Análisis y revisión de *softwares* educativos para el aprendizaje de la programación en entornos lúdicos

- Analysis and Review of Educational Software to Learn to Program in Playful Environments
- Análise e revisão de softwares educacionais para a aprendizagem da programação em ambientes lúdicos

Antonieta Kuz*
María Cecilia Ariste**

Resumen

El desarrollo acelerado de la sociedad de la información está suponiendo desafíos impensables para la educación y el aprendizaje. Hoy en día nos encontramos en una transformación social y cultural, lo que implica cambios de envergadura en el campo educativo a través del rol que juegan las tecnologías de información y la comunicación (TIC). Las TIC, en particular los programas computacionales con orientación lúdica para la enseñanza de la programación, son relevantes dado que tienen cuenta aspectos vinculados al entorno y a la ingeniería de *software* educativo. En este trabajo de investigación documental realizamos un análisis de diferentes recursos educativos como Scratch, Lightbot, PilasEngine, Pilas Bloques, Turtle Art y Blocky, dedicados a la enseñanza y aprendizaje de la programación como alternativas que permiten reforzar las competencias en programación a través de aplicaciones mediante la combinación de componentes pedagógicos, didácticos y lúdicos. Se considera que la recopilación y análisis planteados aquí, podrían resultar en un aporte de interés para los que deseen incluir actividades educativas mediadas por entornos de programación didácticos y lúdicos.

Palabras clave

programación; tecnología de la información; lenguaje de programación; *software* didáctico

Abstract

The accelerated development of the information society is posing unthinkable challenges for education and learning. Today we are in a social and cultural transformation on the verge of major changes in the educational field through the role played by Information and Communication Technologies (ICTs). ICTs, in particular computer programs with a playful orientation for teaching programming, are relevant since they consider aspects related to the educational

* Doctora en Ciencias de la Computación, ingeniera en Sistemas, profesora de la Universidad Metropolitana para la Educación y el Trabajo, Departamento de Informática, Universidad Metropolitana para la Educación y el Trabajo, Buenos Aires, Argentina. antokuz@esgcffaa.edu.ar. Orcid: https://orcid.org/0000-0002-8696-0859

** Ingeniera en Sistemas de Información, Senior Functional Analysis, docente, Fundación Sadosky, Buenos Aires, Argentina. mariste83@gmail.com. Orcid: https://orcid.org/0000-0001-7360-5543

Fecha de recepción: 29/12/2020
Fecha de aprobación: 31/08/2021

environment and educational software engineering. In this documentary research work, we carry out an analysis of different educational resources such as Scratch, Lightbot, PilasEngine, Pilas Bloques, Turtle Art and, Blocky dedicated to the teaching and learning of programming as alternatives that allow reinforcing programming skills through applications through the combination of pedagogical, didactic, playful components. It is considered that the compilation and analysis presented here could be a report and of interest to those who wish to include educational activities mediated by didactic and recreational programming environments.

Keywords

programming; information technology; programming language; didactic software

Resumo

O desenvolvimento acelerado da sociedade da informação apresenta desafios impensáveis para a educação e a aprendizagem. Hoje nos encontramos em uma transformação social e cultural, o que implica grandes mudanças no campo educacional por meio do papel das Tecnologias da Informação e da Comunicação (TICs). As TICs, em particular os programas de computador com uma orientação lúdica para o ensino da programação são relevantes, uma vez que levam em consideração aspectos relacionados ao ambientee à engenharia de software educacional. Neste trabalho de pesquisa documental realizamos uma análise de diferentes recursos educacionais como Scratch, Lightbot, PilasEngine, Pilas Bloques, Turtle Art e Blocky dedicados ao ensino e aprendizagem da programação como alternativas que permitem reforçar as competências de programação através de aplicações por meio da combinação de componentes pedagógicos, didáticos e lúdicos. Considera-se que a compilação e análise aqui apresentada pode resultar em uma contribuição de interesse para quem deseja incluir atividades educacionais mediadas por ambientes de programação didática e lúdica.

Palavras-chave

programação; tecnologia da informação; linguagem de programação; software didático

Introducción

Las nuevas tecnologías forman parte integral de la vida y es una certeza que ocuparán un lugar protagónico también en su futuro. En un mundo gobernado cada vez más por lo digital, la programación es un conocimiento esencial. La programación es uno de los nuevos desafíos que enfrentan los profesores en las aulas en los diferentes tramos de la escolarización, dado que programar no significa simplemente codificar en un lenguaje de programación, sino que también implica usar el pensamiento computacional para resolver problemas, en las diferentes asignaturas de la trayectoria escolar.

Además, este desafío para docentes implica pensar en cómo integrar las diferentes tecnologías en las propuestas educativas, pero no simplemente con la intención de fomentar su uso, sino para lograr que los estudiantes puedan apropiarse y valerse de ellas, integrándolas como herramientas eficaces para resolver problemas reales teniendo en cuenta que la tecnología y la informática están en un proceso de transformación continua, si pensamos en los diversos usos. Existe una gran variedad de plataformas y recursos tecnológicos creados para facilitar y mediar en los procesos de enseñanza-aprendizaje, los cuales tienen como propósito esencial contribuir a la formación integral de los alumnos, y a la adquisición de conocimientos, habilidades, comportamientos y valores, entre otros. Dentro del espectro de recursos educativos, un tipo particular es el *software* educativo, cuya finalidad didáctica es la enseñanza de la programación y fomentar el pensamiento computacional en entornos lúdicos, por medio de la generación de ambientes pedagógicos dinámicos.

El aprendizaje de la programación requiere de entender conceptos abstractos que no suelen ser sencillos de asimilar por la falta de referencias concretas que le permitan al alumno ser experimentados. Es por este motivo que se ha desarrollado una gran variedad de *softwares* educativos con fines didácticos, cuyo principal objetivo es facilitar el aprendizaje de los conceptos de programación, pero que involucran habilidades cognitivas orientadas a la solución de problemas y al pensamiento computacional. Muchos de estos programas se desarrollan en entornos predominantemente visuales con bloques, lo cual resulta una estrategia ampliamente utilizada para aprender programación y desarrollar el pensamiento computacional y la combinación de gráficos, animaciones, fotos y música, entre otros.

Conocer por medio de una descripción detallada las herramientas que acompañan la diversidad de iniciativas vinculadas a la enseñanza de la programación es de relevancia dado que las herramientas seleccionadas sirven para lograr un primer acercamiento y abordaje de la didáctica de la programación. Por este motivo en este trabajo haremos un análisis de las diferentes herramientas (construidas en diferentes plataformas y lenguajes) que son utilizadas como recursos para la enseñanza de la programación para el fortalecimiento de habilidades de pensamiento para la resolución de problemas y el desarrollo de actividades y contenidos. Las herramientas que analizaremos son: Pilas Engine, Pilas Bloques, LightBoth, Scratch, Turtle Art y Blocky.

El resto del artículo se estructura como sigue: en la sección 2, "Antecedentes", analizamos la educación y las habilidades para el siglo XXI. En la sección 3, detallamos el marco teórico y los aspectos que involucran la enseñanza de la programación y la noción de *software* educativo. En la sección 4 detallamos la metodología involucrada. En la 5, "Resultados y análisis", analizamos y revisamos las herramientas seleccionadas. En el apartado 6 se presentan las conclusiones.

