on Learning Technologies (LACLO), pp.1–6. IEEE.

Turing, A.M. (1936). On Computable Numbers, with an Application to the Entscheidungsproblem. Procs. of London Math.Soc. 2-42(1):230–265.

Turing, A.M. (1938). On Computable Numbers, with an Application to the Entscheidungsproblem: A correction. Procs. of London Math.Soc. 2-43(6):544–6.

von Neumann, J. (1945). First Draft of a report on the EDVAC. Technical report of Moore School of Electrical Engineering, University of Pennsylvania.

A. El sinsentido de un algoritmo visto como secuencia de pasos

Para evidenciar la falta de riqueza de la concepción de algoritmo como secuencia de instrucciones, consideremos el problema de lograr que un autómata recorra un camino simple (sin bifurcaciones ni ciclos) indicado con cruces en el piso. Para esto necesitamos saber que las operaciones posibles del autómata son moverse un paso hacia adelante, girar a izquierda o derecha 90 grados y determinar si en su posición actual hay o no una cruz. Para simplificar, supongamos que las cruces en el camino están ubicadas a exactamente un paso de distancia una de las otras, ya sea hacia adelante, a la izquierda o a la derecha, y que, como dijimos, no hay bifurcaciones ni ciclos en el camino determinado por ellas. Una solución sencilla podría ser la siguiente:

```
Determinar la dirección de la próxima cruz, si existe
Repetir hasta que no haya más cruces
  Avanzar en la dirección de la cruz encontrada
  Determinar la dirección de la próxima cruz, si existe
```

Sin embargo, esta solución simple no está expresada en términos de instrucciones individuales al autómata. Necesitamos aclarar cómo expresar la tarea de determinar la dirección de la siguiente cruz en términos de dichas operaciones. Esto podría hacerse de la siguiente manera:

```
Elegir una de las siguientes alternativas
```

```
Informar izquierda cuando hay una cruz hacia la izquierda
Informar adelante cuando hay una cruz hacia adelante
Informar derecha cuando hay una cruz hacia la derecha
Informar que no hay cruces en otros casos
```

A su vez, cada una de estas instrucciones para informar una dirección requiere un refinamiento mayor. En la Figura 1 se recopilan todas las partes de este recorrido de un camino simple, y se completan los faltantes.

Ahora bien, ¿cuál es la secuencia de instrucciones de estos algoritmos? No están expresados solamente en términos de secuencia, sino también de repeticiones, de alternativas y de abstracciones. Y entonces hay un salto conceptual no explicitado desde la definición de algoritmo como secuencia de pasos y la comprensión de este algoritmo particular, que podríamos llamar un recorrido simple sobre las cruces del camino. Incluso, la cantidad de instrucciones de la secuencia dependerá de la longitud del camino. Para más claridad, supongamos el ejemplo de un camino con 3 cruces, como el que se encuentra en la Figura 1. La secuencia completa de comandos sería la que se observa en la Figura 2, donde se agregan algunas líneas de comentarios para facilitar la identificación de los grupos de acciones individuales en términos de las operaciones de más alto nivel.

Otra consecuencia, menos obvia, de este ejemplo es que no es suficiente la especificación informal dada sobre "Informar", ni tampoco es obvio cómo representar los valores a informar (en el ejemplo, la dirección de la próxima cruz, que sería un valor posible entre `‘‘izquierda’’`, `‘‘derecha’’`, `‘‘adelante’’` y `‘‘no hay más cruces’’`) y por lo tanto su pasaje a un lenguaje de programación específico resulta difícil, especialmente para alguien que está dando sus primeros pasos en programación. Por ejemplo, en los ejemplos dados aquí se presume un lenguaje con variables globales y con funciones con efectos; sin embargo, en otros lenguajes, como Gobstones por ejemplo, la solución podría seguir esta misma línea pero ser bastante más sencilla. Entonces, para trabajar con datos y expresiones de forma clara es necesario contar con un lenguaje preciso que establezca los límites y posibilidades de este aspecto de los programas. Esta conclusión excede este ejemplo, y merece un tratamiento aparte, pero sirve para reforzar otros elementos que deberían discutirse, como lo inadecuado que resulta el uso de "pseudocódigo" para enseñar a programar.

Es claro que, incluso con la asistencia de los comentarios, es prácticamente imposible concebir y comprender este algoritmo desde la secuencia de instrucciones elementales. Además, se presumió implícitamente algunas características del manejo de expresiones del lenguaje, que no fueron clarificadas. Y finalmente, también es claro que NO lo hacemos así durante nuestros cursos. ¿Por qué entonces insistir con seguir presentando los algoritmos como secuencias de pasos? ¿No es momento de cuestionar eso, y alcanzar alguna propuesta superadora?

Algoritmo de recorrido de camino simple

```
Determinar la dirección de la próxima cruz, si existe
Repetir hasta que no haya más cruces
  Avanzar en la dirección de la cruz encontrada
  Determinar la dirección de la próxima cruz, si existe
```

Algoritmo para determinar la dirección de la próxima cruz, si existe

```
Elegir una de las siguientes alternativas
  Informar izquierda cuando hay una cruz hacia la izquierda
  Informar adelante cuando hay una cruz hacia adelante
  Informar derecha cuando hay una cruz hacia la derecha
  Informar que no hay cruces en otros casos
```

Algoritmo para determinar si hay una cruz hacia la izquierda

```
Girar a izquierda
Avanzar un paso
Determinar si hay cruz en pos.actual
Girar a izquierda
Girar a izquierda
Avanzar un paso
Girar a la izquierda
```

Algoritmo para determinar si hay una cruz hacia la derecha

```
Girar a derecha
Avanzar un paso
Determinar si hay cruz en pos.actual
Girar a izquierda
Girar a izquierda
Avanzar un paso
Girar a la derecha
```

Algoritmo para determinar si hay una cruz hacia adelante

```
Avanzar un paso
Determinar si hay cruz en pos.actual
Girar a izquierda
Girar a izquierda
Avanzar un paso
Girar a la izquierda
Girar a la izquierda
```

Estado inicial de ejemplo

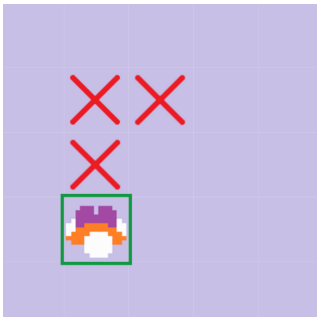

Tabla 1: Algoritmo de recorrido de camino simple y un estado inicial.

// Determinar dir. de la próx. cruz, si existe, 1	// Considerar adelante, 3
// Considerar a la izquierda, 1	Avanzar un paso
Girar a izquierda	¿Hay cruz en la posición actual?
Avanzar un paso	Girar a izquierda
¿Hay cruz en la posición actual?	Girar a izquierda
Girar a izquierda	Avanzar un paso
Girar a izquierda	Girar a la izquierda
Avanzar un paso	Girar a la izquierda
Girar a la izquierda	// Considerar a la derecha, 3
// Considerar adelante, 1	Girar a la derecha
Avanzar un paso	Avanzar un paso
¿Hay cruz en la posición actual?	¿Hay cruz en la posición actual?
Girar a izquierda	Girar a izquierda
Girar a izquierda	Girar a izquierda
Avanzar un paso	Avanzar un paso
Girar a la izquierda	Girar a la derecha
Girar a la izquierda	// Avanzar en la dir. de la prox. cruz, 3
// Avanzar en la dir. de la prox. cruz, 1	Girar a la derecha
Avanzar un paso	Avanzar un paso
// Determinar dir. de la próx.cruz, si existe, 2	// Determinar dir. de la próx. cruz, si existe, 4
// Considerar a la izquierda, 2	// Considerar a la izquierda, 4
Girar a izquierda	Girar a izquierda
Avanzar un paso	Avanzar un paso
¿Hay cruz en la posición actual?	¿Hay cruz en la posición actual?
Girar a izquierda	Girar a izquierda
Girar a izquierda	Girar a izquierda
Avanzar un paso	Avanzar un paso
Girar a la izquierda	Girar a la izquierda
// Considerar adelante, 2	// Considerar adelante, 4
Avanzar un paso	Avanzar un paso
¿Hay cruz en la posición actual?	¿Hay cruz en la posición actual?
Girar a izquierda	Girar a izquierda
Girar a izquierda	Girar a izquierda
Avanzar un paso	Avanzar un paso
Girar a la izquierda	Girar a la izquierda
Girar a la izquierda	Girar a la izquierda
// Avanzar en la dir. de la prox. cruz, 2	// Considerar a la derecha, 4
Avanzar un paso	Girar a derecha
// Determinar dir. de la próx. cruz, si existe, 3	Avanzar un paso
// Considerar a la izquierda, 3	¿Hay cruz en la posición actual?
Girar a izquierda	Girar a izquierda
Avanzar un paso	Girar a izquierda
¿Hay cruz en la posición actual?	Avanzar un paso
Girar a izquierda	Girar a la derecha
Girar a izquierda	
Avanzar un paso	
Girar a la izquierda	

Tabla 2: Secuencia de pasos del algoritmo de la Figura 1 con el ejemplo dado.

SESIÓN 6:
ESTRATEGIAS Y ENSEÑANZA

Moderadora: *Dra. Cecilia Martínez (UNC, CONICET)*

Prácticas de enseñanza de la Informática: Procesos de resignificación de los saberes pedagógicos y didácticos de profesionales - estudiantes del profesorado de Educación Secundaria en Informática

Silvina Cuello, Verónica Pacheco y Natalia Zalazar

Aplicaciones del Pensamiento Computacional en Problemas de Química

Pamela Viale, Claudia Deco y Cristina Bender

Reflexiones sobre el diseño e implementación de una especialización docente en enseñanza de la programación

Araceli Acosta, Cristián Rojo, Débero Cingolani y David Araya

Prácticas de enseñanza de la Informática: Procesos de resignificación de los saberes pedagógicos y didácticos de profesionales - estudiantes del profesorado de Educación Secundaria en Informática

Silvina Cuello
scuello307@isep-cba.edu.ar
ISEP

Verónica Pacheco
vpacheco@isep-cba.edu.ar
ISEP

Natalia Zalazar
nzalazar@isep-cba.edu.ar
ISEP

Resumen

En este trabajo, desarrollado en el marco del grupo de estudio en Educación y Tecnologías Digitales del Instituto Superior de Estudios Pedagógicos de la provincia de Córdoba, se presentan los primeros avances en torno a los procesos de resignificación sobre las prácticas de enseñanza de la Informática de profesionales - estudiantes del Profesorado de Educación Secundaria en Informática.

Este profesorado, de modalidad combinada, forma parte del programa de formación docente complementaria que ofrece el Ministerio de Educación de la Provincia de Córdoba. Está orientado a profesionales, que cuentan con una variada formación de base[1] y se desempeñan o aspiran a desempeñarse en el ejercicio de la docencia en espacios curriculares específicos del área de informática. Una de las finalidades principales es aportar profesionalismo a la tarea docente, lo que exige pensar, revisar y tensionar la formación fuertemente técnica con la que ingresan, así como las concepciones que se tienen acerca de qué es ser docente de Informática en la educación secundaria hoy.

En este marco, nos interesa indagar acerca de la construcción de saberes pedagógicos y didácticos específicos de la informática que realizan los profesionales y los procesos de resignificación/reconfiguración de sus ideas acerca de las prácticas de enseñanza.

Para ello se lleva a cabo un estudio de carácter exploratorio y descriptivo. En esta instancia presentamos los primeros avances a partir de la observación de las producciones de los cursantes -actividades y trabajos de acreditación- de los espacios curriculares de las prácticas (Práctica Docente I, Práctica Docente II y Residencia Profesional Docente) y de los módulos de las didácticas específicas (Didáctica de la informática y La informática como objeto de enseñanza). Se complementa esta exploración con la observación de registros y devoluciones de tutores en los módulos mencionados, así como los registros de reuniones de equipos de seguimiento en instancias de evaluación de cursado y también, los reportes

[1] Graduados de nivel Superior, con título habilitante reconocido por la Junta de Clasificación de Nivel Secundario de la provincia de Córdoba para la enseñanza de la Informática.

elaborados por el Área de Evaluación del Instituto en base a los resultados de las encuestas que los y las cursantes responden al finalizar cada módulo.

Palabras clave: formación docente, didáctica de la informática, resignificación de concepciones, prácticas de enseñanza.

1. Presentación

El Profesorado bajo estudio, que dio apertura en 2018, se creó con la finalidad de aportar a la profesionalidad docente. Desde los distintos módulos[2] (11 en total) se intenta propiciar la construcción de un sistema de saberes de diverso orden, que nutra el análisis de la tarea cotidiana del docente y de la realidad educativa y escolar en la que se inserta -en un contexto global, local e institucional- reconociendo sus complejidades, las herencias de una configuración histórica así como los desafíos que plantean las políticas educativas actuales. Primordialmente, se pretende favorecer la construcción de propuestas de intervención docente situadas. Este proceso requiere la movilización de las formas de pensar, de sentir y de actuar que se han formado en la propia experiencia como estudiantes y como docentes.

En cuanto a los **saberes de orden pedagógico general**, éstos aportan a la construcción de un marco interpretativo integral sobre la educación en distintas dimensiones, poniendo el foco en los procesos que influyen en las dinámicas del conocimiento y el aprendizaje. La idea es acercarles una perspectiva más compleja acerca de la tarea de transmisión cultural, de la que se ocupa la educación en general y de qué manera este proceso de transmisión adquiere determinadas particularidades a partir del formato escolar que todos conocemos.

Respecto de los **saberes de orden didáctico específico**, permiten articular el saber disciplinar con el sentido educativo que asume la enseñanza de ese saber en la educación secundaria. Esto implica la sistematización de métodos y estrategias de enseñanza generales y particulares, en el marco de la Informática. (Propuesta Académica del Profesorado, 2018)

Por su parte, la **propuesta de cursada** se desarrolla en modalidad combinada, las clases virtuales asincrónicas que se implementan en aulas virtuales, se entretejen con encuentros presenciales sincrónicos[3], que se realizan en los Institutos de Formación Docente asociados de diferentes localidades de la geografía provincial; con trabajos de campo y con prácticas que se realizan en las escuelas asociadas. Cada grupo de cursantes está a cargo de un/a **profesor/a tutor/a** que asume la responsabilidad pedagógica en el recibimiento, apoyo en la dificultades, en el acompañamiento virtual y el desarrollo de las clases presenciales, posibilitando una mirada que reconoce los propios ritmos y posibilita el encuentro cara a cara con los/as cursantes.

En cuanto a las **clases virtuales**, si bien la escritura está a cargo de docentes especialistas en la materia, éstas se producen, se tallan, se re-escriben con las sugerencias y propuestas que hace el equipo de Producción de Materiales y los colegas del Departamento de Enseñanza en Tecnologìas Digitales e Informática. Así, las clases están conformadas por una pluralidad de voces, de “decires” que comunican de manera multimodal. En este sentido, decimos que las clases son una obra –nos gusta la imagen del artesano de Sennet (2009)– que resulta de una producción singular y

[2]Módulo Introductorio, Perspectivas sobre la Educación, Procesos de Aprendizaje y Procesos de Enseñanza, Bases Didácticas de la Educación Secundaria, Sistema educativo y problemáticas de la Educación Secundaria, Práctica Docente I, La informática como objeto de enseñanza, Didáctica de la Informática, Práctica Docente II, La Enseñanza de la Informática en la Educación Secundaria, Residencia Profesional Docente.

[3]Desde marzo de 2020 son encuentros virtuales sincrónicos.

colectiva a la vez. (Equipo ISEP, 2020)

En cuanto a los **trabajos de campo y las prácticas** en las escuelas asociadas, se proponen actividades vinculadas con la observación, el diseño, la implementación de propuestas de enseñanza de la informática en la Educación Secundaria y el análisis de éstas, integrando los aportes provistos por los diferentes módulos del profesorado.

Teniendo en cuenta el **perfil de los y las cursantes**, un gran desafío es pensar, revisar y tensionar la formación fuertemente técnica con la que ingresan los profesionales y también sus concepciones acerca de qué es ser docente de Informática hoy en la educación secundaria. En el transcurso de dos años, se espera que quienes egresen, hayan podido incorporar los fundamentos conceptuales y las herramientas metodológicas para la construcción de sus propuestas de enseñanza, pero además elementos suficientes que les permitan ampliar la mirada sobre la realidad educativa y escolar de las instituciones que transitan cotidianamente y es en este marco que nos preguntamos:

¿La resignificación que realizan los profesionales, en cuanto a las concepciones desde las cuales se van acercando a la construcción de un saber pedagógico y didáctico específico de la informática -que ofrecen los distintos módulos del profesorado- contribuyen a reconfigurar sus ideas acerca de las prácticas de enseñanza? ¿En qué sentidos?

2. Metodología

Para comenzar a dar respuesta a esta pregunta nos propusimos observar las producciones de algunos/as cursantes -actividades y trabajos de acreditación- de la cohorte 2018, así como los registros y devoluciones de tutores en los espacios curriculares de las prácticas y de los módulos de las didácticas específicas[4], a fin de reconocer indicios respecto a los cambios que se van produciendo en sus concepciones en el transcurso del cursado.

Para realizar el seguimiento, se elaboró una muestra, seleccionando cursantes según las categorías que ISEP determinó para su inscripción, que incluye a docentes en ejercicio en escuelas públicas y privadas, así como docentes que aún no han trabajado en la escuela secundaria. También se consideraron las formaciones previas de los cursantes: ingeniería en informática, analista de sistemas de computación, licenciatura en computación y licenciatura o tecnicatura publicidad[5]. De un total de 47 cursantes que egresaron, se seleccionaron 9, 7 de los cuales tenían títulos de grado específicos de informática (ingenieros en sistemas, analistas de sistemas y en computación), un cursante contaba con el título de técnico en seguridad y el restante de técnico en publicidad. De todos ellos, 3 habían realizado trayectos formativos pedagógicos, 4 se encuentran dando clases en escuelas de gestión estatal. dos en escuelas de gestión privadas y 3 de ellos están inscriptos en la Junta de Clasificaciones de la Provincia de Córdoba, es decir son profesionales que aún no están trabajando en escuelas secundarias.

En cuanto a la selección de producciones en la Práctica Docente I se decidió relevar las biografías docentes, sus posicionamientos acerca de la enseñanza de la Informática en la escuela secundaria y sus primeras aproximaciones a la planificación de la enseñanza, en la Residencia Profesional Docente, se analizaron los informes finales[6].

En paralelo, se realizó un entrecruzamiento con las producciones del módulo Didáctica de la Informática, esto es la elaboración de un proyecto interdisciplinario que incorpore la enseñanza de la programación, así como la planificación

[4]La informática como objeto de enseñanza, Didáctica de la Informática, La Enseñanza de la Informática en la Educación Secundaria.

[5]Los licenciados y técnicos en Publicidad están habilitados en la provincia de Córdoba para tener a cargo materias relacionadas con la informática.

[6]Debido a la extensión de este artículo no se consideran de la Práctica Docente II la revisión de los análisis de la implementación de una secuencia didáctica a través de los incidentes críticos detectados.

de una clase específica para dicho proyecto.

Además, se decidió considerar los reportes elaborados por el Área de Evaluación del Instituto que se realizan en base a los resultados de las encuestas que los y las cursantes responden al finalizar cada módulo, donde valoran las propuestas formativas.

Lo anteriormente expuesto, nos permitió acercarnos a las trayectorias de estos profesionales en el profesorado y empezar a esbozar las primeras ideas sobre los procesos de resignificación/reinterpretación acerca de sus prácticas de enseñanza específicas de la informática.

3. Descripción/análisis de las producciones

En este apartado se presentan los primeros resultados, organizados en categorías que interesa abordar: las trayectorias formativas, la concepción del objeto de enseñanza, la planificación y la construcción metodológica.

3.1. Las trayectorias formativas como un proceso complejo

El desarrollo de la Práctica Docente I apuesta a una aproximación progresiva a la especificidad del quehacer docente en las escuelas secundarias, profundizando en la dimensión vinculada a la enseñanza. El interés se centró en analizar las autobiografías de los y las cursantes, que indagan en los enfoques, las perspectivas y las tradiciones que atravesaron en sus propios recorridos educativos para posibilitar el reconocimiento de sus perspectivas como docentes (Anijovich, Cappelletti, Mora y Sabelli, 2016, y Alliaud, 2003).

En estos escritos es posible observar diferentes situaciones de aprendizaje y vínculos con el saber que les proponían sus formadores o las instituciones educativas, así como distintas experiencias negativas o positivas y los motivos de la elección de la docencia.

Algunos rescatan el profesionalismo de los docentes que los formaron:

> "...De esa etapa puedo recuperar un docente en particular, de programación, que aparte de enseñarme a codificar me mostró y demostró lo bien que se trabaja sabiendo lo que se hace." (Cursante C)

> "... Fue la profesora de Tecnología [...] me dijo algo que no me olvido: "La carrera docente me ha dejado huellas muy fuertes en mi vida, experiencias con mis alumnos que nunca voy a olvidar; lo mejor que me ha pasado en esta vida es ser docente... Amo mi carrera y a mis alumnos"... fue la primera vez que pensé en hacer la carrera docente seriamente." (Cursante E)

Otros cursantes, desde el lado opuesto, es decir, desde el cuestionamiento a las metodologías empleadas por sus propios profesores, aportan:

> "Por aquellos días la enseñanza y el aprendizaje era bastante "Conductivo" (SIC), algo memorístico, pero muy repetitivo, de mucho ensayo y error... sólo nos dejaba al libre albedrío, el diseño de un programa; los conocimientos que nos transmitían eran bastante mecánicos [...] la única condición para enseñar era ser egresados de esta carrera, algunos profesionales ejerciendo en la ciudad en empresas importantes o bancos, los cuales sólo impartían sus saberes sin interpelar al grupo de alumnos..." (Cursante GG)

> "La enseñanza era un gran porcentaje teórica con una fuerte base en las matemáticas, teniendo acceso a las computadoras solamente una vez por semana, sin posibilidad de comprar o acceder a una. Por estas características, abandoné mis estudios de informática para retomarlos después de una década." (Cursante G)

En otro caso e intentando revertir estas situaciones, deciden comenzar a trabajar como docentes aún sintiendo esta actividad como un gran desafío:

> "... los profesores no sabían mucho de programación ni les interesaba saber, en lugar de eso llenaban las clases con contenidos teóricos cuyas aplicaciones prácticas no sabían explicarnos. [...] Fue tal mi enojo que me marcó para el resto del viaje. [...] Incorporarme como ayudante de cátedra fue todo un desafío [...] Si quería ganarme un lugar de respeto iba a tener que hacerlo desde otro lado, desde el conocimiento y la vocación de servicio..." (Cursante A).

Las pasiones, también estructuran el trabajo con la disciplina, es este sentido se expresan dos cursantes:

> "La situación se puso interesante a partir del cuarto año donde materias como Programación o Tratamiento de la Información me mostraron algo diferente: las estructuras lógicas, los diagramas de flujo, el Basic y el COBOL. Una verdadera pasión despertó en mí el desarrollo de algoritmos y la resolución de situaciones problemáticas. "Era capaz de darle vida a una máquina"."(Cursante CC),

> [...]"...decidí comenzar a estudiar ingeniería en sistemas en la UTN y allí comenzó una atracción por la informática [...] que se mantiene exactamente igual de potente que en su mejor época y actualmente me encuentra en la exploración del pensamiento computacional y la robótica." (Cursante C)

3.2. La concepción del objeto de enseñanza: posicionamientos frente al campo disciplinar y su enseñanza

En Práctica Docente I se presentaron algunos de los debates actuales al interior del campo de la Informática y se les propuso reflexionar acerca de la incidencia -o no- de las tensiones propias del campo en las prácticas docentes, para ello se expusieron los posicionamientos de la Fundación Sadosky[7] y los del grupo de ADICRA[8] . Por último se los invitó a debatir sobre la informática como herramienta transversal al resto de las asignaturas.

Es posible reconocer que la mayor parte de los docentes cursantes tienen un posicionamiento que complementa los postulados de la Fundación Sadosky y los de ADICRA, como también la transversalidad de la Informática. En general ellos aducen la imposibilidad de pensar la Informática respondiendo sólo a uno de los postulados de las organizaciones mencionadas. Algunas opiniones son:

> "Es importante que la informática tenga su lugar como disciplina (en todas las orientaciones) y desde allí interactuar y complementarse con las demás. No alcanza con la transversalidad y la educación digital." (Cursante Br)

[7]La línea de trabajo propuesta por la Fundación Sadosky hace foco en las Ciencias de la Computación.

[8]ADICRA considera que los saberes propios de la Informática deben tener un espacio curricular específico y que deben enseñarse a lo largo de la escolaridad.

> "...pienso que la transversalidad de la informática es indiscutible [...] Si queremos dar un gran paso hacia el progreso, me parece fundamental la incorporación de los fundamentos y competencias propias de las ciencias de la computación." (Cursante Ro)

> "... estoy convencido que el pensamiento formal de la Informática, las estructuras lógicas que se manejan en programación y la aplicación de la teoría sistémica facilitan el desarrollo cognitivo de los estudiantes, logrando la comprensión de saberes que muchas veces resultan complicados para ellos." (Cursante CC)

Por otro lado, en el módulo Didáctica de la Informática se abordaron distintas perspectivas vinculadas a la enseñanza de la informática a lo largo de su reciente historia y se propuso pensar el objeto de manera compleja, recuperando los aportes de referentes claves en la temática, teniendo también en cuenta la influencia de las tecnologías digitales en la vida cotidiana, la cultura y la sociedad para promover la reflexión sobre la disciplina y sus aportes para la construcción de problemas de enseñanza.

En este marco, los y las cursantes desarrollaron incipientes propuestas sobre proyectos interdisciplinarios que permitan incorporar la enseñanza de la programación a partir de una situación problemática. En este planteo de problemas que puedan abordarse computacionalmente, observamos que la gran mayoría optó por proyectos de robótica, especialmente vinculados a la problemática medioambiental: estaciones meteorológicas, sistemas de medición de la contaminación sonora, huertas automatizadas, ahorro energético y energías renovables. Algunas situaciones problemáticas planteadas fueron las siguientes:

> "[...] Frente a esta situación, les proponemos investigar cómo podemos automatizar el encendido y apagado de las luces del jardín de nuestra casa, para poder así optimizar el consumo eléctrico." (Cursante AA)

> "... la necesidad de realizar un dispositivo electrónico robótico que permita medir y recolectar información en diferentes escalas de sonido [...] en los distintos espacios cotidianos que frecuentamos, habitualmente podrían medirse en aulas, salones de reuniones, talleres, fábricas, en la vía pública, etc." (Cursante GG)

En menor medida algunas producciones apuntan a la programación de aplicaciones, específicamente sitios web donde se trabaja tanto desde el *frontend* en el diseño y programación de formularios como en el trabajo con bases de datos que organicen la información a partir de alguna necesidad planteada:

> "Implementar la interfaz gráfica de una aplicación web para administrar el kiosco escolar utilizando Bootstrap y aplicando buenas prácticas de desarrollo." Cursante A

> "[...] aportando desde el taller laboratorio de computación una base datos con cantidad, condiciones y geoposicionamiento de los RSU (Residuos Sólidos Urbanos) de una zona de la ciudad y la información que de ella pudiera resultar" Cursante C

Los casos menos frecuentes son los que apuntan al desarrollo de animaciones o pequeños juegos que incorporan el trabajo con otras áreas como Matemática y Lengua:

> "con la herramienta Scratch, citada anteriormente, se va a programar una simulación donde el estudiante va a trabajar diferentes operaciones matemáticas" Cursante M

Una observación interesante que surge de este análisis es que un gran porcentaje de cursantes decide trabajar con lenguajes visuales o de bloques. La herramienta más frecuente es Scratch, Scratch para Arduino y en menor medida App Inventor. También se destaca el uso de simuladores como la herramienta Tinkercad para el diseño y testeo de los prototipos en los casos de proyectos de robótica.

Por otro lado, si bien podríamos decir que la gran mayoría de cursantes logra identificar una situación problemática donde de situe a la informática como objeto de estudio, observamos que en algunos colegas aún se mantiene la percepción de considerar a la informática, y en especial a las TIC, sólo como una herramienta para lograr otros aprendizajes, incluso confundiendo ambas perspectivas:

> "El objetivo principal de este proyecto, es aprender a desenvolverse en la vida cotidiana mediante el uso de las Tecnologías de la Información y la Comunicación (TIC), es despertar el interés de los alumnos, tratando de desarrollar su pensamiento crítico con la inclusión de las TIC en el proceso de enseñanza-aprendizaje. [...] Con este proyecto, pretendemos contribuir a desarrollar las competencias básicas de los estudiantes del CENMA de adultos donde se establecerá como premisa el buen uso de las TIC, concretamente, con la herramienta Scratch." Cursante M

Asimismo, en la Residencia, último módulo del trayecto de formación, tuvieron que planificar y llevar adelante una propuesta de enseñanza para luego reflexionar sobre la propia práctica. Allí también pueden observarse perspectivas diversas.

Esto es, hacer foco en las tecnologías digitales como herramientas o tomarlas como objeto de estudio, tal como observamos en los siguientes relatos de experiencia:

> "... la inserción de estos adultos a la alfabetización y cultura digital fueron las principales preocupaciones y apuestas que orientaron el diseño de la presente propuesta, mediante la realización de una revista digital, con la intención de explotar las herramientas de la ofimática, pero ampliando las posibilidades del módulo hacia nuevas instancias de construcción [...]. En resumidas cuentas: "que puedan incorporar el uso de dispositivos tecnológicos para resolver cuestiones de la vida cotidiana", "que puedan producir un artículo integral y comunicarlo utilizando los bienes tecnológicos" (Cursante R)

> "... el propósito del espacio curricular de Entornos Digitales es introducir progresivamente a los alumnos en la noción de algoritmos, así como saberes propios del mundo de la programación, relacionados con contenidos específicos del desarrollo de software y otras habilidades y prácticas fundamentales para el desarrollo profesional en el Desarrollo de Software. En la vida cotidiana, se pueden encontrar con frecuencia algoritmos para realizar una tarea y resolver problemas [...] se propusieron inicialmente actividades más amenas, al estilo fichas para completar, de pensamiento computacional, como para introducir a estos nuevos alumnos en estos nuevos conceptos para ellos y con ello, puedan resolver de manera autónoma la propuesta. Por ese motivo, guiar en las estrategias de solución fue uno de los ejes fundamentales que se trabajaron." (Cursante GU)

> "El proyecto se desarrolló con estudiantes de 6to año, el anteúltimo en la modalidad técnica, dentro del espacio curricular Programación III, el cual se dicta en Espacio de Trabajo Compartido con Club de Ciencias [...] consiste en la implementación de un mecanismo de pago electrónico a través de crédito virtual en el que los estudiantes pueden pagar utilizando un token (un llavero que contiene un sensor

RFID). La gestión de pagos y crédito virtual se realiza a través de una aplicación propia que también se desarrolla como parte del proyecto." (Cursante A)

3.3. La planificación como práctica reflexiva

En la Práctica Docente I, nos aproximamos a un abordaje de la planificación de la enseñanza a partir del enfoque investigativo (Achili, 2005 y Calvo, 1992), planteando la importancia del análisis reflexivo como insumo para obtener información que permita anticipar decisiones. A los fines de este trabajo consideramos las producciones donde los cursantes dialogan y reflexionan acerca de los dilemas y las tensiones que se generan al pensar la enseñanza y al poner en acto la planificación. En este sentido pusieron en juego algunas problemáticas como el poder de la institución escolar, la legitimidad/legitimación de los contenidos académicos, la importancia de ampliar las perspectivas a la hora de la selección de éstos y la interpelación que produce en los docentes la mirada de un otro ajeno al aula:

"En las clases, el recorte que se realiza de los saberes académicos y la forma de presentarlos transmiten a los estudiantes una visión del mundo [...] la importancia de la relación de los sujetos con los contenidos escolares reside en que estos son presentados como los "verdaderos", implicando una cierta autoridad por medio de la cual, a la vez, definen implícitamente lo que no es conocimiento válido. Es por la fuerza de la legitimidad de los contenidos académicos transmitidos, que dificulta por igual a maestros y alumnos identificar como conocimiento válido sus propios conocimientos marginales que están presentes también en el aula. Entonces, el contenido que se ha de presentar válido deberá ser pensado desde una óptica más amplia, ya que no siempre lo ofrecido como legítimo resulta significativo." (Cursante C y Cursante R)

"Respecto de la observación de la clase por parte de otros sujetos en términos evaluativos (la mirada de otros) podría percibirse como una tensión, lo cual considero deja de serlo, en la medida que sea planteada desde un lugar genuino y con el objetivo de mejorar la práctica y que permita incorporar herramientas conceptuales y metodológicas que posibiliten el análisis de la propia práctica." (Cursante CC)

En sus consideraciones acerca del sentido de la planificación como un proceso reflexivo, pudo observarse que, en un alto porcentaje de los grupos, pudieron fundamentar adecuadamente, basándose en los autores propuestos en la clase, de manera significativa y fundada en nociones teóricas.

"Al iniciar el trayecto de práctica , me sentía muy capaz de resolver todo lo que se presente sin invertir grandes esfuerzos, la experiencia adquirida me llevaría a superar todos los obstáculos [...]. Sin embargo, la especificidad necesaria para la confección de una buena planificación, ha modificado mi panorama inicial. Pues al momento de expresar específicamente las intencionalidades pedagógicas, luego de un concienzudo diagnóstico, el compromiso personal es mucho mayor. La construcción de un plan de acción debe ser realista, contemplar en consideración aspectos muy precisos y modos de trabajo puntuales. Allí, varias de las competencias docentes (como la autoridad, la oratoria o la improvisación) se desvanecen y el diseño, la ejecución y la evaluación pasan a tener un carácter preponderante, que antes de mi contacto con el profesorado, sólo representaban un trámite burocrático. [...]

[...] en síntesis, según mi recorrido, parte de una buena práctica docente involucra: secuencias organizadas a partir de claras intencionalidades pedagógicas. Propuestas situadas para potenciar la calidad de los

aprendizajes procurados. Generar espacios que den lugar a un vínculo pedagógico entre los estudiantes, los docentes y el saber. Prever los usos del tiempo de aprendizaje, evitando caer en resoluciones utópicas. Planificar y registrar teniendo claridad sobre el rumbo educativo. Compartir con los estudiantes las metas y los objetivos que se proponen. Sobre todo en tiempos virtuales estimular la autonomía actuando como guía hacia la experimentación." (Cursante R)

3.4. La construcción metodológica

En relación al enfoque didáctico/metodológico que toman los y las docentes para abordar la enseñanza de la informática, se observa que sus producciones incorporan elementos del enfoque construccionista desarrollado por Papert (Papert, 1987), haciendo especial énfasis en la experiencia y la aplicación práctica de los saberes:

"Trabajar desde el enfoque construccionista como docentes, como posibilitadores en el aula de otras experiencias de encuentro con el saber habilita modos distintos de aprender en el que cada uno pueda ser partícipe de su proceso de aprendizaje, de la enseñanza en la institución escolar donde lleva a cabo este proyecto." (Cursante E)

"El enfoque consiste en abordar un problema real y significativo para los estudiantes y resolverlo con las herramientas adquiridas hasta el momento en una experiencia que se asemeje lo más posible al desarrollo de una solución informática profesional. Si bien se les propondrá una idea inicial, serán ellos quienes identifiquen, definan y acoten, de manera grupal, el problema. Luego, guiándolos en el uso de las distintas herramientas y técnicas, se desarrollará de manera progresiva la aplicación diseñada." (Cursante A)

Por otro lado, en relación a la incorporación de la interdisciplinariedad para desarrollar propuestas de manera integrada con el colectivo docente, las producciones observadas, en general, presentan coherencia y una articulación fluida de saberes, en especial en aquellos proyectos que trabajan la robótica/automatización a partir de problemáticas ambientales:

"El trabajo colaborativo de las diferentes asignaturas alrededor de un eje transversal, posibilita a los estudiantes responder ¿para qué me sirve lo que me enseñan? a la vez que se convierten en protagonistas del proceso. Por otro lado ejercitan la visión crítica de una actividad que afecta directamente a la sociedad toda, siendo parte de la solución a la situación problema planteada y llevando a sus hogares información valiosa para compartir en el ámbito social." (Cursante C)

Sin embargo, en aquellas propuestas donde las actividades de programación están diseñadas para contribuir al aprendizaje de otras asignaturas, puede percibirse una interdisciplinariedad forzada o débilmente fundamentada:

"Dados los desafíos a resolver a los/las estudiantes sobre temas de las asignaturas de Matemáticas y Lengua para que ellos puedan realizar una construcción crítica con el uso de condicionales, datos y secuencias de conceptos que irán apareciendo." (Cursante G)

"Área informática: Reconocer y diferenciar cada rama que ofrece la informática como fuente de trabajo. Área Lengua: La programación ayudará a que los estudiantes reconozcan y aprendan los diferentes tipos de textos a través de esta herramienta." (Cursante M)

4. Valoración de la propuesta formativa

Al finalizar cada módulo, los cursantes completan una encuesta anónima que indaga sobre las siguientes dimensiones: percepciones en cuanto a las clases, el tiempo de estudio, las actividades de acreditación, la experiencia de navegación, el vínculo con los y las docentes, las expectativas de formación que motivaron su inscripción y los principales aportes que le ofreció la propuesta formativa. A los fines de este escrito, recuperamos algunas valoraciones que hacen referencia a los aportes que los módulos específicos y prácticas docentes. Si bien, no podemos identificar a quiénes corresponden las respuestas se consideran relevantes a los fines de la indagación.

