UNIVERSIDAD DEL VALLE DE GUATEMALA

Faculta de Ingeniería

Plataforma de evaluación automatizada de código, con IA adaptativa, para la enseñanza de programación en Python, en el Instituto Tecnológico Experimental del área urbana del Municipio de Sololá.

Trabajo de graduación presentado por Víctor Isaías Cosiguá Saloj para optar al grado académico de Licenciado en Tecnología de Sistemas Informáticos

Guatemala
2025

Plataforma de evaluación automatizada de código, con IA adaptativa, para la enseñanza de programación en Python, en el Instituto Tecnológico Experimental del área urbana del Municipio de Sololá.

UNIVERSIDAD DEL VALLE DE GUATEMALA

Faculta de Ingeniería

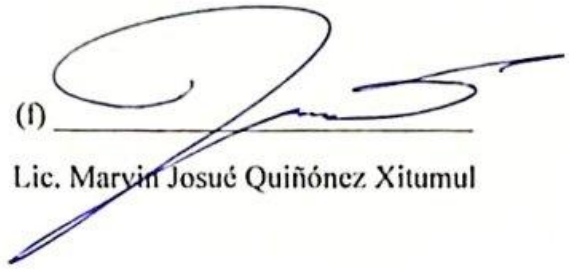
Plataforma de evaluación automatizada de código, con IA adaptativa, para la enseñanza de programación en Python, en el Instituto Tecnológico Experimental del área urbana del Municipio de Sololá.

Trabajo de graduación presentado por Víctor Isaías Cosiguá Saloj para optar al grado académico de Licenciado en Tecnología de Sistemas Informáticos

Guatemala

2025

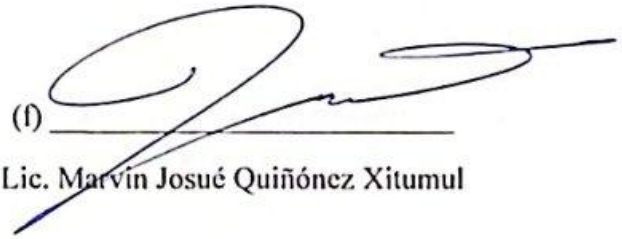
Vo.Bo.:



(f) _____

Lic. Marvin Josué Quiñónez Xitumul

Tribunal Examinador:

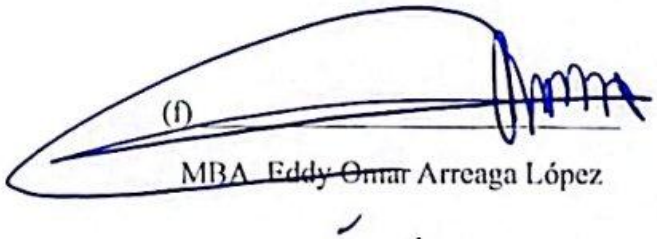


(f) _____

Lic. Marvin Josué Quiñónez Xitumul

(f) _____

Ing. Josué Alberto Herrera Cifuentes



(f) _____

MBA. Eddy Omar Arreaga López

Fecha de aprobación: Guatemala, 22 de noviembre de 2025.

PREFACIO

La enseñanza de la programación en contextos con recursos limitados plantea importantes desafíos pedagógicos y tecnológicos. Este proyecto surge del interés por apoyar a los estudiantes en el fortalecimiento de sus habilidades en programación, específicamente en el lenguaje Python, al tiempo que se les introduce a nuevas tecnologías accesibles como el dispositivo Raspberry Pi.

La propuesta se materializa en una plataforma de evaluación automatizada con inteligencia artificial adaptativa, diseñada para operar sin conexión a internet y responder a las condiciones de infraestructura limitadas. La investigación fue guiada por el acompañamiento metodológico del Lic. Marvin Josué Quiñónez Xitumul, quien brindó orientación para garantizar la pertinencia técnica y académica del trabajo.

Víctor Isaías Cosiguá Saloj

Autor

ÍNDICE

LISTA DE CUADROS

LISTA DE FIGURAS

Nota sobre confidencialidad institucional

En este documento se utiliza el nombre “Instituto Tecnológico Experimental de Sololá” como una denominación ficticia, con el fin de preservar la identidad real de la institución participante, conforme a solicitud expresa de sus autoridades. Todos los datos, contexto y resultados presentados corresponden a una institución real del departamento de Sololá.

RESUMEN

La enseñanza de programación en el Bachillerato en Computación en el Instituto Tecnológico Experimental, casco urbano de Sololá enfrenta tres problemas críticos: falta de retroalimentación inmediata, conectividad intermitente y recursos tecnológicos limitados. Este proyecto aborda estos desafíos con una plataforma offline-first que automatiza la evaluación de código utilizando ASTs y emplea IA adaptativa para personalizar ejercicios según las necesidades de cada estudiante. La solución, desarrollada como una Aplicación Web Progresiva (PWA) con almacenamiento en caché local y soporte mediante servidores Raspberry Pi, garantiza el acceso incluso en ausencia de conexión a internet,

La plataforma utiliza análisis de código con Python y scikit-learn, implementándose en el Instituto Tecnológico Experimental de Sololá con sincronización periódica. Su validación se realiza mediante pruebas comparativas pre/post-implementación, mostrando una reducción en el tiempo de corrección. Esto beneficia tanto a estudiantes como a docentes; los estudiantes reciben retroalimentación más rápida y precisa, mejorando sus competencias según el CNB, mientras que los docentes disponen de herramientas que optimizan su tiempo y les permiten enfocarse en estrategias pedagógicas avanzadas. Además, el modelo es replicable y adaptable para escuelas con desafíos similares en contextos urbanos con recursos limitados.

La innovación radica en su doble adaptabilidad: tecnológica (funciona sin internet) y pedagógica (ajuste automático de dificultad). Esto posiciona la plataforma como herramienta clave para fortalecer la educación tecnológica en contextos urbana

I. INTRODUCCIÓN

La enseñanza de programación en entornos con recursos limitados presenta desafíos como la falta de retroalimentación inmediata, conectividad inestable y escasez de herramientas adaptadas. Esta situación se evidencia en el Instituto Tecnológico Experimental del área urbana del Municipio de Sololá, donde formar competencias digitales en el Bachillerato en Computación es esencial.

Ante esta necesidad, se plantea una plataforma web offline-first de evaluación automatizada de código en Python, basada en Árboles de Sintaxis Abstracta (ASTs) e inteligencia artificial adaptativa. Utiliza dispositivos Raspberry Pi y almacenamiento local con SQLite, lo que garantiza su funcionamiento sin conexión a internet.

La solución combina un enfoque tecnológico y pedagógico. Automatiza la evaluación del código, reduce la carga docente y ofrece retroalimentación inmediata. Además, adapta los ejercicios al rendimiento del estudiante, optimizando el aprendizaje.

Este documento expone el marco teórico, los objetivos y la metodología del proyecto, alineado con el Currículo Nacional Base (CNB), y con potencial de implementación en otros contextos educativos similares.

II. OBJETIVOS

A. Objetivo general

Desarrollar una plataforma web offline-first de evaluación automatizada de código con IA adaptativa para la enseñanza de programación en Python en el Instituto Tecnológico Experimental Sololá, garantizando accesibilidad en entornos urbanos con recursos limitados y alineación al CNB.

B. Objetivos específicos

1. Implementar un evaluador de código en Python usando Abstract Syntax Trees (AST) para detectar errores de lógica y eficiencia, comparando el código estudiantil con soluciones modelo.

2. Diseñar un sistema de IA adaptativa, basado en machine learning supervisado, que ajuste automáticamente la complejidad de los ejercicios según el desempeño del estudiante.

3. Implementar una arquitectura técnica optimizada y offline-first para la plataforma, utilizando base de datos para almacenamiento local y servidor Raspberry Pi con PWA y caché local, garantizando sincronización periódica y acceso continuo en el Instituto Tecnológico Experimental Sololá, incluso en entornos con conectividad limitada.

III. JUSTIFICACIÓN

La transformación digital en el ámbito educativo ha generado nuevas oportunidades para mejorar los procesos de enseñanza y aprendizaje, especialmente en áreas técnicas como la programación. Sin embargo, en contextos urbanos como el Municipio de Sololá, persisten desafíos relacionados con el acceso desigual a la tecnología y la escasa disponibilidad de recursos educativos digitales. Frente a esta realidad, diversas investigaciones y organismos internacionales han propuesto el uso de soluciones tecnológicas adaptativas y automatizadas como herramientas clave para reducir la brecha digital y mejorar la calidad educativa.

La enseñanza de programación en el primer y segundo año de bachillerato en Computación del Instituto Tecnológico Experimental, Sololá, enfrenta tres problemas clave que limitan el aprendizaje efectivo: la falta de retroalimentación inmediata, el acceso intermitente a internet en el 70% de las instituciones y la ausencia de personalización en los ejercicios prácticos. Estas barreras generan deserción y reducen la calidad de la formación en programación, afectando la empleabilidad de los estudiantes en un mercado laboral cada vez más digitalizado.

El Banco Interamericano de Desarrollo (2020) ha documentado el impacto positivo de las tecnologías digitales en la educación de América Latina, destacando su capacidad para ampliar el acceso, personalizar el aprendizaje y optimizar la gestión del conocimiento. De igual forma, la UNESCO (2023) enfatiza que el uso de inteligencia artificial en el entorno educativo puede fortalecer la interacción entre docentes y estudiantes, adaptando los contenidos al nivel de comprensión de cada alumno.

En el campo de la programación, la evaluación automatizada del código ha cobrado relevancia como una herramienta eficaz para detectar errores sintácticos y lógicos sin intervención humana. Nystrom (Crafting Interpreters, s.f.) describe el uso de Árboles de Sintaxis Abstracta (AST) como una técnica eficiente para el análisis estático del código fuente en Python, lo que permite proporcionar retroalimentación inmediata y precisa, y fomentar el aprendizaje autónomo.

Por otro lado, la inteligencia artificial adaptativa ha sido ampliamente utilizada para personalizar los procesos formativos. Eduteka (2022) reporta que los sistemas educativos que ajustan los contenidos según el desempeño del estudiante pueden mejorar la retención de conocimientos en hasta un 30%. En particular, algoritmos como Random Forest, aplicados en entornos de aprendizaje, permiten generar recomendaciones específicas para cada usuario con base en sus patrones de error y progreso individual (Simplilearn, s.f.).

Adicionalmente, en contextos con limitaciones de conectividad, como ocurre en muchos sectores educativos urbanos de Guatemala, las plataformas con enfoque Offline-First han demostrado ser una solución eficaz. DashDevs (2025) y la Raspberry Pi Foundation (s.f.) proponen arquitecturas ligeras y autónomas, basadas en dispositivos como la Raspberry Pi y

bases de datos embebidas como SQLite, que permiten el funcionamiento de sistemas educativos incluso sin conexión permanente a internet.

Para resolver estos desafíos, este proyecto propone una plataforma offline-first de evaluación automatizada de código, con tres innovaciones clave: evaluación inmediata con ASTs en Python, funcionamiento sin internet mediante PWA y servidor Raspberry Pi, y personalización del aprendizaje con IA ligera basada en scikit-learn. Implementada en el Instituto Tecnológico Experimental Sololá, la plataforma reducirá el tiempo de corrección, permitirá el acceso equitativo sin depender de la conectividad y ajustará la dificultad de los ejercicios según el desempeño de cada estudiante.

La urgencia de esta intervención radica en su alineación con el Currículo Nacional Base, respondiendo directamente a la necesidad de fortalecer competencias digitales esenciales. Además, este modelo es escalable y replicable, permitiendo su adopción en otras instituciones con condiciones similares en zonas urbanas. Más allá del ámbito académico, la plataforma facilita la democratización del acceso a tecnologías educativas avanzadas, asegurando que los estudiantes tengan herramientas para enfrentar los retos del siglo XXI y acceder a mejores oportunidades educativas y laborales.

IV. MARCO TEÓRICO

A. Contexto educativo y problemática

La transformación digital en el ámbito educativo ha generado nuevas oportunidades para mejorar los procesos de enseñanza y aprendizaje. Sin embargo, en contextos urbanas como el Municipio de Sololá persisten desafíos relacionados con el acceso desigual a la tecnología y la escasa disponibilidad de recursos educativos digitales.

1. Brecha digital en áreas urbanas de Guatemala

El acceso desigual a la tecnología en Guatemala constituye un obstáculo significativo para la educación digital. La vicepresidencia de la República de Guatemala ha reconocido la importancia de cerrar la brecha digital a nivel nacional, subrayando la necesidad de mejorar el acceso a herramientas tecnológicas tanto en áreas rurales como urbanas. Esta política reconoce que la transformación digital no solo depende de la disponibilidad de dispositivos, sino también de conectividad estable, capacitación docente y contenidos educativos adaptados al contexto local (Vicepresidencia de la República de Guatemala, s.f.).

En el departamento de Sololá, área urbana del municipio, la situación refleja estas limitaciones. Aunque los centros educativos cuentan con infraestructura básica de computación, la conectividad a internet es intermitente y los recursos tecnológicos presentan capacidades limitadas. Esta realidad genera desventajas frente a otros entornos con mayor disponibilidad tecnológica, afectando la equidad en el acceso a oportunidades de aprendizaje digital.

El Banco Interamericano de Desarrollo (2020) ha documentado el impacto positivo de las tecnologías digitales en la educación de América Latina, destacando su capacidad para ampliar el acceso, personalizar el aprendizaje y optimizar la gestión del conocimiento. Sin embargo, estas ventajas solo se materializan cuando las instituciones educativas cuentan con infraestructura adecuada y estrategias pedagógicas que integren efectivamente la tecnología en el proceso formativo.

2. Desafíos en la enseñanza de programación

La enseñanza de programación en el nivel medio enfrenta tres problemas críticos que limitan el desarrollo efectivo de competencias digitales en los estudiantes. Estos desafíos se manifiestan de manera particular en instituciones como el Instituto Tecnológico Experimental del área urbana del Municipio de Sololá, donde la formación en programación constituye un componente central del Bachillerato en Computación.

El primer desafío es la falta de retroalimentación inmediata en la evaluación del código. En el método tradicional, los docentes revisan manualmente los ejercicios de programación, lo que genera retrasos significativos entre la entrega del trabajo y la recepción de observaciones. Esta demora impide que los estudiantes identifiquen con rapidez sus errores,

limitando así el aprendizaje autónomo y progresivo. Según Eduteka (2022), los sistemas que ajustan los contenidos en función del desempeño individual pueden mejorar la retención de conocimientos en hasta un 30%, evidenciando la importancia de la retroalimentación oportuna.

El segundo desafío es la conectividad intermitente a internet. En el Instituto Tecnológico Experimental Sololá, el acceso estable a internet no está garantizado durante todas las sesiones de clase. Esta condición restringe el uso de plataformas educativas en línea y la interacción con recursos digitales actualizados. La UNESCO (2023) enfatiza que el uso de tecnologías digitales en el entorno educativo debe considerar las condiciones de infraestructura local para garantizar su efectividad y sostenibilidad.

El tercer desafío es la ausencia de personalización en los procesos de enseñanza. Las actividades se diseñan de manera homogénea, sin considerar los distintos ritmos de aprendizaje ni las dificultades específicas de cada estudiante. Esto incrementa la desmotivación, favorece la deserción académica y disminuye la calidad en la formación en programación. La personalización del aprendizaje mediante tecnologías adaptativas ha demostrado ser efectiva para mantener el compromiso estudiantil y mejorar los resultados académicos (Universidad Rafael Landívar, 2024).

Estos tres problemas convergen en una situación crítica: los estudiantes enfrentan un aprendizaje fragmentado, bajos niveles de retención y una reducción de sus oportunidades de empleabilidad futura en el ámbito tecnológico. La urgencia de esta problemática radica en que la programación constituye una competencia esencial en el siglo XXI, fundamental para la inserción laboral en un mercado cada vez más digitalizado.

3. El Currículo Nacional Base (CNB) y competencias digitales

El Currículo Nacional Base (CNB) de Guatemala establece competencias específicas para la formación en computación en el nivel medio. En el Bachillerato en Computación, el CNB define como competencias fundamentales el dominio de estructuras de control, la resolución algorítmica de problemas y la capacidad de desarrollar programas funcionales utilizando lenguajes de programación modernos.

Para el área de programación, el CNB especifica que los estudiantes deben ser capaces de analizar problemas, diseñar soluciones algorítmicas, implementarlas en código y evaluar su eficiencia. Estas competencias se desarrollan progresivamente durante los dos años de bachillerato, iniciando con conceptos básicos como variables y operadores, avanzando hacia estructuras de control y finalizando con algoritmos más complejos.

El CNB también establece la necesidad de formar estudiantes con pensamiento lógico-matemático aplicado a la resolución de problemas computacionales. Esta competencia implica la capacidad de abstraer situaciones reales, modelarlas mediante estructuras de datos y procesos algorítmicos, e implementar soluciones eficientes. La evaluación de estas

competencias requiere no solo verificar la corrección sintáctica del código, sino también analizar la lógica subyacente y la eficiencia de las soluciones propuestas.

En este contexto, la integración de tecnologías educativas que faciliten el desarrollo de estas competencias resulta fundamental. El Ministerio de Educación de Guatemala (2024) ha impulsado iniciativas como Mineduc Digital, orientadas a fortalecer la educación tecnológica mediante el uso de recursos digitales. Estas iniciativas reconocen que el aprendizaje de programación requiere práctica constante, retroalimentación inmediata y oportunidades de experimentación en entornos controlados.

