Serie
Cuadernos para la enseñanza

# Aprender a programar para integrar(nos)

Cecilia Martínez y María Emilia Echeveste

UEPC Unión de Educadores de la Provincia de Córdoba

ICIEC Instituto de Capacitación e Investigación de los Educadores de Córdoba

**Aprender a programar para integrar(nos)**

**Editores**
Gonzalo Gutierrez- Lucia Beltramino

**Autoras**
Cecilia Martínez y María Emilia Echeveste

**ISBN**

**Producción:**

**Diseño Gráfico y Diagramación:**
zetas.com.ar

**Impresión:**

**Revisión de contenido:**
Fernando Schapachnik Fundación Manuel Sadosky. Depto de Computación, FCEyN UBA. Nicolás Wolovick, Doctor en Computación. Profesor de FAMAF UNC. F. Matías Cuenca Acuña, Doctor en Computación. Arquitecto de Software de McAfee.

# Aprender a programar para integrar(nos)

**Cecilia Martínez* y María Emilia Echeveste****

* **M. Cecilia Martinez** es pedagoga y Dra. en Política Educativa. Es docente en la Universidad Nacional de Córdoba y trabaja en el Instituto de Humanidades de CONICET. Es tallerista de Cursos de Capacitación docente del ICIEC, UEPC. En los últimos 10 años se ha dedicado a diseñar e investigar formatos de formación docente en servicio en el área de las Ciencias. Más recientemente, ha focalizado su trabajo en la enseñanza de programación.

** **M. Emilia Echeveste** es psicóloga y está realizando su doctorado en Ciencias de la Educación como becaria de CONICET. Fue tallerista de Cursos de Capacitación docente del ICIEC, UEPC. Desde 2010 pertenece al equipo de investigación Ingreso a la Universidad y Relación con el conocimiento en el CEA-UNC y desde 2012 participa en proyectos de extensión e investigación referidos a la relación con el conocimiento en programación, tema específico de su doctorado.

# Introducción

Con esta publicación iniciamos desde el Área de Formación Docente del ICIEC-UEPC la serie "Cuadernos para la enseñanza", donde compartiremos propuestas pedagógicas surgidas de los interrogantes, dilemas, desafíos e invenciones que miles de docentes realizan cotidianamente en las escuelas, para que sus estudiantes aprendan. Dichas propuestas son sociabilizadas, analizadas y revisadas en los diferentes espacios de formación que ofrece el gremio, a través del Instituto de Capacitación e Investigación, desde hace más de 20 años.

Esta serie se constituye en otro de los dispositivos de construcción colectiva para acompañar el trabajo de enseñar de los/as docentes. El proyecto implica ir junto al otro/a y compartir sus preocupaciones, interrogantes, propuestas, alegrías y apuestas para que todos/as los/as niños/as y jóvenes puedan aprender.

A través de diferentes documentos que iremos publicando en el marco de esta serie buscamos articular las orientaciones generales para enseñar determinados temas y/o contenidos, con la socialización de propuestas y/o secuencias didácticas construidas por docentes que han asistido a los cursos ofrecidos por el ICIEC-UEPC.

Esta serie se constituye en otro de los dispositivos de construcción colectiva para acompañar el trabajo de enseñar de los/as docentes. El proyecto implica ir junto al otro/a y compartir sus preocupaciones, interrogantes, propuestas, alegrías y apuestas para que todos/as los/as niños/as y jóvenes puedan aprender.

En esta primera entrega **"Aprender a programar para integrar(nos)",** se realiza un recorrido histórico sobre la enseñanza de la computación en las escuelas, haciendo foco en el contexto actual. Se describen las características y potencialidades del pensamiento computacional y se presentan orientaciones para el trabajo en el aula.

¿Por qué empezar esta serie por la Programación? Porque promueve los proyectos colectivos, posibilita respetar los tiempos de aprendizaje de los/as estudiantes, habilita modos de pensar que implican un hacer y una creación singular. Programar permite integrar(nos), crear, inventar, acercar problemáticas cotidianas a la escuela, pensar posibles soluciones y potenciar e integrar los aprendizajes de otros campos de saberes escolares. Conocer el detrás de escena de los artefactos tecnológicos abre la posibilidad de contribuir desde la escuela, con el pasaje de ser consumidores/as a creadores/as de tecnologías.

A primera vista este nuevo saber que los/as docentes *tenemos* que aprender para enseñar podría pensase como una carga; sin embargo, son muchas las potencialidades que podemos aprovechar enseñando a programar. Por un lado, permite al sujeto que aprende pensarse ya no solo consumidor de un producto creado por otros/as, con intencionalidades implícitas y explicitas, sino también como ciudadano/a que puede compartir sus saberes, construyendo colectivamente, por ejemplo, tecnologías de uso libre. Por otra parte, aprender a pensar en términos computacionales, para lo cual cabe aclarar que no se necesita computadoras ni conexión a internet, moviliza lógicas de razonamiento que pueden implementarse desde el nivel inicial en adelante, por medio de juegos lógicos o el trabajo con problemas que requieren para su solución, el desarrollo de estrategias de comparación y seguimiento de procesos. Es decir, el trabajo cooperativo, el ensayo y el error, la sistematización de los avances logrados, la reflexión sobre los problemas y soluciones alcanzados.

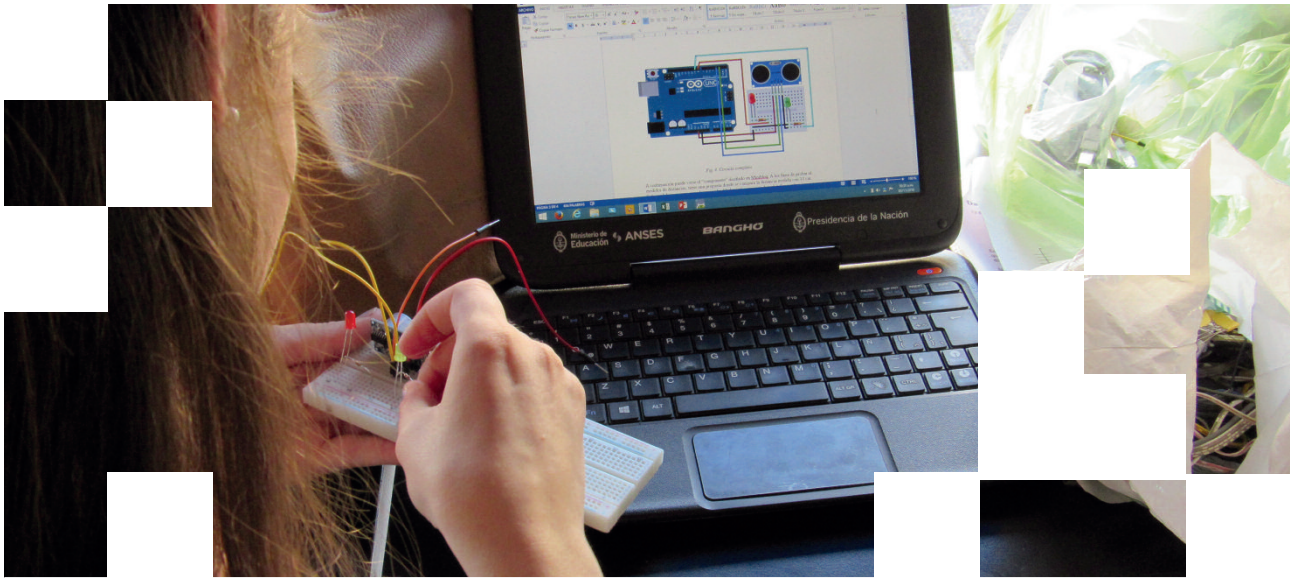
En un contexto donde acechan las lógicas individualistas y meritocráticas, donde el sujeto hace un uso aislado de las tecnologías, como docentes podemos "resistir" enseñando mediante apuestas colectivas de trabajo comprometidas con el desarrollo de modos de pensamientos complejos y críticos.

# ESCENA INTRODUCTORIA

## *Escuela Secundaria - Departamento Colón - 2016*

*Los/as estudiantes de sexto año de una escuela secundaria de las sierras chicas hoy tienen una jornada especial. Visitarán a sus compañeros/as de sexto grado de otra escuela primaria de la misma localidad para trabajar juntos en un proyecto de programación de computadoras para realizar videojuegos y animaciones.*

*Entre ellos está Pedro. Según dicen sus docentes, Pedro se muestra desinteresado de la mayoría de las propuestas de enseñanza de la escuela. No sabemos si tiene problemas de aprendizaje, si "no entiende"; o si, simplemente, es una cuestión de actitud y "no le interesa nada". Pero en este proyecto Pedro ha trabajado sin interrupciones en la clase y ha realizado un videojuego utilizando varias de las funciones propuestas y algunas más que descubrió por su cuenta. Lo mismo ocurrió con María, sus maestras comentan que si bien siempre le pone empeño a la tarea, tiene un muy bajo rendimiento. A María "le cuesta" mucho todo, nos dicen las maestras. Sin embargo, con este proyecto, ha logrado poner en juego conceptos complejos como funciones de matemáticas que antes no comprendía.*



*Los/as docentes de secundaria y primaria se conocieron en un curso de UEPC y como no sabían muy bien de qué se trataba la programación, decidieron trabajar juntos/as en la planificación de la clase y en la puesta en práctica con sus estudiantes. Se les ocurrió hacer esta experiencia compartida porque advierten que es necesario realizar más actividades de articulación entre la primaria y la secundaria. En su reflexión escrita comentaron que la clase superó ampliamente sus expectativas. En primer lugar, porque vieron un gran entusiasmo de sus estudiantes. Los/as estudiantes se agruparon entre ellos/as indistintamente del nivel educativo, no querían salir al recreo, se quedaron trabajando dos módulos seguidos y después sí, decidieron cortar un rato. En segundo lugar, porque los/as estudiantes lograron hacer producciones mucho más complejas de lo que propusieron sus docentes; investigaron dentro de la plataforma de programación y descubrieron por su cuenta nuevas funciones y algoritmos para realizar su proyecto. Pudieron hacer que la computadora haga tareas que no fueron anticipadas por ellos/as. Los/as docentes no tuvieron que dar una clase larga, sino que fueron grupo por grupo, trabajando sobre sus proyectos. En tercer lugar, porque pudieron desarrollar una breve experiencia de articulación entre la primaria y el secundario. Fue una experiencia positiva que les ha permitido generar un nuevo horizonte sobre lo que los/as estudiantes pueden hacer, y sobre lo que los/as docentes pueden generar juntos/as.*

Esta escena ocurrió en un curso de "Enseñanza de la Programación en la Escuela" en el marco del Instituto de Capacitación e Investigación de los Educadores de Córdoba (ICIEC)-UEPC realizado en 10 sedes entre 2015 y 2016. Si bien la misma corresponde a una sede e involucró a dos escuelas, situaciones como estas fueron regulares en nuestros cursos de enseñanza de la Programación.

En esta publicación trabajaremos fragmentos recuperados de las distintas experiencias surgidas en los cursos de formación docente en Programación en UEPC y en la Universidad Nacional de Córdoba. Debido a que el curso incluía en su diseño horas de práctica en las aulas con los/as estudiantes, describimos también situaciones de enseñanza y aprendizaje sobre dicha temática en las escuelas.

Por otra parte, se analizarán algunos modos de enseñar programación y su potencialidad para articular con diferentes asignaturas, en una relación con el saber poco tradicional para los/as estudiantes. Las experiencias que mostraremos, han permitido observar movimientos en los/as alumnos/as en relación a sus modos de razonamiento usualmente no considerados o invisibilizados por la escuela tradicional.

En las páginas que siguen comenzamos a describir cómo históricamente se ha introducido la computación en la escuela, en qué consiste programar y por qué es relevante su enseñanza. Esto nos llevará describir qué tipos de pensamientos permite desarrollar la programación y analizaremos con detalle qué pasó en las experiencias singulares, de qué manera docentes y estudiantes lograron trabajar juntos y cuáles son las implicancias de este tipo de proyectos en las escuelas.

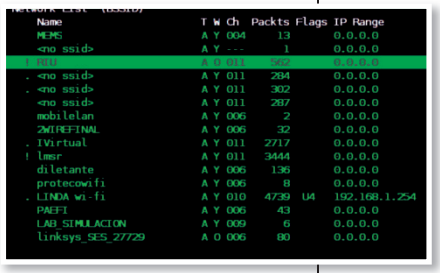
# Tradiciones de enseñanza

Repaso histórico de los diferentes objetivos con los cuales ingresó la computación a las escuelas.