Lo vertiginoso de los cambios en este área del conocimiento provoca que algunos de los conceptos noveles del campo -como por ejemplo las posiciones más cercanas a la enseñanza de las ciencias de la computación, el pensamiento computacional, la informática como una tecnología social, entre otros- sean poco conocidos por una parte de los cursantes y, por ende –en estos casos– la enseñanza de estas conceptualizaciones y metodologías no lleguen a sus aulas. En este sentido, varios cursantes valoran el acercamiento a estas perspectivas y conceptos, para ellos novedosos, así como posibles enfoques para su enseñanza:

> “Conceptos nuevos aprendidos, temas interesantes y muy desafiantes”
>
> “[..] la praxis de la introducción en las ciencias de la computación y el software libre. Creo que como docentes nos debemos pasar del dicho al hecho y necesitamos ser acompañados. Este módulo nos alentó e intento guiarnos en este proceso largo... es un comienzo.”
>
> “Me parecieron desafiantes las propuestas ya que me acercaron a las diferentes plataformas de programación ya sea en bloque o no y las alternativas que podemos tener para plantear actividades para nuestros estudiantes.”
>
> “Las actividades de los encuentros presenciales fueron prácticas, muy agradables, y se utilizaron herramientas que a mi parecer son de una utilización muy amplia en la enseñanza de informática en la actualidad (necesarias digamos)”

Pueden observarse comentarios vinculados a la problematización acerca de cómo se concibe la informática en sí:

> “Poder ver la informática desde otros aspectos más allá del técnico”
>
> “Me desafió a salir de mi zona de confort, a ver desde otra óptica conceptos que ya conocía...”.
>
> “Ilustración sobre las complejidades de las redes sociotécnicas y tener una idea mejor de los principios de la computación.”

Otros comentarios se relacionan a la posibilidad de problematizar la enseñanza:

> “Me posibilitó tensionar algunas prácticas naturalizadas sobre la enseñanza de la informática, a interrogarme qué y cómo estoy enseñando, a saberme interpelado en relación a ¿quién estoy formando?”
>
> “Enseñarnos a tener una mirada más profunda en nuestro rol como docente como así también capacitarnos en nuestra profesión”
>
> “Comprender los conceptos que no se deben omitir de enseñar a los alumnos”

Por otro lado, en la Residencia, evalúan los siguientes aprendizajes como significativos:

> "...trabajar la planificación anual y la planificación didáctica desde todas las dimensiones posibles...",
>
> "me permitió conocer aspectos a tener en cuenta no sólo a nivel curricular sino también a nivel institucional y de convivencia con mis pares. Me ayudó a tener una mirada y una producción más profesional del oficio de ser docente. Me ayudó a deshacerme de viejas prácticas generalizadas y adaptadas de otros, para construir las propias y poder fundamentar la posición que asumí al realizarlas..."

También desde la reflexividad sobre sus propias prácticas valoran que:

> "...se incluyeron recursos interesantes para pensar el trabajo en el aula; implicaron la ida y vuelta constante sobre lo trabajado, lo que permitió la reflexión y re-elaboración de lo hecho"
>
> "Aprendí que independientemente de la antigüedad docente que se pueda tener, siempre hay algo que podemos aprender y mejorar e n nuestro oficio."

Además, apreciaron en varios casos la posibilidad que ISEP brindó de realizar las prácticas y residencia en contexto de aislamiento.

Sin embargo, podemos observar en algunos comentarios la existencia de cierta expectativa vinculada a que la formación docente se focaliza en el trabajo áulico, como una labor meramente instrumental donde se valora en menor medida la actualización/profundización disciplinar y pedagógico-didáctica.

En el transcurso del cursado se pudieron observar dificultades, algunas asociadas a la implementación inicial de la propuesta en una primera cohorte. Otras relacionadas a las propias tensiones que se establecieron entre los modos de vincularse con los conocimientos que ya traen incorporado los/as cursantes y los que se proponen en el profesorado. También, la limitación del tiempo destinado al cursado, a la complejidad de ciertas consignas, entre otras.

5. Reflexiones finales

La presente indagación forma parte de un trabajo de revisión constante por parte de los equipos que participan en esta propuesta. En este marco, la información analizada, proporcionó algunas pistas, sobre las que es necesario seguir profundizando, respecto del aporte de la formación específica en la resignificación de las prácticas de enseñanza de la informática de los y las cursantes.

Algunas preguntas que surgieron en este recorrido están relacionadas a las trayectorias previas de los/las cursantes y a sus títulos de base, ya que pareciera que influenciaron en las maneras de pensar, revisar y tensionar la concepción de la informática como objeto de enseñanza, así como las concepciones que pudieron construir acerca de qué es ser docente de Informática hoy.

Si bien la observación del recorrido por los módulos específicos dio cuenta de la importancia de este espacio de formación como oportunidad para que los y las cursantes vuelvan a pensarse; no solo en el marco institucional y normativo sino también para ir resignificando sus trayectorias y prácticas en el ejercicio de la docencia; también posibilitó reconocer la necesidad de seguir trabajando en la profundización de las articulaciones en los distintos módulos a fin de propiciar una apropiación más integrada de estos saberes e ir incorporando nuevas oportunidades considerando las formaciones de base diversas.

Seguir avanzando en este camino, constituye un desafío para un espacio de formación que pretende promover un movimiento constante entre los saberes de orden pedagógico generales y específicos, saberes didácticos y los propios

de la práctica docente reconociendo sus complejidades en relación al perfil de los destinatarios y los desafíos que plantean las políticas educativas actuales.

Bibliografía

Achilli, E. (2005). Investigar en antropología social: los desafíos de transmitir un oficio. Rosario: Laborde Editor.

Alliaud, A. (2003). La biografía escolar en el desempeño profesional de los docentes noveles. Vol. 1. [Tesis de doctorado en Educación]. Facultad de Filosofía y Letras, Universidad de Buenos Aires, Buenos Aires.

Anijovich, Cappelletti, Mora y Sabelli. (2016). Transitar la formación pedagógica. Dispositivos y estrategias. Buenos Aires: Paidós.

Calvo, B. (1992). Etnografía de la educación. Nueva Antropología, 12(42), pp. 9-26. Disponible en https://www.redalyc.org/pdf/159/15904202.pdf

Caverzacio, L., Cuello, S. y Equipo de Producción de Materiales Educativos en Línea. (2019). Clase 3: Ser docente de Informática en la escuela Secundaria. Práctica Docente I. Formación Docente Complementaria - Profesorados y Formaciones Pedagógicas. Córdoba: ISEP - Ministerio de Educación de la Provincia de Córdoba.

Caverzacio, L., Cuello, S. y Equipo de Producción de Materiales Educativos en Línea. (2019). Clase 10: Planificar la enseñanza. Práctica Docente I. Formación Docente Complementaria - Profesorados y Formaciones Pedagógicas. Córdoba: ISEP - Ministerio de Educación de la Provincia de Córdoba.

Caverzacio, L., Cuello, S. y Equipo de Producción de Materiales Educativos en Línea. (2019). Núcleo III: Reflexividad crítica. Residencia Profesional Docente. Formación Docente Complementaria - Profesorados y Formaciones Pedagógicas. Córdoba: ISEP - Ministerio de Educación de la Provincia de Córdoba.

Equipo ISEP (2020). Clase 2: Módulo Introductorio. Córdoba: ISEP - Ministerio de Educación de la Provincia de Córdoba.

Freire, P. (2003). El grito manso. Buenos Aires: Siglo XXI.

Litwin, E. (2008). El oficio de enseñar. Condiciones y contextos. Buenos Aires. Paidós.

Papert, S. (1987). Desafío de la mente. Buenos Aires, Argentina: Ediciones Galápago.

Propuesta Académica Profesorado de Educación Secundaria en Informática. Disponible en: https://isep-cba.edu.ar/web/profesorado-de-educacion-secundaria-en-informatica-en-concurrencia-con-el-titulo-de-base/

Rodriguez Pesce, E.S., Zalazar, N. y Equipo de Producción Materiales Educativos en Línea. (2020). Clase 1: Corrientes didácticas de la informática. Módulo Didáctica de la Informática. Profesorado de Educación Secundaria en Informática. Instituto Superior de Estudios Pedagógicos - Ministerio de Educación de la Provincia de Córdoba.

Aplicaciones del Pensamiento Computacional en Problemas de Química

Pamela Viale*† pamelaviale@uca.edu.ar UCA - UNR
Claudia Deco*† cdeco@uca.edu.ar UCA - UNR
Cristina Bender*† cbender@uca.edu.ar UCA - UNR

Resumen

La inclusión de la enseñanza del Pensamiento Computacional en diferentes carreras de grado permite formar a futuros profesionales para que puedan hacer uso de las nuevas tecnologías para resolver situaciones problemáticas de manera más rápida y eficiente. El presente artículo presenta una experiencia llevada a cabo durante el segundo semestre del año 2020 junto a los estudiantes del primer año de la carrera Licenciatura en Química, de la Facultad de Química e Ingeniería del Rosario (UCA), en la materia Informática. Con el objetivo de favorecer los procesos de enseñanza y aprendizaje de las competencias involucradas en el Pensamiento Computacional, se desarrollaron nuevos apuntes para la materia y se buscó que las prácticas sean significativas para el alumnado. Para lograrlo, se implementó en la materia, por primera vez, una estrategia de aprendizaje basada en proyectos. Si bien los temas de los proyectos fueron libres, se les pidió a los alumnos que reflexionen sobre aplicaciones que les puedan ser de utilidad en su futura labor como profesionales de la química y que, con eso en mente, propusieran un proyecto a desarrollar. De esta manera se buscó sensibilizar a los estudiantes sobre la relevancia de tener conocimientos en ciencias de la computación. La dificultad de las propuestas fue evaluada por la docente, quien, teniendo en cuenta la propuesta de los estudiantes, brindó el acuerdo o adecuó la dificultad teniendo en mente los alcances de la materia. Se buscó, a través de estos proyectos, que los estudiantes utilicen ciertos conceptos computacionales y que adquieran ciertas prácticas de resolución de problemas usadas comúnmente en el proceso de programación. En este artículo se presentan los proyectos desarrollados por nuestros alumnos, mostrando que es posible enseñar significativamente conceptos del pensamiento computacional a profesionales de otras áreas.

Palabras clave: Pensamiento Computacional, Programación, Licenciatura en Química.

1. Introducción

Desde hace ya varios años, estudiosos de todo el mundo afirman que la enseñanza del Pensamiento Computacional (PC) resulta beneficiosa para cualquier profesional, independientemente de la especialidad en la que se desarrolle. Se

*Facultad de Química e Ingeniería del Rosario, Universidad Católica Argentina, Campus Rosario, Argentina.

†Facultad de Ciencias Exactas, Ingeniería y Agrimensura, Universidad Nacional de Rosario, Rosario, Argentina.

usa el término Pensamiento Computacional para hacer referencia a un conjunto de competencias necesarias para la formulación y resolución de problemas de manera que puedan ser resueltos por un agente de procesamiento de información (Wing, 2014; Brennan y Resnick, 2012; Román-González et al., 2017).

El PC promueve la utilización de cuatro estrategias principales para la solución de problemas:

(1) Descomposición de un problema en subproblemas: involucra el análisis de problemas complejos y su descomposición en problemas más pequeños, más fáciles de analizar.

(2) Reconocimiento de patrones: cada uno de estos problemas más pequeños puede ser analizado en profundidad para identificar problemas similares ya resueltos.

(3) Abstracción: para buscar las soluciones a los problemas encontrados es necesario focalizarse en los detalles importantes e ignorar información no relevante.

(4) Pensamiento algorítmico: pueden proponerse una serie de pasos o reglas a seguir para crear una solución para cada uno de los subproblemas encontrados.

Una solución elaborada utilizando las estrategias del PC, puede ser fácilmente implementada en un sistema computacional, construyendo así una solución eficiente a un problema inicial complejo.

El estudio del Pensamiento Computacional ha tenido influencia en las investigaciones relacionadas al entendimiento y desarrollo de los procesos de enseñanza y aprendizaje, provocando que se aborde el tema desde el punto de vista de la formación en los distintos niveles educativos (García et al., 2017; García et al. 2017b.; Casali et al., 2020).

Se ha decidido entonces focalizar este trabajo en la formación de estudiantes de la Licenciatura en Química de nuestra facultad en las competencias relacionadas al Pensamiento Computacional.

2. Contexto

Según la ISTE (International Society for Technology in Education) y CSTA (Computer Science Teacher Association) el Pensamiento Computacional es un proceso para la resolución de problemas. Este proceso permite, entre otras cosas, aprender a formular problemas de manera que una computadora u otras herramientas puedan asistir en su resolución, permite analizar y organizar datos según cierta lógica, modelar y simular datos, automatizar algoritmos, identificar, analizar e implementar diferentes alternativas de solución para lograr un objetivo, y generalizar y transferir el proceso de solución de un problema a otros similares.

Una forma de lograr la aprehensión de estos conceptos y prácticas de resolución de problemas es mediante la implementación de dichas soluciones en lenguajes que puedan ser interpretados y ejecutados por un ordenador, o sea, en algún lenguaje de programación.

El objetivo es mejorar los procesos de enseñanza y aprendizaje de las competencias involucradas en el Pensamiento Computacional, para que los futuros licenciados estén más preparados para asumir los retos que demanda la sociedad contemporánea. La viabilidad del proyecto está sustentada en los avances obtenidos mediante trabajos previos de los integrantes en el área y su interacción con otros grupos de investigadores latinoamericanos (Rodríguez del Rey et al., 2020). Se propone desarrollar una didáctica para la enseñanza del pensamiento computacional y la programación para nuestros estudiantes y a su vez proponer metodologías activas y lúdicas para la utilización y entrega de los materiales en experiencias educativas presenciales y en línea.

3. Desarrollo de la experiencia

La experiencia que se relata en el presente artículo se desarrolló con los estudiantes del primer año de la carrera Licenciatura en Química durante el segundo semestre del año 2020. Se asumió que estos estudiantes no tenían conocimientos previos sobre algoritmia. El material desarrollado para dictar la materia comienza con una breve introducción a los métodos y las herramientas de resolución de problemas en la informática, continúa con una introducción al pensamiento computacional, definiendo este concepto y describiendo los principios de este pensamiento. Luego continúa con una introducción a la algoritmia, haciendo hincapié en las premisas del pensamiento computacional. Todos los conceptos se presentan de manera general y se usa un lenguaje pseudocódigo para describir los algoritmos, para focalizar la atención en los conceptos y no en la programación. El objetivo principal de la materia no es la de enseñar a programar sino la de sensibilizar a los estudiantes sobre la necesidad de aprender sobre el pensamiento computacional y el impacto que esto tiene sobre el modo de pensar y actuar de cualquier profesional, independientemente de su especialidad. Se buscó siempre dar ejemplos de aplicación relacionados a su área. Luego se introdujo el lenguaje Python como herramienta para la implementación del proyecto.

Antes de continuar con más detalles sobre el desarrollo de los proyectos, se debe precisar que esta experiencia se desarrolló en la modalidad virtual debido a la pandemia covid-19, lo cual implicó un doble desafío. Por un lado, se buscó introducir los principios del pensamiento computacional y que las prácticas sean significativas para el alumnado mediante una estrategia de aprendizaje basada en proyectos. Se intentó motivarlos mediante el desarrollo de proyectos de su propia elección, en temas relevantes a su profesión. Y por otro lado, se trató de buscar las herramientas que hicieran más fácil la participación de los alumnos, el trabajo grupal y colaborativo en un contexto virtual.

Si bien los temas de los proyectos fueron libres se les indicó algunos requisitos mínimos. Uno de ellos fue que la aplicación debía tener un menú de opciones desde el cual el usuario pudiese elegir una tarea entre varias a ejecutar y que una vez ejecutada la opción la aplicación volviese a presentar el menú. Este requisito no fue azaroso sino que se buscó, mediante esta imposición, favorecer el uso de repeticiones y condicionales para la implementación del menú, así como también la abstracción y/o modularización del código en la implementación de cada opción del mismo. A su vez, los estudiantes contaban, al momento de desarrollar sus proyectos, con ejemplos de menús similares desarrollados durante las clases, favoreciendo así la reutilización/adaptación de código en diversos contextos. El lenguaje de programación elegido, como bien se dijo anteriormente, fue Python y, para facilitar el seguimiento de los avances y la colaboración entre los estudiantes incluso entre integrantes de diferentes proyectos, se usó el intérprete online de repl.it. Este intérprete online permite compartir código de manera muy fácil y rápida. Simplemente compartiendo el enlace (link) del código, cualquier persona puede acceder y hacer pruebas en él (sin modificar el código original del propietario).

Antes de comenzar con el trabajo sobre los proyectos, la docente creó un documento compartido de Google donde cada grupo tuvo asignado un espacio a llenar con los datos de su proyecto. La información que debían llenar consistió en los siguientes ítems:

- Apellidos y nombres de todos los integrantes del grupo
- Título del proyecto
- Descripción del programa a desarrollar
- Menú de opciones
- Ejemplos de datos y cálculos a realizar
- Ejemplo de casos de uso (detallando los mensajes en pantalla y los datos que ingresa el usuario)

- Link al código en la plataforma repl.it

De esta forma la docente pudo evaluar la dificultad de los temas propuestos por los alumnos y realizar algunas modificaciones/sugerencias según los contenidos de la materia.

Toda la clase, tanto docente como alumnos, tuvieron en todo momento acceso al documento de Google con la documentación y los enlaces a las implementaciones de los diferentes proyectos. Esto fomentó no sólo el trabajo grupal sino también el trabajo colaborativo entre los diferentes grupos. También ayudó a que la docente pudiese realizar un seguimiento permanente de los avances.

Se presentan a continuación los temas de los cinco proyectos desarrollados por los alumnos y se da, en cada caso, una breve descripción del mismo.

PROYECTO 1: Peligrosidades de Reactivos en Laboratorio

Contexto de Aplicación

Los químicos trabajan constantemente con reactivos los cuales algunos suelen ser muy peligrosos y pueden causar daño a la salud, al medio ambiente o implicar riesgos físicos. Teniendo esto en mente, los estudiantes de este grupo desarrollaron un prototipo que tiene como finalidad informar sobre cuáles son las peligrosidades de los reactivos en el laboratorio, cuál es su correcta manipulación y a su vez agilizar el tiempo de clasificación de los mismos.

Interfaz de Usuario

Elija una de las siguientes opciones:

1. Explosivo
2. Inflamable
3. Comburente
4. Corrosivo
5. Tóxico
6. Irritación Cutánea
7. Peligro por aspiración
8. Peligroso para el medio ambiente
9. Salir

Sobre la Implementación

La idea de este grupo fue la de implementar una aplicación que permitiera al usuario elegir entre los tipos de reactivos que muestra el menú y de manera inmediata el usuario obtuviese una respuesta acerca de su peligrosidad, su correcta manipulación y clasificación.

PROYECTO 2: Calculadora de Peso Molecular

Contexto de Aplicación

Los profesionales en el área de la química necesitan, en muchas ocasiones, realizar cálculos o resolver problemas con pesos moleculares de elementos o compuestos químicos. El fin del programa desarrollado por este grupo es el de agilizar el proceso de una manera interactiva, eficaz y confiable, reduciendo las posibilidades de errores y alivianar el trabajo.

Interfaz de Usuario

¿Qué peso desea saber?

1. Peso molecular
2. Peso atómico
3. Salir

Sobre la Implementación

En este caso la implementación de este grupo permite seleccionar al usuario si desea conocer el peso de un átomo o una molécula. En el caso que se trate de un átomo la respuesta es inmediata, en caso de una molécula el programa solicita al usuario que ingrese cada elemento que compone la molécula junto a la cantidad de átomos que la componen. Al finalizar dicho ingreso el programa arroja de manera automática el peso total de la molécula.

PROYECTO 3: Cálculos Estequiométricos

Contexto de Aplicación

La estequiometría es el cálculo para una ecuación química balanceada que determinará las proporciones entre reactivos y productos en una reacción química. La estequiometría tiene por finalidad establecer aquellas relaciones entre los reactantes y productos en una reacción química. Los reactantes son precursores del proceso y los productos la parte final de la reacción, es decir, lo que se formó.

Interfaz de Usuario

Elija una de las siguientes opciones:

1. Molaridad
2. Normalidad
3. Salir

Sobre la Implementación

El objetivo principal del programa es, a través de la concentración que el usuario ingrese, ya sea en molaridad (moles de soluto por litros de solución) o en normalidad (número de equivalentes por litro de solución), determinar la masa (en el caso de un sólido) o el volumen (en el caso de un líquido) de reactivo que se debe tomar para obtener la solución que se desea. De este modo, el programa, colabora con los cálculos del usuario, minimizando los posibles errores y optimizando el tiempo de trabajo.

PROYECTO 4: Baterías

Contexto de Aplicación

Las celdas electroquímicas son dispositivos por el cual se transfieren electrones, convirtiendo la energía de una corriente eléctrica en energía química de los productos de una reacción redox. La diferencia de potencial de esta corriente se denomina potencial de celda o fuerza electromotriz (fem). Hoy en día las celdas electroquímicas se utilizan en celulares, autos eléctricos, baterías, entre otras aplicaciones. Estas aplicaciones necesitan de cierta energía mínima para poder funcionar correctamente. Es por eso que científicos y químicos buscan, a través de ensayos, poder encontrar los elementos que al reaccionar generan la mayor energía posible en celdas electroquímicas, de forma que sumando la fem en serie se pueda obtener el valor de la tensión de una batería.

Interfaz de Usuario

Bienvenido al programa de investigación de una celda electrolítica. En este programa se podrá determinar la energía de una celda en función de las concentraciones de reactivos y productos para la reacción de Zinc y Cobre. Para ingresar valores con decimales utilizar “EL PUNTO DECIMAL”

Considere ingresar valores positivos de A, B y E

Elija una de las siguientes opciones:

- Cálculo de la fem
- Cálculo de la concentración del producto (A)
- Cálculo de la concentración del reactivo (B)
- Salir

Sobre la Implementación

El programa permite calcular la fem E producida por una celda en condiciones no estándar o determinar la concentración de un reactivo o de un producto mediante la medición de E para la celda. Una vez elegida una de las opciones del menú principal se determina el valor correspondiente elegido, a través de cálculos de suma, resta, multiplicación, división, logaritmo y potencia. Por ejemplo, para calcular E, el usuario deberá ingresar los valores de las concentraciones de [A] y de [B]. Variando una de las variables, A o B, y dejando la otra constante, se podrán obtener distintos valores de E (E1,E2,...En).

PROYECTO 5: Medios de Cultivo

Contexto de Aplicación

En la producción de indicadores biológicos de esterilización (productos para verificar si un proceso de esterilización es efectivo) existe una etapa que consiste en la preparación de medios de cultivos coloreados. El color es un factor crítico y es requisito que siempre sea el mismo. Para ello, como control de calidad de los medios de cultivos, se mide la intensidad de un haz de luz antes y después de pasar a través de una muestra coloreada y a partir de ello se arroja un valor de absorbancia proporcional a la concentración de colorante. Para cada medio de cultivo, se conoce el rango de absorbancia dentro del cual el medio se acepta y por fuera del cual, se rechaza. Se sabe además que algunos medios de cultivo poseen el mismo rango de absorbancia.

Interfaz de Usuario

Ingrese la opción correspondiente al medio de cultivo preparado:

- MC220
- MC222
- MC224
- MC10
- MC110
- MC20
- MC1020
- MC1030
- MC23QR71
- SALIR

Sobre la Implementación

Los integrantes de este grupo desarrollaron un programa que los ayuda a decidir si un medio de cultivo es aceptado o rechazado según el valor de absorbancia medido por los equipos del laboratorio. Los medios de cultivo producidos son varios y se identifican con un código compuesto por letras mayúsculas y números.

Tabla 1

A través de esta experiencia se buscó que los estudiantes utilicen los conceptos computacionales y que adquieran ciertas prácticas de resolución de problemas usadas comúnmente en el proceso de programación. Haremos un breve análisis de los proyectos en la próxima sección.

4. Análisis de los Proyectos

Durante el desarrollo y seguimiento de los mismos se fomentó el uso de los conceptos y prácticas vistos durante el dictado de la materia. En la Tabla 2 se presenta un análisis sintético donde mostramos los conceptos y prácticas computacionales (Brennan y Resnick, 2012) usadas en el desarrollo de cada proyecto (opciones: Sí, No, Parcialmente).

	Conceptos Computacionales (Conceptos usados durante la programación)					**Prácticas Computacionales** (Prácticas de resolución de problemas usadas durante el proceso de programación)		
PROYECTO	**Secuencias**	**Bucles**	**Condicionales**	**Operadores**	**Datos**	**Testeo y Debugging**	**Reutilización de proyectos anteriores**	**Abstracción y Modularización**
1	Sí	Sí	Sí	No	No	Sí	Parc.	Parc.
2	Sí	Sí	Sí	Sí	Sí	Sí	Parc.	Parc.
3	Sí	Sí	Sí	Sí	Sí	Sí	Parc.	Parc.
4	Sí	Sí	Sí	Sí	Sí	Sí	Parc.	Parc.
5	Sí	Sí	Sí	No	Sí	Sí	Parc.	Parc.

Tabla 2: Análisis sintético de los proyectos.

Se puede observar que sólo dos proyectos no hicieron uso de operadores y uno no usó datos ya que los temas involucrados no lo requerían. Además, si bien se fomentaron la reutilización de código (por ejemplo, el uso y adaptación de menús y funciones de búsqueda implementados y discutidos en clase) y la modularización y abstracción de código (haciendo conscientes a los alumnos de aquellas situaciones donde esta estrategia aportaba claridad a la vez que se evitaba la repetición de código) vemos que el uso de estas estrategias fue parcial. Sin embargo, esto se condice con otras experiencias previas de la docente. Estas estrategias llevan un tiempo mayor para su buena asimilación y para que los alumnos sean capaces de utilizarlas de manera correcta y eficiente.

5. Conclusiones y trabajos futuros

Los trabajos realizados por los estudiantes de la carrera de Licenciatura en Química han sido de calidad y mostraron, en buena medida, la adquisición de los conceptos del pensamiento computacional. Consideramos que esta primera experiencia ha sido muy enriquecedora ya que vemos un cambio en el posicionamiento de nuestros estudiantes ante nuevos desafíos, además observamos cambios considerables en su forma de expresarse y nuevos cuestionamientos acerca de la tecnología que los rodea. Estas nuevas perspectivas computacionales (cambios que produce el aprender a programar) son las que buscamos que aparezcan con el plan de innovación que implementamos en nuestra facultad. Buscamos que nuestros egresados puedan hacer uso de las nuevas tecnologías para resolver problemas inherentes a su profesión y así poder dar respuestas a una sociedad cada vez más exigente.

Como trabajo futuro se propone en este segundo semestre del año 2021 aplicar la experiencia con los alumnos de la carrera de Ingeniería Industrial, esperando obtener resultados también alentadores. Se propone, además, la creación de instrumento/s de evaluación de las habilidades del pensamiento computacional por parte del estudiantado, para

lograr un análisis más objetivo de los resultados.

Bibliografía

Brennan K. y Resnick M. (2012). New frameworks for studying and assessing the development of computational thinking. In Proceedings of the 2012 annual meeting of the american educational research association (Vancouver, Canada).

Casali A., Deco C., Viale P., Bender C., Zanarini D. y Monjelat N. (2020). Enseñanza y aprendizaje del pensamiento computacional y la programación en los distintos niveles educativos. Mayo 2020 - XXII Workshop de Investigadores en Ciencias de la Computación (WICC 2020, El Calafate, Santa Cruz). pp 595–599.

García M., Deco C., Bender C. y Collazos C. (2017) Invited paper: Robotics Based Strategies to Support Computational Thinking: The Case of the Pascual Bravo Industrial Technical Institute. Journal of Computer Science and Technology (JCS&T) Vol. 17 No. 1, pp 59–67.

García M., Deco C., Bender C., Collazos C. (2017b). Herramientas de Diseño para el Desarrollo de Competencias en Educación Básica, Media y Tecnológica: Experiencia en el Instituto Técnico Industrial Pascual Bravo de Colombia. Revista TE&ET (Argentina).

Rodríguez del Rey Y., Cawanga Cambinda I., Deco C., Bender C., Avello-Martínez R. y Villalba-Condori K. (2020). Developing Computational Thinking with a Module of Solved Problems. In Computer Applications in Engineering Education.

Román-González M., Perez-González J. C. y Jiménez-Fernández C. (2017). Which cognitive abilities underlie computational thinking? Criterion validity of the Computational Thinking Test. Computers in Human Behavior, 72: 678–691.

Wing, J. (2014). Computational Thinking Benefits Society. Social Issues in Computing.

Reflexiones sobre el diseño e implementación de una especialización docente en enseñanza de la programación

Araceli Acosta
araceli@unc.edu.ar
Universidad Nacional de Córdoba

Cristián Rojo
crirojo@disroot.org
Universidad Nacional de Córdoba

Débora Cingolani
deboracingolani@gmail.com
Universidad Nacional de Córdoba

David Araya
david.araya@unc.edu.ar
Universidad Nacional de Córdoba

Resumen

En este trabajo pretendemos realizar una evaluación de la Especialización Docente en Enseñanza de la Programación poniendo el foco en las decisiones pedagógicas que tomaron los y las docentes al momento de abordar determinados contenidos en el aula. En este sentido, resulta pertinente preguntarnos acerca de qué conceptos, prácticas y perspectivas entran en juego en la introducción de la programación en las aulas de los diversos niveles educativos en los que trabajan los/as docentes que cursaron esta carrera. En esta ponencia, mediante el análisis de los trabajos finales y de los cuestionarios de finalización, intentaremos describir, categorizar y analizar los contenidos de programación que allí aparecen.

Palabras clave: Enseñanza, Programación, Ciencias de la Computación, Especialización, Docencia.

1. Introducción

La enseñanza de la programación y las Ciencias de la Computación en la escuela nos proponen grandes desafíos: por un lado, avanzar en la incorporación de los núcleos de aprendizajes prioritarios (NAP) de Educación Digital, Programación y Robótica a la educación obligatoria; y, por otro, formar una comunidad docente y académica capaz de asumir la responsabilidad de enseñar esos contenidos con propuestas didácticas sólidas. Sin embargo, la crisis sanitaria mundial que vivimos desde el año 2020 provocó un retraso en la implementación de dichos aprendizajes prioritarios, definidos en la Resolución CFE 343/18 pero, a su vez, aceleró forzosamente la incorporación de las tecnologías de la información y comunicación a los procesos de aprendizaje.

Es en este contexto que nos proponemos reflexionar sobre algunos aspectos en base a una propuesta de formación docente, con la expectativa de aportar nuestra experiencia a futuras iniciativas de formación similares. Por un lado, describiremos algunas de las decisiones de diseño que la estructuraron, y por otro, realizaremos un análisis cualitativo sobre los trabajos finales de la carrera que dan cuenta de los aprendizajes alcanzados por quienes cursaron el trayecto de formación y pudieron trasladarlo con propuestas concretas a sus aulas, con sus estudiantes.

2. Diseño de la propuesta de formación

Nos propusimos diseñar una especialización docente en el área de la Ciencias de la Computación, en particular en la programación, estructurada por proyectos. La estrategia principal para el diseño de la currícula fue la de backward mapping (mapeo invertido, Elmore, 1979). Siguiendo esta estrategia, primero se definieron los objetivos generales y particulares que se querían lograr con este postítulo. En otras palabras, ¿qué queremos que los y las docentes aprendan y hagan en sus clases luego de cursar este postítulo? En función de estos objetivos se seleccionó el formato de curriculum basado en proyectos (tomamos como antecedente el programa de masters en Ciencias de la Computación de la Universidad de Manchester) y el modelo de formación docente TPACK[1] que busca integrar los saberes disciplinares, pedagógicos y tecnológicos.

Los proyectos seleccionados para esta propuesta de formación tienen la intencionalidad de integrar los contenidos fundamentales de la programación con los saberes pedagógicos y tecnológicos necesarios para su enseñanza. En ese sentido, se ofrece una formación que integra contenidos de la disciplina con saberes de la didáctica y la pedagogía. Otro aspecto destacado es la vivencia personal de los y las docentes que aprenden por primera vez los contenidos disciplinares, dando lugar a una experiencia de primera mano en el aprendizaje por proyectos y, en particular, en el aprendizaje de los contenidos de programación utilizando esta metodología. Entendemos que esa vivencia implica una mejor apropiación tanto de los conceptos como de la metodología de trabajo, que luego podría ser traducida al trabajo en el aula. Sobre todo si pensamos que estos saberes disciplinares y su didáctica eran nuevos para la mayoría de los y las docentes y los y las estudiantes.

Los contenidos se presentan en un currículum de formato espiralado a modo de nociones que van ampliando su dificultad en tres niveles (desde la familiaridad al uso y, luego, al dominio) a lo largo de la especialidad. De este modo, se espera que los/as docentes, en primera instancia, se familiaricen con algunos conceptos; luego, los usen en un programa o análisis; y, finalmente, produzcan nuevos saberes a partir del dominio conceptual.

El formato espiralado persigue dos objetivos complementarios: en primer lugar, abordar en los primeros módulos (primer año) propuestas y herramientas más definidas y estructuradas para que los y las docentes puedan trasladar al aula de manera más inmediata. En segundo lugar, y durante el segundo año de la carrera, profundizar sobre los conceptos de programación, bajo la hipótesis de que para enseñar es necesario conocer en profundidad aquello que se quiere enseñar. Esto último permite, por un lado, elaborar propuestas creativas y efectivas que se adapten a la realidad del aula y, por otro, contar con la confianza necesaria para incorporar nuevas propuestas a sus estudiantes.

En síntesis, la propuesta tiene el objetivo de formar a los y las docentes en la enseñanza de una área relativamente nueva para las instituciones educativas, recuperando "buenas prácticas" de formación docente: currículum por proyectos, saberes disciplinares y didácticos integrados y contenidos espiralados, con propuestas y herramientas concretas para implementar en el aula.

[1]Technological Pedagogical Content Knowledge, o Conocimiento Técnico Pedagógico del Contenido en español.

3. Decisiones y desafíos en el diseño de la propuesta

3.1. Presencialidad vs. semipresencialidad

La necesidad de diseñar una propuesta semipresencial se estableció desde el origen a partir de un pedido del Ministerio de Educación de la Provincia de Córdoba. Principalmente, perseguía el objetivo de una mayor inclusión geográfica en el acceso a la formación por parte de los docentes de la provincia. A esto se le suma la caracterización de quienes cursan la especialización: adultos, docentes en ejercicio y, en su mayoría, con múltiples responsabilidades laborales y familiares, lo que significaba menor disponibilidad y flexibilidad horaria durante la semana. En ese marco, se diseñó una propuesta semipresencial con un régimen de cursado presencial (dos sábados por mes), complementadas con actividades virtuales asincrónicas.

Entendemos que una propuesta virtual requiere sus propias estrategias didácticas para lograr los mismos objetivos que la presencialidad. Por lo que fue necesario diseñar la propuesta completa en esta clave, dando lugar a una nueva experiencia pedagógica en cuanto a la enseñanza de la programación.

Aprender a programar requiere aprender no solo conceptos sino principalmente "habilidades". El desafío que nos planteamos fue la posibilidad de favorecer el desarrollo de esas habilidades de programación por parte de los/as cursantes mediante el acompañamiento de sus profesores/as y tutores/as para que tuvieran la confianza de ofrecer experiencias de programación a sus propios/as estudiantes.

En este marco, las clases virtuales incluyen especialmente posibilidades de experimentar (con hardware y software) y de comprobar premisas (actividades de laboratorio). Es decir, se propone una educación a distancia en donde docentes, junto con sus pares, puedan reunirse a aprender juntos y practicar en sus escuelas o localidades como parte central de la clase virtual. Esto requirió, en algunos módulos, reducir el desarrollo conceptual y ofrecer experiencias de construcción de conceptos y habilidades a través del desarrollo del oficio de programador. Habida cuenta de que la experiencia no es transferible, sino que se logra de primera mano. Para este equipo el desarrollo de habilidades de manipulación de hardware es también contenido específico de programación y no puede reemplazarse completamente por simulaciones virtuales.

3.2. Profundidad en contenidos de programación vs. formación general en conceptos de Ciencias de la Computación

En relación a esta disyuntiva, optamos por tener un abordaje en profundidad, que permita un dominio de la disciplina y el desarrollo de habilidades tanto conceptuales como de enseñanza de esos conceptos.

Partimos de la hipótesis de que el dominio de los conceptos permite contar con más y mejores herramientas tanto conceptuales como didácticas para diseñar propuestas de enseñanza en el aula que aborden los contenidos específicos. En este sentido, una formación sólida en programación da más herramientas para abordar estos contenidos en el aula, por parte del docente y, además, facilita la actualización y la aproximación a nuevos contenidos.