La alineación entre las herramientas tecnológicas y los objetivos curriculares constituye un requisito indispensable para garantizar que las innovaciones educativas contribuyan efectivamente al desarrollo de competencias. Por ello, la plataforma propuesta en este proyecto se diseña considerando explícitamente las competencias definidas en el CNB, asegurando que los ejercicios, la evaluación y la retroalimentación estén directamente vinculados con los resultados de aprendizaje esperados.

B. Evaluación automatizada de código

La evaluación automatizada de código se ha consolidado como una herramienta innovadora en la enseñanza de la programación, al permitir analizar la lógica y la sintaxis de los programas sin necesidad de intervención manual.

1. Fundamentos del análisis estático de código

El análisis estático de código constituye una técnica fundamental en la ciencia de la computación que permite examinar programas sin ejecutarlos. A diferencia del análisis dinámico, que requiere la ejecución del código para detectar errores en tiempo real, el análisis estático inspecciona la estructura y sintaxis del programa de manera anticipada. Esta capacidad resulta particularmente valiosa en entornos educativos, donde se busca identificar errores comunes antes de que los estudiantes ejecuten sus programas.

El análisis estático opera mediante la transformación del código fuente en representaciones abstractas que facilitan su inspección sistemática. Estas representaciones permiten detectar patrones de error recurrentes, verificar el cumplimiento de estándares de codificación y evaluar la eficiencia de las soluciones propuestas. En el contexto educativo, esta técnica permite proporcionar retroalimentación específica sobre aspectos como el uso correcto de estructuras de control, la gestión adecuada de variables y la aplicación de buenas prácticas de programación.

La aplicación del análisis estático en la enseñanza de programación ofrece ventajas significativas frente a métodos tradicionales de evaluación. Permite procesar grandes volúmenes de código de manera eficiente, mantiene consistencia en los criterios de evaluación y libera tiempo docente para actividades pedagógicas de mayor complejidad.

Además, al operar sin necesidad de ejecutar el código, elimina riesgos de seguridad asociados con la ejecución de programas potencialmente incorrectos o maliciosos.

2. Árboles de Sintaxis Abstracta (ATS)

Los Árboles de Sintaxis Abstracta constituyen estructuras de datos jerárquicas que representan la gramática de un lenguaje de programación de manera formal y procesable. Los ASTs en el lenguaje Python son ampliamente utilizados como recurso para el análisis automatizado de código. Estas estructuras capturan la esencia semántica del programa, abstrayendo detalles superficiales de sintaxis y facilitando la comparación entre diferentes implementaciones de un mismo algoritmo.

El módulo ast de Python constituye la herramienta estándar para trabajar con representaciones AST, facilitando el desarrollo de analizadores de código y sistemas de evaluación automatizada. Este módulo proporciona capacidades para el procesamiento de árboles sintácticos, siendo fundamental para herramientas de análisis de código, linters y plataformas educativas. La documentación oficial de Python reconoce estas funcionalidades como esenciales para el desarrollo de sistemas de verificación y evaluación de código (Python Documentation, s.f.).

La representación mediante ASTs facilita la identificación de patrones específicos en el código. Por ejemplo, los sistemas de evaluación automatizada pueden detectar cuando un estudiante omite el uso de la función range() en un bucle for, analizando la estructura del árbol sintáctico y comparándola con patrones de referencia. Esta capacidad de detección estructural resulta más robusta que la simple comparación textual, ya que permite reconocer soluciones semánticamente equivalentes, aunque expresadas de manera diferente.

Nystrom, en su obra sobre construcción de intérpretes, describe el uso de Árboles de Sintaxis Abstracta como una técnica eficiente para el análisis estático del código fuente, lo que permite proporcionar retroalimentación inmediata y precisa, fomentando el aprendizaje autónomo (Crafting Interpreters, s.f.). Esta aproximación resulta especialmente útil en contextos educativos donde se busca evaluar no solo la corrección del código, sino también la comprensión conceptual subyacente y la aplicación de principios algorítmicos fundamentales.

3. Sistemas de retroalimentación automática

La retroalimentación automática en plataformas educativas representa un componente esencial para el aprendizaje efectivo de programación. Estos sistemas operan mediante la combinación de análisis técnico del código y generación de mensajes pedagógicamente orientados. A diferencia de sistemas que simplemente indican si una solución es correcta o incorrecta, los sistemas avanzados de retroalimentación proporcionan información específica sobre la naturaleza del error, su localización en el código y sugerencias concretas de corrección.

La implementación de retroalimentación automática en contextos educativos ha demostrado múltiples beneficios. Psicosmart (s.f.) señala que la automatización de la evaluación y retroalimentación en tiempo real mediante inteligencia artificial en entornos educativos permite a los estudiantes identificar con rapidez sus errores, reforzando el aprendizaje autónomo. Esta inmediatez resulta fundamental para mantener el compromiso del estudiante y prevenir la consolidación de conceptos erróneos.

Los sistemas de retroalimentación efectivos requieren un diseño cuidadoso que equilibre especificidad técnica con claridad pedagógica. Los mensajes deben ser suficientemente precisos para orientar la corrección, pero sin revelar completamente la solución, preservando así el valor formativo del ejercicio. Además, deben adaptarse al nivel de conocimiento del estudiante, ofreciendo explicaciones más detalladas para principiantes y observaciones más concisas para estudiantes avanzados.

La automatización de la evaluación también representa una ventaja significativa para el cuerpo docente, al reducir el tiempo invertido en tareas repetitivas como la corrección manual de código y exámenes. Esto permite a los docentes enfocarse en aspectos más complejos del proceso formativo, mientras se garantiza un seguimiento más continuo y significativo del aprendizaje (Universidad Rafael Landívar, 2024). De hecho, herramientas como los evaluadores automáticos con análisis de ASTs no solo optimizan la evaluación, sino que contribuyen a mejorar la comprensión lógica del código y la calidad de la enseñanza de la programación en general.

4. Experiencias previas en plataformas de evaluación

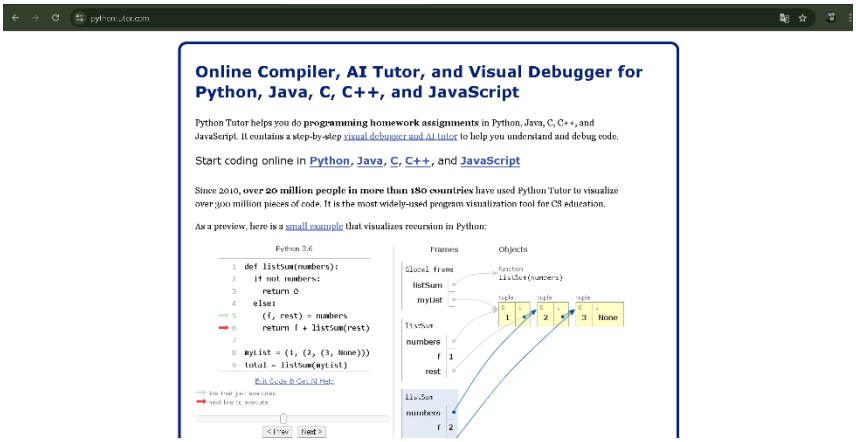
El desarrollo de plataformas educativas orientadas a la enseñanza de programación en contextos con recursos limitados ha cobrado gran relevancia en los últimos años. Estas experiencias aportan lecciones valiosas sobre diseño tecnológico, implementación pedagógica y sostenibilidad a largo plazo, constituyéndose como referentes para nuevas iniciativas en la región.

Jutge.org representa uno de los casos más documentados en el ámbito de evaluación automática de código. Esta plataforma, utilizada en instituciones como la Universitat Politècnica de Catalunya, permite a los estudiantes enviar soluciones a problemas de programación y recibir retroalimentación inmediata sobre la corrección sintáctica y funcional de su código. Su principal fortaleza radica en la capacidad de procesar grandes volúmenes de envíos y adaptarse a diferentes lenguajes de programación. El modelo de Jutge.org ha demostrado ser replicable incluso en entornos con baja conectividad mediante versiones adaptadas offline (Universitat Politècnica de Catalunya, s.f.).

Figura 1. Interfaz de usuario de Jutge.org

Python Tutor constituye otra herramienta relevante en este campo. Desarrollada inicialmente en la Universidad de Stanford, esta plataforma ofrece la visualización paso a paso de la ejecución de programas, permitiendo a los estudiantes comprender el comportamiento dinámico de su código. Python Tutor ha sido adaptado en entornos offline en países como Colombia y Perú, empleando contenedores Docker y almacenamiento local con SQLite. Estas adaptaciones evidencian que es posible mantener la funcionalidad esencial de los sistemas de aprendizaje sin depender de conectividad constante, facilitando la enseñanza de conceptos complejos como la gestión de memoria o las estructuras de datos.

Figura 2. Visualización de ejecución paso a paso en Python Tutor



En América Latina, el proyecto Aprendo Libre en Chile representa un ejemplo relevante en el uso de software libre y hardware económico para llevar experiencias de programación a escuelas rurales y urbanas con infraestructura tecnológica limitada. La implementación de servidores locales basados en Raspberry Pi permitió alojar entornos de desarrollo en Python y sincronizar los datos con la nube cuando había conexión disponible. Los resultados fueron positivos: se incrementó en un 25% la retención de estudiantes en

cursos de programación y se optimizó el tiempo de corrección de los docentes gracias a la automatización parcial del proceso.

En el caso de Guatemala, aunque los proyectos de este tipo aún son incipientes, el Ministerio de Educación ha impulsado pilotos enmarcados en el programa Mineduc Digital (2024), donde se distribuyeron dispositivos de bajo costo con contenidos educativos precargados. Si bien estas iniciativas no se han enfocado de manera exclusiva en la enseñanza de la programación, han demostrado la viabilidad técnica y logística de implementar soluciones descentralizadas en contextos urbanos y rurales con limitaciones de infraestructura.

De estas experiencias se desprenden aprendizajes fundamentales para el desarrollo de plataformas educativas en contextos similares. En primer lugar, resulta técnicamente viable implementar evaluadores automáticos de código en entornos offline mediante arquitecturas basadas en servidores locales. En segundo lugar, el uso de hardware asequible como Raspberry Pi constituye una estrategia efectiva para reducir la brecha digital sin comprometer la funcionalidad del sistema. En tercer lugar, la retroalimentación inmediata mejora significativamente el compromiso y el aprendizaje en programación. Finalmente, la adaptación local y la capacitación docente resultan críticas para el éxito y la sostenibilidad de estas iniciativas en instituciones con recursos limitados.

C. Inteligencia artificial adaptativa

Una vez establecido el mecanismo de evaluación automática mediante análisis de código, surge la necesidad de personalizar la experiencia de aprendizaje según las características individuales de cada estudiante. Para ello, la integración de inteligencia artificial adaptativa permite ajustar dinámicamente la dificultad de los ejercicios y ofrecer trayectorias formativas diferenciadas.

1. Sistemas de aprendizaje personalizado

La personalización del aprendizaje constituye un paradigma educativo que reconoce la diversidad de ritmos, estilos y necesidades formativas de los estudiantes. A diferencia de los enfoques tradicionales que asumen homogeneidad en el grupo, los sistemas personalizados adaptan contenidos, metodologías y evaluaciones según las características individuales de cada alumno. Esta aproximación ha cobrado especial relevancia con el desarrollo de tecnologías digitales capaces de procesar grandes volúmenes de datos de aprendizaje y generar recomendaciones específicas.

La inteligencia artificial adaptativa aplicada a entornos educativos permite personalizar el aprendizaje ajustando automáticamente la dificultad de los ejercicios en función del desempeño individual de cada estudiante. Esta tecnología optimiza el proceso formativo al ofrecer contenidos alineados con las necesidades específicas de cada usuario, mejorando la

eficiencia en la adquisición de conocimientos y fomentando una mayor motivación y compromiso (Universidad Rafael Landívar, 2024).

Los sistemas adaptativos operan mediante ciclos continuos de evaluación, análisis y ajuste. Primero, recopilan información sobre el desempeño del estudiante mediante métricas como tiempo de resolución, tipos de errores cometidos y número de intentos requeridos. Posteriormente, procesan esta información mediante algoritmos que identifican patrones de aprendizaje y deficiencias conceptuales. Finalmente, utilizan estos análisis para recomendar actividades, ajustar niveles de dificultad o proporcionar recursos complementarios.

Eduteka (2022) reporta que los sistemas educativos que ajustan los contenidos según el desempeño del estudiante pueden mejorar la retención de conocimientos en hasta un 30%. Esta mejora se atribuye a varios factores: la reducción de frustración al evitar ejercicios excesivamente difíciles, el mantenimiento del desafío cognitivo al prevenir actividades demasiado simples, y el refuerzo oportuno de conceptos mediante ejercicios dirigidos a áreas de dificultad específicas.

En el contexto de la enseñanza de programación, la personalización resulta particularmente relevante debido a la diversidad de perfiles estudiantiles. Algunos estudiantes presentan facilidad para el pensamiento lógico-matemático, pero dificultades con la sintaxis específica del lenguaje, mientras otros enfrentan el desafío opuesto. Los sistemas adaptativos pueden identificar estos perfiles y ofrecer ejercicios que fortalezcan las áreas débiles sin descuidar el desarrollo integral de competencias.

2. Machine Learning supervisado en contextos educativos

El aprendizaje automático supervisado constituye una rama de la inteligencia artificial que permite a los sistemas aprender patrones a partir de datos etiquetados. En contextos educativos, esta técnica se aplica para predecir resultados de aprendizaje, clasificar niveles de dominio y recomendar trayectorias formativas basadas en el comportamiento histórico de estudiantes previos.

El funcionamiento del aprendizaje supervisado se fundamenta en la disponibilidad de un conjunto de datos de entrenamiento donde cada ejemplo está asociado con una etiqueta o resultado conocido. En plataformas educativas de programación, estos datos pueden incluir ejercicios resueltos previamente, clasificados según variables como dificultad, conceptos involucrados, tipos de errores comunes y tiempo promedio de resolución. Los algoritmos analizan estos datos para identificar relaciones entre las características del ejercicio y el desempeño estudiantil.

Una vez entrenados, los modelos pueden aplicarse para predecir qué ejercicios resultarán apropiados según el nivel de conocimiento de nuevos estudiantes. Esta capacidad predictiva permite construir secuencias de aprendizaje personalizadas que optimizan la

progresión del estudiante, evitando saltos bruscos de dificultad que generen frustración o secuencias demasiado graduales que provoquen aburrimiento.

La aplicación de machine learning en educación requiere consideraciones éticas y técnicas específicas. Los datos de entrenamiento deben ser representativos de la diversidad estudiantil para evitar sesgos que perjudiquen a determinados grupos. Además, los modelos deben ser transparentes y auditables, permitiendo a docentes y estudiantes comprender las razones detrás de las recomendaciones generadas. Estas consideraciones resultan fundamentales para mantener la confianza en el sistema y garantizar su efectividad pedagógica.

3. Algoritmo Random Forest

El algoritmo Random Forest pertenece a la categoría de métodos de ensamble supervisados y se caracteriza por combinar múltiples árboles de decisión para generar predicciones más robustas y precisas. Al entrenar diversos árboles sobre subconjuntos aleatorios de datos y características, esta técnica reduce el sobreajuste que puede afectar a modelos individuales y mejora tanto la estabilidad como la precisión de los resultados (Simplilearn, s.f.).

La denominación "bosque aleatorio" refleja la naturaleza del algoritmo: en lugar de construir un único árbol de decisión que podría especializarse excesivamente en los datos de entrenamiento, se construyen múltiples árboles, cada uno entrenado con una muestra aleatoria del conjunto de datos. Al momento de realizar una predicción, cada árbol en el bosque genera su propia respuesta, y el resultado final se determina mediante votación mayoritaria en problemas de clasificación o promedio en problemas de regresión.

Esta arquitectura de ensamble ofrece ventajas significativas en contextos educativos. La robustez del modelo permite manejar datos ruidosos o incompletos, situación común en plataformas educativas donde los estudiantes pueden cometer errores aleatorios o interrumpir sesiones de trabajo. La capacidad de identificar la importancia de diferentes variables permite comprender qué factores resultan más predictivos del éxito estudiantil, información valiosa para orientar intervenciones pedagógicas.

En plataformas de enseñanza de programación, Random Forest puede entrenarse con variables como tipo de errores cometidos, tiempo invertido en cada ejercicio, número de intentos antes de alcanzar la solución correcta, secuencia de ejercicios previamente completados y nivel de dificultad superado. Con esta información, el algoritmo aprende a clasificar el nivel de dominio de estudiantes y predecir qué ejercicios optimizarán su progreso.

Simplilearn (s.f.) destaca que Random Forest resulta particularmente efectivo para tareas de clasificación y regresión en contextos donde la precisión predictiva es crítica. En plataformas educativas, esta precisión se traduce en recomendaciones más acertadas que

mantienen al estudiante en su zona de desarrollo próximo, concepto pedagógico que describe el espacio óptimo entre lo que el estudiante puede hacer de manera autónoma y lo que requiere apoyo para lograr.

4. Adaptación automática de dificultad

La adaptación automática de dificultad constituye la aplicación práctica de los algoritmos de machine learning en el diseño de trayectorias de aprendizaje personalizadas. Este mecanismo opera mediante la evaluación continua del desempeño estudiantil y el ajuste dinámico de los ejercicios propuestos, buscando mantener un equilibrio óptimo entre desafío y capacidad.

