Mundo Fesc E-ISSN: 2216-0388 P- ISSN: 2216-0353 Artículo Original

Implementación experimental de modelos activos para la enseñanza de programación en entornos universitarios

Experimental implementation of active models for programming education in university settings

Recibido: 26 de mayo de 2024
Aprobado: 12 de agosto 2024

Forma de citar: L. J. Ibatá Soto, J. C. Correa Zapata, S. L. Vallejo Córdoba, and J. D. Tamayo Quintero, "Implementación experimental de modelos activos para la enseñanza de programación en entornos universitarios", *Mundo Fesc*, vol. 14, no. 30, pp. 318–337, Sep. 2024, doi: 10.61799/2216-0388.1794.

Laura Julieth Ibatá Soto
Estudiante de Ingeniería en Software. Correo electrónico: laura.ibata@tdea.edu.co
Afiliación institucional: Tecnológico de Antioquia – Institución Universitaria. ORCID: https://orcid.org/0009-0009-6591-1394. Antioquia, Colombia

Juan Carlos Correa Zapata
Estudiante de Ingeniería en Software, Tecnológico de Antioquia - Institución Universitaria. Correo electrónico: juan.correa@tdea.edu.co. ORCID: https://orcid.org/0009-0009-2025-5127. Antioquia, Colombia

Silvana Lorena Vallejo Córdoba
Magíster en Automatización Industrial, Tecnológico de Antioquia - Institución Universitaria. Correo electrónico: silvana.vallejo81@tdea.edu.co. ORCID: https://orcid.org/0000-0002-6770-0840. Antioquia, Colombia

Juan David Tamayo Quintero
Doctor en Ingeniería, Tecnológico de Antioquia - Institución Universitaria. Correo electrónico: juan.tamayo1@tdea.edu.co. ORCID: https://orcid.org/0000-0002-0159-1573 Antioquia, Colombia

***Autor para correspondencia:**
laura.ibata@tdea.edu.co

Implementación experimental de modelos activos para la enseñanza de programación en entornos universitarios

Resumen

La enseñanza tradicional de la programación presenta dificultades en el desarrollo de pensamiento algorítmico y lógico. Para enfrentar esto, se ha explorado el aprendizaje activo, que involucra directamente al estudiante en la resolución de problemas. El Tecnológico de Antioquia implementó un experimento con una herramienta de aprendizaje activo para enseñar fundamentos de programación a estudiantes de estratos socioeconómicos diversos, buscando superar las dificultades en la comprensión de conceptos como algoritmos y estructuras de control. Objetivo: Evaluar experimentalmente el impacto de una herramienta de aprendizaje activo, basada en ejercicios medidos por siete dimensiones, en comparación con métodos tradicionales en la enseñanza de la programación. Métodos: Se realizó un estudio con estudiantes de primer semestre del Tecnológico de Antioquia. La muestra se dividió en un grupo de control, que recibió enseñanza tradicional, y un grupo experimental, que utilizó una herramienta de aprendizaje activo con más de 1000 ejercicios en Python. Se aplicaron tres pruebas diagnósticas (inicial, intermedia y final) para medir el progreso en siete dimensiones del pensamiento computacional. Los datos se analizaron con Power BI. Resultados: Se observó una mejora en el rendimiento del grupo experimental entre la primera y segunda prueba diagnóstica, especialmente en Pensamiento Lógico y Reconocimiento de Patrones. El análisis del etiquetado manual de 1018 ejercicios de la herramienta arrojó que el 59.86% fueron correctamente generados de forma automática. Las dimensiones con menor rendimiento inicial fueron Resolución de Problemas, Algoritmos y Pensamiento Lógico. Conclusión: La implementación de modelos activos de programación, apoyados por herramientas interactivas, muestra ser efectiva para mejorar el aprendizaje en entornos universitarios. La herramienta demostró ser un enfoque inclusivo, efectivo en estudiantes de diversos perfiles socioeconómicos, aunque se requiere refinar la automatización en la generación de ejercicios para optimizar su calidad y reducir la revisión manual.

Palabras clave: aprendizaje activo, enseñanza de la programación, pensamiento computacional, Python, tecnología educativa.

Experimental implementation of active models for programming education in university settings

Abstract

Traditional programming education faces challenges in developing algorithmic and logical thinking. To address this, active learning has been explored, directly engaging students in problem-solving. The Tecnológico de Antioquia implemented an experiment with an active learning tool to teach programming fundamentals to students from diverse socioeconomic backgrounds, aiming to overcome difficulties in understanding concepts like algorithms and control structures. Objective: To experimentally evaluate the impact of an active learning tool, based on exercises measured by seven dimensions, compared to traditional methods in programming education. Methods: A study was conducted with first-semester students at Technologic of Antioquia. The sample was divided into a control group, which received traditional instruction, and an experimental group, which used an active learning tool with over 1000 Python exercises. Three diagnostic tests (initial, intermediate, and final) were administered to measure progress across seven dimensions of computational thinking. Data was analyzed using Power BI. Results: An improvement in the experimental group's performance was observed between the first and second diagnostic tests, especially in Logical Thinking and Pattern Recognition. The manual labeling analysis of 1018 exercises from the tool showed that 59.86% were correctly generated automatically. The dimensions with the lowest initial performance were Problem Solving, Algorithms, and Logical Thinking. Conclusion: The implementation of active programming models, supported by interactive tools, proves effective for enhancing learning in university settings. The tool demonstrated an inclusive approach, effective across students of diverse socioeconomic profiles, although refinement of the automated exercise generation is required to optimize quality and reduce manual review.

Keywords: active learning, computational thinking, educational technology, programming education, Python.

Introducción

En los últimos años, la educación para hacerle frente a los retos del mundo global y del desarrollo tecnológico, ha comenzado un proceso de transformación y renovación[1], de esta manera es propio decir que el paso del tiempo ha aumentado las necesidades de nuevos recursos educativos y el uso de nuevas tecnologías ha ido sustituyendo los modelos tradicionales[2], esto se está viendo en la educación en programación que ha experimentado una transformación significativa, evolucionando de métodos tradicionales de enseñanza centrados en clases magistrales hacia practicas educativas participativas que involucran de manera directa a los estudiantes en su proceso de aprendizaje [3]. Este cambio responde a desafíos identificados y documentados donde los enfoques tradicionales centrados en el profesor, con altas tasas de fracaso y deserción [4][5][6], no se alinean con las preferencias y nuevas formas de aprendizaje de los estudiantes de ingeniería [7].

En este sentido, la educación en programación no solo requiere conocimiento técnico, sino también el desarrollo del pensamiento computacional (CT) entendido como un conjunto de habilidades cognitivas que permiten a los estudiantes formular problemas y soluciones a problemas de forma lógica y sistemática. En los procesos de enseñanza que desarrollan los docentes se busca que los estudiantes mejoren estas habilidades[8]; el CT son dimensiones clave la descomposición, el reconocimiento de patrones, la abstracción, y el pensamiento algorítmico [9] donde los estudiantes aprenden a desglosar problemas complejos y articular soluciones, que son habilidades fundamentales en el aprendizaje de la programación.