Además, el diseño espiralado de la propuesta, permite abordar los mismos conceptos de programación con diferentes herramientas avanzando sucesivamente en profundidad conceptual. En particular, esta estrategia didáctica permite desarrollar la capacidad de abstraerse del lenguaje de programación o la plataforma de trabajo e identificar las construcciones básicas de un programa. Se observó en los primeros módulos, por ejemplo, que los/as cursantes incorporaron nuevas plataformas sugeridas por los/as docentes, que aprendieron a utilizar por sí mismos/as y sin

mucha dificultad, a partir de los aprendizajes conceptuales con los que contaban y las utilizaron en el diseño de nuevas propuestas didácticas.

3.3. Perfil de los/as destinatarios de la propuesta

Considerando la falta de docentes formados/as en programación y su didáctica, se concibió la propuesta como una iniciativa de reconversión docente, sin desconocer que la disciplina aún no cuenta con un espacio curricular propio en la escuela obligatoria en la mayoría de las provincias. Tal como señala Gómez (2020) esta estrategia también la podemos encontrar en el panorama internacional: "Si bien la mayoría de los investigadores e impulsores de políticas públicas coinciden en que las carreras terciarias o universitarias son la mejor opción para formar docentes calificados, ante la falta de docentes para enseñar programación, muchos países capacitan docentes que ya están en actividad (como Reino Unido, Nueva Zelanda, Estados Unidos y Alemania)". En este sentido, la propuesta debía contemplar herramientas para la integración disciplinar y la apropiación de conceptos y las habilidades de las Ciencias de la Computación en las áreas en las que se desempeña cada docente.

Uno de los sentidos que subyace a la organización pedagógica de la especialización fue el pensar **la enseñanza como una tarea compartida**, por tal motivo al momento de realizar la convocatoria, se promovió la inscripción de grupos de docentes de la misma institución educativa con el interés común de incorporar la enseñanza de la programación en la propuesta pedagógica escolar. La idea de propiciar el trabajo docente en equipo no sólo vincula una nueva forma de pensar la enseñanza en la escuela, sino también de vivenciar formas de aprendizaje entre pares. Esto último es una práctica común en las Ciencias de la Computación, la programación no es una actividad solitaria sino colectiva que se enriquece de los aportes que otros realizan. Por tal motivo, el trabajo en equipo se constituye en un propósito de la especialización pero también en un aprendizaje a promover en los estudiantes que aprenden Ciencias de la Computación. Siguiendo a Brennan y Resnick, una de las perspectivas propias del pensamiento computacional es la de conectar con otros y otras. Al respecto indican que "La creatividad y el aprendizaje son prácticas profundamente sociales por lo que no sorprende que diseñar medios computacionales con Scratch se enriquezca mucho mediante la interacción con otros" (Brennan y Resnick, 2012).

4. Estructura del plan de estudios

La Especialización Docente en Enseñanza de la Programación es una carrera semipresencial de dos años de duración (400 horas), que tiene como objetivo formar a docentes en las bases de la programación y su didáctica. El plan de estudios está organizado en ocho módulos con formato de proyecto, siguiendo los principios del aprendizaje por descubrimiento (enfoque pertinente y privilegiado para su didáctica) y dos seminarios de trabajo final, al terminar cada año de cursado (ver Tabla 1).

En términos generales, durante el primer año de cursado, la propuesta busca abordar los conceptos básicos de programación y diferentes herramientas para la enseñanza en el aula y, durante el segundo año, profundizar los conceptos fundamentales e introducir conceptualizaciones y discusiones más amplias de las Ciencias de la Computación y debates sociales y filosóficos en relación a ellas.

En cada uno de los módulos que conforman la especialización, los/as docentes se vieron desafiados/as en pensar las formas de introducir los contenidos abordados en propuestas pedagógicas concretas, aplicadas en las aulas en donde

	Primer Año		Segundo Año
M1	Lógica, Programación y su Enseñanza	M6	Resolución de Problemas de Programación a partir del Diseño de Juegos Interactivos
M2	Ensamble y Programación de un Robot	M7	Introducción a la Programación Orientada a Objetos a partir del desarrollo de un sistema de votación
M3	Administración y Configuración de Herramientas Informáticas	M8	Procesamiento de Datos a partir de Análisis de Bases de Datos Públicas
M4	Introducción a Lenguajes de Programación a través de Animaciones y Videojuegos	M9	Desarrollo de Aplicaciones Web.
S5	Seminario de Trabajo Final I	S10	Seminario de Trabajo Final II

Tabla 1: Estructura del Plan de Estudios de la Especialización en Enseñanza de la Programación

ellos/as se desempeñaban. Esto se sintetiza en la elaboración de secuencias didácticas que se complementan con la elaboración del proyecto, como parte de la evaluación.

La organización curricular de la especialización prevé el cursado de dos seminarios vinculados a la elaboración del trabajo final. Este último implica la planificación e implementación de una propuesta de enseñanza en la que se aborden contenidos específicos de las Ciencias de la Computación. También, los/as cursantes deben llevar a cabo el análisis pedagógico de la planificación y de la experiencia áulica a fin de poner en diálogo los conocimientos adquiridos durante la especialización con su propia práctica docente.

5. Contexto de aplicación

La carrera se dictó en 2018 y 2019, en el marco de un acuerdo entre la Facultad de Matemática, Astronomía, Física y Computación de la Universidad Nacional de Córdoba, la Fundación Manuel Sadosky y el Ministerio de Educación de la Provincia de Córdoba, que designó como sede de la carrera al Instituto de Formación Superior Simón Bolívar. En esta cohorte, que se utilizará como muestra, se inscribieron 662 personas, participaron 179 personas de la primera clase; de los cuales 155 finalizaron el primer módulo y egresaron de la especialización 105 docentes.

A los fines de caracterizar el universo destinatario final de la propuesta, podemos mencionar que el 65 % del total de docentes inscriptos/as se desempeña en el nivel secundario, el 26 % en escuelas primarias, el 7,9 % en institutos superiores y un 1 % en el nivel inicial. Otro aspecto importante a destacar es que el 21,2 % de los docentes cuenta con formación en el campo de la informática, mientras que un 23,5 % cuenta con titulación relacionada con las tecnologías y los demás inscriptos están formados en otros campos disciplinares. Al momento de iniciar el cursado el 46,6 % manifestó contar con conocimientos previos de programación.

Al finalizar el cursado de los módulos, se realizó una encuesta a los/as docentes a fin de identificar su percepción en relación a la especialización y recuperar información relevante sobre la planificación e implementación de los trabajos finales. Si bien la encuesta nos permite concluir que los/as docentes han recuperado contenidos de las Ciencias de la Computación vistos en todos los módulos de la especialización, los que aparecen con mayor recurrencia son aquellos abordados en los Módulos; “1-Lógica, Programación y su enseñanza”, “4-Introducción a los lenguajes de programación a través de la animación y los juegos”, “6-Resolución de problemas de programación a partir del diseño

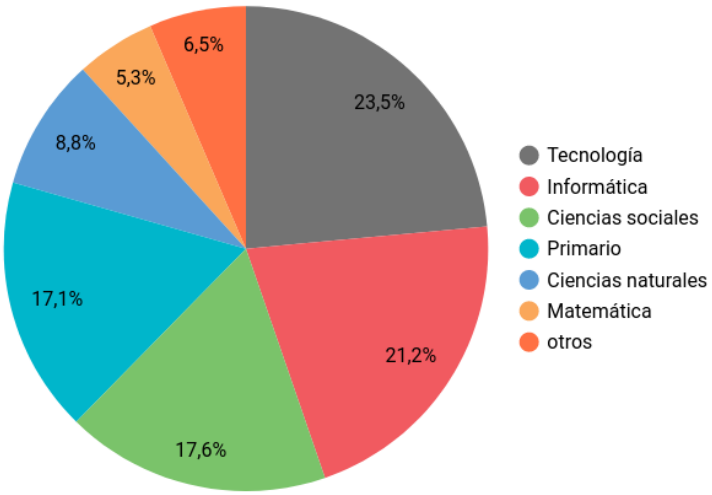

Figura 1: Área disciplinar de los docentes a partir de la titulación de base registrada.

de juegos interactivos", "3-Administración y configuración de herramientas informáticas". Un aspecto importante a recuperar de las encuestas es que el 62,2 % de los/as docentes sostiene que el grupo de estudiantes destinatario del proyecto final no había participado de una experiencia escolar previa vinculada al aprendizaje de saberes propios de las Ciencias de la Computación.

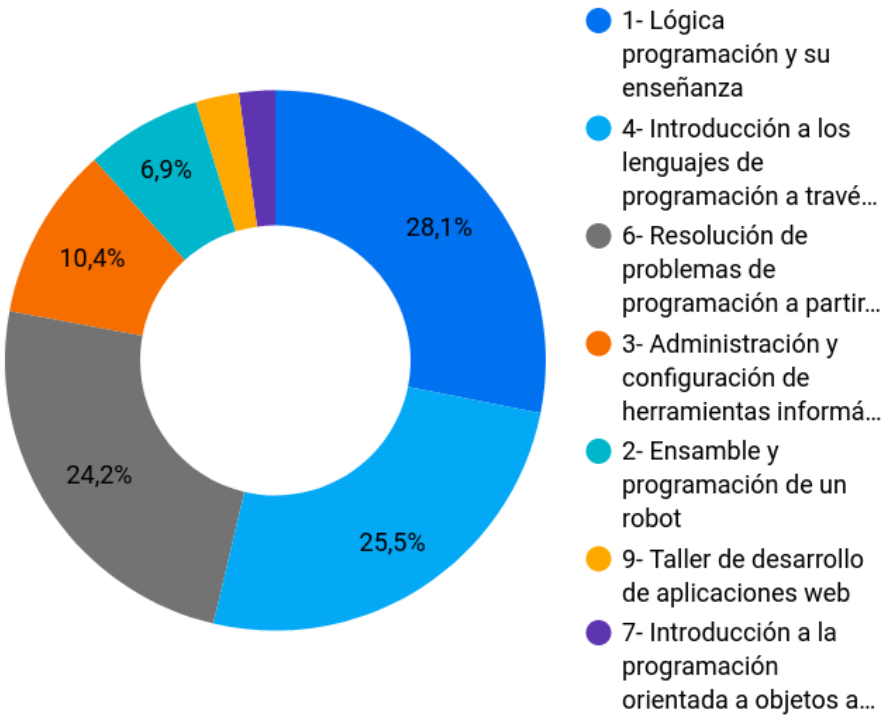

Figura 2: Módulos relacionados con los trabajos finales.

Resulta claro que, para la mayoría de las instituciones escolares en las que se aplicaron los proyectos finales de esta Especialización, la experiencia fue algo novedosa y disruptiva en cuanto a las metodologías, técnicas y conocimientos

que se pusieron en juego. En este sentido, resulta relevante considerar que se trabajaron contenidos propios de una disciplina como la programación y de una ciencia como la computación, que, como dijimos con anterioridad, en la mayoría de las escuelas no cuenta con un espacio curricular propio[2].

6. Incidencia de las prácticas docentes en las escuelas

En términos generales, dentro de las dinámicas que se vieron transformadas, se pueden considerar algunas propias de la estructura escolar como el tiempo y el espacio. Tomando en consideración estos dos aspectos, observamos que si bien los trabajos finales tendieron a desarrollarse en el aula también se llevaron a cabo proyectos que propusieron nuevas disposiciones tanto en el rol de docentes como de estudiantes, habilitando la creación de programas computacionales por parte de éstos/as. Muchas de las propuestas desbarataron la estructura habitual de la práctica de enseñanza-aprendizaje junto a los/as cursantes: se buscaron y gestionaron otros espacios que no son el aula (laboratorios, bibliotecas, patios, SUM); se pactaron tiempos de trabajo que desarticularon la lógica modular de los cuarenta minutos de clases: negociaron horarios con el equipo de gestión y con los/as colegas con quienes construyeron cada propuesta de enseñanza, propusieron actividades extracurriculares, entre otras.

En una investigación que se realizó en escuelas técnicas con orientación en programación de la ciudad de Córdoba, Echeveste y Martínez (2021) indican que: "El trabajar mediante proyectos hace que el tiempo y su distribución tensionen el formato de una clase tradicional de 80 minutos. La actividad propuesta aparece como protagonista en la posibilidad de administrar los tiempos de las materias de programación, en base a las necesidades, los tiempos y los aprendizajes estudiantiles".

A continuación, realizaremos un breve recorrido sobre los contenidos en los cuales se trabajaron, con qué metodologías y en qué tiempos y espacios curriculares se enmarcaron algunos de los proyectos que fueron presentados como trabajos finales en la especialización.

7. Evaluación de la especialización

Realizamos una evaluación de los resultados obtenidos del dictado de la especialización a partir del análisis de los trabajos finales de la misma y una encuesta realizada a la totalidad de los cursantes que terminaron la especialización.

En general, si tomamos los trabajos finales (TF) escritos por los/as docentes, encontramos relatos exitosos de la puesta en práctica de sus secuencias didácticas. En la gran mayoría de los casos, según sus propias palabras, lograron cumplir con el objetivo de "enseñar a programar"[3] a sus grupos de estudiantes, y, de poner en juego en clase determinados conceptos, prácticas y perspectivas propios del pensamiento computacional (Brennan, Resnick, 2012). De todos modos, no en todos los casos se logra una metareflexión clara acerca de cómo éstos ingresaron al aula aunque puedan inferirse en la descripción de la experiencia. Así, son frecuentes las apariciones (explícitas o inferidas) en los distintos proyectos de conceptos como **ciclos, eventos, condicionales, secuencias; de prácticas como ensayar y depurar, reusar y remezclar; y, perspectivas como conectar, expresar y preguntar**.

[2]La mitad de los trabajos (49 %) fueron llevados adelante en espacios curriculares afines a la programación mientras que el 35 % fue hecho en espacios no relacionados a la programación/informática. El resto se divide entre espacios extracurriculares o espacios no afines pero integrados a un proyecto institucional general.

[3]La mayoría de los trabajos tenía como objetivo "Desarrollar el pensamiento computacional".

Resulta importante destacar, además, el valor que se le da al trabajo con determinadas prioridades pedagógicas (Ministerio de Educación de Córdoba) que parecieran tener estrecha relación con algunos de los procedimientos del pensamiento computacional: resolución de situaciones problemáticas, pensamiento crítico y creativo, y trabajo en colaboración. En relación a esto, en la encuesta de final de cursada, el 70,6 % menciona que los contenidos vistos en la especialización *"Son abordables de manera transversal como parte de un proyecto curricular institucional enmarcado en el desarrollo de capacidades fundamentales promovido por el Ministerio de Educación"*. A esto le sumamos la reiterada mención al aprendizaje utilizando como estrategia el **ensayo y error**. Y, en esta lógica también, aparecen los/as docentes enseñando y aprendiendo desde ese lugar: permitiéndose el error, la consulta grupal, el "averiguar para la próxima clase" alguna situación que se presentó y no estaba planificada. En este sentido, es importante mencionar que los/as docentes coinciden en que enseñar programación y exponer a los/as estudiantes a prácticas como **"ensayar y depurar"** contribuyen a nivel emocional al manejo de la frustración, visibilizando al "error" como una instancia de aprendizaje y superación.

Al ser consultados/as en la encuesta final sobre qué destacaban de la experiencia con los/as estudiantes, las respuestas que más se repiten no provienen de los campos disciplinares de la didáctica o la programación, sino que tienen relación con lo actitudinal. El entusiasmo y el interés de los y las estudiantes son los más destacados con 29 menciones. Este punto nos da luces de que, a pesar de que no son los objetivos principales de la especialización, son muy valorados por quienes están al frente del aula. Además, cabe señalar que 15 de estas respuestas fueron destacadas por docentes de escuelas que no habían trabajado previamente programación y 14 donde sí se había realizado. Otro de los aspectos más relevantes para ellos/as fue la capacidad de **colaborar y trabajar en equipo** con 14 menciones.

Entonces, si tomamos como evidencia de aprendizaje los TF, sobre todo los apartados en los que los/as docentes debían realizar un *racconto* acerca de aquellos conceptos, prácticas y perspectivas del pensamiento computacional (Brennan, Resnick, 2012) que entraron en juego en sus secuencias didácticas, podemos decir que en todas se menciona un corrimiento importante de los roles[4], un cambio general en las dinámicas de clase, el cuestionamiento del concepto de "nativo digital" y una profunda transformación en ellos/as mismos/as como docentes. En la reflexión final de los TF son frecuentes las menciones a cómo el cursado de la especialización no sólo les permitió aprender sobre programación sino también cómo modificó la forma de mirar a sus estudiantes, pensar sus clases y el vínculo de enseñanza-aprendizaje. De hecho, el 84 % de los/as cursantes en la encuesta de final de cursada, ante la pregunta "El cursado de esta especialización, ¿modificó tu concepción acerca de la forma de abordar la enseñanza de la computación en las escuelas?", mencionan que les gustaría incorporar algunos conceptos y/o herramientas didácticas en sus clases.

Un cambio que destacan los/as docentes cursantes en sus prácticas pedagógicas, producto de una transferencia de lo aprendido durante el cursado, a las aulas en que implementaron sus trabajos finales, es la posibilidad de haber vivenciado, en el tránsito por los distintos módulos de la especialización, momentos de **exploración, investigación y experimentación** propios de un formato curricular de laboratorio y de un enfoque **de aprendizaje por descubrimiento**. La principal implicancia que esto tiene es reconocer que la enseñanza de las Ciencias de la Computación contribuye al desarrollo de cierta autonomía por parte del estudiante ante los desafíos que se presentan y del ejercicio por parte del docente de un rol de guía, orientando ante la incertidumbre, conteniendo ante el error y colaborando en el replanteo de la estrategia para el abordaje y resolución de las situaciones problemáticas.

[4]El docente se sale del rol del control absoluto, del que lo sabe todo para proponer una experiencia de aprendizaje en la que se puede aprender colectivamente; los/as estudiantes aparecen con un rol más activo, en el relato aparecen "al frente quienes estaban al fondo", con interés los desinteresados de siempre, activos quienes estaban pasivos y al margen.

En términos de las dificultades encontradas y narradas por los/as docentes en los TF encontramos que las trabas en el trabajo de integración de la programación en las instituciones, más allá de condiciones de infraestructura, tienen que ver - entre otras causas - con la disposición del equipo de gestión en garantizar los tiempos y los espacios para que los proyectos tomaran curso en las escuelas: esto llevó a algunos recortes de tiempos en los proyectos y de trabajo en determinados espacios curriculares y no otros. En este sentido, también encontramos en los TF la dificultad de insertar la programación en espacios curriculares que "naturalmente"no son afines como el caso de las artes, la lengua y las ciencias sociales. Presentar un proyecto interdisciplinario que implique ocupar horas de estos espacios curriculares en un proyecto vinculado a la programación parecía no ser bien recibido por parte de las autoridades escolares aunque, además de aprendizajes propios de la programación, hubiera aprendizajes relacionados con estos espacios involucrados.

Al respecto, es interesante recuperar una de las causas enunciadas en el Informe Final de Evaluación de Procesos y Resultados, llevado a cabo en el marco del dispositivo de evaluación de las especializaciones que la Fundación Sadosky desarrolló en las distintas provincias de Argentina. Dicho informe expresa: “se presentan algunos interrogantes en torno a la apropiación efectiva de algunas estrategias, conceptos y perspectivas relacionados a la ciencias de la computación, que quedaron expresados en los trabajos finales. La principal preocupación que surge de parte de los profesores -aunque no se trata de algo generalizado- se vincula con ciertas dificultades para pasar de una perspectiva de uso de TIC a una vinculada a las ciencias de la computación” (Scasso, M. Cura, D. Marino, V. Kaplan, L., 2019). En este sentido, nos parece importante ampliar lo aquí citado explicitando que desde la especialización siempre se promovió el desarrollo de proyectos que partiendo de un problema computacional, pusieran en juego los conceptos, prácticas y perspectivas de las Ciencias de la Computación. De la lectura de los trabajos finales, es posible identificar que la mayoría de las propuestas pedagógicas implementadas y analizadas recuperaron este sentido. Otro grupo de proyectos partió de problemáticas sociales que permitían un abordaje interdisciplinario donde los saberes computacionales pudieron integrarse sin dificultad y, por último, un número reducido de proyectos donde la programación y la informática actuaron como soporte para la construcción de saberes propios de otras disciplinas.

Otra posible barrera detectada, especialmente para los proyectos que incluyeron programación en un lenguaje de alto nivel, es el debate sobre si se deben incluir, o no, este tipo de lenguajes tal como se realizó con Python en M6 y M7 (ver Tabla 1, pag. 7). “Algunos argumentos para optar por lo segundo son, por ejemplo, que la programación de alto nivel no es un objetivo que se persigue en la escuela o, simplemente, que una especialización docente no es apropiada para abordar tal desafío, dado su envergadura” (Acosta, Martínez y Rojo, 2019).

En cuanto al impacto directo de la especialización en el diseño y planificación de sus clases, la mayoría acuerda en que lograron incorporar a sus clases contenidos y herramientas abordados en ésta, además de permitirles actualizarse en debates y herramientas de la didáctica de la programación. Asimismo, en relación a los objetivos definidos por la especialización (y que enunciamos más arriba) la mayoría de los/as egresados/as menciona haber logrado completamente todos los objetivos (ver Figura 3).

8. Conclusiones y trabajo futuro

En términos generales podemos afirmar que se lograron los objetivos que nos planteamos al diseñar la especialización. Además, en relación a los objetivos propios en relación a la enseñanza de la programación, podemos mencionar que la implementación de la especialización contribuyó a que los/as docentes pudiesen diseñar propuestas didácticas que promueven el aprendizaje de conceptos, prácticas y perspectivas computacionales en escuelas donde la enseñanza

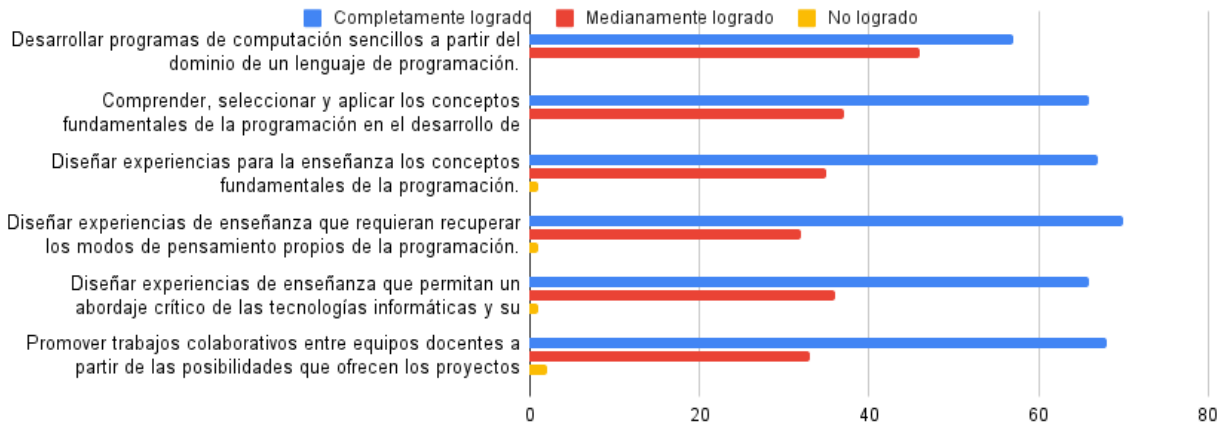

Figura 3: Cumplimiento de objetivos de la especialización. Percepción de las/os egresadas/os a través de una encuesta de evaluación.

de la programación es un asunto pendiente. Asimismo, los/as docentes pudieron experimentar y poner en práctica formatos curriculares que colaboran con una forma de aprendizaje vinculada a la experimentación y descubrimiento; con incidencia en la organización del tiempo y el espacio escolar, así como también en el rol que asumen los docentes y estudiantes. Estos resultados nos inducen a pensar que el diseño de la carrera a partir de los objetivos, y con una organización diseñada por proyectos presenta una alternativa interesante para ser abordada en otras experiencias.

Como trabajo futuro, estamos avanzando en la sistematización de los materiales de la especialización para su publicación con licencias libres, aprovechando que debido al dictado semipresencial, ya se cuenta con material producido y trabajado en el aula. Como trabajo de investigación, nos interesa evaluar cuán efectiva es esta especialización como instrumento de formación docente, para el dictado de la programación y las Ciencias de la Computación en la escuela, de docentes sin formación previa en esta disciplina. Para ello se pretende relevar entre los/as egresados/as, los espacios curriculares en los que se desempeñan, el grado de transformación en sus prácticas y algunas consecuencias en el aula.

Bibliografía

Acosta, A., Martínez, M.C., Rojo, C. (2019). “Enseñando Python en una propuesta de formación docente en enseñanza de la programación”. Congreso Argentino de Ciencias de la Computación. Río Cuarto, Argentina.

Brennan, K., y Resnick, M. (2012). “Entrevistas basadas en artefactos para estudiar el desarrollo del Pensamiento Computacional (PC) en el diseño de medios interactivos”. Documento presentado en el encuentro anual de la American Educational Research Association, AERA 2012, Vancouver, BC, Canada.

Echeveste, María Emilia y Martínez, María Cecilia (2021). El tiempo en las clases de programación: “Hacés y hacés y no te importa si tocó el timbre”. Revista de Sociología de la Educación-RASE , 14 (3) , 307-324. http://dx.doi.org/10.7203/RASE.14.3.21411.

Elmore, R. (1979). Backward Mapping: Implementation Research and Policy Decisions. The Academy of Political Science.

Gómez, Marcos. (2020). Aspectos de adquisición de lenguaje en la enseñanza de programación. Tesis de Doctorado. Universidad Nacional de Córdoba.

Resolución CFE Nº 343/18 (2018). Fecha de promulgación Septiembre 12/2018.

Scasso, M. Cura, D. Marino, V. Kaplan, L. (2019). “Especialización en Didáctica de las Ciencias de la Computación. Evaluación de procesos y resultados.” Fundación Quántitas. Fundación Sadosky - Iniciativa Program.Ar

SIMPOSIO LATIONOAMERICANO

Pensamiento computacional en educación primaria: el caso de Uruguay
Víctor Koleszar, Ana Laura Pérez Espagnolo y Emiliano Pereiro

El Pensamiento Computacional en el sistema educativo público Costarricense
Andrés Rodríguez y Natalia Zamora

Currículo de referência em tecnologia e computação: uma proposta do centro de inovação para a educação Brasileira
Christian Brackmann, André Raabe y Alice Carraturi

Trasfondo del Desarrollo de una Propuesta de Formación en Estándares TIC en Paraguay desde el Portal META de Paraguay Educa
Sascha Rosenberger y Carla Fernández

Educación a distancia. Reto de la superación a través del curso Algoritmización con Scratch
Rosa María Figueredo Rodríguez y Yor Alex Remond Recio

Proyecto IdeoDigital: Implementando ciencias informáticas en el sistema escolar público chileno
Andreas Hein, Claudia Jaña y Catalina Lyon

Pensamiento computacional en educación primaria: el caso de Uruguay

Víctor Koleszar
vkoleszar@ceibal.edu.uy
Plan Ceibal (Uruguay)

Ana Laura Pérez Espagnolo
alperez@ceibal.edu.uy
Plan Ceibal (Uruguay)

Emiliano Pereiro
epereiro@ceibal.edu.uy
Plan Ceibal (Uruguay)

Resumen

En los últimos años ha crecido el impacto de las tecnologías de la información y su presencia en la vida cotidiana y en la educación. En este marco existe un movimiento para incluir al pensamiento computacional formalmente en los sistemas educativos.

Desde Plan Ceibal se impulsa el trabajo en pensamiento computacional, como marco para la aproximación a las ciencias de la computación desde la educación primaria y secundaria, con un enfoque centrado en la resolución de problemas, y con la intención de aprovechar su potencial para educar usuarios y creadores de tecnología

Este artículo aborda la experiencia de implementación del programa Pensamiento Computacional en el territorio uruguayo, sus características, particularidades y aprendizajes en diferentes líneas de trabajo. Algunos de los elementos que se reconocen como centrales y claves en el éxito de este proceso son: la formación y participación docente; la articulación con los subsectores del sistema educativo y las alianzas con organizaciones vinculadas a la temática; el trabajo en la definición y contextualización de los contenidos y prácticas a abordar desde el programa; y el acompañamiento en la implementación en los centros educativos.

Palabras clave: Pensamiento computacional, K-12, Ciencias de la Computación.

1. Introducción

En los últimos años ha crecido de manera considerable el impacto de las tecnologías de la información y su presencia en la vida cotidiana. La educación no está aislada de este fenómeno, por lo que también ha cambiado la forma en la que accedemos al conocimiento, buscamos información, nos relacionamos, aprendemos y enseñamos (redes sociales, plataformas educativas, inteligencia artificial). En este marco algunos autores creen en la necesidad de conceptualizar el pensamiento computacional como una nueva competencia a promover a nivel mundial (Wing, 2006), ya que presenta conceptos clave en el aprendizaje de la ciencia, la tecnología, la ingeniería y las matemáticas de la actualidad, y que no se acota a la programación (Kong y Abelson, 2019).

Además, los alcances y definición del pensamiento computacional abren grandes debates, que no son objeto a tratar de este artículo, y que han dado lugar a numerosas publicaciones. Plan Ceibal considera al pensamiento

computacional como la habilidad para reconocer aspectos del mundo real que pueden ser modelados como problemas y para diseñar y evaluar soluciones algorítmicas que puedan ser implementadas computacionalmente (Fraillon et al., 2019). De alguna manera es el marco para la aproximación a las ciencias de la computación desde la educación primaria y secundaria, con la intención de aprovechar el potencial del pensamiento computacional para educar usuarios y creadores de la tecnología.

Las experiencias de implementación de un currículum de pensamiento computacional o ciencias de la computación alrededor del mundo son muy diferentes y con características muy particulares de acuerdo al país, la región y el sistema educativo en el que están inmersos. La popularidad del término pensamiento computacional ha llevado a que muchos países introduzcan elementos asociados en el currículum de la escuela primaria (Bocconi et al., 2016; Brown et al., 2014; Duncan et al., 2017), como marco para promover distintas habilidades y contenidos, desde la alfabetización digital hasta las ciencias de la computación, incluyendo temáticas cómo STEAM, programación, robótica o resolución de problemas (NRC, 2010).

Desde 2017 Plan Ceibal en conjunto con la Administración Nacional de Educación Pública[1] (ANEP) llevan adelante un programa específico de intervención en pensamiento computacional en la educación primaria. El programa llamado Pensamiento Computacional (PC) se basa en el exitoso modelo de Ceibal en Inglés[2], e implica un docente, que se conecta de manera remota a un equipo de videoconferencia, que trabaja en dupla pedagógica con el docente de aula una vez a la semana en secuencias didácticas interdisciplinarias de PC con otras áreas del conocimiento. Es decir, se trabajan contenidos específicos de pensamiento computacional vinculados a la matemática, a las ciencias, a la lengua, a la educación física, etc. El trabajo en este programa es completamente voluntario y la participación ha pasado de 50 grupos en 2017 a 1764 en 2021. La comunidad docente ha adoptado este modelo para trabajar en sus aulas.

En este artículo se aborda la experiencia de implementación del programa de Pensamiento Computacional de Plan Ceibal en el sistema educativo uruguayo. Algunos de los elementos que se reconocen como centrales y claves en el éxito de este proceso son: la formación y participación docente; la articulación con los subsectores del sistema educativo y las alianzas con organizaciones vinculadas a la temática; el trabajo en la definición y contextualización de los contenidos y prácticas a abordar desde el programa; y el acompañamiento de cercanía en la implementación en los centros educativos.

2. Plan Ceibal

El Plan Ceibal[3] de Uruguay es creado como agencia de innovación en el año 2007 asociado a la iniciativa One Laptop Per Child[4]. Actualmente en el sistema público uruguayo, los estudiantes de Primaria y hasta tercer año de Educación Media reciben un dispositivo tecnológico personal, así como los docentes. El 100 % de los centros educativos públicos cuentan con conexión a internet y la mitad cuentan con un equipo de videoconferencia instalado. Plan Ceibal también provee mantenimiento de la red y un sistema de reparación desplegado en todo el país para los dispositivos entregados.

[1] https://www.anep.edu.uy/

[2] https://ingles.ceibal.edu.uy/

[3] www.ceibal.edu.uy/en/institucional

[4] http://one.laptop.org/

Luego de una fase centrada en proveer recursos y tecnología, hoy Ceibal trabaja, en conjunto con la ANEP, enfocado en promover recursos, programas educativos y formación que vinculan la educación con la tecnología de formas innovadoras.

El programa Pensamiento Computacional recoge algunas experiencias de otros programas de Plan Ceibal como Ceibal en Inglés. El modelo de Ceibal en Inglés[5] hace énfasis en tres aspectos pedagógicos: un docente remoto que se integra al aula; enseñanza en equipo, aprendizaje combinado.

3. El programa Pensamiento Computacional en educación primaria

A partir de una iniciativa conjunta de Plan Ceibal y la ANEP, en 2017 comenzó un proyecto piloto de enseñanza de pensamiento computacional en 30 centros educativos de primaria. Actualmente el programa PC ha sido adoptado por la comunidad educativa, y alcanza a 36 mil niños y niñas distribuidos en casi el 60 % de las escuelas públicas urbanas del país. La participación en el proyecto es optativa, y está dirigido a docentes del segundo ciclo de educación primaria (con grupos de estudiantes de 4to a 6to grado, es decir entre 9 y 12 años).

El modelo de trabajo implica clases semanales de 45 minutos, en horario curricular, con la participación del docente de aula y un docente remoto de pensamiento computacional que se conecta a un equipo de videoconferencia de cada centro educativo. De esta manera se puede lograr potencialmente cobertura en todas las escuelas del país, con docentes remotos formados en temáticas de ciencias de la computación. El pensamiento computacional se propone como un eje transversal, es decir que se trabajan contenidos específicos de pensamiento computacional y programación de forma integrada a diferentes áreas de conocimiento del currículum nacional (ej: lengua, matemática, ciencias, educación física).

En este marco se busca que los estudiantes desarrollen los contenidos fundamentales de las ciencias de la computación y aprendan nuevos enfoques para la resolución de problemas. Las propuestas de PC abordan 5 grandes dimensiones en torno al pensamiento computacional, que sintetizan y recogen experiencias locales y marcos referenciales de otros países en la temática (College Board, 2020; Juškevičienė y Dagienė, 2018; Raabe, et al., 2018; K–12 CS Framework, 2016; Csizmadia et al., 2015; ISTE y CSTA, 2011; NRC, 2010). Estas dimensiones son: comunicación y colaboración; computación, sociedad y equidad; problemas computacionales; datos y abstracciones; algoritmos, programas y dispositivos. Entrelazadas a las dimensiones se suman de forma esquemática las siguientes habilidades: abstracción, descomposición, generalización, evaluación y pensamiento algorítmico (Dagienė y Sentance, 2016).

El programa PC está organizado en propuestas pedagógicas para 3 niveles, en los que se recorre de forma incremental y secuencial los contenidos y competencias de pensamiento computacional. Las propuestas pedagógicas abarcan un conjunto de iniciativas y materiales para cubrir diferentes aspectos del trabajo en el aula, en este sentido se componen de una secuencia didáctica de actividades de aprendizaje, un curso de formación para docentes, materiales y recursos para la plataforma virtual (LMS), y una evaluación final para estudiantes.

Para cada nivel se desarrollaron varias secuencias didácticas en unidades en formato proyecto, que de forma interdisciplinaria integran el pensamiento computacional con otras áreas del conocimiento. Entre otras potencialidades esta modalidad permite abordar problemas más generales a la vez que el docente de aula se ve motivado a incluir estos proyectos ya que trabaja contenidos curriculares de forma innovadora. En estos proyectos se propone el diseño y

[5]Modelo de Ceibal en Inglés recuperado de https://n9.cl/kzxoq

construcción de un dispositivo o programa con una temática a elección del docente de aula.

Estas secuencias entrelazan actividades curriculares con pensamiento computacional, de modo que la formación de una dupla pedagógica de trabajo entre docente remoto y docente de aula es fundamental. En los acuerdos se establece que las propuestas se pueden adaptar y contextualizar a la dinámica de cada aula y las necesidades específicas de los estudiantes. Para cada proyecto los docentes cuentan con una guía de preguntas claves, y actividades con espacios de desafíos y exploración acompañados de un andamiaje cercano, donde se busca ayudar a organizar lo aprendido en ideas y estrategias de pensamiento cada vez más potentes (Furman, 2016). Se combinan estas actividades con espacios de conceptualización y reflexión, que promueven el tomar consciencia de lo aprendido, lo que saben y las estrategias que se ponen en juego en el hacer.

Se busca que los docentes de aula puedan realizar transposiciones didácticas que permitan a los estudiantes resolver problemas relacionados con Matemática, Ciencias Sociales o Naturales, Lengua, etc., o asociado a situaciones de la vida real, a través de las habilidades y competencias que desarrolla el Pensamiento Computacional.

Cada una de estas secuencias cuenta con recursos y actividades en la plataforma LMS, que se trabajan durante la clase con la presencia del docente remoto, en otros espacios definidos por el docente de aula, y/o las realiza el estudiante de forma independiente. En el asesoramiento curricular y desarrollo de materiales didácticos y contenidos, el programa PC ha contado con el trabajo de la Fundación Sadosky[6] desde 2019 a esta parte.

