Desarrollo del pensamiento computacional mediante la programación orientada a objetos y desde los ambientes virtuales de aprendizaje.

Deyvid Mateo Casas Alarcón

Universidad Pedagógica Nacional de Colombia

Facultad de Ciencia y Tecnología, Departamento de tecnología,

Licenciatura en Electrónica

Bogotá, Colombia

2025

Desarrollo del pensamiento computacional mediante la programación orientada a objetos y desde los ambientes virtuales de aprendizaje.

Deyvid Mateo Casas Alarcón

Tesis o trabajo de investigación presentado como requisito parcial para optar al título de: Licenciado en Electrónica.

Director(a):

M.Sc. Nubia Nathaly Sanchez Galvis

Línea de Investigación:

Tecnologías aplicadas a la educación y desarrollo del pensamiento computacional

Universidad Pedagógica Nacional

Ciencia y Tecnología, Tecnología, Licenciatura en Electrónica

Bogotá, Colombia 2025

Resumen

Este estudio atendió la necesidad de fortalecer el pensamiento computacional en la formación de licenciados en electrónica, ante la ausencia de un curso sobre este tipo de pensamiento en el Centro de Innovación, Desarrollo Educativo y Tecnológico (CINNDET) de la Universidad Pedagógica Nacional (UPN). El objetivo general fue desarrollar un curso sobre el pensamiento computacional (PC) mediante la Programación Orientada a Objetos (POO) desde un Ambiente Virtual de Aprendizaje (AVA). Se articuló la metodología Scrum, para el desarrollo del proyecto, junto con la metodología Aprendizaje Basado en Problemas (ABP) y el modelo instruccional de Análisis, Diseño, Desarrollo, Implementación y Evaluación (ADDIE), en cuanto al diseño del curso. El curso en Moodle cuenta con tres módulos que abordan tanto conceptos como metodologías de enseñanza del pensamiento computacional y la POO. Los resultados se recolectaron por medio de formularios que evidenciaron un nivel satisfactorio en la comprensión teórica y aplicación. Sin embargo, se identificaron limitaciones en algunos pilares del pensamiento computacional, como la depuración. El estudio concluye que la integración de un AVA, POO y ABP es una estrategia viable para el desarrollo del pensamiento computacional.

Palabras clave: Ambiente virtual de aprendizaje, Aprendizaje basado en problemas, pensamiento computacional, programación orientada a objetos, Modelo ADDIE.

Abstract

This study addressed the need to strengthen computational thinking in the training of electronics teaching graduates, given the absence of a course on this type of thinking at CINNDET of the National Pedagogical University. The general objective was to develop a course on computational thinking (CT) through Object-Oriented Programming (OOP) within a Virtual Learning Environment (VLE). The Scrum methodology was applied for project development, together with Problem-Based Learning (PBL) and the instructional model of Analysis, Design, Development, Implementation, and Evaluation (ADDIE) for course design. The Moodle-based course consists of three modules that cover both concepts and teaching methodologies of computational thinking and OOP. Results were collected through questionnaires, which showed a satisfactory level of theoretical understanding and application. However, limitations were identified in some pillars of computational thinking, such as debugging. The study concludes that the integration of a VLE, OOP, and PBL is a viable strategy for the development of computational thinking.

Keywords: Computational Thinking, Virtual Learning Environments, Object-Oriented Programming, Problem-Based Learning, ADDIE Model.

Contenido

Pág.

Lista de figuras

Pág.

Lista de tablas

Pág.

Lista de Símbolos y abreviaturas

Abreviaturas

Abreviatura	Término
POO	Programación Orientada a Objetos
ABP	Aprendizaje Basado en Problemas
LMS	Learning Management System
AVA	Ambiente Virtual de Aprendizaje
TIC	Tecnologías de la Información y las Comunicaciones
ADDIE	Análisis, Diseño, Desarrollo, Implementación y Evaluación
CINNDET	Centro de Innovación, Desarrollo Educativo y Tecnológico
SAMR	Substitution, Augmentation, Modification and Redefinition

Introducción

En el marco de la sociedad digital contemporánea, la integración de las Tecnologías de la Información y las Comunicaciones (TIC) en los procesos educativos ha reconfigurado los escenarios de enseñanza y aprendizaje, dando lugar a entornos como los ambientes virtuales de aprendizaje. Estos ambientes, definidos como entornos dinámicos propician desde la interacción, hasta la colaboración e incluso la construcción activa de conocimiento. Durante la práctica educativa realizada en el Centro de Innovación, Desarrollo Educativo y Tecnológico de la Universidad Pedagógica Nacional, se evidenció la ausencia de un curso sobre el pensamiento computacional, competencia esencial en la actualidad porque "este tipo de pensamiento implica un proceso de formulación de problemas y sus respectivas soluciones que pueden ser ejecutadas por un agente que procesa información" *(*Wing, 2006, p. 33*)*.

Su importancia radica en las habilidades que brinda como la descomposición, la abstracción, el reconocimiento de patrones, generalización y depuración, que son indispensables en el análisis de situaciones complejas, en el diseño de sistemas y en la comprensión lógica del mundo digital (Wing, 2006, 2008). Por tal razón, el dominio del PC no debe entenderse tan solo como un valor agregado, sino como una condición necesaria para evitar el analfabetismo digital y, además, como insumo para desenvolverse en una sociedad creativa mediada por la tecnología (Llorens et al., 2017; Resnick, 2008).

Partiendo de esta necesidad formativa, el presente estudio diseñó, implementó y evaluó un curso sobre el pensamiento computacional mediante la programación orientada a objetos y desde un AVA. La propuesta tiene como fundamentos referentes pedagógicos como el constructivismo de Piaget, quien concibe el aprendizaje como una

construcción activa cuando se interactúa con el entorno (Piaget, 1970), y el construccionismo de Papert, quien plantea que entornos como LOGO favorecen la exploración heurística y la resolución de problemas (Papert, 1981), en sintonía con el PC (Wing, 2006, 2008 ,2011). Asimismo, se incorporó la metodología ABP para situar al estudiante como actor principal del proceso de aprendizaje mediante la indagación, colaboración y el pensamiento crítico (Restrepo Gómez, 2005). Así pues, el uso de un AVA en la plataforma Moodle no solo permitió superar barreras de tiempo y espacio, sino que también, de acuerdo con el modelo SAMR (Substitution, Augmentation, Modification y Redefinition) el cual propone, en cuatro niveles, la integración de la tecnología en los procesos educativos (Puentedura, 2010), fue posible redefinir la experiencia de aprendizaje al generar escenarios impensables sin tecnología, potenciando la autorregulación y el autoaprendizaje.

En suma, este trabajo no solo responde a una necesidad identificada en el CINNDET, sino que propone una intervención educativa innovadora que articula un marco teórico robusto, una metodología activa y un entorno tecnológico. El propósito del presente desarrollo fue implementar un curso sobre el PC mediante la POO desde un AVA, y evaluar la comprensión del pensamiento computacional entre los que interactuaron con él, se establecen los siguientes objetivos específicos para su cumplimiento:

1. Sistematizar la teoría del pensamiento computacional en módulos para la implementación del curso.
2. Diseñar contenido educativo desde el enfoque de aprendizaje basado en problemas.

3. Evaluar el grado de comprensión entre quienes interactuaron con el curso y los alcances de este en cuanto al desarrollo del pensamiento computacional.

El presente documento da cuenta de todo el proceso, desde su fundamentación teórica y diseño metodológico, hasta la implementación del curso, el análisis de resultados y la discusión de sus alcances en la formación de los estudiantes de la Licenciatura en Electrónica.

Planteamiento del problema

La formación en pensamiento computacional se ha consolidado como una competencia fundamental en la era digital, no como una habilidad técnica, sino como un marco para la solución de problemas sustentado en procesos como la abstracción, la descomposición, el reconocimiento de patrones y el pensamiento algorítmico (Wing, 2006; Grover y Pea, 2013). Su relevancia es transversal, impactando múltiples disciplinas y siendo demandada por el sector productivo para que se solucionen problemas de diversa índole.

En el contexto de la educación superior, sobre todo en la formación inicial docente, el desarrollo del PC es crucial para que los futuros educadores integren estrategias que favorezcan el aprendizaje en espacios mediados por tecnología (Barr y Stephenson, 2021). Organismos internacionales como la UNESCO (2018) han resaltado la necesidad de incorporar el pensamiento computacional en la formación inicial docente en América Latina para reducir brechas tanto educativas como digitales. No obstante, en Colombia se identifica una limitación estructural: “en los planes de estudio de formación inicial docente no se refleja la enseñanza del pensamiento computacional y sus componentes” (Pérez et al., 2021, p.125).

Esta carencia formativa tiene consecuencias directas. Un estudio del Banco Interamericano de desarrollo (BID) y ProFuturo (2025) reveló que solo el 27% de los docentes en Latinoamérica considera tener un nivel básico en el uso pedagógico de la tecnología. Otra investigación destaca que “una de las principales razones de la ausencia de pensamiento computacional en docentes es la falta de programas o currículos

formales” (Pérez et al., 2021, p. 128). Esta situación contrasta con la de países como Estados Unidos o Reino Unido, que han integrado explícitamente el PC en sus currículos educativos (Wing, 2012).

Fue en el escenario del Centro de Innovación, Desarrollo Educativo y Tecnológico de la Universidad Pedagógica Nacional donde, durante una práctica educativa, se evidenció esta problemática general: la ausencia de un curso formal sobre el PC para estudiantes de licenciatura. Esta observación es un síntoma de la problemática nacional previamente descrita.

Por lo tanto, para contribuir a cerrar dicha brecha, el propósito de este proyecto es desarrollar, implementar y evaluar un curso sobre el pensamiento computacional mediante la programación orientada a objetos en un ambiente virtual de aprendizaje.

Justificación

El presente estudio se justifica ante la urgente necesidad de integrar el PC en la formación inicial docente en Colombia. En un contexto global digitalizado, el pensamiento computacional ha dejado de ser una habilidad técnica opcional para convertirse en un pilar de la alfabetización digital, tan fundamental como la lectura y la escritura (Wing, 2006). Su dominio permite a las personas no solo consumir tecnología, sino analizar y crear soluciones con ella, siendo crucial para reducir brechas digitales y acceder a oportunidades laborales y académicas (Fedesoft, 2023).

Este imperativo global adquiere mayor relevancia en el ámbito de la formación de educadores. Como señalan Pérez et al. (2021), existe una limitación estructural en los planes de estudio de las licenciaturas colombianas, donde el pensamiento computacional brilla por su ausencia. Formar docentes sin esta competencia prolonga de la deficiencia de los aspectos mencionados, impidiendo que las futuras generaciones de estudiantes desarrollen la capacidad de resolver problemas de manera sistemática y eficiente mediante procesos como la descomposición, el reconocimiento de patrones y la abstracción (Grover y Pea, 2013). Por ello, investigar e implementar estrategias para desarrollar el PC en los futuros licenciados no solo es pertinente sino también necesario.

Para atender esta necesidad, la presente propuesta opta por el diseño de un curso en un ambiente virtual de aprendizaje. Esta elección se fundamenta en la idoneidad del entorno digital para fomentar las habilidades propias del PC. Un AVA, como espacio de mediación flexible e interactivo (Castañeda, 2017), permite a los estudiantes interactuar con herramientas de programación, diseñar soluciones iterativas y

trabajar de manera colaborativa, simulando así los entornos problemáticos y digitales en los que luego deberán desempeñarse como docentes. La implementación en un AVA no es solo una cuestión de modalidad, sino una decisión pedagógica alineada con el desarrollo de la competencia digital docente y las propias metas del PC.

Objetivo general

Desarrollar un curso sobre el pensamiento computacional mediante la programación orientada a objetos desde un ambiente virtual de aprendizaje.

Objetivos específicos

1. Sistematizar la teoría del pensamiento computacional en módulos para la implementación del curso.
2. Diseñar contenido educativo desde el enfoque aprendizaje basado en problemas.
3. Evaluar el grado de comprensión entre quienes interactúen con el curso y los alcances de este en cuanto al desarrollo del pensamiento computacional.

Marco teórico y referencial

En este capítulo se constituye el sustento teórico, referencial y conceptual del estudio. Se articula en tres ejes fundamentales: primero, los principios epistemológicos y pedagógicos del constructivismo y el construccionismo (Piaget, Coll, Papert), que legitiman la construcción del conocimiento y el aprendizaje activo; segundo, el marco del pensamiento computacional y su desarrollo mediante el paradigma de la POO, enriquecido con una discusión sobre su relación con el pensamiento crítico y las evidencias más recientes; y tercero, los modelos metodológicos y tecnológicos (SAMR, ADDIE, ABP, E-learning) usados en la implementación del AVA.

Constructivismo y el conocimiento como construcción activa

Jean Piaget (1972), en *Psicología de la inteligencia* afirma que el conocimiento es una reconstrucción activa del sujeto que interactúa con su entorno y no una reproducción pasiva de la realidad. Además, todo pensamiento se compone de una estructura de significaciones por lo que aprender implica desde reorganizar hasta enriquecer continuamente esa estructura. Ahora bien, desde la infancia cada persona desarrolla operaciones cognitivas como la comparación, la organización en el espacio y en el tiempo, la seriación y la clasificación; conformando así la base de la inteligencia lógica. Mientras que estas no se consolidan, surgen las dificultades para comprender nuevos aprendizajes en los estudiantes sobre todo en contextos inéditos. Así pues, el

aprendizaje se produce por medio de la interacción con el medio el cual genera desequilibrios a las nociones previas del estudiantado y permite reequilibrarlas obteniendo nuevas formas de equilibrio cognitivo.

Piaget (1970), en *Seis estudios de psicología* profundiza sobre la idea de que el desarrollo es una constante equilibración, es decir, un tránsito permanente de estados de inferior estabilidad hacia los estados superiores de equilibrio. Aquella dinámica puede ser comparada con la construcción de una torre, cada vez más sólida, o con el ajuste de un mecanismo que aumenta su precisión y movilidad. En este proceso, la inteligencia se transforma de manera gradual: de la acción sensoriomotora a la lógica concreta, y de esta a la abstracción formal, mediada por el lenguaje, la interiorización de la acción junto con la socialización.

A su vez, Jean Piaget (1937) en *La construcción de lo real en el niño*, argumenta que la permanencia del objeto es producto de la acción y la experiencia sensoriomotora y no innata. También, la coordinación de esquemas junto con la interacción con los objetos permite establecer una diferencia entre los estados estables y las apariencias, lo que deriva en la construcción de un universo sólido y coherente. Así, el conocimiento surge de manera progresiva desde la acción práctica hasta la abstracción conceptual, a saber, se aprende haciendo, pero se conoce cuando se reflexiona y aplica lo aprendido en nuevas situaciones.

Psicología del desarrollo y el diseño curricular

César Coll (1987), en *Psicología y currículum* prosigue el enfoque Piagetiano, pero lo proyecta en la práctica educativa. Manifiesta que la psicología del desarrollo constituye un punto de partida legítimo que establece metas educativas, debido a que

cada estadio cognitivos —sensoriomotor, preoperatorio, operaciones concretas y operaciones formales— delimita las posibilidades de razonamiento y aprendizaje.

El autor resalta que todo estudiante inicia un nuevo aprendizaje a partir de sus conocimientos previos, entendidos como representaciones y concepciones que median la interpretación de la realidad. En tal sentido, la diferencia entre el aprendizaje memorístico y significativo radica en la relación entre los nuevos contenidos y los esquemas previos, es decir, cuando el vínculo es substantivo, el aprendizaje se consolida y adquiere funcionalidad; cuando no lo es, queda reducido a mera repetición. En conclusión, hay aprendizaje significativo cuando los esquemas cognitivos se desequilibran para luego reestructurarlos, generando conocimientos sólidos, funcionales y extrapolables.