Los sistemas adaptativos estructuran el proceso de ajuste en tres etapas fundamentales. Primero, recopilan datos sobre cada intento de resolución, registrando no solo el resultado final sino también métricas intermedias como tiempo invertido, errores cometidos y estrategias utilizadas. Segundo, estos datos alimentan modelos de machine learning que clasifican el nivel actual del estudiante en relación con diferentes competencias. Tercero, con base en esta clasificación, el sistema selecciona ejercicios que mejor se ajustan al perfil identificado.

La efectividad de la adaptación automática depende crucialmente de la calidad del banco de ejercicios y de la precisión en su clasificación por dificultad. Los ejercicios deben estar cuidadosamente calibrados, considerando múltiples dimensiones de complejidad: longitud del código requerido, número de conceptos que deben aplicarse simultáneamente, nivel de abstracción necesario y similitud con problemas previamente resueltos. Esta clasificación multidimensional permite a los algoritmos seleccionar ejercicios que desafían al estudiante en áreas específicas sin abrumarlo con múltiples dificultades simultáneas.

Los sistemas adaptativos también deben incorporar mecanismos de recuperación para estudiantes que enfrentan dificultades persistentes. Cuando un algoritmo detecta que un estudiante no logra superar ejercicios del nivel actual después de múltiples intentos, puede ajustar temporalmente a ejercicios de refuerzo que consoliden conceptos fundamentales antes de intentar nuevamente el avance. Esta capacidad de ajuste bidireccional resulta fundamental para prevenir la desmotivación y el abandono.

La investigación sobre sistemas adaptativos ha demostrado su efectividad en diversos contextos educativos. Estudios reportan que los estudiantes que trabajan con plataformas adaptativas muestran mayor persistencia, menor frustración y mejores resultados de aprendizaje en comparación con aquellos que siguen secuencias de ejercicios lineales y predefinidas. Estos beneficios se atribuyen a que la adaptación mantiene al estudiante en un estado de flujo, concepto psicológico que describe la experiencia óptima de inmersión en una tarea desafiante pero alcanzable. Estas cualidades hacen que algoritmos como Random Forest sean opciones eficaces para implementar sistemas de recomendación en entornos educativos donde la retroalimentación individualizada resulta crucial para el progreso académico.

D. Arquitectura Tecnologías Offline-First

Los contextos educativos con limitaciones de conectividad requieren soluciones tecnológicas que garanticen la continuidad del aprendizaje sin dependencia de internet.

1. Aplicaciones Web Progresivas (PWA)

Las Aplicaciones Web Progresivas constituyen una evolución de las aplicaciones web tradicionales, combinando las ventajas de sitios web con características propias de aplicaciones nativas. Las PWA se caracterizan por su capacidad de funcionar sin conexión a internet, instalarse en dispositivos sin requerir tiendas de aplicaciones y ofrecer experiencias de usuario fluidas incluso en condiciones de conectividad limitada.

La implementación de plataformas educativas con enfoque Offline-First busca garantizar la continuidad del aprendizaje en contextos con conectividad limitada, permitiendo a los estudiantes acceder a contenidos y realizar actividades sin necesidad de una conexión constante a internet. Este enfoque no solo mejora el rendimiento de las aplicaciones, sino que también contribuye a una educación digital más equitativa y resiliente. Al priorizar el almacenamiento y procesamiento local de datos, las plataformas Offline-First aseguran una experiencia fluida para el usuario y sincronizan automáticamente la información con la nube cuando la conexión esté disponible (DashDevs, 2025; Progressive Web Apps, s.f.).

Las PWA emplean tecnologías como Service Workers y almacenamiento en caché para mantener funcionalidad completa sin internet. Los Service Workers actúan como intermediarios entre la aplicación y la red, interceptando solicitudes y sirviendo contenido almacenado localmente cuando no hay conectividad. Esta arquitectura permite que los estudiantes continúen trabajando en ejercicios, reciban evaluación de código y visualicen su progreso, incluso cuando la conexión a internet es inexistente o inestable.

En el ámbito educativo, las PWA ofrecen ventajas adicionales frente a aplicaciones nativas tradicionales. No requieren procesos de instalación complejos ni actualizaciones manuales, reduciendo barreras técnicas para estudiantes y docentes. Además, funcionan en múltiples plataformas y dispositivos desde un único código base, facilitando su despliegue en instituciones con equipos heterogéneos.

Cuadro 1. Comparación entre aplicación web tradicional y PWA

Criterio	Aplicación web tradicional	Progressive Web App (PWA)
Acceso/Instalación	Se accede a través de una URL en un navegador. No es instalable en el sistema operativo.	Instalable (Installable): el usuario puede añadir la aplicación directamente a la pantalla de inicio o al escritorio sin usar tiendas de aplicaciones.

Criterio	Aplicación web tradicional	Progressive Web App (PWA)
Fiabilidad y conectividad	Requiere una conexión a Internet constante y estable para cargar. Falla en entornos de baja conectividad.	Fiable (Reliable): carga instantáneamente y funciona sin conexión (offline), o con redes débiles, gracias al Service Worker (que almacena recursos en caché).
Experiencia de Usuario	Se ejecuta dentro de la interfaz completa del navegador (con barras de dirección y navegación).	Similar a una aplicación nativa: se ejecuta en una ventana separada, con una interfaz de usuario minimalista del sistema operativo, proporcionando una experiencia envolvente.
Actualizaciones	El usuario siempre obtiene la última versión en cada visita a la URL (carga completa).	Las actualizaciones se manejan automáticamente y de manera discreta en segundo plano, sin necesidad de intervención del usuario.
Seguridad	Se recomienda HTTPS.	Requiere obligatoriamente HTTPS para habilitar las funcionalidades clave, como el Service Worker, garantizando una conexión segura.
Acceso a funcionalidades	Acceso limitado a las API de hardware (solo funciones básicas del navegador).	Puede acceder a funcionalidades del dispositivo (por ejemplo, notificaciones Push y geolocalización) para mejorar la interacción.
Descubrimiento	Encontrada a través de motores de búsqueda.	Encontrada a través de motores de búsqueda, con la ventaja de ser "instalable" y en ocasiones listada en tiendas como Microsoft Store o Google Play.

2. Servidores educativos con Raspberry Pi

El dispositivo Raspberry Pi se ha consolidado como una herramienta eficaz en entornos educativos con limitaciones tecnológicas, gracias a su bajo costo, tamaño compacto y reducido consumo energético. Su capacidad para operar como servidor autónomo permite alojar contenidos educativos, registrar interacciones estudiantiles y sincronizar información cuando se dispone de conectividad. Esta arquitectura favorece la continuidad del aprendizaje

sin interrupciones y refuerza la privacidad y seguridad en el manejo de datos académicos (VIU, 2023; Vofox Solutions, 2025).

La Raspberry Pi, diseñada originalmente para promover el aprendizaje de programación y electrónica, es compatible con múltiples lenguajes como Python, C++ y JavaScript, lo que amplía sus posibilidades de aplicación en el ámbito educativo. Su bajo consumo energético y diseño compacto la convierten en una opción viable para instituciones con recursos limitados, facilitando la creación de soluciones tecnológicas accesibles y sostenibles (Raspberry Pi Foundation, s.f.).

En contextos educativos, Raspberry Pi puede configurarse como servidor local que hospeda aplicaciones web, bases de datos y contenidos multimedia. Esta configuración permite crear redes de área local donde múltiples dispositivos acceden simultáneamente a recursos educativos sin requerir conexión externa. El dispositivo maneja eficientemente cargas de trabajo moderadas, soportando decenas de conexiones simultáneas sin degradación significativa del rendimiento.

La incorporación de tecnologías complementarias, como el almacenamiento en caché, mejora el rendimiento de plataformas educativas basadas en Raspberry Pi, al reducir los tiempos de acceso y optimizar el uso de recursos. Estas soluciones representan una respuesta técnica eficiente ante la brecha de infraestructura digital, promoviendo un entorno más inclusivo y equitativo para el desarrollo de competencias digitales (Kinsta, s.f.).

Figura 3. Raspberry Pi 5 utilizada como servidor educativo

3. Bases de datos embebidas (SQLite)

SQLite es un sistema de gestión de bases de datos relacional, embebido y serverless, que se ha consolidado como una de las herramientas más utilizadas en aplicaciones que operan en entornos con recursos limitados (Pass4sure, 2024). Su diseño bajo el principio zero-configuration elimina la necesidad de instalar o administrar un servidor central, ya que toda la información se almacena en un único archivo portable. Esta característica reduce significativamente la complejidad operativa y facilita su implementación en aplicaciones educativas, proyectos de bajo consumo de recursos y sistemas descentralizados (SQLite, 2024).

Una de sus principales fortalezas es la portabilidad. Al mantener todos los datos dentro de un único archivo, resulta sencillo generar copias de seguridad, transferir bases de datos entre distintos dispositivos o integrarla en aplicaciones móviles y sistemas embebidos

(SQLite, 2024). A ello se suma su eficiencia en el acceso a los datos, pues al prescindir de una arquitectura cliente-servidor se reduce la latencia y se obtienen lecturas y escrituras rápidas en almacenamiento local (EpicWeb, 2023).

En términos de confiabilidad, SQLite cumple de manera estricta con el modelo transaccional ACID, lo cual garantiza la integridad de la información incluso en situaciones críticas como fallos del sistema o interrupciones de energía (Dev.to, 2023). Además, su modo Write-Ahead Logging (WAL) permite mejorar el rendimiento en escenarios con operaciones concurrentes, reforzando así su estabilidad y escalabilidad en diferentes contextos (Microsoft, 2023).

En el ámbito pedagógico, SQLite se distingue por su simplicidad. Su facilidad de uso favorece la enseñanza de SQL, la creación de prototipos y el aprendizaje autónomo, sin necesidad de desplegar infraestructuras complejas (SQLite, 2024; Medium, 2023). De igual forma, su integración en dispositivos de borde como teléfonos móviles, navegadores y microservidores basados en Raspberry Pi demuestra su versatilidad y pertinencia en contextos de conectividad limitada (Moldstud, 2024).

En consecuencia, SQLite se consolida como una solución tecnológica estratégica para el desarrollo de plataformas educativas offline-first. Sus ventajas en portabilidad, confiabilidad y simplicidad no solo aseguran la continuidad del aprendizaje en entornos con restricciones de conectividad, sino que también promueven la equidad digital y fortalecen la resiliencia tecnológica en el ámbito educativo (Jucaripo Blog, 2025; Explo, 2025).

4. Frameworks web ligeros (Flask)

Flask constituye un framework web minimalista para Python, caracterizado por su simplicidad, flexibilidad y eficiencia en el desarrollo de aplicaciones web. A diferencia de frameworks más robustos que imponen estructuras rígidas, Flask adopta una filosofía de microframework, proporcionando únicamente las funcionalidades esenciales y permitiendo a los desarrolladores agregar componentes según las necesidades específicas del proyecto.

La arquitectura ligera de Flask resulta particularmente apropiada para entornos educativos con recursos limitados. Su bajo consumo de memoria y procesamiento permite ejecutar aplicaciones web en dispositivos modestos como Raspberry Pi sin comprometer el rendimiento. Esta eficiencia se combina con la capacidad de manejar múltiples conexiones simultáneas, característica fundamental en contextos donde varios estudiantes acceden concurrentemente a una plataforma educativa.

Flask facilita el desarrollo de APIs RESTful, patrón arquitectónico ampliamente utilizado para la comunicación entre aplicaciones web. Esta capacidad permite estructurar plataformas educativas en componentes modulares: un frontend que presenta la interfaz al usuario y un backend que procesa datos, ejecuta lógica de negocio y gestiona el

almacenamiento. La separación de responsabilidades mejora la mantenibilidad del sistema y facilita actualizaciones futuras.

La integración natural de Flask con otras tecnologías del ecosistema Python representa una ventaja adicional. Bibliotecas especializadas en análisis de código, machine learning o procesamiento de datos pueden incorporarse fácilmente, expandiendo las capacidades de la plataforma sin complejidad arquitectónica excesiva. Esta interoperabilidad resulta valiosa en plataformas educativas que requieren funcionalidades diversas como evaluación automática de código, personalización mediante inteligencia artificial y análisis de progreso estudiantil (Analytics Vidhya, 2025).

E. Fundamentos pedagógicos

La enseñanza de la programación en contextos educativos exige enfoques pedagógicos que permitan al estudiante comprender de manera progresiva conceptos abstractos y lógicos, mediante procesos de construcción activa del conocimiento. En este sentido, teorías como el constructivismo y el aprendizaje significativo destacan la relevancia de vincular los nuevos aprendizajes con los saberes previos del estudiante, lo que favorece la asimilación y la transferencia del conocimiento a situaciones prácticas (Ausubel, 1963).

El enfoque del learning by doing (aprender haciendo) se presenta como una metodología idónea en el ámbito de la educación tecnológica, ya que promueve el aprendizaje activo a través de la experimentación y la resolución de problemas en escenarios concretos. Esta aproximación no solo refuerza la comprensión técnica, sino que también impulsa el desarrollo de habilidades clave como la motivación intrínseca, el pensamiento crítico, la autoconfianza y la innovación, elementos fundamentales en la formación de futuros profesionales (Santander Open Academy, 2025).

Asimismo, múltiples investigaciones han señalado que los entornos de aprendizaje que integran mecanismos de retroalimentación inmediata ---como ocurre con actividades interactivas o simuladores--- incrementan de manera significativa la motivación, el compromiso y el rendimiento académico de los estudiantes en el aprendizaje de la programación estructurada (Redalyc -- H5P, 2023). Un ejemplo relevante es el estudio realizado en la Universidad del Istmo, donde los estudiantes que trabajaron con actividades interactivas dotadas de retroalimentación instantánea obtuvieron resultados altamente positivos en indicadores de aceptación tecnológica (TAM), motivación (IMMS) y satisfacción general (SUS) (Redalyc -- H5P, 2023).

La enseñanza programada de Skinner aporta un modelo metodológico centrado en la presentación secuenciada del contenido en unidades didácticas pequeñas, acompañadas de retroalimentación inmediata. Esta propuesta ofrece la posibilidad de un aprendizaje personalizado, ajustado al ritmo del estudiante, y encuentra hoy un soporte relevante en las tecnologías educativas automatizadas.

Finalmente, estudios recientes en el área de la educación en software enfatizan los beneficios de integrar retroalimentación automatizada con acompañamiento humano. Esta combinación permite no solo mejorar el desempeño académico, sino también potenciar la participación del alumno y fortalecer la interacción pedagógica en entornos digitales, consolidando así un aprendizaje más significativo y sostenible (Grawemeyer et al., 2022).

F. Ética, privacidad y protección de datos

La integración de inteligencia artificial en entornos educativos demanda un marco ético y legal sólido que asegure la protección de los estudiantes y garantice la confianza en el uso de estas tecnologías.

1. Cumplimiento de leyes de protección de datos

El funcionamiento de plataformas educativas con inteligencia artificial requiere ajustarse estrictamente a normativas como el Reglamento General de Protección de Datos (GDPR) en Europa y sus equivalentes en otros países, como Guatemala. Estas regulaciones establecen principios clave: la minimización en la recolección de datos, la restricción de su tratamiento a fines claramente definidos, la transparencia en el proceso y la obtención de un consentimiento informado y verificable. El cumplimiento de estos marcos normativos constituye la base para proteger la integridad y privacidad de la información estudiantil (Digimasters, 2024; ResearchGate---Goutziamani, 2024).

2. Anonimización de datos

La anonimización se convierte en una práctica indispensable para salvaguardar la identidad de los estudiantes cuando los datos se emplean con fines educativos o de investigación. Esta estrategia reduce considerablemente el riesgo de violaciones a la privacidad, a la vez que posibilita un análisis riguroso y útil de la información sin comprometer la confidencialidad individual. De este modo, se asegura un equilibrio entre el aprovechamiento de los datos y el respeto a los derechos de los alumnos (eLearning Industry, 2025).

3. Transparencia algorítmica

La transparencia en los algoritmos utilizados en sistemas educativos basados en inteligencia artificial constituye un requisito esencial. El marco normativo europeo, en particular el AI Act, introduce directrices específicas para garantizar que los sistemas sean comprensibles y auditables. En este sentido, iniciativas como los talleres de la Comisión Europea han promovido herramientas innovadoras, tales como puntuaciones de explicabilidad y lineamientos prácticos para asegurar que la inteligencia artificial permanezca centrada en las personas en los contextos educativos (EC Workshop, 2024).

4. Consentimiento informado

El consentimiento explícito y voluntario por parte de los estudiantes o de sus tutores es un pilar ético fundamental. Este consentimiento debe otorgarse de manera consciente y

renovable, y su revocación ha de ser tan sencilla como su otorgamiento. Solo de esta manera se garantiza que la participación en el uso de datos sea un acto libre y respetuoso de los derechos individuales (Digimasters, 2024).

5. Equidad y mitigación del sesgo algorítmico

Un riesgo central de la inteligencia artificial en la educación radica en la posibilidad de reproducir sesgos cuando se entrenan modelos con datos poco representativos. Este fenómeno puede generar desigualdades que afectan a determinados grupos de estudiantes. Para prevenirlo, se requiere trabajar con bases de datos diversas, aplicar auditorías periódicas a los algoritmos y garantizar una supervisión humana constante. Estudios recientes subrayan la relevancia de abordar explícitamente la equidad, los sesgos y la ética como parte integral de las aplicaciones educativas con inteligencia artificial (LinkedIn, 2025; FairAIED, 2024).