Investigaciones como la de Scherer [10] demuestran que el aprendizaje de programación no solo mejora habilidades técnicas, sino también el pensamiento crítico, creativo y matemático, con efectos de transferencia significativos ($g = 0.75$ en transferencia cercana). Más que aprender la sintaxis y las estructuras de codificación, el fortalecimiento del CT apoyado por herramientas tecnológicas con enfoque experiencial que integre reflexión y experimentación activa fomentan un proceso metacognitivo de resolución de problemas aplicable en múltiples campos del conocimiento [11][12]. Las Metodologías de Aprendizaje Activo (ALMs) desde sus estilos participativos representan un cambio frente a los enfoques tradicionales, ya que buscan integrar la participación activa del estudiante, el aprendizaje experiencial y el "aprender haciendo" (aprendizaje basado en proyectos) como herramientas principales del proceso formativo [1][2][13]

La educación superior está en constante renovación y mejoramiento continuo, dada su búsqueda de nuevos conocimientos[14], es por ello, que para hablar de educación superior de calidad, es necesaria la implementación de herramientas tecnológicas, permitiendo que los egresados se encuentren mejor capacitados para enfrentar los retos laborales [15], en este sentido la formación en programación se apoya en diversas herramientas

tecnológicas que permiten a los estudiantes interactuar con los conceptos de distintas maneras [16][17][18][19][20][21], desde actividades desconectadas (CS-Unplugged) para favorecer la adquisición de CT, el uso de herramientas de programación visual que fomentan una actitud positiva hacia la programación, el uso de lenguajes textuales como Python para enfocarse en el fortalecimiento del pensamiento algorítmico, hasta la integración de robótica buscando motivación en resolver problemas reales y un enfoque multidisciplinario.

Estudios como los de Hamada [7] y Cakula [22] demuestran que las metodologías activas que involucran trabajo participativo de codificación y aprendizaje basado en proyectos mejoran los resultados de los estudiantes y son igualmente efectivas en entornos presenciales y digitales. Complementariamente García [6] y Esparza [4] destacan que las apuestas educativas activas muestran avances significativos en la comprensión y retención de conceptos básicos y su combinación con tecnologías emergentes fomenta aprendizajes dinámicos y autónomos [23]. La transformación educativa hacia enfoques activos con integración de herramientas digitales es esencial para desarrollar habilidades del siglo XXI [9]; habilidades técnicas y habilidades blandas que permiten a los estudiantes afrontar los desafíos en la industria actual [21][24].

En este contexto, el proyecto *Programación Activa: Comparación de Enfoques de Enseñanza a través de Ejercicios Interactivos para el Aprendizaje de la Programación del Tecnológico* de Antioquia - Institución Universitaria, pone en marcha la implementación experimental de una estrategia activa para la enseñanza de programación en varias fases que involucran la diagnosis, la intervención a partir de metodologías de aprendizaje activo con el uso de una herramienta tecnológica creada con propósito específico, y la aplicación de instrumentos de evaluación y validación de la estrategia. Toda esta implementación se hace alrededor de siete dimensiones que incluyen los componentes fundamentales del CT (descomposición, reconocimiento de patrones, abstracción y diseño de algoritmos) sumado al pensamiento lógico, la resolución de problemas y habilidades de programación en Python. Estas dimensiones buscan facilitar la formación en programación, así como su seguimiento y evaluación, asegurando que los estudiantes desarrollen de manera progresiva un enfoque sistemático para abordar y resolver problemas de manera efectiva [25][26].

La elección de dimensiones específicas para guiar el aprendizaje permite enfocar la enseñanza en habilidades esenciales que, de otro modo, podrían pasar desapercibidas en métodos de instrucción tradicionales [27]. Al estructurar el aprendizaje en torno a estas siete dimensiones, el objetivo es proporcionar a los estudiantes herramientas técnicas como también las habilidades estratégicas que les permitan aplicar ese conocimiento en la resolución de problemas reales. La herramienta de aprendizaje activo con ejercicios de programación medidos a partir de las siete dimensiones busca mejorar la forma en que se enseñan los fundamentos de la programación a grupos heterogéneos conformados por estudiantes provenientes de diversos estratos socioeconómicos, especialmente de los estratos 1, 2 y 3, con diversos niveles de conocimiento previo.

La herramienta cuenta con una base de 1000 ejercicios de programación en diferentes temáticas de la formación en fundamentos de programación en Python generados por la integración de la API de ChatGPT de OpenAI. Si bien la integración de inteligencia artificial en la enseñanza de programación representa una oportunidad prometedora para potenciar el desarrollo del pensamiento computacional, también representa desafíos como la dependencia tecnológica que puede llegar a limitar el pensamiento autónomo de los estudiantes [28][29]. Para abordar este riesgo se aplica la Teoría del Sistema de Andamiaje para Programación utilizando ChatGPT (IPSSC) formulada por Liao et al. [29] que busca limitar el uso de la IA a módulos estructurados de evaluación y retroalimentación para guiar progresivamente a los estudiantes en tareas de codificación. En este sentido, la herramienta desarrollada en el marco del proyecto Programación activa, se alimenta de problemas generados por IA y progresivamente deja disponibles módulos de ayuda y de solución restringidos por clave para que tengan un uso supervisado. Con estas condiciones se busca equilibrar los beneficios de la IA con el desarrollo sostenido del pensamiento computacional y la autonomía.

Las etapas de evaluación y validación de la estrategia activa para la enseñanza de programación se diseñaron para evaluar de manera experimental el impacto de estos modelos activos en comparación con los métodos tradicionales, utilizando pruebas diagnósticas y herramientas de visualización de datos basados en tableros de Power BI para monitorear el progreso y la adaptación de los estudiantes en su proceso de aprendizaje. Los hallazgos después de un ciclo de experimentación (un semestre) con estudiantes de Fundamentos de programación del Tecnológico de Antioquia muestran una mejora significativa sobre el grupo de control en las dimensiones evaluadas al comparar los resultados de la primera (a priori) y la segunda prueba diagnóstica (a posteriori) (en promedio 0.3 puntos porcentuales por encima). Este progreso es indicador de un desarrollo positivo en habilidades como Resolución de Problemas, Pensamiento Lógico, Algoritmos y Conocimientos Básicos de Programación y de la efectividad de la estrategia basada en metodologías de aprendizaje activo con el soporte de herramientas tecnológicas para el aprendizaje personalizado.

Materiales y métodos

El presente estudio utilizó un diseño de investigación cuasiexperimental, enfocado en evaluar el impacto de una herramienta de aprendizaje activo en comparación con los métodos tradicionales de enseñanza. La investigación se centró en comparar experimentalmente los métodos en el curso de fundamentos de programación, monitoreando el progreso de los estudiantes mediante pruebas diagnósticas.

Para el marco teórico, se realizó una consulta en sitios de información académica y científica sobre programación activa y pensamiento computacional. Se empleó la herramienta “Publish or Perish” para extraer publicaciones y sus citas de bases de datos como Google Académico, Scopus y Web of Science. Las búsquedas se realizaron con palabras clave como ‘pensamiento computacional’, ‘Aprendizaje en programación’ y

'Programación activa'. Los criterios para la selección de artículos incluyeron la afinidad de los títulos con los temas y el número de citaciones de cada trabajo.

