Artículo de Investigación. Revista Ciencia e Ingeniería. Vol. 44, No.3 pp. 357-366, agosto-noviembre, 2023.
. ISSN 1316-7081. ISSN Elect. 2244-8780 Universidad de los Andes (ULA

3EPCom: una aplicación para entrenar el pensamiento computacional

EPCom: an application to train computational thinking

Castro, José[1]*; Pérez, Jesús[1]; Besembel, Isabel[2]; Dapena, Eladio[3]
[1]Laboratorio de Sistemas Discretos, Automatización e Integración (LaSDAI),
Universidad de Los Andes, Mérida, Venezuela.
[2]Grupo de Investigación en Ingeniería de Datos y Conocimiento (GIDyC),
Universidad de Los Andes, Mérida, Venezuela.
[3]Universidad Intercontinental de la Empresa UIE. Santiago de Compostela. España
*castroj@ula.ve

Resumen

El pensamiento computacional surge por la necesidad de programar computadores pero su aplicación se ha expandido a diferentes disciplinas y áreas del conocimiento, donde en esencia se considera como un enfoque para la resolución de problemas. Existen diversas propuestas para desarrollar y evaluar las habilidades del pensamiento computacional, sin embargo, hace falta una herramienta que pueda ser utilizada en diferentes contextos y niveles educativos. En ese sentido, el objetivo de este trabajo fue diseñar e implementar una aplicación para Entrenar el Pensamiento Computacional llamada EPCom, para cubrir la limitación mencionada. El método usado en este proyecto es un marco de trabajo para el desarrollo de software en pequeñas empresas denominado Blue Watch. De acuerdo con las fases del Blue Watch: primero, se realizó una planificación inicial del proyecto; segundo, se realizó un análisis del contexto y requisitos iniciales; tercero, se hizo un diseño inicial de la arquitectura, interfaz y estructura de la aplicación; y cuarto, se efectuó una planificación del incremento, el cual fue llevado a cabo mediante la codificación y pruebas del mismo. Por último, para evaluar la usabilidad de EPCom y la aceptación de los estudiantes, se aplicó la Escala de Usabilidad del Sistema (SUS, por sus siglas en inglés). Los resultados fueron positivos, obteniendo una puntuación aceptable según la escala de SUS, y mostrando que EPCom tiene un diseño agradable que es fácil de usar.

Palabras clave: *Pensamiento Computacional; Resolución de Problemas; Ingeniería de Software; Método Blue Watch.*

Abstract

Computational thinking arises from the need to program computers, but its application has extended to different disciplines and areas of knowledge, where it is essentially considered a problem-solving approach. There are various proposals to develop and assess computational thinking skills, however, it is needed a tool that can be used in different contexts and educational levels. For that reason, the objective of this work was to design and implement an application to Train Computational Thinking called EPCom, to cover the aforementioned limitation. The method used in this project is a framework for small business software development called Blue Watch. According to the Blue Watch phases: first, initial planning of the project was carried out; second, an analysis of the context and initial requirements; third, an initial design of the architecture, interface, and structure of the application was made; and fourth, planning of the increase was performed by coding and testing it. Finally, to evaluate the usability of EPCom and the acceptance of the students, the System Usability Scale (SUS) was applied. The results were positive, obtaining an acceptable score according to the SUS scale, and showing that EPCom has a nice design that is easy to use.

Keywords: *Computational Thinking; Problems Solving; Software Engineering; Blue Watch method.*

1 Introducción

En los últimos años, el pensamiento computacional ha ganado popularidad en todos los niveles de educación porque, aunque surge por la necesidad de programar computadores, su aplicación se ha expandido a diferentes disciplinas y áreas del conocimiento como una metodología

de resolución de problemas que se puede automatizar (Zapata-Ros, 2015). Aunque aún no se ha incorporado formalmente en las mallas curriculares, los estudiantes han mostrado tener nociones básicas sobre este pensamiento y sus beneficios (Pérez, 2021).

De acuerdo con Pérez y col. (2021), existen carencias teóricas y prácticas sobre el pensamiento computacional debido a que éste es un tópico de investigación relativamente nuevo. En ese sentido, aunque se han diseñado distintas estrategias y herramientas de aprendizaje y evaluación, existen carencias o limitaciones tanto en las herramientas empleadas como en los diferentes enfoques. Por esa razón, en este trabajo se propone una alternativa para cubrir una de esas limitaciones, la cual está relacionada con el campo de acción de las herramientas utilizadas para entrenar el pensamiento computacional, que por lo general se centra en algún nivel de educación específico. A continuación se mencionan algunos ejemplos.