Antecedentes

El siglo XXI ha estado marcado por la revolución tecnológica, nuevos modelos económicos y sociales. La vertiginosa evolución de la tecnología, principalmente impulsada por la electrónica, implica que hoy las TIC estén en todos los campos del conocimiento y en lo cotidiano de la sociedad. Un factor muy importante es la digitalización de la economía y la transformación del trabajo, que se volverá cada vez más significativa. La digitalización, por un lado, favorece la aparición de nuevas profesiones y la creación de nuevos puestos de trabajo; por otro lado, da lugar a la modificación de algunas profesiones o cargos existentes, al cambiar los métodos, la forma en que se llevan a cabo las tareas, los requisitos del trabajo y, como consecuencia, las habilidades necesarias para llevarlo a cabo. En la era digital, prácticamente todas las actividades de la vida cotidiana se encuentran mediadas por la tecnología, con lo cual quedarse al margen de su dominio podría ser equivalente a llegar a ser analfabeto digital. El concepto de analfabetismo ha tenido una irremediable transformación con el tiempo. Muchas veces se ha identificado y relacionado al analfabetismo digital con el uso de las tecnologías y dispositivos de forma instrumental, pero es posible ampliar este concepto asociándolo con el desarrollo de competencias tecnológicas que implican que al entrar al mundo de la tecnología también entramos en lo desconocido y buscar una forma entender el nuevo paradigma digital (George Reyes y Avello-Martínez, 2021).

Lograr alcanzar las competencias y actitudes para aprender de forma autónoma a lo largo de la vida conlleva que el aprendizaje sea un proceso, no un producto final; esto implica que el alumno sea capaz de continuar aprendiendo y de hacer conscientes las propias formas de aprender. Aprender involucra abordar lo desconocido, moverse a través de lo que probablemente nadie sabe o entiende; una vorágine que lleva a adquirir permanentemente nuevos aprendizajes, con lo cual es de vital importancia disponer de mecanismos que lo faciliten, una capacidad de conocer, organizar y autorregular el propio proceso de aprendizaje. Por lo tanto, aprender involucra, por un lado, desarrollar la metaatención y el tomar conciencia de los propios procesos para atender a lo importante, y por otro lado, la metamemoria, que es la conciencia de los propios procesos para captar y recordar la información, y trasladar lo aprendido más allá del ámbito académico, al contexto personal y laboral.

Marco teórico

La informática y las ciencias de la computación son disciplinas académicas con su propio cuerpo de conocimiento. El *pensamiento computacional* se encuentra dentro de las bases conceptuales de la computación. La International Society for Technology in Education (ISTE, http://iste.org) y la Computer Science Teacher Association (CSTA, http//csteachers.org) entienden el pensamiento computacional como

un proceso de resolución de problemas que incluye (pero no se limita a) la formulación de problemas; la recolección y el análisis de datos; la representación de datos por medio de abstracciones, como modelos y simulaciones; soluciones automatizadas a través del pensamiento algorítmico; identificación, análisis e implementación de soluciones posibles con el objetivo de alcanzar la combinación más eficiente y eficaz de recursos y pasos; y la generalización y transferencia de este proceso de resolución de problemas a una gran variedad contextos (CSTA e ISTE, 2011). Abordar en forma práctica la didáctica del pensamiento computacional es uno de los mayores desafíos desde una perspectiva integrada a la realidad curricular, aplicado en todas sus dimensiones. Una dimensión es la capacidad de programar, esto es, diseñar algoritmos y definir el código que lo lleva a la práctica en lenguaje de computador (véase la figura 1). El pensamiento computacional y la programación proveen al estudiante habilidades para el desarrollo del saber hacer, necesarios para comprender, analizar críticamente y actuar en un espacio fuertemente influenciado por las tecnologías digitales.

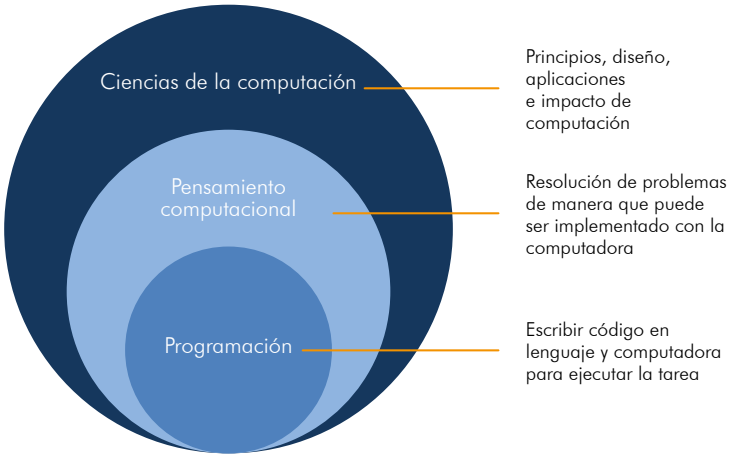

Figura 1. Relación entre los conceptos principales de la computación.

Fuente: tomado de Hara y Hepp (2016).

Basogain Olabe *et al*. (2015) consideran cómo con ejemplos se pueden desarrollar los elementos fundamentales del pensamiento computacional utilizando un lenguaje de programación e integrado en el aula por medio del diseño y la implementación de proyectos de programación. Autores como Eckerdal *et al*. (2005) expresan que saber programar significa: "entender algunos lenguajes de programación, y utilizarlos para escribir códigos de programas, adquirir una forma de pensar que se alinea con los lenguajes de programación", sin embargo, no es obligatorio dominar más de un lenguaje de programación. Según Monjelat *et al*. (2018), "la programación es una instancia que permite elaborar y utilizar contenidos digitales, resulta fundamental eliminar las barreras de acceso a estas herramientas". Desde esta perspectiva, programar implica aprender cómo se debe pensar para poder expresarse a través de la lógica y las reglas de cada lenguaje de programación. No solo contempla el conocimiento en lenguajes de programación, sino que también involucra habilidades cognitivas orientadas a la solución de problemas.

Aprender a programar en lenguajes de alto nivel es una tarea que involucra conocer la sintaxis del lenguaje que se esté usando y el manejo de conceptos y conocimientos previos para obtener un programa relativamente sencillo y que produzca los resultados esperables. Frente al reto que representa empezar a programar utilizando lenguajes de alto nivel de uso profesional, se han desarrollado plataformas que permiten enseñar y abordar la temática lúdicamente para los alumnos. Estos recursos digitales pueden facilitar y ampliar las posibilidades de aprendizaje de manera intuitiva, aunque esto, además de la integración de tecnología, requiere de prácticas innovadoras que construyan un nuevo modelo educativo, en todas las edades y niveles de enseñanza. Un ejemplo es la Hora del Código (https://hourofcode.com/es), un portal que busca desmitificar el *código*, demostrar que todos pueden aprender los conceptos básicos y ampliar la participación en el campo de las ciencias de la computación.

Que un estudiante aprenda a programar implica que puede adquirir un pensamiento que incluya la lógica para abordar problemas. Muchos conocimientos hoy en día no logran ser transferidos en forma práctica a las aulas. Esto puede obedecer a múltiples factores, entre ellos la dificultad en la construcción de un dispositivo didáctico-pedagógico para intentar resolver el dilema que se presenta a los docentes para que sus estudiantes realicen las actividades vinculadas al aprendizaje en las materias de programación. Por lo general, las instituciones educativas plantean esquemas de enseñanza donde el docente es el que imparte la mayoría de los conocimientos por medio de clases magistrales, y los estudiantes los absorben mediante el desarrollo de prácticas para luego demostrarlos en evaluaciones, mayormente escritas, y pruebas orales.