La población del estudio se compuso de estudiantes de primer semestre de la asignatura de Introducción al Desarrollo de Software de la Facultad de Ingeniería del Tecnológico de Antioquia. Los participantes tenían edades comprendidas entre los 16 y 41 años. La mayoría de los estudiantes pertenecían a los estratos socioeconómicos 1, 2 y 3, como se muestra en la Tabla 1.

Tabla I. Información sociodemográfica

Característica		Valores
Género	Femenino	**18%**
	Masculino	**82%**
Estrato	Estrato 1	**26.40%**
	Estrato 2	**52.81%**
	Estrato 3	**21.35%**
	Estrato 4	**1.12%**
	Estrato 5	**0.56%**

Se utilizó un muestreo no probabilístico por conveniencia, ya que la muestra de la investigación se conformó por seis grupos preexistentes del periodo académico 2024-02: tres grupos de control y tres grupos experimentales.

Procedimiento y Grupos de Intervención

El experimento se centró en la comparación de dos enfoques de enseñanza. El grupo de control fue partícipe de una enseñanza tradicional, la cual se basa en clases magistrales donde el alumno mantiene un rol pasivo. En contraste, el grupo experimental recibió un enfoque de programación activa mediante ejercicios interactivos, diseñados para fomentar la mejora en su rendimiento y participación.

El procedimiento de intervención inició en el primer momento del semestre Figura 1. En esta fase inicial, se administró una primera prueba diagnóstica diseñada con un nivel de dificultad básico, puesto que los estudiantes eran nuevos y sus conocimientos previos no habían sido medidos. Dicha prueba estaba compuesta por siete preguntas abiertas y 21 tipo icfes. La calificación de estas preguntas abiertas se efectuó de manera manual para permitir un análisis detallado de cada respuesta. Los temas abordados fueron:

- La sintaxis de Python, la correcta declaración de variables y el uso de estructuras de control como bucles.
- Detección de errores en código y aplicar razonamiento estructurado para resolver problemas, como determinar si un número es primo.
- Proponer soluciones eficientes, como la búsqueda de duplicados en una lista
- Secuencias lógicas para resolver problemas, como la búsqueda eficiente en lista.
- Abstraer un objeto o crear funciones generalizadas e identificar regularidades en secuencias numéricas o en la ejecución de código.

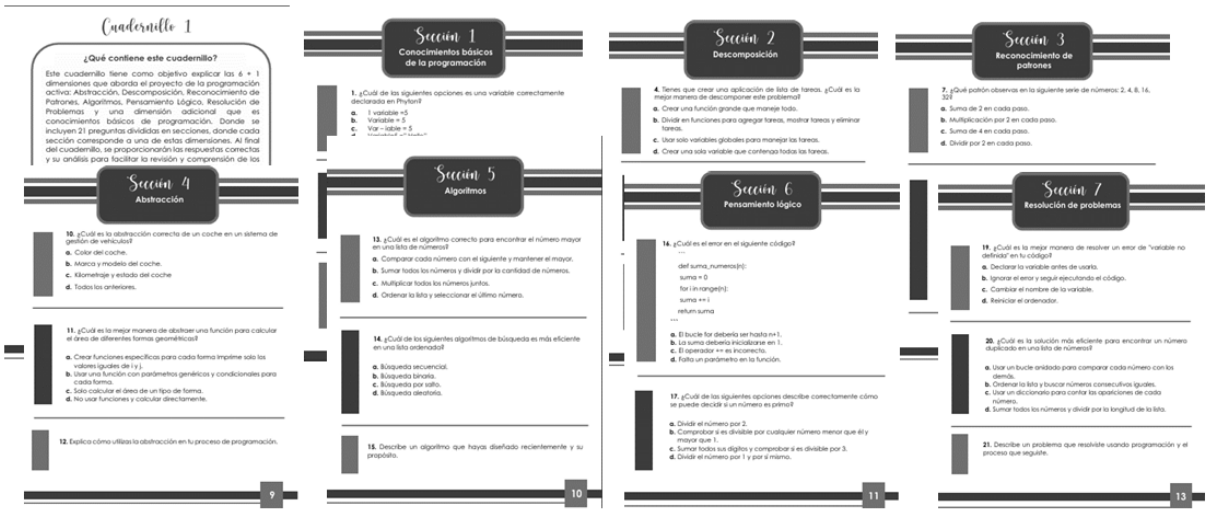

Figura 1: Prueba diagnóstica inicial, semana 1

El segundo momento del procedimiento Figura 2, se llevó a cabo durante las semanas 8 y 9, coincidiendo con la mitad del semestre académico. En esta etapa, se aplicó una segunda prueba diagnóstica (Figura 2) de dificultad intermedia para medir el progreso de los estudiantes en las dimensiones evaluadas. Para este punto, los estudiantes ya habían utilizado la herramienta de programación activa, lo que les permitió desarrollar habilidades clave mediante ejercicios organizados por tema, subtema y nivel de dificultad. Esta prueba se diseñó para examinar cómo los estudiantes abordaban las siguientes tareas:

- Simplificar problemas al centrarse en los detalles relevantes.
- Manejar diferentes operaciones matemáticas mediante funciones genéricas.
- Analizar condiciones y resultados, como en problemas para determinar si dos números son diferentes.
- Realizar la conversión de una cadena a un número entero o la validación de datos de entrada para evitar errores.
- Dividir problemas complejos en funciones específicas, como en el cálculo del área y el perímetro de un rectángulo.
- Diseñar y comprender pasos secuenciales para resolver problemas, como la creación de bucles para iterar sobre una lista.