En el nivel preescolar, Lin y col. (2020) describen un estudio que tuvo como objetivo investigar las influencias de los enfoques de enseñanza para guiar a los niños en el aprendizaje de conceptos de programación y lógica computacional. El enfoque de enseñanza propuesto integra los conceptos de aprendizaje basado en juegos en un sistema de Interfaz de Usuario Táctil Interactiva (TUI). El proceso experimental se aplicó a 7 niños entre edades de 5 y 6 años, y consistió en dos estudios que contenían lecciones de enseñanza, un pre-test, una intervención y un post-test, siendo las herramientas utilizadas robots móviles basados en Arduino y tarjetas de colores extraídas de Bebras. Los resultados obtenidos permitieron observar que el enfoque TUI puede ayudar a los niños a tener una participación activa en el aprendizaje y mejorar su capacidad de pensamiento computacional, no obstante, esta estrategia está dirigida sólo a estudiantes de preescolar.

En educación primaria, Rodríguez-Martínez y col. (2020) presentaron un trabajo cuyo objetivo fue evaluar el potencial de la programación de actividades con Scratch para fomentar la adquisición de capacidades matemáticas y pensamiento computacional de estudiantes de sexto grado. Este estudio tuvo un diseño cuasi-experimental con una participación de 47 estudiantes, donde primero se aplicó un pre-test para identificar los conocimientos de fundamentos de programación y matemáticas de los estudiantes y dividir en dos grupos de estudio, luego, se avanzó a una fase donde se desarrollaron 5 sesiones para dotar de conocimientos sobre Scratch, y 3 sesiones adicionales para la enseñanza de estándares de aprendizaje matemático utilizando Scratch como herramienta pedagógica al grupo experimental, pero métodos tradicionales para el grupo de control; finalmente, fue aplicado un post-test donde los resultados indicaron un efecto significativo de la instrucción explícita en conceptos asociados al pensamiento computacional, sin embargo, hubo poca diferencia significativa en los dos grupos en cuanto a la evaluación de las matemáticas. En este caso, la herramienta (Scratch) está relacionada directamente con la informática y carreras afines, lo cual es una desventaja.

En educación secundaria, Shahin y col. (2022) desarrollaron un trabajo con el objetivo de comprender cómo las niñas de una escuela perciben las prácticas del pensamiento computacional cuando implementan soluciones computacionales con el dispositivo *micro:bit1* en un contexto de aprendizaje basado en problemas con entorno colaborativo. El proceso experimental fue empírico y se aplicó a 203 niñas de 14 y 16 años, que fueron guiadas mediante talleres en el que debían aprender el uso de *micro: bit1,* y luego poner en práctica estos conocimientos a través de soluciones en equipo en la codificación de *micro: bit1*. Los cuestionarios aplicados durante este proceso permitieron concluir que las niñas perciben como difíciles las tareas que requieren depurar, mientras que las prácticas colaborativas son vistas como fáciles. La principal limitación de este estudio es que requiere adquirir el dispositivo físico *micro: bit1*, lo cual representa un costo monetario que sólo tiene posibilidad de utilización en el nivel de secundaria.

En educación universitaria, Rodríguez del Rey y col. (2021) presentaron como trabajo el diseño de un módulo de resolución de problemas (MSP) para el desarrollo del pensamiento computacional de estudiantes del primer año de ingeniería informática. La metodología usada consistió en un pre-test y post-test con dos grupos, uno de control y otro experimental. El diseño del módulo contenía temas como Matemáticas I, Matemáticas II, Matemáticas Discretas, y problemas basados en juegos simples. Los resultados obtenidos mostraron pequeñas diferencias entre los dos grupos, sin embargo, el grupo experimental fue superior, permitiendo decir que la implementación del pensamiento analítico en los cursos de educación puede afectar con éxito la comprensión de los principios del pensamiento computacional por parte de los estudiantes en formación. De manera similar a la utilización de Scratch, el enfoque de esta herramienta está orientado a estudiantes del área de la informática, siendo inviable utilizar la herramienta con estudiantes de otras áreas.

En aras de cubrir estas limitaciones recurrentes, en este trabajo se diseñó, implementó, y evaluó una aplicación versátil y escalable para entrenar el pensamiento computacional. En las secciones restantes se presentan los siguientes aspectos: teoría del pensamiento computacional en la sección 2, metodología utilizada para el desarrollo de EPCom en la sección 3, procedimiento experimental en la sección 4, resultados en la sección 5, y finalmente, conclusiones y trabajos futuros en la sección 6.

2 Pensamiento Computacional

El pensamiento computacional se considera como una competencia fundamental del siglo XXI, la cual permite facilitar la resolución de problemas tanto en las ciencias de la computación como en otras disciplinas, y la vida cotidiana (Pérez, 2019). En general, este pensamiento permite reformular un problema aparentemente difícil en otro cuya solución sea conocida. De acuerdo con Wing (2006), en esencia este pensamiento trata de reducir, trasformar o simular un problema en otro más fácil.

Existen diferentes definiciones, sin embargo, Román-González y col. (2017) distinguen principalmente tres tipos: una definición *Genérica* donde asocia el pensamiento computacional con resolución de problemas, diseño de sistemas, y comprensión del comportamiento humano, basado en conceptos fundamentales de las ciencias de la computación; una definición *Operacional* que lo concibe como un proceso de pensamiento involucrado en la formulación de problemas, por lo que sus soluciones se pueden representar como algoritmos computacionales; y otra definición *Educacional* que relaciona al pensamiento computacional con once conceptos: abstracción, pensamiento algorítmico, automatización, descomposición, depuración, evaluación, generalización, resolución de problemas, trabajo en equipo, comunicación, e inteligencia espiritual.