Otras de las actividades que realizan estudiantes y docentes como parte de los contenidos de PC, es la participación en el Desafío Bebras[7] para la promoción del pensamiento computacional y la computación. Desde 2020, Ceibal se suma a esta iniciativa internacional y organiza la edición uruguaya. Durante aproximadamente un mes se realizan actividades de preparación donde se comparten y enseñan estrategias para la resolución de problemas y luego durante tres semanas se pone a disposición el desafío de ese año que realizan los estudiantes individualmente. Las actividades pretenden contribuir a la conformación y enriquecimiento de la comunidad docente, a la vez que son un insumo para la investigación y evaluación en habilidades de pensamiento computacional.

4. Comunidad de Pensamiento Computacional

Durante estos años de trabajo se viene construyendo una comunidad de docentes, profesionales y otros actores educativos con interés por el pensamiento computacional. Desde el equipo de Plan Ceibal, se desarrollan diversas estrategias para promover el encuentro, la reflexión y el diálogo y así continuar con la formación de la comunidad.

La implementación del programa en los centros educativos ha sido diseñada de manera tal que cada docente esté acompañado por un referente del proyecto de PC que es a su vez un referente regional. Ello posibilita que se puedan promover y desarrollar líneas de trabajo en conjunto con los distintos actores u organizaciones de la región. La alianza con cada subsistema educativo en territorio, genera una mirada integral del proyecto anclado en las prácticas educativas, en los contenidos curriculares y en el contexto local, y por otro lado genera retroalimentación en el desarrollo de las secuencias didácticas.

El acompañamiento en territorio abarca aspectos administrativos (inscripciones, problemas de horarios, solicitudes, recambio de dispositivos), técnicos (problemas con dispositivos tecnológicos, videoconferencias, conexión a la

[6] www.fundacionsadosky.org.ar

[7] www.bebras.org

red), pedagógicos (asesoramiento de cómo integrar los distintos contenidos curriculares a las propuestas pedagógicas, el desarrollo de proyectos de centro o áulicos que integran pensamiento computacional), formativos (talleres a demanda de docentes y estudiantes sobre conocimientos específicos), entre otros.

La formación para los docentes consta de varias líneas que pueden realizarse tanto en conjunto cómo individualmente de manera voluntaria, y que están principalmente dirigidas hacia docentes en general, sin el requisito de contar con formación específica en temáticas de las ciencias de la computación.

La implementación del programa en el aula es un proceso formativo en sí mismo para los docentes que participan. Cada docente recorre una propuesta pedagógica con secuencias didácticas que junto al docente remoto define cómo adaptar e integrar nuevos elementos. Durante este proceso, el docente reflexiona y vivencia cómo se desarrolla el pensamiento computacional en el aula.

Como parte de las propuestas pedagógicas y asociado a cada secuencia didáctica se ofrece un curso virtual combinado. El docente trabaja durante 4-5 semanas en uno de los proyectos, comprendiendo cuál es el objetivo, cómo integrar otros elementos (contenidos, intereses de los estudiantes, vínculo con otros proyectos que se estén desarrollando, etc.), cómo trabajar con la tecnología asociada a ese proyecto, entre otros. A su vez, cuentan con un encuentro sincrónico semanal donde se desarrollan en profundidad algunos aspectos del curso así como las dudas existentes por parte de los participantes.

Por otro lado, las propuestas de formación docente también incluye encuentros tanto a nivel nacional como regional que promueven espacios de aprendizaje, reflexión y diálogo entre pares. Estos encuentros docentes se pueden desarrollar tanto en formato de webinar, así como de manera presencial. Los encuentros son planificados teniendo en cuenta las necesidades de formación en territorio, la participación de expertos y el cronograma anual del proyecto. Al cierre del año se realiza un seminario que cuenta con distintas autoridades educativas, expertos y experiencias de aula que relatan los propios protagonistas: docentes y estudiantes.

Otro rol fundamental en la comunidad educativa de PC y que colabora con el trayecto formativo de los docentes de aula, es el del docente remoto. Trabajamos con instituciones y organizaciones que tienen experiencia probada en la docencia de computación, robótica o programación. Plan Ceibal se asocia con instituciones que cuentan en sus equipos con docentes formados en la temática que incorporan los lineamientos del programa PC, y participan también de las actividades antes mencionadas. Los docentes remotos de las distintas instituciones son locales o extranjeros (hasta el momento de Argentina o España), con perfiles de docentes formados en computación, o de profesionales del área de las ciencias de la computación interesados y con experiencia en la educación.

Siguiendo con la línea anterior, las alianzas que el programa PC realiza han sido fundamentales para su crecimiento y desarrollo. Además de las instituciones que nuclean a los docentes remotos, existen otras sinergias que a distintos niveles potencian el trabajo. Esto tiene que ver con la coordinación con otros programas de Plan Ceibal, con alianzas y vínculos con distintos institutos, organismos y centros del propio sistema educativo, sus subsistemas y del Ministerio de Educación y Cultura[8], con coordinación con distintas universidades; y con actores locales y extranjeros vinculados a la temática.

[8] www.gub.uy/ministerio-educacion-cultura

5. Lecciones aprendidas

Durante estos pocos años de implementación el programa ha sufrido transformaciones y avances. Seguramente también parte del relativo avance de PC tiene que ver con aprender de los errores de la implementación. En este sentido podemos identificar dos etapas del programa: en la primera etapa una apuesta a convocar masivamente docentes de todos los niveles (educación inicial, primaria y media), con propuestas introductorias al pensamiento computacional fundamentalmente desconectadas[9] y con énfasis en abordar problemáticas locales utilizando la tecnología en un sentido amplio; y una segunda etapa, la actual y descrita anteriormente, con énfasis en grupos del segundo ciclo de Primaria (grados 4to a 6to), con contenidos graduados en niveles y proyectos centrados en materializar algún tipo de dispositivo o programa integrando contenidos curriculares. La primera etapa logró un desembarco fuerte, pero también puso de manifiesto claras dificultades para abordar un programa a esta escala. Con poco foco, el desarrollo de contenidos era en ocasiones ambiguo en los objetivos que se proponía sobre logros y aprendizajes de los estudiantes.

El marco de implementación como iniciativa de Plan Ceibal también tiene sus ventajas y desventajas. Si bien el Plan es parte del sistema educativo, la implementación de programas que impactan directamente en el aula es una novedad, que implica y demanda grandes esfuerzos en la coordinación y en los acuerdos con todos los niveles de la ANEP, para poder llegar con sinergia al colectivo docente. Esta hibridez de Ceibal, le permite al programa PC iterar rápidamente sobre el proceso de implementación e ir mejorando.

Además, el programa al ser voluntario tiene la gran ventaja de no presentar las resistencias que podría plantear una reforma exo generada. La resistencia a los cambios en las escuelas es objeto de estudio por parte de diversos autores tanto en América Latina como en el resto del mundo, ya que los conflictos que se generan a partir de las reformas en educación no son exclusivos de nuestro continente. La pregunta acerca de por qué parece imposible reformar los sistemas educativos, es abordada, entre otros, por Dubet (2007) para la realidad de su país, Francia, donde según el autor, hace años que los docentes y alumnos se movilizan en contra de todo intento de reforma, sin importar el tinte político de la administración.

La intervención de PC aún no se enfrenta a estas resistencias, en parte quizás, debido al carácter optativo, y esto permite que se vaya construyendo una política educativa desde el colectivo docente. Así, se busca que los docentes adapten y realicen cambios en su forma de enseñar convencidos de los beneficios de los mismos respecto de sus metodologías actuales (Buckinham citado en Carriego y Carriego, 2011). En esta perspectiva se pone en juego la cuestión de la motivación y como consecuencia de ello, el compromiso con el programa. Pero este modelo voluntario tiene un límite en su expansión, y se requiere de una decisión de política educativa para poder lograr la universalización en todo el sistema, que a su vez esté acompañada del cumplimiento de las condiciones necesarias para llevar adelante tal cometido.

La apropiación del programa y el impacto en los aprendizajes implica docentes formados y capacitados en la temática. La apuesta de PC en este sentido es generar diferentes espacios y diversos formatos de formación, según las estrategias relatadas anteriormente. Se busca que los docentes encuentren, de acuerdo a su disponibilidad, intereses y experiencias, distintas propuestas con grados de profundidad adecuados y con el desafío de la coordinación continua con otros actores que trabajan en la temática.

[9] *actividades unplugged*

6. Perspectiva a futuro

El programa PC viene en proceso de crecimiento en varias de las líneas señaladas. Parte de este crecimiento tiene que ver con la adopción por parte de docentes y estudiantes, que integran la comunidad educativa. El programa se ha formalizado, ha ganado profundidad y está en proceso de estructurar mucho de su quehacer (elaboración de un marco curricular y las secuencias de aprendizaje por niveles, generar evaluaciones en el Sistema de Evaluación de Aprendizajes nacional[10], etc), y quizás a raíz de todo esto hoy está en un momento de quiebre o inflexión.

Aparecen en este contexto, la posibilidad de ir hacia la universalización del programa en la educación pública y la integración a la currícula formal. Y con esto la necesidad de contar con trayectorias de formación con las temáticas y el enfoque pedagógico que aborda el programa, en diálogo y coordinación con las propuestas de formación en informática ya existentes.

Bibliografía

Brown, N. C., Sentance, S., Crick, T., y Humphreys, S. (2014). Restart: The resurgence of computer science in UK schools. ACM Transactions on Computing Education (TOCE), 14(2), 1-22.

Bocconi, S., Chioccariello, A., Dettori, G., Ferrari, A., Engelhardt, K., Kampylis, P., y Punie, Y. (2016). Developing computational thinking in compulsory education. European Commission, JRC Science for Policy Report, 68.

Carriego, E., y Carriego, C. (2011). Los Desafíos de la "Revolución Lenta". Trabajo monográfico. Premio ABA 2010/ 2011

Dagienė, V., y Sentance, S. (2016). It's computational thinking! Bebras tasks in the curriculum. In International conference on informatics in schools: Situation, evolution, and perspectives (pp. 28-39). Springer, Cham.

Dubet, F. (2007). ¿Por qué parece imposible reformar el sistema escolar francés? Revista Mexicana de Investigación Educativa, 12 (32).

Duncan, C., Bell, T., y Atlas, J. (2017, January). What do the teachers think? Introducing computational thinking in the primary school curriculum. In Proceedings of the Nineteenth Australasian Computing Education Conference (pp. 65-74).

College Board. (2020). AP Computer Science Principles. Course And Exam Description. Recuperado en https://apcentral.collegeboard.org/pdf/ap-computer-science-a-course-and-exam-description.pdf?course=ap-computer-science-a

Csizmadia, A., Curzon, P., Dorling, M., Humphreys, S., Ng, T., Selby, C., y Woollard, J. (2015). Computational thinking-A guide for teachers.

Fraillon, J., Ainley, J., Schulz, W., Duckworth, D., y Friedman, T. (2019). IEA International Computer and Information Literacy Study 2018 assessment framework. Springer.

Furman, M. (2017). Educar mentes curiosas: la formación del pensamiento científico y tecnológico en la infancia. Buenos Aires. Disponible en https://www.educ.ar/recursos/131992/educar-mentes-curiosas-de-melina-furman

ISTE y CSTA (2011). Computational Thinking in K–12 Education leadership toolkit. Computer Science Teacher Association

Juškevičienė, A., y Dagienė, V. (2018). Computational thinking relationship with digital competence. Informatics in Education, 17(2), 265-284.

K–12 Computer Science Framework. (2016).

Kong, S. C., y Abelson, H. (Eds.). (2019). Computational thinking education. Springer.

National Research Council. (2010). Report of a workshop on the scope and nature of computational thinking. National Academies Press.

[10] https://sea.anep.edu.uy/

Raabe, A. L. A., Brackmann, C. P., y Campos, F. R. (2018). Currículo de referência em tecnologia e computação: da educação infantil ao ensino fundamental. Centro de Inovação para a Educação Básica-CIEB.

Shute, V. J., Sun, C., y Asbell-Clarke, J. (2017). Demystifying computational thinking. Educational Research Review, 22, 142-158.

Wing, J. M. (2006). Computational thinking. Communications of the ACM, 49(3), 33-35

El Pensamiento Computacional en el sistema educativo público Costarricense

(Computational thinking at Costa Rica's public education system)

Andrés Rodríguez*
andres.rodriguez@fod.ac.cr
Fundación Omar Dengo

Natalia Zamora*
natalia.zamora@fod.ac.cr
Fundación Omar Dengo

Resumen

Desde su creación en 1988, el Programa Nacional de Informática Educativa (PRONIE MEP-FOD), implementa en Costa Rica un modelo pedagógico orientado a desarrollar en los estudiantes competencias cognitivas y sociales de alto nivel, como el razonamiento lógico-matemático, la creatividad, la resolución de problemas y la colaboración, a través del aprendizaje desde y con la programación. Este programa fue creado mediante una alianza público-privada entre el Ministerio de Educación Pública (MEP) y la Fundación Omar Dengo (FOD). A partir de los procesos de investigación, en el 2014 el PRONIE MEP-FOD comenzó a repensar cómo actualizar su propuesta pedagógica desde la orientación de estándares de desempeño hasta el desarrollo de competencias (LIE con guías). Ahora se incluye el pensamiento computacional como una forma de pensar para el desarrollo de habilidades necesarias para participar activamente en la sociedad, como la resolución de problemas de programación(codificación), que permiten a los estudiantes comprender tanto los conceptos fundamentales de la informática como las grandes ideas que los unifican. Tras una fase de pilotaje, la propuesta pedagógica actualizada (LIE++ pensar, crear, programar) comenzó a implementarse en 2018 en las escuelas primarias. A finales de 2019, LIE ++ benefició a 388.258 estudiantes en 1020 escuelas primarias de todo el país. Los resultados de la evaluación mostraron que los estudiantes que participaron en LIE ++ obtuvieron mejores resultados en comparación con los estudiantes de LIE con guías.

Palabras clave: Pensamiento Computacional, Ideas Poderosas, Programación, Resolución de Problemas, Informática Educativa.

1. Introducción

La interacción constante entre el ser humano y las tecnologías ha permitido ampliar y modificar las formas de comunicarnos, relacionarnos, trabajar y aprender. El Movimiento Maker, la computación física, las habilidades

*Fundación Omar Dengo, Costa Rica.

del pensador computacional, el *open hardware*, el *open source* (programas de código abierto) y disciplinas STREAM (Ciencia, Tecnología, Robótica, Ingeniería, Arte y Matemática) por sus iniciales en inglés, son algunas de las tendencias mundiales que se abren paso en procesos educativos formales y no formales. Aunque no conocemos muchos de los trabajos del futuro, al analizar estas tendencias, es posible inferir los nichos de conocimiento y las competencias necesarias a desarrollar en los estudiantes para que puedan entender los conceptos claves de la tecnología con la que trabajarán mañana.

> ... las escuelas deberían enseñar los conceptos esenciales del pensamiento computacional y la tecnología digital. Aquellos que no entienden la naturaleza de los algoritmos corren mayor riesgo de ser manipulados por ellos y de perder el poder de la tecnología, en lugar de ser empoderados por ella. (Schleicher y Partovi, 2019).

Ante esta realidad el Programa Nacional de Informática Educativa (PRONIE) del Ministerio de Educación Pública (MEP) y la Fundación Omar Dengo (FOD) actualiza su propuesta educativa para la atención de estudiantes de Informática Educativa, implementada semanalmente, en condiciones regulares, a través de dos lecciones continuas de 40 minutos cada una, como parte del currículo nacional, las cuales tienen el objetivo de desarrollar habilidades de pensamiento de orden superior, acercando a los estudiantes a nuevas formas de pensar, resolver problemas e interactuar con otros, asegurando de esta manera la equidad en el acceso y la comprensión de las tecnologías digitales. Esta actualización mantiene el modelo pedagógico que el PRONIE MEP–FOD ha desarrollado desde 1988, que se fundamenta en el marco filosófico constructivista y de un quehacer construccionista que orienta la práctica pedagógica.

Hasta el 2017, la propuesta educativa era conocida como Laboratorios de Informática Educativa (LIE), posterior a ese año, a partir del replanteamiento conceptual producto de procesos constantes de investigación, es que cambia su nombre a LIE ++: pensar, crear, programar; incluyendo al acrónimo el doble más, que evoca a esa evolución conceptual y hace evidente habilidades claves de la propuesta como lo son, el desarrollo del pensamiento, la creación y el entender la tecnología mediante la programación. Se contribuye de esta manera a que los estudiantes pasen de ser consumidores a ser productores, capaces de resolver problemas utilizando las tecnologías a medida que las van conociendo y comprendiendo los fundamentos conceptuales detrás de ellas, de manera que puedan trabajar con las tecnologías actuales, así como tener los conocimientos claves para entender y poder trabajar con las del futuro.

2. Antecedentes

2.1. Informática educativa en el currículum nacional costarricense

Desde 1988, el PRONIE MEP–FOD ha venido desarrollando estrategias educativas, las cuales son implementadas por docentes especializados y formados en informática educativa, que permiten a los estudiantes el desarrollo de habilidades de orden superior, concibiendo las tecnologías digitales como herramientas de aprendizaje a través de espacios creativos y lúdicos, siendo la programación uno de los grandes pilares que orientan la práctica educativa en los más de 1.280 laboratorios de informática del PRONIE MEP–FOD a lo largo y ancho del país, abriendo oportunidades a más de medio millón de estudiantes de primaria y secundaria. Seymour Papert, indica que a través de la programación se ejerce una poderosa influencia sobre la manera de pensar de las personas, “a medida que los niños avanzan, programan la computadora para que tome decisiones más complejas y se encuentran involucrados en una reflexión sobre aspectos más complejos de su propio pensamiento” (Papert, 1987, p.43).

En la nota web “Pensamiento computacional: desafíos que debe enfrentar la escuela tradicional”, la programación se torna relevante en nuestros días, no solo para entender las tecnologías, si no para interactuar con ellas y comprender el mundo que nos rodea, “No solo hay que aprender programación, sino que también hay que entender cómo esta programación está relacionada con hechos concretos de nuestra vida como ciudadanos”. (Belluscio et al., 2020)

Por lo anterior, para el PRONIE MEP–FOD la programación es entendida como una manera de pensar y no solamente como el acto de codificar, ya que, desde esta concepción, implica otros elementos fundamentales como lo son el análisis del problema y su descomposición en partes menos complejas, que permitan visualizar su solución sin perder de vista “el todo” que lo origina y para lograrlo se debe recurrir a habilidades como la abstracción, el reconocimiento de patrones, el pensamiento algorítmico, las cuales son transferibles a cualquier área disciplinar e incluso a la vida.

2.2. Modelo Pedagógico

El PRONIE MEP–FOD desde sus inicios ha puesto su mirada en el desarrollo de capacidades de los estudiantes, a través de la resolución de problemas con programación, donde las tecnologías son vistas como una herramienta para pensar y es en este punto en el que se alinea con el pensamiento filosófico de Papert, quien considera a la computadora como un “objeto para pensar” (“objects to think with”). Estos objetos, pueden ser utilizados por un sujeto, para pensar sobre otras cosas, utilizando para ello su propia construcción de dicho objeto (Papert, 1987). Papert dice que creamos nuestro entendimiento del mundo al crear artefactos, experimentar con ellos, modificarlos y ver cómo funcionan. Esto hace al PRONIE MEP–FOD basar sus propuestas educativas en este quehacer construccionista propuesto por Papert, que orienta la práctica pedagógica para utilizar las tecnologías digitales como recursos para aprender y desarrollar habilidades, a través de un marco pedagógico constructivista propuesto por Jean Piaget en 1952, en el que define que el niño construye el conocimiento en su mente con base en su experiencia.

Bajo esta teoría y según indica Watkins (2003) se aprende trabajando de manera activa con materiales, reflexionando sobre y analizando la experiencia, pensando sobre el pensamiento (metacognición), aprendiendo sobre su propio aprendizaje (meta-aprendizaje) y asumiendo responsabilidad sobre él (autodirección), explicando lo aprendido a sí mismos o a otros, partiendo de lo concreto hacia lo abstracto y ligando los aprendizajes a los intereses de los educandos.

3. Pensamiento Computacional en el PRONIE MEP–FOD

El término de Pensamiento Computacional fue acuñado en los 80’s por Seymour Papert; actualmente en el ámbito mundial no se ha llegado a un consenso y continua el debate sobre qué es el pensamiento computacional, lo que explica que se encuentran varias posiciones al respecto. El PRONIE MEP–FOD ha seleccionado aquellos referentes que se alinean con su objetivo y visión pedagógica, para poder así definir lo que para el Programa es el pensamiento computacional, y se define como “la habilidad de resolución de problemas, que se basa en la programación de computadoras y en la comprensión de los fundamentos de la computación”.

4. Propuesta educativa LIE++: pensar, crear, programar

El PRONIE MEP-FOD mantiene una constante revisión y actualización de su propuesta educativa, apoyándose en procesos de revisión de literatura científica, resultados de investigación nacional e internacional, intercambios académicos, monitoreo y evaluación continua tanto interna como externa de su propuesta educativa. Todos estos elementos van aportando insumos y conocimientos sobre la Informática Educativa, sus resultados de aprendizaje, y las condiciones en los centros educativos que propician o dificultan su implementación. Estos procesos de investigación, monitoreo y evaluación permiten también identificar el surgimiento de nuevas alternativas tecnológicas y didácticas, al tiempo que facilitan el ajuste periódico a las necesidades del sistema educativo y de la sociedad costarricense, sin abandonar el modelo pedagógico que se viene implementando desde 1988. De acuerdo con este enfoque, "las computadoras se usan para estimular la producción creativa e intelectual de los niños y jóvenes, propiciar la generación de nuevas dinámicas en el aula y contribuir al logro de una inserción plena y provechosa del país en la sociedad del conocimiento" (FOD, 2016, p.11). A continuación, se presentan los elementos que conforman el diseño didáctico que da origen a la propuesta educativa LIE ++ pensar, crear, programar.

4.1. Competencias estudiantiles

Para comprender mejor el modelo de evaluación por competencias que sustenta el diseño de la propuesta educativa LIE ++, es importante identificar los elementos que lo componen.

A partir del marco de resultados de aprendizaje esperados en estudiantes del PRONIE MEP-FOD, se seleccionan dos de las cuatro dimensiones descritas, estas son 1) Pensamiento Computacional y 2) Creatividad, innovación y emprendimiento. De estas dimensiones se desprenden las siguientes competencias: resolución de problemas con programación, manejo de operaciones y componentes de sistemas computacionales, representación y modelaje de datos, así como la construcción de artefactos físicos y robots (FOD, 2016, p.20) para ser desarrolladas en LIE ++ y lograr conformar los perfiles de salida para cada año escolar desde preescolar hasta el noveno grado. Para el desarrollo de las habilidades que se requieren para alcanzar las competencias antes indicadas, es necesario considerar los tres saberes según Delors (1996); el saber (lo conceptual), saber ser (lo actitudinal) y saber hacer (lo procedimental), lo que propicia una visión humanística centrada en el estudiante. Desde la propuesta de LIE ++ estos saberes se obtienen de las ideas poderosas (el saber), las prácticas del pensador computacional (el saber hacer) y las actitudes del pensador computacional (el saber ser).

4.1.1. Ideas Poderosas

Para Papert, se hace necesario brindarles a los estudiantes, aquellos conceptos y capacidades que no van a modificarse con los avances tecnológicos, de manera que, independientemente de los cambios que van a surgir en el futuro, ellos tengan las herramientas cognitivas y operativas para enfrentar esos cambios (1987, p.16).

A partir de esta declaración el PRONIE MEP–FOD define una serie de saberes a promover en los estudiantes, los cuales son hilos conductores que integran los conceptos por abordar en cada uno de los niveles, desde el preescolar hasta el noveno año. A estos saberes se les denominan ideas poderosas, por su permanencia a lo largo del tiempo: pese a que la tecnología cambie o evolucione, el concepto detrás de cada una de las ideas se mantiene. Por lo tanto, cuando se entienden las ideas poderosas, se facilita la comprensión de las tecnologías actuales, así como las del futuro (Angulo et al., 2018).

Para el diseño de la propuesta LIE ++, se preparó un mapa conceptual de la computación y el pensamiento computacional. Este ejercicio evidenció los conceptos medulares y las relaciones que los asocian, las cuales fueron clave para orientar el establecimiento de los contenidos y la profundidad de lo que se desea enseñar a los estudiantes, según el grado escolar que resulta apropiado para esto (Rodríguez et al., 2018). A partir de este mapa conceptual, se establecieron las cuatro grandes ideas poderosas del PRONIE MEP-FOD, a saber:

1. **Procesamiento de datos:** Esta idea se conforma tomando como base el concepto de dato, el cual se relaciona con todo lo que hacemos, percibimos y procesamos. Por lo tanto, se establece que las computadoras, desde las más pequeñas, hasta las más complejas, fundamentalmente operan sobre datos (registrar, almacenar, representar, analizar y transformar), cualquier análisis o estudio que se hace sobre y con las computadoras y sus usos requiere de una comprensión de las operaciones sobre datos.
2. **Máquinas y programas:** Esta idea poderosa, busca llevar a la profundidad conceptual de entender que estamos rodeados por muchas computadoras, por lo que es importante comprender que todas las computadoras son computacionalmente equivalentes (y equivalentes a la máquina de Turing) y están basadas en una misma arquitectura que consiste de procesador, memoria, entrada y salida, que permite la ejecución de programas en memoria (programa almacenado).
3. **Programación:** Esta idea poderosa promueve el análisis y el modelaje de programas a problemas, mediante la formalización de algoritmos, a través del uso de lenguajes de programación, aprovechando la potencialidad de las computadoras. Las computadoras son controladas mediante programas. El diseño de programas favorece el desarrollo de actitudes y prácticas del pensador computacional.
4. **Abstracciones y modelos:** Cada representación creada a través de la programación o mediante un prototipo, modela una solución que se ha creado a través de una abstracción cognitiva, es decir, mediante un proceso mental que ha implicado una comprensión profunda de un problema para atender lo que en realidad es relevante para solucionarlo. Mediante la construcción de capas o niveles de abstracción computacional, se han podido desarrollar sistemas computacionales más cercanos al usuario y que resuelven problemas más complejos.

4.1.2. Prácticas

Una manera de hacer observable el Pensamiento Computacional, es a través del desarrollo de sus prácticas, por lo que la propuesta LIE ++ promueve en la resolución de problemas prácticas como la descomposición del problema en partes más pequeñas, el reconocimiento de patrones, la abstracción cognitiva al quitar los detalles y quedarse con lo esencial del problema, la generalización y la transferencia al aplicar el aprendizaje obtenido de un dominio a otro diferente. Una vez planteada una solución al problema, lo que sigue es su implementación y evaluación, por eso es que en esta etapa se privilegian prácticas como la modularización de programas, la formulación de algoritmos, el remezclar, el depurar y el programar. Adicionalmente, la propuesta LIE ++ ha identificado otro conjunto de prácticas asociadas que han sido pensadas de manera transversal para ser tomadas en cuenta durante el desarrollo de las actividades de aprendizaje, ya sea fortaleciéndolas en el desarrollo mismo de la actividad (saber hacer) como en el área actitudinal (saber ser), tales como comunicar, colaborar, pensar de forma creativa, autogestionar el aprendizaje, así como manejar las tecnologías de forma ética y segura (FOD, 2018, pp. 1-3).

Reconocer estas prácticas, permite cobrar conciencia de cuándo las empleamos en nuestra cotidianeidad, en qué momentos se hace necesario recurrir a ellas, y cómo es que realizamos los procesos mentales para resolver los problemas tecnológicos o no.

4.1.3. Actitudes

Las actitudes reflejan en el estudiante esos saberes más relacionados al ser. Las actitudes, que persigue desarrollar la propuesta educativa LIE ++ son, gusto por la precisión aprender del error, flexibilidad para manejar problemas y la tolerancia a la frustración (FOD, 2018, p. 7).

4.2. Resultados de Aprendizaje (RdA)

Es a partir de las ideas poderosas, prácticas y actitudes que se definen los resultados de aprendizaje[1] (RdA) que se desean ver en los estudiantes a medida que avanzan por la propuesta educativa de LIE ++. Estos resultados de aprendizaje permiten orientar el desarrollo de las actividades y recursos a implementar con los estudiantes, así como el proceso de evaluación, pues cada vez que un estudiante alcanza un RdA se está acercando cada vez más al desarrollo de la habilidad esperada y por ende a lograr la competencia propuesta.

4.3. Niveles de progresión

Para implementar esta propuesta educativa, se hace necesario dosificarla, de forma que se tengan claros los conceptos que se deben ir enseñando y como estos progresan a lo largo de los 10 años escolares que atiende el PRONIE MEP –FOD (K-9), para esto se realiza la agrupación en cuatro grandes niveles. Cada uno de los niveles se compone de varios grados escolares permitiendo una progresión conceptual, considerando factores como la edad y cercanía de los estudiantes entre un grado y otro, brindando flexibilidad al docente para dosificar los contenidos según las necesidades y el ritmo de aprendizaje de los estudiantes. El Nivel 1 se compone de los años de preescolar, primero y segundo grado. El Nivel 2 lo componen tercero y cuarto año. El Nivel 3 se compone de quinto y sexto año. El Nivel 4 lo integra la secundaria general básica (séptimo, octavo y noveno año). Esta segmentación ofrece andamios cognitivos para construir comprensiones más profundas y complejas según van avanzando por los diferentes grados escolares. Por ejemplo, en los niveles 1 y 2 se realizan construcciones más concretas para luego pasar a los niveles 3 y 4 con relaciones conceptuales más abstractas, pero también más amplias y diversas en procesos y variedad de dispositivos a programar y controlar.

4.4. Metodología de resolución de problemas

Cada uno de estos niveles cuenta con uno o más módulos que promueven el desarrollo del pensamiento computacional mediante una pregunta orientadora, con la cual el docente formula una serie de problemas por resolver con los estudiantes, que permitan buscar la solución al problema. Para lograrlo los estudiantes deben aplicar las destrezas STREAM para su solución, involucrando de manera práctica conceptos del currículum escolar de forma natural. “Nunca nos involucramos en una sola actividad, lo hacemos en varias actividades a la vez, por lo tanto, el sujeto pertenece a múltiples ambientes” (Rodríguez, 2008, p. 83). Es por eso que el docente debe aplicar con sus estudiantes una de las metodologías de resolución de problemas a lo largo del desarrollo de la solución, así como planificar de manera oportuna las actividades que le permitan a los estudiantes la comprensión e incorporación de los aprendizajes

[1]Un resultado de aprendizaje es una “declaración de lo que se espera que un estudiante conozca, comprenda y/o sea capaz de hacer al final de un periodo de aprendizaje” (ANECA, p.15, s.f).

esperados en la solución, según los RdA propuestos. Con estas metodologías de trabajo, los estudiantes de primaria y secundaria podrán transferir sus aprendizajes en resolución de problemas, a distintos ámbitos.

4.5. Plan de Actualización para docentes de informática Educativa

En paralelo al diseño de la propuesta educativa LIE++ y los rediseños necesarios derivados de los procesos de pilotaje, se ha diseñado un plan de actualización docente que se compone de una etapa inicial con actividades de capacitación convocadas oficialmente y otras voluntarias, seguido de una segunda etapa con actividades voluntarias solamente.

El plan contiene cursos virtuales mediados, autogestionados, talleres, webinarios y recursos de aprendizaje que buscan desarrollar en los docentes especializados en informática educativa, las competencias necesarias para ejecutar la propuesta educativa LIE++ pensar, crear, programar. Con el fin de aumentar la calidad de las lecciones de informática educativa, la FOD inicia en 2021 un proceso de certificación docente con una prueba que permita mapear esas áreas de mejora en los docentes, y de manera paulatina ir avanzando hacia formular una prueba de certificación del pensamiento computacional y su didáctica.

4.6. Recursos didácticos: Esferas de aprendizaje

La planificación lleva consigo también la selección de los recursos didácticos que apoyen el proceso de enseñanza y aprendizaje. Como parte de los apoyos que acompañan las propuestas didácticas por nivel, el PRONIE MEP–FOD ha diseñado un repositorio de recursos llamadas “Esferas de Aprendizaje”, en las cuales docentes y estudiantes encontrarán de manera organizada los recursos requeridos para apoyar los aprendizajes esperados, según el nivel en que se encuentren. De igual manera, al estar las esferas de aprendizaje en línea, se espera que los estudiantes tengan la oportunidad de seguir avanzando en su proceso de aprendizaje fuera del horario escolar, motivándoles a seguir aprendiendo por su cuenta de manera autogestionada.

5. Evaluación

LIE ++ inició su implementación a inicios del 2018 con el primer bloque de docentes y centros educativos. Un año más tarde, en 2019, se realizó la primera evaluación que proporcionó evidencia sobre los factores que facilitan el desarrollo del pensamiento computacional (PC) a través de LIE ++ y en comparación con estudiantes que continuaban trabajando todavía con la anterior propuesta educativa enfocada en aprendizajes con tecnologías digitales y basada en estándares de desempeño que se operacionalizaban mediante guías didácticas.

> “En ese estudio participaron 14.795 estudiantes, respondiendo voluntariamente una prueba en línea que se construyó y validó para estimar los puntajes alcanzados en PC. Los resultados mostraron que los estudiantes participantes de LIE++ obtuvieron mejores puntajes en comparación con el grupo de LIE-Guías y mediante un modelo de regresión multinivel se identificaron que variables personales y sociales de los estudiantes y de la misma ejecución de la propuesta inciden en el favorecimiento de estos aprendizajes” (Picado-Arce et al., 2021, p. 85).

6. Acciones y retos a Futuro

Para el PRONIE MEP–FOD, el convertir la programación, codificación y habilidades asociadas a la resolución de problemas con programación, en una nueva alfabetización en el sistema público costarricense, se vuelve tan importante como aprender a leer y escribir. Al analizar los resultados del estudio realizado en 2019, se destaca que la propuesta educativa LIE ++ ofrece más y mejores oportunidades a los estudiantes que transitan por ella, tanto a nivel de acercamiento a las tecnologías y su conocimiento de ellas, como de ofrecer nuevas formas de pensar y resolver problemas. Por lo que se concluye que se debe seguirse avanzado en la consolidación de LIE ++, con estrategias cada vez más construccionistas, a la vez que se beneficien a más estudiantes del sistema público costarricense. Como consecuencia el PRONIE MEP–FOD inicia en el 2021 un proceso de acercamiento dirigido a las poblaciones beneficiadas con propuestas educativas de Aprendizaje con Tecnologías Móviles (ATM) del Programa pero que están enfocadas en la apropiación tecnológica para el aprendizaje y en donde los estudiantes cuentan con una computadora, pero no con un laboratorio ni docente de informática. Este acercamiento inicia con una serie de talleres piloto extracurriculares de programación y pensamiento computacional, los cuales se proyecta llevar a una población mayor de estudiantes en el 2022, buscando la participación voluntaria de docentes que deseen certificarse en Pensamiento Computacional y su didáctica.

Sin duda el principal reto es implementar esta propuesta a escala nacional, en un sistema educativo poco constructivista, en el que se ha masificado la educación. Somos conscientes que es una buena opción para empoderar a las futuras generaciones con nuevas formas de pensar, capaces de resolver problemas complejos, promoviendo el pensamiento crítico y sistémico, comprendiendo los fundamentos detrás de las tecnologías y así poder formar parte de esta sociedad de conocimiento, que cada vez más demanda habilidades de orden superior.

Bibliografía

ANECA. (s.f.). Guía de apoyo para la redacción, puesta en práctica y evaluación de los resultados del aprendizaje (Vol. 1). Madrid: Cyan, Proyectos Editoriales, S.A.

Angulo, C., Cañas, A., Castro, G., Muñoz, L., y Zamora, N., (2018). Think, Create and Program: Envolving to a K-9 Nationwide Computational Thinking Curriculum inCosta Rica. En Dagiené, V. y Jasuté, E. (Eds.). International Conference Constructionism: constructionism, computational thinking and educationalinnovation (p.78). Conferencia llevada a cabo en Lituania. Disponible en http://www.constructionism2018.fsf.vu.lt/file/repository/Proceeding_2018_Constructionism.pdf

Belluscio, M., Pintos, P., y Bordone, M. (2020). Pensamiento computacional: el desafío que debe enfrentar la escuela | ISEP.

Delors, J. (1996). Los cuatro pilares de la educación. En Delors, J (Ed.). La educación encierra un tesoro: Informe a la UNESCO de la Comisión internacional sobre la educación para el siglo XXI (pp. 91-103). Madrid, España: Santillana/UNESCO.

Fundación Omar Dengo [FOD] (2016). Tecnologías digitales y capacidades para construir el futuro: aportes del Programa Nacional de Informática Educativa MEP-FOD. San José, Costa Rica: FOD.

Fundación Omar Dengo [FOD] (2018). Propuesta actualizada de Informática Educativa: fundamentos pedagógicos y curriculares. San José, C.R.: FOD.