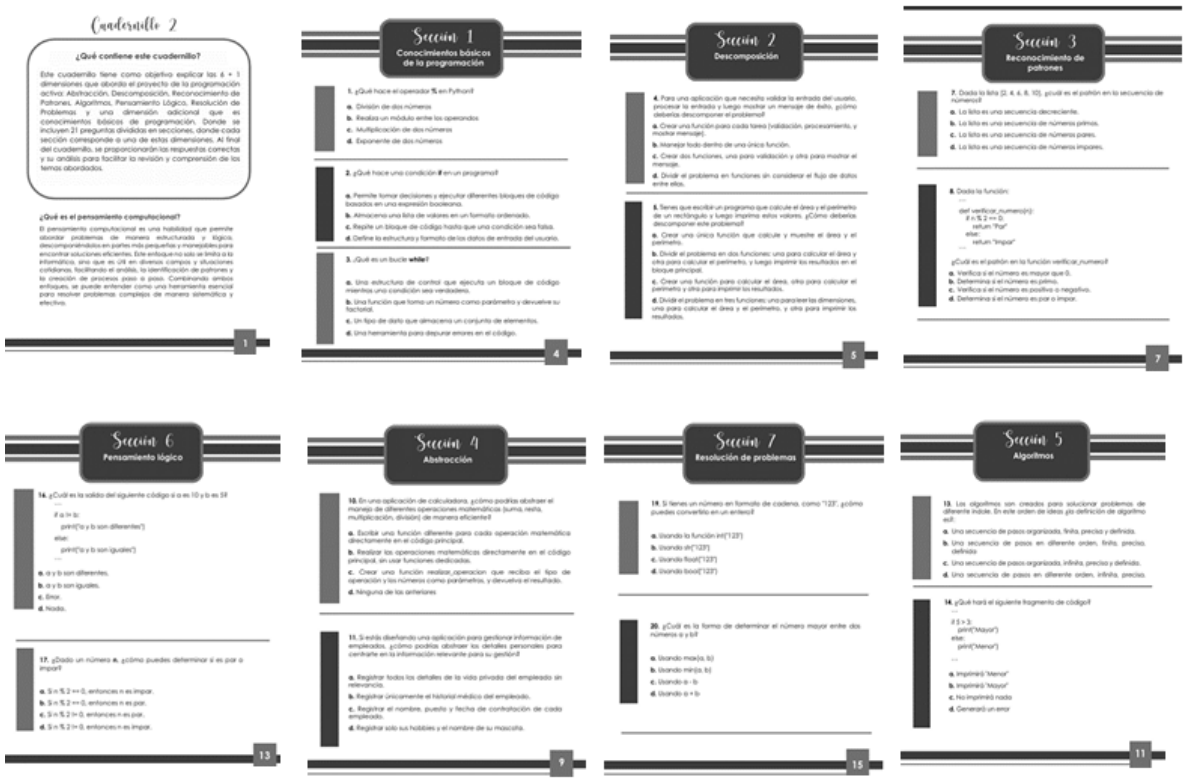

Figura 2. Prueba diagnóstica 2, mitad del semestre, semana 8 y 9

La fase final del procedimiento tuvo lugar entre las semanas 15 y 16 del semestre académico, momento en el cual se aplicó la última prueba diagnóstica (Figura 3). Esta evaluación fue diseñada con un grado de dificultad superior a las anteriores para medir el dominio avanzado de los conceptos.

La prueba se centró en evaluar la capacidad de los estudiantes para:

- Utilizar estructuras fundamentales como listas, operadores lógicos y la instrucción break en bucles.
- Dividir problemas complejos en funciones independientes, aplicado a casos como la creación de un sistema de inventario o el procesamiento de archivos grandes.
- Identificar patrones repetitivos en listas o al sumar elementos bajo condiciones específicas.
- Corregir errores en el código y evitar bucles infinitos, proponiendo soluciones lógicas para optimizar el funcionamiento de los programas.
- Aplicar funciones para simplificar tareas comunes.

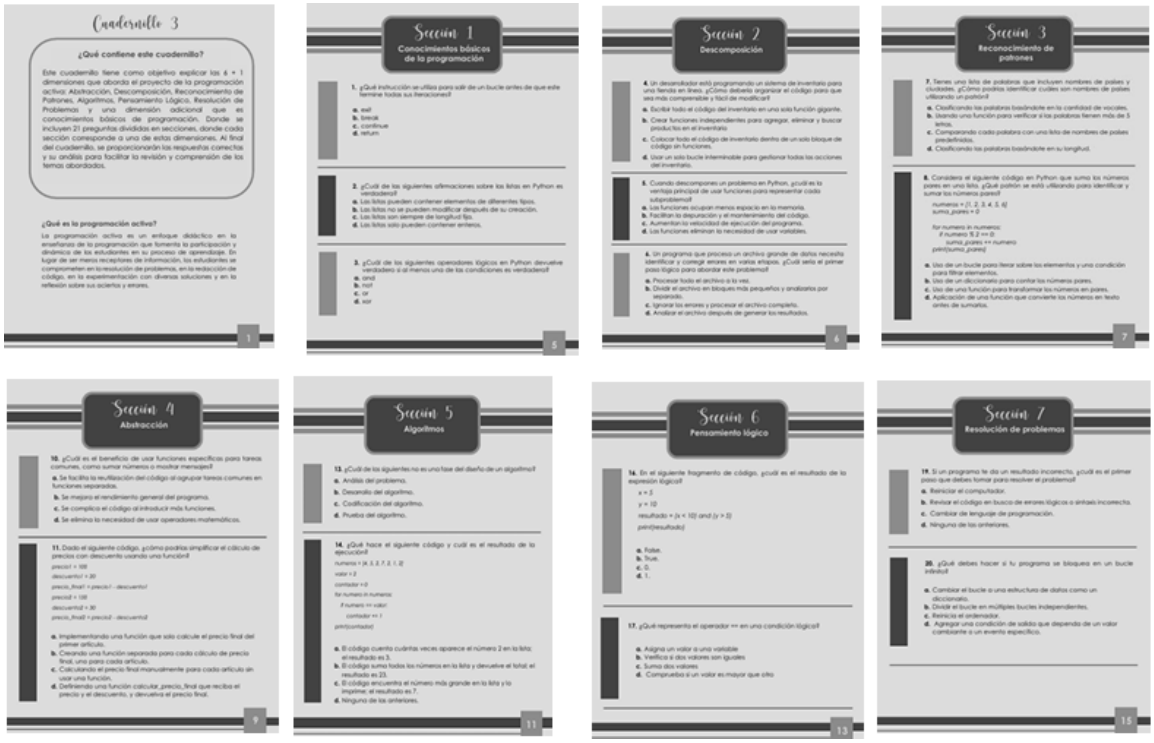

Figura 3. Prueba diagnóstica final, semana 16

Procesamiento y Análisis de Datos

Los resultados de las tres pruebas diagnósticas fueron almacenados en una base de datos estructurada en SQL Server. Para este fin, se diseñaron tablas que contenían la información de los estudiantes, sus respuestas, los puntajes obtenidos en cada dimensión y una nota general. Previo al almacenamiento, se realizó un proceso de limpieza de datos para reemplazar campos vacíos, eliminar registros repetidos y corregir inconsistencias, garantizando así la integridad y confiabilidad de la información.

Para la visualización de los datos, se empleó la herramienta de inteligencia de negocios Power BI. Esta tecnología permite analizar grandes volúmenes de información de manera eficiente. Su compatibilidad con SQL Server facilitó el acceso directo a los datos, ayudando a automatizar la carga y el procesamiento de la información. La plataforma ofreció una amplia gama de opciones para presentar los resultados de forma clara y efectiva, tales como gráficos de barras, líneas de tiempo, matrices y tablas dinámicas.

Herramienta de Aprendizaje Activo

Para la intervención, se trabajó con una herramienta de aprendizaje activo propia (Figura 4) que incorpora más de 1000 ejercicios en Python. El objetivo de la plataforma es enseñar programación de una manera activa y dinámica para aumentar la participación y facilitar un entorno de aprendizaje personalizado.