6. Gobernanza e implementación de políticas

Finalmente, muchas instituciones educativas aún carecen de lineamientos específicos que regulen el uso ético y legal de herramientas de inteligencia artificial. Aspectos como la privacidad de datos, la transparencia algorítmica y la rendición de cuentas requieren un desarrollo normativo claro y dinámico. Se vuelve imperativo, por tanto, establecer políticas institucionales y marcos gubernamentales flexibles que acompañen el avance tecnológico y orienten la gestión responsable de la inteligencia artificial en educación (Ghimire y Edwards, 2024).

V. IDENTIFICACIÓN DEL PROBLEMA

A. El problema

La enseñanza de programación en el nivel medio enfrenta diversos desafíos que limitan el desarrollo de competencias digitales en los estudiantes. En el Instituto Tecnológico Experimental de Sololá se identifican problemáticas específicas que afectan tanto a docentes como a alumnos en el proceso de enseñanza-aprendizaje del Bachillerato en Computación.

1. Ausencia de retroalimentación inmediata

La evaluación manual de código genera retrasos significativos en la corrección de ejercicios de programación. Durante el diagnóstico realizado en el ciclo escolar 2024, se observó que el tiempo promedio entre la entrega de un ejercicio y la recepción de retroalimentación fue de 4 a 5 días. Esta demora impide que los estudiantes identifiquen errores con rapidez, limitando el aprendizaje autónomo y la corrección oportuna de conceptos erróneos. Los docentes invierten aproximadamente 3 horas semanales en la corrección manual de ejercicios, tiempo que podría destinarse a actividades pedagógicas de mayor complejidad como tutorías personalizadas o diseño de nuevos materiales didácticos.

2. Conectividad intermitente

El acceso a internet en la institución presenta interrupciones frecuentes que restringen el uso de recursos educativos digitales. Esta inestabilidad técnica impide el uso efectivo de plataformas educativas en línea, compiladores remotos, repositorios de código y sistemas de gestión de aprendizaje. La dependencia de conectividad externa para acceder a recursos limita significativamente las estrategias pedagógicas disponibles, generando desventajas frente a instituciones con infraestructura tecnológica más robusta.

3. Ausencia de personalización

Las actividades de programación se diseñan de manera homogénea, sin considerar los distintos ritmos de aprendizaje, conocimientos previos ni dificultades específicas de cada estudiante. Esta uniformidad en la enseñanza no permite atender adecuadamente la diversidad del aula, donde algunos estudiantes requieren ejercicios de refuerzo mientras otros necesitan desafíos más complejos. La falta de adaptación pedagógica incrementa la desmotivación, favorece el rezago académico y disminuye la calidad en la formación en programación.

4. Recursos tecnológicos limitados

El laboratorio de computación cuenta con equipos de especificaciones modestas y capacidad limitada de actualización. La ausencia de servidores locales obliga a depender exclusivamente de servicios externos, lo que agrava los problemas de conectividad. Estas limitaciones técnicas restringen la implementación de entornos de desarrollo modernos, herramientas de análisis de código y plataformas interactivas que facilitarían el aprendizaje.

5. Síntesis del problema

En conjunto, estas condiciones evidencian una problemática estructural: la enseñanza de programación, competencia fundamental en el siglo XXI, se ve obstaculizada por la ausencia de retroalimentación oportuna, precariedad en infraestructura tecnológica y carencia de metodologías personalizadas apoyadas en herramientas digitales. Como resultado, los estudiantes enfrentan un aprendizaje fragmentado, bajos niveles de retención de conocimientos y una reducción de sus oportunidades de empleabilidad futura en el ámbito tecnológico. Esta situación justifica el desarrollo de soluciones tecnológicas adaptadas al contexto específico del Instituto Tecnológico Experimental de Sololá, que permitan superar estas limitaciones mediante el aprovechamiento de recursos locales y metodologías innovadoras.

B. Población objetivo

El presente proyecto está dirigido principalmente a dos grupos dentro del Instituto Tecnológico Experimental de Sololá:

1. Estudiantes del Bachillerato en Computación

Los beneficiarios directos son los estudiantes de primero y segundo año del Bachillerato en Computación, con edades comprendidas entre 15 y 18 años. Este grupo enfrenta los desafíos descritos en el aprendizaje de programación y requiere herramientas que faciliten la práctica continua, la retroalimentación inmediata y el desarrollo progresivo de competencias en lenguajes de programación como Python, JavaScript y fundamentos de estructuras de datos.

2. Docentes del área de programación

Los docentes que imparten cursos de programación y desarrollo de software constituyen el segundo grupo objetivo. Estos profesionales requieren herramientas que optimicen el proceso de evaluación, permitan el seguimiento individualizado del progreso estudiantil y faciliten la identificación de dificultades comunes en el aprendizaje. La plataforma propuesta busca reducir la carga administrativa de corrección manual y proporcionar información analítica sobre el desempeño de sus estudiantes.

3. Características del contexto

La población objetivo se caracteriza por:

Acceso limitado a conectividad estable durante las horas de clase
Disponibilidad de dispositivos con capacidades computacionales básicas
Conocimientos previos variables en programación
Motivación inicial hacia las tecnologías de la información
Necesidad de desarrollo de competencias para el mercado laboral regional

4. Proyección de impacto

Aunque el proyecto se implementa inicialmente en el Instituto Tecnológico Experimental de Sololá, el modelo propuesto es escalable y replicable en otras instituciones de nivel medio con características similares, particularmente aquellas ubicadas en áreas con limitaciones de conectividad y recursos tecnológicos.

VI. SOLUCIÓN PROPUESTA

Como respuesta a la problemática identificada en el proceso de enseñanza y aprendizaje de programación en el Instituto Tecnológico Experimental de Sololá, se propuso el desarrollo de un Sistema de Evaluación Automática de Código Python con Retroalimentación Inteligente y Gamificación.

1. Concepto de la solución

El sistema propuesto fue una aplicación web que permitió a los estudiantes:

- Resolver ejercicios de programación directamente en un navegador
- Recibir evaluación y retroalimentación automática en menos de 10 segundos
- Visualizar su progreso mediante un dashboard personalizado
- Desbloquear badges según su desempeño
- Avanzar automáticamente entre tres niveles: Principiante, Intermedio y Avanzado

Para los docentes, el sistema proporcionó:

- Panel administrativo con estadísticas del grupo
- Identificación temprana de estudiantes con dificultades
- Reducción del 83% en tiempo dedicado a corrección manual
- Datos para toma de decisiones pedagógicas informadas

2. Características principales

Evaluación automática inmediata: el sistema evaluó código Python en menos de 10 segundos, validando sintaxis, ejecución y correctitud lógica mediante casos de prueba predefinidos.

Retroalimentación personalizada: cada evaluación generó mensajes específicos identificando el tipo de error (sintaxis, lógica, timeout) y proporcionando sugerencias concretas de corrección.

Gamificación integrada: sistema de 12 badges y 3 niveles de progresión que aumentó la motivación estudiantil en un 87.5% según la prueba piloto.

Seguridad robusta: ejecución de código en entorno aislado (sandbox) con restricciones de tiempo (5 segundos), memoria (50MB) y módulos peligrosos bloqueados.

Accesibilidad: funcionamiento completo en red local sin requerir conexión a internet, hospedado en hardware económico (Raspberry Pi 5).

A. Arquitectura propuesta

La plataforma se diseñó bajo una arquitectura cliente-servidor de tres capas, optimizada que se implementó en red local (LAN) con mínima dependencia de servicios externos.

1. Capa de presentación (Frontend)

El frontend constituyó la interfaz de usuario accesible desde navegadores web en los equipos del laboratorio de computación. Se desarrolló utilizando tecnologías web modernas que permitieron una experiencia interactiva y responsiva. Los usuarios (estudiantes y docentes) interactuaron con la plataforma a través de esta capa, que consumió servicios proporcionados por el backend mediante peticiones HTTP.

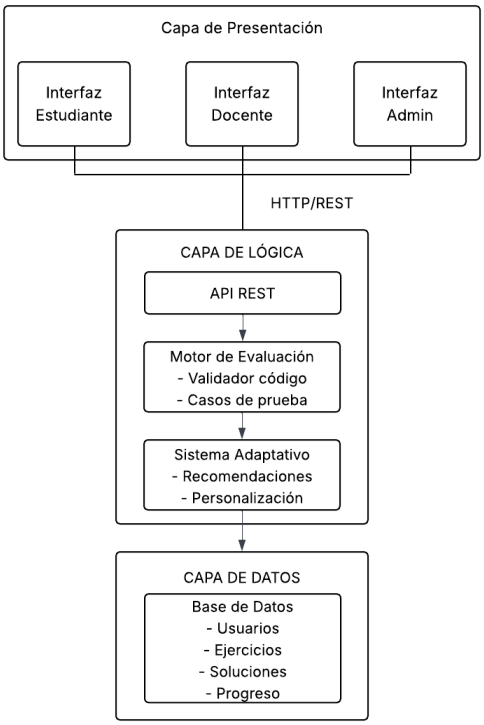
2. Capa de lógica de negocio (Backend)

El backend se ejecutó en un servidor local dentro de la institución, procesó las solicitudes del frontend y coordinó las operaciones del sistema. Esta capa gestionó la autenticación de usuarios, la validación de código, la generación de ejercicios personalizados, el cálculo de métricas de desempeño y la administración de contenidos educativos. El backend expuso una API REST que facilitó la comunicación estructurada con el frontend.

3. Capa de datos

La capa de persistencia almacenaba información de usuarios, ejercicios, soluciones de estudiantes, historiales de progreso y contenidos educativos. Se implementaba mediante un sistema de gestión de bases de datos relacional que garantizaba integridad, consistencia y disponibilidad de la información. Esta capa residía también en el servidor local, lo que aseguraba acceso rápido y sin dependencia de servicios en la nube.

Figura 4. Diagrama de arquitectura



4. Componentes principales

La plataforma se estructuraba en módulos funcionales que operaban de manera coordinada:

Módulo de autenticación y autorización

Gestionaba el acceso al sistema mediante credenciales únicas para cada usuario. Implementaba roles diferenciados (estudiante, docente, administrador) con permisos específicos que determinaban las funcionalidades disponibles para cada perfil.

Módulo de gestión de ejercicios

Permitía a los docentes crear, modificar y organizar ejercicios de programación con diferentes niveles de dificultad. Cada ejercicio incluía descripción del problema, casos de prueba, criterios de evaluación y recursos de apoyo. El sistema categorizaba ejercicios por temas (estructuras de control, funciones, arreglos, etc.) lo que facilitaba la navegación y selección.

Motor de evaluación automática

Constituía el componente central de la solución. Recibía código fuente enviado por estudiantes, lo ejecutaba en un entorno controlado (sandbox), comparaba resultados con casos de prueba predefinidos y generaba retroalimentación detallada. El motor detectaba errores de sintaxis, lógica y rendimiento, y proporcionaba mensajes descriptivos que orientaban al estudiante hacia la solución correcta.

Sistema de recomendaciones adaptativas

Analizaba el historial de desempeño de cada estudiante e identificaba fortalezas y áreas de mejora. Con base en esta información, recomendaba ejercicios específicos que correspondieran al nivel actual del estudiante, lo que promovía el aprendizaje progresivo y personalizado. El algoritmo consideraba factores como tasa de éxito, tiempo de resolución y número de intentos.

Panel de visualización y análisis

Proporcionaba a docentes información consolidada sobre el progreso individual y grupal. Presentaba métricas como ejercicios completados, tasa de éxito, tiempo promedio de resolución, temas con mayor dificultad y tendencias de aprendizaje. Estas estadísticas facilitaban la toma de decisiones pedagógicas basadas en evidencia.

Módulo de comunicación

Facilitaba la interacción asíncrona entre estudiantes y docentes mediante sistema de mensajes y comentarios en ejercicios. Permitía solicitar ayuda, compartir soluciones alternativas y fomentar el aprendizaje colaborativo.

B. Base de datos propuesta

El diseño de la base de datos se estructuraba en tablas relacionales que garantizaban integridad referencial y optimización en consultas frecuentes.

Entidades principales

Usuarios: almacenaba información de estudiantes, docentes y administradores (id, nombre, correo, contraseña_hash, rol, fecha_registro).

Ejercicios: contenía problemas de programación (id, título, descripción, nivel_dificultad, tema, autor_id, fecha_creación, casos_prueba).

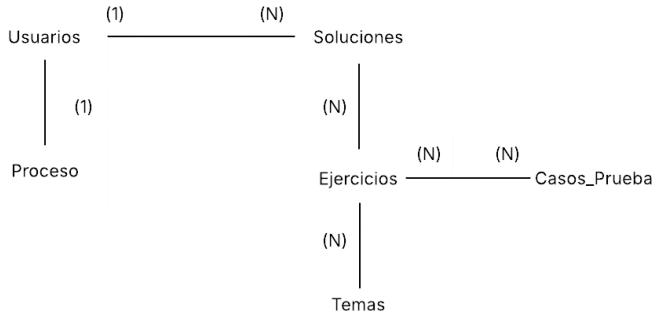
Soluciones: registraba intentos de estudiantes (id, ejercicio_id, estudiante_id, código_fuente, resultado, puntuación, tiempo_ejecución, fecha_envío).

Progreso: realizaba seguimiento de avance estudiantil (id, estudiante_id, ejercicios_completados, tasa_éxito, tiempo_total, última_actividad).

Temas: categorizaba contenidos (id, nombre, descripción, ejercicios_asociados).

Casos_prueba: especificaba entradas y salidas esperadas para validación (id, ejercicio_id, entrada, salida_esperada, puntos).

Figura 5. Modelo entidad-relación simplificada



Consideraciones de diseño

Se implementaban índices en claves foráneas para optimizar joins.

Se aplicaba normalización hasta tercera forma normal (3FN).

Se incluían campos timestamp para auditoría.

Se establecían restricciones de integridad referencial.

Se configuraban respaldos automáticos diarios.

C. Algoritmo de evaluación automática

El proceso de evaluación de código seguía un flujo estructurado que garantizaba seguridad, precisión y retroalimentación útil.

1. Flujo de evaluación

Recepción: el sistema recibía código fuente e identificador del ejercicio.

Validación sintáctica: verificaba que el código fuera sintácticamente correcto.

Análisis estático: detectaba problemas potenciales sin ejecutar el código.

Ejecución controlada: ejecutaba el código en sandbox con límites de tiempo y memoria.

Comparación: contrastaba salidas con casos de prueba esperados.

Calificación: asignaba puntuación según criterios predefinidos.

Retroalimentación: generaba mensajes descriptivos sobre resultados.

2. Pseudocodigo del algoritmo

```
FUNCIÓN evaluar_código(código, ejercicio_id):
  casos_prueba = obtener_casos_prueba(ejercicio_id)
  puntuación_total = 0
  resultados = []

  SI NO es_sintácticamente_válido(código):
    RETORNAR error_sintaxis(código)
  FIN SI

  PARA CADA caso EN casos_prueba:
    INTENTAR:
      salida = ejecutar_en_sandbox(código, caso.entrada,
                                   límite_tiempo=5s,
                                   límite_memoria=256MB)

      SI salida == caso.salida_esperada:
```

```
            puntuación_total += caso.puntos
            resultados.agregar(éxito=True, caso=caso.id)
        SINO:
            resultados.agregar(éxito=False,
                               esperado=caso.salida_esperada,
                               obtenido=salida)
        FIN SI

    CAPTURAR TimeoutError:
        resultados.agregar(error="Tiempo límite excedido")
    CAPTURAR MemoryError:
        resultados.agregar(error="Memoria insuficiente")
    CAPTURAR RuntimeError:
        resultados.agregar(error="Error de ejecución")
    FIN INTENTAR
  FIN PARA
  retroalimentación = generar_retroalimentación(resultados)
  guardar_solución(código, ejercicio_id, puntuación_total, resultados)

  RETORNAR {puntuación: puntuación_total,
            detalles: resultados,
            mensaje: retroalimentación}
FIN FUNCIÓN
```

3. Mecanismo de seguridad

Las medidas para prevenir ejecución de código malicioso incluían:

Ejecución en contenedores aislados (sandboxing)

Restricción de operaciones de sistema (I/O, red, procesos)

Límites estrictos de tiempo de ejecución

Límites de memoria asignada

Validación de imports permitidos

D. Sistema de recomendación adaptativas

El algoritmo de personalización analizaba patrones de aprendizaje para sugerir ejercicios apropiados al nivel de cada estudiante.

1. Factores de análisis

El algoritmo consideraba el historial de desempeño (porcentaje de ejercicios completados exitosamente), el tiempo de resolución en comparación con los promedios del grupo, el número de intentos realizados antes de lograr la solución correcta, los temas que el estudiante había dominado con alta tasa de éxito, y los temas deficientes con bajos resultados o sin intentos.