Desde el punto de vista educacional, Wing (2008) resalta que el pensamiento computacional no se trata sólo de enseñar programación, sino de enseñar a los estudiantes a pensar de manera algorítmica y sistémica. En ese sentido, algunos autores (Csizmadia y col., 2015; Peréz, 2016; Mooney y Lockwood, 2020) coinciden en asociar cinco conceptos o habilidades básicas. La primera es la *Abstracción*, que consiste en seleccionar los detalles correctos para filtrar e ignorar toda la información que no es necesaria cuando se resuelve un problema determinado; la segunda es la *Descomposición*, que es la capacidad de dividir un problema en varios subproblemas para comprender, desarrollar y evaluar separadamente, permitiendo resolver problemas complejos con mayor facilidad; la tercera es el *Pensamiento Algorítmico*, donde se requiere desarrollar una estrategia de forma secuencial, es decir, establecer un orden en las instrucciones para resolver el problema; la cuarta es la *Generalización*, que consiste en identificar los patrones, similitudes o conexiones, dentro del mismo problema o con otros problemas, en aras de resolver rápido nuevos problemas basados en soluciones previas; y la quinta es la *Evaluación*, cuyo objetivo es asegurarse de que una solución, algoritmo, sistema o proceso funciona correctamente.

Las herramientas utilizadas para estimular el pensamiento computacional se pueden clasificar según el objetivo planteado. De acuerdo con Román-González y col. (2019), existen cinco tipos de herramientas: primero, herramientas de diagnóstico, que se enfocan en medir el nivel de aptitud del estudiante y se pueden emplear tanto antes como después de una intervención educativa para verificar si se logró algún cambio, por ejemplo, la ECTD que fue creada con temas de matemáticas, resolución de problemas y cálculo, para estudiantes de ingeniería (Diaz y col., 2022); segundo, herramientas formativas-iterativas que se enfocan en proporcionar retroalimentación al estudiante, generalmente de forma automática, para desarrollar y mejorar sus habilidades, por ejemplo Dr. Scratch (Moreno-León y Robles, 2015) que permite evaluar en términos de las habilidades del pensamiento computacional; tercero, herramientas de procesamiento de datos, que se enfocan en el proceso de aprendizaje, registrando y recuperando la actividad del estudiante en tiempo real, y proporcionando datos para el análisis. Un ejemplo fue aplicado en (Grover y col., 2017), para interpretar los registros de los estudiantes y medir su nivel de aprendizaje y comprensión del proceso del entorno de programación ALICE; cuarto, herramientas de transferencia de habilidades, cuyo objetivo es evaluar en qué medida los estudiantes pueden transferir sus habilidades a diferentes tipos de problemas, contextos y situaciones. Un ejemplo son los problemas del concurso anual de Bebras (Dagiene, y Futschek, 2008), que se centran en medir la transferencia de habilidades del pensamiento computacional a planteamientos de problemas de la vida cotidiana; y finalmente, herramientas de evaluación sumativa, que miden si el estudiante ha logrado aprender lo suficiente del contenido y es capaz de desempeñarse adecuadamente después de recibir alguna capacitación. El principal uso de este tipo es posterior a una intervención educativa, por ejemplo, en (Meerbaum-Salant, 2013) se diseñó un cuestionario para medir el aprendizaje adquirido con el uso de Scratch.

3 Método de desarrollo

3.1 Método Blue Watch

Para el desarrollo de este trabajo se utiliza el Método Blue Watch, que de acuerdo con Barrios y Montilva (2013), es un marco de trabajo metodológico para el desarrollo de aplicaciones empresariales. Éste es un método híbrido entre el tradicional y el ágil y está compuesto de tres modelos: el Modelo de Productos, que describe los productos intermedios y finales necesarios para desarrollar una aplicación ya sea de pequeña o mediana complejidad; el Modelo de Procesos, que describe las actividades técnicas, de gestión y de soporte requeridas para elaborar la aplicación; y finalmente, el Modelo de Actores que

identifica los roles necesarios para desarrollar la aplicación y propone una estructura organizacional para el grupo de trabajo. Cabe destacar que estos tres modelos pueden ser adaptables y extensibles según las necesidades.

3.2 Propuesta

La propuesta de este trabajo se denomina EPCom, una aplicación para entrenar el pensamiento computacional, cuya finalidad es que el estudiante estimule y fortalezca a través de la resolución de problemas sus habilidades del pensamiento computacional. La aplicación considera dos roles: el profesor, que pueda gestionar los problemas, y monitorear las estadísticas particulares y generales de los estudiantes; y el estudiante, que puede practicar y visualizar sus propias estadísticas. EPCom puede incluirse en las categorías de herramientas formativas-iterativas, procesamiento de datos, y transferencia de habilidades. A continuación, se describen las etapas principales que se implementaron para el desarrollo de EPCom.