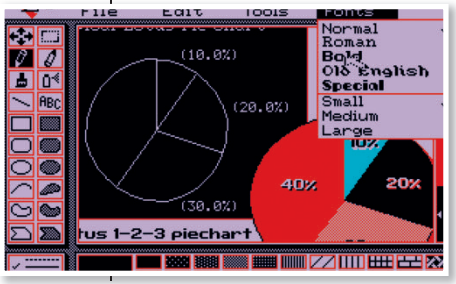
## ESCENA DE UNA CAPACITACIÓN DOCENTE

*En una mañana de sábado, como actividad para comenzar la clase, les proponemos a los/as docentes que han venido a realizar nuestro curso, tomarnos cinco minutos contados con reloj para cerrar los ojos y transportarnos a nuestra clase de computación o tecnología o informática, cuando éramos estudiantes de primaria o secundaria. Tratamos de recordar los detalles. Nos trasladamos a una clase en particular que por diversos motivos haya quedado grabada en nuestra memoria. ¿Cuál fue la tarea que ofreció el/la docente? ¿Qué teníamos que hacer nosotros, alumnos/as de primaria o secundaria? ¿Cómo resolvimos esa tarea? ¿Qué disposiciones pusimos en juego? Teniendo en cuenta el tipo de tarea que teníamos que resolver, ¿qué nociones/ideas construimos sobre lo que era la computación como disciplina?*

*Luego de una reflexión silenciosa, los/as docentes que participan del curso reunidos en grupos de tres comienzan a compartir sus memorias, eligen uno de los recuerdos que los integrantes trajeron al grupo y lo comparten con el resto de la clase. Un profe que ubica su experiencia a fines de los ochenta, principios de los noventa recuerda que programaban el Logo o Basic, que la computadora no tenía programas "como los que tienen ahora", sino que "la compu no hacía nada" que el usuario no programara. Que fueron estas experiencias las que les enseñaron cómo operaba una computadora por dentro. Les permitieron entender los mecanismos básicos de cómputo. Otros docentes ubican su clase a mediados y fines de los noventa. Se ven usando procesador de texto y planilla de cálculo de una firma monopólica, y otros programas como el Paint. No sé si será la diferencia en el tiempo lo que hace que veamos las experiencias*




▶ **Basic.**




▶ **Paint.**

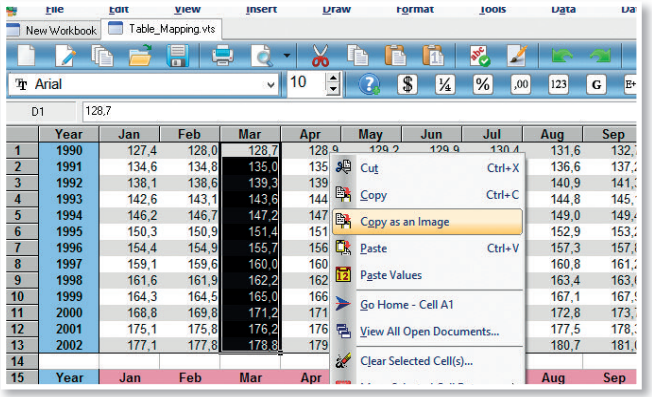
*más lejanas como las más románticas. Pero advertimos que quienes atravesaron su escolaridad a fines de los ochenta y principios de los noventa hablan con más entusiasmo de las clases de computación que aquellos/as más jóvenes que estuvieron en la escuela a fines de los noventa y principios del 2000. Estos/as últimos/as recuerdan las clases de computación como aburridas y poco desafiantes. Mientras que los/as más veteranos/as cuentan con emoción cómo tenían que programar e instalar sistemas operativos porque nada estaba pre armado.*

En este fragmento vemos que en las diferentes décadas ocurrieron situaciones de enseñanza de la computación muy dispares. Los relatos de los/as docentes, experiencias de primera mano vividas como alumnos/as, fueron construyendo modos de enseñar la disciplina. En esta sección desarrollaremos los diferentes objetivos con los cuales ingresó la computación a las escuelas, para poder así comprender y hacer conscientes muchas de las decisiones que tomamos cuando enseñamos informática. De esta manera reconocemos que las experiencias, en este caso desde sus relatos de aprendizaje, son de carácter situado y, por lo tanto, no responden a un atributo individual.

El ingreso a la escuela de las tecnologías desarrolladas para otros ámbitos de la vida social no es nuevo. El cine, la fotocopia, las filminas, la pizarra, entre otros dispositivos se fueron instalando gradualmente en la escuela a lo largo de la historia de diferentes maneras. A partir del desarrollo de la PC (del inglés *Personal Computer*), las escuelas comenzaron a ofrecer formación en computación enfatizando diferentes aspectos según las distintas décadas (Levis, 2006):

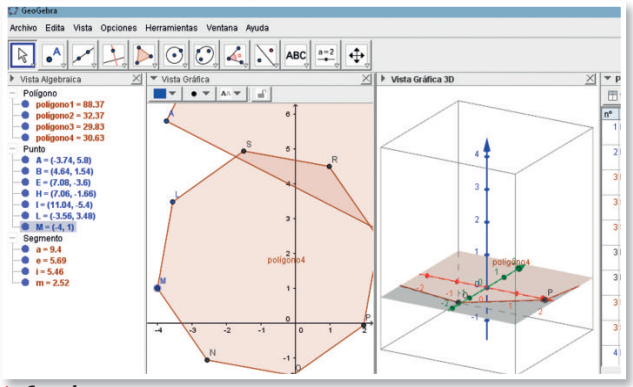
▶ En los ochenta predomina un enfoque técnico que aborda *hardware* y comandos básicos necesarios para que la computadora pudiera arrancar y procesar información. Eran los tiempos de encender la computadora con un diskette que tuviera el sistema operativo y el *software*. Se dio lugar a un enfoque constructivista-construccionista de la enseñanza de la programación, casi exclusivamente con LOGO (Papert, 1980) acompañado del enfoque técnico de operación de hardware.

▶ Office-Excel.




▶ Los noventa se centran en la formación del usuario a través de programas de oficina desarrollados por grandes monopolios (Office, por ejemplos, uno de los más difundidos hasta la actualidad).

Predominan el uso de procesadores de textos y planillas de cálculo de *software* privativo, es decir, sin acceso libre ni al software ni al código. Predomina la marca Windows que se hace masiva a mediados de esta década con su plataforma de ventanas y el uso de mouse. Específicamente en Argentina, el Estado Nacional estableció un acuerdo de cooperación con Microsoft e IBM delegándole los recursos informáticos para las escuelas. Estas empresas privadas se




▶ ***Geogebra.***

encargan de la formación de muchos/as docentes de enseñanza básica y media de todo el país con una fuerte concepción operativa e instrumental de la tecnología en detrimento de los conceptos propios de la disciplina Ciencias de la Computación, brindando así la ilusión de estar formando a los/as jóvenes y niños/as en esa área (Busaniche, 2006). Esto fue denominado por Levis (2007) como "enfoque utilitario".

▶ La primera década del milenio busca integrar a las TIC para potenciar los contenidos escolares con programas desarrollados para propósitos didácticos tales como los simuladores, Geogebra, entre tantos otros. Un enfoque de trabajo que Levis (2007) denomina "integrador", donde se potencian aprendizajes de otras disciplinas utilizando la computadora, pero sin lograr abordar la especificidad de los conceptos propios de la computación como disciplina. También ingresan plataformas *on line* y aparece con más protagonismo el *software* libre, es decir programas informáticos accesibles a todos. Algunos de estos paquetes son de código abierto, es decir que, además de gratuitos, tienen disponible el código con el cual se escribieron los programas para que puedan ser modificados por los usuarios.

Cada una de estas etapas implicó el desarrollo de diferentes saberes y habilidades relacionados con el uso de la computación. El paradigma técnico enfatizó la memorización de elementos de *hardware* e instrucciones o comandos para operar funciones básicas. La formación del usuario de planillas de cálculos y procesadores de texto se centró casi exclusivamente en habilidades de uso y manejo del *software*. No se advierte en esta etapa una preocupación por ligar estos saberes y habilidades de la informática con los propósitos más amplios de la escuela, como tampoco una reflexión o recuperación del sentido pedagógico, en tanto formativo, del aprendizaje de la informática.

Oficialmente desde los Ministerios de Educación predominó un discurso pedagógico tecnocrático del uso de las computadoras en la escuela (Gutierrez, Beltramino, Viano, 2015), donde el currículum era pensado por grandes empresas monopólicas de *software* y el/la docente era aplicador de este mandato de una manera fragmentada del proyecto propiamente educativo que tenía la escuela. Sin embargo, convivieron en esta etapa grupos y movimientos que disputaban los sentidos de la máquina en la escuela.

A nivel mundial, reconocemos el trabajo de Seymour Papert (1980), quien piensa la introducción de la programación en la escuela para promover el pensamiento lógico y matemático y habilidades de construcción, creación de proyectos, trabajo en equipo, expresión de los/as niños/as y jóvenes a


Seymour Papert and Paulo Freire Debate Technology and the Future of Schools

**https://www.youtube.com/watch?v=4V-0KfBdWao**

▶ ***Este video radica en el simbolismo de pensar la introducción de la tecnología de manera crítica para profundizar los aprendizajes de los estudiantes.***

través de los artefactos tecnológicos que desarrollaran la comunicación entre ellos/as y las generaciones adultas, e inclusión educativa de jóvenes y niños/as de diferentes puntos del planeta que no pudieran ir a la escuela. En efecto, Papert sería uno de los generadores del proyecto "One Laptop per child" (Una computadora por alumno/a, conocido aquí como Modelo 1 a 1) que se originó en el MIT, EEUU; para comunidades del África a fines de los noventa para ofrecer acceso al conocimiento a través de las tecnologías.

En Argentina, a partir de la década de los ochenta alrededor del 50 % de las escuelas privadas comenzaron a ofrecer cursos extracurriculares de computación y luego, en algunos casos, los incorporaron a sus planes de estudio. Los contenidos incluían rudimentos de programación en BASIC, Logo y Pascal. Una encuesta realizada a personas que nacieron entre 1945 y 1995 y que estudian o trabajan en áreas relacionadas con la informática muestra que al menos 35 % de ellos eligieron dedicarse al área de computación por el acceso temprano a la programación (durante la primaria o secundaria) (Cotik y Monteverde, 2015).

A nivel local, durante los noventa, en Córdoba, la UEPC realizó varias acciones que disputaban el sentido tecnocrático. En particular, ofreciendo cursos de formación docente para reflexionar sobre el sentido pedagógico de la introducción de la tecnología e integración con los objetivos generales de la educación (Gutierrez, Beltramino y Viano, 2015).

A principios del 2000 hubo esfuerzos importantes por integrar la computadora a la escuela para promover diferentes aprendizajes. Sin embargo, y a pesar de las diferencias cualitativas, estos enfoques no abordaron la computación como objeto de estudio.

Diversos motivos impulsaron la introducción de la computadora en la escuela. El historiador de la educación Larry Cuban (1993) reconoce tres preocupaciones que fueron pilares para introducir la computadora en la escuela. Revisar estos argumentos contribuye a comprender parte de nuestras prácticas de enseñanza. A continuación, describimos cuáles son:

**1) Preparación de los/as estudiantes al mundo del trabajo.** Dotar a la escuela con equipamiento tecnológico que esté a la altura de lo que se requiere usar en el mundo del trabajo ha sido siempre una preocupación de quienes ven la escuela como la maquinaria social encargada de la preparación de la mano de obra calificada. Desde esta visión de la educación, para que los/as egresados/as sean competitivos en el mercado laboral, deben manejar máquinas, productos con códigos de barras, contestadores telefónicos, etc. Así, la introducción de la tecnología en la escuela busca familiarizar a los/as estudiantes con los dispositivos que se encuentran en el mundo laboral.

**2) La preocupación para que la enseñanza sea eficiente.** Desde la conformación de la escuela moderna ha estado presente la preocupación de que todos los contenidos propuestos se transmitan en el menor tiempo posible, bajo el menor costo y a toda la población escolar. A esta inquietud responde, por ejemplo, la agrupación homogénea por edad, el currículum unificado y la enseñanza sincrónica. Siguiendo a Cuban (1993), la introducción del cine, la radio y las diapositivas en las escuelas trajo consigo la promesa de que mejorarían el aprendizaje, la motivación, asegurarían la transmisión de la cultura oficial y permitirían la supervisión, mejorando la efectividad de la escuela. Desde esta preocupación se dio lugar a un gran desarrollo de programas informáticos educativos que, bajo el paradigma del conductismo, ofrecían lecciones sobre diferentes temas a través de la computadora. Generalmente, en estos programas el/la alumno/a colocaba un *input* y la computadora le respondía con un correcto o incorrecto sin demasiada orientación sobre el trabajo. Este tipo de propuestas se reconoció dentro del enfoque "eficientista de la educación" y que algunos más críticos han denominado "enfoque donde la computadora programa al niño/a". El aprendizaje se caracterizaba por ser memorístico y repetitivo y los programas eran de ejerci-

cios de múltiple opción, respuesta correcta o requerían completar valores, etc. Esta concepción dominó el panorama educativo del uso de la computadora en el aula desde los sesenta a los noventa.