Papert, S. (1987). Desafío a la mente: computadoras y educación. Buenos Aires, Argentina: Ediciones Galápago

Picado-Arce, K., Matarrita-Muñoz, S., Núñez-Sosa, O., y Zúñiga-Céspedes, M. (2021). Drivers for the development of computational thinking in Costa Rican students. [Facilitadores del desarrollo del pensamiento computacional en estudiantes costarricenses]. Comunicar, 68, 85-96. https://doi.org/10.3916/C68-2021-07

Rodríguez, V. (2008). Del constructivismo al construccionismo: implicaciones educativas. Revista Educación y Desarrollo Social. Bogotá.

Rodríguez, A., Zamora, N., Angulo, C., y Cañas, A. (2018, septiembre). Uso de mapas conceptuales para descubrir ideas poderosas y guiar el diseño de un currículo de pensamiento computacional. Concept Mapping: Renewing Learning and Thinking, Medellín, Colombia.

Schleicher, A. and Partovi, H., 2019. Computer Science and PISA 2021 - OECD Education and Skills Today. [online] OECD Education and Skills Today. Available Disponible en: https://oecdedutoday.com/computer-science-and-pisa-2021/ [Accessed 25 March 2021].

Watkins, C. (2003). Learning: a sense making guide. Londres: Association of Teachers and Lecturers. Disponible en: https://www.atl.org.uk/Images/Learning%20a%20sense%20makers%20guide%20-%202011.pdf

Currículo de referência em tecnologia e computação: uma proposta do centro de inovação para a educação Brasileira

Christian Brackmann* André Raabe† Alice Carraturi‡
brackmann@iffar.edu.br raabe@univali.br alice@cieb.net.br
IFFAR (Brasil) UNIVALI (Brasil) CIEB (Brasil)

Resumo

A elaboração do Currículo de Referência em Tecnologia e Computação do Centro de Inovação para a Educação Brasileira (CIEB) partiu de uma análise criteriosa e da sistematização dos principais aprendizados advindos das referências curriculares nacionais e internacionais de países e territórios que já implantaram os temas da inovação, tecnologia e computação em seus currículos de educação básica. O referencial curricular resultante está organizado em três eixos estruturantes (Pensamento computacional, Tecnologia Digital e Cultura Digital), dez conceitos e 147 habilidades, cada qual associada com práticas conectadas à Base Nacional Comum Curricular (BNCC), rubricas de avaliação, materiais de referência e com os níveis de maturidade da escola e do docente. O currículo está disponível gratuitamente e já foi utilizado como referência por diversas instituições de ensino público e privado em nível municipal e estadual no Brasil.

Palabras clave: Currículo, Educação Básica, Computação, Tecnologia Digital, Cultura Digital, Computação na Escola.

1 Introdução

A tecnologia e a computação hoje são onipresentes em diversos aspectos de nossas vidas: na maneira como acessamos conhecimento, buscamos e trocamos informações, na comunicação com outras pessoas, nos sistemas de saúde, transporte, produção de bens e serviços, entre outros. Neste contexto, é fundamental que os jovens aprendam os conceitos, mecanismos e implicações destas áreas, de forma que possam atuar criticamente enquanto cidadãos do século XXI.

A recém-aprovada Base Nacional Comum Curricular (BNCC) (BRASIL/MEC, 2017) da educação infantil ao fundamental aborda temas de tecnologia e computação de forma transversal em todas as áreas do conhecimento e componentes curriculares – conforme detalha a Nota Técnica #12 “Conceitos e conteúdos de inovação e tecnologia

*Instituto Federal Farroupilha (IFFAR).

†Universidade do Vale do Itajaí (UNIVALI).

‡Centro de Inovação para a Educação Brasileira (CIEB).

(I&T) na BNCC" (CIEB, 2018), elaborada pelo CIEB. A competência geral número 1 fala na valorização de conhecimentos construídos no mundo físico, social, cultural e digital, enquanto a número 2 ressalta a importância de fomentar nos alunos a resolução de problemas e a criação de soluções (inclusive tecnológicas). Notadamente, a competência geral número 5 explicita a necessidade de se trabalhar com o tema de tecnologias digitais de informação e comunicação (TDICs), colocando os estudantes como aprendizes ativos e criativos - e não apenas consumidores passivos de tecnologias (BRASIL/MEC, 2017):

> Compreender, utilizar e criar tecnologias digitais de informação e comunicação de forma crítica, significativa, reflexiva e ética nas diversas práticas sociais (incluindo as escolares) para se comunicar, acessar e disseminar informações, produzir conhecimentos, resolver problemas e exercer protagonismo e autoria na vida pessoal e coletiva.

A partir da aprovação da Base, está posto o desafio para as redes de ensino construírem ou revisarem seus próprios currículos. E é justamente neste momento que surgem inúmeras questões: o que ensinar na área de tecnologia? Como ensinar? Quais as práticas pedagógicas que podem inspirar novas formas de aprender e ensinar? Como implementar e avaliar as habilidades propostas em sala de aula?

É com o objetivo de ajudar os profissionais da educação a responder estas perguntas, somando-se ao imenso esforço que já vem sendo empreendido por professores e gestores, que o CIEB apresenta este Currículo de Referência em Tecnologia e Computação[1]. Ele busca apoiar as redes de ensino oferecendo um material de excelência, de forma prática e flexível, para que elas possam trabalhar o tema de tecnologia e computação nos seus currículos tanto de maneira transversal quanto em uma área de conhecimento específica.

Ciente da complexidade deste desafio, o CIEB convidou especialistas para desenvolver um extenso e minucioso trabalho, até chegar em uma proposta curricular. Este documento traz os marcos conceituais, inspirações, bases metodológicas e teóricas para construção do Currículo de Referência em Tecnologia e Computação, de forma a oferecer um material de qualidade e útil para as redes de ensino.

Um dos grandes diferenciais deste material é apresentar referências sobre como os professores podem desenvolver cada uma das habilidades propostas em sala de aula, por meio da sugestão de práticas pedagógicas. Além disso, também é fundamental entender se os alunos aprenderam determinado conteúdo abordado em uma habilidade – e é por isso que são apresentadas sugestões de avaliação e, da mesma forma, materiais de referência (sites, plataformas, objetos digitais de aprendizagem, jogos, programas etc.) que podem apoiar os professores no planejamento e na sala de aula.

Outro elemento importante da proposta é a associação de cada uma das habilidades com o nível de adoção de tecnologia da escola, que refere-se à presença de tecnologias no ambiente escolar e, ainda, com o nível de adoção de tecnologia do docente, que é o conhecimento específico necessário para desenvolver as habilidades a partir de práticas. É evidente que estas são apenas recomendações, mas levam em consideração a realidade da infraestrutura atualmente disponível nas escolas brasileiras, bem como o conjunto de conhecimentos costumeiramente presentes na formação inicial de professores. Mais ainda, diversas práticas sugeridas são desplugadas, ou seja, não necessariamente exigem recursos digitais, conectividade ou infraestrutura tecnológica complexa, trabalhando os conceitos por meio de metodologias e diversos materiais didáticos.

[1]A opção pelo termo "Tecnologia e Computação" nesta proposta parte do entendimento de que esta nomenclatura abarca tanto conceitos abstratos quanto suas aplicações (e implicações) práticas em instrumentos, técnicas e métodos.

O CIEB disponibilizou gratuitamente todo o conteúdo deste Currículo de Referência em Tecnologia e Computação de forma simples, prática e intuitiva na plataforma[2], onde as redes de ensino podem selecionar as referências que fazem mais sentido para o seu contexto – filtrando por etapa da educação, ano, eixo ou conceito para explorar habilidades específicas do currículo proposto.

Espera-se que este material colabore com a autonomia das redes que buscam trabalhar com tecnologia e computação nos seus currículos, apoiando professores na organização de práticas pedagógicas inovadoras e alunos por meio da oferta de uma aprendizagem mais contemporânea e significativa.

2 Referências para elaboração do currículo

A fim de propor um currículo inovador e ao mesmo tempo compatível com a realidade escolar brasileira, foram pesquisadas diversas referências internacionais e nacionais. O olhar para essas referências buscou equilibrar conhecimentos e práticas curriculares de países que já têm em seus documentos conteúdos de tecnologia e computação, ao mesmo tempo em que dialoga com o conhecimento existente sobre o ensino desses temas na educação básica no Brasil. Para isso, foram selecionados e analisados os seguintes materiais:

- Referências Nacionais: Base Nacional Comum Curricular (BNCC), referenciais de formação para Educação Básica da Sociedade Brasileira de Computação (SBC) e componente curricular Tecnologias para Aprendizagem do Currículo da Cidade de São Paulo (2017).
- Referências Internacionais: componente curricular de Tecnologia do currículo da Austrália, currículo de Computação do Reino Unido (*National Curriculum for Computing*) e o Currículo NGSS (*Next Generation Science Standards*) dos Estados Unidos da América.

Cada uma dessas referências contribuiu para a concepção e a construção deste Currículo de Referência em Tecnologia e Computação: sua estrutura conceitual se fundamenta no currículo australiano, mesclado com ideias do currículo NGSS dos Estados Unidos. A definição das habilidades e da progressão, ano a ano, pautaram-se principalmente no referencial de formação da SBC e da BNCC, incluindo elementos do currículo do Reino Unido, da Austrália e da cidade de São Paulo. A fim de evidenciar os conhecimentos curriculares que foram referência para a construção do currículo.

3 O currículo de referência em tecnologia e computação

A proposta se alicerça em princípios pedagógicos que consideram o currículo plural, orientador e integrativo. A pluralidade envolve agregar conhecimentos e saberes diversos, culturas e intenções de perspectivas diversas, a partir de todos os envolvidos no processo de ensino e aprendizagem. O currículo é orientador, na medida em que tem função de definir aprendizagens e referenciais de atividades que podem ser realizadas em sala de aula. Já o aspecto integrativo do currículo propõe a convergência de saberes, ou seja, a interdisciplinaridade.

Praticar o currículo significa dizer que temos um caminho concreto a percorrer, direcionando ações, em um determinado momento, para um ano escolar específico. A proposição de um currículo necessita, além de considerar perspectivas de construção de autonomia dos sujeitos e sua emancipação, dialogar com volatilidade, incerteza,

[2]Disponível em: http://curriculo.cieb.net.br/

complexidade e ambiguidade do mundo atual, contribuindo para que os alunos possam navegar, criar e transformar realidades.

O contato das crianças com recursos tecnológicos, já na educação infantil, é defendido pelo Ministério da Educação pelo menos desde a elaboração das Diretrizes Curriculares Nacionais para a Educação Infantil (DCNEI), em 2010. De acordo com o documento, "as práticas pedagógicas que compõem a proposta curricular da Educação Infantil devem ter como eixos norteadores as interações e a brincadeira" (MEC, 2010, p. 25, grifos do original). Dentre essas interações, as DCNEI devem garantir, entre outras experiências, a possibilidade de "utilização de gravadores, projetores, computadores, máquinas fotográficas, e outros recursos tecnológicos e midiáticos" (idem, p. 26).

Nesse sentido, Behar et. al. (2011) sugere que o uso de tecnologia na educação infantil é importante, uma vez que o trabalho com as múltiplas linguagens – incluindo a linguagem digital – com crianças dessa faixa etária permite estabelecer redes de relações. A partir dessas redes as crianças poderiam reestruturar suas significações anteriores, produzir boas diferenciações e construir outras/novas significações. De acordo com este paradigma, não basta utilizar os recursos informáticos, é preciso problematizá-los e produzir novas relações numa pedagogia reflexiva (p.6).

Tão pertinente quanto na educação infantil, a discussão das tecnologias no ensino fundamental é essencial para ampliar e ressignificar o uso das TDICs, na medida em que estas podem favorecer a emancipação e a proatividade dos estudantes, a autonomia para tomar decisões e a inserção deles em uma sociedade cada vez mais tecnológica, contribuindo para o desenvolvimento de competências e habilidades fundamentais para se viver com criatividade e criticidade.

3.1 Concepção do currículo

A sociedade vem passando por mudanças profundas: por meio dos smartphones, por exemplo, as pessoas se conectam a tudo que está à sua volta, pesquisando, assistindo, comprando, jogando, relacionando-se, aprendendo, investindo, entre tantas outras atividades. Esses aparelhos integram em um único dispositivo funções que, pouco a pouco, vão se tornando indispensáveis para as pessoas. Gradativamente, robôs e a inteligência artificial surgem em nosso cotidiano, e não apenas como objetos de filmes de ficção científica ou de empresas de tecnologia; recursos tecnológicos, como impressoras 3D, fabricam casas, próteses, órgãos humanos e estão cada dia mais acessíveis. Além disso, o universo da tecnologia nos permite: organizar uma viagem usando apenas um celular, criar ferramentas próprias ou romper com os modelos mais tradicionais de trabalho, de comunicação, de educação etc. É visível o direcionamento para uma sociedade mais colaborativa e para uma cultura de compartilhamento e construção coletiva, seja no mundo real ou no virtual.

Os relatórios da Unesco[3], da OIT[4] e da OCDE[5] evidenciam as profundas transformações pelas quais estão passando as relações humanas e de trabalho. O Fórum Econômico Mundial[6], por sua vez, destaca que 60% das crianças que nascem hoje irão trabalhar em empregos que ainda não existem. Cada vez mais, as tecnologias digitais de informação e comunicação criam um cenário de mudanças na sociedade que oportunizam à escola repensar sua estrutura, seus currículos e seu papel na transformação do mundo. Pode-se verificar em diferentes países (como

[3]Organização das Nações Unidas para a Educação, Ciência e Cultura.

[4]Organização Internacional do Trabalho.

[5]Organização para Cooperação e Desenvolvimento Econômico.

[6]World Economic Forum. The Future of Jobs. Report.

Austrália, Reino Unido e EUA) o incentivo a políticas educacionais que visam ampliar o contato dos alunos com as tecnologias nas escolas

A incorporação de tecnologias digitais na educação nunca foi tarefa fácil. Desde a Linguagem Logo[7], criada por Seymour Papert na década de 1960 – a qual revolucionou e impulsionou o desenvolvimento de diversas tecnologias para uso no processo de ensino e aprendizagem –, temos visto um aumento exponencial de tecnologias que apoiam e organizam esse processo. Contudo, a adoção de tecnologias pelas escolas e por professores está cercada de desafios, tanto na esfera pública quanto na privada.

Seja o uso de softwares e jogos educativos ou o uso da internet, da robótica e da fabricação digital, cada nova tecnologia que entra no universo da educação formal requer diferentes perspectivas para ser adotada, nas variadas realidades locais. Existem escolas que desconsideram as inovações tecnológicas, outras aderem parcialmente, e há aquelas que incorporam e ainda repensam suas práticas pedagógicas baseadas nas possibilidades oferecidas pelas TDICs. Da mesma forma, a aderência dos professores a essas inovações também é diversa e pode ou não estar associada aos conhecimentos e as experiências que eles vivenciaram ao longo das suas trajetórias profissionais e pessoais.

Em muitos casos, o uso de tecnologia apenas reforça uma prática educativa tradicional e que não contribui para a emancipação e para a autonomia do aluno (Almeida, 2016; Amante, 2007; Brackmann, 2017; Campos, 2013; Moran, 2014; Raabe, 2017; Valente, 2002).

Diversos autores, como Almeida (2016), Campos (2017), Raabe et al. (2015) e Valente (2016), têm destacado a importância da incorporação das tecnologias ao processo de ensino e aprendizagem, seja pelo potencial enriquecedor para o trabalho do professor, seja pela atuação criativa e domínio do processo de construção de conhecimento por parte do aluno. O uso de tecnologias pelo ser humano está cada vez mais evidente, e escolas do mundo todo têm incorporado tecnologias ao seu dia a dia, com o intuito de transformar a realidade escolar e a realidade do aluno. Um dos fatores essenciais do uso das tecnologias na educação é que, concomitantemente ao uso, surgem metodologias e estratégias de ensino e aprendizagem que buscam inovar o interior da escola, atualizando-a e colocando-a à altura de seu tempo.

Nesse sentido, este currículo de tecnologia e computação busca orientar a escola e o professor quanto às aprendizagens essenciais em relação às tecnologias e as premissas da computação, destacando o que é necessário para se alcançar os objetivos de cada ano escolar, desde a educação infantil até o último ano do ensino fundamental.

3.2 Organização do currículo

O currículo está organizado em três eixos, que se subdividem em dez conceitos associados a 147 habilidades. Os eixos e conceitos se mantêm os mesmos ao longo de todas as etapas da educação que o currículo abrange, ou seja, da educação infantil aos anos finais do fundamental. Cada conceito promove o desenvolvimento de uma ou mais habilidades, as quais têm indicações de práticas pedagógicas para ajudar o professor no trabalho com os alunos e estão relacionadas com as habilidades e as competências gerais propostas pela BNCC. A organização do currículo segue com indicações para avaliação para compreender se o aluno assimilou os conteúdos trabalhados, e indica materiais de referências que podem ajudar o professor no planejamento da sua aula. Por fim, são indicados o nível de adoção de tecnologia da escola e o nível de adoção de tecnologia do docente, que diz respeito à presença, à apropriação e ao uso da tecnologia pelos diversos atores da escola, enfatizando os conhecimentos do professor. Esta proposta de organização do Currículo de Referência em Tecnologia e Computação pode ser vista na Figura 1 e na sequência serão

[7]Logo Foundation: https://el.media.mit.edu/logo-foundation/what_is_logo/logo_programming.html.

Figura 1: Estrutura do Currículo de Referência em Tecnologia e Computação

detalhados os elementos que compõem a estrutura deste currículo.

3.3 Eixos estruturantes

Os eixos estruturantes são entendidos como os grandes temas que este currículo compreende. Esses temas são abrangentes e contêm os conceitos (ou conceitos-chave), que ajudam na organização das habilidades por proximidade semântica. Neste currículo estão contemplados três eixos estruturantes: Cultura Digital; Tecnologia e Sociedade; e Pensamento Computacional, os quais são detalhados a seguir, juntamente com seus respectivos conceitos.

a) **Cultura Digital:** A cultura digital se aproxima de outros temas, como sociedade da informação, cibercultura, revolução digital e era digital. Compreende as relações humanas fortemente mediadas por tecnologias e comunicações digitais. Trabalha ainda o letramento digital. Ser letrado, atualmente, seja no mundo virtual ou não, é compreender os usos e possibilidades das diferentes linguagens na comunicação, incluindo a linguagem narrativa verbal, oral ou escrita. Nesse sentido, ler é mais do que identificar letras e números, palavras, desenhos, imagens etc. Para analisar e avaliar criticamente textos narrativos, verbais ou não verbais, é preciso identificar e problematizar as informações recebidas, conhecendo e usando os diferentes tipos de mídias, tanto para identificar como transformar as diferentes situações vividas no cotidiano e o seu contexto, por exemplo, sua escola ou

comunidade (MEC. Cultura Digital, Série Cadernos Pedagógicos, 2013).

b) **Tecnologia Digital:** O termo Tecnologia Digital é amplo, mas, no escopo deste currículo, representa o conjunto de conhecimentos relacionados ao funcionamento dos computadores e suas tecnologias, em especial as redes e a internet. Outras formas de tecnologia digital (relógios, por exemplo) não são foco de interesse do currículo. A área de computação tradicionalmente aborda muitos dos conceitos compreendidos aqui como tecnologia digital, o que inclui hardware, software, internet, sistemas operacionais, bancos de dados etc.

c) **Pensamento Computacional:** O termo Pensamento Computacional se refere à capacidade de resolver problemas considerando conhecimentos e práticas da computação (RAABE, 2017). Compreende sistematizar, representar, analisar e resolver problemas. Tem sido considerado como um dos pilares fundamentais do intelecto humano, ao lado de leitura, escrita e aritmética, pois, como estes, serve para descrever, explicar e modelar o universo e seus processos complexos.

4 Como acesssar o currículo

Sabe-se que com a partir da aprovação da Base, coloca-se o desafio imediato de implementação da mesma pelas redes de ensino, a partir da construção de seus próprios currículos. Ciente desta demanda e buscando apresentar as contribuições aqui elaboradas de forma útil, prática e flexível para os profissionais da educação que estão engajados nesta empreitada, este currículo de referência pode ser acessado de duas formas: a partir da página virtual (`http://curriculo.cieb.net.br`) ou do formato para impressão (.pdf), que também está disponível para download na página on-line.

No site, os interessados podem navegar pelas páginas explorando os conteúdos do currículo de forma mais dinâmica e interativa – a começar, por exemplo, com o gráfico circular (Figura 2), selecionando o eixo e o conteúdo desejado para explorar.

Outra forma de navegar e explorar o material do currículo é pelo menu. Em "Sobre" são apresentados os principais aspectos e objetivos com a disponibilização do Currículo de Referência; já em "BNCC" descreve-se a relação deste currículo com a estrutura e implementação da Base; e finalmente em "Currículo" é possível conhecer todos os conteúdos propriamente deste currículo, em que o usuário pode selecionar a etapa de ensino e o ano, além do eixo e do conceito desejados.

Para procurar determinados assuntos, pode-se utilizar a ferramenta de busca. O usuário, portanto, além dos recursos já mencionados – consultar informações pelo gráfico circular ou pelo cabeçalho na página "Currículo" – pode realizar buscas por materiais de referência, palavras-chave ou mesmo por competências gerais e habilidades da BNCC, que exibem a ligação da Base com as habilidades do currículo proposto.

As habilidades da BNCC também estão contempladas neste Currículo de Referência em Tecnologia e Computação. Para isso, foram consideradas não apenas aquelas que fazem menção direta, como também algumas que fazem menção indireta à tecnologia. Assim, o profissional poderá, ao mesmo tempo em que trabalha um determinado conteúdo deste currículo de referência, contribuir com o desenvolvimento de habilidades propostas pela BNCC.

Figura 2: Os eixos e conceitos do Currículo CIEB para navegação no site

5 Considerações finais

O currículo de Referência em Tecnologia e Computação descrito neste documento é uma iniciativa que traz conteúdos complementares à BNCC. Entende-se que as redes de ensino poderão utilizar este currículo a fim de integrar, paralelamente à implementação da BNCC, conhecimentos e práticas referentes à tecnologia e a computação.

Uma vez implantado, este material permitirá aos jovens concluir o ensino fundamental com conhecimentos sólidos sobre as temáticas que o currículo aborda e exercer sua cidadania, tornando-se aptos a se expressar, aprender e produzir inovação utilizando tecnologia.

Quanto à implementação de um currículo que contemple tecnologia e computação nas redes e escolas, além da infraestrutura, que nem sempre é satisfatória, o principal obstáculo é a baixa disponibilidade de docentes com perfil e formação adequada para trabalhar alguns dos conceitos apresentados.

Espera-se que o Currículo de Referência em Tecnologia e Computação possa auxiliar as redes de ensino no Brasil a incluir no seu dia a dia práticas que desenvolvam a autonomia e o protagonismo, o pensamento reflexivo e a análise crítica, a ética e a responsabilidade dos alunos. Assim, os cidadãos formados estarão melhor preparados para os desafios do futuro.

Bibliografía

Almeida, V., y Doneda, D. (2016, October 13). O emprego e o futuro digital. Valor Econômico. http://www.valor.com.br/opiniao/4742271/o-emprego-e-o-futuro-digital

Brackmann, C. (2017). Desenvolvimento do Pensamento Computacional Através de Atividades Desplugadas na Educação Básica [Universidade Federal do Rio Grande do Sul (UFRGS)]. http://hdl.handle.net/10183/172208

Brasil. (2010). Diretrizes curriculares nacionais para a educação infantil (DCNEI). http://portal.mec.gov.br/dmdocuments/diretrizescurriculares_2012.pdf

Brasil/MEC. (2017). Base Nacional Curricular Comum (BNCC). http://basenacionalcomum.mec.gov.br/

Campos, F. R. (2013). Paulo Freire e Seymour Papert: Educação, Tecnologias e Análise do discurso. Curitiba: Editora CRV.

CIEB. (2018). CIEB: notas técnicas #12: Conceitos e conteúdos de inovação e tecnologia (I&T) na BNCC. https://cieb.net.br/wp-content/uploads/2020/08/NotaTecnica12.pdf

Raabe, A. L. A. (2017). Pensamento Computacional na Educação: Para todos, por todos!. Revista Computação Brasil, SBC, p. 54-63, 01 jul. 2017.

Valente, J. A. (2016). Integração do Pensamento Computacional no Currículo da Educação Básica: Diferentes Estratégias Usadas e Questões de Formação de Professores e Avaliação do Aluno. Revista E-Curriculum, 14(3).

Trasfondo del Desarrollo de una Propuesta de Formación en Estándares TIC en Paraguay desde el Portal META de Paraguay Educa

Sascha Rosenberger*
srosenberger@paraguayeduca.org
Paraguay Educa

Carla Fernández†
cfernandez@paraguayeduca.org
Paraguay Educa

Resumen

Paraguay se encuentra en un proceso de cambio en cuanto a su enfoque en educación tecnológica. Este tema se viene desarrollando hace casi dos décadas, aunque sólo en la última década cuenta con planes nacionales de desarrollo y educación a largo plazo. Sin embargo, estos no parecen contar con un andamiaje teórico y operativo necesario para apuntalar los objetivos de desarrollo. Estas circunstancias son el trasfondo del desarrollo del Portal Educativo META, el cual ha empezado un proceso de análisis de la situación actual de docentes y estudiantes con una lógica de proceso. Se presentan los resultados de un primer curso piloto para docentes para establecer una línea de base de estándares tecnológicos, que demuestran un esperado bajo desempeño por no ser este un tema del currículo nacional. Así mismo, se presentan los resultados preliminares de una encuesta a estudiantes que igualmente confirma los bajos resultados en pruebas estandarizadas internacionales. Sin embargo, también se presentan brevemente los resultados de un taller de fabricación digital anterior al Portal META que permite proponer un camino diferente al actual para llegar a estudiantes y trabajar sobre la relevancia de la educación tecnológica. Los datos indican la necesidad de seguir inicialmente una lógica inductiva de investigación para el desarrollo de una propuesta curricular amplia, adecuada y adecuable.

Palabras clave: Estándares ISTE, Desarrollo Curricular, Calidad Educativa, Formación Docente.

1. Introducción

En Paraguay Educa se ha desarrollado un documento interno sobre Calidad Educativa, el cual acaba de ser aprobado para publicación local (Rosenberger y Fernández, 2021). En él, se sigue la aseveración del Reporte de

*Doctor en Desarrollo Internacional por la Ruhr-Universität Bochum, Alemania. Coordinador de Evaluación y Monitoreo del Portal META de Paraguay Educa.

†Doctora en Psicología Cognitiva por la Pennsylvania State University, Estados Unidos. Analista de datos de Evaluación y Monitoreo del Portal META de Paraguay Educa

Monitoreo Global de la educación de UNESCO que indica que es *difícil monitorear la calidad de la educación porque cada país la define de forma diferente*[1] (UNESCO, 2020). Es decir, "calidad" es una definición local, por lo que su definición debe ser analizada localmente. Es por ello que se toman en cuenta tres documentos referenciales nacionales, los cuales son los primeros en la historia post-dictadura del país en superar una vigencia superior a un periodo de gobierno. Estos son el Plan Nacional de Educación Paraguay 2024 (Ministerio de Educación y Cultura, 2011), el Plan Nacional de Desarrollo Paraguay 2030 (Secretaría Técnica de Planificación, 2014), y el Libro Blanco de los Lineamientos para una Política de Ciencia, Tecnología e Innovación del Paraguay (Consejo Nacional de Ciencia y Tecnología, 2014).

Los últimos dos planes, el 2030 y el Libro Blanco, siguen la idea de desarrollo nacional publicada originalmente por Sábato y Botana (1968), que propone una interacción entre el sector productivo, el académico y el gubernamental sobre la base del desarrollo de conocimiento localizado. Sin embargo, como lo exponía Alsina (2011) para el caso Argentino, estos planes no contemplan los requerimientos básicos de un sistema educativo para lograr la sinergia de interacción requerida para el desarrollo de conocimiento local. En el caso Paraguayo, el Plan Nacional de Desarrollo se puso como meta para el año 2030 llegar al Nivel 3 de PISA, lo cual es apenas suficiente como piso absoluto para un desempeño básico en la sociedad del conocimiento y se encuentra sólo un nivel por encima de *no lograr las capacidades más básicas* (Ministerio de Educación y Ciencias, 2018).

En cuanto al Plan 2024, el mismo menciona el uso de tecnología y de conocimiento, y brevemente se refiere a la necesidad de "desdoblar el conocimiento" para su "apropiación" a través de varios medios. Sin embargo, esta aseveración sobre el desdoblamiento del conocimiento y su apropiación no se refieren al objetivo de aprendizaje de los estudiantes, sino a la tarea oficial del ministerio de seleccionar conocimiento, re-empaquetarlo según los niveles educativos a los que se orienta y proceder a su distribución pedagógica (Rosenberger, 2020).

Un punto en común entre estos tres planes es que todos mencionan la necesidad de introducir tecnología en la educación, pero no se proponen caminos para lograrlo. Es más, el Ministerio de Educación no cuenta actualmente con un currículum sobre ciencias de la computación o que cuente con estándares de uso de tecnología en la educación, excepto para los bachilleratos técnicos. Si bien puede decirse que esto se debe a que estos documentos están por cumplir una década de vigencia, la "priorización curricular" que se desarrolló en tiempos de pandemia vuelve a centrarse exclusivamente en la alfabetización digital, como la identificación de partes básicas de una computadora, uso de procesadores de texto y hojas de cálculo así como servicios de videollamadas (Ministerio de Educación y Ciencias, 2021, pp. 59–60). Es decir, la visión a futuro es una de *adopción* de tecnología en vez de una más necesaria y relevante en la que se apunte al desarrollo de un relacionamiento cognitivo con el conocimiento y la tecnología que permita su desagregación y re-creación.

En base a la coyuntura nacional vis-a-vis los requerimientos que la globalización impone a los sistemas educativos, Paraguay Educa busca re-enfocar el término "calidad educativa" en capacidades cognitivas. El término central en esta discusión es la "apropiación" del conocimiento y la tecnología, que significa entender a cabalidad los elementos constituyentes para poder criticarlos y evaluarlos y, en base a ello, desarrollar soluciones nuevas y/o adecuadas y adecuables al ambiente inmediato. Esto requiere encarar la educación de forma tal que todo proceso lleve a un relacionamiento cognitivo con el conocimiento y la tecnología que permita su desagregación, crítica y una subsecuente creación de conocimiento - su apropiación - y elegir elementos que lo permitan y propicien (Rosenberger y Fernández,

[1]Traducción de los autores. Texto original: "It is hard to monitor good quality education because each country understands and defines it in different ways."

2021). Este es el trasfondo para el resto del artículo.

2. Formación Docente en TIC

En 2020 el Portal META de Paraguay Educa, como parte de su despliegue inicial con un grupo de *friendly users*, hizo una encuesta para docentes del sector público. La encuesta se repartió en dos zonas, una periurbana - Caacupé, a 50 km de la capital Asunción - y una rural - Caazapá, a 204 km de la capital y una de las zonas más pobres del país. Se obtuvieron en total 274 respuestas, que arrojaron datos sobre el nivel de formación en TIC y dieron pie a revisión de literatura sobre estos resultados por su naturaleza aparentemente inusual.

Encontramos similitudes inesperadas entre docentes en los niveles de formación en TIC entre Caazapá y Caacupé. Las similitudes sorprendieron porque Caacupé fue el centro de implementación del programa Una Computadora Por Niño de 2009 a 2020, lo que contrasta con la situación de pobreza económica y educativa de Caazapá. La similitud se debe a un fenómeno conocido localmente como “profesores taxi”, en el que los profesores se trasladan de una ciudad a otra y de una escuela a otra con mucha frecuencia por diversas razones.

En la misma muestra encontramos que los docentes de más edad y con una media de 15 o más años de experiencia fueron quienes más reportaron haber recibido formación en TIC. Es importante destacar que los profesores menores de treinta años no reportaron tener formación formal en TIC. Este inesperado hallazgo llevó a una revisión de literatura local, a través de la cual se encontró que esto se debe a que la formación formal en TIC dependió de la financiación de la Agencia Española de Cooperación Internacional (AECI). Su programa de capacitación en TIC, llamado ’PRODEPA’ se desarrolló entre 1997-8 y 2010, con algunos proyectos en ejecución hasta 2014. Sin embargo, también hubo una miríada de diferentes proyectos de capacitación en TIC financiados por PRODEPA, con unos 14 proyectos hasta 2014 enfocados principalmente en alfabetización digital básica (Proyecto PRODEPA - Paraguay, n.d.). Los resultados de la encuesta así como la escasa literatura oficial disponible nos indica que no ha habido ninguna capacitación formal, masiva y prolongada en TIC desde el final de PRODEPA, excepto por la provisión de laptops a maestros en poblaciones específicas con capacitaciones cortas sobre el uso básico de un sistema operativo conocido.

En este contexto, iniciamos el desarrollo de cursos de capacitación docente en base a Estándares ISTE (Sociedad Internacional para la Tecnología en la Educación, por sus siglas en Inglés). Se eligieron como marco referencial los Estándares ISTE para docentes por tres razones: hay un convenio activo entre el Ministerio de Educación y Ciencias de Paraguay e ISTE; los Estándares ISTE son comparables con otros estándares de Habilidades del Siglo XXI; y Paraguay Educa es *partner* oficial de ISTE. Sin embargo, dada la limitada capacitación previa en estos temas en el plantel docente nacional en general, el Portal META eligió 6 sub-puntos específicos de los Estándares ISTE, listados a continuación:

- 3c: Mentorear a los estudiantes en prácticas seguras, legales y éticas con herramientas digitales, y en la protección de los derechos intelectuales y de la propiedad
- 4c: Utilizar herramientas colaborativas para ampliar las experiencias de aprendizaje auténticas del mundo real de los estudiantes al interactuar virtualmente con expertos, equipos y estudiantes, a nivel local y global
- 5a: Utilizar la tecnología para crear, adaptar y personalizar experiencias de aprendizaje que fomenten el aprendizaje independiente y se adapten a las diferencias y necesidades de los estudiantes
- 5b: Diseñar actividades de aprendizaje auténticas que se alineen con los estándares del área de contenido y utilice herramientas y recursos digitales para maximizar el aprendizaje activo y profundo.
- 6c: Crear oportunidades de aprendizaje que desafíen a los estudiantes a usar un proceso de diseño y pensamiento

computacional para innovar y resolver problemas.

- 7a: Proporcionar formas alternativas para que los estudiantes demuestren competencia y reflexionen sobre su aprendizaje utilizando la tecnología.

Estos sub-puntos de los estándares están siendo modificados para adaptarlos a los términos locales, a las visiones locales sobre educación, a las experiencias locales previas, y una visión construccionista del empleo de la tecnología en la educación que permita su apropiación más allá de su adopción (Rosenberger, 2020).

En base a esto se desarrolló un primer curso piloto con contenido que se adentra en el sub-punto 5a de los Estándares ISTE, particularmente en lo que se refiere a adaptación del aprendizaje, sincronicidad, asincronicidad, aprendizaje personalizado y diferencias de necesidades de aprendizaje de estudiantes. El aprendizaje se evaluó sobre la base de un cuestionario inicial de conocimientos previos, un cuestionario final con los mismos puntos que el inicial, y el desarrollo de un producto final que cada docente debe subir a la plataforma y que se califica según una rúbrica en una escala de 0 a 5. El curso se finalizó en marzo, se probó internamente y se puso a prueba en abril, y se implementó en mayo y junio en un total de 11 talleres en 4 ciudades: las ciudades urbanas de Asunción, Luque y Ciudad del Este, así como en Caacupé que es más bien periurbano. Las pruebas iniciales y finales han demostrado ser un método válido para evaluar los conocimientos previos y adquiridos. Los análisis de estos cuestionarios demuestran una mejora sustancial en los temas relacionados con el contenido.

Los análisis del producto final entregados por 96 del total de 155 maestros que se inscribieron en el curso indican que el nivel de referencia de conocimiento de los estándares ISTE es particularmente bajo, lo cual resulta obvio dado que no forman parte del currículo nacional. 23 profesores varones se inscribieron en el curso, pero solo 9 de ellos completaron el curso (tasa de finalización del 39,1 %). Su tasa de adquisición fue comparable a la de las maestras. El resultado final del curso es como sigue:

- de un máximo de 5 puntos (un punto por cada sub-ítem arriba mencionado), el docente llega en promedio a una nota de **1.44**.
- de los 5 sub-ítems evaluados, el ítem con promedio más alto fue 2.52, el cual se refiere al aprendizaje sincronónico
- la moda más alta de un sub-ítem fue 4, mientras que la moda de los demás fue 0.
- la mediana más alta de un sub-ítem fue 3, en el mismo que el del punto anterior(aprendizaje sincrónico)

Es decir, el rendimiento promedio es bastante bajo en general, excepto en en el sub-ítem específico que se refiere al aprendizaje sincrónico, donde es un poco más alto. Esta diferencia no es del todo sorprendente ya que los docentes se encuentran mayormente familiarizados con el aprendizaje sincrónico. Es importante señalar que estos son resultados del primer curso sobre estándares tecnológicos, que además es un piloto extendido.

Es necesario destacar que, aunque las pruebas de conocimientos iniciales y finales demuestran que la mayoría de los profesores han oído hablar de estos estándares, no tienen conocimientos prácticos sobre cómo aplicarlos en sus aulas lo cual se ve reflejado en el bajo promedio de la tarea final. El curso inicial implementado en estos talleres nos brinda una imagen clara de la necesidad de cursos que exploren estos estándares con mayor profundidad y permitan a los maestros comenzar a planificar su implementación de una manera práctica. El desafío del Portal META consiste en presentar estos estándares y su desarrollo en el contexto del currículum nacional de modo a lograr establecer una ruta de aprendizaje que cimente esto como parte del aprendizaje de cada docente formado o en formación para su implementación general.