Constructivismo de Seymour Papert y el pensamiento computacional

Seymour Papert amplía el constructivismo hacia lo que se denomina construccionismo, a saber, la idea de que las personas aprenden mejor cuando construyen productos significativos para sí mismas, como un modelo, un programa o una representación. En Desafío a la mente: computadoras y educación (1981), el autor argumenta que el aprendizaje mediado por computadoras no debe reducirse a dar solo respuestas a problemas puntuales, sino más bien se debe introducir al estudiante en un método heurístico que le permita enfrentar múltiples problemas.

El lenguaje de programación LOGO, desarrollado por Papert, ejemplifica esta visión. LOGO no busca enseñar programación como un fin en sí mismo, sino desarrollar

en los estudiantes habilidades de organización jerárquica, planificación, depuración y reflexión sobre los propios procesos de pensamiento. Estas prácticas coinciden con las habilidades que Wing (2006, 2008, 2011) atribuye al pensamiento computacional: descomposición, abstracción, reconocimiento de patrones, generalización y depuración.

Papert muestra que los entornos digitales y la programación son nuevos objetos de conocimiento con los que el estudiante interactúa de manera activa, prolongando así el constructivismo piagetiano en un construccionismo centrado en la acción sobre objetos digitales.

La relación entre el Pensamiento Crítico y Pensamiento Computacional en la Formación Académica

El pensamiento computacional, constituye un proceso para resolver problemas donde sus soluciones se ejecutan por un agente de procesamiento de información (Wing, 2006). Wing (2006) enfatiza que se trata de “un conjunto de actitudes y habilidades universalmente aplicables que todos, no solo los informáticos, deberían estar ansiosos por aprender y usar” (p.34). Por su parte, Scriven y Paul (2003) definen el pensamiento crítico como “el proceso intelectualmente disciplinado de conceptualizar, aplicar, analizar, sintetizar y/o evaluar información de manera activa y hábil, ya sea recopilada u originada por observación, experiencia, reflexión, razonamiento o comunicación, como guía para la creencia y acción” (p.1).

La relación entre ambos pensamientos no es accidental, sino que representa una sinergia cognitiva estructurada. Las investigaciones han identificado que, tanto el pensamiento crítico como el pensamiento computacional comparten etapas de procesamiento cognitivo que se refuerzan mutuamente. Kannadass et al. (2023)

demuestran empíricamente esta relación, hallando que “el pensamiento crítico para el análisis y toma de decisiones, y el pensamiento computacional permiten superar problemas, comprender mejor las condiciones y expresar resultados” (p.1372). Este hallazgo es significativo pues sugiere que estos dos tipos de pensamiento además de coexistir operan sinérgicamente en la resolución de problemas.

Investigaciones en educación científica respaldan esa sinergia. Ogegbo y Ramnarian (2022), tras haber analizado 23 estudios sobre la integración del PC en aulas de ciencias, concluyen que “el pensamiento computacional promueve habilidades de razonamiento lógico y resolución de problemas que benefician el aprendizaje en prácticamente todos los dominios” (p.206), mientras que el pensamiento crítico proporciona la estructura evaluativa que validan esas soluciones. Kules (2016) articuló que “el pensamiento computacional se complementa con el pensamiento crítico como una forma de razonamiento para resolver problemas, tomar decisiones e interactuar con nuestro mundo” (p.97), extrayendo conceptos como “abstracción, descomposición, diseño algorítmico, generalización, evaluación e iteración de la ciencia informática” (Kules, 2016, p. 98), pero reconociendo que estos conceptos encuentran aplicación más profunda cuando se combinan con la evaluación crítica.

La escala de pensamiento computacional desarrollada por Korkmaz y Xuemei (2019) reconoce esta integración al incluir el pensamiento crítico como una de sus cinco dimensiones, junto con creatividad, pensamiento algorítmico, cooperatividad y resolución de problemas. Este modelo implica que el desarrollo integral del pensamiento computacional no es un proceso puramente técnico, sino que requiere capacidades reflexivas y evaluativas inherentes al pensamiento crítico. Korkmaz y Xuemei (2019)

argumentan que “el pensamiento computacional y el pensamiento crítico son herramientas para que los individuos perfeccionen su creatividad” (p.73), indicando que la creatividad surge precisamente de la interacción entre la lógica computacional y la evaluación crítica.

Por tal razón, en este estudio el pensamiento crítico se conceptualiza como una competencia metacognitiva que complementa y potencia el pensamiento computacional. Mientras el PC proporciona el cómo, el pensamiento crítico provee el por qué y el qué tal. La enseñanza de la POO, con su necesidad de tomas decisiones de diseño fundamentadas sobre abstracción, herencia y encapsulamiento, se convierte en un escenario ideal para el desarrollo integrado de ambas formas de pensamiento. La relación sinérgica entre el PC, el pensamiento crítico y la POO no es tan solo una postulación teórica; ha sido objeto de estudio y validación en investigaciones que se sintetizan en el siguiente marco referencial.

Marco Referencial: Pensamiento Computacional y Programación Orientada a Objetos

El pensamiento computacional se ha consolidado como una competencia fundamental en la educación contemporánea, sobre todo en el dominio de las ciencias de la computación. La POO surge como un paradigma pedagógicamente privilegiado para su desarrollo porque propicia la construcción de abstracciones complejas y la descomposición sistemática de problemas. En este apartado se articulan algunas evidencias investigativas recientes que fundamentan esta relación conceptual.

La POO como medio que Desarrolla el Pensamiento Computacional

Investigaciones recientes documentan la efectividad de la POO como medio pedagógico para desarrollar el PC de manera integrada y progresiva. Krügel y Hubwiesser (2017) diseñaron un Curso en Línea Masivo Abierto (MOOC) titulado "*LOPP: Learning Object Oriented Programming"* que incorporaba una introducción formal en programación. Los autores indican que existe

> Un dilema fundamental en la enseñanza de POO: por una parte, los enfoques pedagógicos modernos postulaban enseñar en contextos del mundo real, es decir, se podría exigir demasiado a los estudiantes, pues deberían aprender un número enorme de conceptos nuevos, parcialmente muy difíciles, simultáneamente (Krügel y Hubwiesser, 2017, p. 1).

Para resolver ese dilema se implementó una estrategia de objetos primero, distribuyendo los objetivos de aprendizaje a lo largo del curso de tal modo que evitara enfrentar a los estudiantes con una gran cantidad de conceptos desconocidos cuando escribían su primer programa (Krügel y Hubwiesser, 2017). El estudio reportó que "el pensamiento computacional, conceptualizado como una capacidad personal universal, puede ser integrado como elemento constitutivo de cualquier disciplina, y particularmente resulta beneficioso cuando se acopla a problemas auténticos que resalten su utilidad profesional" (Krügel y Hubwiesser, 2017, p. 5). Esta aproximación redujo considerablemente la carga cognitiva y facilitó la comprensión de conceptos abstractos fundamentales como encapsulación, polimorfismo y herencia.

Recientes investigaciones sobre el PC y la POO

Malik et al. (2025) desarrollaron la aplicación OPP-SOLVE, una herramienta que operacionaliza el pensamiento computacional en el dominio de la POO a través del pseudocódigo. Aquel estudio, se realizó con una muestra de 224 estudiantes en cursos de POO, demostró que la aplicación promovía el pensamiento algorítmico y las capacidades de resolución de problemas. Según los autores

> Las metodologías de enseñanza tradicionales frecuentemente dedican más tiempo a la enseñanza de la sintaxis de programación que a otras habilidades requeridas en el dominio de la programación, lo que resulta en estudiantes que están familiarizados con la sintaxis y semántica de la programación, pero que no son capaces de utilizar estas reglas al escribir un programa válido (Malik et al., 2025, p.4).

La aplicación OPP-SOLVE aborda este déficit al enfatizar en el pensamiento algorítmico y las estrategias de resolución de problemas, esto le permite a cada estudiante enfocarse en el análisis del problema y el diseño del programa mientras se abstraen temporalmente de la sintaxis del lenguaje.

Por otro lado, Herlambang y Rachmadi (2024) en el contexto de la enseñanza de programación informática en educación vocacional, produjeron varias conceptualizaciones relevantes que enriquecen la comprensión del PC en el dominio de la POO.

Los autores concluyen que el PC constituye “un conjunto de habilidades requeridas en el proceso de resolución de problemas en programación informática, siendo no solamente un proceso de pensamiento sino una expresión de habilidades que

otros pueden observar” (Herlambang y Rachmadi, 2024, p. 6). Esta caracterización es significativa porque desplaza el concepto de PC del plano meramente cognitivo-abstracto hacia una dimensión ejecutable y observable, materializada en competencias demostrables cuando se operacionaliza mediante la POO (Herlambang y Rachmadi, 2024).

La investigación identificó que conceptos que influencian de manera decisiva al proceso de dominio del PC: las fuentes de información y las estrategias de guía. Lo que sugiere que el desarrollo efectivo del PC no es un proceso espontáneo, sino que requiere de intervenciones pedagógicas intencionales, estructuradas y con acceso a recursos informáticos pertinentes (Herlambang y Rachmadi, 2024).

Zeng et al. (2023) desarrollaron un marco tridimensional para comprender el pensamiento computacional que articula tres componentes interrelacionados: conceptos computacionales, prácticas y perspectivas computacionales. El primero, secuencia instrucciones, condicionales, bucles, representación de datos y hardware/software; el segundo, por su parte, consiste en el diseño algorítmico, reconocimiento de patrones, abstracción, depuración, descomposición, iteración y generalización; finalmente, el tercer componente se relaciona con la expresión y creación, conexión, perseverancia y elecciones conductuales.

La POO, cuando se enseña con énfasis en el PC, promueve de manera integrada estos tres componentes. Los conceptos computacionales se materializan en estructuras de control dentro de métodos; las prácticas computacionales se desarrollan tras modelar sistemas complejos mediante abstracciones de clases y objetos; y las perspectivas computacionales se cultivan al abordar retos de diseño que requieren perseverancia,

decisiones deliberadas y comunicación efectiva de soluciones (Herlambang y Rachmadi, 2024).

El Pensamiento Computacional en Colombia: Estudios desde la Universidad Pedagógica Nacional

En el contexto institucional la Universidad Pedagógica Nacional educadora de educadores ha desarrollado estudios relacionados con el presente proyecto. La investigación de Pinzón y Paredes (2022) demostró la efectividad de las actividades conectadas y desconectadas para desarrollar habilidades fundamentales del PC, como la secuenciación y la depuración, evidenciando que su desarrollo va más allá de la herramienta porque recae en el diseño pedagógico. Por otro lado, Quevedo (2025) profundizó en la relación que hay entre el PC y las matemáticas, diseñando un AVA que logró mejoras del 72% en pensamiento algorítmico y del 65% en resolución de problemas, con una reducción del 58% en errores de lógica.

Estos antecedentes sientan un precedente crucial para el presente estudio porque muestran la viabilidad y el impacto de los AVA en el desarrollo del pensamiento computacional. Partiendo de esta base, se esbozan las siguientes implicaciones para el diseño de Ambientes Virtuales de Aprendizaje.

Implicaciones para los Ambientes Virtuales de Aprendizaje (AVA)

La articulación entre Piaget, Coll y Papert permite fundamentar el uso de un Ambiente Virtual de Aprendizaje (AVA) como medio para la construcción activa de

conocimiento en pensamiento computacional. Un AVA no transmite información de forma pasiva; por el contrario, ofrece actividades, problemas, y recursos que provocan desequilibrios cognitivos y favorecen la reorganización de esquemas.

En este contexto, los principios de Piaget explican cómo se construye el conocimiento a partir de la interacción con situaciones de aprendizaje; Coll aporta la noción de aprendizaje significativo y su relación con los conocimientos previos; y Papert introduce la tecnología y la programación como nuevos espacios de construcción. En este sentido, un AVA basado en estos referentes puede convertirse en un laboratorio de aprendizaje donde cada estudiante descompone problemas, genera hipótesis, diseña algoritmos, experimenta, corrige y consolidan aprendizajes significativos sobre la programación orientada a objetos.

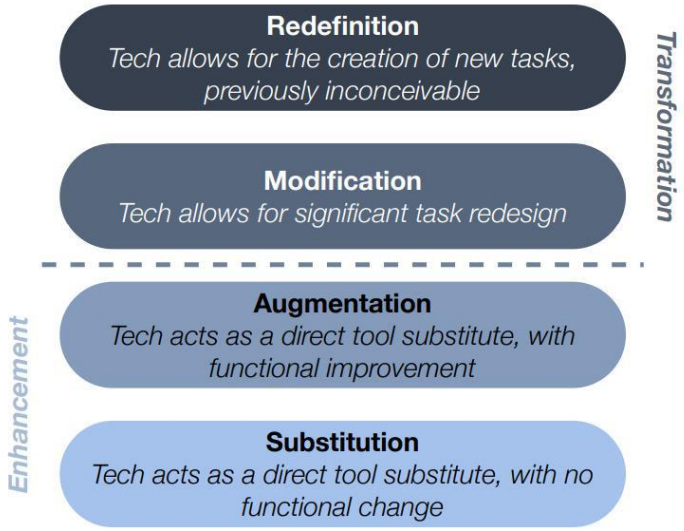
En suma, la perspectiva piagetiana del conocimiento como construcción activa, complementada por el enfoque curricular de Coll y el construccionismo de Papert, proporciona un marco teórico robusto para justificar la creación de un curso de pensamiento computacional en un AVA, basado una metodología activa como lo es el aprendizaje basado en problemas.

Modelo SAMR

Puentedura (2010), en *SAMR and TPCK: Intro to Advanced Practice* presenta un modelo que ayuda a los docentes en la evaluación sobre la manera en la que incorporan la tecnología en el aula y así, conocer cuáles son los usos que tienen mayor o menor incidencia en el aprendizaje de cada estudiante. Este modelo consiste en un conjunto de cuatro niveles ordenados jerárquicamente junto con dos capas que describen el uso de

las herramientas tecnológicas. A continuación, en la figura 1 se ilustra lo anteriormente mencionado.

Figura 1.
Modelo SAMR.



Nota. Fuente: Tomado de https://hippasus.com/resources/sweden2010/SAMR_TPCK_IntroToAdvancedPractice.pdf#page=3.00

Dicho modelo propone como se debe usar la tecnología para que, sea posible redefinir la experiencia de aprendizaje en escenarios imposibles de pensar sin la tecnología.

Marco conceptual

Pensamiento computacional

"La sociedad demanda nueva ciencia y tecnología. Una habilidad necesaria en la innovación tecnológica es el pensamiento computacional en tanto aporta a la solución de problemas, diseña sistemas y estructuras sobre la comprensión humana" (Wing, 2006, p. 33). Adicionalmente, el pensamiento computacional reformula un problema aparentemente difícil en uno ya conocido para resolverlo por medio de la transformación, reducción, asignación o simulación del problema en cuestión (Wing, 2006, 2008).

El pensamiento computacional es fundamental porque contribuye no solo a la solución de problemas sino también al desarrollo de habilidades matemáticas, convirtiéndolo en una competencia clave en múltiples ámbitos (Wing, 2006). Además, influye en múltiples disciplinas o campos del saber debido a la forma en la que soluciona problemas, tanto así que diferentes empresas y departamentos requieren de talento humano capacitado para resolver problemas de diversa índole mediante el uso del pensamiento computacional (Wing, 2006).

Ahora bien, el pensamiento computacional está presente en la vida cotidiana, debido al uso frecuente de términos como algoritmo, método, modelo y sistema. Esta

práctica demuestra que sus beneficios no se limitan tan solo al ámbito científico, sino que constituye una serie de habilidades que son valiosas para cualquier persona. Además, se proyecta como una competencia esencial en el siglo XXI (Wing, 2006).

Pero y ¿qué es el pensamiento computacional? Pues bien, Wing (2011) en *Research Notebook: Computational Thinking What and Why* define al pensamiento computacional como: “El pensamiento computacional es el proceso de pensamiento involucrado en la formulación de problemas y soluciones, de manera que las soluciones se representan en una forma que pueda ser ejecutada eficazmente por un agente de procesamiento de información”.