**3) Promoción del aprendizaje por construcción.** Grupos de académicos, educadores, fundaciones e inclusive funcionarios públicos tienen una visión que Cuban (1993) caracteriza como "neoprogresista" de la educación, basado en las corrientes de la Escuela Nueva y la Psicología del desarrollo para promover un cambio en el formato escolar. Este cambio consiste en que los/as estudiantes construyan sus saberes a partir de explorar el mundo y resolver problemas. En esta posición, el conocimiento debe ser significativo para los/as alumnos/as. En ese sentido, ven a la computación como una herramienta que promueve el pensamiento. A diferencia de otras tecnologías, la computación es considerara como una "tecnología disruptiva" porque promueve el crecimiento de las capacidades cognitivas, es decir, habilitan el acceso a problemas y soluciones no accesibles de otro modo. El cómputo y la lecto-escritura, por ejemplo, son ejemplos de tecnologías que amplían nuestra capacidad de conocer y entender. El papel usado para la escritura amplió significativamente el acceso a la palabra, y junto a la imprenta, en tanto automatización de la copia del texto escrito, permitieron la primera ola de masificación del conocimiento escrito aumentando la capacidad de lectura y escritura (Simari, 2011). Del mismo modo, la automatización de cómputos que permitieron las primeras computadoras, contribuyeron al desarrollo de la física, tecnología, etc. Un referente en el área de la computación que ven a la informática como una tecnología que permite desarrollar la cognición es **Seymour Papert**, discípulo directo de Jean Piaget, matemático y desarrollador del sistema LOGO, un lenguaje que permitía enseñar conceptos de matemáticas a través de la programación. A esta corriente se pliegan también las teorías críticas de la educación, que buscan la construcción del saber significativo para los/as estudiantes para su emancipación.

Estas tres preocupaciones que proponen un modo diferente de uso de la computadora en la escuela, conviven en los discursos de Introducción de la tecnología en la escuela desde los ochenta hasta la actualidad. Es decir, ninguno ha logrado reemplazar o superar a otro. La preparación para el mundo del trabajo, la organización eficiente de la escuela y la formación en resolución de problemas son desafíos que la escuela siempre tuvo en la formación de los futuros ciudadanos.

Desde el 2005, sin embargo, han cobrado vigencia los enfoques "lingüísticos", que proponen enseñar lenguajes y conceptos de computación que permitan no solo entender cómo funciona una computadora, sino crear nuevas tecnologías a partir del desarrollo de programas. Estos enfoques centran su mirada en la promoción de construcción de saberes, resolución de problemas y desarrollo del pensamiento analítico y crítico. Diferentes sectores se han unido para fomentar que, además de usar la computadora como herramienta para el aprendizaje de distintas disciplinas, la misma sea "objeto de estudio", y que las Ciencias de la Computación sea quien aborde su estudio.

Las organizaciones relacionadas con el *software* libre tales como la Fundación Vía Libre son los principales voceros de incluir la enseñanza de la programación en computación en la escuela. Quienes apoyan y promueven el uso de *software* libre sostienen que, para poder acceder a productos culturales disponibles para todos/as, es necesario conocer y entender fundamentos de computación. El problema es que al desconocer conceptos básicos, no podemos entender el software que ya está publicado y liberado y que podría ser usado y modificado por todo aquel que tenga una computadora.

Si hacemos una analogía con una receta de una torta de chocolate comprenderemos mejor de qué se trata el *software* libre.

Un programa de computación y una receta de cocina incluyen instrucciones para que la computadora o una persona las ejecute. Si lo pensamos como ingredientes, por un lado, en el escenario del *software* privativo, tendríamos una premezcla de torta donde no sabemos qué ingredientes contiene realmente, o si sabemos cuáles son no podríamos tener acceso. En el otro, desde el *software* libre, buscaríamos la receta, y como sabemos leer y comprender lo que leemos, interpretamos los ingredientes. Observaremos que tiene azúcar, y como uno de los comensales es diabético, tenemos la opción de cambiarla por otro endulzante.

Quienes defendemos la enseñanza de la programación en computación en las escuelas sostenemos que todos los/as ciudadanos/as tienen derecho a entender mínimamente cómo están desarrollados los productos tecnológicos que consumen para usarlos críticamente, y potencialmente crear nuevas tecnologías a partir de ellos.


Vimos en esta sección que desde hace más de treinta años la computadora ha ingresado a la escuela con diferentes objetivos. Sin embargo, es en estos últimos años que se habla de pasar de ser consumidores o usuarios de programas, a creadores de tecnología. En la sección que sigue analizaremos los objetivos de la enseñanza de la computación hoy.

Mientras terminábamos de editar esta publicación, el 18 de Julio salió una noticia replicada masivamente en los diarios sobre un robot de seguridad que se cae a una pileta. En la mayoría de los periódicos la titulación hizo referencia a una personalidad "suicida" del robot. Los medios masivos no explicaron por qué el robot se cayó a la pileta sino que replicaban los chistes que se hicieron del hecho. Afortunadamente un lector realizó el siguiente comentario en una publicación *on line*:

"No es complicado, se ve que está lloviendo y el piso está húmedo. Los sensores que detectan los desniveles son infrarrojos y utilizan la reflexión en el suelo para determinar si hay piso o vacío. Al estar mojado, la reflectividad aumenta y en lugar de detectar el escalón, siguió recibiendo la reflexión, solo que se estaba reflejando en el escalón de abajo... y siguió de largo lo suficiente para desbalancearse y caer."

Es preocupante que se reproduzcan mitos e ideas falaces sobre la tecnología en medios de comunicación masiva alimentando la noción sobre la imposibilidad de aprender de ese objeto, como si la computación fuera inalcanzable para la mayoría. Si bien del modo en que está presentada la noticia, no tienen relevancia para nosotros/as; la tecnología con la cual está construida sí se relaciona con varios desafíos de nuestro entorno. Algunos sensores son usados para la medicina y los satélites meteorológicos, por ejemplo. Mucha de la tecnología utilizada en la robótica se puede aplicar para hacer más eficiente el uso de energías renovables. También podría haber sido relevante analizar que todas las computadoras pueden fallar, como en el caso del robot que se ahoga. Esta información sería necesaria para tomar posición frente al voto electrónico que se propone como una herramienta que podría poner en juego la democracia. Con la alfabetización en programación básica, la escuela puede hacer una diferencia en la formación del ciudadano crítico y reflexivo.


Robot de seguridad se 'suicida'

19 Julio, 2017 |

# Situación de la enseñanza de la programación hoy

Hacemos foco en la posibilidad de enseñar programación en el contexto de la actual escuela

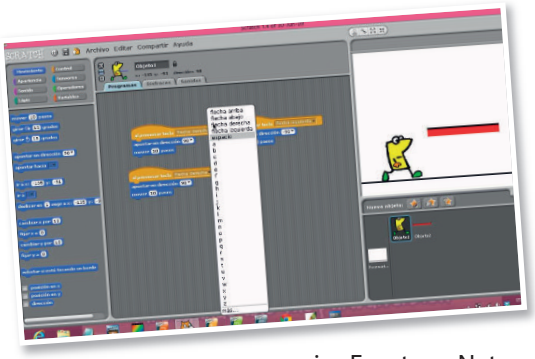
Como mencionamos anteriormente, en la última década hubo un importante desarrollo y capacitación de los/as docentes para que las escuelas dejaran de usar solamente programas de oficina y pudieran integrar las TIC para potenciar diferentes áreas del saber. La apropiación que hacemos de las tecnologías en la escuela, y de la computación en particular, está en constante desarrollo. Diversos estudios muestran que año tras años los/as maestros/as y profesores/as van incorporando la computadora de manera cada vez más significativa (Caldeiro, Casablancas, Odetti, 2016; Lago Martínez, 2015). Como resultado del programa "Conectar Igualdad", Caldeiro (2016) y sus colegas encontraron que los/as docentes usan la computadora para ofrecer propuestas de enseñanza donde se incluyen a las TIC y si la propuesta lo requiere se realizan usos de programas sofisticados para desarrollar contenidos interactivos.

Como caso paradigmático tomamos el relevamiento de una escuela técnica, la cual recibió en 2012 las netbooks del programa Conectar Igualdad, que si bien el lanzamiento fue en 2010, su distribución a todo el país fue progresiva. En esta escuela, estas computadoras permanecieron en la biblioteca para ser usadas solo en actividades puntuales con la previa solicitud formal y planificada de los/as docentes en Dirección. Hacia el segundo año la escuela decidió distribuir las netbooks entre sus alumnos/as (tal como especificaba el programa) y al ver el uso intensivo que realizaban los/as estudiantes, actualmente, ya en el tercer año, los/as profesores/as de espacios técnicos decidieron comenzar talleres de robótica con placas de *hardware* abierto (placas Arduino). Historias como estas se repiten en numerosas escuelas. Similares acontecimientos fueron documentados en escuelas primarias que recibieron las computadoras de Primaria Digital desde 2015.

Es decir, desde nuestra experiencia trabajando con docentes tanto en la Universidad Nacional como con el ICIEC de UEPC, observamos una constante apropiación por parte de los/as docentes, estudiantes y familias de las computadoras. Este hecho nos sitúa en un momento de posibilidades diferentes previas a los programas de distribución de equipamiento y formación docente masivos en TIC.

En los últimos años, diversos grupos en universidades de todo el mundo han desarrollado plataformas didácticas para enseñar a programar. Estas, por lo general, recuperan un aspecto lúdico, son ricas visualmente y permiten a los/as estudiantes realizar ejercicios o animaciones y ver sus resultados rápidamente (por caso, citamos a Scratch, pero también hay otras de desarrollo local como Pilas Bloques, Gobstone, Mumuki, Chatbot y UNC++ Duino). El diseño de estas plataformas se vio enriquecido con


▶ *Placa Arduino.*

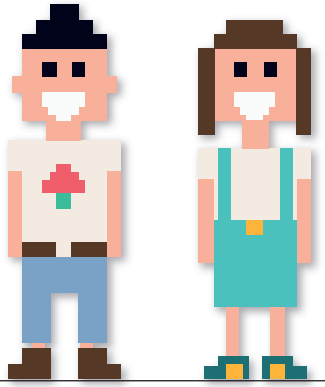


▶ *Scratch.*

el concepto de intuición para aprender computación, que investigaron Miller en los ochenta y Pane en el 2000 (Guzdial, 2008). Estos estudios mostraron que la mayoría de adultos y niños/as cuando se ponían en situación de resolver un problema lógico matemático (tales como los problemas que resuelve la Informática), usaban un lenguaje natural, es decir, el lenguaje coloquial, que incluía algunas nociones de lógica como saberes previos. Por ejemplo, la noción de condicional se puede usar para indicar una condición entre una proposición y una acción tal como "si se da la condición A, entonces se hace B". Este tipo de estudios ha informado el desarrollo de plataformas que invitan a "aprender haciendo" desde las intuiciones o saberes previos. Estas herramientas sirven para elaborar un proyecto específico tal como programar un videojuego, una animación, un robot que responde chats de manera automática, el recorrido de un robot en un laberinto o camino, o la resolución de un juego. Las funciones que tiene la plataforma se ofrecen en ventanas visibles al usuario y en su mayoría con un lenguaje en formato de "bloques de arrastre", que ya tiene previamente programada la función que realizará. Los/as estudiantes llevan a cabo combinaciones de bloques que al ejecutarse muestran una secuencia de líneas de código de un programa. Los/as alumnos/as ejecutan inmediatamente funciones observando errores y aciertos. A partir de ver cómo trabaja su programa, hacen nuevas búsquedas y combinaciones de bloques disponibles para que el programa realice otras acciones que ellos están pensando.

A partir de informes que emitieron varios sectores preocupados por la enseñanza de la Programación en las escuelas, muchos países han avanzado en acordar y definir qué conceptos de computación son necesarios enseñar y en qué secuencia. Si bien, a nivel nacional, en nuestro país no hay un currículum oficial, sí hay documentos emitidos por el Iniciativa Program.AR de la Fundación Sadosky y el CUCEN (Consejo Universitario de Ciencias Exactas y Naturales).

Vemos entonces que hay tres situaciones claves que nos permiten pensar en la posibilidad de enseñar Programación en el contexto de la escuela actual: 1) la creciente apropiación que hacen docentes y estudiantes de la computadora y las tecnologías en general en el ámbito de la escuela; 2) los avances en el desarrollo de las plataformas didácticas para enseñar a programar; 3) los desarrollos curriculares. A estos desarrollos, se suma la significativa reflexión sobre la noción de "pensamiento computacional" que desde 2006 se publica en revistas científicas, de divulgación y paneles de expertos en todo el mundo.



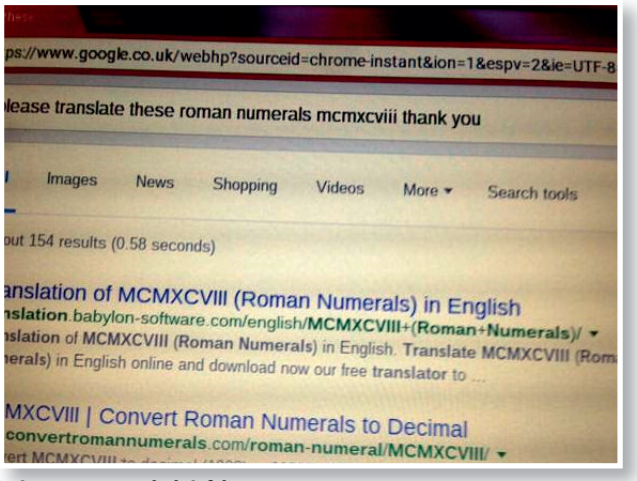
# El pensamiento computacional

A qué nos referimos cuando decimos programar. Singularidades y potencialidades del pensamiento computacional.