En esta misma línea, tomamos estos aprendizajes y los orientamos al desarrollo de cursos y materiales para la enseñanza de pensamiento computacional, que es parte de los Estándares ISTE adoptados como objetivo del programa.

Aquí se deben tomar en cuenta por los menos tres aspectos: los requerimientos docentes, que se desprenden del currículo nacional; los niveles de relacionamiento cognitivo del sistema educativo nacional, que está en relación estrecha con el punto anterior; y el desempeño de los estudiantes en pruebas nacionales así como en proyectos internos de Paraguay Educa.

3. Currículum, Formación Docente y Desempeño Estudiantil

Cuando decimos requerimientos docentes, que se desprenden del currículo nacional, nos referimos a aquello que se espera que cada docente desarrolle en sus clases así como la capacitación requerida para lograrlo. Si bien el "Análisis Curricular del Estudio Regional Comparativo y Explicativo (ERCE 2019)" indica que el currículo de Paraguay tiene una orientación socio-constructivista (Sotomayor y Cabello, 2020, pp. 32, 39), es difícil conciliar esta orientación con el tipo de medición de logros de aprendizaje de SNEPE que, hasta 2011, se limitaba al nivel 3 de Bloom (Ministerio de Educación y Ciencias, 2011, p. 37) - nivel muy por debajo del necesario para la creación de conocimiento socioconstructivista.

En cuanto al segundo punto, si bien el currículo y los planes de estudio incorporan el aprendizaje basado en proyectos, por lo menos en lo tecnológico y en el uso de TIC un estudio indica que el nivel de relacionamiento cognitivo se mantiene igualmente en el nivel 3 de la Taxonomía de Bloom (Rosenberger, 2019). Según este estudio la tecnología es concebida como una herramienta de uso con características específicas dadas, y no como un objeto de estudio desagregable, analizable y adecuable que sirva para como elemento para la construcción de conocimiento. Dado el actual énfasis del desarrollo en lo *glocal* (Bolívar, 2001) y de la necesidad de la educación de incluir conceptos computacionales para el desarrollo de y con tecnología, elegimos el Estándar ISTE 6c - pensamiento computacional y de diseño - en la formación docente. Esto lleva a discutir la actual situación del alumnado.

Las pruebas nacionales e internacionales indican que el rendimiento escolar promedio en Paraguay fue y sigue siendo muy bajo. El 68 % de estudiantes no tienen las competencias mínimas en lectura comprensiva, el 92 % no tienen las competencias mínimas en matemáticas, y el 76 % no las tienen en ciencias (Ministerio de Educación y Ciencias, 2018). Sin embargo, la realidad es algo diferente bajo otros estándares, como se pudo comprobar en un programa de introducción a la fabricación digital de 6 meses en el que participaron 200 estudiantes de colegio públicos de la ciudad de Caacupé. En base a los hallazgos en estos talleres, hipotetizamos que parte de la explicación de porqué el rendimiento es bajo puede deberse a la falta de oportunidades de dedicarse a la construcción de objetos a través de los cuales se pueda entender la finalidad real del proceso de aprendizaje.

En 2019, Paraguay Educa llevó a cabo el proyecto "Taller de Introducción a la Fabricación Digital" con 200 estudiantes de colegios públicos de Caacupé. Los talleres se desarrollaron con grupos de 20 estudiantes, y la carga horaria fue 16 horas para cada grupo. Si bien los formularios de postulación confirmaron el bajo desempeño en lectoescritura que demuestran las pruebas nacionales e internacionales, durante el taller los estudiantes demostraron poder desarrollar documentos en los que explican los procesos de fabricación digital junto con *scripts* básicos, usando términos técnicos y tecnológicos correctamente. Es decir, es posible que una de las razones por las cuales el desempeño escolar es bajo sea la motivación extrínseca. Aquí es relevante citar lo siguiente:

> En el año 2006, **uno de cada 10 jóvenes de 15 a 17 años que abandonaban los estudios lo hacían porque no estaban lo suficientemente motivados para seguir estudiando** -– 11 % entre los jóvenes de las áreas urbanas y 13 % entre los jóvenes de las áreas rurales [...]. Aún más preocupante, **el porcentaje**

> **de jóvenes de 15 a 17 años que está desmotivado para seguir estudiando creció sustancialmente en la última década. Para el año 2016, tres de cada 10 jóvenes de 15 a 17 años que abandonaban los estudios lo hacían porque no estaban lo suficientemente motivados para seguir estudiando** -– 34 % entre los jóvenes de las áreas urbanas y 23 % entre los jóvenes de las áreas rurales. Por lo tanto, la falta de preparación de los estudiantes durante la EEB y la aparente poca relevancia del currículo y la instrucción en la EM son problemas complejos que parecen estar contribuyendo en forma importante al abandono escolar de los jóvenes. (Yanez-Pagans et al., 2018, p. 14)

En Abril de 2021 el Portal META lanzó una encuesta sobre intereses educativos a estudiantes de colegios públicos en los departamentos Paraguarí, Caazapá, Alto Paraná y Central; la misma recibió 334 respuestas. En esta encuesta preguntamos a los estudiantes qué áreas del currículo desearían profundizar y por qué. Si bien todavía no se ha terminado el análisis, encontramos dos puntos interesantes que reflejan las experiencias del FabLab de 2019. Primero, hay un gran interés por disciplinas que los mismos estudiantes presentan como 'relevantes – de importancia universal – como matemáticas o física, o 'pertinentes' – de importancia personal – como lengua castellana y matemáticas para el ingreso a carreras universitarias. Sin embargo, y este es el segundo punto, la calidad de las respuestas a la pregunta "por qué te parecen importantes estas áreas [disciplinas]" demuestra en la gran mayoría de los casos una incapacidad de expresar propiamente un hilo narrativo causal del tipo "me interesa la física porque quiero estudiar ingeniería". Es decir, los estudiantes presentan como necesarias ciertas disciplinas, pero no tienen las capacidades de explicar precisamente por qué las ven como relevantes o pertinentes más detalladamente.

Sin embargo, podemos proponer una hipótesis de por qué esto se da en formularios escritos y no tanto en talleres. Durante el proceso de contactar con instituciones educativas, directores y docentes para recabar información para el despliegue del Portal META, encontramos grandes dificultades al proceder a distancia a través de llamadas, mensajes y videollamadas. Los actores del sistema educativo empezaron a responder y contactar activamente luego de varias visitas presenciales. Hipotetizamos por ello que la misma reticencia y desinterés que presentaron los docentes a la hora de entablar conversación y establecer fechas de visita se da con los estudiantes. Es decir, por motivos culturales, sería tal vez más adecuado realizar visitas y entablar conversaciones presenciales con grupos de estudiantes para obtener respuestas más elaboradas y elocuentes y poder así establecer así sus capacidades.

4. Discusión

El enfoque tecnológico del sistema educativo nacional actual, aún luego de más de un año de pandemia y de experiencia con el aprendizaje a distancia, no parece apuntalar adecuadamente los objetivos de los planes nacionales de desarrollo, ni las necesidades tecnológicas que se han puesto en evidencia durante el último año y medio. Sin embargo, los mismos planes de desarrollo no parecen proveer un andamiaje teórico educativo apropiado para sus metas. Por ello, para desarrollar un currículo tecnológico para la capacitación docente a lo largo de toda la vida, así como para la introducción del estudiantado a una lógica educativa construccionista, el Portal META seguirá una lógica inductiva y orientada al proceso (Maxwell, 2004). Sería imprudente proceder a desarrollar una propuesta curricular completa para docentes y estudiantes sin entender a cabalidad las razones de los procesos por los cuales los docentes y estudiantes parecen no tener las competencias y capacidades necesarias para la construcción con y de tecnología. Estamos en proceso de construir las bases para el desarrollo de un currículo tecnológico adecuado y adecuable, que permita su contínua co-construcción, cuyos resultados serán socializados acorde estén terminados.

Bibliografía

Alsina, F. (2011). Investigación, transferencia, tecnología. In El pensamiento latinoamericano en la problemática ciencia-tecnología-desarrollo-dependencia (pp. 199–214). Ministerio de Ciencia, Tecnología e Innovación Productiva. http://www.mincyt.gob.ar/adjuntos/archivos/000/022/0000022594.pdf

Bolívar, A. (2001). Globalización e identidades: (Des)territorialización de la cultura. Revista de Educación, número extraordinario "Globalización y educación," 265–288. https://doi.org/10.4438/1988-592X-0034-8082-RE

Consejo Nacional de Ciencia y Tecnología. (2014). Libro Blanco de los Lineamientos para una Política de Ciencia, Tecnología e Innovación del Paraguay. http://www.conacyt.gov.py/sites/default/files/Libro%20Blanco%20PNCTI_web.pdf

Maxwell, J. A. (2004). Causal Explanation, Qualitative Research, and Scientific Inquiry in Education. Educational Researcher, 33(2), 3–11.

Ministerio de Educación y Ciencias. (2011). Construyendo juntos la Nueva Escuela Pública Paraguaya. https://www.mec.gov.py/cms_v2/adjuntos/7006

Ministerio de Educación y Ciencias. (2018, December 14). Reporte Nacional PISA-D Paraguay. https://mec.gov.py/cms_v2/adjuntos/15247?1545325232

Ministerio de Educación y Ciencias. (2021). Priorización Curricular 2021. https://aprendizaje.mec.edu.py/dw-recursos/system/otros_recursos/Gu%C3%ADas_de_priorizaci%C3%B3n_y_del_educador_para_Aprendizaje/1-PRIO2021.3%C2%BAyEM.DGDE.final.verif.pdf

Ministerio de Educación y Cultura. (2011). Plan Nacional de Educación 2024. Hacia el Centenario de la Escuela Nueva de Ramón Indalecio Cardozo. Ministerio de Educacion y Cultura. http://www.mec.gov.py/cms/adjuntos/2344

Proyecto PRODEPA - Paraguay. (n.d.). Retrieved December 2, 2020, from http://www.educacionyfp.gob.es/contenidos/ba/actividad-internacional/cooperacion-educativa/paebas/prodepa.html

Rosenberger, S. (2019). Grounded in Paraguay: An Appropriation Theory of ICTs and Education for Development [Ruhr-Universität Bochum]. https://doi.org/10.13154/294-6378

Rosenberger, S. (2020). Desde Paraguay: Hacia una redefinición de Apropiación. Revista CTS, 15(43), 35–64.

Rosenberger, S., y Fernández, C. (2021). Propuesta para una Articulación entre las Políticas Educativas para Redefinir Calidad Educativa en Paraguay. artículo en prensa.

Sábato, J., y Botana, N. (1968). La Ciencia y la Tecnología en el Desarrollo Futuro de América Latina. Revista de la Integración, Banco Interamericano de Desarrollo, 3(3), 15–36.

Secretaría Técnica de Planificación. (2014, December). Plan Nacional de Desarrollo | Paraguay 2030: País de oportunidades. http://www.stp.gov.py/pnd/wp-content/uploads/2014/12/pnd2030.pdf

Sotomayor, C., y Cabello, V. (2020). ¿Qué se espera que aprendan los estudiantes de América Latina y el Caribe? - Análisis curricular del Estudio Regional Comparativo y Explicativo (ERCE 2019). OREALC/UNESCO Santiago. https://unesdoc.unesco.org/ark:/48223/pf0000373982

UNESCO. (2020). GEM Report SCOPE. https://www.education-progress.org/en

Yanez-Pagans, M., Bedoya, J., y Zarza, D. (2018). Paraguay: Invertir en Capital Humano—Una revisión del gasto público y de la gestión en los sectores sociales. Capitulo 2: Educación. Grupo Banco Mundial. http://documents.worldbank.org/curated/en/145651542655422647/pdf/132203-replacement-Chapter-Education-final1.pdf

Educación a distancia. Reto de la superación a través del curso Algoritmización con Scratch

Rosa María Figueredo Rodríguez
rosafr@uo.edu.cu
Universidad de Oriente (Cuba)

Yor Alex Remond Recio
reymond@uci.cu
Universidad de las Ciencias Informáticas (Cuba)

Resumen

Los profesionales de informática y ramas afines reciben actividades de superación a través de la modalidad a distancia. La educación a distancia con el uso de las Tecnologías de la Información y las Comunicaciones (TIC), es una necesidad para la formación continua de los profesionales. Como parte del III perfeccionamiento educativo, se lleva a cabo transformaciones en los planes de estudio de las diferentes enseñanzas. Uno de ellos es enseñar a los estudiantes a programar usando el lenguaje de programación Scratch. Los docentes responsables de dirigir el proceso de enseñanza aprendizaje no están preparados técnica ni metodológicamente en la utilización de la aplicación Scratch para estimular el pensamiento lógico algorítmico en los mismos, aspecto que fue constatado en el diagnóstico realizado para caracterizar las necesidades de superación. Este trabajo tiene como objetivo brindarle al docente una alternativa que les permita fortalecer el proceso de enseñanza aprendizaje de la Informática ante los nuevos retos y los requerimientos que demandan las actuales transformaciones de la Educación. Se desarrolló un curso de superación con contenidos de fundamentos de la programación y en el uso de la aplicación informática para su empleo, estos conocimientos se revierten en la calidad del proceso de enseñanza aprendizaje. Se comparte la experiencia desarrollada en la institución en la Escuela Internacional de Verano a Distancia 2020 (EVD2020).

Palabras clave: Educación a distancia, Superación; Scratch.

1. Introducción

En los momentos actuales, en el que el mundo se ha visto inmerso en una crisis epidemiológica producto a la aparición del nuevo Coronavirus SARS-Cov-2 (COVID-19) el Sistema de Educación en Cuba ha desarrollado innumerables alternativas, para lograr una educación inclusiva, equitativa y de calidad y de promover oportunidades de aprendizaje en la enseñanza de la Educación Superior, a partir del uso intensificado de las Tecnologías de la Información y la Comunicación (TIC) y de la educación a distancia en la impartición de cursos de superación para la formación continua del docente, que es aquella promovida por parte de las instituciones educativas o por el propio docente para mejorar como profesional de la enseñanza.

Estudiosos del tema sobre la Educación a Distancia como modalidad educativa como (Segovia, 1991; García, 1993; García, 2002; Foulcade, 1997; Noa, 1998; Collazo, 1999; Collazo, 2004) entre otros relacionaron el antecedente de partida de la Educación a Distancia con los cursos por correspondencia desarrollados en Inglaterra a principios del siglo XVIII. Además, Michael Moore (1975), Börje Holmberg (1981), Desmond Keegan (1990) y J. Verduin y Thomas Clark (1991), Garrison (1997) quienes, entre otros, han realizado esfuerzos para la construcción de una teoría que desde una perspectiva pedagógica permita organizarla y desarrollarla.

La Educación a Distancia en la Educación Superior cubana, dio inicio en los años sesenta del pasado siglo cuando se reconoce la importancia de utilizar diferentes variantes de esta modalidad educativa para contribuir a la preparación de los profesionales que exige el desarrollo social. (Miranda, 1990; Collazo, 1999; Bravo, 1999).

Algunos autores agrupan los diferentes acontecimientos por generaciones (Collazo, 2004; Artagey et al., 2005), mientras otros lo clasifican atendiendo a distintas etapas (Herrera, 2005c) o modelos (Noa, 2004a). Es indiscutible que la modalidad de cursos de educación a distancia mediados hoy por las tecnologías de la información y las comunicaciones ofrece alternativas concretas que la universidad pone en manos de la sociedad para acceder a la enseñanza superior (...)" (Vecino, 2003: 7).

Desde una visión actual de las propuestas de diseño de cursos a distancia (Hernández, 2000; Collazo, 2004; Michel, 2004; Solís, 2004; Gracia Fernández, 2009) puede decirse que estas centran su base en aspectos organizativos o tecnológicos del proceso, por los diferentes recursos que brinda la plataforma en que está sustentada.

Si bien es cierto que los estudiantes de hoy día necesitan desarrollar competencias que le permitan ser competitivos y exitosos en una sociedad donde la tecnología crece a un ritmo vertiginoso; asegurar la calidad del proceso educativo, es uno de los desafíos que tiene que afrontar las instituciones educativas del país. Existe dificultad en la preparación del docente para enfrentar estos nuevos retos, por lo que necesita ser superado, con la finalidad de facilitar el proceso de enseñanza aprendizaje teniendo en cuenta las características y ventajas de esta herramienta como recurso didáctico (Figueredo Rodríguez, 2018).

La Educación, como sector impulsor y enriquecedor de la sociedad, no ha quedado exenta del impacto del desarrollo tecnológico, quedando expuesta, además, a fortalecer el proceso de enseñanza aprendizaje. Este fenómeno se puede observar en dicho sector debido al uso creciente de las TIC en sus procesos vitales, donde la interacción con herramientas educativas en líneas, sistemas adaptativos, Cursos Online Masivos y Abiertos (MOOCs, siglas en inglés), entre otras plataformas, genera un gran cúmulo de datos educacionales que pueden ser aprovechados para encontrar conocimiento oculto y mejorar de forma sustancial y efectiva el proceso educativo docente (Santos, 2015).

Con la incorporación de las TIC, en conjunto con los enfoques pedagógicos más centrados en el alumno y el aprendizaje, el quehacer docente se amplía en todas direcciones. Con las TIC se crean espacios de enseñanza y aprendizaje no sólo en un aula convencional, aquella donde los estudiantes y el profesor se encuentran en el mismo tiempo y espacio, sino que se generan espacios virtuales donde, además de intercambiar información, se dan relaciones mediáticas, de formación, interacción, trabajo, colaboración e investigación (Ruiz Méndez y Aguirre Aguilar, 2013).

A esos cambios se suman las nuevas exigencias por parte de las instituciones que demandan respuestas ante el desarrollo vertiginoso de estas tecnologías (Colina y Bustamante Uzcátegui, 2009).

En el III Perfeccionamiento del Sistema Nacional de Educación (SNE) en Cuba, el Instituto Central de Ciencias Pedagógicas (ICCP), entre el 2010 al 2013 realizó el diagnóstico de la realidad del Sistema de Educativo que sirvió para saber cómo trabajar el tercer perfeccionamiento. El *estudio teórico del currículo actual* de las diferentes enseñanzas, reveló la sobrecarga de algunos programas y sus contenidos, exceso de horas clases en la medida que se transita por el

SNE, desbalance entre las horas clases de ciencias y humanidades y desactualización de los contenidos de los Planes y programas, Libros de Textos, orientaciones metodológicas y cuadernos de trabajos a partir del propio desarrollo de la ciencia, las exigencias sociales, los cambios operados en la sociedad, la necesidad de poner a la escuela a la altura de los tiempos, se están llevando a cabo transformaciones en los planes de estudios de las diferentes enseñanzas y una de ellas es la introducción de la programación y la robótica en el currículo escolar. Seleccionándose los niveles educativos para la implementación, la Educación Primaria hasta la Educación Media superior y la Educación Superior no está ajena, por ser la encargada de formar a los futuros profesionales de este sector.

Con el objetivo de flexibilizar el horario docente y los programas de estudios y propiciar una mayor participación de la familia y la comunidad. Además de acercarnos a los estándares TIC internacionales y con la programación en específico por contribuir al desarrollo del pensamiento computacional, al desarrollo del pensamiento lógico y algorítmico que tributa además al desarrollo de la creatividad, la experimentación con nuevas ideas, la comunicación, la perseverancia, las habilidades de solución de problemas, de diseño, de colaboración todas tan necesarias para formar ciudadanos competentes en entornos tecnológicos y digitales.

También se ha retomado por su relación con la Robótica, ciencia o rama de la tecnología que se impone cada vez más en los tiempos actuales y para el futuro.

El Perfeccionamiento se previó en tres etapas: en un curso se trabajó en la experimentación a través de la introducción, el próximo se enmendaron los resultados y se generalizaron en el curso siguiente. Las cuales se experimentaron y comenzaba a generalizarse, pero la situación de la pandemia no lo ha permitido.

El Sistema Nacional de Educación en el III Perfeccionamiento, tiene 124 asignaturas en 16 disciplinas. Una de las disciplinas es Informática, con cuatro líneas directrices que son: Utilización de juegos y software educativos, procesamiento multimedia de la información, el empleo de Herramientas de productividad y la Programación (pensamiento computacional).

El estudio de la Programación en la Educación Primaria comienza en 3er y 4to grado con el Lenguaje de programación ScratchJr, en 5to y 6to grado se estudia el Scratch, que se continúa profundizando en la Educación Secundaria Básica, en los grados 7mo y 9no, donde se abordan conceptos básicos de programación y elementos de lógica de programación y se introduce en este último grado otro Lenguaje de Programación: LiveCode que se implementa hasta el 11 grado de la Educación Preuniversitaria con la Programación basada en objetos y dirigida por eventos.

La alternativa aplicada comienza con el estudio de la programación porque sienta las bases para introducir posteriormente la robótica educativa como disciplina.

Por lo que, padres, tutores y familia en general deben de estar capacitados para ayudar a los niños(as) y jóvenes en desarrollar su pensamiento computacional, a través de algoritmos en la solución de problemas de la vida cotidiana o docente.

Esta investigación contribuye a que los docentes se preparen técnica y metodológicamente apropiándose de nuevas formas de enseñanzas acorde a las exigencias actuales en especial con la inclusión de las tecnologías aplicadas a la enseñanza a través de la aplicación Scratch. La praxis permitió constatar insuficiencias de los docentes de Informática y ramas afines a ella en cuanto al escaso conocimiento en fundamentos de programación y de la aplicación Scratch.

Este trabajo tiene como propósito socializar la superación impartida a los docentes. El objetivo general de la misma es contribuir a la preparación de los mismos aplicando los fundamentos de programación desde la aplicación Scratch, de manera que permita la apropiación de los conocimientos y habilidades para dirigir el proceso de enseñanza-aprendizaje

y su posterior vínculo con la robótica.

Se propone la aplicación Scratch para la enseñanza de la programación, por ser “un lenguaje de programación gráfico de fácil uso, donde se aprende a seleccionar, crear, manejar e integrar textos, se pueden mezclar imágenes, sonido y movimiento para uso educativos”.

2. Desarrollo

2.1. Materiales y métodos o Metodología computacional

Esta investigación es un resultado del proyecto nacional “Introducción paulatina de la enseñanza de la Robótica en la Educación General” perteneciente al Ministerio de Educación. Una de sus tareas es la capacitación y superación científica de especialistas, profesores, investigadores sobre la programación y la robótica educativa.

Se recepcionaron 17 solicitudes, de ellas, fueron aprobadas (para pre-matrícula) 16 profesionales. De estas solicitudes aprobadas, 13 hicieron efectiva la matrícula a través de la plataforma. La Figura 1 muestra la relación entre solicitudes aprobadas, matrícula inicial y matrícula final como ilustra, tomado del informe final (Dirección de Educación de Posgrado, 2020).

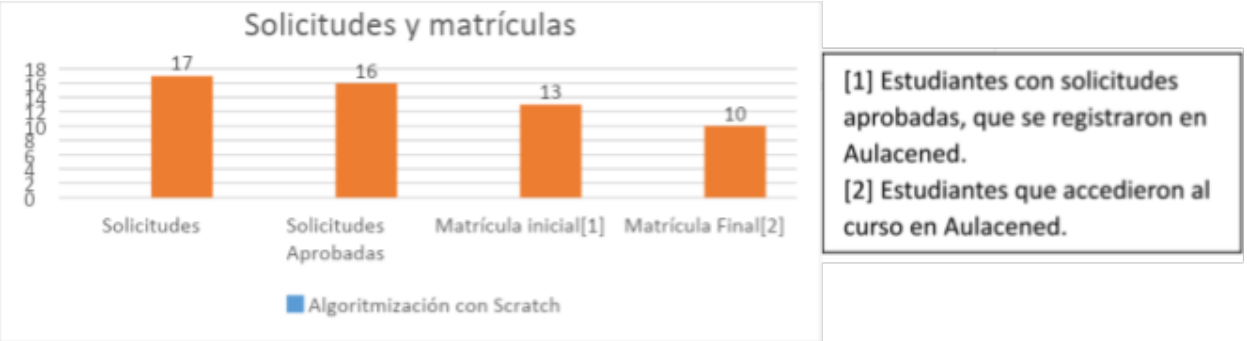

Figura 1

Se aplicaron distintos métodos que permitieron diagnosticar el conocimiento que ellos poseen sobre programación y de las principales características del Lenguaje de Programación Scratch, su empleo en el proceso de enseñanza aprendizaje, resultados que permitieron diseñar, planificar y ejecutar la superación que se propone.

El curso de superación “Algoritmización con Scratch” es una experiencia desarrollada en la Universidad de las Ciencias Informáticas en la Escuela Internacional de Verano a Distancia 2020 (EVD2020) en La Habana, impartido por profesores de dos instituciones educativas: la Universidad de Oriente de Santiago de Cuba y la UCI de la provincia La Habana.

El mismo, estuvo estructurados por 4 temas fundamentales:

1. Interactuando con Scratch. Estructura lineal o secuencial.
2. Estructura de control alternativa o condicional.
3. Estructura de control repetitiva, iterativa o cíclica.
4. Introducción al desarrollo de videojuegos.

Se les ofreció bibliografías de forma general, que sirvió de base para consultar y profundizar en los contenidos de los diferentes temas. Además de guías didácticas de carácter general o específico, de actividades, juegos,

entretenimiento, recursos educativos y orientaciones para desarrollar cada evaluación.

Cada tema propició que aplicaran y sistematizaran conceptos y procedimientos algorítmicos de acuerdo a la estructura de control estudiada, que les permitió resolver diferentes problemas del ámbito docente y de la vida práctica. Además, para la impartición de los contenidos se tuvo en cuenta contenidos de asignaturas que se imparten desde el plan de estudio de las enseñanzas Primaria hasta el nivel superior que permitieron sentar las bases de la Robótica Educativa.

Cada tema finalizó con actividades evaluativas, permitiéndoles crear programas, juegos y videos juegos sencillos. Algunos de los docentes se apoyaron en sus hijos para la aplicación de conocimientos de las asignaturas del plan de estudio de las diferentes enseñanzas, permitió el desarrollo de habilidades intelectuales, motoras, sociales, y de trabajo en equipo, reforzando el conocimiento en las demás ciencias como evidencia de la factibilidad de la aplicación y adquirieron habilidades como:

Caracterizar los diferentes bloques de programación de la aplicación Scratch.

Explicar los procedimientos básicos y su funcionabilidad en la solución de problemas.

Modelar diferentes actividades para la solución de problemas.

Resolver problemas del ámbito docente o de la vida cotidiana que permita que los estudiantes desarrollen:

- El pensamiento crítico y sistemático.
- Identificación, formulación y solución de problemas.
- Creatividad y curiosidad intelectual.

Habilidades de comunicación.

Habilidades interpersonales: Adaptabilidad, responsabilidad social y trabajo colaborativo.

Para el desarrollo de este curso, se contó con medios tales como: medios de comunicación, computadoras, internet, teléfonos, que permitieron desarrollar diferentes escenarios tecnológicos: Sin conectividad, conectividad parcial o limitada y conectividad total.

Esta modalidad de la Educación a Distancia tuvo sus ventajas tales como: Acceso a los beneficios de la tecnología educativa, obviar las limitaciones de tiempo y espacio y la posibilidad de estudiar individual y en equipos, independientemente de las distancias físicas o temporales.

Para lograr esto fue necesario:

- Una propuesta curricular flexible, adaptable a las condiciones e intereses de los estudiantes.
- Materiales didácticos relevantes, interesantes y motivadores.
- Evaluaciones integrales, multimétodos y formativas.
- Aprovechamiento de todas las facilidades que presenta la red para alcanzar aprendizaje significativo.
- Docentes y alumnos comprometidos con la calidad de la educación.

Se utilizaron algunas herramientas para su implementación de la enseñanza a distancia y semipresencial: Correo electrónico, Videoconferencia, Foro virtual, Chat académico y otros.

3. Resultados y discusión

La matrícula inicial fue de 13 y culminaron exitosamente 10 estudiantes. Se superaron 3 cuadros y 1 trabajador del sector no estatal. Además, 2 trabajadores de la UCI y 4 graduados de la UCI. La Tabla 1 describe la composición de los egresados.

Curso	Mat. Final[1]	Total Egresados	Externos	Trab. UCI[2]	Ext. Conv.[3]	Ext. Comp.[4]	Cuadros	Grad. UCI[5]	No Estat.[6]
Algoritmización con Scratch	10	10	8	2	0	0	3	4	1

Tabla 1: Composición del egresado.

Para obtener los resultados alcanzados, se realizó una encuesta de satisfacción a los cursistas con los siguientes indicadores: aspectos académicos, recursos empleados y valoración general. Se alcanzó unos $4,94$ puntos en el control de la calidad del curso desarrollado. La Tabla 2 describe los puntos alcanzados en cada uno de los indicadores.

	Respuestas						Promedio
Aspectos académicos	5	4	3	2	1	NS	
Valore la guía de estudio del curso.	8	1	1	0	0	0	4.70
Cumplimiento de los objetivos.	9	1	0	0	0	0	4.90
Labor desempeñada por los profesores.	10	0	0	0	0	0	5.00
Dominio del contenido por parte de los profesores.	10	0	0	0	0	0	5.00
Orientación, interacción y retroalimentación brindada por los profesores.	10	0	0	0	0	0	5.00
Criterios de evaluación utilizados.	10	0	0	0	0	0	5.00
Exigencia de los profesores en las evaluaciones.	10	0	0	0	0	0	5.00
Carga de trabajo asignada al estudiante para desarrollar el curso.	10	0	0	0	0	0	5.00
Canales de comunicación (foros, chats, correo. . .) utilizados.	9	1	0	0	0	0	4.90
Recursos empleados							
Bibliografía recomendada	9	1	0	0	0	0	4.90
Variedad de recursos utilizados	10	0	0	0	0	0	5.00
Calidad de los recursos empleados	9	1	0	0	0	0	4.90
Valoración general							
Cumplimiento de las expectativas	9	1	0	0	0	0	4.90
Satisfacción del curso	9	1	0	0	0	0	4.90
Evaluación general del curso	132	7	1	0	0	0	4.94

Tabla 2: Resultados de la encuesta, según indicadores.

Logros:

- Revisión continúa de la calidad técnico-didáctica del curso.
- Evaluación satisfactoria de los aspectos generales del curso, así como, los recursos y actividades empleados en cada tema.
- Creación video con el objetivo de explicar el valor práctico del curso, así como los objetivos y temas abordados.

4. Conclusiones

- Se obtuvo una alta satisfacción por parte de los cursistas con la forma de impartición del curso, debido a su calidad y especialmente con la preparación y la labor desempañada por sus profesores.
- Se desarrolló por primera vez una escuela de posgrado a distancia, que planteó retos y ventajas de esta modalidad a organizadores, profesores y estudiantes.
- La mayor fortaleza de la EVD2020 estuvo en la participación de un claustro conformado por diferentes instituciones, motivado y comprometido con una educación de posgrado virtual y sostenible.

Bibliografía

Alcántara Trapero, M. (2009). Importancia de las TIC para la Educación. Revista Digital Innovación y Experiencias Educativas.

Chaves Torres, A. (2017). La educación a distancia como respuesta. Revista Academia & Virtualidad 10(1): 23-41, 2017, 41.

Colina, L. (2008). Las TIC en los procesos de enseñanza - aprendizaje en la educación a distancia. Obtenido de Redalyc: http://www.redalyc.org/articulo.oa?id=76111716015

Colina, L., y Bustamante Uzcátegui, S. (2009). Educación a distancia y TIC: transformación para la innovación en educación superior. Obtenido de Redalyc: http://www.redalyc.org/articulo.oa?id=78411785007

Collazo Delgado, R. (2004). Una concepción teórico-metodológica para la producción de cursos a distancia basados en el uso de las tecnologías de la información y las comunicaciones. Ciudad de la Habana

Dirección de Educación de Posgrado. (Septiembre 2020). Escuela Internacional de Verano a Distancia 2020. Universidad de las Ciencias Informáticas. Ciudad de La Habana. Cuba.

Escontrela Mao, R. (2008). Hacia un modelo integrador en el uso de las TIC en la educación a distancia. Apuntes y comentarios desde la investigación y la experiencia. Obtenido de Scielo: http://ve.scielo.org/scielo.php?script=sci_arttext&pid=S1010-29142008000300003

Falcón Villaverde, M. (2013). La educación a distancia y su relación con las nuevas tecnologías de la información y las comunicaciones. Obtenido de Scielo: http://scielo.sld.cu/scielo.php?script=sci_arttext&pid=S1727-897X2013000300006

Figueredo Rodríguez, R. M., Martínez Cabrales, R. L. y Figueroa Hernández, M. (2018). Scratch: Metodología para programar. Universidad de Oriente. Santiago de Cuba. Cuba.

García Aretio, L. (2002). La educación a distancia. De la teoría a la práctica. Editorial S: A.

González Herrera, C. Y., y Aragón Barreda, Y. L. (2020). La educación a distancia en Cuba: Modelo de educación a distancia en la Universidad de las Ciencias Informáticas. Obtenido de Serie Científica: https://publicaciones.uci.cu/index.php/serie/article/view/683

Herrera Ochoa, E. (2005). Concepción teórico-metodológica desarrolladora del diseño didáctico de cursos para la superación a distancia de profesores en ambientes virtuales de enseñanza- aprendizaje. Ciudad de La Habana.

Pérez Fernández, V. (2006). La preparación informática del docente para la educación a distancia en entornos virtuales de enseñanza -aprendizaje. Ciudad de La Habana.

Ruiz Méndez, M., y Aguirre Aguilar, G. (2013). Quehacer docente, TIC y educación virtual o a distancia. Obtenido de Apertura: http://www.udgvirtual.udg.mx/apertura/index.php/apertura/article/view/412/339

Torres Alfonso, A., y Manso Urbay, Y. (2020). Acciones estratégicas para la implementación de la Educación de Posgrado en la Modalidad a Distancia. Obtenido de Serie Científica: https://publicaciones.uci.cu/index.php/serie/article/view/547

Proyecto IdeoDigital: Implementando ciencias informáticas en el sistema escolar público chileno

Andreas Hein
ahein@kodea.org
Fundación Kodea (Chile)

Claudia Jaña
cjana@kodea.org
Fundación Kodea (Chile)

Catalina Lyon
Fundación Kodea (Chile)

Resumen

En los últimos 20 años, han habido esfuerzos a nivel mundial por avanzar en temáticas de alfabetización digital. Actualmente, existe interés por expandir la educación tecnológica para que jóvenes sean capaces de enfrentar la transformación digital de forma creativa. Por lo anterior, diversos países en el mundo han incorporado la enseñanza de conceptos de ciencias de la computación (CC) en el currículum escolar.

La evidencia sugiere que aprender CC puede contribuir a desarrollar habilidades para ser ciudadanos del siglo XXI tales como: enfrentar, analizar y diseñar soluciones a problemas, mejorar el desarrollo cognitivo, estimular el desarrollo económico, aumentar el interés en áreas STEM y potenciar la innovación.

En Chile, desde 1992 han habido iniciativas para introducir en centros escolares, infraestructura tecnológica, conectividad, recursos digitales, formación docente e incorporación de CC en el sistema escolar. Sin embargo, aún no se ha abordado sistemáticamente la enseñanza de CC en el currículo chileno. En esta línea la mayoría de los profesores en el sistema público, no son nativos digitales y se les dificulta entender conceptos informáticos.

Frente a esto, Fundación Kodea en alianza con Fundación BHP, se encuentra implementado la iniciativa IdeoDigital, cuya meta es proveer a los estudiantes del sistema escolar público chileno de conocimiento y habilidades digitales para alcanzar su pleno desarrollo, en un mundo en que la innovación y la tecnología serán la base para la creación y desarrollo de los campos y disciplinas del futuro.

La estrategia se enfoca en aprovechar el tiempo dedicado a la enseñanza de “tecnología” en la escuela para introducir la enseñanza de contenidos de CC basados en Code.org en una primera etapa. Para esto, contempla un innovador modelo de transferencia de capacidades cuya implementación es gradual; comienza con un piloto (2021) y, el segundo año, incorpora las demás regiones del país.

Para lograr la meta propuesta, la estructura general de implementación se basa en tres ámbitos de acción que se retroalimentan: (i) **Sensibilización:** Sobre la necesidad, viabilidad y beneficios de implementar CC en el sistema escolar del país. (ii) I**mplementación de las CC en el aula:** Desarrollando cursos de CC en base a Code.org, asociados con asistencia técnica educativa (ATEs) para capacitar profesores en CC y desarrollando una red de escuelas líderes. (iii) **Incidencia en las políticas públicas:** Trabajando con actores políticos para destacar la relevancia de las CC e instalar la importancia de las habilidades digitales.

Palabras clave: Ciencias Informáticas, Sistema Escolar Público.

1. Antecedentes

En los últimos 20 años, han habido importantes esfuerzos de alfabetización digital a nivel mundial. En América Latina, las políticas de alfabetización han estado fuertemente centradas en la reducción de la brecha de acceso a un computador y en favorecer el acceso a tecnologías de la información y comunicación (OCDE, 2020).