Una manera de comenzar a programar es aprender usando entornos lúdicos, que permiten a los alumnos asimilar conceptos, ideas básicas y flujos de trabajo de manera más atrayente. El juego puede ser un mediador fundamental en el desarrollo emocional y social, que ocupa un lugar importante en el alumno en la medida en que potencia la interacción con sus pares, favorece el interés por lo que hacen sus compañeros, ayuda a expresar y comprender emociones, respetando el ritmo de aprendizaje y el nivel de madurez de cada alumno. Además, es una actividad propia del ser humano, que está relacionada con el entretenimiento. Según Melo Herrera y Hernández Barbosa

> […] el juego, como proceso de asimilación, permite dar significado a las cosas a partir de las relaciones que se establecen con él […] se debe pensar en el juego como un recurso que permite construir conocimiento no sólo en una dirección y para un solo sujeto, dado que brinda la posibilidad de aprender de manera distinta y en diferentes sentidos: las simples muecas que le hace el abuelo a su nieto para que lo imite generan mayores posibilidades de aprendizaje recíproco. En este tipo de aprendizaje el educador se ha limitado; su papel se ha centrado simplemente en transmitir conocimientos y no en devenir un sujeto facilitador del desarrollo cognitivo. Es difícil propiciar un espacio de transformación en todos los sentidos si limitamos las acciones, los pensamientos y los sentimientos. (2014)

El *software* educativo se ha constituido como tema de investigación de diversos trabajos que abordan el concepto de este tipo de *software*, sus características y potencialidades. Incluye todas aquellas herramientas mediadoras del proceso de enseñanza-aprendizaje utilizadas por docentes y alumnos, que contribuyen a la participación activa, tanto individual como colectiva, y se caracterizan por ser interactivos mediante la aplicación de recursos multimedia tales como sonidos y juegos que apoyan las funciones de evaluación y diagnóstico. Esto constituye una valiosa fuente para aprender y adquirir conocimientos.

Según Marqués (1999), el *software* educativo son todos "aquellos programas creados con fines didácticos". Según Sánchez (1999), el *software* educativo es como cualquier programa computacional cuyas características estructurales y funcionales sirvan de apoyo al proceso de enseñar, aprender y administrar. Por su parte, Rodríguez Lamas (2000) considera que es una aplicación informática que, incluida en una estrategia pedagógica definida, apoya directamente el proceso de enseñanza-aprendizaje y constituye un efectivo instrumento para el desarrollo educacional del hombre del próximo siglo. Labañino Rizzo (2000) lo define como una aplicación informática concebida especialmente como medio, integrada al proceso de enseñanza-aprendizaje. Autores como Almaguel Guerra *et al*. (2016) mencionan que las teorías pedagógicas han creado muchas formas de mejorar el proceso de enseñanza-aprendizaje mediante la inclusión de la tecnología que permite crear nuevos ambientes lúdicos de aprendizaje tanto para estudiantes como para profesores. Debido a las características de los juegos de computadora, estos pueden ser aprovechados para mejorar el proceso de enseñanza y aprendizaje, ya que lo hace lúdico y amigable, de esta manera se estimulan la creatividad, la imaginación y el autoaprendizaje tanto del docente como del alumno. Los autores concuerdan en que el *software* educativo debe tener fines didácticos, para ser utilizado con una computadora o dispositivo digital en los procesos de enseñar y aprender. Las definiciones aportadas por los autores mencionados coinciden en el carácter instrumental que tiene el *software* educativo en el proceso de enseñanza-aprendizaje, esto hace que la definición sea abarcadora e inclusiva. Existe un amplio abanico de *softwares* educativos en entornos lúdicos; dentro de estos encontramos aquellos que tienen el fin de enseñar a programar en ambientes educativos. Estos programas están diseñados para lograr que los estudiantes tengan plataformas simples de programación, que no llegan a ser entornos profesionales.

El juego digital por medio del *software* educativo y didáctico se ha implementado como recurso en diferentes campos del saber, lo que ha dado lugar a una gran cantidad y variedad de aportes. La implementación de juegos digitales para la enseñanza de la programación permite entender y ver el juego como un factor crucial para el mejoramiento de la calidad de la educación, debido a los resultados y al cambio de actitud observado en el educando frente a la frustración que muchas veces genera aprender la lógica de la programación. Desde la pedagogía de la enseñanza de la programación, el juego contribuye a mejorar significativamente sus procesos, a alcanzar los objetivos trazados para una actividad o programa, y en la construcción social del conocimiento determinado por aspectos tales como el cognitivo, el afectivo y el comunicativo.

Muchas veces la forma de utilizar el material didáctico y su incorporación en el aula dependen de dos factores. En primer lugar está la concepción que tiene el profesor acerca de la enseñanza de la programación, así como las prestaciones asociadas a la usabilidad del *software*. Lo segundo está estrechamente

vinculado a la importancia de la interfaz de usuario de la aplicación, al tratarse del medio por el cual el usuario visualiza y por el que accede a las prestaciones y servicios. Así, la funcionalidad del *software* educativo es el conjunto de características que hacen que este sea práctico y utilitario.

Ahora bien, para poder evaluar un *software* educativo es necesario tener en cuenta aspectos asociados a las características y a la valoración que corresponden a la dimensión tanto pedagógica tanto como a la técnica. La primera debe contemplar las condiciones del *software* relativas a los destinatarios, como la edad o el nivel educativo; la enseñanza, que involucra aspectos como la motivación, retroalimentación y la metodología usada, en cuanto a los tipos de contenidos y las estrategias y el aprendizaje que promueve la aplicación. A su vez, desde la dimensión técnica hay que tener en cuenta las características de la interface, que involucran, entre otras, el diseño de pantallas, la disposición de menús, iconos, imágenes, color, gráficos y animaciones; y la usabilidad, que incluye la facilidad de utilizar el recurso, el acceso al programa, la instalación, operación y las formas de navegación, entre otras características.

El *software* educativo diseñado para apoyar el proceso de enseñanza y aprendizaje de la programación tiene un alto aporte de desarrollo técnico mediante la construcción de interfaces atractivas; una de sus principales ventajas es permitir la interactividad entre los estudiantes, retroalimentándolos y evaluando lo aprendido. Este tipo de *softwares* están apoyados en la teoría constructivista (Sesento García, 2021) como un proceso que permite construir el conocimiento, pensado como una interacción dialéctica entre los conocimientos del docente y los del estudiante, que entran en discusión, oposición y diálogo, para llevar a una síntesis productiva y significativa: el aprendizaje. Sobre el aprendizaje de la programación encontramos el trabajo realizado por Espíndola *et al*. (2018), en el cual plantean el desarrollo de un marco de referencia para evaluar las características de las herramientas visuales o lúdicas utilizadas en la enseñanza inicial de la programación, y orientar a los docentes hacia la selección de las herramientas más adecuadas para incorporar en sus propuestas docentes, en función de los objetivos específicos de aprendizaje, el nivel educativo de los niños o jóvenes y las expectativas del docente.

Metodología

Se llevó a cabo una investigación documental cualitativa en la que se hizo un análisis y una síntesis de las características de los lenguajes de programación en entornos lúdicos mediante la recolección, selección y evaluación de información no estandarizada de diversas fuentes bibliográficas. En este análisis presentaremos los lenguajes Pilas Engine, Pilas Bloques, LightBoth, Scratch, Turtle Art y Blocky, los cuales son recursos digitales a los que los educadores pueden acceder para enseñar a programar basados en una concepción constructivista de la enseñanza y del aprendizaje, desde la interactividad, la colaboración y

la construcción y reconstrucción como dinámica. Además, en este tipo de entornos el estudiante puede familiarizarse rápidamente con los contenidos, debido a que son lúdicos (Sáez López, 2019) y no llegan a ser lenguajes de programación profesionales de alto nivel; esto permite ofrecer a los alumnos recursos que activan sus capacidades de construir el conocimiento (Papert, 1987).