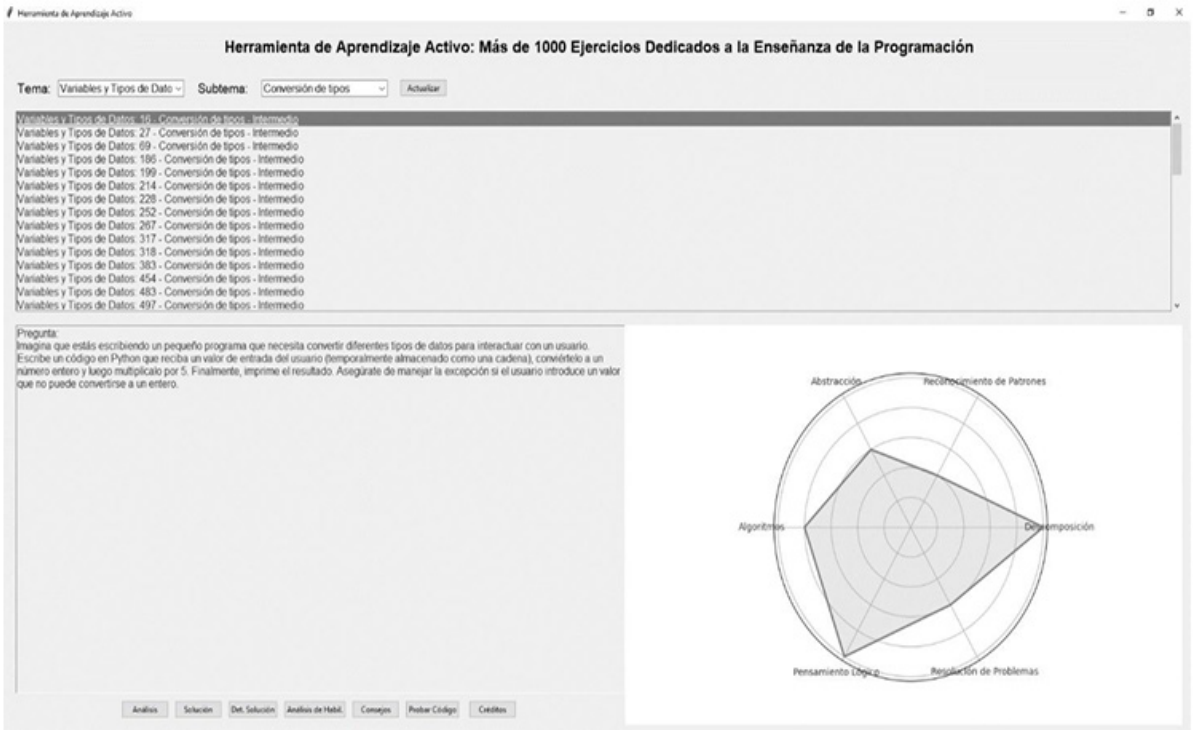

Figura 4. Herramienta de Aprendizaje activo. Desarrollado en el proyecto programación activa en el TdeA

C**aracterísticas y Diseño Pedagógico**

El enfoque pedagógico de la herramienta busca que los estudiantes dejen de ser receptores pasivos de información y se involucren directamente en la resolución de problemas (figura 5), la escritura de código, la experimentación con soluciones y la reflexión sobre sus aciertos y errores. Para lograrlo, la plataforma ofrece:

- Ejercicios didácticos en Python.
- Planteamiento de problemas para su análisis y solución.
- Retroalimentación y desafíos que se adaptan al nivel de conocimiento individual de cada estudiante.

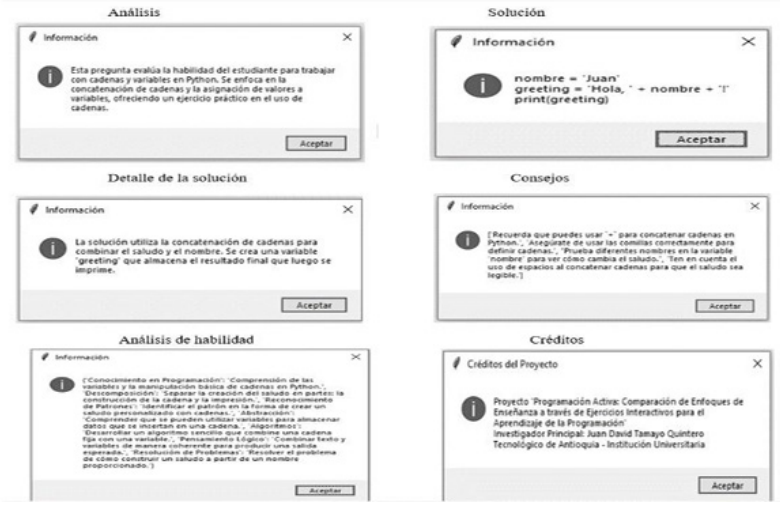

Figura 5: Interacción con la herramienta de programación activa: pistas, consejos, solución, análisis y habilidades desarrolladas

El contenido está organizado en temas y subtemas, lo que permite a los estudiantes navegar fácilmente entre los conceptos clave y abordar cada uno a su propio ritmo (Tabla 2).

Tabla II. Temas y tópicos de la herramienta de aprendizaje activo

TOPIC	*SUBTOPIC*	*DIMENSIÓN*
Variables y Tipos de Datos	Conversión de tipos	**Pensamiento Lógico.** Requiere lógica para manejar distintos tipos de datos y convertirlos de forma adecuada según el contexto.
	Tipos de Datos: int, float, bool, str	**Conocimientos Básicos de Programación**. Estos son fundamentos esenciales que forman parte de la base de la programación.
	Variables y asignación	**Abstracción.** Crear y asignar variables implica representar datos en formas simplificadas.
Cadenas y Métodos	Creación y asignación de cadenas	**Conocimientos Básicos de Programación**. La manipulación de cadenas es un conocimiento fundamental en la programación
	Funciones de cadena: len, split, join, replace	**Algoritmos**. Las funciones de cadena son algoritmos que automatizan tareas comunes con las cadenas.
	Métodos comunes de cadenas: concatenación, slicing, formateo	**Reconocimiento de Patrones**. Se aplican patrones repetitivos en la manipulación y modificación de cadenas.
Estructuras de Control	Bucles Anidados	**Descomposición**. Al dividir tareas complejas en iteraciones más pequeñas y anidadas, se facilita la resolución del problema.
	Bucles basados en patrones	**Reconocimiento de Patrones**. Se detectan patrones repetitivos para automatizar procesos mediante bucles.
	Bucles: for	**Algoritmos**. Los bucles son estructuras clave dentro de los algoritmos que automatizan la ejecución repetida de tareas.
	Bucles: while	**Pensamiento Lógico**. La toma de decisiones basada en múltiples condiciones simultáneas requiere lógica avanzada.
	Condicionales Anidados	**Pensamiento Lógico**. Las estructuras condicionales también dependen de una lógica sólida para evaluar las decisiones.
	Condicionales Mixtos	**Pensamiento Lógico**. Los operadores son herramientas esenciales para el procesamiento lógico y matemático en la programación.
	Condicionales: elif	**Pensamiento Lógico**. Las estructuras condicionales también dependen de una lógica sólida para evaluar las decisiones.
	Condicionales: else	
	Condicionales: if	
	Operadores Aritméticos	**Pensamiento Lógico**. Los operadores son herramientas esenciales para el procesamiento lógico y matemático en la programación.
	Operadores de Comparación	
	Operadores Lógicos	
Funciones	Definición y llamadas a funciones	**Abstracción**. Las funciones permiten encapsular bloques de código, simplificando su reutilización y reduciendo la complejidad.
	Funciones Lambda	**Abstracción**. Las funciones lambda permiten simplificar las definiciones de funciones cortas, contribuyendo a la abstracción.
Estructuras de Datos	Conjuntos	**Abstracción**. Son formas de representar y estructurar datos complejos de manera más sencilla.
	Diccionarios	
	Listas	
	Tuplas	
	Varias estructuras	**Descomposición**. La elección de estructuras adecuadas implica descomponer el problema y seleccionar la mejor forma de representarlo.
Manejo de Excepciones	Captura y manejo de excepciones	**Resolución de Problemas**. El manejo de errores y excepciones forma parte integral de la resolución eficaz de problemas.
	Creación de Excepciones Personalizadas	**Algoritmos**. Crear excepciones específicas permite diseñar algoritmos más robustos y adaptados a los problemas.
Entrada y Salida de Archivos	Lectura y escritura de archivos de texto	**Algoritmos**. La manipulación de archivos sigue un conjunto de instrucciones precisas.
	Trabajo con archivos CSV y JSON	**Descomposición**. Trabajar con formatos de archivo estructurados implica descomponer la información para su correcta manipulación.