3.2.1 Requisitos de la aplicación

El desarrollo inicial de los requisitos fue generado a partir de los casos de uso principales de la aplicación, que luego fueron definidos como requisitos funcionales, tal como se observa en la Tabla 1, donde se divide en dos roles de usuario. El primero es el rol de profesor, que debe acceder con un registro previo en la aplicación, para poder monitorear y analizar los resultados individuales y generales de los estudiantes, además de gestionar los problemas de interés sobre el pensamiento computacional; y el segundo, es el rol de estudiante, que puede acceder mediante un registro, para así practicar los problemas disponibles y poder visualizar su rendimiento.

Tabla 1. Requisitos funcionales para la aplicación

ID	Nombre	Rol
1	Registrar usuario	Profesor y estudiante
2	Crear problemas	Profesor
3	Actualizar problemas	Profesor
4	Leer problemas	Profesor
5	Borrar problemas	Profesor
6	Consultar estadísticas	Profesor
7	Practicar problema	Estudiante
8	Visualizar progreso	Estudiante

3.2.2 Diseño inicial

La aplicación fue desarrollada mediante el patrón de diseño que proporciona el framework Django. Este patrón de diseño es denominado Model-View-Template (MVT), y se compone de tres capas: la primera, es la capa *Model* que consiste en proporcionar de forma abstracta los modelos para estructurar y manipular los datos de la aplicación web; la segunda, es la capa *View*, que sirve para procesar cada petición del usuario a través del navegador y devolver una respuesta mediante la conexión con la primera capa; y la tercera, es la capa *Template*, que permite organizar la información recibida de la segunda capa y mostrarla en el navegador; en la Fig. 1 se observa la relación entre estas capas.

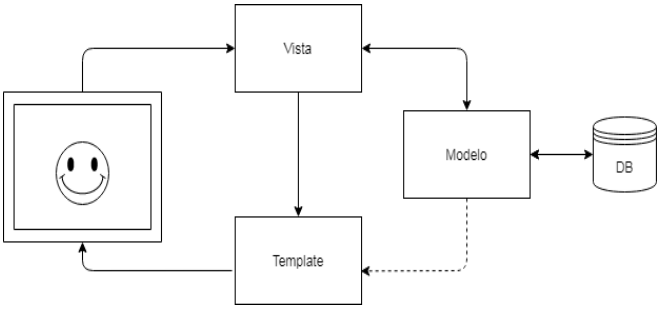

Fig. 1. Patrón de diseño MVT.

Con respecto al almacenamiento de la información, Django tiene compatibilidad con varios sistemas de gestión de bases de datos como por ejemplo: PostgreSQL, MySQL, y SQLite. Para este trabajo inicial se ha decidido usar SQLite, la cual viene integrada en Django y permite manejar los datos en un alto nivel de abstracción, además que se integra bien en el objetivo de este trabajo.

En el diseño de la aplicación fue priorizada la estética, de tal forma que presente un diseño atractivo, donde se hace uso de formas redondeadas que tienden a ser interesantes y se asocia a cualidades como la accesibilidad, amabilidad y armonía (Zhang y col., 2006); y se combina con una vista modo oscuro, que generalmente es preferida por los usuarios (Kim y col., 2019). A continuación, se muestran algunas de las pantallas y se describen sus funcionalidades.

En la Fig. 2 se refleja parte del rol del profesor en el que puede monitorear las estadísticas generales de los estudiantes, tanto del dominio de habilidades, como el rendimiento mensual. Así mismo, el profesor puede seleccionar y buscar a un estudiante específico y observar sus estadísticas. Por último, en el panel de la izquierda se puede acceder a los problemas disponibles para gestionarlos o crear otros.

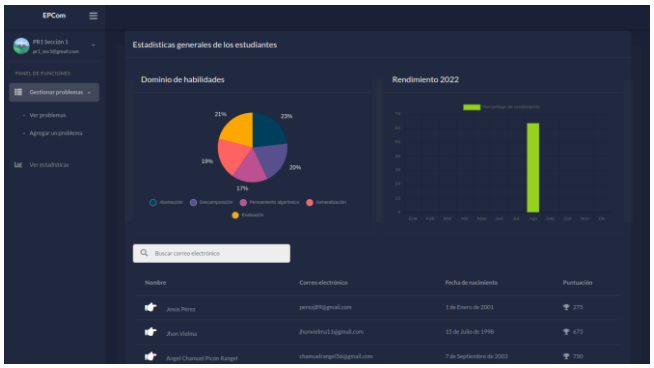

Fig. 2. Consultar estadísticas generales.

En la Fig. 3 se proporciona una de las funciones del estudiante que permite visualizar sus estadísticas, las cuales se actualizan a través del tiempo. De manera similar, el profesor puede ver estas estadísticas del estudiante desde su rol.