Resultados y análisis

Por un lado, encontramos que Pilas Engine (http://pilas-engine.com.ar/) es una plataforma orientada a programar haciendo videojuegos de manera simple y que permite consolidar el aprendizaje de la programación. Mantenida por una comunidad de desarrolladores, cuenta con una colección de actores, objetos y escenas, y se codifica a través de Python, como se observa en la figura 2. En la investigación de Graziani *et al*. (2016) se presenta a Pilas Engine como recurso pedagógico y como una alternativa que se puede usar para aprender a programar, desde una fundamentación en el constructivismo. Analizan y proponen las claves de un análisis que busca ver en qué medida se logra un acercamiento sencillo y motivante a la programación de aquellos que no tienen un experiencia previa, frente a la dificultad que representa empezar a aprender a programar utilizando un lenguaje de alto nivel de uso profesional. También valoran las ventajas de realizar prácticas de programación en computadora por sobre el uso exclusivo de papel.

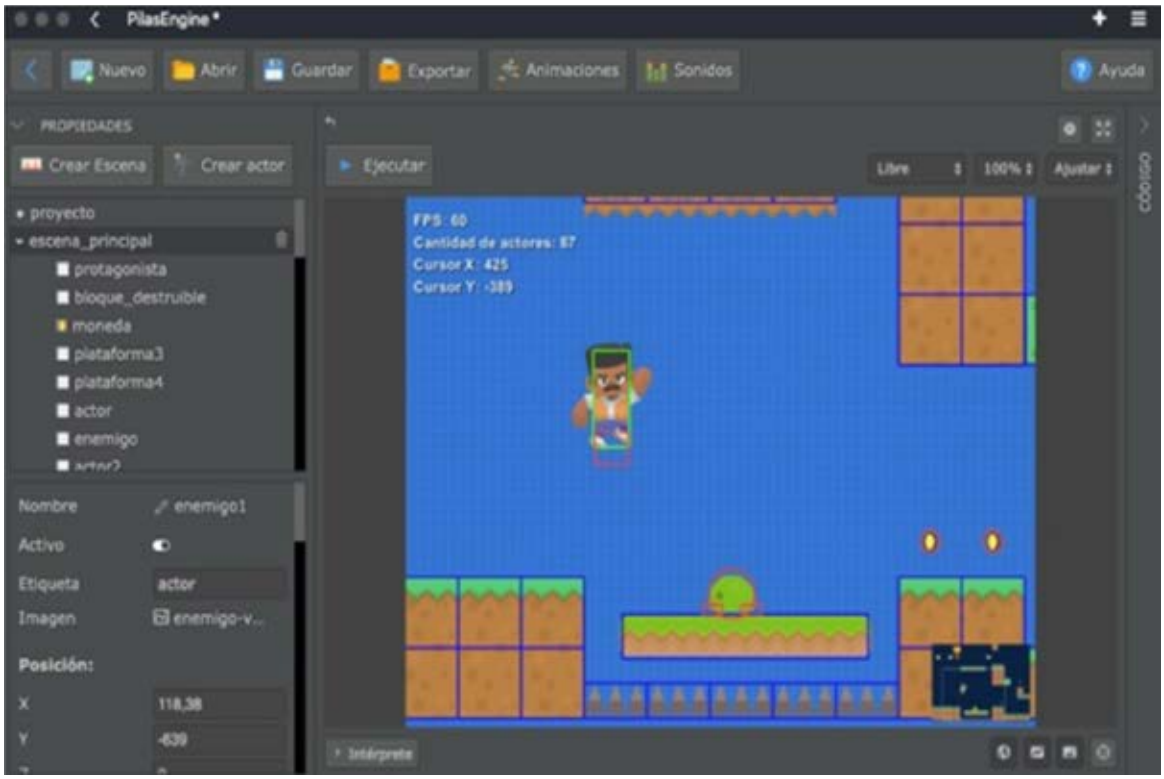

Figura 2. Interfaz gráfica de Pilas Engine.

Fuente: elaboración propia.

PilasBloques (http://pilasbloques.program.ar/) es un *software* educativo creado por la Fundación Sadosky (http://www.fundacionsadosky.org.ar/) en Argentina. Fue especialmente desarrollado para la enseñanza de la programación, mediante bloques para dar instrucciones. Se puede usar acompañado con distintas secuencias didácticas planteadas por el docente y está recomendado para niños de diez años (Ahumada *et al*., 2018). Su particularidad es que se compone de distintos desafíos, cada uno de los cuales se relaciona con un concepto del pensamiento computacional completando secuencias que se le proponen. Además, incluyen personajes animados que se repiten en los distintos desafíos (véase la figura 3). Podemos ver un ejemplo en la investigación de Sanzo *et al*. (2017), donde presentan a Pilas Bloques como una propuesta y estrategia pedagógica basada en la indagación, la descomposición, desafíos breves y enfocados, y la construcción de abstracciones.

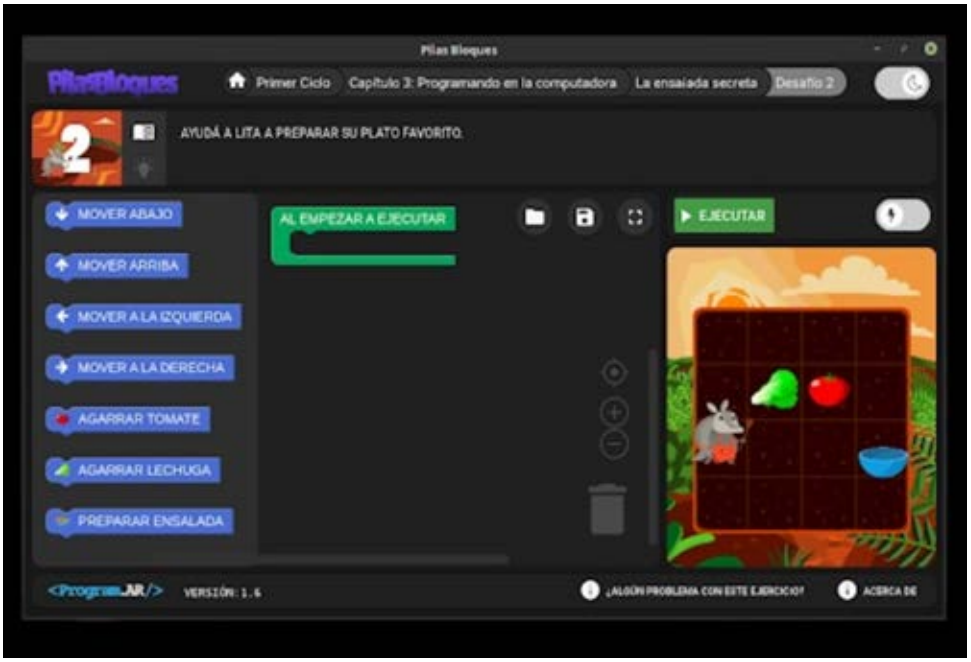

Figura 3. Entorno de Pilas Bloques

Fuente: elaboración propia.

Yaroslavski creó en el 2014 LightBot (http://lightbot.com/hour-of-code.html), una herramienta que busca la resolución de *puzzles* mediante la lógica de la programación. Tiene desafíos con juegos de ingenio cuyo protagonista es un robot que mediante instrucciones debe llegar a un determinado destino (González *et al*., 2017). Está pensado para niños de nueve años de edad en adelante. La herramienta divide los conceptos en prácticas de programación y flujos de control, su particularidad es la utilización de íconos y símbolos en lugar de palabras o bloques. Como puede verse en la figura 4, se busca que el usuario pueda

enfocarse en los pasos de una secuencia. En el trabajo de Aedo López *et al.* (2016) se describe una experiencia de uso de LightBot presentando a los estudiantes conceptos claves que no son fáciles de enseñar ni de aprender, como abstracción, función y reutilización.