Así pues, según esta definición la agenda investigativa de diversas ciencias e ingenierías se ha visto influenciada por el PC dado que su esencia radica en la abstracción, considerada un pilar clave para su desarrollo. Las abstracciones son herramientas mentales de computación útiles para científicos e ingenieros y usadas en varios campos del saber (Wing, 2008, 2011).

El PC puede considerarse como una forma, avanzada, de pensamiento analítico que integra elementos del pensamiento algorítmico junto con el paralelo. Esta perspectiva lo vincula con el concepto algoritmo, entendido como un proceso de abstracción que toma entradas, ejecuta pasos secuenciales y produce salidas con el fin de lograr un objetivo (Wing, 2008).

Ahora bien, Wing (2006, 2011) en sus artículos *Computational thinking* y *Research Notebook: Computational Thinking What and Why* determina las características y beneficios del pensamiento computacional de la siguiente manera:

- Conceptualizar más no programar.

- Fundamentar en vez de habilidades de repetición.

- Una forma en la que piensan los humanos, no las computadoras.

- Complementa y combina el pensamiento matemático e ingenieril.

- Ideas a cambio de artefactos.

- Para todos en todas partes.

- Entiende qué aspectos de un problema son susceptibles de ser computables.

- Evalúa la combinación entre herramientas computacionales, técnicas y un problema.

- Entiende las limitaciones junto con el poder de las herramientas computacionales y técnicas.

- Aplica o adopta una herramienta o técnica computacional a un nuevo uso.

- Identifica oportunidades para usar la computación de una nueva forma.

- Aplica estrategias computacionales como dividir y superar en cualquier dominio.

Debido a la influencia del pensamiento computacional en distintos campos del conocimiento y, a su carácter como desafío educativo, se considera pertinente diseñar un curso e incluso varios, que aborden sus conceptos fundamentales. Esta distinción ha motivado diferentes sectores a promover su aprendizaje por medio de programas formativos y herramientas específicas (Wing, 2008, 2011).

Empero, el pensamiento computacional modifica sustancialmente la manera en que se estudia, analiza y comprende diferentes áreas, debido a su enfoque en la resolución de problemas. Esta característica justifica la necesidad de no solo diseñar cursos que fortalezcan su desarrollo sino también de formar docentes competentes que lo implementen en el aula. Además, esta competencia contribuye al modelamiento de sistemas complejos e incluso al análisis de grandes volúmenes de datos (Wing, 2008).

Pensamiento crítico

Para los fines de este proyecto, se conceptualiza como el proceso metacognitivo de evaluación, razonamiento fundamentado y análisis que enriquece además de acompañar al PC. Se manifiesta en la capacidad de cuestionar supuestos, evaluar la idoneidad de las abstracciones creadas, juzgar la eficiencia de los algoritmos diseñados y reflexionar sobre el proceso de resolución de problemas en su conjunto, derivado de los principios de reequilibración cognitiva (Piaget, 1970), aprendizaje significativo (Coll, 1987) y depuración reflexiva (Papert, 1981).

Programación orientada a objetos

Como paradigma de desarrollo de software la POO modela sistemas complejos mediante objetos que encapsulan comportamientos y datos, promoviendo la claridad estructural, la modularidad y la reutilización. Se fundamenta en la idea de que cada componente del sistema es un objeto que interactúa con otros a través del envío y recepción de mensajes, manteniendo su estado interno y ocultando su implementación. Entre sus pilares se encuentran la abstracción, que permite definir clases para

representar entidades con métodos y atributos; la herencia, que facilita la reutilización y extensión de funcionalidades mediante jerarquías; y el polimorfismo, que otorga flexibilidad al permitir que objetos de diferentes clases respondan a una misma interfaz. Además, la POO se puede entender como una metodología basada en el diseño por contrato, donde cada objetos colabora mediante acuerdos que definen sus responsabilidades y garantías, es decir, no se limita a una estructura sintáctica, sino que también involucra un conjunto de prácticas de diseño limpio y modular, donde la organización coherente de clases y responsabilidades es clave para el mantenimiento y escalabilidad del software (Kay, 2003; Martin, 2003; Meyer, 1997; Stroustrup, 1988, 1991; Wegner, 1990).

Método

Los métodos se definen dentro de una clase con el objetivo de ejecutar instrucciones que modifican el estado de un objeto y manipulan su información. Es decir, un método representa o describe el comportamiento de algún objeto en particular y, hay dos tipos: los que retornan valores y los que no, ambos son invocados desde el 'main' o método principal e incluso pueden llamarse dentro de otro método (Aguilar, 1996; Ávila & Bailón, 2023; Berges, 2015; López, 2013).

Atributo

Los atributos son las características de los objetos, es decir, almacenan datos e información que describen el estado de una instancia. Cada atributo consta de un nombre y un valor (Aguilar, 1996; Ávila & Bailón, 2023; Cerón, s. f.; Moreno, s. f.).

Clase

Una clase es un modelo e incluso una abstracción que representa o describe un conjunto de objetos que comparten una misma estructura o comportamiento, es decir, es una plantilla que define la información sobre los objetos y, además, permite instanciarlos. En otro orden de cosas, una clase contiene los atributos y método de un objeto, a saber, establece las características compartidas que serán heredadas por cada instancia (Aguilar, 1996; Ávila & Bailón, 2022; Berges, 2015; López, 2013).

Objeto

La programación orientada a objetos se fundamenta en la representación de problemas reales a través de los objetos, que actúan como abstracciones de elementos del mundo físico o conceptual. Estos objetos combinan desde datos hasta comportamientos debido a que poseen atributos, quienes definen su estado y métodos que permiten modificarlos. Cada objeto corresponde a una instancia de clase, compartiendo no solo las características sino también reglas comunes con otros objetos del mismo tipo (Aguilar, 1996; Berges, 2015; Cerón, s. f.; López, 2013).

Herencia

La herencia constituye, en la POO, un principio que posibilita la creación de nuevas clases a partir de otras definidas previamente, esto permite que las subclases hereden atributos y métodos de sus superclases. Este mecanismo favorece la reutilización de código junto con la extensibilidad, además de facilitar la modificación y adaptación de funcionalidades en clases derivadas (Microsoft, 2025; OpenWebinars, 2023).

Encapsulamiento

Es un principio de la POO que agrupa datos y métodos relacionados dentro de una misma estructura, ocultan la implementación interna y restringe el acceso directo a los componentes del objeto. Este mecanismo permite que solo los métodos públicos específicamente definidos puedan manipular o acceder al estado interno de la clase, proporcionando control y seguridad sobre los datos (Escuela Directa, 2021; OpenWebinars, 2023).

El encapsulamiento favorece la modularidad, reutilización del código y facilita el mantenimiento al reducir el acoplamiento entre componentes (NetMentor, 2025; Portal Académico CCH-UNAM, 2022).

Polimorfismo

El polimorfismo es considerado el tercer pilar de la Programación Orientada a Objetos, después del encapsulamiento y la herencia. La palabra polimorfismo proviene del griego y significa "que posee formas diferentes", siendo uno de los conceptos esenciales de la POO. Es decir, el polimorfismo se define como "la propiedad por la que es posible enviar mensajes sintácticamente iguales a objetos de tipos distintos". Es la capacidad que tiene un objeto de tomar distintas formas, permitiendo que una referencia a una clase acepta direcciones de objetos de dicha clase y de sus clases derivadas (Institución Universitaria Politécnico Grancolombiano, 2025; Microsoft, 2025; Moncho, 2016; Solano & Sánchez, 2021).

El polimorfismo permite que objetos de diferentes clases puedan ser tratados como objetos de una clase base común, especialmente en parámetros de métodos, colecciones o matrices. En tiempo de ejecución, se invoca la implementación específica del método según el tipo real del objeto, proceso conocido como enlace dinámico (Microsoft, 2025; OpenWebinars, 2023).

Metodología y modelo del curso

Aprendizaje Basado en Problemas

Es una metodología activa que se enfoca en el estudiante. El aprendizaje sucede por medio de la resolución de problemas relevantes y auténticos que demandan la adquisición y aplicación de conocimientos de manera no solo autónoma sino también colaborativa. En esta metodología se promueve el desarrollo de habilidades de indagación, análisis y trabajo en equipo, posicionando al alumno como actor principal en la construcción de su propio conocimiento. El problema funciona como motor del proceso formativo porque genera preguntas que guían la exploración constante y la integración de saberes. El ABP crea un entorno significativo que articula teoría y práctica, apoyado en fundamentos psicológicos que favorecen el aprendizaje. Además, esta metodología fomenta competencias profesionales mediante prácticas evaluativas que estimulan desde la reflexión, hasta la autoevaluación e incluso la responsabilidad sobre el propio proceso formativo. En contextos educativos reales, el ABP fortalece habilidades para enfrentar problemas de manera similar a como se hará en la vida profesional, siendo clave el rol del docente como facilitador que guía tanto el razonamiento, como la búsqueda de información y la formulación de preguntas para profundizar en el análisis (Barrows, 1986; Donovan & Boud, 2012; Luy-Montejo, 2019; Savery, 2006; Schmidt, 1983).

Modelo ADDIE

El modelo instruccional ADDIE tiene un enfoque sistemático y estructurado que organiza el diseño de cursos en cinco fases: análisis, diseño, desarrollo, implementación y evaluación. Ahora bien, este modelo sigue una secuencia lógica e iterativa, es decir, la salida de cada fase es el punto de partida de la siguiente para mejorar continuamente la instrucción. Como metodología secuencial y rigurosa asegura desde el análisis de necesidades, la elaboración de estrategias y materiales, hasta su desarrollo, aplicación en contextos reales e incluso una evaluación para optimizar el aprendizaje. Asimismo, su marco estructurado garantiza coherencia y organización en el diseño de recursos formativos, por lo que, como herramienta crea y perfecciona programas educativos mediante una evaluación constante de la calidad del aprendizaje. Por otro lado, desde la perspectiva didáctica el modelo ADDIE se asegura una planificación eficiente y contextualizada de la formación docente debido a su flexibilidad y aplicabilidad en el desarrollo de módulos e-learning, al permitir tanto evaluaciones formativas como sumativas a lo largo del proceso. En suma, ADDIE es un modelo integral, adaptable y orientado a la mejora continua de la enseñanza y el aprendizaje (Branch, 2009; Dick & Carey, 1978; Molenda, 2003; Morales González, 2022; Patel et al., 2018; Reigeluth, 1983; Smith & Ragan, 1999).

E-learning

El e-learning es una modalidad educativa que emplea medios y dispositivos electrónicos que facilita el acceso, la evolución y la mejora de la calidad educativa porque puede integrarse de manera total o parcial en los procesos formativos. Además, Forma

parte de la educación a distancia debido a que implementa tecnologías que permiten métodos síncronos y asíncronos, apoyados en diversos recursos como video, sonido o animaciones con el fin de presentar la información y enriquecer los materiales didácticos. Así pues, al estar mediado por internet, elimina barreras espaciotemporales, promueve una formación flexible y sitúa al estudiante en el centro del proceso, quien debe autogestionar su aprendizaje con el acompañamiento de tutores y compañeros. Asimismo, se apoya en las TIC para favorecer el aprendizaje en contextos presenciales, semipresenciales o a distancia, consolidándose como una modalidad adoptada por los sistemas educativos para ofrecer una formación académica flexible, accesible y coherente con los avances tecnológicos (Álava, 2022; Cabrera et al., 2011; García, 2015; Jara, 2023; Mujica, 2020).

Plataformas E-learning

Las plataformas e-learning como Learning Management System (LMS), por ejemplo, Moodle, son espacios que propician el desarrollo de habilidades del PC y la reflexión sobre los propios procesos de aprendizaje. Para enfrentar problemas complejos y transformarlos en soluciones que se puedan procesar de forma computable se requiere el PC. En este sentido, las plataformas e-learning se integran de manera natural con metodologías activas como el ABP, en las que dar solución al problema planteado funciona como motor del proceso formativo. Así pues, un AVA no se limita a la simple transmisión de contenidos, sino que plantea desde actividades, desafíos hasta recursos que generan desequilibrios cognitivos y estimulan la reorganización de esquemas previos, lo que propicia la construcción activa del conocimiento.

El modelo instruccional ADDIE, organiza el proceso educativo en cinco fases que son: análisis, diseño, desarrollo, implementación y evaluación. Esta secuencia asegura la

coherencia, pertinencia contextual junto con un ciclo de mejora continua, aspectos que son valiosos en la elaboración de módulos.

Por otro lado, aportes como el constructivismo de Piaget, el aprendizaje significativo de Coll y el construccionismo de Papert conforman un marco teórico solido que respalda la incorporación de plataformas e-learning en la enseñanza del PC. Ahora bien, estas perspectivas sostienen que el conocimiento no se adquiere de manera pasiva, sino que se construye activamente mediante la interacción con el entorno, la reestructuración de esquemas previos y la producción de artefactos. En suma, un AVA bien diseñado puede convertirse en un laboratorio de aprendizaje, donde cada estudiante experimenta, reflexiona y consolida saberes que se aplican en múltiples contextos.

Metodología

Si bien el modelo ADDIE proporciona una estructura solida en cuanto al diseño instruccional, su carácter secuencial puede limitar la capacidad de respuesta ante ciertos cambios durante el desarrollo del curso. Por tal razón, se integró la metodología ágil Scrum como estrategia de gestión del proyecto, lo que permite trabajar en iteraciones, realizar ajustes continuos con el propósito de garantizar la calidad del producto final. Esta articulación favoreció la implementación del curso en la plataforma Moodle por medio de estrategias parciales y revisiones periódicas, asegurando así que el diseño instruccional se mantuviera alineado con los objetivos pedagógicos y las necesidades detectadas.

Implementación de Scrum

El desarrollo de este estudio fue posible debido al uso de Scrum como metodología de gestión, dado su carácter iterativo e incremental favorece tanto la organización sistemática de las tareas, como la calidad del proceso investigativo. Esta decisión permitió manejar el proyecto con un enfoque flexible y adaptativo, en coherencia con los principios de las metodologías ágiles aplicadas en contextos académicos (Salazar, 2020; UNIR, 2023).

En el marco de las metodologías ágiles, Scrum resalta por promover ciclos de trabajo breves y continuos que facilitan la entrega temprana y la mejora de productos. Esta dinámica asegura que los resultados se ajusten a las necesidades del contexto, convirtiéndose en un enfoque pertinente para proyectos educativos e investigativos.

Consiste en la gestión de proyectos mediante ciclos denominados sprints, en los que se planifica, desarrolla y entrega de manera continua un producto funcional, esto

permite desde un mayor control, hasta flexibilidad e incluso capacidad de adaptación a las nuevas necesidades. Su estructura promueve la colaboración y la comunicación efectiva entre equipos, enfatizando la autoorganización, las reuniones de seguimiento y las revisiones periódicas que garantizan tanto calidad como la coherencia con los objetivos e incluso la mejora continua del proceso de desarrollo (Andrade, 2023; Asana, 2025; Salazar, 2020; UNIR, 2023).

El desarrollo del presente estudio consideró los principios éticos en investigación educativa, es decir, se garantizó el consentimiento informado de cada participante, la confidencialidad de la información, el uso adecuado de los datos y el respeto por la autonomía de los estudiantes en su proceso de aprendizaje.

Roles Scrum en el contexto investigativo

- El tesista asumió el rol de Scrum master, facilitando el proceso, eliminando impedimentos y asegurando el cumplimiento de los principios ágiles.
- La directora del trabajo actuó como product owner, quien definió los criterios de aceptación y validó los incrementos de investigación.
- El tesista fue el responsable de ejecutar las actividades de investigación planificadas tomando el rol de development team.