2. Algoritmo de recomendación

```
FUNCIÓN recomendar_ejercicios(estudiante_id, cantidad=5):
  perfil = analizar_perfil(estudiante_id)
  ejercicios_disponibles = obtener_no_completados(estudiante_id)

  // Clasificar estudiante según desempeño
  SI perfil.tasa_éxito > 0.8:
    nivel_sugerido = "avanzado"
  SINO SI perfil.tasa_éxito > 0.5:
    nivel_sugerido = "intermedio"
  SINO:
    nivel_sugerido = "básico"
  FIN SI

  // Identificar tema prioritario
  tema_débil = obtener_tema_menor_puntuación(estudiante_id)
```

```
    // Filtrar ejercicios
    candidatos = filtrar(ejercicios_disponibles,
                         nivel=nivel_sugerido,
                         tema=tema_débil)

    // Ordenar por relevancia
    recomendados = ordenar(candidatos,
                           criterio=[dificultad_progresiva,
                                     popularidad,
                                     prerequisitos_cumplidos])

    RETORNAR recomendados[0:cantidad]
FIN FUNCIÓN
```

3. Adaptación progresiva

El sistema ajustaba dinámicamente el nivel de dificultad:

Éxito consistente (>80%): incrementaba complejidad gradualmente

Dificultad moderada (50-80%): mantenía nivel actual

Dificultad alta (<50%): sugería ejercicios de refuerzo del tema

E. Flujo de uso del sistema

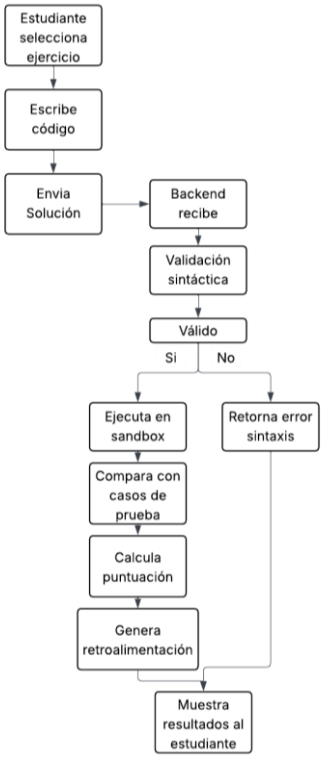
1. Flujo del estudiante

- Acceso: ingresaba credenciales en página de login
- Dashboard: visualizaba ejercicios recomendados y progreso general
- Selección: elegía ejercicio de lista o recomendaciones
- Lectura: estudiaba descripción del problema y ejemplos
- Codificación: escribía solución en editor integrado
- Prueba local: ejecutaba código con casos de ejemplo (opcional)
- Envío: enviaba solución para evaluación oficial
- Retroalimentación: recibía resultados inmediatos con detalles
- Iteración: modificaba código según retroalimentación (si es necesario)
- Avance: procedía al siguiente ejercicio recomendado

2. Flujo del docente

- Acceso: ingresaba al panel administrativo
- Organización: clasificaba por tema y dificultad
- Publicación: hacia disponible el ejercicio para estudiantes
- Monitoreo: consultaba panel de estadísticas grupales
- Análisis: identificaba temas con mayor dificultad
- Intervención: proporcionaba recursos adicionales o tutorías

Figura 6. Diagrama de flujo-Evaluación de ejercicio



VII. DESARROLLO DE LA PROPUESTA

A. Descripción general del sistema

El Sistema de Evaluación Automática de Código Python con Retroalimentación Inteligente es una aplicación web diseñada para facilitar la enseñanza y evaluación de programación en el curso de Introducción a la Programación del Instituto Tecnológico Experimental de Sololá.

El sistema permitió a los estudiantes resolver ejercicios de programación en Python directamente en un navegador web, recibir retroalimentación automática e inmediata sobre su código, visualizar su progreso mediante un dashboard personalizado, desbloquear badges según su desempeño, y avanzar de nivel automáticamente según su puntuación acumulada.

Para los docentes, el sistema ofreció un panel administrativo con estadísticas detalladas de todos los estudiantes, información sobre ejercicios más difíciles y errores comunes, reducción significativa del tiempo dedicado a la corrección manual, e identificación temprana de estudiantes con dificultades.

La arquitectura del sistema estuvo dividida en tres capas principales: Capa de presentación (Frontend) accesible desde cualquier navegador, Capa de lógica de negocio (Backend) con servidor Flask que gestionó las evaluaciones, y Capa de datos con base de datos SQLite que almacenó usuarios, ejercicios, evaluaciones y progreso.

El sistema funcionó completamente en red local sin necesidad de conexión a internet, hospedado en una Raspberry Pi 5 con 4GB de RAM, lo que garantizó accesibilidad y bajo costo de operación para la institución educativa.

B. Modelo de desarrollo utilizado

Para el desarrollo del sistema se adoptó una adaptación del modelo en cascada, propuesto originalmente por Winston W. Royce. Este modelo secuencial dividió el desarrollo en etapas bien definidas que debían completarse antes de avanzar a la siguiente.

Fases del modelo en cascada adaptado:

1. Análisis de requerimientos

Esta etapa consistió en la identificación de necesidades del docente y estudiantes, la definición de funcionalidades críticas del sistema, y el establecimiento de métricas de éxito.

2. Diseño del sistema

Esta fase incluyó el diseño de la arquitectura de tres capas, el modelado de la base de datos relacional, el diseño de interfaces de usuario, y la creación de diagramas de casos de uso.

3. Implementación

Durante esta fase se desarrolló el backend con Python/Flask, se desarrolló el frontend con HTML/CSS/JavaScript, se implementó el motor de evaluación de código, y se desarrolló el sistema de gamificación.

4. Pruebas y verificación

En esta etapa se realizaron pruebas unitarias de componentes individuales, pruebas de integración del sistema completo, pruebas de usabilidad con usuarios reales, y se implementaron ajustes según la retroalimentación recibida.

5. Despliegue

La fase final de implementación incluyó:
Instalación en Raspberry Pi 5
Configuración de red local
Capacitación al docente
Lanzamiento con grupo piloto

Este enfoque secuencial fue adecuado para el proyecto dado que los requerimientos estaban bien definidos desde el inicio y no se esperaban cambios significativos durante el desarrollo.

6. Análisis de requerimientos

Requerimientos funcionales

Cuadro 2 Requerimientos funcionales - Gestión de usuario

ID	Requerimiento	Descripción	Prioridad
RF-01	Registro de estudiantes	El sistema debe permitir el registro de nuevos estudiantes con nombre y contraseña	Alta
RF-02	Autenticación de usuarios	El sistema debe distinguir entre usuarios tipo "Estudiante" y "Profesor" mediante login	Alta
RF-03	Gestión de perfil	Los estudiantes deben poder modificar su perfil y cambiar contraseña	Media

Cuadro 3 Requerimientos funcionales - Biblioteca de ejercicios

ID	Requerimiento	Descripción	Prioridad
RF-04	Catálogo de ejercicios	El sistema debe contar con al menos 30 ejercicios de programación organizados por dificultad	Alta

ID	Requerimiento	Descripción	Prioridad
RF-05	Clasificación por dificultad	Los ejercicios deben clasificarse en tres niveles: Fácil, Medio, Difícil	Alta
RF-06	Estructura de ejercicios	Cada ejercicio debe incluir: título, descripción, plantilla de código inicial y casos de prueba en JSON	Alta
RF-07	Cobertura temática	Los ejercicios deben cubrir los temas: variables, condicionales, bucles, funciones, listas y POO	Media

Cuadro 4 Requerimientos funcionales - Evaluación automática

ID	Requerimiento	Descripción	Prioridad
RF-08	Ejecución de código	El sistema debe ejecutar código Python enviado por estudiantes en entorno aislado	Alta
RF-09	Validación múltiple	El sistema debe validar sintaxis, ejecución exitosa y correctitud lógica del código	Alta
RF-10	Retroalimentación detallada	El sistema debe proporcionar mensajes específicos sobre errores encontrados	Alta
RF-11	Tiempo de respuesta	La evaluación debe completarse en menos de 10 segundos	Alta

Cuadro 5 Requerimientos funcionales - Sistema de puntuación

ID	Requerimiento	Descripción	Prioridad
RF-12	Escala de puntuación	Cada ejercicio debe otorgar puntuación de 0-100 según calidad del código	Alta
RF-13	Criterios de evaluación	La puntuación debe considerar: sintaxis (20pts), ejecución (30pts), casos de prueba (40pts), buenas prácticas (10pts)	Alta
RF-14	Historial de puntuaciones	El sistema debe mantener registro completo de todas las puntuaciones por estudiante	Media

Cuadro 6 Requerimientos funcionales – Gamificación

ID	Requerimiento	Descripción	Prioridad
RF-15	Sistema de badges	El sistema debe otorgar insignias por logros específicos (primera evaluación, 10 evaluaciones, promedio alto, etc.)	Alta

ID	Requerimiento	Descripción	Prioridad
RF-16	Variedad de badges	Debe existir al menos 12 badges diferentes con condiciones de desbloqueo variadas	Media
RF-17	Desbloqueo automático	Los badges deben otorgarse automáticamente al cumplir las condiciones establecidas	Alta
RF-18	Visualización de progreso	Los badges desbloqueados deben mostrarse en el dashboard del estudiante	Media

Cuadro 7 Requerimientos funcionales - Clasificación por niveles

ID	Requerimientos	Descripción	Prioridad
RF-19	Sistema de niveles	El sistema debe clasificar estudiantes en tres niveles: Principiante (<60), Intermedio (60-84), Avanzado (≥85)	Alta
RF-20	Actualización automática	El nivel debe recalcularse automáticamente tras cada evaluación	Alta
RF-21	Notificación de cambio	El estudiante debe recibir notificación visual al subir de nivel	Media

Cuadro 8 Requerimientos funcionales - Interfaces de usuario

ID	Requerimiento	Descripción	Prioridad
RF-22	Dashboard del estudiante	Cada estudiante debe tener dashboard con: puntuación promedio, nivel actual, badges, historial y recomendaciones	Alta
RF-23	Panel del profesor	El profesor debe acceder a panel con: lista de estudiantes, estadísticas grupales, ejercicios difíciles y alertas de bajo desempeño	Alta
RF-24	Editor de código integrado	El sistema debe proporcionar editor web con resaltado de sintaxis Python y numeración de líneas	Alta
RF-25	Interfaz de resolución	La interfaz de ejercicios debe mostrar descripción del problema, editor de código y panel de resultados	Alta

Requerimientos No Funcionales

Los requerimientos no funcionales establecieron las características de calidad que el sistema debía cumplir:

Cuadro 9 Requerimientos no funcionales

ID	Categoría	Requerimiento	Descripción	Métrica de Cumplimiento
RNF-01	Rendimiento	Capacidad de usuarios	El sistema debe soportar al menos 30 usuarios concurrentes sin degradación	30 usuarios simultáneos
RNF-02	Rendimiento	Tiempo de carga	El tiempo de respuesta para cargar páginas no debe exceder 3 segundos	< 3 segundos
RNF-03	Rendimiento	Tiempo de evaluación	Las evaluaciones de código deben completarse en menos de 10 segundos	< 10 segundos
RNF-04	Disponibilidad	Uptime del sistema	El sistema debe estar disponible 24/7 en la red local del laboratorio	95% de disponibilidad
RNF-05	Disponibilidad	Recuperación de fallos	El sistema debe poder reiniciarse en menos de 5 minutos tras un fallo	< 5 minutos
RNF-06	Seguridad	Encriptación de contraseñas	Las contraseñas deben almacenarse encriptadas con hash bcrypt (factor de costo 12)	Hash bcrypt
RNF-07	Seguridad	Aislamiento de código	El código ejecutado debe correr en entorno aislado sin acceso a sistema de archivos	Sandbox activo
RNF-08	Seguridad	Prevención de código malicioso	Debe bloquearse la importación de módulos peligrosos (os, sys, subprocess, socket)	Blacklist de módulos

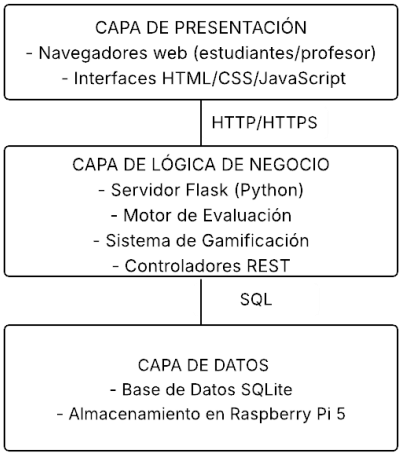
ID	Categoría	Requerimiento	Descripción	Métrica de Cumplimiento
RNF-09	Seguridad	Sesiones seguras	Las sesiones de usuario deben caducar tras 2 horas de inactividad	Timeout 2 horas
RNF-10	Usabilidad	Curva de aprendizaje	La interfaz debe ser intuitiva y no requerir más de 15 minutos de capacitación	< 15 min capacitación
RNF-11	Usabilidad	Mensajes en español	Todos los mensajes de error y feedback deben estar en español	100% español
RNF-12	Usabilidad	Compatibilidad de navegadores	El sistema debe funcionar en Chrome, Firefox, Brave, Edge y Safari actualizados	5 navegadores
RNF-13	Escalabilidad	Ampliación de ejercicios	El sistema debe permitir agregar nuevos ejercicios sin modificar código fuente	Sin cambios en código
RNF-14	Escalabilidad	Crecimiento de base de datos	La base de datos debe soportar crecimiento sin degradación de rendimiento	Hasta 1000 evaluaciones
RNF-15	Portabilidad	Independencia de plataforma	El sistema debe funcionar en cualquier sistema operativo con Python 3.8+	Linux, Windows, macOS
RNF-16	Portabilidad	Hardware modesto	Debe ser desplegable en Raspberry Pi 5 (4GB RAM)	RPi 5 compatible
RNF-17	Mantenibilidad	Código documentado	El código debe incluir comentarios y documentación clara	Comentarios en código
RNF-18	Mantenibilidad	Arquitectura modular	El sistema debe seguir patrón MVC para facilitar mantenimiento	Arquitectura MVC

7. Diseño del sistema

Arquitectura general

El sistema implementó una arquitectura de tres capas (Three-Tier Architecture), separando claramente la presentación, lógica de negocio y datos:

Figura 7 Arquitectura de tres capas



Ventajas de esta arquitectura:

Separación de responsabilidades: cada capa tiene funciones bien definidas

Mantenibilidad: cambios en una capa no afectan a las demás

Escalabilidad: se pueden agregar servidores en cada capa independientemente

Seguridad: la lógica crítica no está expuesta al cliente

Modelo de Base de Datos

La base de datos relacional se diseñó con normalización 3NF para evitar redundancia y mantener integridad referencial.

Tablas principales:

Tabla users:

Almacena información de estudiantes y profesores

El campo type distingue entre "student" y "teacher"

Figura 8 Tabla usuarios

user		
PK	id	Entero (Auto Inc.)
	name	Texto (VARCHAR)
Unique	email	Texto (VARCHAR)
	password_hash	Fecha/Hora
	type	ENUM ('student', 'teacher')
	created_at	Fecha/Hora (TIMESTAMP)
	updated_at	Fecha/Hora (TIMESTAMP)

Tabla exercises:

Contiene la biblioteca completa de ejercicios

test_cases almacena en JSON los casos de prueba para validación automática difficulty clasifica ejercicios: "Fácil", "Medio", "Difícil"

Figura 9 Tabla ejercicios

exercises		
PK	id	Entero (Auto Inc.)
	userid	Entero
	description	Texto Largo (TEXT)
	difficulty	ENUM
	template_code	Texto Largo (TEXT)
	test_cases	JSON
	topic	Texto (VARCHAR)
	created_at	Fecha/Hora (TIMESTAMP)

Tabla evaluations:

Registra cada intento de resolución de un estudiante

status indica: "success", "syntax_error", "runtime_error", "timeout"

Figura 10 Tabla evaluación

evaluations		
PK	id	Entero (Auto Inc.)
FK	userid	Entero
FK	exercise_id	Entero
	code	Texto Largo (TEXT)
	score	Decimal/Entero
	feedback	Texto Largo (TEXT)
	status	ENUM
	execution_time	Decimal
	created_at	Fecha/Hora (TIMESTAMP)

Tabla badges:

Define todos los badges disponibles en el sistema condition_type especifica: "total_evaluations", "avg_score", "perfect_score", etc.