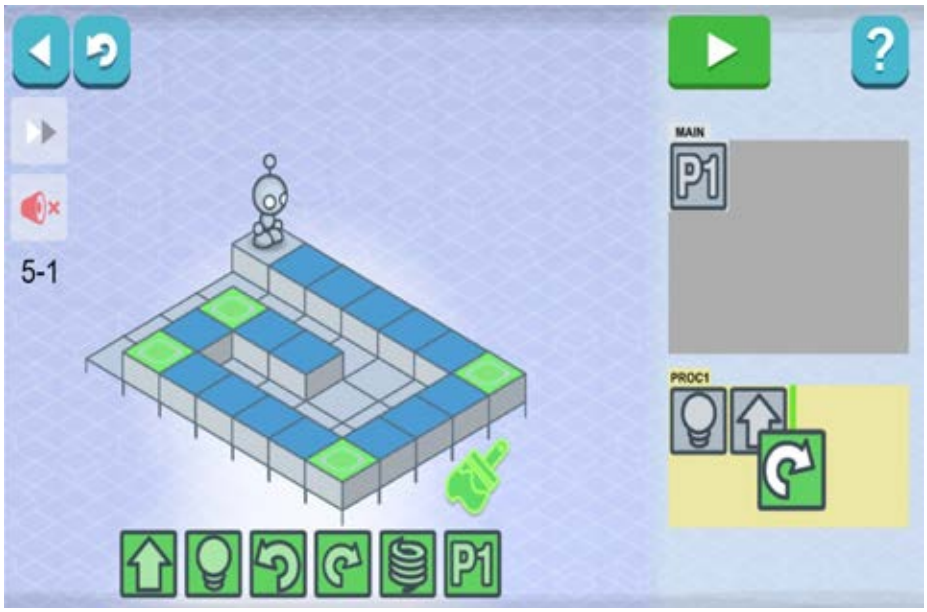

Figura 4. Interfaz de LightBot.

Fuente: elaboración propia.

Los costos del pago de licencias dan lugar a que muchas veces los *softwares* educativos no estén al alcance de los alumnos más desfavorecidos; surge entonces la incorporación de Linux y el *software* y proyectos de código abierto y gratuitos para la enseñanza. Alrededor del año 2006, con el objetivo de brindar oportunidades educativas a los niños en situaciones más desfavorecidas, Negroponte, Lou Jepsen y Bender fundaron la asociación One Laptop for Child (OLPC) y construyeron la *laptop* OLPC XO, económica, con contenidos y *software* diseñados para un aprendizaje colaborativo, entretenido y autónomo bajo el sistema operativo Linux. Para poder informatizar a los niños con menos recursos con las computadoras OLPC XO, desarrollaron el *software* educativo Sugar (Bender, 2017). Sugar http://wiki.sugarlabs.org/go/Sugar_on_a_Stick es un *software* libre y un entorno gráfico bajo distribución Linux Fedora desarrollado en Python. Es fácil de personalizar, de modificar el código fuente y aplicar los cambios a las actividades al instante. A pesar de que Sugar (véase la figura 5) fue diseñado para XO, puede utilizarse en cualquier computadora, descargándolo y ejecutándolo desde una USB o un CD, independientemente del sistema operativo instalado. Sugar es un entorno que brinda varias opciones para la programación, como Turtle Arts (http://www.turtleart.org)/ (figura 6), el cual es un entorno de programación gráfico basado en el lenguaje LOGO, que consta de una pequeña tortuga (cursor gráfico) que debe ser programada para que tenga comportamientos, desplazamientos, y también dibujar pintar y realizar diferentes diseños en la pantalla. Por un lado, cuenta con bloques que ordenan a la tortuga dibujar

líneas y arcos de diferentes colores, o dirigirse a un lugar específico de la pantalla; por otro, cuenta con bloques que le permiten repetir secuencias y realizar operaciones lógicas.

Figura 5. Interfaz de Sugar.

Fuente: elaboración propia.

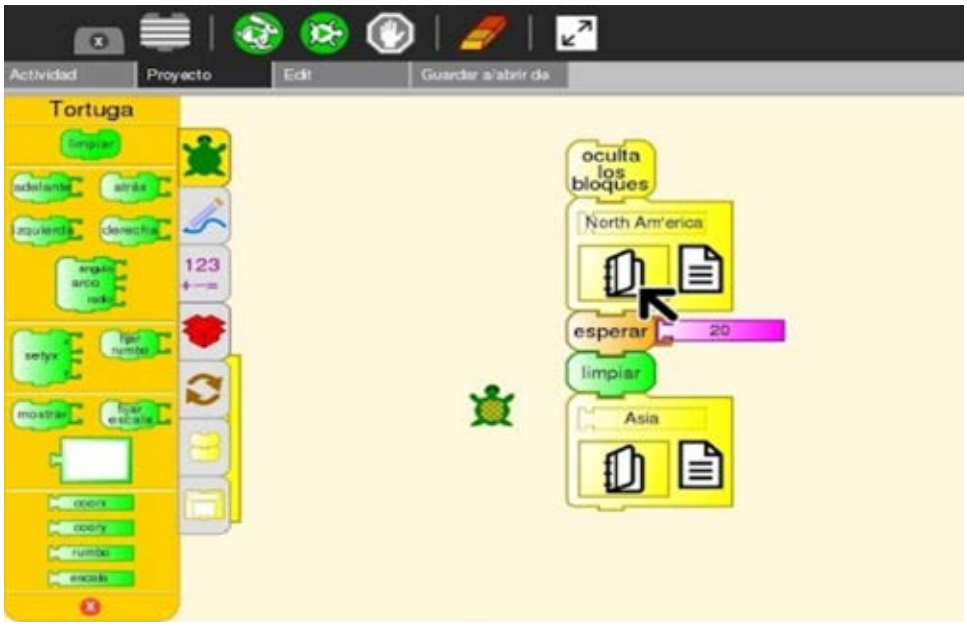

Figura 6. Interfaz de Turtle Arts.

Fuente: elaboración propia.

También encontramos Blocky (https://developers.google.com/blockly/), una herramienta de programación visual compuesta por un conjunto de instrucciones en forma de bloques gráficos que se pueden combinar, para crear programas de diferente complejidad (véase la figura 7). Estos bloques representan conceptos de código tales como variables, expresiones lógicas, bucles, entre otras. Blocky es Open Source y tiene licencia Apache License 2.0. Además, puede ser usado por desarrolladores, para efectuar nuevos formatos o adaptaciones, y educadores. Podemos mencionar que es capaz de exportar código a varios lenguajes, como JavaScript, Python, PHP, Lua, Dard, xml. También se pueden personalizar los bloques y el editor. Por ejemplo, Visualino, entorno de programación visual para Arduino, está basado en Google Blockly. Además, cuenta con la funcionalidad para traducir automáticamente los bloques a uno de estos lenguajes, tales como JavaScript, Python y PHP, entre otros.