Artefactos Scrum en el proceso investigativo

La gestión del flujo de trabajo se configuró en tres artefactos centrales de Scrum (Product Backlog, sprint e incrementos de investigación), adaptados para el desarrollo del proyecto:

El Product Backlog de investigación constituyó el plan del proyecto. No fue tan solo una lista de tareas, sino una herramienta dinámica y priorizada que agrupó todos los requisitos necesarios para completar la tesis, desde la revisión de literatura hasta el diseño metodológico e incluso la redacción de capítulos y la validación de resultados. Priorizar fue fundamental porque enfocó los esfuerzos iniciales en las actividades de mayor valor e incertidumbre, como la delimitación del marco teórico y la sistematización de la teoría del pensamiento computacional. A partir de este backlog general, se obtuvo el Sprint Backlog para cada ciclo. Este artefacto, desglosó cada ítem del Product Backlog seleccionado para un sprint en tareas concretas y ejecutables, transformando objetivos amplios en acciones específicas.

Finalmente, el avance se materializó en incrementos de investigación, que eran entregables revisados al final de cada sprint. Estos incrementos, que iban desde la redacción de capítulos hasta la recolección de datos por medio de instrumentos, permitieron visualizar frecuentemente el progreso y la validación por parte de la asesora, asegurando que el trabajo avanzara en la dirección correcta con la calidad requerida

Eventos Scrum aplicados a la investigación

El proceso se estructuró en torno a eventos Scrum, los cuales proporcionaron el ritmo junto con los espacios de revisión y adaptación necesarios para un proyecto de esta naturaleza. El ciclo comenzó con el sprint de planificación, donde se definieron los objetivos específicos y el conjunto de actividades del product backlog a desarrollar en la siguiente iteración. Luego, se ejecutó cada sprint de investigación, que cumplieron su función como iteraciones de tiempo para producir los incrementos de valor. La secuencia de sprints se diseñó estratégicamente para continuar el flujo lógico de una investigación (sprint 1: marco teórico; sprint 2: diseño metodológico; etc). Esta aproximación iterativa

permitió, por ejemplo, ajustar los instrumentos de recolección de datos tras una profundización en la literatura.

El progreso de cada sprint se monitoreó mediante reuniones periódicas (Stand-ups) autogestionadas, donde se identificaron obstáculos de manera oportuna, una práctica que resultó ser particularmente útil para mantener tanto la disciplina como el enfoque durante las fases intensivas de análisis y escritura.

Al finalizar cada ciclo, la revisión de sprints con la directora de tesis fue crucial para presentar el incremento terminado, discutirlo a profundidad y recabar feedback que podía ser incorporado o no inmediatamente en el siguiente ciclo, haciendo del proceso más interactivo y menos lineal.

Finalmente, la retrospectiva de cada sprint propició una reflexión metacognitiva sobre el proceso de trabajo, que permitió identificar ineficiencias y así, proponer mejoras para los sprints siguientes, de esta manera se optimizó constantemente la productividad y la calidad del trabajo.

Articulación ADDIE-ABP

La articulación entre el ABP y el modelo ADDIE se justifica en la medida en que ambos enfoques se complementan para favorecer entornos de aprendizaje significativos. ABP sitúa al estudiante como eje central de su proceso formativo, promoviendo la construcción activa del conocimiento al solucionar problemas y estimulando competencias de indagación, análisis y trabajo colaborativo (Barrows, 1986; Schmidt, 1983). A su vez, el modelo ADDIE aporta una estructura metodológica clara y rigurosa

que asegura la coherencia y organización en la elaboración de recursos educativos (Branch, 2009; Dick & Carey, 1978).

Así pues, mientras el ABP transforma los problemas en el motor que impulsa la experiencia de aprendizaje, ADDIE garantiza que dicha experiencia sea planificada de manera sistemática y contextualizada dentro de un AVA. Esta articulación no solo fortalece la pertinencia pedagógica del curso, sino que también favorece la implementación de procesos de evaluación sumativa y formativa.

Fases metodológicas

El AVA se estructuró en tres módulos secuenciales que conformaron el andamiaje pedagógico del curso, complementados con dos secciones adicionales de carácter informativo y de consulta.

El módulo 1 tuvo como propósito sentar las bases conceptuales del curso, abordando tres ejes temáticos: el primero abordó los pilares y componentes del PC, el segundo, los fundamentos del paradigma POO y, el tercero, consistió en una revisión de metodologías para la enseñanza del pensamiento computacional.

El módulo 2 constituyó el núcleo de aplicación de los conocimientos, donde los estudiantes ejercitaron lo aprendido mediante la resolución de talleres asignados de manera aleatoria a los cuatro grupos de trabajo. Estos talleres se diseñaron para aplicar de manera integrada los conceptos de la POO en la solución de problemas que demandan el uso de los pilares del pensamiento computacional junto con un componente esencial en la formación de licenciados como lo es la enseñanza.

El módulo 3 concentró los instrumentos para la valoración tanto del aprendizaje de los estudiantes como de la efectividad del curso. Este módulo permitió medir el logro de los objetivos de aprendizaje y recabar evidencia para la mejora continua de la propuesta.

Fase 1: Análisis

Propósito: Analizar las posibles necesidades formativas y establecer los fundamentos del curso sobre el PC desde la POO en un AVA.

Actividades desarrolladas:

- Se identificó la necesidad de fortalecer las habilidades de PC en los maestros en formación de la Licenciatura en Electrónica, particularmente en su fase de fundamentación, dada su importancia para la resolución de problemas complejos y su aplicación en contextos educativos.
- Se establecieron objetivos de aprendizaje enfocados en los pilares del PC que consistieron en: aplicar la descomposición para analizar problemas complejos del ámbito tecnológico y educativo, dividiéndolos en subproblemas que se modelan como clases y métodos independientes, facilitando tanto el diseño como la implementación de sistemas modulares en lenguajes como Python, Java y Scratch; Usar la abstracción con el fin de identificar y representar las características de entidades y procesos, creando modelos computacionales por medio de la definición de clases, atributos y métodos, mientras se descartan deliberadamente detalles que son irrelevantes para dar solución al problema; emplear el reconocimiento de patrones, para determinar similitudes estructurales

y comportamentales en distintos problemas, permitiendo la reutilización de código por medio de mecanismos de la POO como la herencia; lograr la generalización de soluciones, diseñando clases base y algoritmos flexibles que puedan extrapolarse y adaptarse mediante herencia y polimorfismo para resolver diferentes problemas en contextos educativos diversos; desarrollar la depuración con el propósito de detectar, diagnosticar y corregir errores de manera sistemática en el código, aplicando pruebas unitarias, validaciones y reflexión metacognitiva sobre la solución implementada.

- Se definieron los recursos tecnológicos necesarios para la implementación del AVA, incluyendo plataformas como genially y Google forms, herramientas de programación (Python, Java, Scratch) y recursos didácticos.

Articulación con la metodología ABP:

En esta fase se definieron los problemas para cada taller, los cuales fueron diseñados en contexto para ser relevantes y vinculados al ámbito escolar, cumpliendo con la primera etapa del ABP que es la presentación del problema.

Fase 2: Diseño

Cada uno de los módulos se puede consultar en el Anexo B

Propósito: Planificar la estructura del curso, diseñar las actividades de aprendizaje pertinentes, seleccionar estrategias pedagógicas adecuadas y establecer los instrumentos de evaluación coherentes con los objetivos formativos.

Actividades desarrolladas:

- Se estructuró el curso en tres módulos:

  - Fundamentos del PC, la POO y metodologías para la enseñanza del PC.
  - Implementación de la POO por medio de la solución de talleres.
  - Evaluación del PC y reflexiones finales.

- Se organizaron cuatro talleres que articulan los pilares del PC junto con los conceptos de POO.
- Se incorporaron estrategias activas como gamificación, aprendizaje colaborativo, micro proyectos interdisciplinarios y evaluación formativa.
- Se diseñaron rúbricas para evaluar desde habilidades computacionales, hasta la comprensión de conceptos de POO, calidad técnica del código, elaboración de material educativo y trabajo colaborativo.
- Se elaboraron recursos de apoyo como diagramas de flujo, modelos UML, ejemplos de código en diferentes lenguajes y material complementario que propicia la apropiación de los contenidos.

Articulación con la metodología ABP:

En esta fase se planificaron las etapas del ABP para cada taller:

- **Presentación del problema:** Situaciones problemáticas del contexto escolar.
- **Definición y análisis:** Actividades para clarificar términos y delimitar el problema.
- **Búsqueda de información:** Estrategias para investigar conceptos y soluciones.
- **Generación de hipótesis:** Espacios para proponer posibles soluciones.
- **Evaluación de soluciones:** Rúbricas para validar las propuestas desarrolladas.
- **Presentación de resultados:** Exposiciones para socializar las soluciones a los problemas planteados.
- **Reflexión:** Procesos de metacognición sobre el aprendizaje.

Fase 3: Desarrollo

Los materiales didácticos junto con la implementación en la plataforma Moodle se encuentra en el Anexo B. Además, cada presentación en Genially junto con el documento base se pueden consultar en el Anexo C.

Propósito: Elaborar material didáctico, recursos educativos e instrumentos de evaluación que respalden la implementación del curso en la plataforma Moodle.

Actividades desarrolladas:

- Se creó material formativo sobre PC, algoritmos, diagramas de flujo, UML, estructuras de control y fundamentos de POO (clases, objetos, herencia, encapsulamiento, polimorfismo).
- Se diseñaron cuatro talleres con sus respectivos problemas, actividades y productos esperados:
    1. Taller 1: Implementación de POO para simular sensores de laboratorio.
    2. Taller 2: Pensamiento algorítmico para sistema de gestión bibliotecaria.
    3. Taller 3: Desarrollo de aplicaciones con patrones de diseño y metodologías ágiles.
    4. Taller 4: Proyecto integral para gestión escolar.
- Se realizaron videos explicativos, infografías, tutoriales de programación y ejemplos interactivos.
- Se programó la plataforma Moodle para albergar los contenidos, actividades y herramientas de comunicación síncrona y asíncrona.
- Se implementaron las rúbricas diseñadas, cuestionarios de autoevaluación y coevaluación.

Articulación con la metodología ABP: Los materiales desarrollados apoyan cada una de las etapas del ABP, al ofrecer andamiajes que facilitan la comprensión de los problemas, la indagación de posibles soluciones, la implementación de código y reflexión crítica sobre los aprendizajes.

Fase 4: Implementación

Propósito: Implementar el curso diseñado con los estudiantes de la asignatura fundamentos de tecnología 2 de la licenciatura en electrónica, para evaluar su pertinencia pedagógica y su impacto en el desarrollo del PC.

Actividades desarrolladas: Los contenidos, actividades y recursos se publicaron en la plataforma Moodle acorde con la estructura previamente definida para el curso. Asimismo, se emplearon 10 horas distribuidas entre el trabajo en plataforma y trabajo autónomo, también, se organizaron 17 estudiantes en cuatro grupos de trabajo para que desarrollaran los talleres.

Articulación con la metodología ABP: Durante la implementación, cada estudiante transitó por las siete etapas del ABP al interactuar con el curso y resolver el taller:

1. Presentación del problema: Cada taller incluía un problema en particular.
2. Definición y análisis: Se Identificó los términos clave y se delimitó del alcance del problema.
3. Búsqueda de información: Consulta sobre conceptos de POO y estrategias de solución a la problemática.

4. Generación de hipótesis: Se formularon propuestas de solución basadas en los pilares del PC.
5. Evaluación de soluciones: Mediante rúbrica se analizaron las soluciones planteadas.
6. Presentación de resultados: Cada grupo expuso la solución implementada mediante código funcional y material educativo.
7. Reflexión y retroalimentación: Por medio de formularios los estudiantes se evaluaron, coevaluaron y, además, midieron sus conocimientos e incluso reflexionaron sobre lo aprendido.

Fase 5: Evaluación

Cada instrumento de evaluación que se aplicó se encuentra disponible en el (Anexo A).

Propósito: Valorar la efectividad del curso, el grado de logro de los objetivos de aprendizaje y la calidad de los productos desarrollados, para identificar oportunidades de ajuste y mejora.

Actividades desarrolladas:

- Se realizó un seguimiento que valoraba los siguientes aspectos:
    - Habilidades del PC (abstracción, descomposición, reconocimiento de patrones, generalización y depuración).
    - Comprensión y aplicación de conceptos de POO.
    - Calidad técnica del código implementado.
    - Elaboración de material educativo.
    - Trabajo colaborativo y participación.

- Se evaluaron los productos finales de cada taller (código funcional, diagramas UML, material educativo) mediante rúbricas.
- Los estudiantes reflexionaron sobre su propio aprendizaje y evaluaron el trabajo de sus pares.
- Se recogieron opiniones sobre la pertinencia, utilidad y calidad del curso a través de cuestionarios.

Integración con ABP: La evaluación se alineó con las etapas finales del ABP, enfocándose en la valoración de las soluciones desarrolladas, la reflexión sobre el proceso de aprendizaje y la identificación de mejoras para futuras iteraciones.

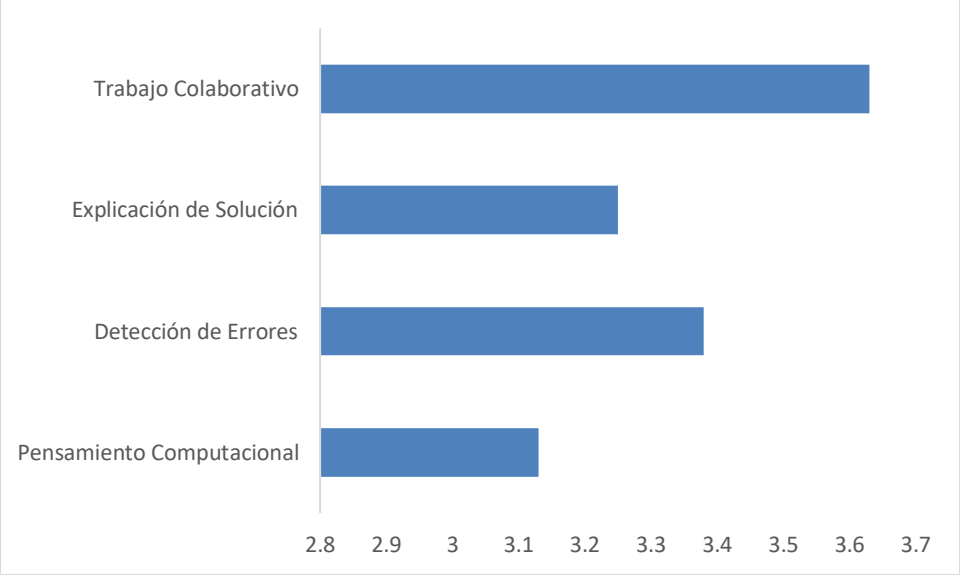
Resultados y análisis

El presente capítulo expone los resultados obtenidos luego de la aplicación de cinco instrumentos de evaluativos, diseñados con el propósito de evaluar el grado de comprensión y los alcances del curso en el desarrollo del pensamiento computacional entre los participantes. El análisis integra datos cuantitativos y cualitativos recabados mediante un instrumento de evaluación, un formulario de autoevaluación, otro de coevaluación, un test de conocimientos y un formulario de reflexión. El objetivo es ofrecer una visión multidimensional que permita valorar no solo la adquisición de conocimientos teóricos, sino también la aplicación práctica de los pilares del pensamiento computacional, las habilidades colaborativas y la percepción del curso.

Autoevaluación

Este primer formulario (Anexo B) permitió que cada estudiante valorara su autopercepción respecto a la comprensión general del curso y la aplicación de un conjunto de habilidades transversales que, si bien no constituyen los pilares del pensamiento computacional, representan destrezas fundamentales para el desarrollo y aplicación del PC en el contexto de la POO. El análisis de las 17 respuestas válidas, categorizadas en niveles de desempeño (Básico, Satisfactorio, Avanzado), proporciona un indicio del grado de apropiación de estas habilidades transversales. En la figura 2 se muestra el promedio para cada criterio de la autoevaluación.

Figura 2.
Distribución de niveles de desempeño en autoevaluación.



Nota. Fuente: elaboración propia, a partir de los datos recabados por el formulario de autoevaluación.