Fig. 3. Consultar estadísticas personales.

En la Fig. 4 se encuentra otra de las funciones del estudiante, donde se accede a un problema determinado en el que el estudiante debe seleccionar una de las posibles soluciones, u omitir temporalmente ese problema. Si el estudiante responde correctamente la pregunta, el problema es eliminado de su cuenta, mientras que si responde incorrectamente u omite, aparecerá en cualquier otro momento hasta que sea completado.

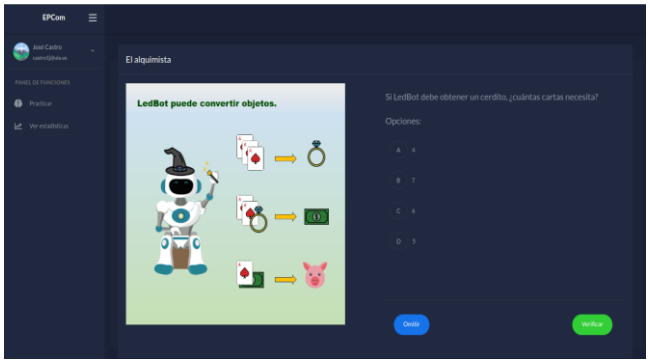

Fig. 4. Representación de un problema.

Se utilizaron cinco ecuaciones estadísticas: las dos primeras están relacionadas a las estadísticas particulares, la tercera y la cuarta se asocian con las estadísticas a nivel general; y la quinta, forma parte de ambas según la manera en que se muestran los datos. La primera, consiste en calcular el dominio de las habilidades del pensamiento computacional del estudiante, representando los porcentajes mediante $H = (N / T) * 100$, donde H es el porcentaje de una habilidad específica, N es el número de apariciones de la habilidad en todos los problemas resueltos por el estudiante, y T es la sumatoria total de apariciones de las habilidades en los problemas resueltos.

La segunda ecuación se utiliza para calcular el rendimiento del estudiante por cada mes mediante $P = (A / N) * 100$, donde P es el porcentaje de aciertos del mes, A es el número de aciertos, y N es el total de intentos realizados de los problemas. Utilizando esta ecuación se genera un histograma, donde el eje X representa los meses del año y el eje Y representa los porcentajes de aciertos de cada mes.

La tercera ecuación se enfoca en verificar a nivel general el dominio de las habilidades en los estudiantes, utilizando $D = Ht / U$, donde D es el porcentaje promedio de apariciones de una habilidad específica en los problemas resueltos de todos los estudiantes, Ht es la suma de todos los porcentajes particulares de esa habilidad (obtenida de la primera ecuación), y U es la cantidad total de estudiantes que han resuelto al menos un problema.

La cuarta ecuación permite ver mediante un histograma el rendimiento general que tienen los estudiantes en cantidad de aciertos. Es decir, $R = Pt / U$, donde R es el porcentaje promedio de rendimiento de los estudiantes en el mes, Pt es la suma de porcentajes de cada estudiante en el mes (se obtiene de la segunda ecuación), y U es la cantidad total de estudiantes que han resuelto al menos un problema en el mes. Finalmente, la quinta ecuación consiste en calcular el puntaje acumulado (Ac) de los problemas resueltos por el estudiante, aplicando $Ac = Ac + i$, donde el valor inicial de Ac es 0 y la razón de incremento i es 25 que equivale a puntos/problema.

3.2.3 Desarrollo de EPCom

En el desarrollo de EPCom se planteó como objetivo realizar un incremento de producto mínimo viable (MVP) que permitiera completar los requisitos; por este motivo, se establecieron las siguientes actividades: (1) extraer los requisitos a desarrollar; (2) codificar el incremento; y (3) realizar las pruebas para la validación.

3.2.4 Pruebas

Para esta última parte del desarrollo del método Blue Watch, se realizaron dos tipos de pruebas: las *pruebas* unitarias mediante la técnica de caja negra donde se utilizaron las tablas de decisión para validar y verificar cada una de las funcionalidades; y las pruebas de usuario donde se desplegó EPCom en la web y se creó una práctica con 30 problemas de pensamiento computacional extraídos de Bebras. Luego, para validar esta prueba fue usado el modelo SUS (Brooke, 1996), que proporciona un cuestionario para medir la percepción de los usuarios sobre la usabilidad de un software. En la siguiente sección se detalla el procedimiento aplicado.

4 Procedimiento Experimental

El objetivo de este experimento es medir la usabilidad de EPCom y la satisfacción de los estudiantes. Para ello, los estudiantes utilizan EPCom y luego responden un cuestionario basado en el modelo SUS. El experimento fue aplicado a 31 estudiantes del primer semestre de Ingeniería de Sistemas de la Universidad de Los Andes, en Mérida, Venezuela. Específicamente, a los estudiantes se les solicitó: (1) registrarse en EPCom, (2) probar las funcionalidades, (3) completar los problemas disponibles, y (4) responder el cuestionario.