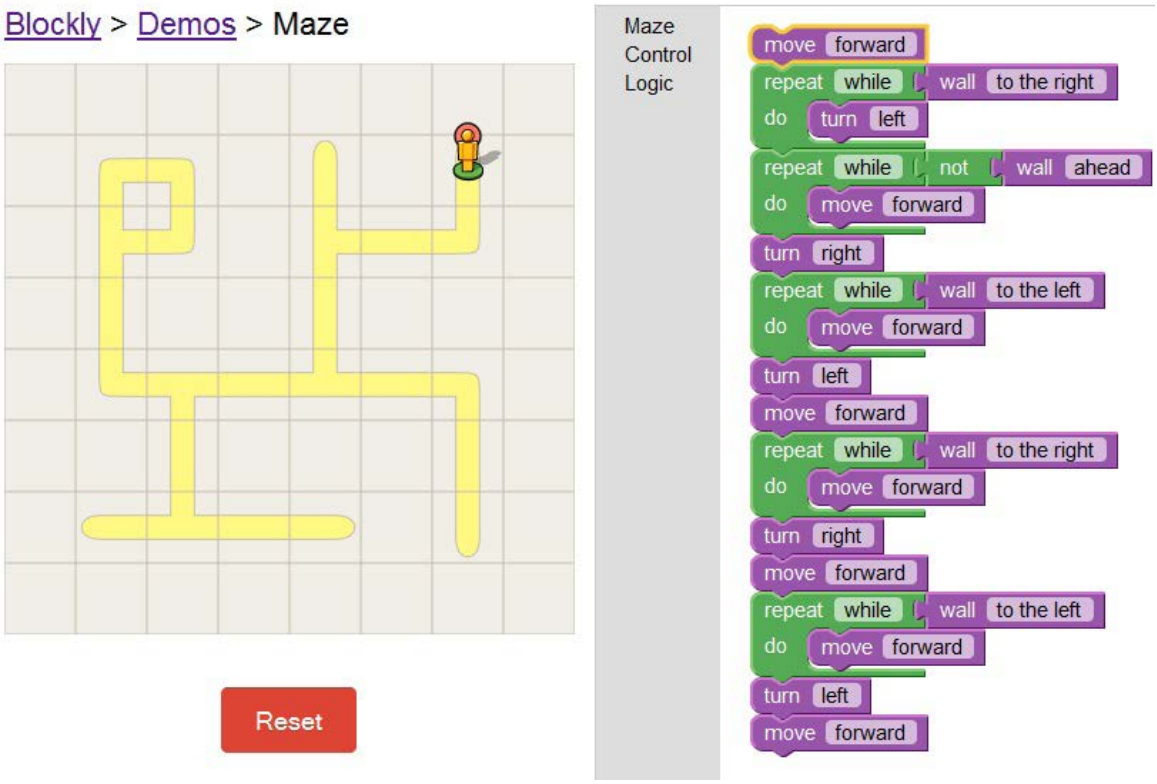

Figura 7. Interfaz de Blocky

Fuente: elaboración propia.

Finalmente tenemos en cuenta en nuestro análisis a Scratch (https://scratch.mit.edu/). El proyecto para su desarrollo nació en el 2003, de la mano del grupo de investigación Lifelong Kindergarten del MIT Media Lab, liderado por Mitch Resnick en colaboración con un grupo de la UCLA. El fin didáctico de Scratch es enseñar a programar; es posiblemente la herramienta de aprendizaje de programación más extendida en el mundo (Almeida Santos Silva y Bigotte de Almeida, 2017). Scratch ha sido desarrollado en torno a varios conceptos, como la facilidad

de uso y el trabajo colaborativo, ya que todos los trabajos pueden ser alojados en la web de Scratch para la visualización de su código, lo cual permite su ampliación o modificación por cualquier otro usuario. Esta herramienta ahorra la escritura de código que puede inducir a errores, reemplazándola por el arrastre de bloques que se conectan entre sí como un puzle, lo que facilita su uso y comprensión (Raj Sidhu, 2019). A lo largo del tiempo, este *software* ha ido evolucionando a través de las diferentes versiones y ha sido desarrollado en diferentes lenguajes (como C, ActionScript y JavaScript). Es ampliamente usado por los docentes debido a que es un proyecto de desarrollo cerrado pero de código abierto, y la programación está dirigida por eventos. Esta herramienta de programación educativa, al igual que las otras mencionadas, convierte un lenguaje de programación en una aplicación visual para resolver un problema, lo que permite programar utilizando bloques prediseñados con una acción o propiedad en particular, arrastrando estos bloques de código hacia el área de trabajo para formar bloques de acciones más complejas (véase la figura 8). En la investigación que realizan Jiménez *et al*. (2018), sugieren que los entornos de programación basados en bloques han tenido un impacto positivo en la retención, efectividad, eficiencia, compromiso, actitudes y percepciones de los estudiantes hacia la informática, pero también en la usabilidad. En esta investigación se analiza el impacto de la usabilidad en Scratch, dentro de entornos de programación basados en bloques, desde la navegación como un factor más relevante a la hora de mantener y guiar de forma eficaz, con fluidez y claridad a los usuarios de Scratch.

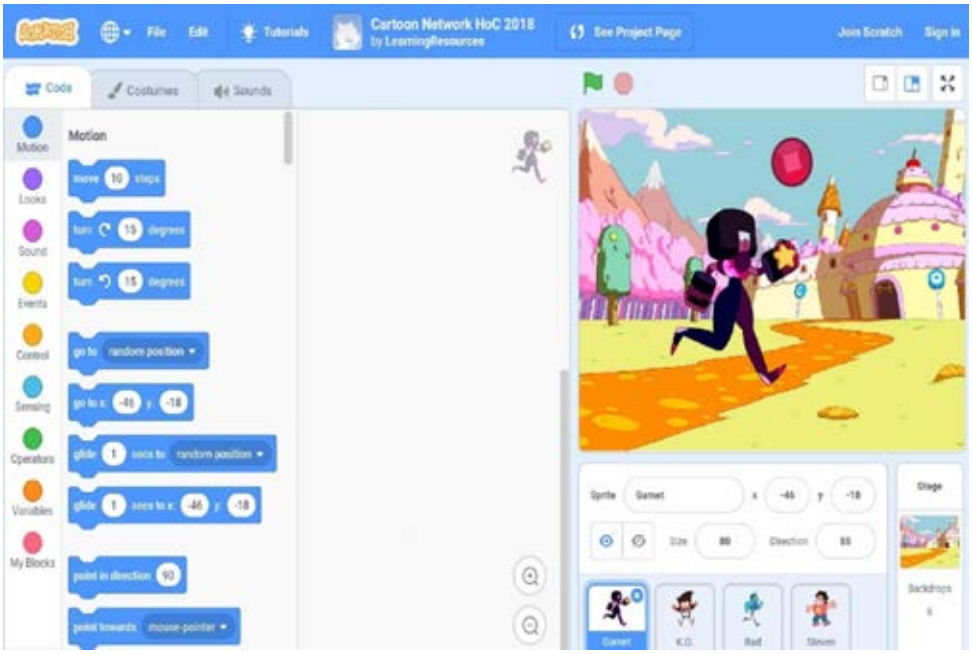

Figura 8. Interfaz de Scratch.

Fuente: elaboración propia.

Profundizando sobre Scratch, es muy importante como herramienta de aprendizaje programación ya que permite crear programas y compartirlos. Además, al ser un *software* muy dinámico se pueden trabajar estructuras de control lógicas, como la secuencia, selección o condicionales e iteración o ciclos y se pueden combinar dichas estructuras de programación para armar programas que brinden una solución a diversos planteos.

Del relevamiento realizado y del análisis de los lenguajes con fines didácticos estudiados se deprenden las tablas 1 y 2, que presentamos a continuación.