En este sentido, en Chile y en la región, los esfuerzos de alfabetización digital de los jóvenes se ha orientado principalmente a formar buenos usuarios de las aplicaciones de la tecnología, tales como ofimática y navegación de internet. En los últimos años, se está empezando a hacer evidente la necesidad de expandir el foco de la formación tecnológica, para dotar de elementos a niños y jóvenes para poder comprender cómo funcionan las tecnologías y sus principios fundantes, específicamente en el ámbito de las Ciencias de la Computación (CC) y el pensamiento computacional. Es importante educar a las nuevas generaciones para convertirse en agentes creativos del mundo digital y no solo en consumidores (Jara y Hepp, 2016).

Diversos estudios sugieren que la enseñanza de las CC, así como las habilidades de codificación, permite a los niños, niñas y adolescentes (NNA) ejercitar un amplio rango de habilidades como por ejemplo la solución de problemas matemáticos (Kalelioğlu, 2015), el pensamiento crítico, las habilidades sociales (comunicarse con otros), y la autogestión del aprendizaje (Popat y Starkey, 2019), habilidades de planificación (Arfé et al., 2020), pensamiento lógico, pensamiento reflexivo, resolución de problemas (Kalelioğlu, 2015). También existe evidencia que las habilidades desarrolladas a través de la enseñanza de la codificación en escolares, pueden transferirse a otros contextos (Scherer et al., 2019). Asimismo, se ha observado que algunas de las habilidades mencionadas pueden ser desarrolladas a través de la enseñanza de la codificación tempranamente, incluso desde la edad de los 4 ó 5 años (SÇiftci y Bildiren, 2020).

La alfabetización digital ya no es suficiente para enfrentar las demandas de la transformación digital. Por ello, muchos países desarrollados han revisado el currículum escolar para incorporar en el aula la enseñanza de conceptos de CC y desarrollar el pensamiento computacional en los alumnos. Inglaterra, por ejemplo, actualizó el plan de estudio nacional para reemplazar el ramo de Tecnología de la Información y las Comunicaciones con una nueva asignatura llamada Informática en el 2014 . Otro ejemplo es el caso de Estados Unidos, donde en 2016 se inicia la implementación de una iniciativa llamada “Ciencias Informáticas para Todos” cuya misión es convertir las CC de alta calidad en una parte integral de la experiencia educativa de todos los estudiantes y profesores de enseñanza básica y media del país.

2. El contexto Chileno

Desde 1990 se han implementado una serie de políticas orientadas a promover la alfabetización digital en el país. Entre ellas, se destaca el programa Enlaces, yo elijo mi PC, Me Conecto para Aprender y el Plan Nacional de Lenguajes Digitales.

Programa Enlaces: Fue creado 1992 con el objetivo de construir una red educacional nacional entre todos los establecimientos escolares que reciben subvención del estado, para incorporar las nuevas tecnologías de información y comunicación en la educación.

El programa se ha enfocado en desarrollo de infraestructura, redes y recursos digitales (software educativo, de productividad y recursos en Internet), capacitación y asistencia técnica a docentes, e incentivando la implementación de las tecnologías de información (TIC’s) en las prácticas de aula. También promueve la modernización y agilización de los procesos administrativos de profesores y directivos (CIDE, 2004).

Sus principales logros están relacionados con 1) la reducción de la brecha digital en profesores capacitando a la fecha a 210.852 profesores en uso de TIC; 2) cambiar la percepción del rol que la tecnología puede desempeñar en la educación; 3) desarrollar competencias esenciales del siglo XXI en los alumnos (búsqueda y selección de información, la comunicación y el trabajo en equipo, el análisis crítico y la resolución de problemas) y (4) brindar acceso a las nuevas tecnologías a través de las escuelas (Quiénes Somos, s/f).

Yo elijo mi PC (YEMP): Lanzado en el año 2009, se enfocó en lograr que estudiantes de 7mo básico, de establecimientos que reciben subvención del estado accedan y usen recursos tecnológicos para apoyar los procesos de aprendizaje (Dirección de Presupuestos, 2020).

Me Conecto para Aprender (MCPA): Implementado desde 2015, busca universalizar la entrega de computadores e internet a los estudiantes de 7mo básico de establecimientos públicos. Desde el año 2013 en adelante, los elementos principales entregados a los estudiantes seleccionados son un computador, acceso a internet gratuito por un año y acceso a una serie de plataformas educativas vía instalación de software o acceso mediante internet. Estos programas han mostrado tener impacto en incrementar habilidades y acceso a las YIC. Por ejemplo, los participantes de YEMP, muestran un incremento significativo en el conocimiento y manejo de las TIC (Universidad Diego Portales, 2012). Por otra parte, de los beneficiarios del programa MCPA, 85 % reporta que al menos un miembro del hogar utiliza actualmente el computador entregado, y 53 % declara usarlo todos los días. Pese a ello el 50 % de los participantes declara que rara vez un establecimiento educacional les pide utilizar el dispositivo (Pontificia Universidad Católica de Chile, 2017). En la mayoría de los casos, el no uso estaría asociado a fallas técnicas que no se pudieron reparar. El 88 % de los estudiantes beneficiados con un PC, reporta usarlo para buscar información para estudiar, el 75 % crea o edita documentos y el 77 % de los encuestados lee enciclopedias o diccionarios en línea, al menos una vez por semana (Katalejo, 2019).

Hoy en día, el 75 % de los escolares chilenos usan Internet regularmente y 92 % tienen acceso a laboratorios de computación en sus escuelas. Estas cifras posicionan a Chile como el país latinoamericano con mejor uso tecnológico.Pese a la alta cobertura en recursos, aún existen limitaciones respecto a cómo se utilizan las TICs dentro de la escuela. En este sentido, la práctica pedagógica de los docentes ha mostrado ser más difícil de cambiar (Hepp et al., 2017; Weng y Tang, 2014).

Plan Nacional de Lenguajes Digitales (PNLD): En la actualidad, existe interés por expandir la educación tecnológica para que los jóvenes sean capaces de enfrentar la transformación digital de forma creativa y no solo como meros usuarios de computadores e internet. Por ello, en 2018 se lanzó el PNLD que buscó impulsar la inclusión de las CC en el sistema escolar nacional a través de la capacitación de 120 docentes en la plataforma Code.org en formato presencial y a distancia. Se tradujeron íntegramente 6 cursos, 118 lecciones, 835 actividades para la comunidad educativa chilena. Esto permitió a los profesores acceder a materiales estructurados, traducidos y adaptados para enseñar Pensamiento Computacional y Programación en el aula. Esta experiencia contribuyó a posicionar la temática en el Ministerio de Educación contribuyendo a posicionar el debate sobre cómo incorporar la enseñanza de pensamiento computacional y programación en las escuelas (Kodea, 2019).

Si bien este programa logró un importante avance, aún la implementación de formación en ciencias de la computación y pensamiento computacional en las aulas escolares de Chile, no es sistemática (Kodea, 2019).

3. Iniciativa IdeoDigital

Con propósito de aportar a la expansión de la formación en CC en el aula, Fundación Kodea, en conjunto con la Fundación BHP y el patrocinio del Ministerio de Educación, diseña la iniciativa IdeoDigital. Esta iniciativa inició su implementación en el año 2021 y tiene una duración proyectada de 5 años. El objetivo principal de este proyecto es crear las condiciones necesarias para implementar las CC en el sistema escolar público en Chile. La estructura de implementación del proyecto se basa en acciones en tres componentes que se retroalimentan y se implementan en paralelo a lo largo del proyecto:

(I) ***Sensibilización en la comunidad educativa:*** El objetivo principal de este componente es incentivar a las escuelas a capacitar a sus profesores en CC y movilizar y comprometer a la comunidad educacional ampliada en su implementación. Para ello, se contempla el desarrollo de acciones dirigidas a autoridades educativas, comunidades directivas escolares, profesores, padres y alumnos. Se busca además generar adhesión a la agenda de ciencias computacionales e instalar conceptos clave de CC en la opinión pública. Para ello se proporciona a las comunidades educativas, ***información sobre la factibilidad y beneficios de aprender CC*** y el impacto que tiene el aprendizaje en CC a partir de experiencias nacionales e internacionales y testimonios en las habilidades de los alumnos. Estas actividades serán apoyadas por difusión de medios para instalar el valor de aprender CC. Su programación fue diseñada para proceder de manera incremental, comenzando en las regiones centrales de Chile y expandiéndose gradualmente para abarcar las 16 regiones. Asimismo se buscará crear una ***red de escuelas líderes*** en la incorporación de CC en el aula, que sirvan de modelos de éxito para inspirar a nuevas escuelas a capacitar a sus profesores en CC. Adicionalmente, Kodea trabajará con actores institucionales incluyendo el Ministerio de Educación para generar una distinción para las escuelas que hayan incorporado CC en sus aulas. Se espera que esta distinción refuerce la posición de esas escuelas en las comunidades educativas y que muestre a actores clave, como apoderados, que la escuela está tomando medidas activamente para asegurar que sus alumnos estén preparados para enfrentar los desafíos digitales del siglo XXI.

(II) ***Implementación de CC en el aula:*** Se contempla la implementación de un modelo de transferencia de habilidades, metodologías y materiales que permitan expandir sistemáticamente la implementación de la formación en ciencias de computación en el aula. Se tiene como meta formar a 850 profesores de 1000 escuelas en 5 años. Para ***disponibilizar los materiales*** a los profesores de escuelas públicas, material de clase mundial para enseñar CC. Por lo anterior, se desarrollarán cursos de CC basados en la plataforma Code.Studio de 1ero básico a 4to medio. De esta manera, todos los cursos están libremente disponibles para ser usados por cualquier escuela. En este sentido, ya se cuenta con traducción de los cursos de 1 a 4 básico. Se espera que la totalidad de los cursos se encuentren disponibles el en el año 2022 en la plataforma Code.org. Dado que la capacidad de Kodea para capacitar a profesores a gran escala es limitada, se requiere de la ***articulación de una red de socios*** de implementación capaz de capacitar profesores en CC por todo el país. Para ello, se trabajará con socios implementadores (ATEs), organizaciones sin fines de lucro, que tengan la capacidad de realizar programas de capacitación de profesores a los largo de 12 regiones del país y se especialicen en adopción de recursos tecnológicos en las escuelas. Kodea desarrolló un programa para capacitar a los facilitadores basado en Code Studio, para asegurar que los socios de implementación proporcionen capacitación de alta calidad a los profesores. Se buscan socios implementadores calificados en la enseñanza de CC basada en Code.Studio y, a la vez, competentes en habilidades bandas como: empatía, liderazgo y resiliencia. En la actualidad se han formado facilitadores de nueve ATEs. En una segunda

etapa se procederá a la expansión. La Figura 1, describe el modelo de transferencia.

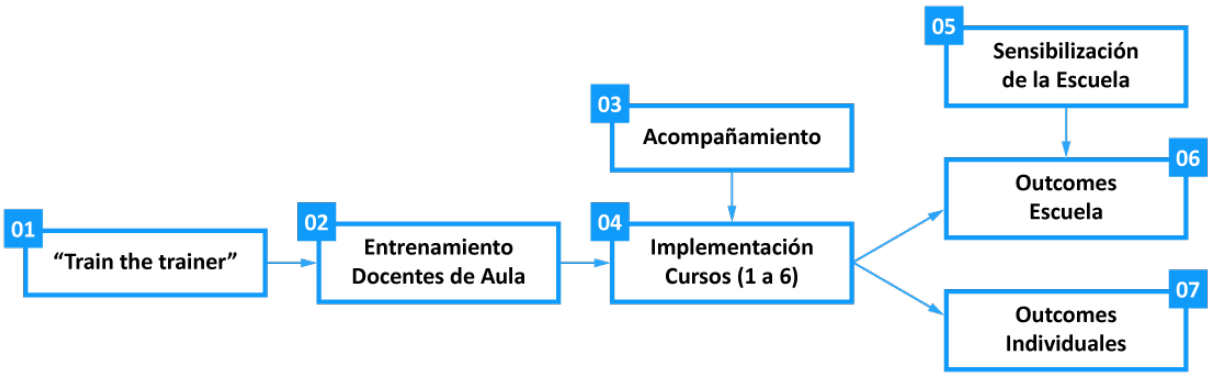

Figura 1: Síntesis del modelo de transferencia

El proceso de transferencia comienza por la preparación de un **equipo de facilitadores** (**"train the trainer"**) especialista en asesoría a escuelas quienes replican la capacitación a los docentes de aula y lideran el proceso de acompañamiento posterior. La preparación se basa en Code.Studio y es implementada por expertos de la fundación Kodea. Tiene una duración de 28-30 horas, que se distribuyen en 7 días, 4 horas diarias. En el contexto de la pandemia se han implementado en formato online. Los contenidos que se abordan son: Ciudadanía digital, Conceptos de CC y pensamiento computacional, plataforma Code, habilidades socioemocionales, currículum, y evaluación de aprendizajes. En una segunda etapa estos facilitadores proceden a entrenar docentes utilizando un formato **"Bootcamp"** de 20 horas que consideran alfabetización digital, formación práctica inmersiva basada en "Aprender Haciendo", Curriculum Code.org, plataforma y sus funcionalidades, competencias técnicas, competencias socio-emocionales, creativas y de pensamiento crítico. Los facilitadores luego **acompañan** a los docentes durante la implementación del currículo en el aula (6 horas por mes por escuela). Las escuelas pueden optar a diferentes formas de financiamiento público para la implementación. Se espera de este modo que los niños desarrollen habilidades relacionadas con el pensamiento computacional y que las escuelas desarrollen la capacidad de sostener la implementación en el tiempo.

Incidencia en políticas públicas: El principal objetivo de este componente es transformar el programa nacional de la asignatura de tecnología, promoviendo la inclusión de CC en el plan nacional y desarrollando un programa modelo de CC que sea aprobado por el Ministerio de Educación. Asimismo se busca apoyar este proceso mediante la generación y difusión de evidencia sobre cómo implementar efectivamente la enseñanza de las CC en el aula y su impacto y sus alcances. Kodea trabajará con actores relevantes en la comunidad educativa y legisladores para influenciar las políticas públicas destinadas a implementar CC en el Currículo Nacional.

Un elemento central de este aspecto tiene que ver con la sistematización de los aprendizajes y la generación de evidencia que permita informar la toma de decisiones. Para ello se ha diseñado una estrategia de monitoreo y evaluación que permita por un lado, ir informando el desarrollo y maduración del programa y por el otro generar evidencia que permita facilitar su escalamiento e informar la toma de decisiones de políticas públicas. En este sentido, el modelo de evaluación se enfocará en generar y gestionar conocimiento sobre los siguientes temas:

- Estrategias y prácticas que permiten superar barreras de implementación.
- Identificar factores críticos de éxito para la implementación de distintos componentes de la iniciativa.

- Determinar la efectividad de la iniciativa, con especial atención a las estrategias y los factores que pueden moderar sus resultados.

El modelo de monitoreo y evaluación combinará diversas actividades e hitos de evaluación interna y externa que permitan informar adecuadamente el desarrollo, escalamiento de la iniciativa, así como organizar y difundir las principales lecciones y aprendizajes. Los hitos se organizan según las etapas de desarrollo del programa: a) Aprendizaje y maduración, y b) Consolidación y escalamiento.

El primer hito del año 2021, es la implementación de la *evaluación externa de la implementación piloto del modelo de transferencia* (en implementación). Mediante el uso de encuestas, entrevistas y grupos focales el estudio se centra por una parte en la evaluación *del proceso de transferencia* (preparación de facilitadores; preparación de docentes estrategia de acompañamiento a los docentes, y barreras y facilitadores en general de la implementación y recomendaciones específicas de mejoramiento) y por otra parte en estudiar la *aceptabilidad del concepto del programa*, esto es caracterizar la disponibilidad de actores elegibles a implementar el programa e identificar potenciales barreras de implementación.

Así mismo en esta etapa se están desarrollando los instrumentos y estrategias de monitoreo y evaluación a ser implementadas en el año 2022. Los indicadores que se monitorearán activamente en esta etapa son, i) los resultados de aprendizaje en facilitadores y docentes, ii) el desarrollo de habilidades de enseñanza de las CC, iii)la satisfacción y actitudes en docentes y facilitadores, iv) las barreras y facilitadores de implementación y el v) uso y expansión de los cursos de CODE por parte de los establecimientos. Se contempla además en esta etapa la implementación de una evaluación de procesos interna, a fines del segundo año de implementación.

En la etapa de consolidación y escalamiento (2023-2025), se centrarán los esfuerzos en monitorear el progreso del escalamiento así como preparar e implementar una estrategia de impacto y sostenibilidad de la iniciativa. En esta etapa se contempla la implementación de dos evaluaciones de proceso externas (años 2023 y 2025) y la implementación de una evaluación de impacto externa entre los años 2023 y 2025.

Existen algunos diseños tentativos de la evaluación de impacto, pero su diseño final será resuelto a partir de la experiencia de implementación en la primera etapa.

En la Figura 2, se describen en detalle los componentes y actividades de la iniciativa.

4. Reflexiones al cierre

Si bien no hay dudas sobre la relevancia de incorporar sistemáticamente la enseñanza de las ciencias de la computación en el aula, los desafíos pendientes aún son inmensos.

Afortunadamente existe una amplia base de experiencia comparada en el mundo sobre distintos modelos y aproximaciones prácticas sobre su implementación. Del mismo modo la investigación académica en el área se encuentra en aumento contribuyendo a un creciente cuerpo de conocimiento sobre la relevancia y las posibilidades que tiene la enseñanza de las ciencias de la computación para el desarrollo del ciudadano del siglo XXI. Al definir la estrategia de implementación de la iniciativa de IdeoDigital, se ha optado por buscar contribuir activamente a la construcción de este cuerpo de conocimiento de modo de poder contribuir al esfuerzo global de trabajo en la materia.

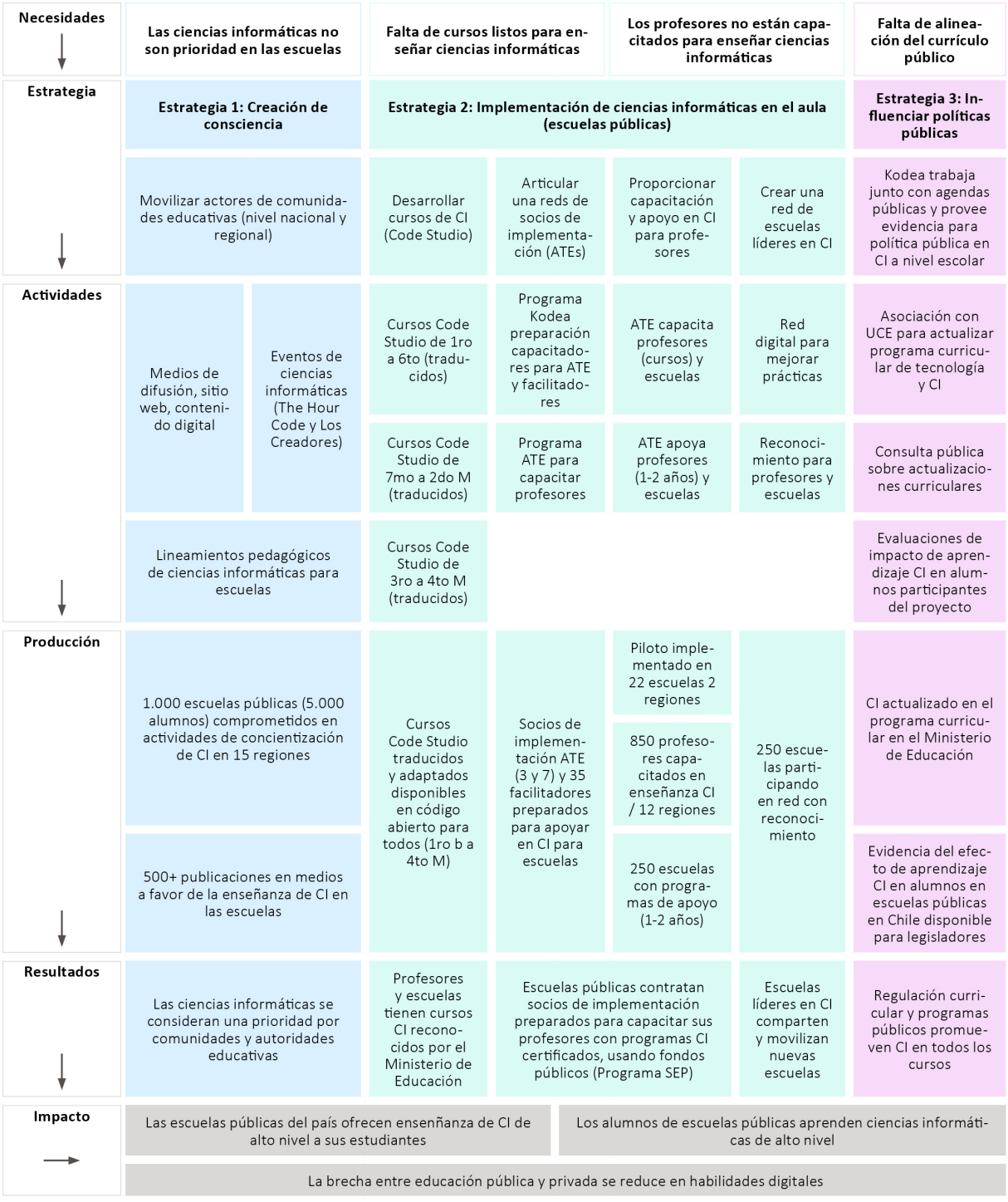

Figura 2: Componentes y actividades de la iniciativa.

Bibliografía

Arfé, B., Vardanega, T., y Ronconi, L. (2020). The effects of coding on children's planning and inhibition skills. Computers & Education, 148, 103807.

Çiftci, S., y Bildiren, A. (2020). The effect of coding courses on the cognitive abilities and problem-solving skills of preschool children. Computer science education, 30(1), 3-21.

10 Consorcio integrado por el Centro de Investigación y Desarrollo de la Educación [CIDE], INVERTEC-IGT, Universidad Alberto Hurtado, (2004). Informe final evaluación en profundidad.

Hepp, P., Pérez, M., Aravena, F., y Zoro, B. (2017). Desafíos para la integración de las TIC en las escuelas: Implicaciones para el liderazgo educativo. Informe Técnico, (2).

Jara, I., y Hepp, P. (2016). Enseñar Ciencias de la Computación: creando oportunidades para los jóvenes de América Latina.

Kalelioğlu, F. (2015). A new way of teaching programming skills to K-12 students: Code.org. Computers in Human Behavior, 52, 200-210.

Katalejo (2019). Evaluación de la satisfacción usuaria del Programa Becas TIC Yo Elijo mi PC y Me Conecto para Aprender: Informe final. Encargado por JUNAEB, Chile: Feller, C., Alvarado, P. y García, I.

Kodea (2019). Proyecto piloto plan nacional de lenguajes digitales.

OCDE (2020), Making the Most of Technology for Learning and Training in Latin America, https://doi.org/10.1787/ce2b1a62-en. 2020 OCDE, París

Popat, S., y Starkey, L. (2019). Learning to code or coding to learn? A systematic review. Computers & Education, 128, 365-376.

Scherer, R., Siddiq, F., y Sánchez Viveros, B. (2019). The cognitive benefits of learning computer programming: A meta-analysis of transfer effects. Journal of Educational Psychology, 111(5), 764.

Weng, C. H., y Tang, Y. (2014). The relationship between technology leadership strategies and effectiveness of school administration: An empirical study. Computers & Education, 76, 91-107.

PRIMERA SESIÓN DE PÓSTERS

Aprendiendo a desarrollar un intérprete de un lenguaje de programación. Un software educativo para entender cómo funciona un lenguaje

Francisco Sánchez Guijarro, Lucas Spigariol, Sergio Viera, Juan Bono, Nicolás Ulmete y Adrián Bielsa

La participación en competencias: Un modelo de aprendizaje. El caso del Club ATP

Gustavo Cierra

Club de Ciencias: Una experiencia de enseñanza y fomento de vocaciones científicas y tecnológicas

Gustavo Cierra

Computadoras, Simulaciones y Programas Paralelos en Escuela Primaria

Marcos J. Gómez y Nicolás Wolovick

Educación digital, Programación y Robótica en el sistema educativo municipal

Alicia Olmos, Mariana J. Perez, Gabriel Caeiro, Moira Ragona, Ileana N. Gómez, Pablo Cabral, Adrián Vera, Ariel Martín, Erich Kunath y Nicolás Laje

Hacia una propuesta didáctica para la enseñanza de la programación

Gustavo Astudillo y Silvia Bast

***LudiNEAndo*: Modelo de abordaje ludificado para la capacitación docente en los niveles inicial y primario en CC**

Carina Mellibovsky y Julieta Lombardelli

***Misconceptions* de Ciencias de la Computación en niños/as escolarizados/as**

Lucía Parral, Herman Schinca, Fernando Schapachnik y Hernán Czemerinski

Pensamiento Computacional con CodyRoby

Marcelo Fabián Páez

Primeras experiencias de formación sobre la programación y su didáctica en docentes secundarios de San Juan

Flavia Millán, Manuel Ortega, Marcelo Mondre y Johana Arce

Proyectos de Programación en la Universidad: una Experiencia de Evaluación y Trabajo Colaborativo en Pandemia

Natalia Colussi, Pamela Viale y Natalia Monjelat

Aprendiendo a desarrollar un intérprete de un lenguaje de programación

Un software educativo para entender cómo funciona un lenguaje

Francisco Sánchez Guijarro
franleplant@gmail.com
UTN - FRD

Lucas Spigariol
lspigariol@gmail.com
UTN

Sergio Viera
sergioviera@gmail.com
UTN - FRD

Juan Bono
juanbono94@gmail.com
UTN - FRD

Nicolás Ulmete
nicoulmete1@gmail.com
UTN - FRD

Adrián Bielsa
adrianbielsa1@gmail.com
UTN - FRD

Resumen

Se presenta una herramienta de software, cuyo desarrollo está en progreso en el marco de un proyecto de investigación (UTN - Facultad Regional Delta), que tiene como primer objetivo facilitar la comprensión del proceso de interpretación de un lenguaje de programación de alto nivel y luego posibilitar su modificación y extensión.

Su elemento central es un intérprete propiamente dicho y está articulado con una serie de componentes, entre los que se destacan los visuales, que lo convierten en un software con sentido pedagógico.

El contexto educativo en el cuál se gestó la idea y que a su vez se propone como ámbito de aplicación es en carreras universitarias de sistemas, informática o ciencias de la computación, pero no quita que pueda ser utilizado en otros trayectos formativos. No está pensada como primer acercamiento a la programación, sino que se aprovecha mejor la herramienta teniendo cierto conocimiento sobre la disciplina. Esta pensado con dos formas de uso, que más que ser alternativas habilitan a diferentes perfiles de estudiantes o nivel de profundidad que se quiere alcanzar.

La utilidad básica consiste en ubicarse como usuario de la herramienta: Se escribe una porción de código al hacerla evaluar se desencadena una serie de procesos, cuyos pasos y estructuras intermedias son mostrados mediante gráficos adecuados (cadena de tokens, representación del *Abstract Syntax Tree*, etc), hasta que se obtiene el resultado final. En caso que todo funcione adecuadamente, el énfasis está puesto por un lado en mostrar el paralelismo entre cada porción del código fuente y sus representaciones correspondientes, como también poder visualizar la secuencia interna de evaluación. Pero en caso que el código tenga algún problema o haya alguna inconsistencia en la evaluación, lo que se informa es dónde y por qué se produjo el error, utilizando los mismos gráficos anteriores para que sea más simple de localizar y apelando a un lenguaje acorde a quienes están aprendiendo.

Una utilidad avanzada consiste en modificar el componente del interprete mismo, y utilizando la misma interfaz y demás elementos de la aplicación poder ver cómo se comporta diferente. Para ello,

además de haber cuidado criterios de expresividad en el código y una adecuada modularización para que sea más sencillo de modificar, se dejaron adrede “huecos” en la formulación del lenguaje para habilitar a la formulación de consigna de trabajo acotadas y precisas.

Palabras clave: Lenguajes, Intérpretes, Software educativo, Programación.

La participación en competencias: Un modelo de aprendizaje

El caso del Club ATP

Gustavo Cierra
gustavocierra@gmail.com
Club de Ciencias ATP

Resumen

Desde el Club de Ciencias ATP de Villa María y la región se ha promovido la participación de los/as niños/as y jóvenes en olimpiadas y competencias de robótica y programación ya que en estos espacios se fortalecen habilidades, se integran conocimientos de diversas asignaturas, se potencia el trabajo en equipo, la socialización, la formación en valores.

El póster de caso describirá los objetivos de las experiencias, contenidos, herramientas y aprendizajes alcanzados por los/as niños/as y jóvenes en las competiciones en las que se participó: RoboRAVE, Robo Sensei, Robot Virtual Game, Olimpiadas Roboliga y Robocup y la Copa Robótica 2019 y 2021 donde jóvenes de la ciudad representaron a la provincia de Córdoba.

En estas experiencias, a través de distintas instancias y desafíos, los/as niños y jóvenes utilizan herramientas de la programación y la robótica para resolver situaciones, encontrar soluciones a problemas de la vida cotidiana en algunos casos.

El proceso de aprendizaje en estos espacios motiva la exploración, el análisis, la investigación, la búsqueda de alternativas orientadas a solucionar una situación y/o mejorar el entorno. Durante ese proceso aprenden, comparten conocimientos con los integrantes de sus equipos, con otros niños/as y jóvenes de otros equipos. En cada competencia está presente la cooperación, el intercambio basado en el respeto y la solidaridad, que es lo que facilita las tareas y actividades de cada uno y que se fortalece en estos eventos.

Muchos de los desafíos propuestos en las competencias fomentan la adquisición de habilidades STEAM (Ciencia, Tecnología, Ingeniería, Arte y Matemáticas) y requieren no sólo de conocimientos sino también de capacidad estratégica y de habilidades como la creatividad, la colaboración y el trabajo en equipo.

Palabras clave: Coopertición, Trabajo de equipo, Pensamiento científico, Resolución de problemas, Robótica educativa, Pensamiento Computacional, Aprender Haciendo.

Club de Ciencias: Una experiencia de enseñanza y fomento de vocaciones científicas y tecnológicas

Gustavo Cierra
gustavocierra@gmail.com
Club de Ciencias ATP

Resumen

El póster de caso describirá la metodología, contenidos y estrategias implementadas en el Club de Ciencias ATP para formar a niños/as de 9 a 12 años en saberes y prácticas fundamentales en la actualidad.

El Club de Ciencias ATP es una iniciativa del Fab-Lab "ATPlab" que se dedica a la educación basada en la tecnología, con el apoyo de la Tecnoteca y el Clúster de Impulso Tecnológico. Reúne actividades y talleres que se vienen desarrollando desde hace más de 10 años en la ciudad de Villa María. También se han abierto Clubes en otras localidades de la región, donde se replica la experiencia: Ticino, Idiazábal, Arroyo Algodón, Palestina, Tío Pujio, Oncativo.

En el Club, se desarrollan encuentros semanales entre un coordinador y niños/as interesados en participar en las actividades. Cada jornada se basa en una didáctica que combina estrategias lúdicas y recreativas, promoviendo la cooperación, el trabajo en equipo y el "aprender haciendo".

Los/as niños/as adquieren conocimientos sobre reacciones químicas, fenómenos físicos, electrónica, software libre, pensamiento computacional, robótica educativa, Alice, Scratch, Pilas Bloque y otras herramientas mientras los aplican en un desafío. Por ejemplo, con Tinkercad diseñan cohetes hidrodinámicos y utilizan la química, física, fundamentos de la ciencia aeroespacial y la aerodinámica para hacer los lanzamientos. De la misma forma, diseñan y fabrican juguetes, como un helicóptero, que luego en equipo programan para hacerlo volar o un auto que puede funcionar con energía solar. También han realizado la fabricación de chalecos luminosos para medios de transportes alternativos (monociclos, vehículos eléctricos) con guiño y luces led, pensando estrategias para mejorar la seguridad vial en la utilización de estos medios que sigue creciendo.

En el Club se promueve también la participación de los/as niños/as en cooperticiones y olimpiadas, espacios donde aprenden y comparten experiencias con otros equipos. Los/as niños/as del Club han participado en eventos mundiales como:-La hora del Código, Scratch Day, Flisol, CodeWeek Semana europea de la programación, Copa Robótica, Roborave.

Palabras clave: Ciencia de la Computación, Club de ciencias, Aprender haciendo, Aprendizaje divertido, Programar para aprender.

Computadoras, Simulaciones y Programas Paralelos en Escuela Primaria

Marcos J. Gómez
marcos.gomez@unc.edu.ar
UNC

Nicolás Wolovick
nicolasw@famaf.unc.edu.ar
UNC

Resumen

En simultáneo al interés de incorporar la enseñanza de la programación en todos los niveles educativos, comienza a ser necesaria la Computación Paralela. Actualmente, la mayoría de las computadoras que se utilizan cuentan con dos o más CPU. Feng et al., definen necesario que los programadores modernos sean capaces de escribir programas que utilicen las CPU en paralelo para poder aprovechar al máximo los recursos. A pesar de la importancia actual de la computación paralela, son pocos los programadores formados con las habilidades y prácticas para implementarla.

Podría ser relevante poder introducir conceptos relacionados a la computación paralela en edades tempranas. Son pocas las experiencias documentadas en las cuales se haga foco en la enseñanza de la computación paralela en las escuelas. Feng et al. realizan una extensión del entorno de enseñanza de programación basado en bloques *Snap!* para incorporar bloques que implementan conceptos relacionados a computación paralela. Ghafoor et al. diseñaron actividades unplugged para llevar la enseñanza de la computación paralela.

Nuestra experiencia fue pensada y diseñada para motivar el aprendizaje de la computación paralela en escuela primaria. En este caso, realizamos experiencias en contextos educativos reales en nivel primario para motivar el aprendizaje de la computación paralela. Para ello, diseñamos una secuencia didáctica basada en el análisis de simulaciones moleculares. Utilizamos los programas LAMMPS y OVITO para motivar la exploración de programas paralelos.

Palabras clave: Computación Paralela, Escuela Primaria, Simulaciones moleculares

Educación digital, Programación y Robótica en el sistema educativo municipal

Alicia Olmos*
aeolmos.municba@gmail.com
Municipalidad de Córdoba

Mariana Pérez*
cafrjo3@hotmail.com
Municipalidad de Córdoba

Gabriel Caeiro*
gabrielcaeiro@gmail.com
Municipalidad de Córdoba

Moira Ragona*
mragona@gmail.com
Municipalidad de Córdoba

Ileana N. Gómez*
ilengom@gmail.com
Municipalidad de Córdoba

Pablo Cabral*
pablosebacabral@hotmail.com
Municipalidad de Córdoba

Adrián Vera *
profesortecnologico@gmail.com
Municipalidad de Córdoba

Ariel Martín *
martin.ubp@gmail.com
Municipalidad de Córdoba

Erich Kunath*
erichkunath@gmail.com
Municipalidad de Córdoba

Nicolás Laje*
lajenicolas@gmail.com
Municipalidad de Córdoba

Resumen

La Municipalidad de la ciudad de Córdoba, a través de la Dirección de Aprendizaje y Desarrollo Profesional, dependiente de la Dirección General de Educación, Secretaría de Educación, cuenta en su política educativa con un trabajo persistente en materia de Educación Digital, Programación y Robótica (EDIPRO). La unidad EDIPRO, perteneciente a la Dirección de Aprendizaje y Desarrollo Profesional, está conformada por diversos profesionales que llevan adelante tareas de formación y actualización docente, asesoramiento técnico y pedagógico-curricular, entrega de equipamiento tecnológico y capacitaciones para su uso, entre otras.

Todas las acciones mencionadas son realizadas en base al objetivo de alfabetizar digitalmente para reducir la brecha digital y de ese modo, ampliar el horizonte de oportunidades de aprender, en un marco de inclusión socioeducativa con equidad y aprendizajes de calidad como principios básicos de la política educativa de un sistema educativo que pretende acompañar el desarrollo sostenible de la ciudad de Córdoba.

Palabras clave: Política educativa, Brecha digital, Educación digital, Programación, Robótica.

*Dirección de Aprendizaje y Desarrollo Profesional-Municipalidad de Córdoba.

Hacia una propuesta didáctica para la enseñanza de la programación

Gustavo Astudillo*
astudillo@exactas.unlpam.edu.ar
UNLPam

Silvia Bast*
bast@exactas.unlpam.edu.ar
UNLPam

Resumen

Hay una tendencia, a nivel global, que se enfoca en la generación de propuestas para la enseñanza de la programación de computadoras. Desde el grupo de investigación GrIDIE se viene desarrollando una propuesta didáctica para el aprendizaje de nociones básicas de programación. La misma fue evaluada desde el equipo docente de Introducción a la Computación (IC), concluyendo que al abordar lo conceptual, había dejado de lado la resolución de problemas. Por este motivo, se plantea la pregunta de investigación ¿Cómo incorporar a la propuesta de enseñanza de IC una estrategia de resolución de problemas que pueda ser apropiada por los estudiantes durante la cursada de la asignatura?