Aunque el modelo SUS fue desarrollado en el siglo pasado, éste ha sido ampliamente considerado para evaluar la usabilidad. SUS proporciona un cuestionario con 10 afirmaciones para ser respondidas según una escala de Likert de 5 puntos. Estas afirmaciones se refieren principalmente a la facilidad de uso, satisfacción y confiabilidad del sistema. Adicionalmente, SUS contiene un método para evaluar los resultados e interpretarlos según la puntuación obtenida.

5 Resultados y Discusión

En la Tabla 2 se muestran las 10 preguntas proporcionadas por SUS y 1 pregunta adicional para conocer la apreciación de los estudiantes sobre la estética de la aplicación. Para cada pregunta se presentan los porcentajes de cada categoría con respecto a las respuestas obtenidas de los estudiantes. Es importante destacar que estos porcentajes permiten obtener una visión general de las respuestas obtenidas, pero que SUS tiene sus propias normas para calcular la puntuación de la aplicación.

De esta forma, se observa gran porcentaje positivo en las preguntas impares, las cuales son destinadas a confirmar las posibles ventajas que puede generar la aplicación. Por el contrario, las preguntas pares intentan contradecir o desfavorecer, siendo bueno que las respuestas sean "Desacuerdo" o en "Totalmente en Desacuerdo". Específicamente, para la pregunta 1 se resalta que a la mayoría de los estudiantes les gustaría usar frecuentemente EPCom, lo cual es importante porque la aplicación está orientada a que los estudiantes puedan practicar sus habilidades del pensamiento computacional por un tiempo prolongado; según la pregunta 2, muchos estudiantes consideran que la aplicación no es vista como compleja, estando relacionado con la pregunta 3 y 4, donde hay un buen porcentaje que considera facilidad en el uso de EPCom, y que no es necesario algún tipo de ayuda para aprender a usarla, respectivamente.

Así mismo, en la pregunta 5, muchos de los estudiantes consideran que las funciones de EPCom están bien integradas, siendo coherente con los resultados de la pregunta 6, donde la mayoría no cree que haya inconsistencias. En la pregunta 7 se observa que los estudiantes piensan también que otras personas podrían usarla rápidamente, siendo consistente con la pregunta 8 donde no se refleja que sea difícil de usar. Esto apoya los planes futuros de implementar sesiones de trabajo dentro de los diferentes niveles educativos para fomentar el pensamiento computacional.

Por su parte, en la pregunta 9, los estudiantes consideran que el uso de la aplicación es seguro, lo cual sugiere que perciben privacidad en su información personal, que de hecho es sólo observada por ellos y su profesor. En la pregunta 10, los estudiantes opinan que no es necesario aprender muchas cosas para usar la aplicación, y en definitiva es así, ya que es muy intuitiva. Finalmente, la pregunta 11, permite observar que efectivamente el uso de colores cálidos, y formas redondeadas es atractivo para los usuarios. Esto también se puede confirmar en las aplicaciones más populares del momento porque también han adoptado este estilo.

Tabla 2. Resultados porcentuales del cuestionario

Ítem	TA	A	I	D	TD
1. Me gustaría usar la aplicación con frecuencia	23%	58%	6%	13%	
2. Encontré a la aplicación innecesariamente compleja		5%	30%	25%	40%
3. Creo que la aplicación es fácil de usar	35%	45%	10%	10%	
4. Pienso que necesitaré apoyo técnico para usar la aplicación	5%	20%	15%	30%	30%
5. Encontré funciones que están muy bien integradas en la aplicación	25%	55%	15%	5%	
6. Creo que hay mucha inconsistencia en la aplicación		10%	25%	35%	30%
7. Pienso que la mayoría de las personas aprendería a usar rápidamente esta aplicación	45%	45%	10%		

8. La aplicación es muy difícil de usar	5%	10%	10%	30%	45%
9. Me sentí muy seguro al estar usando la aplicación	35%	30%	30%	5%	
10. Necesité aprender muchas cosas antes de empezar a usar la aplicación	10%	10%	15%	35%	30%
11. Creo que la aplicación tiene un diseño y colores agradables a la vista	42%	39%	13%	6%	

Leyenda: TA (Totalmente de Acuerdo), A (De Acuerdo), I (Indiferente), D (En desacuerdo), TD (Totalmente en Desacuerdo)

Los resultados obtenidos fueron alentadores, indicando que EPCom es una herramienta que los estudiantes aceptan y podrían utilizar cómodamente. La principal ventaja de EPCom es que puede ser utilizada en diferentes contextos y niveles educativos. Esto es así porque por su diseño o estructura permite agregar problemas que pueden ser ajustados según la necesidad del profesor en función del contexto y el nivel educativo. En otras herramientas no es posible porque su enfoque se orienta sólo a alguna área específica que comúnmente está directamente relacionada con la programación como es el caso de Scratch (Rodríguez-Martínez y col., 2020).