Tabla 1. Análisis comparativo entre softwares estudiados

Software	Versión	Ficha técnica resumida
Pilas Engine	Versión actual: v2.6.10. Es posible participar como programador del proyecto o enviar correcciones	Objetivo principal: construir videojuegos de manera sencilla
Pilas Bloques	Versión actual: V 1.7.8. Se puede usar bajo sistemas operativos Windows, Linux y Mac.	El objetivo principal es aprender a programar. Es una aplicación desarrollada especialmente para el aula, en la cual se proponen desafíos con diversos niveles de dificultad para acercar a las y los estudiantes al mundo de la programación por medio de bloques.
LightBot	LightBot está disponible como un juego bajo Flash en línea y una aplicación para teléfonos móviles con Android e iOS. LightBot ha sido construido con Flash y OpenFL	Es un videojuego educativo para aprender conceptos de programación de *software*
Turtle Art	*Software* libre, disponible para Mac/ Windows, Paid app en la iTunes store	Objetivo: lograr que los estudiantes usen el razonamiento matemático, la resolución de problemas, el conteo, la medición, la geometría y la programación de computadoras para crear hermosas imágenes
Scratch	Versión 3.0 de Scratch publicada en el 2018. Presenta variadas novedades, pero se sigue manteniendo la esencia de Scratch 2.0. La nueva interfaz de Scratch 3.0 ha abandonado Adobe Flash para basarse en HTML5.	Objetivo: programación en un entorno visual
Blocky	Es *software* libre y de código abierto, con licencia de Apache 2.0.	Blockly Games es una serie de juegos educativos que enseñan programación. Está diseñado para niños que no han tenido experiencia previa con la programación de computadoras. Al final de estos juegos, los jugadores están listos para usar lenguajes convencionales basados en texto.

Fuente: elaboración propia.

Tabla 2. Ventajas y desventajas generales de los programas relevados

Análisis de lenguajes de programación con fines didácticos	
Ventajas	• Permiten a los alumnos concentrarse en la lógica de la programación abstrayéndose de la gramática del propio lenguaje. • Permiten aplicar principios de programación sin tener que preocuparse por la sintaxis de un lenguaje de programación tradicional o tener que enfrentarse a la intimidación de un lenguaje de programación de uso profesional. • Buscan que las instrucciones sean a nivel visual, mediante bloques ensamblables, que tienen diferentes instrucciones, de manera que programar se reduce a seleccionar y ensamblar ordenadamente las instrucciones para ejecutar. • Son entornos lúdicos ya que incluyen personajes, sonido e interacciones que permiten generar programas construidos con elementos de programación básica estructurada • Favorecen el aprendizaje de los alumnos dado que se fundamentan en el aprendizaje colaborativo, el pensamiento crítico, el desarrollo de las cualidades creativas, en visualizar, expresar, explorar y comprender. • Poseen interfaces simples, con lo cual la interacción los hace accesibles, y está orientada a la rápida comprensión de los alumnos. Asimismo, los entornos son motivadores, orientados al aprendizaje a través del juego. • Favorecen el pensamiento crítico en el contexto de la exploración y el descubrimiento y la independencia del nivel de destreza de los alumnos, tanto a nivel de comprensión lectora como en experiencia informática. • Permiten a los alumnos formular problemas simples y construir estrategias para su resolución, incluyendo su descomposición en pequeñas partes, utilizando secuencias ordenadas de instrucciones, valiéndose de la creatividad y experimentando con el error como parte del proceso. • Favorecen el aprendizaje gracias a sus prestaciones, ya que existen una gran variedad de herramientas muy completas. Es importante tener en cuenta que este tipo de herramientas permite abordar el pensamiento computacional y la programación desde un enfoque más amplio, interactivo y lúdico.
Desventajas	• Deben ser utilizados según las edades y la didáctica por aplicar según el contexto. • Es importante tener presente el diseño de narrativas de las actividades didácticas y adecuarlas para que combinen con el lenguaje y el medio digital y construir el conocimiento en un marco lúdico y creativo. • Es reducida la cantidad de sentencias: si bien en la programación visual con bloques el código se crea arrastrando bloques y encajándolos en la zona de codificación, no es necesario escribir las sentencias, solo se requiere mover los bloques. Entonces, tanto el número de categorías de bloques de programación como el número de bloques dentro de cada categoría son reducidos, de manera que solo se conservan los más básicos de la sintaxis de cualquier lenguaje de programación. • Para comprender mejor la importancia de cada material didáctico, es necesario tener en cuenta las prestaciones que proveen para comprender si se adaptan a las necesidades, expectativas y recursos. Esta valoración y percepción depende del docente y de la aplicación práctica que desee hacer, así como del contexto y los recursos tecnológicos con los que se cuente.

Fuente: elaboración propia.

Conclusiones

Inicialmente, el campo de la programación era exclusivo de profesionales, matemáticos y físicos, luego de los informáticos; hoy en día, la sociedad ha ido cambiando, la programación forma parte del presente y probablemente lo

seguirá siendo en el futuro. Es una actividad que involucra el uso de recursos y herramientas disponibles para crear programas que resuelvan problemas. A lo largo del tiempo se han desarrollado y evolucionado los lenguajes de programación y han surgido aquellos basados en bloques, en entornos lúdicos muy intuitivos, fáciles de utilizar y de entender. Los diferentes entornos que hemos considerado en el análisis son en sí mismos herramientas o medios que brindan estrategias y herramientas pedagógicas que fortalecen las habilidades cognitivas y de resolución de problemas en los estudiantes y que el docente podrá seleccionar en función del contexto educativo. Sin embargo, existe un gran desafío a la hora de seleccionar las herramientas, debido a la diversidad de lenguajes y de herramientas puesto que estamos en un contexto cambiante en donde hay nuevas tendencias, nuevos entornos de desarrollo, pero también diferentes necesidades curriculares. Ante el reto que representa aprender a programar, es importante consignar las herramientas de *software* específicas que se están utilizando actualmente para la programación, en las asignaturas de los currículos escolares y así conocer el estado de la cuestión. Esto contribuye a conocer más sobre las diferentes propuestas y el uso de las tecnologías de la información para el desarrollo de actividades que ayuden a los docentes —no solo de programación, sino de otras áreas, como ciencias, arte (dibujo, teatro, cine), educación física, prácticas del lenguaje y matemáticas— a preparar a los alumnos para que entiendan un mundo cambiante. De esta forma se logra integrar la programación a la alfabetización digital.

El *software* educativo diseñado para que los alumnos programen puede ser un medio para establecer un contacto entre la tecnología y el área de educación. El propósito de esta investigación es analizar y realizar una revisión de propuestas que tienen como estrategia el aprendizaje de programación como herramientas pedagógicas que apoyen en la mejora de las habilidades cognitivas y de resolución de problemas en alumnos que se encuentran en la escuela.

Hemos visto diferentes programas vinculados y asociados a entornos virtuales para la programación que pueden reacomodarse en la utilización para la enseñanza, lo que daría lugar a una nueva configuración formativa de los sistemas convencionales. Depende del entorno de posibilidades de las TIC que los docentes se adapten y redescubran cuál es la opción más adecuada para sus clases, según sea la oportuna combinación de elementos tecnológicos, pedagógicos y organizativos y las posibilidades de la tecnología desde la perspectiva de la formación flexible. A lo largo del análisis vimos que este tipo de *software* busca que el alumno aprenda o refuerce sus conocimientos dentro de un entorno de aprendizaje. Los *softwares* educativos diseñados para la enseñanza de la programación pueden ser recursos pedagógicos cuya aplicación didáctica en el aula depende de la forma y del propósito para el cual el docente pueda integrarlos a las clases, con lo cual, pueden aplicarse en diferentes asignaturas. En el análisis de los diferentes *softwares* educativos podemos llegar a pensar que para adaptarlos a las clases y sacar el mayor potencial es importante tener un conocimiento operacional y estructural (*know-how*), conocer la naturaleza del *software* y las relaciones entre el conocimiento tecnológico y pedagógico acerca de este.