Las estadísticas generales se evidencian en la tabla 5 donde se relaciona cada criterio con el promedio de las respuestas de cada estudiante y consisten en:

- Promedio de puntaje total: 13.63 puntos.
- Mediana: 14 puntos.
- Desviación estándar: 2.16 puntos.

Tabla 1.
Promedios por criterio de autoevaluación.

Criterio	Promedio	Mediana	Desviación Estándar
Pensamiento Computacional	3,13	3	0,72
Detección de Errores	3,38	3	0,81
Explicación de Solución	3,25	3	0,86
Trabajo Colaborativo	3,63	4	0,96

Nota. Fuente: elaboración propia, a partir de los datos recabados por el formulario de autoevaluación.

Estos resultados indican que el curso tuvo un alcance satisfactorio en el desarrollo de habilidades transversales asociadas al pensamiento computacional, como el trabajo colaborativo. Sin embargo, se evidencia un área de mejora en la destreza para la explicación de soluciones, que es esencial en la formación de licenciados porque estos requieren no solo resolver problemas técnicos sino también comunicar efectivamente sus procesos de pensamiento.

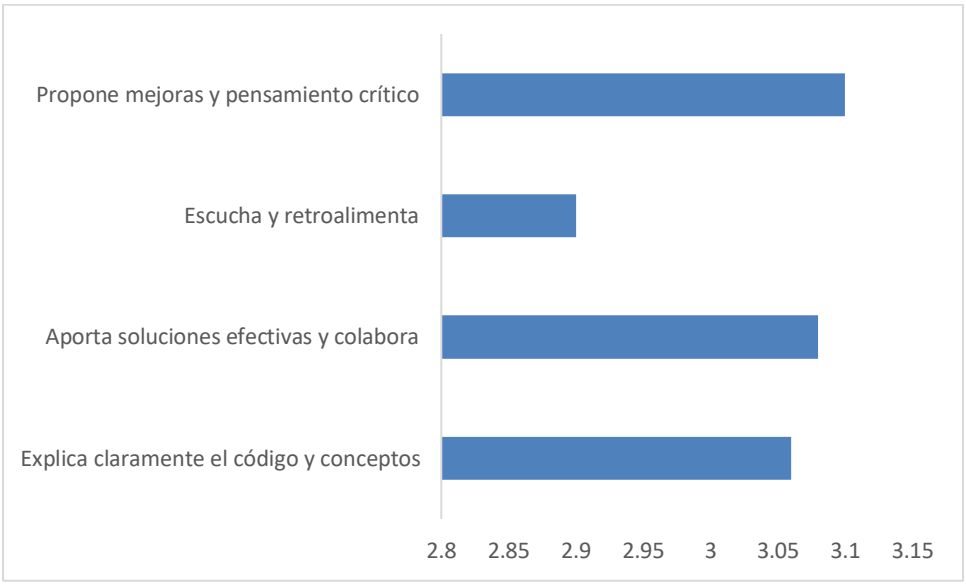
Coevaluación

El segundo instrumento (Anexo A) se diseñó para valorar habilidades inherentes al pensamiento computacional y la programación orientada a objetos en entornos de trabajo grupal. Los cuatro aspectos por evaluar consisten en: Claridad explicativa, que evalúa la capacidad de explicar el código junto con los diseños y las soluciones implementadas; aporte colaborativo, valora la contribución a la solución del taller; capacidad de escucha y retroalimentación, mide la disposición para recibir y procesar

aportes de los integrantes del grupo de trabajo; pensamiento crítico, evalúa la habilidad de analizar, cuestionar y proponer mejoras a las soluciones tanto propias como ajenas.

Estos criterios se relacionan con los pilares del PC porque se requiere de la abstracción, la descomposición, la depuración, el reconocimiento de patrones y la generalización para no solo trabajar en equipo sino también dar solución al taller propuesto. En la figura 3 se ilustra el promedio para cada criterio de la coevaluación.

Figura 3.
Distribución en niveles de coevaluación.



Nota. Fuente: elaboración propia, a partir de los datos recabados por el formulario de autoevaluación.

Las estadísticas generales se ilustran en la tabla 6 y consisten en:

- Promedio de puntaje total: 12.44 puntos.
- Mediana: 15 puntos.
- Desviación estándar: 6.08 puntos.

Tabla 2.
Promedios por criterio de coevaluación.

Criterio	Promedio	Mediana	Desviación Estándar
Explica claramente el código y conceptos	3,06	3	1,42
Aporta soluciones efectivas y colabora	3,08	3	1,38
Escucha y retroalimenta	2,9	3	1,45
Propone mejoras y pensamiento crítico	3,1	4	1,36

Nota. Fuente: elaboración propia, a partir de los datos recabados por el formulario de coevaluación.

La brecha entre los niveles Insuficiente (25%) y Avanzado (39.6%) refleja una comprensión y aplicación desigual de las competencias colaborativas entre cada estudiante. El hecho de que el pensamiento crítico sea el criterio mejor evaluado sugiere un alcance positivo del curso en fomentar esta habilidad de alto nivel, lo que indica que los estudiantes fueron capaces de analizar y proponer mejoras en los diseños de POO. Sin embargo, la baja puntuación en escucha y retroalimentación señala un límite en el alcance del curso para desarrollar integralmente las habilidades interpersonales necesarias para un trabajo colaborativo efectivo.

Test del PC

El test (Anexo A) evaluó la comprensión teórica de los pilares del pensamiento computacional y conceptos de POO, por medio de 13 preguntas cerradas y 2 abiertas. En el diseño del instrumento se categorizaron las preguntas con el fin de analizar por separado el dominio en: los pilares del PC, los conceptos de la POO y la capacidad de aplicación integrada en problemas abiertos. Los resultados, evidenciados en la figura 4 miden de forma directa el grado de asimilación conceptual.

Análisis por categoría

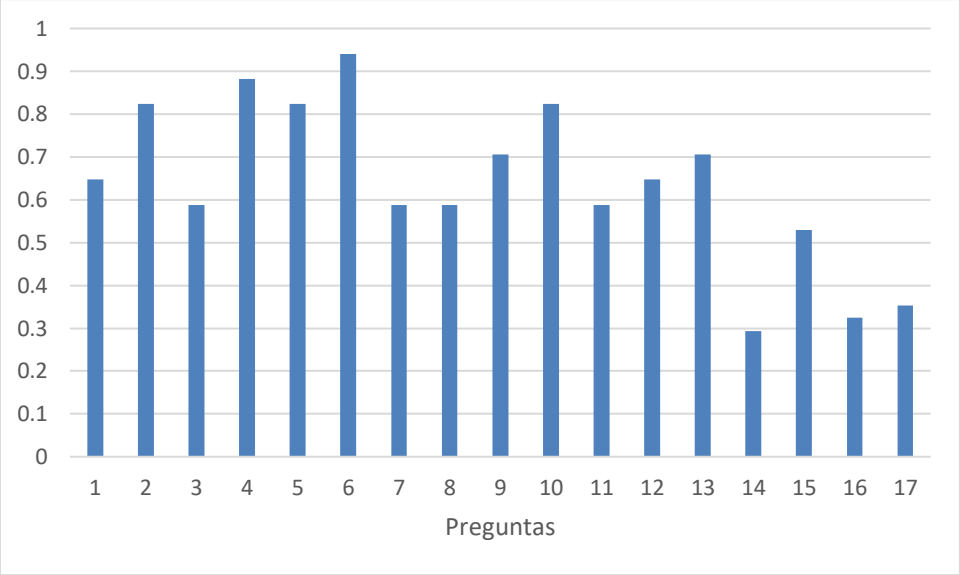
Pilares del pensamiento computacional: El bloque de preguntas sobre los pilares del PC evaluó la comprensión de cada uno. Las preguntas 4 y 6 relacionadas con la descomposición presentaron un alto porcentaje de acierto, lo que sugiere que los estudiantes internalizaron la importancia de descomponer un problema en subproblemas. Las preguntas 1 y 5 relacionadas con la abstracción y el reconocimiento de patrones mostraron un desempeño medio, indicando una comprensión aceptable en estos pilares. La depuración junto con la generalización abordadas en las preguntas 2 y 3 respectivamente, fueron los pilares con menor comprensión, lo que apunta a una dificultad para distinguirlos de otros procesos mentales en un contexto teórico.

Conceptos de programación orientada a objetos: Los conceptos de herencia y polimorfismo abordados en las preguntas 9 y 10 fueron los conceptos mejor comprendidos. En relación con los conceptos de objeto y atributos abordados en las preguntas 13 y 12, evidencian un dominio moderado. Los estudiantes mostraron un bajo

dominio en las definiciones de clase, encapsulamiento y método, abordadas en las preguntas 7, 8 y 11.

Aplicación integrada en problemas abiertos: Las preguntas abiertas (14 – 17) mostraron dificultades significativas, en comparación con las preguntas cerradas se evidencia una brecha que revela dificultades cuando se trata de aplicar los conceptos.

Figura 4.
Distribución de niveles en test sobre el PC.



Nota. Fuente: elaboración propia.

La distribución general de los puntajes muestra que 70% de los estudiantes se sitúan en los niveles básico y alto. Con respecto a las estadísticas generales se tiene:

- Promedio: 11.15 puntos.
- Mediana: 11.0 puntos.
- Desviación estándar: 3.83 puntos.

Tabla 3.
Porcentaje de acierto por pregunta del test

Estudiante	Puntaje Total
1	6
2	16
3	13
4	9
5	13,5
6	8
7	14
8	10,5
9	11
10	9,5
11	16
12	15,5
13	10
14	15,5
15	11
16	9

Nota. Fuente: elaboración propia, a partir de los datos recabados por el formulario test del PC.

La concentración de estudiantes en los niveles básico y alto evidencia un grado de comprensión teórica medio. La dificultad en las preguntas abiertas, que requieren articular una respuesta no predefinida, revela un alcance limitado en la capacidad de transferir y aplicar los conceptos a problemas nuevos de forma explícita. Esto indica que la comprensión operativa es inferior a la comprensión conceptual.

Evaluación de talleres

Como complemento al test de conocimientos se elaboró un taller grupal cuyo resultado fue socializado entre los integrantes que participaron en el ejercicio, su evaluación se efectuó bajo los lineamientos del instrumento de evaluación (Anexo A)

sobre los pilares del pensamiento computacional en cada uno de los talleres entregados por los estudiantes, cuyos resultados se encuentran en las tablas 1, 2, 3 y 4.

Tabla 4.
Taller 1: Sistema escolar híbrido.

Criterio	Nivel	Justificación
Abstracción	Avanzado (4.5)	Modela eficientemente entidades (usuarios, cursos, asistencia) con atributos esenciales, anticipando relaciones complejas mediante UML.
Descomposición	Competente (4.0)	Divide en módulos lógicos (backend, frontend, BD) pero con cierta redundancia en esquemas de datos.
Reconocimiento de patrones	Avanzado (4.8)	Identifica y aplica patrones REST, MVC, JWT, y genera adaptaciones para reportes CSV/PDF.
Generalización	Avanzado (4.7)	Propone arquitectura escalable (Python/Java) transferible a otras instituciones educativas.
Depuración	Competente (3.8)	Incluye pruebas manuales y automáticas, pero no evidencia validación de casos edge.

Nota. Fuente: elaboración propia.

Tabla 5.
Taller 2: Sistema de gestión de biblioteca.

Criterio	Nivel	Justificación
Abstracción	Competente (3.5)	Representa elementos clave (libros, préstamos, multas) pero con detalles innecesarios en flujos.
Descomposición	En desarrollo (2.8)	Separa en funciones con pasos redundantes (ej: registro repetitivo en Java/Python).

Reconocimiento de patrones	Competente (3.2)	Aplica POO de forma básica pero no optimiza estructuras de datos para búsquedas.
Generalización	En desarrollo (2.5)	Solución limitada a casos específicos sin mecanismos de extensión claros.
Depuración	Inicial (1.8)	No incluye pruebas validadas; código con errores (ej: variable "paso = true" en Python).

Nota. Fuente: Elaboración propia.

Tabla 6.

Taller 3: Modelado de sistemas electrónicos con POO.

Criterio	Nivel	Justificación
Abstracción	Avanzado (4.9)	Generaliza magistralmente elementos electrónicos mediante herencia y polimorfismo.
Descomposición	Avanzado (4.8)	Jerarquiza componentes (sensores/actuadores) reutilizando lógica base eficientemente.
Reconocimiento de patrones	Avanzado (5.0)	Conecta explícitamente patrones de POO con pilares del pensamiento computacional.
Generalización	Avanzado (4.9)	Soluciones transferibles a cualquier sistema de IoT con material educativo incluido.
Depuración	Competente (4.0)	Incluye ejemplos de testing pero no cubre todos los escenarios de error.

Nota. Fuente: Elaboración propia.

Tabla 7.
Taller 4: Sistema de gestión académica con Scrum.

Criterio	Nivel	Justificación
Abstracción	Competente (3.8)	Modela entidades académicas, pero con limitada representación de relaciones.
Descomposición	Avanzado (4.5)	Organiza módulos funcionales y roles Scrum de manera óptima.
Reconocimiento de patrones	Competente (4.0)	Aplica Singleton y metodologías ágiles, aunque sin profundizar en otros patrones.
Generalización	En desarrollo (3.0)	Propone transferencia educativa, pero con implementación técnica limitada.
Depuración	Inicial (2.0)	Carece de mecanismos de validación; código con posibles errores de consistencia.

Nota. Fuente: Elaboración propia.

La evidencia recopilada demuestra que el AVA con POO se configuró como un ambiente efectivo para desarrollar el pensamiento computacional. Los resultados globales por cada pilar del PC revelan un desempeño destacado en abstracción y reconocimiento de patrones ubicándose ambas en un nivel avanzado. La descomposición muestra un dominio solido por parte de los estudiantes ubicándose en un nivel entre competente y avanzado. En relación con la generalización se categoriza como aceptable porque se sitúa en un nivel competente. Sin embargo, la depuración se ubicó en un nivel de desarrollo.

Reflexión

El cuarto formulario (Anexo A) recogió las percepciones cualitativas de los estudiantes a través de seis preguntas sobre la utilidad del PC, su pertinencia en la

formación profesional y las estrategias de enseñanza. Para sintetizar esas respuestas y tener una visión general sobre la precepción del curso, se realizó un análisis de contenido que categorizó cada pregunta en una de las siguientes tres dimensiones:

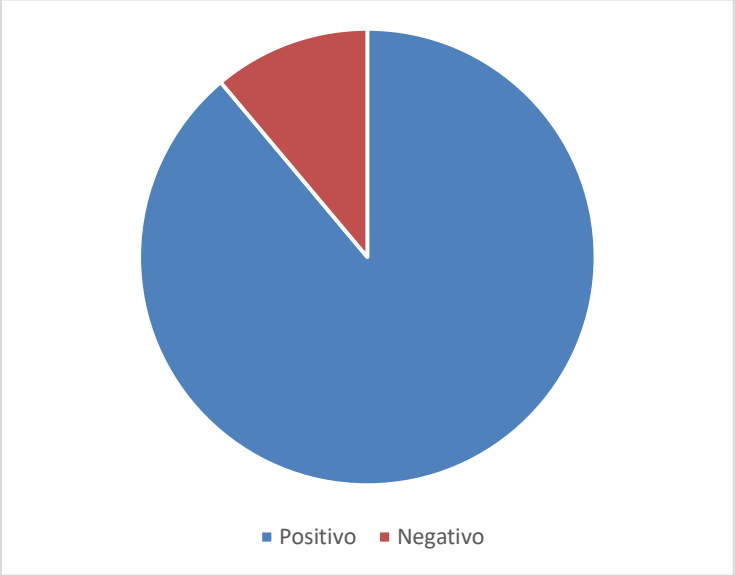
Percepción positiva: Respuestas que destacan la utilidad, relevancia o aspectos favorables del curso.

Percepción negativa: Respuestas que hacían hincapié en las dificultades, críticas a la metodología o una valoración general desfavorable

Percepción mixta: Respuestas que combinan comentarios positivos y negativos de manera equivalente.