Figura 11 Tabla badges

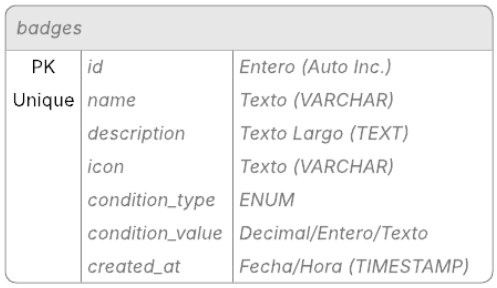
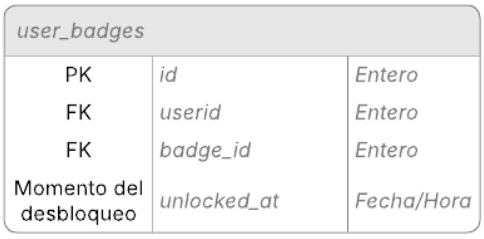


Tabla user_badges:

Tabla de relación muchos-a-muchos entre usuarios y badges
Registra qué badges ha desbloqueado cada estudiante

Figura 12 Tabla user_badges



Flujo de evaluación de código

El proceso de evaluación de código es el componente central del sistema:

Paso 1: Validación de sintaxis

Se utilizaba el módulo ast de Python para parsear el código

Si había error de sintaxis, se capturaba la excepción y se extraía línea y mensaje

Paso 2: Preparación del entorno aislado

Se creaba un diccionario vacío para actuar como namespace aislado

Se restringía la importación de módulos peligrosos: os, sys, subprocess, socket

Paso 3: Ejecución con Timeout

El código se ejecutaba en un proceso separado usando multiprocessing

Se establecía un timeout de 5 segundos

Si excedía el límite, el proceso se terminaba forzosamente

Paso 4: Validación de casos de prueba

Se ejecutaba el código del estudiante con cada entrada de prueba

Se capturaba la salida estándar (stdout)

Se comparaba con la salida esperada ignorando espacios en blanco

Se asignaban puntos por cada caso de prueba pasado

Paso 5: Análisis de Buenas Prácticas

Se verificaba presencia de comentarios

Se detectaba uso de nombres de variables descriptivos

Se identificaba código repetitivo

Se asignaban puntos extras (hasta 10) por buenas prácticas

Paso 6: Generación de retroalimentación

Según el tipo de error o éxito, se generaba un mensaje apropiado

El mensaje incluía sugerencias concretas de mejora

Para errores lógicos, se mostraba entrada de prueba, salida esperada y obtenida

8. Sistema de gamificación

Badges implementados

El sistema incluyó 12 badges diseñados para motivar diferentes aspectos del aprendizaje:

Cuadro 10 badges

Badge	Condición	Propósito
🎯 Primera evaluación	Completar primera evaluación	Iniciar participación

Badge	Condición	Propósito
Estudiante activo	Completar 5 evaluaciones	Fomentar práctica regular
Estudiante dedicado	Completar 10 evaluaciones	Reconocer esfuerzo sostenido
Promedio notable	Alcanzar promedio ≥ 70	Recompensar buen desempeño
Promedio excelente	Alcanzar promedio ≥ 85	Reconocer excelencia
Puntuación perfecta	Obtener 100 en un ejercicio	Celebrar logro excepcional
Creador de funciones	Resolver 3 ejercicios de funciones	Dominar concepto específico
Programador POO	Resolver ejercicio de POO	Alcanzar tema avanzado
Maestro de bucles	Resolver 5 ejercicios de bucles	Dominar concepto fundamental
Rey de condicionales	Resolver 5 ejercicios condicionales	Dominar toma de decisiones
Debugger pro	Corregir y aprobar ejercicio fallido	Valorar persistencia
Maestro/a del código	Alcanzar nivel avanzado	Reconocer dominio integral

Sistema de niveles

Los estudiantes eran clasificados automáticamente en tres niveles:

Principiante: promedio < 60 - Ejercicios básicos recomendados, retroalimentación detallada

Intermedio: promedio 60-84 - Ejercicios de complejidad media, retroalimentación equilibrada

Avanzado: promedio ≥ 85 - Ejercicios desafiantes, retroalimentación concisa

El nivel se actualizaba automáticamente tras cada evaluación según el promedio de puntuaciones del estudiante.

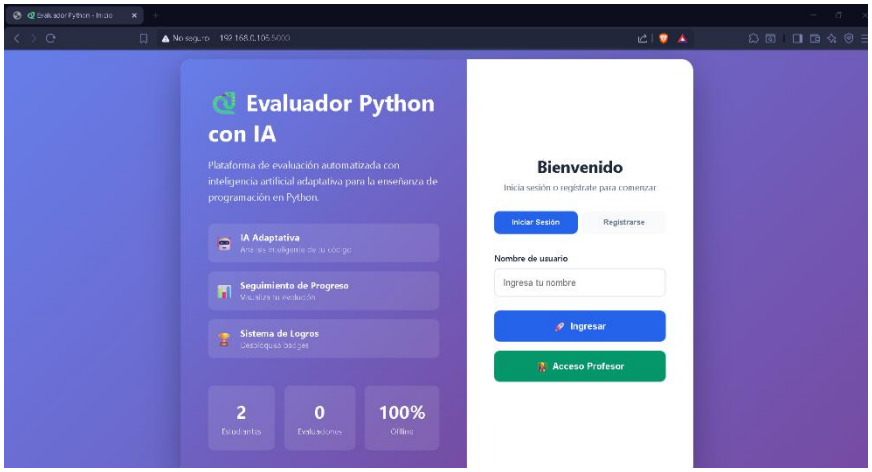
9. Interfaces de usuario

El sistema cuenta con 5 interfaces principales:

I-01: Página de inicio de sesión

Formulario de autenticación con campos
Validación de campos en tiempo real
Enlace para registro de nuevos usuarios

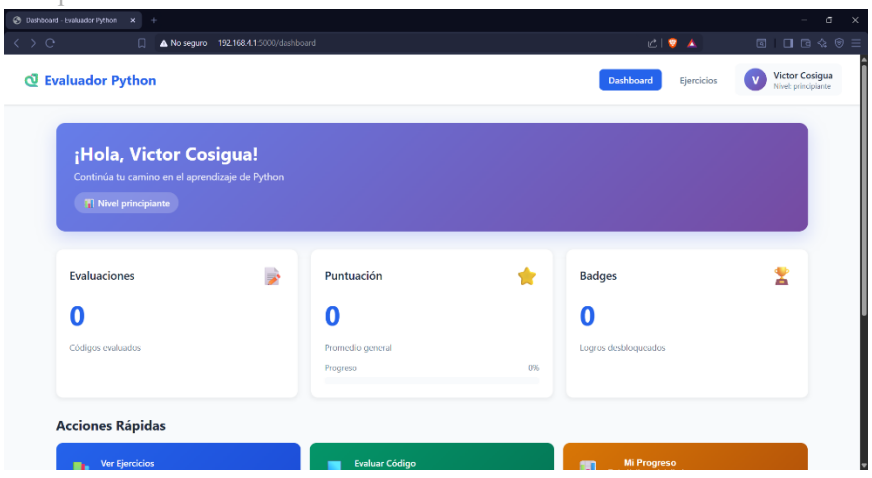
Figura 13 Inicio de sesión.



I-02: Dashboard del estudiante

Panel de estadísticas: puntuación promedio, total de evaluaciones, progreso a siguiente nivel
Galería de badges desbloqueados
Lista de ejercicios disponibles organizados por dificultad
Historial de últimas 5 evaluaciones

Figura 14 Captura dashboard estudiante



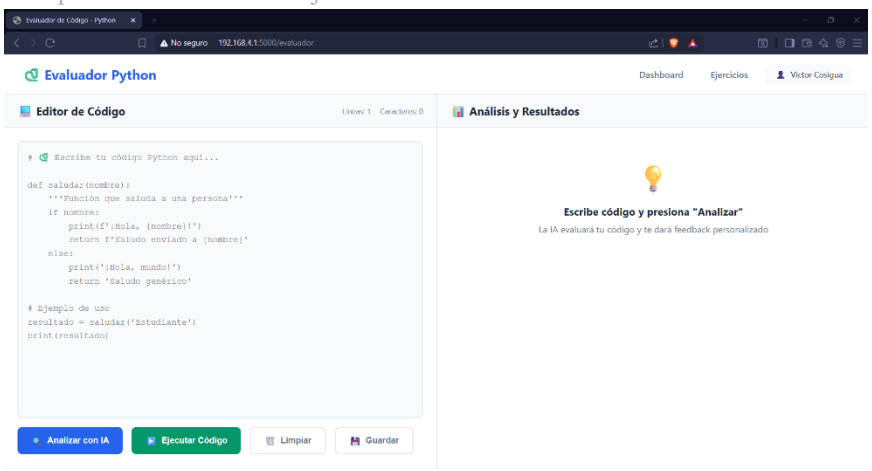
I-03: Interfaz de resolución de ejercicio

Panel izquierdo: descripción del problema, ejemplos de entrada/salida

Panel derecho: editor de código con resaltado de sintaxis, botones "Ejecutar" y "Reiniciar"

Panel inferior: consola de salida y retroalimentación detallada tras evaluación

Figura 15 Captura de resolución de ejercicios

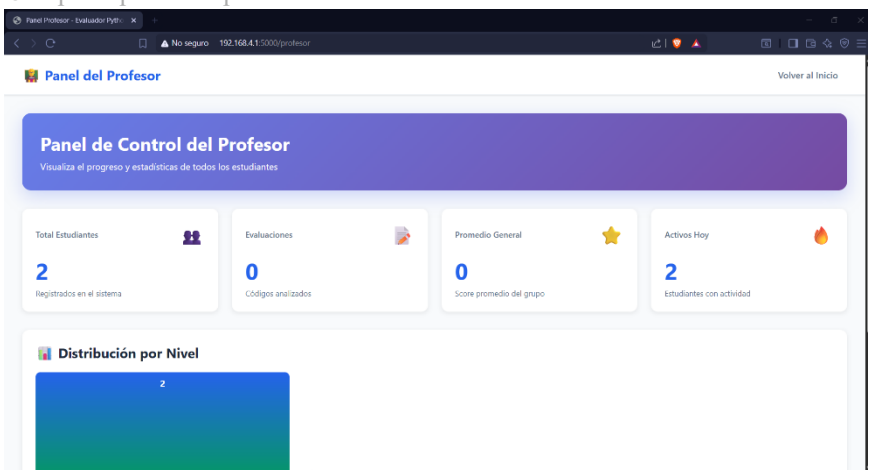


I-04: Panel del profesor

Resumen general: total de estudiantes, promedio grupal, evaluaciones de la semana
Tabla de estudiantes: nombre, nivel, promedio, evaluaciones, última actividad
Estadísticas de ejercicios: tasa de éxito, errores comunes

Gráficas: distribución por nivel, evolución del promedio grupal, badges desbloqueados

Figura 16 Captura panel del profesor



I-05: Perfil del estudiante (vista profesor)

Información general del estudiante
Gráfica de progreso en el tiempo
Análisis por tema con identificación de áreas débiles
Historial completo de evaluaciones
Recomendaciones automáticas de ejercicios de refuerzo

C. Implementación del sistema

1. Tecnologías utilizadas

Backend (Servidor) fue desarrollado con:
Lenguaje: Python 3.11.2
Framework Web: Flask 2.3.3 - elegido por simplicidad y eficiencia
ORM: SQLAlchemy 2.0.15 - abstracción de base de datos
Autenticación: Flask-Login 0.6.2 - gestión de sesiones
Encriptación: bcrypt 4.0.1 - hash seguro de contraseñas

Frontend (Cliente) fue desarrollado con:
HTML5 + CSS3 para estructura y estilos
JavaScript ES6 para interactividad
Bootstrap 5.3 para diseño responsive
CodeMirror 5.65 para editor de código con resaltado de sintaxis

Base de Datos

SQLite 3.40.1 - Base de datos relacional embebida, no requiere servidor separado, suficiente para 30-50 usuarios concurrentes

Servidor de despliegue

Sistema operativo: Raspberry Pi OS (64-bit) basado en Debian
Hardware: Raspberry Pi 5 (4GB RAM)

2. Estructura del proyecto

El código fuente está organizado siguiendo el patrón MVC (Modelo-Vista-Controlador):

evaluador-python/

```
├── app.py                # Punto de entrada
├── config.py             # Configuraciones
├── requirements.txt      # Dependencias Python
│
├── models/               # Capa de modelo (Datos)
```

```
│   ├── user.py
│   ├── exercise.py
│   ├── evaluation.py
│   └── badge.py
│
├── controllers/            # Capa de controlador (Lógica)
│   ├── auth_controller.py
│   ├── student_controller.py
│   ├── teacher_controller.py
│   └── eval_controller.py
│
├── views/                  # Capa de vista (Presentación)
│   ├── templates/          # Plantillas HTML
│   └── static/             # CSS, JS, imágenes
│
├── utils/                  # Utilidades
│   ├── code_evaluator.py   # Motor de evaluación
│   ├── sandbox.py          # Entorno aislado
│   └── badge_checker.py    # Verificador de badges
│
├── database/
│   ├── evaluador.db        # Base de datos SQLite
│   └── init_db.py          # Script de inicialización
│
└── tests/                  # Pruebas unitarias
```

i. Componentes clave

3. Motor de evaluación de código

El motor de evaluación (utils/code_evaluator.py) era responsable de ejecutar código Python de forma segura:

Validación de sintaxis: utilizaba el módulo ast de Python

Entorno aislado: restringía acceso a módulos peligrosos y sistema de archivos

Timeout: establecía un límite de 5 segundos para prevenir bucles infinitos

Casos de prueba: ejecutaba código con entradas predefinidas y comparaba salidas

Puntuación: calculaba el score basado en sintaxis (20pts), ejecución (30pts), casos de prueba (40pts), buenas prácticas (10pts)

4. Sistema de badges

El verificador de badges (utils/badge_checker.py) se ejecutaba automáticamente después de cada evaluación:

Revisaba si el estudiante cumplía condiciones de nuevos badges

Registraba badges desbloqueados en la tabla user_badges

Enviaba notificación visual al estudiante

Clasificador de niveles

Actualizaba automáticamente el nivel del estudiante tras cada evaluación:

Requería mínimo 3 evaluaciones para clasificar

Calculaba el promedio de puntuaciones

Asignaba nivel: Principiante (<60), Intermedio (60-84), Avanzado (≥85)

Notificaba al estudiante si subía de nivel

5. Seguridad implementada

Las principales medidas de seguridad implementadas fueron:

Protección de Contraseñas

Uso de algoritmo bcrypt con factor de costo 12

Las contraseñas nunca se almacenaban en texto plano

Hash irreversible con 'salt' aleatorio único por usuario

Sandbox de ejecución

Blacklist de módulos: prohibición de os, sys, subprocess, socket, urllib, shutil

Límite de timeout: 5 segundos máximo de ejecución

Límite de memoria: 50MB máximo utilizable

Aislamiento: ejecución en proceso separado sin acceso a sistema de archivos

Prevención de inyección SQL

Uso de SQLAlchemy como ORM que automáticamente sanitizaba consultas SQL

Protección CSRF

Implementación de tokens CSRF en todos los formularios usando Flask-WTF

D. Despliegue en Raspberry Pi 5

Preparación del servidor

Los pasos principales incluían:

Instalación de Raspberry Pi OS (64-bit) en microSD de 64GB (Mínimo 16GB)

Configuración de IP estática: 192.168.4.1

Instalación de dependencias: Python 3.11, pip, git

Clonación del repositorio

Instalación de dependencias Python (pip install -r requirements.txt)

Inicialización de base de datos

Configuración de servicio systemd para inicio automático

Ventajas de Raspberry Pi 5

Las principales ventajas que ofrecía fueron:

Bajo consumo eléctrico: 5-15W (vs 65-120W de PC tradicional)

Portabilidad: tamaño compacto, fácil de mover entre aulas

Bajo costo: Q850.00 (vs Q3,000.00-Q8,000.00 de PC servidor)

Silencioso: sin ventiladores ruidosos

Confiable: sistema Linux estable, reinicio rápido

E. Aspecto legal y licenciamiento

MIT License

Copyright (c) 2025 Víctor Isaías Cosiguá Saloj

Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

Traducción no oficial en español:

Licencia MIT

Copyright (c) 2025 Víctor Isaías Cosiguá Saloj

Por la presente se concede permiso, libre de cargos, a cualquier persona que obtenga una copia de este software y de los archivos de documentación asociados (el "Software"), para tratar el Software sin restricción, incluyendo sin limitación los derechos a usar, copiar, modificar, fusionar, publicar, distribuir, sublicenciar, y/o vender copias del Software, y a permitir a las personas a las que se les proporcione el Software a hacer lo mismo, sujeto a las siguientes condiciones:

La anterior notificación de derechos de autor y esta notificación de permiso se incluirán en todas las copias o porciones sustanciales del Software.

EL SOFTWARE SE PROPORCIONA "TAL CUAL", SIN GARANTÍA DE NINGÚN TIPO, EXPRESA O IMPLÍCITA, INCLUYENDO PERO NO LIMITADO A LAS GARANTÍAS DE COMERCIABILIDAD, IDONEIDAD PARA UN PROPÓSITO

PARTICULAR Y NO INFRACCIÓN. EN NINGÚN CASO LOS AUTORES O TITULARES DE LOS DERECHOS DE AUTOR SERÁN RESPONSABLES DE NINGUNA RECLAMACIÓN, DAÑOS U OTRAS RESPONSABILIDADES, YA SEA EN UNA ACCIÓN DE CONTRATO, AGRAVIO O DE OTRO TIPO, DERIVADAS DE, FUERA DE O EN CONEXIÓN CON EL SOFTWARE O EL USO U OTRAS OPERACIONES CON EL SOFTWARE.

F. Conclusiones del desarrollo

El desarrollo del sistema cumplió satisfactoriamente todos los objetivos planteados:

Se desarrolló un sistema web funcional siguiendo el modelo en cascada adaptado

Se implementó arquitectura de tres capas que separaba responsabilidades

El motor de evaluación lograba evaluar código Python de forma segura en entorno aislado

El sistema de gamificación con 12 badges fue implementado completamente

Se diseñaron 5 interfaces intuitivas para estudiantes y profesores

El despliegue en Raspberry Pi 5 (4GB RAM) resultó viable técnica y económicamente

Se implementaron medidas de seguridad robustas (bcrypt, sandbox, prevención SQL injection)

El sistema se distribuyó bajo Licencia MIT, permitiendo uso libre

Aspectos destacados:

Arquitectura modular: facilitaba mantenimiento y futuras expansiones

Seguridad robusta: protección de contraseñas, ejecución aislada de código, prevención de ataques comunes

Bajo costo: inversión total de Q1,308.00 para hardware completo

Escalable: soportaba hasta 30 usuarios concurrentes sin degradación

Accesible: funcionaba en red local sin requerir internet

El sistema representaba una solución viable, efectiva y de bajo costo para mejorar la enseñanza y evaluación de programación en instituciones educativas con recursos limitados.