El pensamiento computacional requiere entender las capacidades y limitaciones de una computadora, y ser capaz de expresar un problema de forma tal que una computadora lo pueda resolver. Veamos un ejemplo: esta es una captura de pantalla tomada por un nieto del equipo de su abuela en Inglaterra que se hizo viral.

La captura de pantalla muestra las palabras que la abuela utiliza para buscar en Google: "Por favor, tradúzcame estos números romanos mcmxcviii gracias". Su nieto John, le preguntó a la abuela por qué usó tan buenos modales para realizar la búsqueda en Google. La respuesta de la abuela fue genuina: ella pensaba que una persona en las oficinas de Google iba a atender más rápido su pedido si ella lo solicitaba amablemente. ¿Qué es lo que diferencia el pensamiento de la abuela con el del nieto? Simari diría que la diferencia está en el "pensamiento computacional", es decir, la abuela no vislumbra la posibilidad de que una computadora realice una búsqueda con determinados algoritmos y ofrezca una respuesta automática. Por el contrario, su nieto sí sabe que una máquina puede realizar esa búsqueda en forma automatizada. El pensamiento computacional consiste, en términos generales, en entender qué problemas pueden ser resueltos por una máquina y cómo ordenar estos problemas para que la máquina los resuelva.

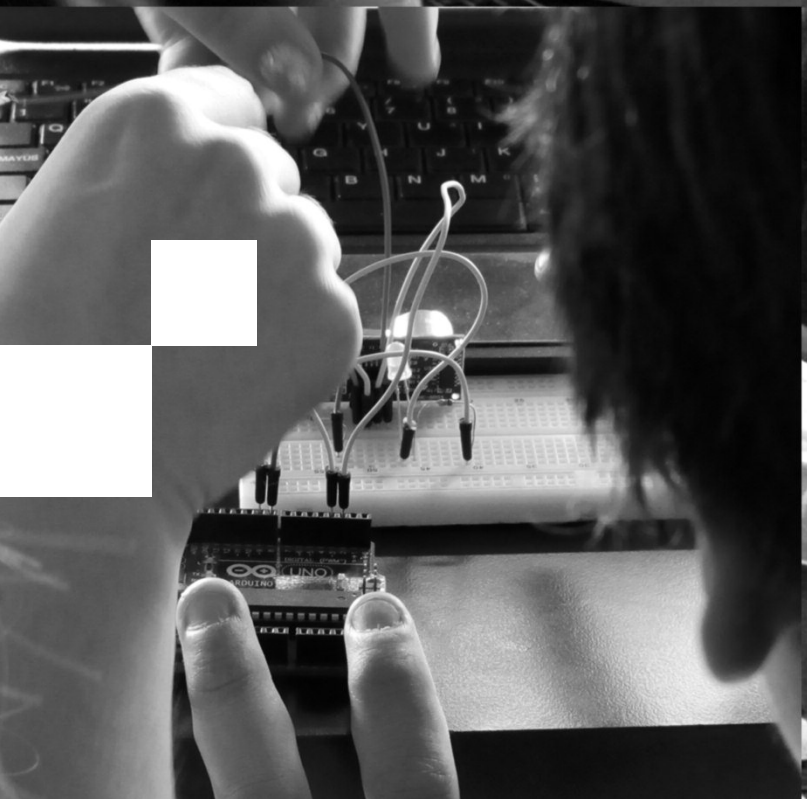
Siguiendo a Simari (2011), la Informática cambia la forma de representar el mundo porque permite, entre otras cosas, realizar simulaciones, modelizaciones y desarrollar la virtualidad. Estas representaciones posibilitan crear otras realidades dentro de las restricciones de una máquina. Esta




▶ *Imagen recuperada de Infobae.*

posibilidad de crear otros mundos dentro de las restricciones de un equipo, de entender a nivel intuitivo las capacidades y limitaciones de la computación para pensar soluciones computacionales factibles por agentes que procesan información, es parte de lo que Simari denomina pensamiento computacional.

Si bien, las primeras ideas sobre pensamiento computacional pueden rastrearse en los años sesenta, decantan en el trabajo empírico de Seymour Papert (1980) en las décadas de los ochenta y noventa. Para Papert el pensamiento computacional, en el marco de la escuela, consiste en una serie de procedimientos que los/as niños/as ponen en juego para programar una computadora. Pensemos una actividad sencilla que requiere de las principales acciones in-



volucradas en el pensamiento computacional tal como identificar la ruta más eficiente para que el transporte escolar haga su recorrido tratando de no repetir calles y en el menor tiempo posible. Probablemente lo que haríamos, en primer lugar, sería identificar y hacer una lista de los/as niños/as que viajan en el transporte. Luego, los agruparíamos por zonas, barrios y calles. Seguramente podríamos focalizarnos en resolver el trayecto de una zona por vez. Quizás podamos elaborar una secuencia del trayecto con algunas reglas tales como: no incorporar en el recorrido a niños/as por fuera de las zonas frecuentes. Bien, todas estas operaciones analíticas, son parte del pensamiento computacional.

Papert identifica cuatro dimensiones del pensamiento computacional que podemos reconocer en el ejemplo anterior:

▶ **Descomposición de problemas:** tomar un problema y dividirlo en subproblemas para que la máquina pueda resolverlo. En el ejemplo anterior, primero definir en qué orden conviene recorrer los barrios, luego, dentro de cada barrio, definir el recorrido que sea mejor para los/as niños/as de ese barrio. Cada una de estas acciones contribuyen al problema más amplio que es diagramar la ruta más eficiente del transporte escolar.

▶ **Reconocimiento de patrones:** identificar aspectos similares entre los problemas para que las soluciones se apliquen a varios casos. En el ejemplo que venimos trabajando, la trayectoria marcada para un barrio puede transferirse a otro barrio.

"... el pensamiento computacional incluye el pensamiento crítico, porque implica reconocer, entender y resolver un problema, pero lo hace con la colaboración de computadoras".

▶ ***Programación con software Alice.***

▶ **Abstracción:** capacidad de focalizar en la información importante y segregar aquella que no es relevante para la solución del problema, es decir, los detalles. La abstracción también consiste en capturar características, patrones o acciones similares en una categoría que permita representarlas a todas (Lee, 2011). Seguramente, en el ejemplo del transporte prescindiremos del nombre y grado de los/as alumnos/as, pero no así del nombre de las calles.

▶ **Algoritmos:** desarrollo de secuencia o reglas para resolver el problema tal como elaborar el trayecto siguiendo algunas reglas mínimas.

En esta última década, diferentes referentes sostienen que el pensamiento computacional requiere de resolución de problemas, diseño de sistemas y desarrollo de abstracciones que puedan ser representadas por el mundo computacional, es decir, pensar cómo haría un computólogo ante tal situación (Simari, 2011; Wing, 2006). Jeannette Wing, lo describe como un pensamiento que proporciona habilidades y competencias intelectuales que constituyen una forma de pensar que tiene características propias y diferentes a las de otras ciencias, como ser: la descomposición en subproblemas, abstracción de casos particulares, procesos de diseño, implementación y prueba de lógicas algorítmicas, para nombrar las más significativas. Para Simari (2011) el pensamiento computacional incluye el pensamiento crítico, porque implica reconocer, entender y resolver un problema, pero lo hace con la colaboración de computadoras.

En general, hay consenso en torno a seis grandes ideas del pensamiento computacional.

**1.** Hacer computación es una actividad creativa.
**2.** La abstracción permite reducir los detalles para centrarse en la información relevante para resolver un problema.
**3.** Los datos facilitan el desarrollo de saberes en diferentes disciplinas. Los conocimientos computacionales permiten el desarrollo de las ciencias, las tecnologías, capacidades de visualización de fenómenos que no serían observables de otro modo.
**4.** Los algoritmos son herramientas para expresar soluciones a problemas que se resuelven con una computadora.
**5.** Los artefactos digitales, los sistemas y las redes que los conectan, promueven enfoques computacionales para resolver problemas.
**6.** La computación permite innovaciones en otros campos, incluyendo Ciencias Exactas, Ciencias Naturales, Humanidades, Artes, Medicina, Ingeniería, etc.

Respecto de la innovación en otras áreas, por ejemplo, en el área de la salud, la computación ha permitido hacer diagnósticos por imágenes que permiten ver el funcionamiento de órganos y ubicar tumores con precisión. Han sido programas de computadoras los que han permitido encontrar compatibilidad entre ADN de bancos de datos genéticos para asignar filiación entre personas. Este avance ha permitido identificar a nietos ilegalmente apropiados en Argentina durante la última dictadura militar. Tanto la música como el cine se han beneficiado de los avances de la computación para mejorar la experiencia estética.

Advertimos que estas seis grandes ideas sobre el alcance del pensamiento computacional se ligan íntimamente con los objetivos de la educación general de nivel primario y secundario. Resolver problemas, representar información a través de diferentes lenguajes, producir enunciados y diferentes tipos de textos, encontrar regularidades, descomponer un problema, etcétera, son aprendizajes básicos que se espera desarrollar tanto en la educación primaria y secundaria y donde el aprendizaje de la programación puede hacer un aporte.

Las diferentes conceptualizaciones y estudios realizados en torno a la noción de pensamiento computacional apuntan a que enseñar este modo de pensar redundará en una mejora en otras áreas del saber, porque en definitiva, estimulará el desarrollo de la mente del sujeto. En palabras de Resnick (2008), enseñar a programar era para Papert y para él, "proveer a todos con oportunidades para encontrar y se-

guir sus propias pasiones, explorar y experimentar con nuevas ideas, desarrollar y hacer escuchar sus propias voces".

Programar es definido habitualmente como la acción de darle instrucciones precisas a una máquina en un lenguaje que la computadora pueda entender para que ejecute y automatice alguna acción o conjunto de acciones. Automatizar es hacer que una máquina repita tareas de manera más rápida y eficiente de lo que lo harían los humanos (Lee, 2011). Para Papert (1980), sin embargo, programar es esencialmente crear con tecnología, programar es construir con una computadora. Y más allá de ofrecer instrucciones precisas, la construcción es la representación de ciertas ideas a través de un lenguaje, es decir, expresar una forma de resolver un problema a partir de la creación de un artefacto. Pensemos, por ejemplo, en la construcción de una habitación. En ella está involucrada el propósito de esa habitación, para qué se usará y la técnica constructiva, los detalles de terminación que permitirán explotar sus funciones, los materiales, la estética, la robustez, la posibilidad de replicarla, etc. Del mismo modo, programar es construir.

Por ello, cuando hablamos de programación nos referimos a una disciplina que requiere **integrar** simultáneamente el uso de cierto grado de creatividad, un conjunto de conocimientos técnicos asociados y la capacidad de operar constantemente con abstracciones, tanto simbólicas como enteramente mentales (Martínez López, 2014). Según "Fidel" Martínez López (2014); lo que hace especial a la programación es que requiere emplear un conjunto de conocimientos técnicos asociados a la manipulación de las computadoras. Los programadores se dedican principalmente a construir programas, algo así como una "descripción ejecutable de soluciones a problemas computacionales, es decir, un texto descriptivo que al ser procesado por una computadora dará solución a un problema propuesto por los humanos". Y como bien menciona el autor, no cualquier texto es ejecutable por una computadora, la tarea del programador es codificar, es decir, escribir programas y dar soluciones a problemas de diversa índole. Por lo tanto, la computadora debe razonar las ideas de los programadores y traducirlas a código ejecutable, que es el que finalmente resolverá el problema en cuestión. Vemos entonces que inherente a la programación está la apropiación de un lenguaje formal y su uso para transmitir ideas.

Desde el trabajo de Lev Vigotsky (1999) en adelante, se ha discutido bastante en el campo de la educación la importancia del lenguaje para desarrollar el pensamiento.

Sabemos que el lenguaje contribuye, en primer lugar, para abstraer objetos de contenidos, tales como el contenido "perro", de cada uno de los perros que hemos conocido; también implica ordenar y jerarquizar ideas, desarrollar el pensamiento simbólico, crear y comunicar. Los lenguajes más formales como los que se usan en programación, requieren además abstraerse del significado del lenguaje, para que los símbolos sean operados como objetos matemáticos. Este proceso de abstracción de la semántica en el uso de un lenguaje ha sido denominado "des-semantización" (Dutilh Novaes, 2011). Para Dutilh Novaes, el razonamiento con lenguajes formales permite revisar creencias y argumentos, es decir, razonar sobre nuestras posiciones (Soler y Santacana, 2013). La enseñanza de la programación contribuye entonces a desarrollar este tipo de pensamiento.