Finalmente, al estar en un mundo cada vez más digitalizado, no basta con enseñar al alumno a ser usuario de la tecnología disponible. Saber programar ayuda a comprender mejor el entorno, tener los fundamentos de la programación estructurada independientemente

de la herramienta de *software* que se utilice. La inclusión de la programación en forma transversal requiere de una actualización permanente y es fundamental en la construcción de una educación actualizada y pertinente a los tiempos en los que se desarrolla. Podemos vislumbrar que las plataformas actuales de programación seguirán evolucionando y simplificando mucho su forma de trabajar, para que sea más sencillo su adaptación y uso por parte de los nativos digitales. Podemos pensar entonces que estos recursos tecnológicos representan instrumentos para crear situaciones y abordar contenidos que permiten a los alumnos experimentar cambios y transformaciones.

Referencias

Aedo Lopez, M., Vidal Duarte, E., Castro Gutiérrez, E. y Paz Valderrama A. (2016). Teaching Abstraction, Function and Reuse in the first class of CS1: A Lightbot Experience. En *Proceedings of the 2016 ACM Conference on Innovation and Technology in Computer Science Education (ITiCSE '16)*. Association for Computing Machinery, 256-257. https://doi.org/10.1145/2899415.2925505

Ahumada, H., Rivas, D., Contreras, N., Miranda, M. y Poliche, M. (2018). *Pensamiento computacional mediante programación por bloques: Intervención didáctica usando Pilas Bloques*. XIII Congreso de Tecnología en Educación y Educación en Tecnología. TE&ET Posadas.

Almaguel Guerra, A., Álvarez Mora, D., Pernía Nieves, L. A., Mota Pimentel, G. J. y Coello León, C. (2016). Software educativo para el trabajo con matrices. *Revista Digital: Matemática, Educación e Internet, 16*(2). https://doi.org/10.18845/rdmei.v16i2.2525

Almeida Santos Silva, R. y Bigotte de Almeida, M. E. (2017). All in scratch project. *2017 12th Iberian Conference on Information Systems and Technologies (CISTI)* (pp. 1-4). 10.23919/CISTI.2017.7975891

Basogain Olabe, X., Olabe Basogain, M. Á. y Olabe Basogain, J. C. (2015). Pensamiento computacional a través de la programación: Paradigma de aprendizaje. *Revista de Educación a Distancia (RED)*, *46*(6). 10.6018/red/46/6

Bender, W. (2017). La plataforma de aprendizaje Sugar: Affordances educativas para el pensamiento computacional. *Revista de Educación a Distancia (RED)*, *17*(54). https://revistas.um.es/red/article/view/298791

Computer Science Teachers Association (CSTA) e International Society for Technology in Education (ISTE). (2011). *Computational thinking leadership toolkit*. Computer Science Teachers Association (CSTA) and International Society for Technology in Education (ISTE). http://www.iste.org/docs/ct-documents/ct-leadershipt-toolkit.pdf?sfvrsn=4

Eckerdal, A., Thuné, M. y Berglund, A. (2005). What does it take to learn 'programming thinking'? En *Proceedings of the First International Workshop on Computing Education Research (ICER'05)*. Association for Computing Machinery, 135-142. https://doi.org/10.1145/1089786.1089799

Espíndola, M. C., Greiner, C. L. y Dapozo, G. N. (2018). *Evaluación de calidad de herramientas utilizadas en la enseñanza de la programación basada en ISO 25000*. XX Workshop de Investigadores en Ciencias de la Computación, Universidad Nacional del Nordeste, Chaco, Corrientes, Argentina

George Reyes, C. E. y Avello-Martínez, R. (2021). Alfabetización digital en la educación. Revisión sistemática de la producción científica en Scopus. *Revista de Educación a Distancia (RED)*, *21*(66). https://doi.org/10.6018/red.444751

González, J., Paparoni, V. y Vallejos, L. (2017) *Encendiendo las luces del conocimiento con LightBot 1.0: La construcción del conocimiento en la clase de computación*. XII Congreso de Tecnología en Educación y Educación en Tecnología (TE&ET), La Matanza, Buenos Aires, Argentina

Graziani, L., Cayú, G., Vivas, H. y Britos, P. (2016) Módulos didácticos digitales como herramienta de apoyo a la enseñanza y el aprendizaje para el desarrollo del pensamiento computacional. XIX Concurso de Trabajos Estudiantiles (EST 2016)-JAIIO 45, Tres de Febrero, Buenos Aires, Argentina.

Hara, I. y Hepp, P. (2016, 21 de diciembre). *Enseñar Ciencias de Computación. Creando oportunidades para los jóvenes de América Latina*. https://news.microsoft.com/uploads/2016/10/Commputer-Science-Whiter-Paper-LATAM-Spanish.pdf

Jiménez, Y., Kapoo, A. y Gardner-McCune, C. (2018). Usability challenges that novice programmers experience when using Scratch for the first time. *2018 IEEE Symposium on Visual Languages and Human-Centric Computing (VL/HCC)*, Lisboa. 10.1109/VLHCC.2018.8506560

Labañino Rizzo, C. A. (2000) *Multimedia para la educación*. Pueblo y Educación.

Marqués, G. (1999). *Software educativo: Guía de uso y metodología de diseño*. Estel.

Melo Herrera, M. P. y Hernández Barbosa, R. (2014). El juego y sus posibilidades en la enseñanza de las ciencias naturales. *Innovación Educativa, 14*(66), 41-63. http://www.scielo.org.mx/scielo.php?script=sci_arttext&pid=S1665-26732014000300004&lng=es&tlng=es

Monjelat, N., Cenacchi, M. y San Martín, P. (2018). ¿Programación para todos? Herramientas y accesibilidad: Un estudio de caso. *Revista Latinoamericana de Educación Inclusiva*, *12*(1), 213-227. https://dx.doi.org/10.4067/S0718-73782018000100213

Papert, S. (1987) *Desafío a la mente: computadoras y educación*. Galápago.

Raj Sidhu (2019). *Coding for kids in Scratch 3: The complete guide to creating art, artificial intelligence, and computer games for beginners*.

Rodríguez Lamas, R. (2000). *Introducción a la Informática educativa*. Instituto Superior Politécnico José A. Echevarría.

Sáez López, J. M. (2019) *Programación y robótica en Educación Infantil, Primaria y Secundaria*. UNED.

Sánchez, J. (1999). *Construyendo y aprendiendo con el computador*. Universidad de Chile.

Sanzo, A., Schapachnik, F., Factorovich, P. y O'Connor, F. S. (2017). Pilas Bloques: A scenario-based children learning platform, *2017 Twelfth Latin American Conference on Learning Technologies (LACLO)* (pp. 1-6). 10.1109/LACLO.2017.8120926

Sesento García, L. (2021). El constructivismo, posibilidades en el aula universitaria. *Milenaria, Ciencia y Arte*, *17*, 35-37. http://www.milenaria.umich.mx/ojs/index.php/milenaria/article/view/131

TED
Nº 52 • Segundo semestre de 2022 • pp. 117-136
ISSN 2665-3184 impreso • ISSN 2323-0126 Web

Para citar este artículo

Kuz, A. y Ariste, M. (2022). Análisis y revisión de *softwares* educativos para el aprendizaje de la programación en entornos lúdicos. *Tecné, Episteme y Didaxis: TED, (52)*, 117-136. https://doi.org/10.17227/ted.num52-13159