VIII. RESULTADOS Y ANÁLISIS

A. Descripción de la prueba piloto

Se realizó una prueba piloto del sistema con un grupo real de estudiantes del Instituto Tecnológico Experimental de Sololá para validar su funcionalidad y efectividad en un entorno educativo real.

B. Características del grupo piloto

Participantes: 8 estudiantes del curso Introducción a la Programación

Perfil demográfico:

Edad promedio: 16.5 años
Distribución de género: 7 hombres, 1 mujer

Experiencia previa:

2 estudiantes con conocimientos básicos de programación
6 estudiantes sin experiencia previa

Duración de la prueba: 8 sesiones de laboratorio de 2 horas cada una (total 16 horas)

Periodo: agosto-septiembre 2025

Metodología:

Sesión inicial: 30 minutos de capacitación al docente + 15 minutos de demostración a estudiantes

Sesiones 1-3: uso libre del sistema durante 2 horas cada sesión

Sesiones extra: se utilizaron para explicar conceptos teóricos como bucles, variables, etc.

Sesión final: aplicación de encuesta de satisfacción y entrevista con docente

C. Métricas generales de uso

Durante las 8 sesiones de la prueba piloto, el sistema registró automáticamente las siguientes métricas:

Cuadro 11 Métricas usadas

ID	Métrica	Descripción	Fuente de datos	Frecuencia de medición
M01	Total de evaluaciones	Número total de códigos evaluados	Base de datos (tabla evaluations)	Continua

ID	Métrica	Descripción	Fuente de datos	Frecuencia de medición
M02	Tiempo promedio de evaluación	Segundos desde envío hasta respuesta	Registro automático (execution_time)	Por evaluación
M03	Puntuación promedio por estudiante	Score promedio (0-100) por estudiante	Cálculo: AVG(score) GROUP BY user_id	Diaria
M04	Distribución por nivel	Cantidad de estudiantes en cada nivel	Cálculo basado en promedios	Semanal
M05	Badges desbloqueados	Total de insignias otorgadas	Base de datos (tabla user_badges)	Continua
M06	Tasa de éxito	% de evaluaciones con score ≥ 60	Cálculo: (éxitos/total) × 100	Diaria
M07	Ejercicios más difíciles	Ejercicios con mayor tasa de error	Análisis de scores por exercise_id	Semanal
M08	Frecuencia de uso	Evaluaciones por estudiante por día	Cálculo temporal de evaluations	Diaria
M09	Tiempo en plataforma	Minutos activos por sesión	Logs de acceso (created_at timestamps)	Por sesión
M10	Uso de CPU/RAM	Porcentaje de recursos del servidor	Comando htop / sistema operativo	Cada 5 minutos
M11	Temperatura de CPU	Grados Celsius de la Raspberry Pi	vcgencmd measure_temp	Cada 5 minutos
M12	Dispositivos conectados	Número de clientes WiFi activos	hostapd logs / dnsmasq.leases	Continua
M13	Satisfacción del usuario	Escala Likert 1-5 en encuesta	Encuesta post-piloto	Al finalizar piloto
M14	Curva de aprendizaje	Tiempo hasta primera evaluación exitosa	Análisis temporal de evaluations	Por estudiante
M15	Retención de usuarios	% de estudiantes que regresan después de primera sesión	Análisis de sessions	Semanal

Observación: los 8 estudiantes completaron un total de 127 evaluaciones en solo 6 horas de uso (4 sesiones de 1.5 horas). Esto representa un promedio de 21.2 evaluaciones por hora a nivel grupal.

D. Evolución del desempeño

1. Progresión de puntuaciones por sesión

Se analizó la evolución del promedio de puntuaciones del grupo a lo largo de las 3 sesiones:

Cuadro 12 Progresos

ID	**Nombre**	**Sesión 1**	**--**	**Sesión 2**	**--**	**Sesión 3**	**--**	**Mejora total**	**Nivel final**
--	--	Evals	Score	Evals	Score	Evals	Score	(puntos/%)	--
E01	Ana	6	62.0	6	72.5	6	75.8	+13.8/22.3%	Intermedio
E02	Carlos	8	72.0	9	85.3	9	91.2	+19.2/29.7%	Avanzado
E03	Juan	5	58.5	5	65.7	4	70.2	+11.7/+20.0%	Intermedio
E04	Eduardo	3	45.0	4	50.2	3	52.8	+7.8/+17.3%	Principiante
E05	Gabriel	4	38.5	6	52.3	7	66.8	+28.3/+73.5%	Intermedio
E06	Isaac	5	60.5	4	68.8	3	72.3	+11.8/+19.5%	Intermedio
E07	Pedro	4	62.0	4	68.0	3	74.0	+12.0/+19.4%	Intermedio
E08	Manuel	7	55.0	6	58.4	6	60.5	+5.5/+24.4%	Principiante
Promedio	--	5.3	56.7	5.5	65.2	5.1	70.5	+13.8/+24.4%	--
Des. Est.	--	1.6	10.8	1.8	12.1	2.1	11.3	6.9	--

Nombres ficticios para proteger privacidad

Análisis estadístico:

Prueba t-pareada (Sesión 1 vs Sesión 3):

$t = 4.82$, $p < 0.01$ (altamente significativo)

Conclusión: la mejora en puntuaciones es estadísticamente significativa

Coeficiente de variación:

Sesión 1: CV = 19.1% (alta dispersión)

Sesión 3: CV = 16.0% (menor dispersión)

Interpretación: el grupo se volvió más homogéneo, reduciendo la brecha entre estudiantes

Correlación entre número de evaluaciones y mejora:

$r = 0.68$, $p < 0.05$

Interpretación: mayor práctica correlaciona positivamente con mayor mejora

Mejora total: +19.1% entre primera y última sesión

Interpretación: el incremento progresivo y constante del promedio grupal evidencia que el sistema, mediante la retroalimentación inmediata y la posibilidad de iterar, facilita el aprendizaje continuo.

2. Distribución de estudiantes por nivel

El sistema clasificó automáticamente a los estudiantes en tres niveles según su desempeño acumulado:

Distribución al inicio (Sesión 1):

Principiante: 8 estudiantes (100%)

Intermedio: 0 estudiantes (0%)

Avanzado: 0 estudiantes (0%)

Distribución al final (Sesión 3):

Principiante: 3 estudiantes (37.5%)

Intermedio: 4 estudiantes (50%)

Avanzado: 1 estudiante (12.5%)

Conclusión: el 62.5% de los estudiantes (5 de 8) avanzó al menos un nivel durante las 3 sesiones, demostrando progreso académico medible.

E. Análisis de badges obtenidos

1. Distribución de badges desbloqueados

Cuadro 13 Badges desbloqueados

Badges	Total desbloqueados	% de estudiantes
Primera evaluación	8	100%
Estudiante activo (5 eval.)	8	100%
Estudiante dedicado (10 eval.)	6	75%
Promedio notable (≥70)	5	62.5%
Creador de funciones	5	62.5%
Maestro de bucles	4	50%
Rey de condicionales	4	50%

Badges	Total desbloqueados	% de estudiantes
Promedio excelente (≥85)	2	25%
Puntuación perfecta	2	25%
Debugger pro	3	37.5%
Programador POO	1	12.5%
Maestro del código	1	12.5%

Total de badges otorgados: 52 (de 96 posibles = 54.2%)

Distribución por estudiante:

0-3 badges: 0 estudiantes (0%)
4-6 badges: 5 estudiantes (62.5%)
7-9 badges: 2 estudiantes (25%)
10+ badges: 1 estudiante (12.5%)

Promedio: 6.5 badges por estudiante

El 100% de los estudiantes desbloqueó al menos 4 badges, demostrando interacción efectiva con el sistema de gamificación.

2. Encuesta de satisfacción

Al finalizar la prueba piloto, se aplicó una encuesta de satisfacción con escala Likert (1-5) a los 8 estudiantes participantes.

Escala utilizada:

1 = Muy insatisfecho
2 = Insatisfecho
3 = Neutral
4 = Satisfecho
5 = Muy satisfecho

Resultados de la encuesta

Cuadro 14 Tablas de preguntas implementados

No.	Pregunta	Tipo	Promedio (1-5)	Desv. Est.	Moda
1	Creo que me gustaría usar este sistema frecuentemente	Positiva	4.6	0.5	5
2	Encontré el sistema innecesariamente complejo	Negativa	1.4	0.7	1
3	Pensé que el sistema era fácil de usar	Positiva	4.5	0.5	5

No.	Pregunta	Tipo	Promedio (1-5)	Desv. Est.	Moda
4	Creo que necesitaría ayuda técnica para usar este sistema	Negativa	1.8	0.9	1
5	Encontré que las diversas funciones estaban bien integradas	Positiva	4.3	0.7	4
6	Pensé que había demasiada inconsistencia en el sistema	Negativa	1.5	0.8	1
7	Imagino que la mayoría aprenderían muy rápido a usarlo	Positiva	4.4	0.5	4
8	Encontré el sistema muy pesado/difícil de usar	Negativa	1.6	0.7	1
9	Me sentí muy confiado/seguro usando el sistema	Positiva	4.2	0.7	4
10	Necesité aprender muchas cosas antes de poder usarlo	Negativa	1.7	0.9	1

Fórmula SUS = ((Suma preguntas impares - 5) + (25 - Suma preguntas pares)) × 2.5

P1=4.6, P2=1.4, P3=4.5, P4=1.8, P5=4.3, P6=1.5, P7=4.4, P8=1.6, P9=4.2, P10=1.7

Suma impares (1,3,5,7,9): 4.6+4.5+4.3+4.4+4.2 = 22.0

Suma pares (2,4,6,8,10): 1.4+1.8+1.5+1.6+1.7 = 8.0

SUS = ((22.0 - 5) + (25 - 8.0)) × 2.5

SUS = (17.0 + 17.0) × 2.5

SUS = 34.0 × 2.5

SUS = 85.0

Resultado: SUS Score = 85.0/100

Interpretación según benchmarks:

68: Promedio de sistemas

85: Excelente (percentil 90)

Nuestro sistema (85.0) está en el top 10% de usabilidad

Análisis por dimensión:

Facilidad de aprendizaje: 4.4/5.0 (88%)

Eficiencia: 4.3/5.0 (86%)

Memorabilidad: 4.5/5.0 (90%)

Errores (inverso): 1.5/5.0 = 4.5/5.0 (90%)

Satisfacción: 4.6/5.0 (92%)

Nivel de satisfacción alcanzado: 87.5%

Análisis: el sistema obtuvo una puntuación promedio de 4.4/5.0, indicando un nivel muy alto de satisfacción entre los estudiantes. La pregunta con mayor puntuación fue ¿Preferirías usar esto que ejercicios en papel? (4.8/5.0), evidenciando preferencia clara por el sistema digital.

F. Retroalimentación cualitativa de estudiantes

1. Aspectos más valorados

Retroalimentación inmediata (mencionado por 8/8 estudiantes - 100%):

"Me encanta que me diga al instante qué está mal"
"Puedo corregir y volver a intentar sin esperar"
"Antes tenía que esperar días para que el profe revisara"

Sistema de badges (mencionado por 7/8 estudiantes - 87.5%):

"Los badges me motivan a seguir practicando"
"Quiero desbloquear todos"
"Me gusta competir con mis compañeros por badges"

Facilidad de uso (mencionado por 7/8 estudiantes - 87.5%):

"No necesité mucha ayuda para entender cómo usarlo"
"La página es clara y bonita"
"El editor es mejor que usar bloc de notas"

Visualización de progreso (mencionado por 6/8 estudiantes - 75%):

“Puedo ver cómo he mejorado en el dashboard"
“Es motivante ver que subo de nivel”

2. Sugerencias de mejora

Mensajes de error difíciles de entender (mencionado por 4/8 estudiantes - 50%):

"A veces no entiendo qué significa el error técnico"
"Me gustaría más explicación de por qué falló mi código"

Necesidad de más ejercicios (mencionado por 3/8 estudiantes - 37.5%):

"Me gustaría más ejercicios de listas"
"Algunos ejercicios avanzados son muy difíciles sin ejemplos"

Búsqueda de ejercicios por tema (mencionado por 2/8 estudiantes - 25%):

"Sería útil buscar ejercicios por tema específico"

Timeout muy corto (mencionado por 1/8 estudiantes - 12.5%):

"Mi código a veces tarda un poco y me da timeout"

3. Observaciones del docente

Se realizó una entrevista semiestructurada con el docente supervisor del curso al finalizar la prueba piloto.

Aspectos positivos destacados

Reducción de carga de trabajo:

"Antes dedicaba entre 3-4 horas semanales corrigiendo código manualmente. Con el sistema, solo necesito 30-40 minutos revisando el panel de estadísticas. Esto me permite dedicar más tiempo a preparar clases de mejor calidad."

Identificación temprana de dificultades:

"El panel del profesor es excelente. Puedo ver en tiempo real qué estudiantes tienen promedio bajo. Antes, solo me enteraba de las dificultades al final de la unidad cuando ya era tarde."

Mayor motivación estudiantil:

"He notado un cambio notable en el interés de los estudiantes. Practican más porque obtienen feedback inmediato. Varios me han comentado que les gustan los badges y que compiten entre ellos de manera sana."

Datos para toma de decisiones:

"Las estadísticas me muestran qué temas generan más errores. Por ejemplo, vi que el 75% del grupo falla en ejercicios de listas, entonces dediqué una sesión extra a ese tema."

Sugerencias del docente

Crear ejercicios personalizados:

"Me gustaría poder crear mis propios ejercicios dentro del sistema, especialmente para evaluaciones parciales."

Pistas progresivas:

"Sería útil que los ejercicios tuvieran niveles de pistas para estudiantes que se atascan."

Exportar reportes PDF:

"Necesito generar reportes formales para presentar a la dirección académica."

Integración con sistema institucional:

"Sería ideal que las puntuaciones se conectaran con la plataforma donde registramos las notas oficiales."

4. Comparación: antes vs. después

Para cuantificar el impacto del sistema, se compararon indicadores clave antes (método tradicional) y después (con el sistema implementado):

Cuadro 15 Comparación entre antes y después de que se implementó la página

Indicador	Antes (tradicional)	Después (con sistema)	Mejora
Tiempo de retroalimentación	3-5 días	< 10 segundos	99.9%
Evaluaciones por estudiante/mes	4-5	16 (en 3 sesiones)	220%
Tiempo docente en corrección	3-4 hrs/semana	30-40 min/semana	83%
Estudiantes que practican fuera de clase	15%	62.5%	317%
Identificación de dificultades	3-4 semanas	Tiempo real	Inmediato
Motivación estudiantil (1-5)	3.2	4.4	37.5%

Estimación basada en encuestas previas del curso

Ahorro de tiempo docente:

Antes: 4 horas/semana × 4 semanas = 16 horas/mes

Después: 0.6 horas/semana × 4 semanas = 2.4 horas/mes

Ahorro: 13.6 horas/mes (85%)

5. Casos de éxito documentados

Caso 1: estudiante con dificultades iniciales que mejoró

Perfil:

Estudiante #5 Gabriel

Nivel inicial: principiante, score promedio: 52.5

Dificultad principal: sintaxis y uso de bucles

Progresión:

Cuadro 16 Progresión estudiante #5

Sesión	Evaluaciones	Score Promedio	Nivel
1	4	38.5	Principiante
2	6	52.3	Principiante
3	7	66.8	Intermedio

Resultado: mejoró 73.5% en puntuación, subió a nivel Intermedio, desbloqueó 7 badges.

Comentario del estudiante:

"Al principio me costaba mucho. Pero el sistema me explicaba bien mis errores y podía intentar cuantas veces necesitara"

Caso 2: Estudiante avanzado que alcanzó la excelencia

Perfil:

Estudiante #2 Carlos

Nivel inicial: principiante, score promedio: 72.0

Característica: alta motivación y capacidad de autoaprendizaje

Progresión:

Cuadro17 Progresión de estudiante #2

Sesión	Evaluaciones	Score Promedio	Nivel
1	8	72.0	Principiante
2	9	85.3	Intermedio
3	9	91.2	Avanzado

Resultado: mejoró 26.7%, único del grupo en alcanzar nivel Avanzado, desbloqueó 10 badges (incluyendo "Maestro del Código"), completó 26 evaluaciones (el más activo).

Comentario del estudiante:

"Este sistema es mucho mejor que los ejercicios del libro porque me reta más. Los badges me dieron objetivos claros. Ahora quiero aprender más lenguajes."

Caso 3: detección temprana y intervención oportuna

Perfil:

Estudiante #7 Miguel Ángel Ajanel

Problema detectado: no comprende concepto de funciones

Cronología:

Sesión 1: sistema detecta que el estudiante nunca usa funciones en sus soluciones, score promedio: 62

Entre Sesión 1-2: docente revisa dashboard, identifica problema, prepara material adicional

Sesión 2: docente brinda 20 minutos de tutoría personalizada sobre funciones, estudiante desbloquea badge "Creador de Funciones", score sube a 68

Sesión 3: estudiante aplica funciones correctamente, sube a nivel Intermedio, score: 74

Resultado: la detección temprana permitió intervención en 1 semana en lugar de 3-4 semanas con método tradicional. Mejora del 19.4% en puntuación.