En la figura 5 se encuentra el resultado de la percepción de los estudiantes categorizada a partir del análisis de respuestas cualitativas.

Figura 5.
Distribución de percepciones estudiantiles.



Nota. Fuente: Elaboración propia, a partir de los datos recabados por el cuarto formulario.

Hallazgos cualitativos: Los estudiantes reconocen la utilidad del curso para el desarrollo del razonamiento lógico, la resolución de problemas y su aplicabilidad en el programa de Licenciatura en Electrónica. Un estudiante respondió: “Me ayuda a estructurar mejor mis ideas para resolver problemas no solo en programación”. La crítica más frecuente se relaciona con la metodología, percibida por algunos como “poco interactiva” y con “exceso de contenido textual”, lo que sugiere una demanda de una experiencia más dinámica. También, se destacó la necesidad de un enfoque más práctico, un mayor uso de entornos virtuales como Scratch y la implementación de proyectos alternativos que diversifiquen las actividades. Se resalta que el tiempo de interacción de los estudiantes con el curso fue limitado y ajustado a los tiempos permitidos por el docente a cargo del espacio académico quien permitió la aplicación de este ejercicio.

La percepción positiva (88.9%) demuestra que los estudiantes comprenden y valoran los objetivos formativos del curso y sus alcances transversales en su formación. Sin embargo, las críticas constructivas señalan una desconexión entre el potencial del pensamiento computacional y la metodología de enseñanza, sugiriendo que el alcance lúdico podría ser mayor.

Discusión

Los resultados obtenidos en este estudio permiten una valoración multidimensional del alcance del curso sobre PC mediado por POO en un AVA. El análisis conjunto de los datos recabados por medio de los diferentes instrumentos refleja logros significativos en la comprensión tanto teórica como aplicada del pensamiento computacional, también, identifica áreas que requieren ajustes en el diseño pedagógico.

Los resultados globales de la evaluación aplicada en cada taller por medio de un instrumento que evaluaba los pilares del PC muestran un desempeño destacado en abstracción y reconocimiento de patrones. Esto indica que los estudiantes pudieron modelar entidades y aplicar soluciones preexistentes a nuevos contextos. Ahora bien, la descomposición mostró un dominio sólido, mientras que la generalización se ubicó en un nivel competente. Sin embargo, la depuración fue el pilar con menor desarrollo, evidenciando dificultades para identificar el tipo de error y su corrección en los códigos presentados. Este hallazgo sugiere que, a pesar de los buenos resultados en cuanto al desarrollo de habilidades en diseño y modelado, la fase de implementación y validación requiere un mayor énfasis.

Con relación al grado de comprensión teórica, los datos del test sobre el pensamiento computacional revelan una asimilación conceptual media. Además, el análisis por categoría mostró un porcentaje alto de acierto en preguntas cerradas relacionadas con la descomposición y conceptos de POO como herencia y polimorfismo. No obstante, es evidente la brecha entre el dominio conceptual y la capacidad de

comunicar aquello que se sabe. Esta brecha se identifica luego de comparar las preguntas cerradas con las abiertas donde los alumnos mostraron dificultades en transferir los conceptos a problemas nuevos. Esto indica que la capacidad de explicar el proceso de pensamiento y la comprensión operativa son inferiores a la comprensión conceptual.

En cuanto a las habilidades transversales e inherentes al PC, los resultados de la coevaluación mostraron un desarrollo desigual porque, por ejemplo, el pensamiento crítico fue la habilidad mejor valorada y, aspectos como la capacidad de escucha y retroalimentación obtuvieron las calificaciones más bajas. Estas habilidades son esenciales para el pensamiento computacional en entornos colaborativos debido a que pilares como la depuración y la generalización dependen de la retroalimentación entre pares. La baja puntuación en aquellos ítems sugiere una limitación en las dinámicas de interacción social dentro del AVA, que a pesar de estar planteadas en la metodología ABP no lograron fomentar de manera integral las competencias interpersonales.

Finalmente, la percepción estudiantil fue mayormente positiva, lo que demuestra que los alumnos valoraron la utilidad del curso y, además, comprendieron sus objetivos formativos. Comentarios como “Me ayuda a estructurar mejor mis ideas para resolver problemas” respaldan la idea sobre articular el ABP en un AVA porque proporcionó un espacio que, entre otras cosas, incentivó la organización del razonamiento frente a problemas de diferentes contextos. Esta valoración se alinea con el modelo SAMR (Puentedura, 2010), pues el AVA permitió crear una experiencia de aprendizaje que trasciende la mera sustitución de lo presencial. No obstante, las críticas sobre el exceso de contenido textual, la demanda de una experiencia más práctica y lúdica junto con el

poco tiempo de implementación, indican que el potencial del AVA para redefinir la experiencia de aprendizaje no fue explotado en su totalidad en términos de interactividad.

Conclusiones y recomendaciones

Conclusiones

Se demuestra la sistematización de la teoría del pensamiento computacional integrando desde sus fundamentos, hasta la POO e incluso sus metodologías de enseñanza en tres módulos, en la plataforma Moodle, dando cumplimiento al objetivo general propuesto, lo que indica que es viable no solo crear sino también implementar una propuesta formativa de esta naturaleza. La articulación de los referentes pedagógicos con la plataforma Moodle permitió estructurar un entorno de aprendizaje coherente que abordó los contenidos propuestos.

Adicionalmente, se diseñó contenido educativo desde la metodología ABP y, se evaluó el grado de comprensión entre los participantes, el análisis de los datos permite concluir que el curso favoreció una comprensión teórica y aplicada del PC y la POO. Los resultados de los talleres mostraron un dominio satisfactorio y, en ciertos casos, avanzado. Sin embargo, también se identificó una brecha en la capacidad de los estudiantes para extrapolar el conocimiento en nuevos escenarios y comunicar sus soluciones. Asimismo, la depuración fue el pilar del PC de menor desarrollo en la aplicación práctica.

Recomendaciones

Se recomienda incorporar talleres específicos junto con ejercicios guiados centrados en la depuración. Además, incluir problemas de diversa índole que orienten a los estudiantes en adaptar y generalizar soluciones, con el propósito de cerrar la brecha entre la comprensión conceptual y aplicación práctica. También, se sugiere enriquecer el AVA con recursos interactivos y multimedia que complementen los textos, como simuladores de código o microjuegos.

Finalmente, se recomienda desarrollar estudios que exploren la efectividad de diferentes estrategias instruccionales dentro de un AVA para fortalecer los pilares del pensamiento computacional. Asimismo, sería valioso realizar investigaciones que evalúen el impacto a largo plazo de estos cursos en las prácticas de enseñanza de los Licenciados en Electrónica.

Referencia Bibliográficas

Álava, W. L. S. (2022). Impacto del uso de e-learning en la educación superior. *Revista Unesum Ciencias*, *6*(1), 1-12. https://revistas.unesum.edu.ec/index.php/unesumciencias/article/download/690/626/2236

Álvarez, M. C., Ocampo, L. M., & Torres, S. A. C. (2024). Definiciones del pensamiento computacional. Una revisión de la literatura. *Revista EIA*, *21*(42), 4207-4221. https://doi.org/10.24050/reia.v21i42.1812

Andrade, A. (2023, 15 de marzo). *Metodología Scrum: Roles, procesos y artefactos*. Blog Innevo. https://innevo.com/blog/metodologia-scrum

Asana. (2025). *Scrum: conceptos clave y cómo se aplica en la gestión de proyectos*. https://asana.com/es/resources/what-is-scrum

Ávila, J., & Bailón, J. (2022). Clases y objetos. En *Análisis y diseño en POO*. Portal Académico del CCH, UNAM. https://portalacademico.cch.unam.mx/cibernetica1/analisis-y-diseno-en-poo/clases-y-objetos

Banco Interamericano de Desarrollo & ProFuturo. (2025). *Aproximación a las competencias digitales de docentes en América Latina* (Nota Técnica No IDB-TN-03160). https://www.iadb.org

Barr, D., & Stephenson, C. (2011). Bringing computational thinking to K-12: What is involved and what is the role of the computer science education community? *ACM Inroads*, *2*(1), 48–54. https://doi.org/10.1145/1929887.1929905

Barrows, H. S. (1986). A taxonomy of problem-based learning methods. *Medical Education*, *20*(6), 481-486. https://doi.org/10.1111/j.1365-2923.1986.tb01386.x

Berges, M. P. (2015). *Object-oriented programming through the lens of computer science education* [Tesis doctoral, Technische Universität München]. https://doi.org/10.14459/2015md1220866

Branch, R. M. (2009). *Instructional design: The ADDIE approach*. Springer. https://doi.org/10.1007/978-0-387-09506-6

Bustos Sánchez, A., & Coll Salvador, C. (2010). Los entornos virtuales como espacios de enseñanza y aprendizaje: Una perspectiva psicoeducativa para su caracterización y análisis. *Revista Mexicana de Investigación Educativa*, *15*(44), 163–184. https://www.redalyc.org/pdf/140/14012501008.pdf

Carmona-Mesa, J. A., Villa-Ochoa, J. A., & Restrepo, E. (2021). Pensamiento computacional en la formación de profesores de matemáticas: una revisión sistemática. *Educación Matemática*, *33*(2), 7-36. https://doi.org/10.24844/EM3302.01

Castañeda, S. B. (2017). Los ambientes virtuales. Aprendizaje y conocimiento tecnológico y didáctico del contenido. En A. Ramírez y E. Hernández (Eds.), *Memorias del Congreso Internacional de Innovación Educativa* (pp. 45-60). Editorial Universitaria.

Cerón Garnica, C. (s. f.). *POO en Java*. Ecosistema BUAP. Benemérita Universidad Autónoma de Puebla. Recuperado el 18 de octubre de 2024, de https://ecosistema.buap.mx/forms/files/dspace-23/index.html

Coll, C. (1987). *Psicología y currículum: Una aproximación psicopedagógica a la elaboración del currículum escolar*. Laia.

Cruz, C. E. Z., Colado, A. Z., Ocegueda, A. T. S., & Escobedo, R. M. V. (2020). Análisis crítico de ambientes virtuales de aprendizaje. *Utopía y Praxis Latinoamericana: Revista Internacional de Filosofía Iberoamericana y Teoría Social*, *25*(11), 33–47. https://doi.org/10.5281/zenodo.4153122

Dick, W., & Carey, L. (1978). *The systematic design of instruction*. Scott, Foresman and Company.

Donovan, J. M., & Boud, D. (2012). Sustainable assessment and problem-based learning. *Assessment & Evaluation in Higher Education*, *37*(2), 203-214. https://doi.org/10.1080/02602938.2010.515016

Escueladirecta. (2021, 23 de diciembre). *Encapsulamiento – Pilares de la programación orientada a objetos* [Entrada de blog]. https://escueladirecta-blog.blogspot.com/2021/10/encapsulamiento-pilares-de-la.html

Fedesoft. (s. f.). *Fedesoft: Federación Colombiana de la Industria de Software y TI*. Recuperado el 18 de octubre de 2024, de https://fedesoft.org/

García-Peñalvo, F. J. (2015). Una revisión actualizada del concepto de e-learning. *Education in the Knowledge Society (EKS)*, *16*(4), 119-144. https://doi.org/10.14201/eks2015164119144

Gómez, B. R. (2005). Aprendizaje basado en problemas (ABP): Una innovación didáctica para la enseñanza universitaria. *Educación y Educadores*, *8*, 9–20. https://www.redalyc.org/pdf/834/83400803.pdf

Grover, S., & Pea, R. (2013). Computational Thinking in K–12: A Review of the State of the Field. *Educational Researcher*, *42*(1), 38–43. https://doi.org/10.3102/0013189X12463051

Herlambang, A. D., & Rachmadi, A. (2024). Computational thinking skills theorization in the vocational high school computer programming subject context. *Elinvo (Electronics, Informatics, and Vocational Education)*, *9*(1), 64–75. https://doi.org/10.21831/elinvo.v9i1.64501

Jara, N. (2023). E-learning en estudiantes de una universidad pública. *Revista Horizontes*, *7*(29), 1-15. https://revistahorizontes.org/index.php/revistahorizontes/article/view/1262/2355

Joyanes Aguilar, L. (1996). *Programación orientada a objetos*. McGraw-Hill.

Kannadass, P., Hidayat, R., Siregar, P. S., & Husain, A. P. (2023). Relationship between computational and critical thinking towards modelling competency among pre-service mathematics teachers. *TEM Journal*, *12*(3), 1370–1382. https://doi.org/10.18421/TEM123-17

Kay, A. (2003). *Clarification of 'Object-Oriented'. Dr. Alan Kay on the Meaning of Object-Oriented Programming*, 23. http://www.purl.org/stefan_ram/pub/doc_kay_oop_en

Korkmaz, Ö., & Xuemei, B. (2019). Adapting computational thinking scale (CTS) for Chinese high school students and their thinking scale skills level. *Participatory Educational Research*, *6*(2), 65–76.

Krügel, J., & Hubwieser, P. (2017). Computational thinking as springboard for learning object-oriented programming in an interactive MOOC. En *IEEE Global Engineering Education Conference (EDUCON)* (pp. 1634–1642). IEEE. https://doi.org/10.1109/EDUCON.2017.7993301

Kules, B. (2016). Computational thinking is critical thinking. *Proceedings of the 47th ACM Technical Symposium on Computing Science Education*, 97–102.

Llorens Largo, F., García-Peñalvo, F. J., Molero Prieto, X., & Vendrell Vidal, E. (2017). La enseñanza de la informática, la programación y el pensamiento computacional en los estudios preuniversitarios. *Education in the Knowledge Society (EKS)*, *18*(2), 7–17. https://doi.org/10.14201/eks2017182717

López, L. (2013). *Metodología de la programación orientada a objetos*. Alpha Editorial.

López Pinzón, L. K., & Pineda Paredes, J. O. (2022). *Desarrollo de habilidades de pensamiento computacional por medio de actividades conectadas y desconectadas en estudiantes de grados sexto y séptimo* [Tesis de maestría, Universidad Pedagógica Nacional]. Repositorio Institucional

UPN. https://repositorio.upn.edu.co/server/api/core/bitstreams/0948bec5-979f-467f-9f5d-5e639854f9cf/content

Malik, S. I., Mathew, R., Tawafak, R. M., Al-Farsi, G., & Al-Sideiri, A. (2025). Investigating the impact of the OOP-SOLVE application on the user's behavior using the technology acceptance model in the programming course. *Frontiers in Computer Science*, *7*, 1510577. https://doi.org/10.3389/fcomp.2025.1510577

Martin, R. C. (2003). *Agile software development: Principles, patterns, and practices*. Prentice Hall.

Meyer, B. (1997). *Object-oriented software construction* (2.a ed.). Prentice Hall.

Microsoft. (2025, 18 de junio). *Programación orientada a objetos: herencia – C#*. Microsoft Learn. Recuperado el 18 de octubre de 2024, de https://learn.microsoft.com/es-es/dotnet/csharp/fundamentals/object-oriented/inheritance

Microsoft. (2025, 18 de julio). *Polymorphism (C# Fundamentals)*. Microsoft Learn. Recuperado el 18 de octubre de 2024, de https://learn.microsoft.com/es-es/dotnet/csharp/fundamentals/object-oriented/polymorphism

Molenda, M. (2003). The ADDIE model. En D. H. Jonassen (Ed.), *Handbook of research on educational communications and technology* (pp. 356–358). Lawrence Erlbaum Associates.