Además, EPCom presenta otras ventajas. Primero, cómo está basada en Web, permite que los estudiantes accedan en cualquier momento estableciendo su propio ritmo de entrenamiento. Otras herramientas obligan a completar sesiones de entrenamientos presenciales que pueden no ser eficientes, tal como pasa en (Lin y col., 2020). Segundo, es muy accesible porque sólo requiere conexión a Internet, a diferencia de otras herramientas que, por ejemplo, requieren dispositivos físicos, implicando una inversión monetaria mayor, tal como es el caso de propuestas basadas en Arduino (Shahin y col., 2022). Tercero, ofrece seguimiento del rendimiento de los estudiantes según un dominio de habilidades, permitiendo actualizar el rendimiento cada vez que el estudiante resuelve un problema. El seguimiento es una ventaja importante porque, generalmente, el estudiante es evaluado con algún método para conocer su rendimiento en periodos fijos de tiempo, por ejemplo, en el caso (Rodríguez-Martínez y col., 2020).

6. Conclusiones

Las herramientas propuestas por otros autores para entrenar el pensamiento computacional están diseñadas para ser utilizadas en contextos particulares o niveles específicos de educación, limitando su campo de acción. A diferencia, EPCom es una herramienta versátil que puede ser configurada para permitir abordar los contextos deseados con su respectivo nivel de dificultad. Por ejemplo, se pueden agregar planteamientos relacionados a las problemáticas de los hospitales dirigidos específicamente a estudiantes de medicina, o situaciones de administración pública para estudiantes de política.

El diseño de EPCom consideró las desventajas de otras herramientas de diferentes niveles educativos. En ese sentido, su diseño resultó escalable porque permite agregar tantos problemas como sean necesarios, y dependiendo del enfoque (las habilidades varían según el enfoque), permite etiquetar las habilidades del pensamiento computacional asociadas a cada problema, para posteriormente hacer seguimiento del rendimiento de los estudiantes en función de las diferentes habilidades.

Dado que la prueba de usabilidad arrojó resultados favorables sobre la experiencia de usuario, resaltando su diseño agradable y facilidad de uso, los trabajos futuros estarán orientados a fortalecer el punto de vista pedagógico. Específicamente, se incorporará un modelo de usuario que permite predecir su rendimiento en aras de optimizar el proceso de entrenamiento con un enfoque personalizado tal como lo indican Pérez y col. (2022). Luego, en la misma herramienta EPCom, se incluirán tutoriales básicos de resolución de problemas con el enfoque de concepto-ejemplo-actividad utilizado por Pérez y Azuaje (2019).

Referencias

Barrios, J., Montilva, J., (2013). Integrating the methodological frameworks WATCH and SCRUM: A method engineering approach. 2013 XXXIX Latin American Computing Conference (CLEI), Caracas, Venezuela, pp. 1-11, doi: 10.1109/CLEI.2013.6670640.

Brooke, J., (1996). SUS: A “quick and dirty” usability scale. In Usability evaluation in industry (pp. 189-194). Taylor and Francis.

Csizmadia, A., Curzon, P., Dorling, M., Humphreys, S., Ng, T., Selby, C., Woollard, J., (2015). Computational thinking - A guide for teachers. United Kingdom: Computing At School.

Dagiene, V., Futschek, G., (2008). Bebras international contest on informatics and computer literacy: Criteria for good tasks. In International Conference on Informatics in Secondary Schools-Evolution and Perspectives, pp. 19-30. doi:10.1007/978-3-540-69924-8_2

Diaz, N. V. M., Trytten, D. A., Meier, R., Yoon, S. Y. (2021). An Engineering Computational Thinking Diagnostic: A Psychometric Analysis. In 2021 IEEE Frontiers in Education Conference (FIE), pp. 1-5. Lincoln, NE, USA. doi:10.1109/FIE49875.2021.9637142.

Grover, S., Basu, S., Bienkowski, M., Eagle, M., Diana, N., Stamper, J. (2017). A framework for using hypothesis-driven approaches to support data-driven learning analytics in measuring

computational thinking in block-based programming environments. ACM *Transactions on Computing Education* (TOCE), 17(3), pp. 1-25. doi:10.1145/3105910

Kim, K., Erickson, A., Lambert, A., Bruder, G., Welch, G., (2019). Effects of dark mode on visual fatigue and acuity in optical see-through head-mounted displays. In Symposium on spatial user interaction, pp. 1-9.

Lin, S.-Y., Chien, S.-Y., Hsiao, C.-L., Hsia, C.-H., Chao, K.-M., (2020). Enhancing computational thinking capability of preschool children by game-based smart toys. Electronic Commerce Research and Applications, 44, 101011. doi:10.1016/j.elerap.2020.101011

Meerbaum-Salant, O., Armoni, M., Ben-Ari, M., (2013). Learning computer science concepts with scratch. Computer Science Education, 23(3), pp. 239-264.

Mooney, A., Lockwood, J., (2020). The Analysis of a Novel Computational Thinking Test in First Year Undergraduate Computer Science Course. All Ireland *333*, 12(1), pp. 1-26.