Módulos y Paquetes	Creación de módulos personalizados	**Abstracción**. Permite encapsular funcionalidades en módulos, promoviendo la reutilización de código.
	Importación de Módulos	**Conocimientos Básicos de Programación**. Importar y utilizar módulos es un conocimiento esencial para organizar y extender el código.
	Uso de Matplotlib	**Conocimientos Básicos de Programación**. Estas son bibliotecas fundamentales para tareas específicas como visualización, procesamiento numérico, y aprendizaje automático.
	Uso de Numpy Uso de Pandas Uso de scikit-learn	**Conocimientos Básicos de Programación**. Estas son bibliotecas fundamentales para tareas específicas como visualización, procesamiento numérico, y aprendizaje automático.
	Uso de paquetes y gestión con pip	**Conocimientos Básicos de Programación**. El uso de herramientas de gestión de paquetes es un conocimiento esencial para la programación moderna.
Programación Orientada a Objetos	Clases, objetos y atributos	**Abstracción**. La POO permite modelar problemas del mundo real mediante la abstracción de objetos.
	Encapsulamiento	**Abstracción**. Ocultar los detalles internos de una clase refuerza el uso de la abstracción.
	Herencia y Polimorfismo	**Reconocimiento de Patrones**. Al identificar patrones comunes entre clases, se pueden reutilizar y adaptar mediante herencia y polimorfismo.

Arquitectura Técnica y Contenido

La herramienta se desarrolló con una arquitectura MVC en Python para garantizar la escalabilidad y la flexibilidad para futuras actualizaciones.

El conjunto de más de 1000 ejercicios fue diseñado estratégicamente con una dificultad progresiva para promover un aprendizaje continuo. Cada ejercicio está vinculado a una de las siete dimensiones básicas del aprendizaje: Descomposición, Reconocimiento de patrones, Abstracción, Algoritmos, Pensamiento lógico y Resolución de problemas. La Tabla 2 describe en detalle los temas abordados y su correspondencia con cada una de estas dimensiones. La Figura 5 muestra los diferentes módulos de interacción disponibles para el estudiante, como pistas, consejos y análisis de la solución.

Proceso de Generación y Validación de Ejercicios

La creación inicial de los ejercicios se realizó mediante componentes de inteligencia artificial en línea. Este proceso incluyó la integración de la API de ChatGPT de OpenAI para generar el contenido, la automatización a través de scripts de Python y el uso de una base de datos para almacenar la información.

Posteriormente, para garantizar la calidad y pertinencia del contenido generado, se llevó a cabo un riguroso proceso de etiquetado manual. En esta fase, un equipo humano realizó un análisis detallado de cada ejercicio propuesto para verificar si estaba correctamente alineado con el nivel de dificultad, el tópico y el subtópico asignados. En los casos donde se encontraron discrepancias entre el enunciado, la solución y los temas, se documentó la observación del inconveniente para su posterior corrección.

Resultados y Discusión

Los resultados se obtuvieron a lo largo del semestre mediante la aplicación de tres pruebas diagnósticas que evaluaron las dimensiones del pensamiento computacional. Cada dimensión se evaluó con tres preguntas, calificadas con 33.33 puntos si eran correctas, y al final de cada prueba se promediaron las notas para obtener una visión

global del rendimiento.

3.1 Resultados de la Prueba Diagnóstica 1

La primera evaluación sirvió para establecer una línea base de las habilidades de los estudiantes tal como se presenta en la Figura 6.

Figura 6: Prueba diagnóstica 1, diagrama de radar

▪**Mejor desempeño:** Se observó en las dimensiones de Conocimientos Básicos de Programación (0.67) y Descomposición (0.60).
▪**Desempeño moderado:** Se registró en Abstracción (0.56) y Reconocimiento de Patrones (0.55).
▪**Desempeño más bajo:** Las áreas con mayor dificultad fueron Resolución de Problemas (0.49), Algoritmos (0.41) y Pensamiento Lógico (0.32).

3.2 Resultados de la Prueba Diagnóstica 2

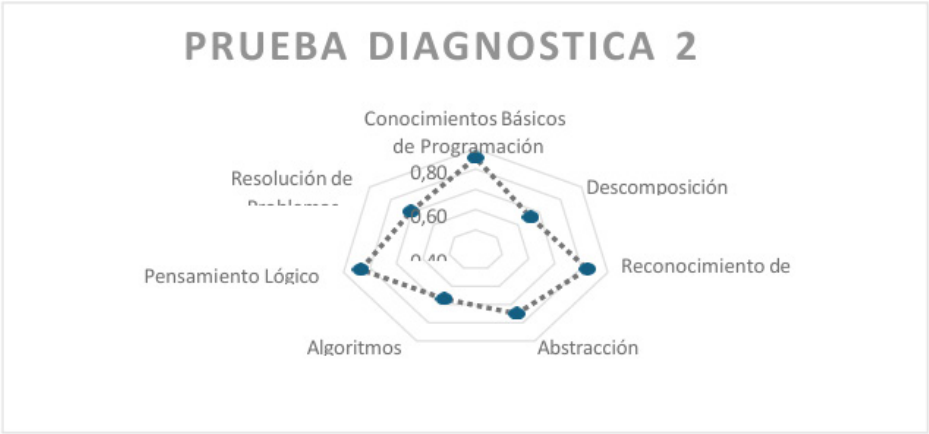

Figura 7: Prueba diagnóstica 2, diagrama radar

Los resultados de la segunda prueba (Figura 7) revelaron un notable progreso en varias dimensiones.

▪Mejor desempeño: Los promedios más altos se encontraron en Conocimientos Básicos de Programación (0.91), Pensamiento Lógico (0.87) y Reconocimiento de Patrones (0.85).
▪Desempeño aceptable: Se mostró en las dimensiones de Abstracción (0.70) y Resolución de Problemas (0.61).
▪Desempeño más bajo: Las áreas que continuaron siendo un desafío fueron Descomposición (0.52) y Algoritmos (0.54).

Los hallazgos muestran una mejora significativa entre la Prueba 1 y la Prueba 2, con diferencias notables en las dimensiones de Pensamiento Lógico y Abstracción, lo que sugiere la efectividad de las intervenciones aplicadas.

3.3 Resultados del Etiquetado Manual de la Herramienta

Se realizó un análisis manual de 1018 (Figura 8) ejercicios generados automáticamente para la herramienta.
▪El 59.86% (813 ejercicios) fueron etiquetados correctamente en su asignación de tópicos.
▪El 27.66% (237 ejercicios) requirieron una reasignación de subtemas.
▪El 12.49% (107 ejercicios) fueron clasificados como incorrectos.
▪Adicionalmente, se modificó la dificultad de 96 ejercicios (11.20% del total) de "intermedio" a "fácil".