Moncho Mas, V. (2016). *Introducción a la programación orientada a objetos* (PID_00220485). Universitat Oberta de Catalunya. https://openaccess.uoc.edu/bitstream/10609/10541/6/inf_ed_cast.pdf

Mora Mora, D. P., & Bejarano Aguado, G. A. (2016). Prácticas educativas en ambientes virtuales de aprendizaje. *Aletheia. Revista de Desarrollo Humano, Educativo y Social Contemporáneo*, *8*(2), 48–63. https://doi.org/10.11600/21450366.8.2aletheia.48.63

Morales González, B. (2022). Diseño instruccional según el modelo ADDIE en la formación inicial docente. *Revista Apertura*, *14*(1), 80-95. https://doi.org/10.32870/Ap.v14n1.2129

Moreno Díaz, Ó. (s. f.). Clases y objetos. En *Programación orientada a objetos* (Módulo Aula en Abierto del INTEF). INTEF. Recuperado el 18 de octubre de 2024, de https://formacion.intef.es/aulaenabierto/mod/book/view.php?id=5149&chapterid=6791

Mujica-Sequera, R. (2020). E-learning como estrategia pedagógica en la educación superior. *Revista Tecnológica-Educativa Docentes 2.0*, *9*(1), 37–41. https://doi.org/10.37843/rted.v9i1.103

Ogegbo, A. A., & Ramnarain, U. (2022). A systematic review of computational thinking in science classrooms. *Studies in Science Education*, *58*(2), 203–230. https://doi.org/10.1080/03057267.2022.2044417

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

Patel, S. R., Sampat, D., & Trowbridge, J. J. (2018). Using instructional design, analyze, design, develop, implement, and evaluate (ADDIE) model in the

development of e-learning modules. *Perspectives on Medical Education*, *7*(Suppl. 1), S45-S46. https://doi.org/10.1007/s40037-018-0415-z

Piaget, J. (1937). *La construcción de lo real en el niño*. Morata.

Piaget, J. (1970). *Seis estudios de psicología*. Seix Barral.

Piaget, J. (1972). *Psicología de la inteligencia*. Crítica.

Portal Académico CCH-UNAM. (2022, 8 de mayo). *Características de la POO*. https://portalacademico.cch.unam.mx/cibernetica1/algoritmos-y-codificacion/caracteristicas-POO

Puentedura, R. (2010, abril). *SAMR and TPCK: Intro to advanced practice*. [Presentación]. Recuperado de https://hippasus.com/resources/sweden2010/SAMR_TPCK_IntroToAdvancedPractice.pdf

Pérez, A. J. C., García, L. M., & López, R. (2021). Habilidades del pensamiento computacional en docentes en formación en Colombia. *Revista Latinoamericana de Educación Digital*, *2*(1), 120–135. https://doi.org/10.1234/rled.v2i1.12345

Quevedo Delgado, M. A. (2025). *Ambientes virtuales de aprendizaje: una estrategia para el desarrollo del pensamiento algorítmico y la resolución de problemas matemáticos en sexto grado* [Tesis de maestría, Universidad Pedagógica Nacional]. Repositorio Institucional UPN. https://repositorio.upn.edu.co/server/api/core/bitstreams/5d026b5b-2288-4b38-b44c-8d07266ea068/content

Resnick, M. (2008). Sowing the seeds for a more creative society. *Learning & Leading with Technology*, *35*(4), 18–22.

Restrepo Gómez, B. (2005). Aprendizaje basado en problemas (ABP): una innovación didáctica para la enseñanza universitaria. *Educación y Educadores*, *8*, 9–19. https://www.redalyc.org/pdf/834/83400803.pdf

Salazar, E. Y. H. (2020). SCRUM, un enfoque práctico de metodología ágil para la gestión de proyectos de software. *Revista TIA*, *8*(2), 45-60. https://revistas.udistrital.edu.co/index.php/tia/article/view/12345

Scriven, M., & Paul, R. (2003). *Defining critical thinking: A draft statement for the National Council for Excellence in Critical Thinking*. Foundation for Critical Thinking.

UNESCO. (2018). *Competencias y estándares TIC desde la dimensión pedagógica: Una perspectiva desde los niveles de apropiación de las TIC en la práctica educativa docente*. https://unesdoc.unesco.org/ark:/48223/pf0000265721

Velásquez, B., & Vieira, C. (2021, 15 de marzo). *Proyecto de formación en pensamiento computacional inicia con 80 docentes latinoamericanos*. Universidad del Norte. https://uninorte.edu.co

Villa-Ochoa, J. A., & Castrillón-Yepes, A. (2020). El pensamiento computacional en la formación de profesores de matemáticas: avances y desafíos. *Revista Educación en Ingeniería*, *15*(30), 70-78. https://doi.org/10.26507/rei.v15n30.1123

Wing, J. M. (2006). Computational thinking. *Communications of the ACM*, *49*(3), 33–35. https://doi.org/10.1145/1118178.1118215

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

Wing, J. M. (2011). Research notebook: Computational thinking—What and why? *The Link Magazine*. Carnegie Mellon University. https://www.cs.cmu.edu/link/research-notebook-computational-thinking-what-and-why

Wing, J. M. (2012). *Computational thinking across the curriculum*. [Presentación]. Microsoft Asia Faculty Summit.

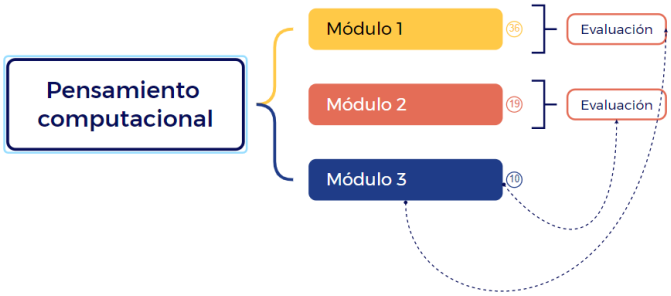
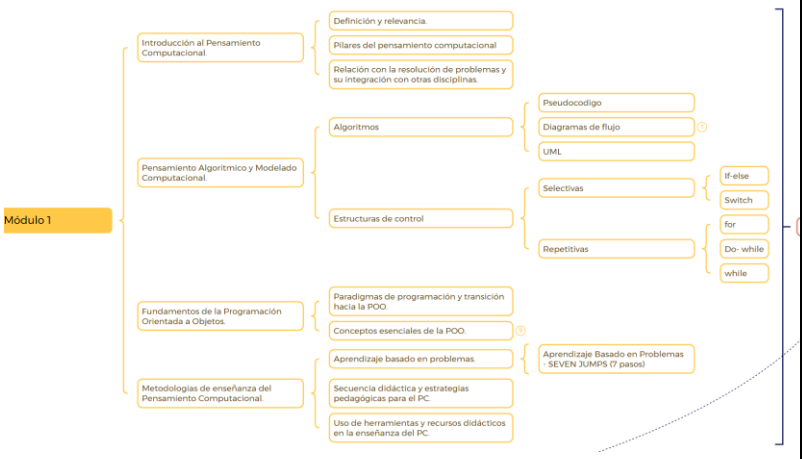
Zeng, Y., Yang, W., & Bautista, A. (2023). Teaching programming and computational thinking in early childhood education: A case study of content knowledge and pedagogical knowledge. *Frontiers in Psychology*, *14*, 1252718. https://doi.org/10.3389/fpsyg.2023.1252718

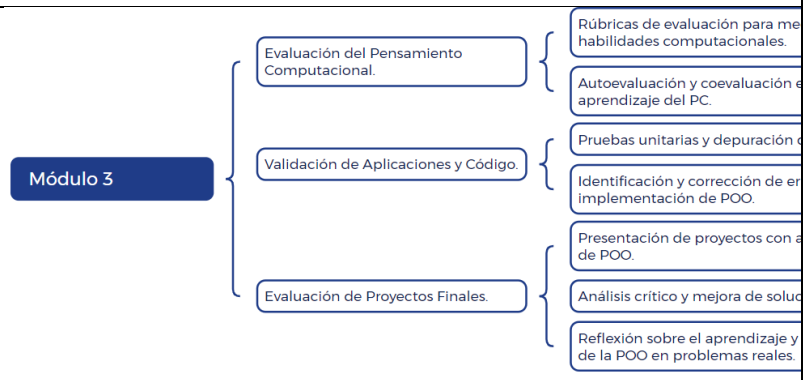
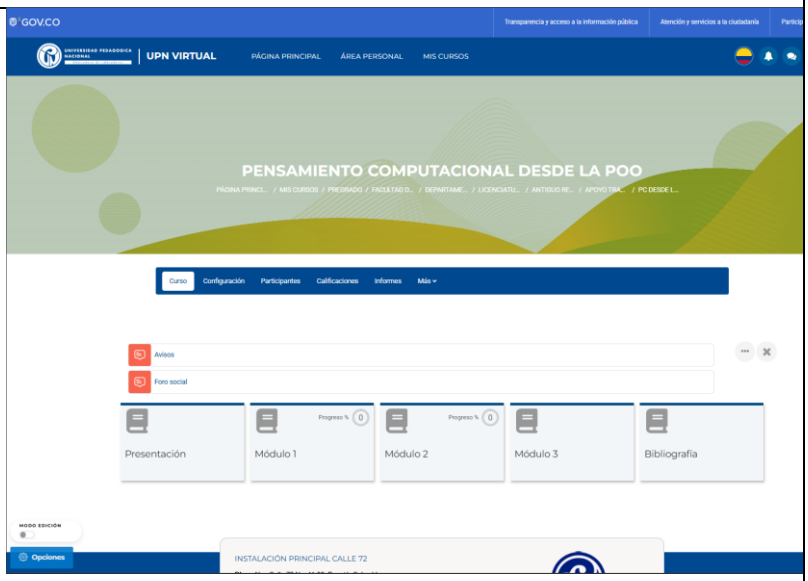
Anexos

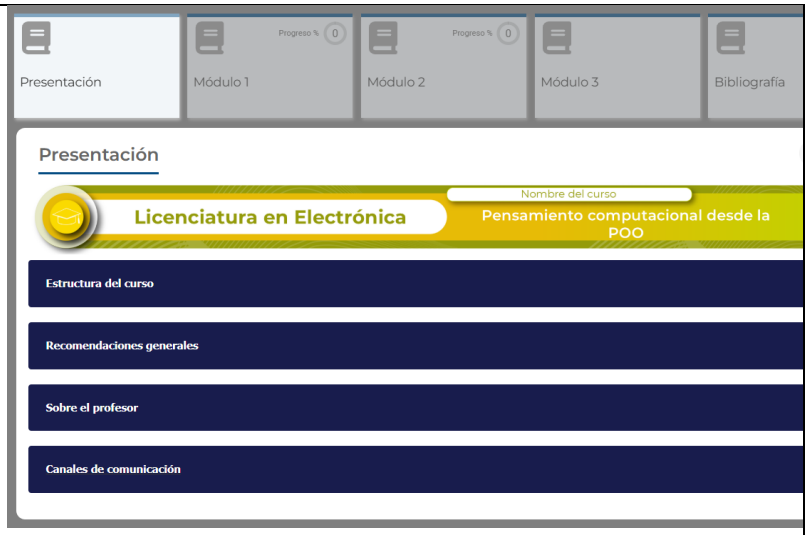
Anexo A: Contenido del curso

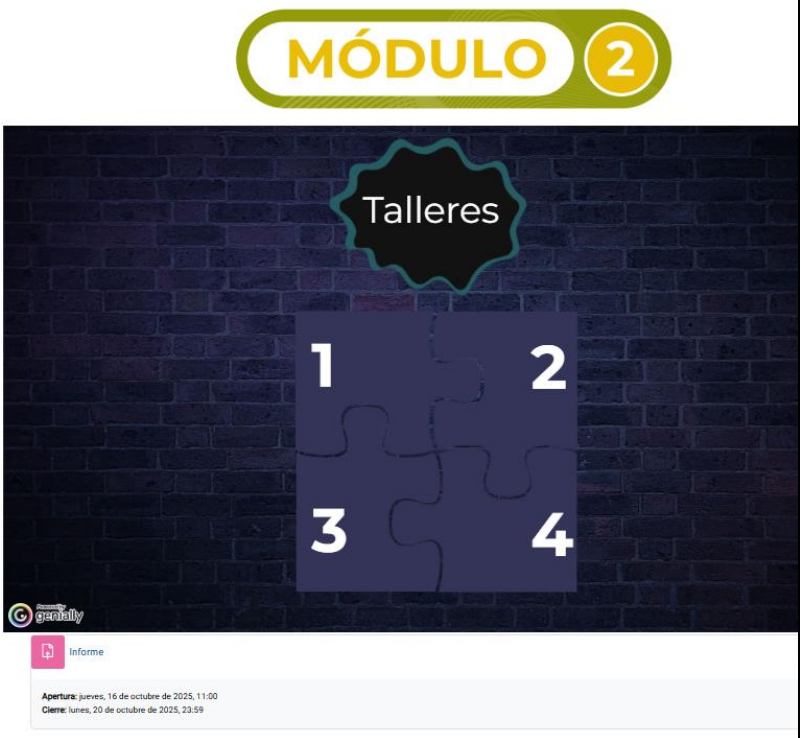
Tabla 8.

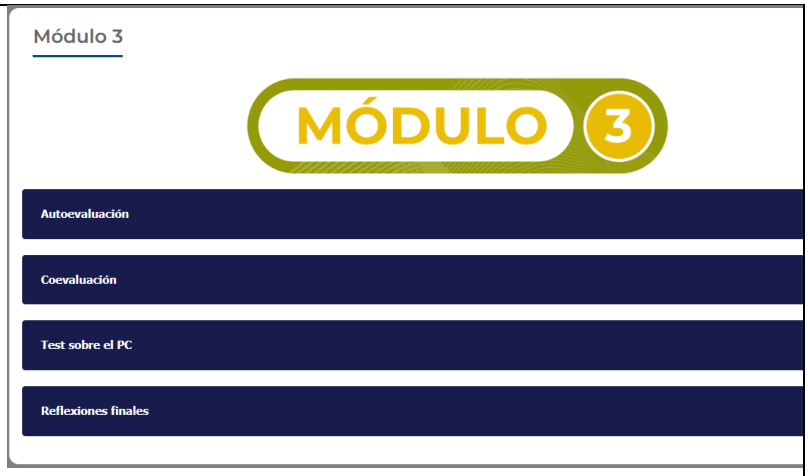
Pantallazos sobre el contenido del curso

Estructura del curso	
Módulo 1	

Módulo 2	
Módulo 3	
Página principal del curso en Moodle	

Presentación del curso en Moodle	
Módulo 1 en Moodle	

Contenido del módulo 1 en Moodle	
Módulo 2 en Moodle	

Módulo 3 en Moodle	

Bibliografía en Moodle

Bibliografía

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

Wing, J. (2011). Research Notebook: Computational Thinking-What and Why. *The Link/Carnegie Mellon*.

Wing, J. M. (2008). Computational thinking and thinking about computing. *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 366(1881

Fedesoft. (s.f.). *Fedesoft: Federación Colombiana de la Industria de Software y TI*. https://fedesoft.org/

Llorens Largo, F., García-Peñalvo, F. J., Molero Prieto, X., & Vendrell Vidal, E. (2017). La enseñanza de la informática, la programación y el pensamiento computacional en los estudios pre *Education in the Knowledge Society (EKS)*, 18(2), 7-17.

Álvarez, M. C., Ocampo, L. M., & Torres, S. A. C. (2024). Definiciones del pensamiento computacional. Una revisión de la literatura. *Revista EIA*, 21(42), 4207-pp.

Resnick, M. (2008). Sowing the seeds for a more creative society. *Learning & Leading with Technology*, 35(4), 18-22.

Osorio Gómez, H. M. (2023). Enseñanza del pensamiento computacional en niños, niños y adolescentes, desde una mirada interdisciplinar.

Rodríguez Chitiva, P. A. (2021). Estrategias para el desarrollo de habilidades de pensamiento computacional: experiencia de aprendizaje en el Seminario Pensamiento Computacional.

Ortega, W. (2020). Recursos didácticos para el desarrollo del pensamiento computacional en estudiantes de básica secundaria y media. *Monografía[. Repositorio Institucional UNAD*. http *edu. co/handle/10596/37455*.

Pérez Oñate, C. A., & Urrea Rodríguez, L. M. (2022). Efectos de las actividades conectadas y desconectadas en el desarrollo del pensamiento computacional y en la aplicación de concept durante la solución de problemas de programación siguiendo el modelo de progresión de tres estados.