Moreno-León, J., Robles, G., (2015). Dr. Scratch: A web tool to automatically evaluate Scratch projects. In Proceedings of the workshop in primary and secondary computing education, pp. 132-133.

Pérez, J., (2019). El pensamiento computacional en la vida cotidiana. Revista Scientific, 4(13), pp. 293-306.

Pérez J., Azuaje, M., (2019). LE1: una estrategia amistosa para un curso introductorio de programación. *Revista Educación En Ingeniería*, 14(28), pp. 45-53.

Pérez J., (2021). Percepción de estudiantes universitarios sobre el pensamiento computacional. REDU. *Revista de Docencia Universitaria*, 19(1), pp. 111-127.

Pérez J., Dapena E., Aguilar J., Carrillo G., (2022). Reinforcement Learning for Estimating Student Proficiency in Math Word Problems, 2022 XVII Latin American Conference on Learning Technologies (LACLO), Armenia, Colombia, 01-06.

Pérez J., Castro J., Pedroza O., (2021). Carencias en la evaluación del pensamiento computacional. Revista de Filosofía, 38(99), pp. 369-379.

Rodríguez del Rey, YA., Cawanga Cambinda, IN., Deco, C., Bender, C., Avello-Martínez, R., Villalba-Condori, KO., (2021). Developing computational thinking with a module of solved problems. *Comput Appl Eng Educ*. 29, pp. 506 - 516. doi:10.1002/cae.22214

Rodríguez-Martínez, J. A., González-Calero, J. A., Sáez-López, J. M., (2020). Computational thinking and mathematics using Scratch: an experiment with sixth-grade students. Interactive Learning Environments, 28(3), pp. 316-327. doi:10.1080/10494820.2019.1612448

Román-González, M., Moreno-León, J., Robles, G., (2019). Combining Assessment Tools for a Comprehensive Evaluation of Computational Thinking Interventions. Computational Thinking Education, pp. 79-98. doi:10.1007/978-981-13-6528-7_6

Román-González, M., Pérez-González, J. C., Jiménez-Fernández, C., (2017). Which cognitive abilities underlie computational thinking? Criterion validity of the Computational Thinking Test. Computers in Human Behavior, 72, pp. 678-691. do:i10.1016/j.chb.2016.08.047

Shahin, M., Gonsalvez, C., Whittle, J., Chen, C., Li, L., Xia, X., (2022). How secondary school girls perceive Computational Thinking practices through collaborative programming with the micro:bit. Journal of Systems and Software, 183, 111107. doi:10.1016/j.jss.2021.111107

Zapata-Ros, M., (2015). Pensamiento computacional: una nueva alfabetización digital. RED, Revista de Educación a Distancia, 46(4).

Zhang, Y., Feick, L., Price, L. J., (2006). The Impact of Self-Construal on Aesthetic Preference for Angular Versus Rounded Shapes. Personality and Social Psychology Bulletin, 32(6), pp. 794-805. doi:10.1177/0146167206286626

Wing, J., (2006). Computational thinking. *Communications of the ACM*, *49*(3), pp. 33-35. doi:10.1145/1118178.1118215

Wing, J., (2008). Computational thinking and thinking about computing. Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences, 366(1881), pp. 3717-3725. doi: 10.1098/rsta.2008.0118

Recibido:*12 de febrero de 2023*

Aceptado: *20 de julio de 2023*

Castro, José*: Br. en Ciencias, estudiante de Ingeniería de Sistemas en la Universidad de Los Andes (ULA), investigador del Laboratorio de Sistemas Discretos, Automatización e Integración (LaSDAI), y asistente de profesor en la asignatura programación*
https://orcid.org/0000-0001-6353-9941

Pérez, Jesús*: Postdoctorado en Investigación Educativa (2022), Dr. en Ciencias de la Educación (2019), MSc. en Educación Superior (2014), Ing. de Sistemas (2014), Ing. en Electrónica (2012). Es Profesor Agregado en la*

Universidad de Los Andes (ULA), e Investigador del Laboratorio de Sistemas Discretos, Automatización, e Integración (LaSDAI). jesuspangulo@ula.ve
https://orcid.org/0000-0002-6585-2648

Besembel, Isabel*: Ph.D. en Computación (1999), D.E.A. en Sistemas Informáticos (1983), Ing. de Sistemas (1978). Profesora Titular Jubilada de la Universidad de Los Andes (ULA) y Grupo de Ingeniería de Datos y Conocimiento (GIDyC). Correo electrónico: ibc@ula.ve*
https://orcid.org/0009-0007-1450-9771

Dapena, Eladio*: Doctor Ingeniero Industrial. UC3M. España (2002) Automatización Industrial (Esp.), UFSC. Brasil. Ingeniería de Sistemas. ULA. Venezuela (1990). Profesor jubilado Universidad de Los Andes (ULA). Profesor Titular Universidad Intercontinental de la Empresa UIE. Correo electrónico: eladio.dapena@uie.edu*
https://orcid.org/0000-0002-9135-0967