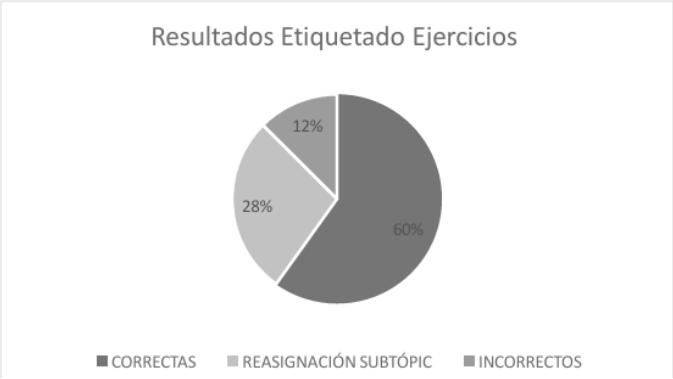

Figura 11: Etiquetado manual de los ejercicios obtenidos de forma automática por medio de la API de CHATGPT

Resultados comparativos de la prueba diagnóstica 1 y 2.

Los hallazgos del estudio muestran una mejora significativa (figura 9) en las dimensiones evaluadas al comparar los resultados de la primera y la segunda prueba diagnóstica. Este progreso indica un desarrollo positivo en habilidades como Resolución de Problemas, Pensamiento Lógico, Algoritmos y Conocimientos Básicos de Programación.
Las diferencias más notables se observaron en las siguientes áreas:

Las diferencias más notables se observaron en las siguientes áreas:

▪**Pensamiento Lógico y Abstracción:** Estas fueron las dimensiones con la mejora más destacada.
▪**Interpretación:** Este notable avance sugiere que las intervenciones aplicadas, como el uso de la herramienta de aprendizaje activo, fueron efectivas para fortalecer estas competencias fundamentales en la programación.

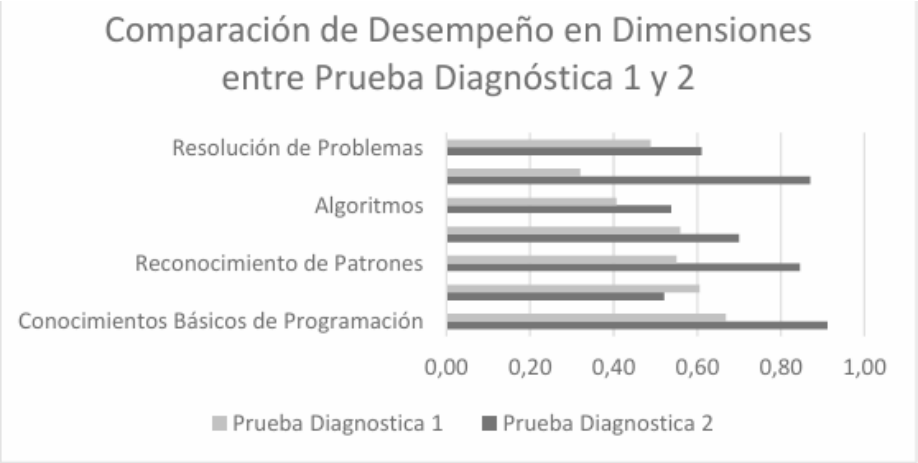

Figura 9: Comparación Prueba diagnóstica 1 vs la prueba diagnóstica 2

El análisis de los resultados es claro sobre el impacto de la intervención con la herramienta de aprendizaje activo. La Prueba Diagnóstica 1 dibuja un perfil de entrada típico para estudiantes novatos: una comprensión básica de conceptos concretos como la programación y la descomposición, pero una marcada debilidad en habilidades de mayor abstracción como el pensamiento lógico, los algoritmos y la resolución de problemas.

El hallazgo más destacado es la drástica mejora en el **Pensamiento Lógico**, que pasó de ser la dimensión con el puntaje más bajo (0.32) a uno de los más altos (0.87) en la segunda prueba. Esto sugiere que el enfoque práctico y la resolución constante de ejercicios interactivos son particularmente efectivos para desarrollar la lógica de programación, una competencia que a menudo es difícil de transmitir con métodos puramente teóricos. La mejora en el Reconocimiento de Patrones refuerza esta idea, indicando que los estudiantes aprendieron a identificar y aplicar soluciones recurrentes. Sin embargo, los resultados también exponen desafíos persistentes y áreas de mejora. Es interesante notar que la puntuación en Descomposición disminuyó ligeramente. Esto podría indicar que, aunque los estudiantes manejaban la descomposición de problemas simples al inicio, encontraron más dificultades al enfrentarse a los ejercicios más complejos presentados por la herramienta, aun cuando sus otras habilidades mejoraban. Por su parte, la dimensión de Algoritmos se mantuvo como una de las más bajas. Esto sugiere que, si bien la herramienta es excelente para la lógica y la sintaxis, el diseño de

algoritmos eficientes puede requerir un enfoque pedagógico complementario que vaya más allá de la práctica de ejercicios.

Conclusiónes

La implementación experimental de la herramienta de programación activa demuestra ser un enfoque eficaz para la enseñanza en entornos universitarios. Se registraron mejoras significativas en las dimensiones del pensamiento computacional evaluadas, con un impacto notable en el Pensamiento Lógico y el Reconocimiento de Patrones, lo que sugiere que este modelo no solo afianza la comprensión teórica, sino que también impulsa la aplicación práctica. Un hallazgo relevante es el potencial inclusivo de la herramienta, la cual fue igualmente efectiva en estudiantes provenientes de diversos estratos socioeconómicos, validando su adaptabilidad a distintos perfiles estudiantiles.
No obstante, el estudio identifica desafíos importantes que requieren atención. A pesar del progreso general, las áreas de Algoritmos y Resolución de Problemas continúan siendo un reto para los estudiantes. Adicionalmente, la formulación automática de ejercicios, que alcanzó un 60% de acierto en su generación, evidencia la necesidad de un refinamiento técnico para mejorar la precisión y reducir la dependencia de la supervisión manual, un paso crucial para la escalabilidad del sistema. Es así que las evidencias sugieren que los modelos de aprendizaje activo, combinados con herramientas interactivas, son una estrategia pedagógica valiosa. Se recomienda su adopción y perfeccionamiento continuo para la enseñanza de la programación en la educación superior.

Agradecimientos

Los autores desean expresar su más sincero agradecimiento al Tecnológico de Antioquia (TdeA) por el espacio y los recursos proporcionados para la realización de este estudio. Asimismo, extienden un especial reconocimiento al grupo de investigación GIISTA y al semillero de investigación Autómata por su valioso apoyo y colaboración en el desarrollo de este proyecto.