Durante 2020 se desarrolló e implementó una propuesta apoyada en la didáctica de la programación de la Fundación Sadosky (didáctica por indagación, abstracción y división en subtareas), y en la estrategia de resolución de problemas de Simon Thomson (comprender el problema, diseñar el programa, escribir el programa y mirar hacia atrás). La propuesta debió compatibilizarse con el uso del lenguaje Pascal y el EDI Lazarus.

Inicialmente, los docentes presentan la metodología resolviendo, junto con los estudiantes, problemas de las Guías Prácticas (GP). Posteriormente, se solicita que entreguen los planes de resolución en grupo, y finalmente, de forma individual. Estas entregas, junto con la encuesta de final de cursada fueron los insumos para el análisis/evaluación de la propuesta.

Las etapas de la propuesta se pueden resumir en: comprender el problema donde se analiza el enunciado y se pregunta ¿Qué hay que hacer?, las respuestas van definiendo las acciones que llevarán a resolver el problema (subtareas). En diseñar se propone un plan de resolución tomando como base el formulario (interfaz) que es parte del enunciado y se asocian a este las acciones (inicia con un verbo) en forma de órdenes.

En escribir el programa se coloca en cada procedimiento el plan en forma de comentarios en el código fuente, luego se traducen los comentarios a código Pascal. Finalmente, en mirar hacia atrás se revisa el programa en busca de mejoras. Del análisis de resultados, se puede inferir que la mayoría de los estudiantes se apropian de la metodología y que la propuesta tiene potencial para andamiar el proceso de resolución de problemas.

*GrIDIE, Departamento de Matemática, FCEyN, UNLPam.

Como trabajos futuros, se rediseñará el material de cátedra para adaptarlo a la propuesta y se profundizará en el análisis de las producciones para observar la evolución en la apropiación de la propuesta en 2021.

Palabras clave: Política educativa, Brecha digital, Educación digital, Programación, Robótica.

LudiNEAndo: Modelo de abordaje ludificado para la capacitación docente en los niveles inicial y primario en CC

Carina Mellibovsky
nea-cslp@nea.edu.ar
NEA Nueva Escuela Argentina

Julieta Lombardelli
innova@nea.edu.ar
NEA Nueva Escuela Argentina - UNQ -UNLP

Resumen

Desde un nuevo ecosistema de aprendizaje, la ludificación se ha incorporado gradualmente como metodología dentro de la innovación educativa, con el objetivo de incrementar la motivación, el dominio de habilidades, propiciar el establecimiento de instrucciones y la socialización. Esta disciplina cuyo término deriva del inglés gamification, y suele traducirse con el anglicismo gamificación, responde al uso de elementos y estrategias que son propias del campo de los juegos y videojuegos en espacios o entornos que tienen otros objetivos, para motivar y facilitar la realización o aprendizaje de un contenido específico.

Sus múltiples ejes de abordaje demostraron numerosos resultados positivos en su implementación en estudiantes para potenciar cambios significativos en los procesos de incorporación de saberes.

En este sentido y considerando las características y cualidades propias de la disciplina, este artículo se centra en el diseño de un modelo ludificado que hemos llamado LudiNEAndo, orientado a la capacitación docente en los niveles inicial y primario de NEA, Nueva Escuela Argentina. Situada en la ciudad de La Plata de la provincia de Buenos Aires.

De esta manera empoderar a los docentes de la institución como líderes del cambio y mediadores de procesos de aprendizajes. Desde un contexto situado que identifica a docentes de diversas disciplinas como el público objetivo entendiendo la importancia de generar estrategias específicas para asegurar el éxito del programa y conseguir los objetivos propuestos que en este caso se propone desarrollar las habilidades y competencias relacionadas con las tecnologías de la información, dándole un enfoque interdisciplinar transversal e integrador.

LudiNEAndo se origina para impactar positivamente en los procesos de enseñanza y aprendizaje de nuestros estudiantes, orientando los esfuerzos hacia la mejora continua a través de la capacitación y actualización de los equipos docentes. Se desarrolla en forma secuenciada en fases diagramadas de complejidad creciente con objetivos específicos en cada una.

La primera instancia de capacitación buscó que los docentes adquieran las competencias digitales básicas, incorporando las habilidades necesarias para trabajar con estrategias didácticas y metodologías ágiles que fomenten la motivación y la adquisición de aprendizajes significativos a través de la ludificación. En una segunda fase, se centró en los conceptos del pensamiento computacional y una introducción a la programación con herramientas seleccionadas acorde a la edad de los grupos de estudiantes en donde se implementará.

Palabras clave: Ludificación, Formación, Interdisciplinar, Programación Docentes.

Misconceptions de Ciencias de la Computación en niños/as escolarizados/as

Lucía Parral*
luciaparral@gmail.com
UBA

Herman Schinca
hschinca@fundacionsadosky.org.ar
Fundación Sadosky

Fernando Schapachnik
fschapachnik@fundacionsadosky.org.ar
Fundación Sadosky

Hernán Czemerinski
hczemerinski@campus.ungs.edu.ar
UNGS†

Resumen

Las *misconceptions* son razonamientos que constituyen sistemas explicativos para un cierto fenómeno que difieren de la definición correcta del mismo. Son difíciles de desarraigar, ya que se encuentran muy incorporadas al sistema de creencias de la persona que las posee y la explicación correcta del fenómeno no garantiza su desaparición. Conocer las misconceptions sobre un tema permite idear estrategias de enseñanza que las tengan en cuenta y de esta manera, sean más efectivas.

En este estudio, trabajamos con niños y niñas de Argentina de alrededor de 10 años de edad para entender si presentan misconceptions en algunos temas de Ciencias de la Computación. Para esto realizamos un cuestionario online que contenía distintas preguntas que nos permitieran entender si había misconceptions en temas tales como el almacenamiento de grandes volúmenes de datos en *YouTube*, la manera en la cual se comparten archivos por *WhatsApp* y qué pasa con los archivos compartidos, la infraestructura interviniente en la red de telefonía móvil y el por qué de la gratuidad de algunas aplicaciones en Internet. Encuestamos a 144 niños y niñas dentro de una modalidad de clase "virtual" con el docente presente. De esta manera nos aseguramos de que sus respuestas fueran lo más directas posibles y que pudieran dar sus opiniones sin intervención de otras personas o la posibilidad de buscar información en Internet.

Algunos de los hallazgos tras el análisis de los datos obtenidos en las encuestas son por ejemplo que, a pesar de utilizar YouTube cotidianamente, la mayoría de los alumnos y alumnas no conoce cómo funciona el almacenamiento de videos en la plataforma. Por otro lado, si bien entienden quién tiene acceso a los archivos almacenados en sus celulares y cómo se envían, existe una misconception en cuanto a si el archivo compartido es una copia o una referencia del archivo original. Pudimos observar que la cantidad de alumnos y alumnas con y sin misconceptions respecto al funcionamiento de la red de telefonía móvil

*Departamento de Computación, FCEyN, Universidad de Buenos Aires.

†Instituto de Ciencias, Universidad Nacional de General Sarmiento.

es similar, aunque la mayoría respondió no saber sobre el tema. El porcentaje de ellos que eligió como respuesta “La nube” es el menor.

Con estos resultados, podemos entrever algunos indicios de que piensan a “La nube” como un espacio de almacenamiento “etéreo” de datos más que un medio por el cual se envía y recibe información.

Palabras clave: Misconceptions, Didácticas Ciencias de la Computación, Nivel primario.

Pensamiento Computacional con CodyRoby

Marcelo Fabián Páez
paez@educ.ar
Escuela Municipal Dr. Juan B. Justo
Villa Siburu (Córdoba Capital)

Resumen

Al momento de entrar en la virtualidad del ciclo 2021, desde el espacio de Educación Digital Programación y Robótica se nos dificultaba el dictado de las clases virtuales y la elaboración de cuadernillos para estudiantes sin conexión. Por la falta de dispositivos, conectividad en las familias y una propuesta didáctica desconectada que aborde secuencias de programación.

Para poder abordar esta problemática utilizamos un recurso STEAM, cartas de "CodyRoby" (http://code.intef.es/cody-roby/) que es el nombre de un conjunto de juegos DIY ("Hazlo tú mismo") que proporcionan una manera fácil de empezar a jugar con robots y programación a cualquier edad, sin necesidad de usar ordenadores, tabletas o móviles. Elaboramos un material para distribuir en las entregas que hacían habitualmente les docentes, para que les estudiantes puedan realizar sus actividades de programación y robótica cómo lo venían haciendo en la presencialidad, con una propuesta unplugged. Cómo recurso para la difusión de la propuesta en WhatsApp, utilizamos la red social TikTok, que nos permite editar los vídeos y comprimirlos para que no superen el minuto de duración. Podíamos insertar audios, imágenes y grabar los desafíos de programación. En esta secuencia didáctica nos proponemos, en un contexto de virtualidad poner en juego el Pensamiento Computacional, con escasa disponibilidad de recursos y medios tecnológicos, en este sentido Francesco Tollucci nos aporta "El error fundamental de este tiempo fue no escuchar a los niños". En este sentido expone que les estudiantes:

> *...se hayan vuelto "transparentes e invisibles" para las autoridades a la hora de tomar decisiones. Y que muchas escuelas hayan seguido adelante con sus contenidos escolares a distancia sin incorporar la mirada de las infancias sobre el presente inédito del confinamiento.*

Teniendo en cuenta los aportes de Tollucci, nuestra intencionalidad, es partir de lo que sienten piensan y viven les estudiantes en contexto de pandemia, desarrollar actividades donde se pongan en juego el Pensamiento Computacional. En una primera instancia proponemos trabajar con cartas de "CodyRoby es el nombre de un conjunto de juegos DIY ("Hazlo tú mismo") que proporcionan una manera fácil de empezar a jugar con robots y programación a cualquier edad, sin necesidad de usar ordenadores, tabletas o móviles."

Palabras clave: CodyRoby, TikTok, Pandemia, Clases Virtuales, Programación, Pensamiento Computacional.

Primeras experiencias de formación sobre la programación y su didáctica en docentes secundarios de San Juan

Flavia Millan
flavia.millan@gmail.com
Universidad Nacional de San Juan

Manuel Ortega
manuel.ortega@gmail.com
Universidad Nacional de San Juan

Marcelo Mondre
mmondre@gmail.com
Universidad Nacional de San Juan

Johana Arce
johana.ailen.arce@gmail.com
Universidad Nacional de San Juan

Resumen

Desde el 2015, el Consejo Federal de Educación, por Resolución N 263/15, declaró de enseñanza estratégica, la programación durante la escolaridad obligatoria. Las Ciencias de la Computación, planteadas como potenciadoras del desarrollo de la curiosidad y creatividad en estudiantes. Atravesando y modificando casi todos los ámbitos de la experiencia humana. Si bien, son más que programación de computadoras, ésta, es fundamental, incrementa la creatividad, pensamiento lógico, precisión en la resolución de problemas, todas capacidades requeridas actualmente.

La Universidad, proporciona información sobre sus carreras a estudiantes secundarios. Sin embargo, se cree, insuficiente, para despertar vocaciones informáticas. Entonces, el Departamento de Informática de la Facultad de Ciencias Exactas, Físicas y Naturales, se presentó a la convocatoria 2019-2020, de Fundación Sadosky. Surgiendo, por primera vez, un proyecto de formación, avalado por el Ministerio de Educación provincial, focalizado en lograr que docentes provinciales, construyan experiencias educativas, al enseñar programación en sus aulas a sus estudiantes. Este proyecto de formación, impactó; directamente, al ofrecer una propuesta, hasta ese momento, ausente desde la Universidad a docentes de Nivel Secundario. Indirectamente, porque a través de profesores formados en Ciencias de la Computación, se aplicarían prácticas educativas efectivas para despertar vocaciones.

El proyecto de formación, dirigido, abierto a docentes del secundario, cualquier gestión y modalidad de: matemática, tecnología, proyecto tecnológico, informática, computación. Se implementó virtualmente entre agosto-noviembre del 2020, por plataforma digital de la Universidad, con encuentros sincrónicos/asincrónicos. Las actividades académicas, desarrolladas por cuatro docentes del Departamento, previamente formados por Fundación Sadosky. El proyecto, incluyó actividades didáctico-pedagógicas para guiar al profesor asistente, en la enseñanza de programación, cobrando significación su abordaje como objeto de estudio y no como herramienta al servicio de otros aprendizajes.

Finalizado el proyecto, se encuestó los 33 docentes que concluyeron la formación. Se recopilaron necesidades, aspiraciones, preocupaciones en los profesores a la hora de abordar estos contenidos en

sus aulas, con el afán de contextualizar la enseñanza y proponiendo buenas y efectivas prácticas educativas. Entre ellos, el 73 % manifiesta necesidad de enseñar programación en el secundario, el 60 % advierte falta de convenios con la Facultad que tiene carreras informáticas. El 76 % solicita fortalecer la formación continua. Al 42 % le pareció una propuesta original y al 70 % de los docentes le gustó la didáctica empleada. El 65 % expresó dificultades para enseñar a programar. Los datos obtenidos, retroalimentaron y fortalecieron la propuesta, ayudando a orientar la enseñanza de Ciencias de la Computación en el nivel secundario.

Palabras clave: La programación y su didáctica, Formación docente, Enseñanza de Ciencias de la Computación.

Proyectos de Programación en la Universidad: una Experiencia de Evaluación y Trabajo Colaborativo en Pandemia

Natalia Colussi*
colussi@fceia.unr.edu.ar
UNR

Pamela Viale*†
pamela@fceia.unr.edu.ar
UNR - UCA

Natalia Monjelat‡
monjelat@irice-conicet.gov.ar
CONICET - UNR

Resumen

El presente trabajo refiere a una experiencia de evaluación desarrollada durante el 2020 en la Facultad de Ciencias Exactas, Ingeniería y Agrimensura (FCEIA) dependiente de la Universidad Nacional de Rosario (UNR), particularmente en las cátedras de Programación de las carreras de Lic. en Ciencias de la Computación, Lic. en Matemática y Profesorado en Matemática. Desde la cátedra y considerando que se trata del redictado o segunda cursada de la materia, se propone una estrategia didáctica enmarcada en el Aprendizaje Basado en Proyectos y Problemas, con el objetivo de ofrecer a los estudiantes una posibilidad diferente de acceso a los contenidos propios de la asignatura. Aunque esta modalidad ha sido implementada en años anteriores con resultados altamente positivos, el paso a la educación virtual forzada por la pandemia, llevó a una revisión de la propuesta de evaluación y exposición de los trabajos grupales; tradicionalmente la exposición de los proyectos en el aula implicaba la defensa del mismo ante las preguntas de los compañeros/as de cursado y cuerpo de docentes, para posteriormente socializar lo realizado en las jornadas anuales que organiza la carrera de Ciencias de la Computación. En el nuevo contexto, para la instancia de evaluación final de los proyectos se diseñó un espacio que permitiera recrear la situación de intercambio y colaboración que se presentaba en la propuesta presencial, habilitando un sitio web especialmente diseñado al que se llamó “Vidriera de Exposición de Proyectos”. Cada grupo elaboró y subió a ese sitio un video donde defendía los proyectos realizados durante el cursado. Para fomentar la participación desde el diálogo y la reflexión crítica, la cátedra solicitó al alumnado una devolución hacia el trabajo de los grupos, empleando un formulario disponible junto a cada producción grupal. Para la realización de los proyectos, los estudiantes contaron con un espacio de acompañamiento semanal, organizado a modo de tutorías personalizadas donde cada grupo presentaba sus avances en función de objetivos pautados previamente, se realizaban correcciones y se acordaban nuevos objetivos para continuar delineando cada proyecto. Estas sesiones fueron grabadas y puestas a disposición de todo el alumnado. Los datos recolectados al finalizar la cursada dan cuenta de una alta recepción de la

*Facultad de Ciencias Exactas, Ingeniería y agrimensura (UNR), Rosario, Argentina.

†Facultad de Química e Ingeniería del Rosario (UCA), Rosario, Argentina.

‡Instituto Rosario de Investigaciones en Ciencias de la Educación (IRICE, CONICET-UNR), Rosario, Argentina.

actividad por parte de los estudiantes, quienes señalaron que la experiencia no solo les permitió revisar los conceptos trabajados, sino también conocer a sus compañeros en este contexto adverso y reflexionar sobre el proceso realizado.

Palabras clave: Evaluación en Virtualidad, Aprendizaje Basado en Proyectos y Problemas, Programación por Grupos, Pensamiento Computacional, Ciclo Inicial Universitario.

SEGUNDA SESIÓN DE PÓSTERS

Análisis de habilidades de pensamiento computacional en una asignatura de programación inicial en la universidad

Cristina L. Greiner, Gladys Dapozo, Raquel H. Petris, Maria Cecilia Espindola y Ana Maria Company

Aprendizaje automático para clasificación de actos de habla ilocutivos en mensajes de foros para educación a distancia

Mariano Piatti

Arte algorítmico

Fabio González

El abordaje de la alfabetización digital en dos propuestas de formación docente continua

Paola Roldán, Mariana Belluscio y Painé Pintos

Experiencia de introducción a la programación en formación inicial docente del Profesorado de Matemática

Silvina Elena Busto y Natalia Daiana Gomez

Nivel de satisfacción de los docentes en una capacitación en didáctica de la programación en tipos de pandemia

María Valeria Poliche, Hernán Cesar Ahumada, Daniel Armando Rivas, Nelson Contreras y Oscar Quinteros

Prácticas de prevención de phishing: una experiencia en la educación secundaria

Luciana Terreni

Taller de degustación. Programación Creativa

Marisa Elena Conde y Andrea Rocca

Zamba: una plataforma de aprendizaje como soporte a conceptos computacionales

Gustavo Del Dago, María Virginia Brassesco y Maximiliano Urso

Análisis de habilidades de pensamiento computacional en una asignatura de programación inicial en la universidad

Cristina L. Greiner
cgreiner@exa.unne.edu.ar
UNNE

Gladys N. Dapozo
gndapozo@exa.unne.edu.ar
UNNE

Raquel H. Petris
rpetris@exa.unne.edu.ar
UNNE

María C. Espíndola
mcespindola@exa.unne.edu.ar
UNNE

Ana M. Company
acompany@exa.unne.edu.ar
UNNE

Resumen

Este trabajo presenta los resultados de una evaluación de habilidades del pensamiento computacional realizada a estudiantes iniciales en una formación universitaria en Informática. El desarrollo de actividades que impliquen el uso de habilidades del pensamiento computacional permite que los estudiantes no consideren a la programación como una actividad puramente técnica, en el sentido de capacidad para escribir código, sino como la puesta en juego de un conjunto de habilidades combinadas de resolución de problemas. Detectar tempranamente las habilidades de menor desarrollo en los estudiantes permitirá planificar actividades o problemas que las fortalezcan.

Palabras clave: Enseñanza de la programación, Pensamiento computacional, Educación universitaria.

Aprendizaje automático para clasificación de actos de habla ilocutivos en mensajes de foros de aprendizaje de programación

Mariano Piatti
marianopiatti@mi.unc.edu.ar
Universidad Nacional de Córdoba

Resumen

En este trabajo se realiza una investigación sobre un conjunto de datos proveniente de Mumuki: un sistema online para aprender a programar, el cual cuenta con un foro de consultas donde los estudiantes consultan dudas sobre ejercicios de programación. A veces los estudiantes usan el foro para hablar entre ellos o para agradecer o comentar sobre mensajes previos. El objetivo de este trabajo es utilizar el conjunto de datos para entrenar modelos de aprendizaje automático que sean capaces de determinar si un mensaje requiere atención de un docente o no. Para esto, se decide utilizar los conceptos de forward looking act y backward looking act de los actos del habla ilocutivos de análisis conversacional: si un mensaje es un forward looking act, el mensaje requiere atención de un docente, mientras que si es un backward looking act, no necesita atención de un docente. En cursos multitudinarios, donde realizar un seguimiento personalizado es complejo, poder contar con un foro de consultas para generar un espacio de consulta, puede sumar más trabajo de seguimiento. Contar con herramientas que permitan reconocer mensajes que requieran atención, puede ser de importancia para identificar a estudiantes con dificultades. Para ello, primero se analiza el conjunto de datos el cual incluye los mensajes enviados y se concluye que los mensajes que abren un hilo de conversación son de una longitud menor en caracteres a los mensajes que ocurren luego del primer mensaje, lo cual también se debe a que un problema general es que los estudiantes no son profundos ni extensos a la hora de explicar sus dudas. Luego, se realizan las primeras observaciones que dan lugar a la idea de crear bolsas de palabras representativas de mensajes que son forward y backward looking acts, con el fin de utilizarlas para crear modelos de clasificación automática, que permitan brindar nuevas herramientas a los docentes para hacer un seguimiento de sus estudiantes.

Palabras clave: Aprendizaje automático, Foro educativo, Acto del habla, Procesamiento de lenguaje natural.

Arte Algorítmico

Fabio Fernando González
fabiofergon@gmail.com
EPAV Mantovani

Resumen

La experiencia a compartir, narra una secuencia de actividades llevadas a cabo en la EPAV "Prof. Juan Mantovani", en la ciudad de Santa Fe. Se trata de una escuela con especialización en artes visuales y audiovisuales. Ofrece niveles secundario y superior; esta experiencia ha sido implementada en ambos niveles, en la materia Experimentación Multimedial, que explora los innumerables cruces entre arte y tecnología contemporánea.

La actividad pretende tender puentes entre dos disciplinas (aparentemente) tan lejanas como el arte y la algoritmia. Se toma como antecedente de este tipo de manifestaciones, la obra de Sol Lewitt. Artista creador de una obra de carácter minimalista, geométrica y conceptual.

Rompió con lo establecido, al enunciar sus obras como si fueran algoritmos para que otros las ejecuten. Algo tan habitual en la música, como separar la autoría enunciada en un pentagrama (una especie de algoritmo), de la interpretación llevada a cabo por otro músico.

Hoy día se montan muestras en homenaje a Sol Lewitt, tomando sus indicaciones claras y precisas, que permiten recrear sus obras. Estas reflexiones en torno a otras formas de crear arte, nos conducen a enunciar secuencias ordenadas de instrucciones para que las ejecute un dispositivo digital. Y ¿qué es esto si no es programar?

Processing es el entorno de programación elegido para ese desafío. Fue concebido como puerta de entrada para la programación creativa. Hecho por artistas (desarrollado en el MIT), para que lo usen artistas. El inmediato feedback visual que devuelve Processing, genera un efecto casi hipnótico en los estudiantes. El tedio que usualmente genera el estudio del (aparentemente abstracto) espacio cartesiano, se ve diluido ante el interés por dibujar mediante código en ese espacio que se va descubriendo.

La introducción de los conceptos de estructuras de control, variables y condicionales (característico de cualquier sistema programable), permite programar sketchs más complejos, y controlar experiencias interactivas: aunque sea algo tan simple como detectar la pulsación de teclas, o la posición del mouse, permite tomar decisiones que modifican nuestros dibujos.

Finalmente, con la inclusión de Arduino como interface para medir variables físicas, ciertas alteraciones del entorno (luz, temperatura, sensores, etc.) terminan modificando y construyendo las obras de arte que resultan interactivas. La apropiación respetuosa para la reutilización y el remix de código open source compartido en plataformas como OpenProcessing, es propiciada y alentada para comprender las dinámicas de creación propias de nuestros tiempos.

Palabras clave: Programación creativa, Algoritmos, Arte, Arte algorítmico, Processing, Arduino.

El abordaje de la alfabetización digital en dos propuestas de formación docente continua

Paola Roldán*
proldan@isep-cba.edu.ar
ISEP

Mariana Belluscio*
mbelluscio945@isep-cba.edu.ar
ISEP

Painé Pintos*
ppintos@isep-cba.edu.ar
ISEP

Resumen

La preocupación por la alfabetización digital no es algo nuevo en las políticas educativas argentinas. Para nombrar una referencia insoslayable, en el marco de la Ley de Educación Nacional Nº 26.206 - 2006, se propone el desarrollo de propuestas de "opciones educativas basadas en el uso de las tecnologías de la información y de la comunicación y de los medios masivos de comunicación social" (pág. 17, artículo 100). Además señala que estas tecnologías no ingresarán sólo como herramientas para el desarrollo de propuestas de enseñanza sino también como "parte de los contenidos curriculares indispensables para la inclusión en la sociedad del conocimiento" (pág. 15, artículo 88).

Lo que si se renueva a medida que las tecnologías digitales se entraman de modo cada vez mas profundo en nuestros usos cotidianos y en las prácticas docentes, es la revisión y la resignificación del concepto mismo de alfabetización digital. Concepto que abarca desde competencias para un uso crítico e inteligente hasta marcos conceptuales que permitan pensar estas tecnologías, no sólo como herramientas sino entramadas en el mundo social y en la manera en que los seres humanos interactuamos con otros y producimos sentidos y saberes.

Con el convencimiento de que es necesario formar y proveer a los futuros docentes de herramientas que les permitan habitar, pensar, estudiar y recrear ese mundo desde la escuela, en el ámbito educativo provincial se se vienen desarrollando diversas experiencias que buscan afrontar ese desafío. Desde ISEP se vienen desarrollando diferentes propuestas de formación docente que han puesto a los medios y tecnologías digitales como objeto de estudio. En este sentido se han diseñado e implementado postítulos, profesorados, formaciones académicas, seminarios y talleres. En el presente trabajo se toman como casos para el análisis la actualización académica "Enseñar con herramientas digitales" y la Especialización Docente de Nivel Superior en Tecnologías Digitales y Educación.

El presente trabajo describe como en estas propuestas se aborda la alfabetización digital y resignifica el concepto de tecnologías digitales integrándolo a otros conceptos de uso más extendido como TIC, medios masivos de comunicación, herramientas y medios digitales y algunas nociones de las ciencias de la computación. Al mismo tiempo con este trabajo nos interesa explicitar algunos puntos de encuentro

*Instituto Superior de Estudios Pedagógicos (ISEP).

entre las diferentes prácticas de alfabetización que se contemplan en ambas propuestas y que definen una base común en relación a la alfabetización digital, que va más allá de un abordaje disciplinar concreto.

Palabras clave: Alfabetización digital, Tecnologías digitales, Tecnologías de información y comunicación, Formación docente continua.

Experiencia de introducción a la programación en formación inicial docente del Profesorado de Matemática

Silvina Elena Busto*
silvinabusto@gmail.com
ISFDyT 24 Bernal

Natalia Daiana Gómez*
nataliadaiangomez@gmail.com
ISFDyT 24 Bernal

Resumen

La experiencia se desarrolló en la asignatura Computación del cuarto año del Profesorado de Matemática del ISFDyT N° 24 Bernal. El objetivo de programar suele ser resolver un problema pero lo importante es que desarrolla una serie de capacidades de orden superior.

Las estudiantes que investigaron sobre los lenguajes de programación y su aplicación a las matemáticas, diseñaron una presentación digital que expusieron durante la clase y luego compartieron con sus compañeros mediante el aula virtual del curso, también prepararon el blog https://progscratch.blogspot.com, que permitió a los compañeros profundizar sobre la propuesta. Luego de la exposición de las compañeras, los estudiantes tuvieron el desafío de crear un proyecto que respondiera y sirviera para desarrollar un contenido matemático. Se propusieron varios proyectos, desde la aplicación del Teorema de Pitágoras, el cálculo de raíces de una función cuadrática, hasta un proyecto que realiza operaciones aritméticas de variada complejidad. La docente y las estudiantes que expusieron, realizaron un seguimiento del proceso que los estudiantes fueron realizando, acompañándolos y salvando las dudas y dificultades que surgían, tomando siempre el error como estrategia de aprendizaje.

Se propuso desde un primer momento la coevaluación entre pares, para lo que las estudiantes prepararon una rúbrica, considerando que este era el instrumento de evaluación más adecuado, ya que deja bien claro lo que se espera de los estudiantes. Se dejó un documento con la misma a disposición de los estudiantes, para que consultarán cómo iban a ser evaluados, qué aspectos tener presentes para la realización de la actividad y que tuvieran conocimiento de los ejes centrales de la misma, siendo ellos la creatividad del proyecto realizado, el funcionamiento del programa, la interfaz gráfica y la programación.

Palabras clave: Programación, Matemática, Scratch, Formación docente, Pensamiento lógico.

*Instituto de Formación Docente y Técnica Nro 24 de Bernal. Profesorado de Matemática.

Nivel de satisfacción de los docentes en una capacitación en didáctica de la programación en tipos de pandemia

María V. Póliche*
vpoliche@tecno.unca.edu.ar
Universidad Nacional de Catamarca

Hernán C. Ahumada*
hcahumada@tecno.unca.edu.ar
Universidad Nacional de Catamarca

Daniel A. Rivas*
darivas@tecno.unca.edu.ar
Universidad Nacional de Catamarca

Nelson A. Contreras*
nacontreras@tecno.unca.edu.ar
Universidad Nacional de Catamarca

Oscar E. Quinteros*
oequinteros@tecno.unca.edu.ar
Universidad Nacional de Catamarca

Resumen

El departamento de Informática de la Facultad de Tecnología y Ciencias Aplicadas de la Universidad Nacional de Catamarca resulto nuevamente seleccionada por la Fundación Sadosky para dictar el curso "La Programación y su didáctica – Método Program.AR – Parte 1", durante el año 2020-2021. Para esta ocasión y dada la situación de pandemia la capacitación se llevó a cabo a distancia y en forma VIRTUAL. El curso estuvo dirigido a docentes de niveles Primario, Secundario y Terciario. El objetivo de la capacitación fue que los docentes cursantes adquieran conocimientos básicos de programación, así como estrategias pedagógicas y recursos didácticos adecuados para su enseñanza a niños y jóvenes. El curso consto de 14 encuentros con actividades síncronas (videollamadas) y asíncronas (aula virtual moodle). Las clases síncronas por videollamada se realizaron los días sábados desde las 9hs. hasta las 12hs. Participaron 138 docentes de la provincia de Catamarca, siendo aproximadamente el 65 % de ellos del interior provincial, habiendo aprobado 80. A los efectos de conocer el grado y nivel de satisfacción de los docentes sobre la modalidad del dictado implementada en contexto de pandemia, se realizo una encuesta electrónica través a Google forms, obteniendo resultados muy satisfactorios.

Palabras clave: Aplicación, Formación Docente, Pensamiento Computacional, Catamarca.

*Fac. de Tecnología y Ciencias Aplicadas - Universidad Nacional de Catamarca. Maximio Victoria 55, 4700 Catamarca, Argentina.

Prácticas de prevención de phishing: una experiencia en la educación secundaria

Luciana Terreni
luciterreni@gmail.com
Instituto Sedes Sapientie - Escuela N°5 Soldado Carlos Mosto

Resumen

El phishing es el abuso informático en el que el perpetrador busca obtener información confidencial acerca de la víctima haciéndose pasar por otra persona, empresa, sitio o entidad en quien la víctima confía, a la manera de un cebo que induce al damnificado a morder el anzuelo y proporcionar la información. Posteriormente, la información robada puede usarse para fines espurios, tales como enviar mensajes a nombre de la otra persona o realizar compras por Internet.

Este trabajo expone una secuencia didáctica desarrollada con alumnos de quinto año de la escuela secundaria N°5 Soldado Carlos Mosto en el espacio Informática Educativa, de la ciudad de Gualeguaychú, Entre Ríos. La experiencia tenía como objetivo acercar a los estudiantes al concepto de phishing y a prácticas de prevención del mismo, incursionando en términos como privacidad, ingeniería social e identidad digital. El abordaje desde la indagación y la construcción colaborativa de conocimiento, permitió no solo la construcción conceptual en torno al phishing sino también la formación de competencias específicas y transversales. Dentro de las especificas podemos mencionar la construcción de criterios para establecer puntos críticos en la detección de amenazas por phishing y estrategias para la generación de claves y contraseñas. Dentro de las transversales se construyeron y fortalecieron competencias digitales a partir de la experimentación con TIC y las socio-comunicativas formadas a partir de la producción y socialización de material audiovisual.

Palabras clave: Phishing, Identidad Digital, Ciudadanía Digital.

Taller de degustación. Programación Creativa

Marisa Conde
marisacon@gmail.com
UNPAZ - INSPT - UTN - ISTIC

Andrea Rocca
cursosmoviles@gmail.com
ISTIC - Escuela de Comercio N° 22 - ACP Informática

Resumen

Esta idea de dictar talleres simultáneos fue organizada por las docentes Marisa E Conde, Andrea Rocca (Geniateka), y Cristina Velázquez (e-ducadores y Creabóticos), a las que luego se sumaron otras colegas Leonora Daniela Massa y Andrea Rocha (Haciendo robótica creativa), Susana Escobar (e-ducadores y Creabóticos) Mirta Gaspari (Club Code), Melisa Bay (Ohmbú) Nadia Mir (Nambot) y Nancy Morales (Geniateka).

Cada una de las docentes hace ya mucho tiempo que dictan talleres para niñas/niños, adolescentes y docentes por fuera de las instituciones que las albergan. En esta oportunidad se reunieron para ofrecer talleres temáticos gratuitos on line para celebrar el Día Internacional de la Niña y la Mujer en Ciencia y Tecnología. El objetivo de este evento está relacionado con que el abordaje a la programación y a la robótica educativa tenga un anclaje en la didáctica. La introducción a estos aprendizajes debe contemplar aspectos básicos de la electrónica y la física para que sobre ellos se pueda ir incorporando nuevos saberes y adquiriendo habilidades que estimulen a los aprendices a indagar, pensar creativamente soluciones a problemas que se encuentran en su entorno y mejorar el contexto y la vida de la sociedad que las/os rodea.

El factor de unión de todas las docentes involucradas en esta actividad es, indudablemente, la pasión por la docencia y por el uso significativo de la tecnología. Todas coinciden en que la escuela en el formato actual limita los aprendizajes y la forma de abordar todo lo que tenga que ver con el uso de la tecnología por diferentes razones, las más significativas son la falta de capacitación docente y la falta de recursos. Los cambios de gestión afectan los planes de inclusión de la tecnología por lo que suele suceder que el equipamiento se usa a un 30 % de su capacidad y debido al cambio veloz de tecnología e interfaces el material queda vetusto sin haber sido aprovechado.

Para la convocatoria a niñas y adolescentes se preparó una presentación realizada en Genial.ly. https://view.genial.ly/601634c0a4816d0d13b572f1/interactive-image-11defebdia-internacional. La oferta incluía talleres de ScratchJr., Scratch, Microbit, AppInventor y Arduino y Modelado en 3D.

Palabras clave: Programación, Pensamiento Computacional, Creatividad, Degustación.

Zamba: una plataforma de aprendizaje como soporte a conceptos computacionales

Gustavo del Dago
gustavo.deldago@unipe.edu.ar
Universidad Pedagógica Nacional

Virginia Brassesco
virginia.brassesco@unipe.edu.ar
Universidad Pedagógica Nacional

Maximiliano Urso
maximiliano.urso@unipe.edu.ar
Universidad Pedagógica Nacional

Resumen

En la Universidad Pedagógica Nacional, y en estrecha vinculación con la Fundación Sadosky, se está implementando un Profesorado en Informática con un enfoque didáctico innovador, alineado con los núcleos de aprendizajes prioritarios del Programa Nacional de Ciencia y Tecnología en la Escuela del MinCyT.

Se espera que al egresar el estudiantado pueda generar sus propios materiales, continuar investigando en estos temas y que el contenido continúe vigente cuando cambien las herramientas existentes para enseñar o programar. Los temas están centrados en los conceptos computacionales y no en las herramientas o lenguajes.

El eje principal del profesorado es la programación, que cuenta con seis materias, cuatro disciplinares y dos didácticas. El objetivo de la materia Programación I se centra en poder pensar estrategias simples centradas en la legibilidad de código, utilizando fuertemente la división en subtareas y modularización. Los desafíos proponen algoritmos de recorridos secuenciales.

Dentro de este conjunto de materias es fundamental contar con plataformas de aprendizaje y enseñanza apropiadas para cada nivel, personalizables y configurables.

En los cursos introductorios de programación suelen usarse plataformas por bloques para que la sintaxis no agregue complejidad y para poder impedir errores comunes como por ejemplo errores de tipos.

En esa búsqueda se desarrollo Zamba, una plataforma de programación por bloques, en español y personalizable, con entorno minimalista y sin instalación. Un requisito que se sumó para poder asegurar que el entorno funcione en cualquier computadora a distancia sin tutoría de instalación.

Esta herramienta está pensada además para permitir la transición a lenguajes de programación textuales, necesario para encarar Programación II. Este entorno fue pensado como soporte para estrategias didácticas, ejercicios y conceptos computacionales de lenguaje imperativo, objetivo de la materia; que permita la enseñanza de conceptos de variables, entrada y salida, con tipos de datos básicos (Cadenas de texto, numéricos y lógicos).

Usar herramientas propias, y mostrar que es posible desarrollar las propias plataformas creemos incentiva al estudiantado a hacer lo propio en un futuro inmediato. Teniendo en la mira una soberanía pedagógica y tecnológica para las aulas de Argentina.

Palabras clave: Programación por Bloques, Didáctica, Soberanía Tecnológica, Lenguaje Imperativo.

Actas

JADICC

Jornadas Argentinas de Didáctica de las Ciencias de la Computación, 2021

Fundación SADOSKY <Program.AR/> Ministerio de Ciencia, Tecnología e Innovación Argentina CAF