En este sentido, es interesante la noción de educación en Programación que menciona Federico Aloi (2016) y su equipo de la Universidad de Quilmes, donde consideran que:

> *La educación en Programación requiere del dominio de diferentes herramientas informáticas tales como lenguajes de programación, bibliotecas, entornos de desarrollo, compiladores, etc., que se suman a lo más importante, que es la comprensión y utilización de herramientas conceptuales, de conocimientos y de estrategias que conforman una manera de hacer y sobre todo una manera de pensar (p: 210).*

Es decir, la programación **integra** el uso de un lenguaje para aplicar conceptos de lógica en la resolución de situaciones que requieren procesar información. Es justamente


la puesta en juego de conceptos lógicos y abstracción a través de una máquina para obtener un resultado lo que diferencia al pensamiento computacional de las TIC, cuyo énfasis está en el uso de tecnología previamente programada (Grover y Pea, 2013).

Crear un programa es una forma de externalizar nuestros pensamientos en un lenguaje no ambiguo. Esto nos permite analizar nuestros razonamientos y encontrar errores. La forma más efectiva de desarrollar el pensamiento computacional es aprender a resolver problemas con una computadora.

El ingreso del pensamiento computacional en las escuelas debe ser de manera "genuina", es decir, lo más parecido al pensamiento computacional que se pueda. Enseñar a construir programas implica romper con algunos patrones de la enseñanza clásica. En vez de partir de verdades, es necesario partir de problemas, el trabajo de copiado y memoria se cambia por la creación, el ensayo y el error, por la formulación de hipótesis y el testeo. Se elimina el trabajo individual por el colectivo, donde son los otros quienes activan también el pensamiento. De esta manera, el enfoque de enseñar Programación por indagación tensiona algunas estructuras clásicas de la escuela (Echeveste y Martínez, 2016).

Introducir genuinamente la computación y con ella el pensamiento computacional en la escuela, nos permitirá estimular procesos cognitivos en las situaciones de aprendizajes. Ahora bien, para lograr enseñar a programar de la manera más genuina posible, abordando el pensamiento computacional, debemos ser cuidadosos. Nos dice Zapata Ros (2015), haciendo referencia a su país, España:

> *Desgraciadamente la forma más frecuente de enseñar a programar y la que se está empezando a utilizar en nuestro país (…) está en la clave señalada: conducir a los alumnos, en este caso, de secundaria, por el camino más áspero, el de la programación per se. Si no se proporcionan otro tipo de ayudas o de claves, se excluye a los/as que no tienen el don de programar directamente, ante solo la comprensión de procedimientos puramente de programación, creando con ello el estereotipo de que la programación es solo cosa de los programadores (pp.6-7).*

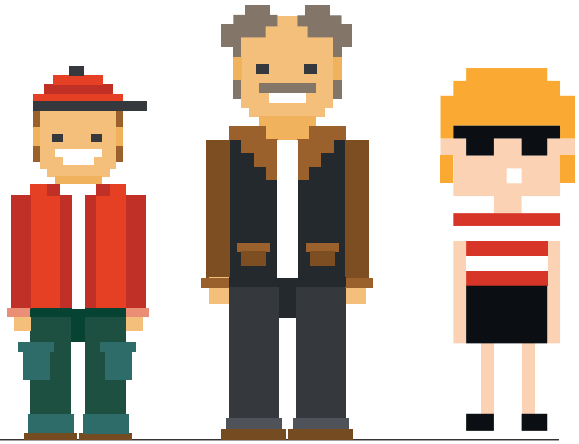
Zapata Ros nos advierte de que no basta incluir programación en el currículum para desarrollar el pensamiento computacional, sino que es necesario también abordar su enseñanza de manera que permita la construcción, la creación, la expresión, la representación, la resolución de problemas, la abstracción y la descomposición de un problema mayor en subproblemas.

# Algunas orientaciones para **enseñar a programar** en las aulas

La didáctica de la programación es un área en construcción, aquí recuperamos algunas experiencias singulares de trabajos docentes en el cual se introduce la programación en las aulas.

La didáctica de la Programación es un área en construcción. Si bien hay referentes nacionales e internacionales como Resnick y Silverman (2015), Papert (2001 y Rusk (2012); Martínez López, Bonelli y Sawady O'Connor (2012) poco sabemos todavía de cómo enseñar a programar en nuestras escuelas, que atienden a una población masiva con sus contextos particulares. Durante los cursos de capacitación que ofrecimos en UEPC, trabajamos con los/as docentes en dos dimensiones: 1) la dimensión conceptual. En ella abordamos qué es la programación y una selección de conceptos de programación a través del desarrollo de un proyecto que requiere programar. En esta dimensión dialogamos con los/as docentes en tanto profesionales adultos que están aprendiendo rudimentos de una disciplina nueva. 2) En una instancia posterior al proyecto individual de programación, abordamos la construcción didáctica. Como profesionales de la educación los/as docentes tienen experiencias, saberes y conocimiento de sus estudiantes para aportar al campo de la enseñanza de la Programación como veremos en las páginas que siguen. Los/as docentes realizan una selección de contenidos y elaboran proyectos integradores para enseñar conceptos de programación en el marco de su materia o sus clases.

En esta oportunidad queremos recuperar algunas experiencias singulares de trabajos de docentes que nos llevarán a repensar algunos aspectos relacionados a la estructura tradicional de nuestra escuela. En algunas prácticas realizadas de formación en Programación, los/as docentes han llevado lo aprendido a sus aulas de maneras muy diversas: talleres fuera del horario de cursado, clases integradas con diferentes materias, diversidad de edades dentro de módulos de clases, clases tutorizadas por alumnos/as más expertos o más grandes e incluso alumnos autodidactas. Estas modalidades mostraron una fuerte tendencia al trabajo grupal, incluso entre jóvenes de distintas edades y con diferentes trayectorias; otra característica que interpela al formato escolar tradicional.



# Escenas de experiencia

**En esta sección veremos diferentes experiencias que fueron llevadas a cabo por docentes que tomaron nuestro curso. Éstas tienen elementos en común que analizaremos en detalle a través de algunas categorías didácticas para pensar en la enseñanza de la programación.**

## ESCENA 1

### *Escuela Secundaria. Departamento Colón. 2016*

*La profesora Analía comienza la clase indagando entre sus alumnos/as de segundo año sus saberes sobre videojuegos. Seguramente muchos/as de los/as estudiantes se habrán preguntado qué tendrá que ver la Química (materia a su cargo) con los videojuegos. Luego de recuperar algunas de sus experiencias como expertos "gamers", Analía les presenta una plataforma para programar videojuegos llamada Scratch y les muestra cómo cargarla en sus netbooks.*

*Les propone desarrollar un videojuego que complemente las actividades que están haciendo en la huerta de la escuela donde los/as estudiantes realizan un "Bioinsecticida" para mostrar en la Feria de Ciencias de la Escuela. El desafío propuesto era desarrollar un videojuego que permita eliminar insectos que afectan a la huerta, pero dejando vivos aquellos que no destruyen las plantaciones.*

*En su reflexión sobre la clase, Analía comenta que si bien los/as estudiantes no tenían conocimientos formales de programación, pudieron ir estableciendo relaciones entre los conceptos y las acciones conocidas por ellos para jugar a videojuegos. Analía, además, escribe: "Los alumnos se mostraron expectantes y ansiosos ante la posibilidad de poder crear sus propios videojuegos. La clase se realizó con mucha dinámica producida por sus propios intereses. Los estudiantes querían seguir estas producciones de videojuegos en otras asignaturas".*

*El trabajo se realizó de manera grupal y en un clima de colaboración donde los/as estudiantes se ayudaban mutuamente. Tres grupos, que continuaron trabajando fuera del espacio del aula, presentaron sus videojuegos terminados en la Feria anual del IPEA.*

# La recuperación de saberes e intereses previos

Reconocer que tenemos saberes y experiencias para brindar a nuestros/as estudiantes, al tiempo que aprender de sus destrezas de usuarios.

Una dimensión que trabajamos fuertemente en nuestro curso de enseñanza de la Programación es el reconocimiento de los saberes previos de nuestros/as estudiantes. En primer lugar, trabajamos nociones que muchos/as docentes traen cuando mencionan que **"los/as chicos/as saben más que nosotros de computación y tecnología".** Esta idea nos inquieta porque sabemos que lo que nos constituye como docentes es en parte nuestro saber. Es decir, somos docentes porque tenemos un conocimiento para transmitir que otros no tienen. En el caso de la tecnología, no pocos docentes pensamos que no sabemos (Cabello, 2006) y por tanto, trabajar sobre tecnología en el aula pone en riesgo nuestra legitimidad, nuestro rol en tanto poseedores del saber.

En las primeras experiencias docentes frente la programación, analizadas por Papert (2003) en su libro "La máquina de los niños", relata que la conciencia de ser profesores les impedía entregarse por completo a la experiencia de aprendizaje. Una experiencia que fuera pensada como una construcción colaborativa, donde los/as estudiantes pueden recibir pero también pueden dar, un proceso de aprendizaje donde el/la docente contribuía al proceso que ellos/as seguían. Beatriz Greco (2012) también tensiona la idea de pensar disociadamente que quien enseña no es pensado aprendiendo y viceversa. Y reconoce

> Somos docentes porque tenemos un conocimiento para transmitir que otros no tienen. En el caso de la tecnología no pocos docentes pensamos que no sabemos (Cabello, 2006) y por tanto, trabajar sobre tecnología en el aula pone en riesgo nuestra legitimidad, nuestro rol en tanto poseedores del saber.

que la autoridad que poseen los/as docentes en el territorio educativo se vuelve autorización cuando se hace habilitadora de aquello que aún no acontece y que solo puede desplegarse en ese "entre" de enseñantes y aprendientes.

Nuestra experiencia es, que si bien los/as jóvenes y niños/as saben usar redes sociales y aplicaciones de celulares, tienen poca experiencia en programación. Si la programación está relacionada con el pensamiento lógico, matemático y la construcción de un lenguaje; nosotros/as como docentes tenemos mucho que ofrecer en estas áreas. En ese sentido, es importante reconocer que tenemos saberes y experiencias para brindar a nuestros/as estudiantes, al tiempo que aprender de sus destrezas de usuarios.

En segundo lugar, analizamos junto con los/as docentes la famosa premisa didáctica de recuperar los saberes previos. Lejos de ser una formalidad, aquel supuesto permite disponer una estructura cognitiva, un esquema mental para apropiarnos de nuevos conceptos. Solo podemos comprender conceptos de los cuales hacemos un registro. Ese registro es habilitado por los saberes previos. Del latín, la palabra "registrar" significa "llevar a atrás". Es decir, para que nuestros/as estudiantes puedan ofrecer significado a los nuevos saberes que ofrecemos, tenemos que hacer el esfuerzo de llevarlos hacia atrás, hacia experiencias y conocimientos que ellos/as ya tienen.

La pregunta con la cual se inician muchas de las experiencias: "¿Quiénes juegan a videojuegos?", tiene una intencionalidad didáctica, que es recuperar esas nociones previas sobre los videojuegos y esa motivación y emoción que tienen muchos/as jóvenes, niños/as por jugar a los videojuegos. No comenzamos la clase mostrando la plataforma o explicando qué es programar. Sino que la propuesta es recuperar sus pasiones: a qué les gusta jugar, cuál es su juego favorito, y sobre todo qué hace que ese juego sea el que más les guste. Esa última consigna invita a analizar a los videojuegos: sus personajes, escenarios, movimientos, interacciones entre objetos. Este análisis va preparando la estructura mental de los/as alumnos/as para presentar los conceptos que permiten que esos personajes hagan lo que hacen en los videojuegos. Hacer que estos personajes "hagan cosas" es justamente programar.

# ESCENA 2

## *Escuela Secundaria. Departamento Colón. 2016*

*La clase se inicia con la recuperación de saberes previos que sirvió para motivar, desafiar y generar interés en los/as estudiantes. La docente Sandra les pregunta si les gusta jugar a los videojuegos e inmediatamente les pide que describan las animaciones de sus videojuegos favoritos. En estas descripciones deben estar incluidos los escenarios, los personajes y las acciones que realizan cada uno/a y su relación entre ellos. Luego de poder observar, recuperando la descripción de los/as alumnos/as, cuáles eran los elementos centrales de un videojuegos, la docente les plantea el desafío de realizar un videojuego sobre prevención del Dengue, de manera similar al videojuego que Sandra había realizado en el curso dictado por la UEPC. Para ello les presenta la plataforma Scratch.*

*Intencionalmente, Sandra muestra algunos aspectos de la plataforma y algunos menús básicos para comenzar a programar sus videojuegos. De esta manera promueve que sean los/as propios/as alumnos/as quienes exploren y aprendan a utilizar la herramienta. Los/as estudiantes no tenían saberes previos formalizados de programación. Los/as jóvenes usaban la computadora para jugar, pero nunca para crear sus propios juegos.*

*Inicialmente, la actividad estuvo pensada para un grupo de alumnos/as del colegio, pero al enterarse los/as otros/as docentes de la escuela, este proyecto se transformó en una propuesta interdisciplinaria que integró Biología y Química y que se presentó en la Feria de Ciencias del Ipem.*

*La docente relata: "El clima de la clase al comienzo fue bastante complicado, ellos no entendían el motivo de la utilización del programa porque en Ciudadanía y Participación estamos acostumbrados a trabajar de otra manera. Me costó hacerles entender el objetivo de la clase. Una vez que lo entendieron se motivaron y ahí sí, el clima cambió y se hizo una clase amena, pero sobre todo de interacción mutua entre ellos y yo. Y el rendimiento fue bueno, la mayoría de los alumnos se engancharon... todos terminaron el proyecto... la calidad fue buena, si bien hubo estudiantes que solo lo hicieron para cumplir, sé que la mayoría lo hizo a conciencia y se divirtió".*

# El trabajo por proyectos significativos que impliquen DESAFÍOS DE PROGRAMACIÓN

El aprendizaje por proyectos permite ofrecerles a los/as estudiantes la oportunidad de trabajar en el desarrollo de un proyecto significativo y concreto de programación.