Referencias

[1] M. Acurero, M. Pérez y A. Martínez, "Uso de las Tecnologías de Información y Comunicación por parte de los Docentes de Instituciones Educativas de Sucre", *ECONÓMICAS CUC*, vol. 38, Corporación Universidad de la Costa, 2017. https://hdl.handle.net/11323/2327

[2] J. A Zamora-Araya, J. Ramírez-Jiménez, y F. Delgado-Navarro, "Uso de herramientas tecnológicas y su impacto en el rendimiento en el curso de Cálculo II de la Universidad Nacional", *Eco Matemático,* 11 (1), 20-30

[3] T. Jenkins, "On the Difficulty of Learning to Program," in Proceedings of the 3rd Annual

Conference of the LTSN Centre for Information and Computer Sciences, 2002, pp. 53–58. https://www.psy.gla.ac.uk/~steve/localed/jenkins.html

[4] J. Esparza, "La Enseñanza de la Programación en la Universidad: Retos y Perspectivas," *Revista de Educación a Distancia* (RED), no. 44, 2015.

[5] G. Prince, "Does Active Learning Work? A Review of the Research," *Journal of Engineering Education*, vol. 93, no. 3, pp. 223–231, 2004. https://doi.org/10.1002/j.2168-9830.2004.tb00809.x

[6] A. García-Valcárcel, "Innovación en la enseñanza de la programación a través de entornos gráficos interactivos," Pixel-Bit. *Revista de Medios y Educación,* no. 36, pp. 155–170, 2010.

[7] M. Hamada, "Programming Education Based on Active Learning for the Understanding of Concepts," *International Journal of Information and Education Technology*, vol. 5, no. 6, pp. 417–421, 2015.

[8] G. Rueda-Vera, J. A. González-Mendoza, W. R. Avendaño-Castro, "Gestión del riesgo frente a movimientos telúricos en construcción de edificaciones de San José de Cúcuta", *Mundo Fesc*, vol. 11, no. 21, pp. 130-139, 2021.

[9] J. Wing, "Computational Thinking," *Communications of the ACM*, vol. 49, no. 3, pp. 33–35, Mar. 2006. https://dl.acm.org/doi/pdf/10.1145/1118178.1118215

[10] R. Scherer et al., "The Cognitive Benefits of Learning Computer Programming: A Meta-Analysis of Transfer Effects," *Journal of Educational Psychology,* vol. 114, no. 4, pp. 665–684, 2022. https://dl.acm.org/doi/pdf/10.1145/1118178.1118215

[11] D. Weintrop et al., "Defining Computational Thinking for Mathematics and Science Classrooms," *Journal of Science Education and Technology*, vol. 25, pp. 127–147, 2016. DOI: https://doi.org/10.1007/s10956-015-9581-5

[12] S. Grover and R. Pea, "Computational Thinking in K–12: A Review of the State of the Field," *Educational Researcher*, vol. 42, no. 1, pp. 38–43, 2013. DOI: 10.3102/0013189X12463051

[13] C. Bonwell and J. Eison, *Active Learning: Creating Excitement in the Classroom*, ASHE-ERIC Higher Education Report No. 1, 1991. https://eric.ed.gov/?id=ED336049

[14] Y. Pérez, Y. Chirinos, O. Padilla, A. Valdés, A., y E. Cancio, "Programa de superación para perfeccionar el desempeño profesional del docente universitario para el uso de las redes académicas", En *Tendencias en la Investigación Universitaria, Una Visión desde Latinoamérica*. Vol. XXI, Chirinos, Y., Ramírez, A., Godínez, R. Barbera, N. y

Rojas, D. (Eds.) Fondo Editorial Universitario Servando Garcés, 2023. DOI: https://doi.org/10.47212/tendencias2023vol.xxi.6

[15]J. Quintero, L. Orjuela, J. Gordillo y A. Sánchez-Quiñones, "Análisis de implementación del Big Data en empresas y en profesionales de Contaduría Pública en Colombia", *Revista Temario Científico*, 2 (1) 50-60, 2022. https://doi.org/10.47212/rtcAlinin.1.2.5

[16]T. Bell, I. Witten, and M. Fellows, "Computer Science Unplugged: Introducing Computer Science Without a Computer," *The New Zealand Journal of Mathematics,* vol. 31, pp. 117–128, 2002. https://www.researchgate.net/publication/266882704_Computer_Science_Unplugged_school_students_doing_real_computing_without_computers

[17]D. Maloney, E. Peppler, and Y. Kafai, "Programming by Choice: Urban Youth Learning Programming with Scratch," *ACM SIGCSE Bulletin,* vol. 40, no. 1, pp. 367–371, 2008.

[18]M. Resnick et al., "Scratch: Programming for Allí," *Communications of the ACM*, vol. 52, no. 11, pp. 60–67, 2009. https://doi.org/10.1145/1352135.1352260

[19]A. Vihavainen, T. Vikberg, M. Luukkainen, and J. Kurhila, "Massive Open Online Course for Introductory Programming: Experience and the Result," in *Proceedings of the 13th Annual Conference on Innovation and Technology in Computer Science Education,* 2012, pp. 117–122.

[20]S. Papert, *Mindstorms: Children, Computers, and Powerful Ideas.* New York: Basic Books, 1980.

[21]A. Kaleğliolu, "A New Way of Teaching Programming Skills to K-12 Students: Code.org," *International Journal of Academic Research in Education,* vol. 1, no. 1, pp. 1–9, 2015. DOI:10.1016/j.chb.2015.05.047

[22]S. Cakula and I. Sedleniece, "Modelling of Personalized E-Learning Course," *Procedia Computer Science*, vol. 104, pp. 375–382, 2017.

[23]M. Gadanidis et al., "Coding in K–8 Mathematics Education: Why, When and How?," *Canadian Journal of Learning and Technology*, vol. 43, no. 5, 2017. https://www.onted.ca/monographs/what-works/computer-coding-in-the-k-8-mathematics-curriculum

[24]L. Román-González, J. Pérez-González, and C. Jiménez-Fernández, "Which Cognitive Abilities Underlie Computational Thinking? Criterion Validity of the Computational Thinking Test," *Computers in Human Behavior*, vol. 72, pp. 678–691, 2017. DOI: https://doi.org/10.1016/j.chb.2016.08.047

[25] M. Guzdial, "Education: Paving the Way for Computational Thinking," *Communications of the ACM*, vol. 51, no. 8, pp. 25–27, Aug. 2008. https://doi.org/10.1145/1378704.1378713

[26]K. Brennan and M. Resnick, "New Frameworks for Studying and Assessing the Development of Computational Thinking," in Proceedings of the 2012 Annual Meeting of the American Educational Research Association, 2012. https://scratched.gse.harvard.edu/ct/files/AERA2012.pdf

[27]B. Yadav, P. Stephenson, and H. Hong, "Computational Thinking for All: Pedagogical Approaches to Embedding 21st Century Problem Solving in K–12 Classrooms," *TechTrends*, vol. 60, no. 6, pp. 565–568, 2016. https://link.springer.com/article/10.1007/s11528-016-0087-7

[28]F. Ahmad, W. Asif, M. Qamar, and A. Alshamrani, "The Impact of Artificial Intelligence on Student Learning in Programming Education," *Education and Information* Technologies, vol. 28, pp. 2109–2127, 2023.

[29]P. Liao, X. Tang, Y. Wang, and W. Wang, "Instructional Programming Scaffold System based on ChatGPT: Theoretical Foundation and Practice," *Computers and Education: Artificial Intelligence*, vol. 5, 2024.