Pinzón, H. (2022). Secuencia didáctica para la enseñanza del pensamiento computacional con el uso de la tarjeta programable micro: Bit, para estudiantes de 8 grado de educación básic secundaria. *Recuperado de: http://hdl. handle. net/20.500. 12749*, 18769.

Rondón Barragán, G. A. (2020). Propuesta para desarrollar habilidades de pensamiento computacional en estudiantes de décimo grado del Colegio Facundo Navas Mantilla.

López, L. (2013). *Metodología de la programación orientada a objetos*. Alpha Editorial.

Delgado, F. P., & Velazquez Amador, C. E. (2014). Problemario de algoritmos resueltos con diagramas de flujo y Pseudocodigo. Agua Caliente. Anniversario UAA.

Roughgarden, T. (2020). *Algorithms Illuminated: Part 1 - The Basics* (2.ª ed.). Soundlikeyourself Publishing.

Gillespie, T. (2020). Algorithmically recognizable: Algorithms, platforms, and power. *Social Media + Society*, 6(2), 1–3. https://doi.org/10.1177/2056305120915619

Chinofunga, M. D., Chigeza, P., & Taylor, S. (2025). How can procedural flowcharts support the development of mathematics problem-solving skills? *Mathematics Education Research Jou* https://doi.org/10.1007/s13394-024-00483-3

Pan, H., Zhang, Q., Careaga, C., Dragut, E., & Latecki, L. J. (2024). FlowLearn: Evaluating Large Vision-Language Models on Flowchart Understanding. *arXiv preprint*. https://arxiv.org/abs/2

Allawi, N. (2020). What is the FlowChart. *ResearchGate*. https://www.researchgate.net/publication/342572223_What_is_the_FlowChart

Arlow, J., & Neustadt, I. (2005). *UML 2 and the unified process: Practical object-oriented analysis and design* (2nd ed.). Addison-Wesley.

Rumbaugh, J., Jacobson, I., & Booch, G. (2005). *The unified modeling language reference manual* (2nd ed.). Addison-Wesley.

Pérez, A., Castellanos, L. A., & Gómez, L. F. (2021). *UML: Una manera de representar, interpretar, analizar y desarrollar proyectos multidisciplinares*

López, L. (2013). *Metodología de la programación orientada a objetos*. Alpha Editorial.

mpru.github.io. (s. f.). *Estructuras de control* (Introducción a la Programación). Recuperado de https://mpru.github.io/introprog/estructuras-de-control.html

Alvarritmos. (s. f.). *Estructuras de control: ejemplos y tipos explicados*. Recuperado de https://alvarritmos.com/estructuras-de-control-ejemplos-tipos-explicados/

KeepCoding. (s. f.). *¿Qué son las estructuras de control en programación?* Recuperado de https://keepcoding.io/blog/que-son-estructuras-de-control-en-programacion/

Joyanes Aguilar, L. Programación orientada a objetos. (1996).

Berges, M. P. (2015). *Object-oriented programming through the lens of computer science education* (Doctoral dissertation). Technische Universität München. https://doi.org/10.14459/20

Floyd, R. W. (2005). *The paradigms of programming*. Resonance, 10(5), 86–98. Reimpreso de *Communications of the ACM*, 22(8), 455–460 (1979).

Ávila, J. y Balón, J. (2022). *Clases y objetos. En Análisis y diseño en POO*. Portal Académico del CCH, UNAM. https://portalacademico.cch.unam.mx/cibernetica1/analisis-y-diseno-en-p

Ávila, J. y Balón, J. (2023). *Métodos. En Principios de la Programación Orientada a Objetos*. Portal Académico del CCH, UNAM. https://portalacademico.cch.unam.mx/cibernetica2/prin orientada-a-objetos/metodos

Moreno Díaz, Ó. (s. f.). *Clases y objetos*. En *Programación orientada a objetos* (módulo de Aula En Abierto del INTEF). Instituto Nacional de Tecnologías Educativas y de Formación del Pro de https://formacion.intef.es/aulaenabierto/mod/book/view.php?id=5149&chapterid=6791

Cerón Garnica, C. (s.f.). *POO en Java*. Recurso educativo abierto. Ecosistema BUAP. Benemérita Universidad Autónoma de Puebla. Recuperado de https://ecosistema.buap.mx/forms/file 23/index.html

Portal Académico CCH-UNAM. (2023, 8 de junio). Modificadores de acceso. https://portalacademico.cch.unam.mx/cibernetica2/principios-programacion-orientada-a-objetos/modificado

Tutoriales Programación Ya. (s.f.). 38 - POO - Parte private, protected y public. https://www.tutorialesprogramacionya.com/delphiya/detalleconcepto.php?punto=38&codigo=120&inicio=3

El Libro de Python. (s.f.). Encapsulamiento. https://elibrodepython.com/encapsulamiento-poo

Escuela Directa. (2021, 23 diciembre). Encapsulamiento - Pilares de la Programación Orientada a Objetos. https://escueladirecta-blog.blogspot.com/2021/10/encapsulamiento-pilares-de

NetMentor. (2025, 19 septiembre). Encapsulamiento en programación orientada a objetos. https://www.netmentor.es/entrada/encapsulamiento-poo

OpenWebinars. (2023, 9 noviembre). Introducción a POO en Java: Encapsulamiento. https://openwebinars.net/blog/introduccion-a-poo-en-java-encapsulamiento/

Portal Académico CCH-UNAM. (2022, 8 mayo). Características de la POO. https://portalacademico.cch.unam.mx/cibernetica1/algoritmos-y-codificacion/caracteristicas-POO

Microsoft. (2025, 18 junio). Programación orientada a objetos: herencia - C#. https://learn.microsoft.com/es-es/dotnet/csharp/fundamentals/object-oriented/inheritance

OpenWebinars. (2023, 16 noviembre). Introducción a POO en Java: Herencia y polimorfismo. https://openwebinars.net/blog/introduccion-a-poo-en-java-herencia-y-polimorfismo/

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

Eggen, P., & Kauchak, D. (2015). *Estrategias docentes: Enseñanza de contenidos curriculares y desarrollo de habilidades de pensamiento*. Pearson Educación.

Mamo, R. M., Jonassen, D. H., Palmer, B., & Lufi, S. (2014). Why problem-based learning works: Theoretical foundations. *Journal on Excellence in College Teaching*, 25(3-4), 221-238.

Nilson, L. B. (2010). *Teaching at its best: A research-based resource for college instructors* (3rd ed.). Jossey-Bass.

Restrepo, B. (2005). Aprendizaje basado en problemas (ABP): una innovación didáctica para la enseñanza universitaria. *Educación y Educadores*, 8, 9-19.

Schmidt, H. G. (1983). Problem-based learning: Rationale and description. *Medical Education*, 17(1), 11-16.

Zwaal, W., & Otting, H. (2016). Performance of the seven-step procedure in problem-based learning. *Journal of Problem Based Learning in Higher Education*, 4(1), 1-11.

Nota. Fuente: Elaboración propia.

Anexo B: Instrumentos de evaluación

Evaluación de talleres

Tabla 9.
Primer instrumento de evaluación

Criterio	Inicial (1-2)	En desarrollo (2-3)	Competente (3-4)	Avanzado (4-5)
Abstracción	Identifica superficialmente los elementos clave del problema.	Distingue algunos elementos esenciales, pero mezcla detalles innecesarios.	Selecciona y representa adecuadamente los elementos esenciales.	Generaliza y modela el problema de manera eficiente, anticipando variaciones.
Descomposición	Divide el problema en partes mínimas o poco funcionales.	Separa el problema en subproblemas con pasos redundantes.	Descompone en pasos lógicos y funcionales.	Organiza y jerarquiza subproblemas de manera óptima, reutilizando componentes.
Reconocimiento de patrones	No identifica patrones o repite soluciones sin justificación	Reconoce patrones simples, pero no los aplica consistentemente.	Identifica y aplica patrones, optimizando la solución.	Generaliza patrones y los adapta a nuevos contextos, proponiendo mejoras.
Generalización	Aplica soluciones solo a un caso específico.	Adapta soluciones con errores o limitaciones.	Reutiliza soluciones con éxito.	Propone soluciones escalables y transferibles.
Depuración	No detecta errores o depende de otros.	Identifica algunos errores con ayuda.	Corrige errores de forma autónoma.	Anticipa posibles errores e implementa pruebas de validación.

Nota. Fuente: Elaboración propia.

Autoevaluación

Tabla 10.
Segundo instrumento de evaluación

Aspecto	1	2	3	4	5
Comprendí y apliqué los pilares del pensamiento computacional.					
Detecté y corregí errores en mi código de forma autónoma.					
Logré explicar mi solución con claridad y lenguaje pedagógico.					
Participé activamente en el trabajo colaborativo y la documentación.					
Desarrollé material educativo útil para otros estudiantes.					

Nota. Fuente: Elaboración propia.

Coevaluación

Tabla 11.

Tercer instrumento de evaluación.

Aspecto	1	2	3	4	5
Explica claramente el código y los conceptos teóricos.					
Aporta soluciones efectivas y colabora con su grupo.					
Escucha y retroalimenta a cada compañero o compañera.					
Propone mejoras y demuestra pensamiento crítico.					

Nota. Fuente: Elaboración propia.

Test sobre el PC

Tabla 12.
Cuarto instrumento de evaluación

N°	Pregunta	Opciones
1	¿Cuál de los siguientes enunciados describe mejor el proceso de *abstracción*?	a) Dividir un problema complejo en partes pequeñas. b) Identificar y conservar solo los elementos esenciales del problema. c) Reconocer patrones que se repiten. d) Corregir errores en un código.
2	Cuando un programador identifica errores de sintaxis y los corrige en el código, está aplicando principalmente el pilar:	a) Abstracción b) Depuración c) Descomposición d) Generalización
3	Un estudiante adapta un algoritmo de cálculo del área de un triángulo para figuras rectangulares. ¿Qué pilar aplica?	a) Generalización b) Reconocimiento de patrones c) Descomposición d) Abstracción
4	Cuando un estudiante descompone un problema complejo en pasos más simples, está aplicando el pilar de:	a) Generalización. b) Descomposición. c) Depuración. d) Reconocimiento de patrones.
5	Identificar similitudes entre distintos problemas para aplicar soluciones previas corresponde a:	a) Reconocimiento de patrones. b) Abstracción. c) Depuración. d) Polimorfismo.
6	En el proceso de enseñanza, los pilares del pensamiento computacional contribuyen principalmente a:	a) Automatizar tareas sin comprensión. b) Promover la memorización de algoritmos. c) Fomentar el razonamiento

		lógico y la resolución creativa de problemas. d) Limitar la exploración de los estudiantes.
7	Una clase en POO puede definirse como:	a) Una instancia individual del programa. b) Una plantilla que define atributos y métodos para crear objetos. c) Un bloque de código que ejecuta tareas automáticas. d) Una variable temporal.
8	El encapsulamiento se utiliza para:	a) Permitir acceso libre a los atributos. b) Ocultar la información y proteger los datos. c) Crear múltiples instancias de una clase. d) Reutilizar código mediante herencia.
9	La herencia permite:	a) Copiar atributos de otras clases sin modificaciones. b) Crear nuevas clases a partir de clases existentes. c) Ocultar la información sensible. d) Ejecutar métodos repetidamente.
10	El polimorfismo sirve para	a) Que un objeto tome distintas formas. b) Crear nuevas clases a partir de clases existentes. c) Ocultar la información sensible. d) Ejecutar métodos repetidamente.
11	Un método en POO	a) Una instancia individual del programa. b) Una plantilla que define atributos y métodos para crear objetos. c) Representa o describe el comportamiento de algún objeto. d) Una variable temporal.
12	Un atributo es	a) Una instancia individual del programa.

<table>
<tr><td></td><td></td><td colspan="2">b) Una plantilla que define atributos y métodos para crear objetos.
c) Representa o describe el comportamiento de algún objeto.
d) Una característica de los objetos.</td></tr>
<tr><td>13</td><td>Un objeto representa</td><td colspan="2">a) Una instancia individual del programa.
b) Una plantilla que define atributos y métodos para crear objetos.
c) Algún elemento del mundo real.
d) Una característica de los objetos.</td></tr>
<tr><td>14</td><td>

```python
class Promedio:
    def __init__(self):
        self.suma = 0
    def calcular(self, n):
        for i in range(n):
            nota = str(input("Ingrese nota: "))
            self.suma += nota
        promedio = self.suma / n
        print("El promedio es: ", promedio)
if __name__ == "__main__":
    calc = Promedio()
    calc.calcular(2)
```

</td><td>¿Qué errores contiene el código?</td><td>¿Qué pilar del pensamiento computacional se aplica al corregir el error?</td></tr>
<tr><td>15</td><td>

```java
import java.util.Scanner;
public class Promedio {
    int suma;
    public Promedio() {
        suma = 0;
    }
    public void calcular() {
        Scanner sc = new Scanner(source:System.in);
        System.out.print(s: "¿Cuántas notas ingresará? ");
        int n = sc.nextInt();
        for (int i = 0; i <= n; i++) {
            System.out.print(s: "Ingrese nota: ");
            String nota = sc.nextLine();
            suma += nota;
        }

        double promedio = suma / n;
        System.out.println("El promedio es: " + promedio);
    }
    public static void main(String[] args) {
        Promedio p = new Promedio();
        p.calcular();
    }
}
```

</td><td>¿Qué errores contiene el código?</td><td>¿Qué pilar del pensamiento computacional se aplica al corregir el error?</td></tr>
</table>

Nota. Fuente: Elaboración propia.

Reflexión

1. ¿De qué manera los pilares del pensamiento computacional favorecen la enseñanza de la programación en el aula?
2. ¿Qué tan pertinente es este curso para su desarrollo cognitivo a lo largo de la carrera?
3. ¿Qué estrategias emplearía para fomentar la solución de problemas en sus futuros estudiantes?
4. ¿Cómo contribuye el pensamiento computacional a la formación integral de un maestro en la licenciatura en electrónica?
5. Reflexione sobre el impacto del trabajo colaborativo en la creación de soluciones computacionales educativas.
6. ¿Como explicaría la temática impartida en este curso para estudiantes de educación media usando material educativo?

Anexo C: Links de presentaciones y documento base

Tabla 13.
Links de consulta.

Estructura del curso	https://view.genially.com/6834b1986c48dfebcac97d41/horizontal-infographic-diagrams-estructura-del-curso
Estructura del módulo 1	https://view.genially.com/68be90c1de21529dbeaccf87/interactive-content-estructura-del-modulo-1
Sobre el pensamiento computacional	https://view.genially.com/67fd9eb3276c18e5b86ddc50/interactive-content-introduccion-al-pensamiento-computacional
Pensamiento algorítmico y modelado computacional	https://view.genially.com/6834cde06a1059131067e181/interactive-content-pensamiento-algoritmico-y-modelado-computacional
Fundamentos de la programación orientada a objetos	https://view.genially.com/68bea1c9fa3e52e95fcd98d2/interactive-content-fundamentos-de-la-poo
Metodologías de enseñanza del pensamiento computacional	https://view.genially.com/68da4aaec6b707012fcaffdf/presentation-metodologias-de-ensenanza-del-pensamiento-computacional
Talleres	https://view.genially.com/68e3d7a2e068022d39eaa060/interactive-content-1
Autoevaluación	https://forms.gle/cUFzUnkFXLfck46p9
Coevaluación	https://forms.gle/XrULMNBEcfhPaZqK9
Test sobre el PC	https://forms.gle/GfhvPSQn5x46Ha189
Reflexión	https://forms.gle/kC3qYNZdSf9HyDHX7
Documento base	https://docs.google.com/document/d/1UzGtKDoEXWl3H_bGzx39vFG_8FvAULjw/edit?usp=drive_link&ouid=102564928374808655323&rtpof=true&sd=true

Nota. Fuente: Elaboración propia