En las experiencias que propusieron los/as docentes a sus alumnos/as se ofrecía un desafío para que ellos/as elaboraran un proyecto de video juegos o animaciones. Más que una consigna o una tarea, la propuesta era una invitación a crear un producto tecnológico significativo para ellos/as.

Según el equipamiento, software y saberes que busquemos construir junto con los/as alumnos/as, los proyectos de programación pueden ser de lo más diversos. Desde aplicaciones con celular hasta trayectorias de un robot que se mueve solo y esquiva objetos. Todo eso es programable.

El aprendizaje por proyectos es una idea bastante elaborada de la didáctica general. Los proyectos permiten crear artefactos, programas, bienes intangibles como textos o películas, o realizar intervenciones sociales en donde los/as alumnos/as pongan a disposición e integren saberes de diferentes disciplinas que les permitan elaborar ese proyecto. Como vimos en las experiencias, los proyectos que requerían crear videojuegos generaban entusiasmo. Resnick (2005) sostiene:

> *Como Papert, creemos que las mejores experiencias de aprendizaje, para la mayoría de la gente, vienen*

> Como Papert, creemos que las mejores experiencias de aprendizaje, para la mayoría de la gente, vienen cuando están activamente comprometidos en diseñar y crear cosas, especialmente cosas que son significativas para ellos o para otros alrededor de ellos.

*cuando están activamente comprometidos en diseñar y crear cosas, especialmente cosas que son significativas para ellos o para otros alrededor de ellos (Resnick & Silverman, 2005, p. 1).*

La idea de elaboración de proyectos significativos nos remite a lo que Papert denomina "resonancia cultural", es decir, que tenga eco, que pueda ser llamado dentro de su esquema cultural.

Muchos de nuestros/as estudiantes no serán computólogos ni ingenieros, como tampoco serán escritores o químicos. No obstante, ofrecerles la oportunidad de trabajar en el desarrollo de un proyecto significativo para ellos/as que requiere de nociones y conceptos de la programación les permite encontrarse con "ideas poderosas" como una parte natural de la experiencia de diseñar (Resnick & Silverman, 2005, p. 2). Ahora bien, la aparición de estas ideas no pueden forzarse ni tampoco pueden enseñarse de manera directa, una propuesta didáctica direccionada y prescriptiva muy probablemente inhiba y empobrezca la aparición de las mismas o las ignore en el caso de que surjan.

El trabajo por desafíos que promovemos en nuestro curso consiste en proponer un proyecto concreto de programación. Por ejemplo, como vimos en la escena recuperada: realizar un videojuego para matar mosquitos, que es una propuesta amplia, pero orientada con actividades intermedias. Las actividades intermedias se presentan en forma de desafíos de programación que al resolverse van solucionando una parte del proyecto. Como dijimos antes, la programación implica descomponer un problema en subproblemas. Para los ejemplos que trabajamos, el desarrollo de un videojuego sobre el Dengue es el problema, y hacer que los mosquitos vuelen con una trayectoria en espiral sería un subproblema donde a su vez, hacer el que mosquito aparezca en la pantalla constituye otro subproblema de ese subproblema. Los desafíos de programación que proponemos abordan estas cuestiones en sus distintos niveles constituyendo actividades intermedias para orientar un proyecto. Al contrario de la tarea direccionada, el desafío invita a explorar, descubrir y resolver en subproblemas más acotados y alcanzables. Por ejemplo, un desafío puede ser "ahora hagamos que aparezca un mosquito y dé tres vueltas". Este tipo de trabajo resulta una estrategia para favorecer su emergencia e invita a nuestros/as estudiantes a explorarlas y entenderlas.

# ESCENA 3

## *Escuela Primaria. Córdoba Capital. 2015*

*Entre el alumnado, había un caso especial: un chico con síndrome de Asperger. Andrea ya me había contado un poco de él, y que a veces le costaba la interacción con sus pares o gente nueva. Me decía: "Te vas a dar cuenta de quién es"; y efectivamente me di cuenta, fue el primero que llegó a la clase. (...) al principio se lo vio un poco tímido en la relación, pero luego me sorprendió. (...) Quizás, en algún sentido, le cueste un poco la interacción con sus pares (puede ser algo típico de su condición), pero demostró un interés enorme por aprender, por desafiarse y por conocer más sobre las nuevas herramientas y la tecnología".*

# ESCENA 4

## *Escuela Primaria. Córdoba Capital. 2015*

*Hacho, un nene "integrado", supuestamente tiene cierto nivel de retardo, así me lo plantearon, mientras todos hacían las actividades estaba solo sentado al lado de su grupo (...) entonces le ofrecí acompañarlo a hacerla. Nos sentamos en un costado los dos y la verdad que el nene la hizo de goma literalmente, lo hizo todo solo y super rápido. Es más, en la página ocho del libro, al final hay un ejercicio extra para "expertos", y como había terminado antes que los otros, le pregunté: '¿Te animas a hacerlo?'. Me dijo: 'Sí' (con un poquito más de confianza) y como no le salía muy bien así en el aire el ejercicio, ¡lo representó con lápices de distintos tamaños! ¡Sinceramente quedé sorprendido! Yo sin conocimientos psicológicos, me atrevería a decir que ese chico no tiene retraso, pero eso queda en manos de los especialistas".*

(Relatos de los tutores que acompañaron algunas experiencias de enseñanza en las escuelas)

# Piso bajo y techo alto

Los proyectos de se pueden completar con programaciones sencillas para obtener rápidamente resultados (piso bajo) y al mismo tiempo ofrecen la posibilidad de construir proyectos cada vez más complejos (techo alto).

En la didáctica específica de la Programación se habla de que los proyectos diseñados para enseñar a programar deberían tener **"piso bajo y techo alto"** (Resnick y Silverman, 2005). Para explicar esta intencionalidad didáctica Resnick utiliza la experiencia con los bloques de Rastri o Lego como modelo de lo que muchos/as de nosotros/as intentamos conseguir con la apropiación que hacen los/as niños/as de las tecnologías. Los Rastri consisten en una cantidad reducida de bloques de diferentes formas y tamaños. Con ellos se puede construir desde un puente (muy bajito) con tan solo tres piezas, hasta una persona de tamaño natural o ciudades en escala que son verdaderas obras de ingeniería.

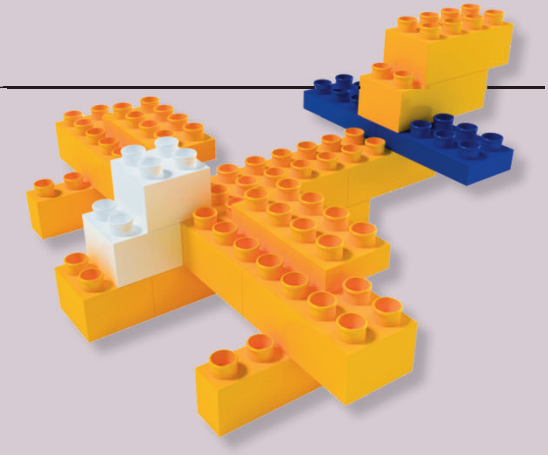
Justamente esta es la base de muchas plataformas diseñadas para aprender a programar, entre ellas la plataforma Pilas Engine: ofrecen pocas herramientas simples y versátiles, pero que combinadas con creatividad y con la aplicación de la Matemática y la Física, pueden generarse construcciones complejas.

Los proyectos diseñados con el criterio "piso bajo-techo alto" son proyectos que se pueden completar con programaciones sencillas que requieren pocas líneas de código para obtener rápidamente resultados (piso bajo) y al mismo tiempo ofrecen la posibilidades de construir proyectos cada vez más complejos y sofisticados a medida que se explora el entorno y se conocen y combinan sus herramientas (techo alto). Además, permiten la alternancia en una amplia gama de exploraciones y habilitan una gran variedad de proyectos diferentes (paredes anchas). En este sentido, son "accesibles y atractivas para niños/as con diferentes estilos de aprendizaje y modos de conocimiento" (Resnick, 2008).

Papert (2003), retoma de Levis Strauss (1966) un tipo de pensamiento denominado "Bricolage", como una metodología para la actividad intelectual donde se utiliza lo que se tiene, intentando encontrar la manera más adecuada para resolver el problema y si hay una herramienta que no sirve, simplemente buscar otra.

Es importante que la propuesta didáctica sea accesible e invite a los/as estudiantes a "ponerse manos a la obra" en proyectos que surjan de sus propios intereses obteniendo rápidamente resultados básicos; y que contemple a su vez estrategias para profundizar y ahondar en estas exploraciones iniciales de cara a los objetivos específicos de cada proyecto.

> Papert (2003), retoma de Levis Strauss (1966) un tipo de pensamiento denominado "Bricolage", como una metodología para la actividad intelectual donde se utiliza lo que se tiene, intentando encontrar la manera más adecuada para resolver el problema y si hay una herramienta que no sirve, simplemente buscar otra.



En todas las experiencias los/as docentes comentaron que hubo grupos que lograron mejores programas que otros. Por ejemplo, en el caso del Dengue, algunos grupos lograron que los mosquitos hicieran varias trayectorias antes de caer, o que el insecticida hiciera un ruidito cuando fuera colocado, o que los jugadores sumaran puntaje. Mientras que los proyectos más básicos eliminaban mosquitos de manera menos sofisticada. En ambos casos, el diseño del proyecto requirió que se usaran conceptos básicos como el condicional. Si tiro el veneno, el mosquito muere, sino sigue. En efecto, la experiencia de enseñanza debería estar diseñada de modo tal que la única forma de completarla sea desplegando las habilidades de diseño y creación que pensamos son aprendizajes relevantes para nuestros alumnos/as (Newmann, 1996).

Sin embargo, algunos grupos pudieron seguir explorando la herramienta y usar otras funciones para completar su proyecto. El proyecto no tiene techo, puede avanzar cuanto más puedan también los/as estudiantes según intereses y saberes previos.

# Habilitar diferentes estilos de trabajo y formas de resolución

Relatos recuperados de las aulas sugieren que es posible repensar algunas concepciones de sujeto "educable", centrada en presunciones obtenidas de conductas que difieren a las" esperadas" por el sistema escolar tradicional.

La versatilidad de las plataformas diseñadas para la enseñanza de la Programación mediante el desarrollo de videojuegos, combinada con propuestas abiertas que, invitan a los/as estudiantes a explorar herramientas y desarrollar estrategias para dar solución a diferentes problemas, suelen promover distintos estilos de trabajo y formas de resolución para una misma situación problemática.

Los relatos recuperados de las aulas sugieren que es posible repensar algunas concepciones de sujeto "educable", centrada en presunciones obtenidas de conductas que difieren a las esperadas por el sistema escolar tradicional. Los/as jóvenes catalogados previamente por el sistema o por sus docentes como "los que nunca se enganchan", "los repitentes", "los/as niños/as con retraso" han logrado un desempeño que no solo sorprendió a sus docentes, sino que superó el desempeño del resto de sus compañeros/as.

Esta diversidad se aleja del ideal de normalidad con el que fue forjado el sistema escolar, donde dispositivos homogéneos disrumpen en las nuevas poblaciones heterogéneas que habitan las escuelas de estos tiempos. Al decir de Barquero, la enseñanza de la Programación pondría en tensión la noción tradicional de educabilidad "como la de-

> Papert (2003), retoma de Levis Strauss (1966) un tipo de pensamiento denominado "Bricolage", como una metodología para la actividad intelectual donde se utiliza lo que se tiene, intentando encontrar la manera más adecuada para resolver el problema y si hay una herramienta que no sirve, simplemente buscar otra.

limitación de las condiciones, alcances y límites que posee potencialmente la acción educativa sobre sujetos definidos en situaciones definidas" (2001, 72). Sostener un dispositivo escolar común sin tener en cuenta la necesidad de adaptar y/o recrear propuestas educativas **integradoras** para involucrar a todos/as, atenta contra la diversidad que deseamos para nuestras escuelas. Si lo pensamos como la capacidad de ser educados, se vuelve a ubicar la cuestión en la individualidad del/la estudiante como problema de desarrollo cognitivo e intelectual. Desde los nuevos paradigmas de integración, en cambio, debemos adecuar las condiciones y las ofertas de enseñanza para que todas tengan la posibilidad de habitar y aprender en la escuela.

Desde sus orígenes, la escuela se acostumbró a realizar distinciones entre sus alumnos/as. Separarlos por edad en salones graduados, de acuedo al género: escuelas de señoritas o liceos de hombres; por creencia religiosa: instituciones judías, laicas o católicas. Separarlos por clases sociales (Bourdieu y Passeron, 1996); diferenciarlos según sus procesos cognitivos, como los casos de niños/as con conductas diferentes diagnosticados con desatención e hiperactividad (Janin, 2005); o jóvenes que no responden a los recorridos esperados por el sistema educativo con trayectorias escolares que Flavia Terigi (2008) menciona como teóricas, las cuales expresan recorridos de progresión lineal y de tiempos marcados por una periodización estándar; lo cual genera en los sujetos escolares descripciones de "sobreedad" con respecto a ciertos sentidos ideales y esperados por las instituciones escolares (Baquero, 2009). Todas estas distinciones abarcadas desde la individualidad de los sujetos generan estigmas y etiquetamientos, que permiten distinguir, como lo expone Kaplan (1992), entre "buenos" o "malos" alumnos/as o experiencias traducidas en "fracasos escolares".

Los enfoques socioculturales consideran que la categoría de "educabilidad" se trata más de una propiedad de las situaciones educativas en las que están implicados los sujetos con su singularidad que de una propiedad de los individuos evaluable en forma descontextualizada (Baquero, 2009). La educabilidad aparentaba ser un atributo evaluable en los individuos con independencia de las propiedades de contexto, desconociendo su carácter situacional y subjetivo.

La tarea del/la docente debe atender a los distintos modos de trabajo habilitándolos y validándolos. Resulta importante también gestionar instancias de intercambio entre los/las estudiantes que permitan compartir las distintas ideas, fomentando el aprendizaje entre pares.

# ESCENA 5

## *Actividad de interacción y relación entre niños/as de una escuela primaria y una escuela secundaria. Departamento Colón. 2016*

*Lorena comenta que los saberes previos de los/as estudiantes son bastante diferentes como es de esperar de acuerdo a la edad de cada grupo. Mientras los/as alumnos/as de primer grado tenían un manejo acotado de la computadora, los/las de sexto podían reconocer secuencias y algoritmos en los videojuegos. Por su parte, los/las estudiantes del secundario poseían conceptos de manera intuitiva de la programación, pero que lograron formalizar con el uso de Scratch.*

*El desafío fue crear un videojuego para enseñar a cruzar la calle. Los/as estudiantes de secundario comenzaron la clase preguntando a los/as más pequeños/as cómo cruzan la calle y qué cuidados tienen para caminar por la calle. A partir de esta pregunta los/as niños/as introdujeron el objeto semáforo en su animación y aplicaron el concepto de condicional para indicar cuando cruzaba el auto, seguía de largo o frenaba.*

*Por una parte, los/as estudiantes de secundario mostraron mucho interés por conocer qué juegos les gustaban a los/as más pequeños/as mostrando relaciones de solidaridad, respeto e interés. Por otra parte, los/as alumnos/as de primer grado lograron elegir un escenario para un videojuego, un objeto y vincular los objetos a través de un condicional. Asimismo trabajaron en la resolución de problemas. Integraron saberes de lengua para dar instrucciones, y de numeración para poder crear los movimientos.*

*Ambos querían seguir creando y llenando de detalles su animación, pero el tiempo fue escaso. Según el relato de la profesora Marcela (docente a cargo de los/as alumnos/as de secundaria): "La experiencia fue maravillosa (subrayado en el original), fue muy rica. Los chicos del secundario fueron a conocer a los pequeños e interactuaron con sus conocimientos y sus nociones. El clima fue increíble ya que era la primera vez que realizaban intervención en esta escuela primaria con el Ipem. los pequeños se sintieron muy contenidos y preguntaban a los mayores que iban a enseñarles en la "sala de la compu". Los más grandes abordaron el proyecto con mucho entusiasmo y responsabilidad. Todos aprendieron el programa Scratch.*

# ESCENA 6

## *Escuela secundaria. Córdoba Capital- 2015*

*En el caso de la alumna Teresa, ella empieza a copiar y pegar pedazos de un código que ya estaba escrito. Le pregunto qué está haciendo, ya que estaba sentada al lado, y me dice que está adaptando algo ya escrito a lo que tiene que hacer ahora. Prueba para ver si le sale, pero no lo logra. Entonces vuelve al código anterior y sigue probando. Intenta, aparecen las flechas, pero no de la forma que ella quería, entonces vuelve al código y continúa. Cada vez que prueba va mejorando. Luego de unos minutos me dice: 'Lo hice', y sonríe.*

# Alentar el trabajo colaborativo

El intercambio y colaboración entre los/as estudiantes, permite que se asistan unos/as a otros/as pensando estrategias para sortear problemas o ideas para continuar desarrollando cada proyecto.

En consonancia con lo expuesto anteriormente resulta relevante promover instancias de intercambio y colaboración entre los/as estudiantes para que estos comenten los avances y dificultades de sus proyectos, así como también sus "descubrimientos". Estos intercambios permiten que se asistan unos/as a otros/as pensando estrategias para sortear problemas o ideas para continuar desarrollando cada proyecto.

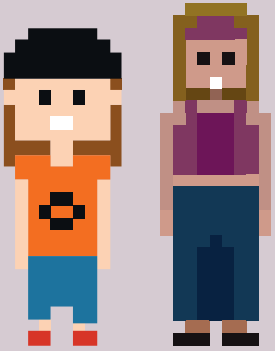
Por otro lado, si bien la secuencia didáctica debe ser pensada, prevista, proyectada y planificada; el trabajo con proyectos, basado en los criterios anteriores, propicia la aparición de situaciones que el/la docente nunca antes ha visto. Como nos explica Papert (2001):

Ni la simulación ni el autocontrol son necesarios cuando el/la maestro/a y el/la estudiante se enfrentan a un problema real que surge de manera natural en el desarrollo de un proyecto. El problema es un reto para los dos. Ambos pueden aportar lo suyo.

Así, el/la docente ya no ocupa un lugar central impartiendo conocimientos y se convierte en un coaprendiz. Este tipo de propuestas alienta el trabajo colaborativo donde los/as estu-

> Ni la simulación ni el autocontrol son necesarios cuando el/la maestro/a y el/la estudiante se enfrentan a un problema real que surge de manera natural en el desarrollo de un proyecto. El problema es un reto para los dos. Ambos pueden aportar lo suyo.

diantes entre sí y con el/la docente comparten los genuinos descubrimientos que emergen como parte del desarrollo del proyecto. De este modo, este tipo de trabajo promueve la ayuda mutua entre compañeros/as. Lo que se refuerza con la cotidianeidad de observar en las demás clases que se consultan entre ellos/as cuando alguno no puede continuar con su programa. Meirieu (1998), al respecto, propone renunciar a querer formar grupos homogéneos debidamente preparados para seguir un tratamiento estandarizado y considera pertinente afrontar la heterogeneidad en el mismo grupo de trabajo, tal como se manifiesta ante una tarea y sobre todo ante una situación problemática.



# Tratamiento del error y las emociones

Actividades de prueba y error nos muestran que equivocarse permite acercarnos a conocimientos inéditos. El aprendizaje de algo nuevo, generalmente, provoca una serie de estados y emociones que van desde la frustración frente a la dificultad hasta el entusiasmo y la satisfacción cuando se consigue dar solución a un problema.

En términos generales, podemos decir que el aprendizaje de algo nuevo, generalmente, provoca una serie de estados y emociones que van desde la frustración frente a la dificultad, hasta el entusiasmo y la satisfacción cuando se consigue dar solución a un problema. En estos últimos veinte años los psicólogos cognitivos nos advierten que las emociones son parte de los procesos de aprendizaje. La emoción guía la cognición. Porque tenemos curiosidad, necesidad de saber algo, de resolver un problema, de ayudar a alguien que queremos a través de apropiarnos de información, es que aprendemos.

Resulta importante que, como docentes, acompañemos estos procesos de aprendizaje. En primer lugar, presentando propuestas que despierten emociones. Como preparar una campaña de salud sobre alguna enfermedad que haya afectado a personas que conocemos, o fomentar propuestas de convivencia, por caso, son propuestas de enseñanza que tienen el potencial de despertar emociones e involucrar a los/as estudiantes. En segundo lugar, ofrecer estrategias para manejar frustraciones, errores, códigos fallidos o intentos que no salieron puede dar buenos resultados. En este sentido, Natalie Rusk sostiene que todas estas emociones tienen funciones importantes en el proceso de aprendizaje de nuestros estu-

> Porque tenemos curiosidad, necesidad de saber algo, de resolver un problema, de ayudar a alguien que queremos a través de apropiarnos de información, es que aprendemos.

diantes: "Si los estudiantes pueden aprender a manejar efectivamente la frustración, es mucho más probable que persistan en lugar de darse por vencidos cuando encuentran alguna dificultad" (2012).

Actividades de prueba y error, como las que mencionamos en la escena anterior, nos muestran que equivocarse permite acercarnos a nuevos conocimientos. En la programación, muchas veces, es la misma práctica la que marca los errores y los aciertos a la hora de hacer funcionar un programa sin necesidad de una supervisión constante, en este caso del/la docente. El *feedback* inmediato que nos posibilita la computadora favorece no solo la exploración y descubrimiento de los contenidos, sino que les permite avanzar y otorgar autonomía y confianza a sus alumnos/as en la aprehensión de los conocimientos (Echeveste, 2017).

Una estrategia que podemos usar para evitar la frustración y provocar la emoción en el aprender es tratar de identificar el potencial de nuestros/as estudiantes para diseñar una propuesta que los convoque. Nos dice el psicólogo cognitivo Dylan William (2006):

> *Cuando el potencial de aprendizaje del estudiante es alto y el nivel del desafío de la tarea es bajo, el resultado es el aburrimiento. Cuando el potencial de aprendizaje es bajo y el desafío es alto, el resultado es la alineación y la frustración. Pero cuando el desafío está justo al límite del potencial del estudiante, la sensación es de 'fluidez'.*

En ese sentido, la motivación no sería un requisito previo para aprender, sino que un resultado de la propuesta de enseñanza. Muchos hablamos de la motivación como si fuera algo que los/as estudiantes deben traer antes de entrar a la escuela, como una configuración previa. Autores como Dylan William (2006), nos invitan a ver la motivación de otra manera, es decir, no como la causa del rendimiento académico sino como el resultado del mismo: porque aprendo, me desafío y me motivo.

Por otra parte, William plantea algo muy común que les sucede a los programadores, que es que alguien diga: "Voy a programar por cinco minutos más y dejo, y esos cinco minutos se convierten rápidamente en tres horas, porque el programador se ha perdido completamente en la actividad". Trabajar con el potencial de nuestros/as estudiantes, o en palabras Vigotskianas, "con su zona de Desarrollo Próximo", requiere que los conozcamos, aprendamos de sus experiencias, saberes e intereses.

# Reflexiones finales

En el transcurso de este escrito presentamos experiencias en escuelas de Córdoba donde docentes de primaria y secundaria ofrecieron propuestas de enseñanza de la Programación incluidas en proyectos más amplios. De las escenas relatadas rescatamos:

- La experiencia del trabajo docente con otros docentes, de su misma escuela y de otras escuelas.
- La articulación entre primaria y secundaria.
- La posibilidad de aprendizaje que desplegaron muchos/as estudiantes cuando programaban su videojuego, siendo que algunas veces no pueden completar las tareas propuestas por la escuela.
- La construcción de vínculos con un saber que los convoca, como es la programación de videojuegos, de alumnos/as que, por lo general, tienen relaciones de baja intensidad con la escuela.
- La ruptura con algunas características del formato escolar tradicional (quedarse en el recreo, agruparse heterogéneamente).
- El rol docente como diseñador de la experiencia y coordinador de los proyectos.
- Y el desarrollo de un tipo de pensamiento que se denomina "computacional".

Esta publicación no intenta ofrecer verdades acabadas ni recetas iguales para todos. En el campo educativo, "qué funciona y qué no funciona" es generalmente una pregunta equivocada:

> *"Todo funciona en algún lugar, y nada funciona en todos los lugares y es por eso que en educación la pregunta debería ser 'cómo funciona una propuesta y en qué condiciones'. (Dylan William, 2016).*

Como educadores/as e investigadores/as, somos conscientes de que tenemos mucho que aprender sobre qué pasa cuando enseñamos Programación en nuestras escuelas. Hace tan solo cinco años atrás, pedagogos, computólogos, docentes, funcionarios y toda la comunidad educativa nos preguntábamos: "¿Será posible enseñar a programar en las escuelas?". Recordamos al director de la carrera de Computación de una prestigiosa Universidad Nacional de Argentina mirándonos a los ojos con genuina curiosidad diciéndonos: "Yo he enseñando todas las materias en esta carrera, hace 15 años que soy docente en computación y no me imagino cómo enseñar programación en primaria y secundaria. No digo que no se pueda, simplemente que yo no me lo imagino".

Durante muchos años la Programación fue estereotipada como una disciplina solo "para inteligentes", "nerds" o "para unos pocos". Sin embargo, sosteniendo las premisas de la educación como derecho, motor del desarrollo del pensamiento crítico y de la alfabetización, comenzamos a construir experiencias de enseñanza de Programación en las escuelas junto con los/as docentes.

Para alfabetizar en los lenguajes de nuestro tiempo e incluir a todos nuestros/as jóvenes en una tecnología disruptiva que permita desarrollar el modo en que aprendemos, conocemos, entendemos y transformamos nuestro mundo con los nuevos artefactos tecnológicos, necesitamos ampliar la mirada sobre la enseñanza de la computación e incluir la Programación. ¿Por qué? Porque es la mejor manera que conocemos de enseñar el *pensamiento computacional*.

Diversos cursos se han ofrecido en los últimos años desde la Universidad y UEPC. La estructura y el compromiso que tiene el gremio de los educadores de Córdoba con la descentralización a las localidades del interior de la Provincia, nos ha permitido llevar la enseñanza de la Programación a ciudades muy pequeñas donde ni siquiera habían llegado

los cursos de Conectar Igualdad. En muchas ocasiones los/as docentes nos decían: "Es el primer curso en tecnologías que tenemos desde que llegaron las netbooks". Varias provincias están llevando a cabo también un postítulo en Enseñanza de la Programación, siendo Córdoba la primera en el país.

Cinco años después de embarcarnos en esta propuesta y de alcanzar junto a UEPC a más de 600 docentes de la provincia, nuestras preguntas han cambiado. A partir de las experiencias con los/as docentes en las escuelas de Córdoba nos preguntamos: ¿qué hace que los/as niños/as con necesidades especiales tengan un alto rendimiento cuando les proponemos programar?, ¿qué tipos de pensamientos estamos requiriendo de ellos/as que el abordaje de las materias tradicionales generalmente no requiere?, ¿cuáles son los patrones de trabajo colaborativo que observamos cuando los/as jóvenes programan en grupos?, ¿cuáles son los usos del tiempo escolar en las clases donde se enseña Programación?, ¿qué conceptos transversales podemos abordar de manera efectiva?, ¿qué proyectos tecnológicos podemos desarrollar junto a nuestros/as estudiantes con el aporte de computadoras programables?, ¿cómo son los intercambios entre docentes y alumnos/as?

Estas y otras preguntas han surgido al observar que la enseñanza de la Programación en las escuelas, permite integrar a estudiantes con capacidades especiales o con potencial de aprendizaje que no era visibilizado previamente; articular disciplinas y abordar conceptos de manera transversal; y coordinar entre niveles ofreciendo una nueva oportunidad para trabajar cooperativamente entre los/as docentes.

Los invitamos a incluir el pensamiento computacional en sus aulas, repensar los modos de enseñar a programar *a* y *con* sus estudiantes, potenciar la integración con materias afines y continuar reflexionando sobre nuestras prácticas en el aula y sobre las nuevas propuestas de enseñanza.

# Bibliografía


▶ Aloi, F., Bulgarelli, F., Palumbo, N., & Spigariol, L. (2016). Corrección automatizada de programas como recurso pedagógico. En XI Congreso de Tecnología en Educación y Educación en Tecnología (TE&ET 2016).

▶ Baquero, R. (2001). La educabilidad bajo sospecha. Cuaderno de Pedagogía. Rosario. Año IV, N 9 71-85.

▶ Bourdieu, P., & Passeron, J. C. (1996). La Reproducción: elementos para una teoría del sistema educativo. México: Fontamara.

▶ Busaniche, Beatriz (2006). Alfabetización digital: las fronteras del aprendizaje y el control de la Información. Recuperado de: http://www.bea.org.ar/wpcontent/uploads/2006/10/alfabetizaciondigital.html

▶ Casablancas, Silvina; Caldeiro, Graciela Paula; Odetti, Valeria (2016). La mirada de los sujetos de educación secundaria en los nuevos escenarios educativos. ¿Qué cambió a partir de la llegada de las netbooks de Conectar Igualdad? En III Congreso Internacional de Educación de la Universidad Nacional de La Pampa, Argentina.

▶ Cotik, V., & Monteverde, H. Evolución de la enseñanza de la informática y las TIC en la Escuela Media en Argentina en los últimos 35 años. Virtualidad, Educación y Ciencia, 7 (12), 11-33.

▶ Cuban, L. (1993). Computers meet classroom: Classroom wins. Teachers College Record, 95 (2), 185.

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

▶ Dutilh Novaes, C. (2011). The different ways in which logic is (said to be) formal. History and Philosophy of Logic, 32 (4), 303-332.

▶ Guzdial, M. (2008). Education Paving the way for computational thinking. Communications of the ACM, 51 (8), 25-27.

▶ Greco, M.B. (2012). La autoridad nuevamente pensada. En Emancipación, educación y autoridad. Prácticas de formación y transmisión democrática. Buenos Aires: Noveduc.

▶ Grover, S., & Pea, R. (2013). Computational thinking in K–12: A review of the state of the field. Educational Researcher, 42 (1), 38-43.

▶ Gutiérrez, G., Beltramino, L., & Viano, I. Políticas de Formación Docente en TIC: La experiencia desde un sindicato docente (2000-2015). Virtualidad, Educación y Ciencia, 6 (10), 24-37.

▶ Lago Martínez, S. (2015). De tecnologías digitales, educación formal y políticas públicas: Aportes al debate. Ed. Teseo.

▶ Lee, I., Martin, F., Denner, J., Coulter, B., Allan, W., Erickson, J., & Werner, L. (2011). Computational thinking for youth in practice. Acm Inroads, 2 (1), 32-37.

▶ Martínez López, P (2013). Las bases conceptuales de la Programación: Una nueva forma de aprender a programar. La Plata, Buenos Aires, Argentina.

▶ Martínez López, P.; Bonelli, E. y Sawady (2014) El nombre verdadero de la programación. Universidad Nacional de Quilmes. Recuperado de http://elaulayeltrabajo.proyectoslibres.unq.edu.ar/images/3/35/MartinezLopez-Bonelli-Sawady.pdf [10/02/2016]

▶ Meirieu, P. (1998) Frankestein educador. Barcelona: Laertes.

▶ Papert, S. (1980). Mindstorms: Children, computers, and powerful ideas. Basic Books, Inc..

▶ Papert, S. (1995). La máquina de los niños: replantearse la educación en la era de los ordenadores. Paidós.

▶ Resnick, M. (2008) Fallen in Love with Seymour's ideas. Presented at a Special Session of the 2008 American Educational Research Association Annual Meeting.

▶ Rusk, N. (2012). Entrevista a Natalie Rusk. En Eduteka. Disponible en: http://eduteka.icesi.edu.co/modulos/9/271/2086/1

▶ Simari, G. (2011). Los fundamentos computacionales como parte de las ciencias básicas en las terminales de la disciplina Informática. Bahía Blanca: Universidad Nacional del Sur. Recuperado de http://sedici.unlp.edu.ar/handle/10915/27579

▶ Soler, E. A., & Santacana, A. B. (2013). Innovative scaffolding: Understanding innovation as the disclosure of hidden affordances. Revista Iberoamericana de Argumentación, (7).

▶ Vygotski, L. S. (1999). Pensamiento y lenguaje: teoría del desarrollo cultural de las funciones psíquicas. Ediciones Fausto.

▶ William, D. (2006) Assessment for Learning. Why, What and How. Conferencia ofrecida en la Red de Evaluación en la Cambridge Univertiy. 15 de Septiembre de 2006.

▶ William, D (2016) Why teaching will never be research-based, Osiris conference, London, UK, January.

▶ Wing, J. M. (2006). Computational thinking. Communications of the ACM, 49 (3), 33-35.

▶ Zapata-Ros, M. (2015). Pensamiento computacional: Una nueva alfabetización digital. Revista de Educación a Distancia, (46).


**LISTADO DE PLATAFORMAS PARA ENSEÑAR A PROGRAMAR MENCIONADAS**

**Scratch: https://scratch.mit.edu/accounts/login/?next=/accounts/password_change/**
**Gobstone: http://www.gobstones.org/**
**Mumuki: https://mumuki.io/**
**UNC++Duino y Chatbot: http://umm.famaf.unc.edu.ar/?page_id=101**

## UEPC - Junta Ejecutiva Central

**Secretario General:**
MONSERRAT, Juan Bautista

**Secretaria General Adjunto:**
MIRETTI, Zulema del Carmen

**Secretario de Organización**
CRISTALLI, Roberto Orlando
Suplente: NIETO, Nicolás Gustavo

**Secretario de Coordinación Gremial**
RUIBAL, Oscar Ignacio David
Suplente: SOSA,Mario Nicolás

**Secretario Administrativo y de Actas**
LUDUEÑA, Carlos Fernando
Suplente: PERALTA, Luis Valentín

**Secretario de Finanzas**
GONELLA, Marcelo Luis
Suplente: SIMES, Juan Antonio

**Secretaria Gremial de Nivel Inicial y Primario**
FAUDA, Estela Maris
Suplente: VIDAL, Beatríz Elizabeth

**Secretario Gremial de Nivel Medio, Especial y Superior**
ZALAZAR, Daniel Armando
Suplente: ROJAS, Adriana

**Secretaria Gremial de Jurisdicción Privada**
CHAVES, Marcela Beatríz
Suplente: MATEO, Fernando Javier

**Secretaria de Asuntos Jubilatorios y Previsionales**
STRASORIER, Graciela
Suplente: GARZÓN, Mónica Beatríz

**Secretaria de Cultura y Educación**
CAVALLERO, Aurorita del Valle
Suplente: BAGGINI, Daniel

**Secretario de Prensa y Propaganda**
MAZZOLA, Fabián Leonardo
Suplente: FRONTROTH, Oscar Andrés

**Secretario de Acción Social**
ZAMMATARO, Hugo Daniel
Suplente: FONTANESI,Graciela Esther

**Secretaria de Derechos Humanos y Género**
MARCHETTI, Silvia Teresita
Suplente: ACOSTA, Héctor Manuel

**VOCALES**

**1º Vocal titular**
YEDRO, Viviana Luján
Suplente: GIACOMELLI, Ana Elizabeth

**2º Vocal titular**
ZAMORA, Lorena Fernanda
Suplente: GIACOMELLI, Carlos César

**3º Vocal titular**
RODRIGUEZ, Eduardo Omar
Suplente: BIANCO,Gabriela María

**4º Vocal titular**
SEDANO, María Monserrat
Suplente: GONZALEZ, Nélida Lucía

**5º Vocal titular**
STRASORIER, Ricardo Daniel
Suplente: MOYANO, María del Carmen

## Órgano de Fiscalización

**1º Miembro Titular**
FOSSATTI, Cleve Dominga
Suplente: CURIOTTO, Norma Dominga

**2º Miembro Titular**
CUESTA, Laura Esther
Suplente: LESCANO, Nestor Prioto

**3º Miembro Titular**
MIRADA, Beatríz Mercedes
Suplente: CORNATOSKY, Sergio Gustavo

Con esta publicación iniciamos desde el Área de Formación Docente del ICIEC-UEPC la serie "Cuadernos para la enseñanza". Donde compartiremos propuestas de enseñanza surgidas de los interrogantes, dilemas, desafíos e invenciones que miles de docentes realizan cotidianamente en las escuelas, para que sus estudiantes aprendan. Dichas propuestas son socializadas, analizadas y revisadas en los diferentes espacios de formación que ofrece el gremio, a través del Instituto de Capacitación e Investigación, desde hace más de 20 años.

En esta primera entrega **"Aprender a programar para integrar(nos)",** se realiza un recorrido histórico sobre la enseñanza de la computación en las escuelas, haciendo foco en el contexto actual. Se describen las características y potencialidades del pensamiento computacional y se presentan orientaciones para el trabajo en el aula.

¿Por qué empezar esta serie por la Programación? Porque promueve los proyectos colectivos, posibilita respetar los tiempos de aprendizaje de los/as estudiantes, habilita modos de pensar que implican un hacer y una creación singular. Programar permite integrar(nos), crear, inventar, acercar problemáticas cotidianas a la escuela, pensar posibles soluciones y potenciar e integrar los aprendizajes de otros campos de saberes escolares. Conocer el detrás de escena de los artefactos tecnológicos abre la posibilidad de contribuir desde la escuela, con el pasaje de ser consumidores/as a creadores/as de tecnologías.

UEPC Unión de Educadores de la Provincia de Córdoba

ICIEC Instituto de Capacitación e Investigación de los Educadores de Córdoba