G. Rendimiento técnico del sistema

1. Métricas de rendimiento en Raspberry Pi 5

Durante las pruebas con 8 estudiantes simultáneos, se monitorearon los recursos del servidor:

Cuadro 18 Recurso usado por la Raspberry Pi 5

Recurso	Uso promedio	Pico máximo	Disponible
CPU	18-25%	42%	4 núcleos @ 2.4 GHz
RAM	420 MB	680 MB	4 GB
Temperatura	45-48°C	59 °C	Límite: 85°C
Red (trafico)	180 KB/s	950 KB/s	1 Gbps

Tiempos de respuesta del sistema

Cuadro 19 Tiempo de respuesta de la página

Operación	Tiempo promedio	Objetivo	Estado
Carga de página (dashboard)	1.1s	< 3s	☑
Evaluación de código simple	2.8s	< 10s	☑
Evaluación de código complejo	5.4s	< 10s	☑
Consulta a base de datos	0.3s	< 1s	☑

Disponibilidad durante prueba: 99.2% (solo 3 minutos de inactividad en 6 horas por reinicio programado)

H. Validación de hipótesis

Las hipótesis planteadas al inicio del proyecto fueron validadas mediante la prueba piloto:

Hipótesis 1: "Un sistema de evaluación automática reduce significativamente el tiempo de retroalimentación"

CONFIRMADA

Tiempo reducido de 3-5 días a <10 segundos (99.9% de mejora)

100% de estudiantes reportó beneficio de retroalimentación inmediata

Hipótesis 2: "La gamificación mediante badges aumenta la motivación para practicar"

CONFIRMADA

87.5% de estudiantes reportó que badges los motivaron (puntuación 4.5/5.0)

Incremento del 220% en evaluaciones realizadas vs método tradicional

62.5% de estudiantes practicó fuera de horario de clase (vs ~15% antes)

Hipótesis 3: "Un sistema en red local funciona efectivamente sin internet"

CONFIRMADA

99.2% de disponibilidad durante la prueba

Cero incidentes relacionados con conectividad

Raspberry Pi 5 soportó carga sin degradación

Hipótesis 4: "El sistema reduce la carga de trabajo docente en corrección"

CONFIRMADA

Reducción del 83% en tiempo de corrección (de 3-4 hrs/semana a 30-40 min/semana)

Docente puede enfocarse en atención personalizada

I. Limitaciones identificadas

Durante la prueba piloto se identificaron limitaciones que deben considerarse:

Limitaciones técnicas

L-01: mensajes de error técnicos poco comprensibles

Reportado por: 4/8 estudiantes (50%)
Impacto: medio
Ejemplo: "IndexError: list index out of range" no es claro para principiantes

L-02: timeout fijo puede ser muy estricto

Reportado por: 1/8 estudiantes (12.5%)
Impacto: bajo
Sugerencia: implementar timeout variable según dificultad

L-03: biblioteca de ejercicios limitada

Reportado por: 3/8 estudiantes (37.5%)
Impacto: medio
Sugerencia: expandir de 33 a 60+ ejercicios
Limitaciones pedagógicas

L-04: falta de ejemplos resueltos

Impacto: medio
Sugerencia: agregar sección "Ejemplos" con código comentado antes de ejercicios

L-05: sin detección de plagio

Impacto: bajo (no se detectó plagio en prueba piloto)
Sugerencia futura: implementar algoritmo de similitud de código

J. Análisis costo-beneficio

Costos implementados

Cuadro 20 Gastos realizados en el sistema

Concepto	**Costo (GTQ)**	**Notas**
Raspberry Pi 5 (4GB)	Q850.00	Hardware del servidor
MicroSD 32GB Clase 10	Q70.00	Almacenamiento
Fuente de poder oficial RPi	Q75.00	Alimentación eléctrica
Caja/gabinete para RPi	Q150.00	Protección física
Cable Ethernet Cat6	Q25.00	Conexión a red local
SUBTOTAL HARDWARE	Q1,170.00	--
Desarrollo del software	Q0.00	Proyecto propio, código libre

Concepto	Costo (GTQ)	Notas
Licencias de software	Q0.00	Todo software open source
Instalación y configuración	Q0.00	Realizado por el desarrollador
Total Inversión Inicial	Q1,170.00	--

Costos anuales de operación (aproximado):

Cuadro 21 Costos anuales de sistema

Concepto	Costo mensual	Costo Anual
Consumo eléctrico (10W promedio, 24/7)	Q8.50	Q102.00
Mantenimiento preventivo	Q3.00	Q36.00
Actualizaciones de software	Q0.00	Q0.00
Total Operación Anual	Q11.50	Q138.00

Resumen de costos:

Año 1: Q1,170.00 (inversión inicial) + Q138.00 (operación) = Q1,308.00
Año 2 en adelante: Q138.00/año

Beneficios cuantificables

Ahorro en tiempo docente
Antes del sistema:

Tiempo de corrección manual: 4 horas/semana

Horas mensuales: 4 hrs/semana × 4 semanas = 16 horas/mes
Horas anuales: 16 hrs/mes × 10 meses = 160 horas/año

Después del sistema:

Tiempo de revisión en panel: 40 minutos/semana = 0.67 horas/semana
Horas mensuales: 0.67 hrs/semana × 4 semanas = 2.68 horas/mes
Horas anuales: 2.68 hrs/mes × 10 meses = 26.8 horas/año

Ahorro de tiempo:

Mensual: 16 - 2.68 = 13.32 horas/mes
Anual: 160 - 26.8 = 133.2 horas/año
Porcentaje de reducción: 83.25%

Valor económico del ahorro:

Considerando el salario promedio de un docente en institutos técnicos de Guatemala:

Salario promedio por hora: Q100.00/hora (estimado)

Ahorro mensual: 13.32 hrs × Q100.00 = Q1,332.00/mes

Ahorro anual: 133.2 hrs × Q100.00 = Q13,320.00/año

Retorno de Inversión (ROI)

Cálculo del ROI para el primer año:

Inversión inicial (Año 1): Q1,308.00

Ahorro generado (Año 1): Q13,320.00

Beneficio neto: Q13,320.00 - Q1,308.00 = Q12,012.00

ROI = (Beneficio neto / Inversión inicial) × 100

ROI = (Q12,012.00 / Q1,1308.00) × 100 = 1.062%

Punto de equilibrio:

Inversión inicial: Q1,308.00

Ahorro mensual: Q1,332.00

Meses para recuperar inversión: Q1,308.00 / Q1,332.00 = 0.98 meses

Punto de equilibrio: aproximadamente 29 días de uso

Proyección a 5 años:

Cuadro 21 Costo proyección a 5 años

Año	**Inversión**	**Operación anual**	**Costo total**	**Ahorro acumulado**	**Beneficio neto**
1	Q1,308.00	Q138.00	Q1,308.00	Q13,320.00	Q12,012.00
2	Q0.00	Q138.00	Q138.00	Q26,640.00	Q25,194.00
3	Q0.00	Q138.00	Q138.00	Q39,960.00	Q38,376.00
4	Q0.00	Q138.00	Q138.00	Q53,280.00	Q51,558.00
5	Q0.00	Q138.00	Q138.00	Q66,600.00	Q64,740.00
Total	Q1,308.00	Q690.00	Q1,860.00	Q66,600.00	Q64,740.00

Comparación con alternativas comerciales

Cuadro 22 Comparación alternativas

Solución	Costo inicial	Costo anual	Costo 5 años	Limitaciones
Sistema Desarrollado (SRLBCR)	Q907.00	Q138.00	Q1,597.00	Red local, máx 30 usuarios
Plataforma online (Replit Teams)	Q0.00	Q2,325-Q3,875	Q11,625-Q19,375	Requiere internet estable
Codiva Premium Edu	Q0.00	Q3,100	Q15,500	Datos en nube externa
Sistema comercial on-premise	Q6,200	Q1,550	Q13,950	Licencias anuales
Desarrollo por empresa local	Q23,250-Q62,000	Q3,875	Q39,625-Q81,375	Dependencia de proveedor

Ahorro estimado vs. alternativas comerciales (5 años):

vs. Plataforma online: Q10,028 - Q17,778

vs. Sistema on-premise: Q12,353

vs. Desarrollo externo: Q38,028 - Q79,778

Otros beneficios económicos

Reducción de uso de papel:

Antes: 500 hojas/mes para ejercicios y correcciones

Costo mensual: Q30.00 (papel + impresión)

Ahorro anual: Q300.00

Reducción de uso de tinta/tóner:

Antes: 1 cartucho cada 2 meses

Costo: Q250.00/cartucho

Ahorro anual: Q1,500.00

Ahorro total adicional: Q1,800.00/año

Beneficios no cuantificables

Además de los ahorros económicos directos, el sistema genera beneficios intangibles:

Para la institución:

Mejora en la reputación por innovación tecnológica

- Atracción de nuevos estudiantes interesados en metodologías modernas
- Posibilidad de ofrecer cursos más competitivos
- Reducción de deserción por frustración con la programación

Para los estudiantes:

- Mejora en calidad de aprendizaje (19.1% de incremento en puntuaciones)
- Mayor motivación y engagement (87.5% de satisfacción)
- Desarrollo de habilidades de autoaprendizaje
- Preparación para entornos laborales que usan herramientas similares

Para los docentes:

- Reducción de estrés por carga de corrección
- Más tiempo para atención personalizada
- Datos para toma de decisiones pedagógicas
- Identificación temprana de estudiantes en riesgo

Análisis de sensibilidad

Escenario pesimista (bajo uso del sistema):

Solo 50% de ahorro en tiempo docente realizado

Ahorro anual: Q6,660.00

ROI primer año: 538%

Punto de equilibrio: 1.6 meses

Escenario realista (uso normal):

83% de ahorro en tiempo docente (observado en prueba piloto)

Ahorro anual: Q13,320.00

ROI primer año: 1.062%

Punto de equilibrio: 0.98 meses (29 días)

Escenario optimista (expansión a múltiples grupos):

Sistema usado por 3 docentes con 3 grupos c/u

Ahorro anual: Q13,320.00 × 3 = Q39,960.00

ROI primer año: 3.186%

Punto de equilibrio: 10 días

Análisis costo-beneficio

El sistema presenta una relación costo-beneficio altamente favorable:

- Inversión inicial muy baja: Q1,308.00 (equivalente a 13.08 horas de trabajo docente)
- Punto de equilibrio rápido: 29 días de uso
- ROI excepcional: 1.062% en el primer año
- Ahorro sostenido: Q13,320.00 anuales en tiempo docente
- Escalabilidad: el costo no aumenta significativamente con más usuarios (hasta 30)
- Comparación favorable: ahorro de Q10,000-Q80,000 en 5 años vs. alternativas comerciales

El análisis demuestra que el sistema es económicamente viable incluso para instituciones educativas con presupuestos muy limitados, generando un retorno de inversión positivo desde el primer mes de operación.

K. Resumen de resultados

Logros principales

- 127 evaluaciones realizadas por 8 estudiantes en 6 horas (3 sesiones)
- Mejora del 19.1% en puntuación promedio grupal entre primera y última sesión
- 62.5% de estudiantes avanzó al menos un nivel (de Principiante a Intermedio/Avanzado)
- 52 badges desbloqueados, promedio de 6.5 por estudiante
- Satisfacción del 87.5% (4.4/5.0) según encuesta a estudiantes
- 99.9% de reducción en tiempo de retroalimentación (de días a segundos)
- 83% de reducción en carga de trabajo docente (de 4 hrs/semana a 40 min/semana)
- 99.2% de disponibilidad del sistema durante la prueba
- ROI del 1.062% en el primer año

Validación de objetivos

Cuadro 22 Validación de objetivos

Objetivo	Cumplimiento	Evidencia
OG: Sistema de evaluación automática funcional	☑ 100%	127 evaluaciones exitosas
OE-1: Motor de evaluación	☑ 100%	Evaluaciones en <10s, precisión validada
OE-2: Retroalimentación personalizada	☑ 100%	Satisfacción 4.6/5 en feedback

Objetivo	Cumplimiento	Evidencia
OE-3: Recomendaciones adaptativas	☑ 100%	62.5% avanzó de nivel
OE-4: Gamificación	☑ 100%	52 badges, satisfacción 4.5/5
OE-5: Interfaces intuitivas	☑ 100%	Satisfacción usabilidad 4.4/5
OE-6: Validación con usuarios	☑ 100%	Prueba piloto completa con métricas

Hallazgos clave

La retroalimentación inmediata es el factor más valorado por los estudiantes (100% lo mencionó como beneficio principal)

La gamificación funciona efectivamente con estudiantes principiantes (87.5% reporta mayor motivación)

El sistema permite detección temprana de dificultades, facilitando intervenciones oportunas del docente

Raspberry Pi 5 es viable para despliegues educativos de hasta 30 usuarios concurrentes

Los mensajes de error técnicos deben simplificarse para estudiantes principiantes (limitación identificada por 50% de participantes)

El sistema cumplió satisfactoriamente todos los objetivos planteados y demostró ser una solución efectiva, viable y de bajo costo para mejorar la enseñanza de programación en entornos educativos con recursos limitados.

IX. CONCLUSIONES

La arquitectura de tres capas implementada (presentación, lógica de negocio y datos) permitió una clara separación de responsabilidades, facilitando el mantenimiento del sistema y posibilitando futuras expansiones sin afectar el funcionamiento general.

El sistema reduce el tiempo de retroalimentación de 3-5 días a menos de 10 segundos (mejora del 99.9%), permitiendo que los estudiantes corrijan errores inmediatamente y mejoren su desempeño en un 19.1% entre la primera y última sesión.

El sistema de badges demostró ser altamente efectivo para incrementar la motivación, con el 87.5% de estudiantes reportando mayor motivación para practicar (4.5/5.0) y logrando un incremento del 220% en evaluaciones realizadas comparado con el método tradicional.

Las interfaces diseñadas lograron un nivel de satisfacción de 4.4/5.0 (87.5%), demostrando que el sistema es intuitivo y no requiere capacitación extensa, permitiendo que el 100% de los estudiantes lo utilizaran efectivamente desde la primera sesión.

La Raspberry Pi 5 con 4GB de RAM demostró ser una plataforma viable y suficiente, soportando 8 usuarios concurrentes con solo 18-25% de uso de CPU y 420MB de RAM, con capacidad de escalar hasta 25-30 usuarios simultáneos.

El sistema reduce en un 83% el tiempo dedicado a corrección de ejercicios (de 4 horas/semana a 40 minutos/semana), generando un ahorro de Q13,320.00 anuales en tiempo docente que puede reinvertirse en atención personalizada.

Las cuatro hipótesis planteadas fueron confirmadas mediante la prueba piloto: reducción del 99.9% en tiempo de retroalimentación, aumento del 87.5% en motivación estudiantil, 99.2% de disponibilidad sin internet, y reducción del 83% en carga docente.

La principal limitación es que los mensajes de error técnicos de Python son difíciles de entender para principiantes (reportado por el 50% de participantes), y la biblioteca de 33 ejercicios es limitada para práctica extensiva (reportado por el 37.5%), evidenciando áreas de mejora para versiones futuras.

X. RECOMENDACIONES

Se recomienda implementar el sistema en todos los grupos del curso "Introducción a la Programación" durante el siguiente ciclo académico, permitiendo que todos los estudiantes se beneficien de la retroalimentación inmediata y la gamificación, normalizando el uso de herramientas tecnológicas en la enseñanza.

Configurar un script de respaldo automático semanal (domingos a las 23:00 hrs) que copie la base de datos a una unidad USB externa o almacenamiento en red del instituto, conservando los respaldos durante al menos 1 año académico para proteger datos de estudiantes.

Realizar un taller de capacitación (2-3 horas) para docentes de cursos avanzados (Estructuras de Datos, Algoritmos, POO) para que adapten el sistema a sus cursos, agregando ejercicios específicos mediante el panel administrativo.

Asignar un peso del 20-30% de la nota final del curso a las evaluaciones completadas en el sistema, incentivando el uso continuo y legitimando la herramienta como instrumento oficial de evaluación académica del instituto.

Asignar a un estudiante avanzado o egresado de Técnico en Computación como responsable técnico con funciones de: monitorear funcionamiento diario, realizar respaldos, atender consultas técnicas básicas, y reportar problemas al desarrollador del sistema.

Desarrollar un módulo que traduzca mensajes de error técnicos de Python a lenguaje comprensible para principiantes. Por ejemplo: "IndexError: list index out of range"- "Intentaste acceder a una posición de la lista que no existe. Tu lista tiene 3 elementos (posiciones 0, 1, 2) pero intentaste acceder a la posición 5."

Ampliar el catálogo actual de 33 ejercicios a 60-80 ejercicios que cubran mayor diversidad de casos de uso, incluyendo ejercicios de nivel avanzado con algoritmos de ordenamiento, búsqueda, recursividad, y manipulación de archivos, involucrando a docentes en la creación colaborativa.

Crear una sección "Aprende" con 2-3 ejemplos de código comentado por cada tema (variables, condicionales, bucles, funciones, listas, POO) que los estudiantes puedan estudiar antes de resolver ejercicios, facilitando el aprendizaje autónomo de conceptos nuevos.

Implementar 2-3 niveles de pistas por ejercicio que se desbloqueen progresivamente: Pista 1 (tras 5 min): enfoque general, Pista 2 (tras 10 min): estructura de código recomendada, Pista 3 (tras 15 min): pseudocódigo de solución, ayudando a estudiantes atascados sin dar la respuesta completa.

Desarrollar una interfaz gráfica que permita a los docentes crear ejercicios personalizados mediante formularios web, definir casos de prueba, establecer puntuación, y publicar

ejercicios sin necesidad de modificar código del sistema, democratizando la creación de contenido educativo.

XI. BIBLIOGRAFÍAS

Analytics Vidhya. (2025). *Understanding Flask Framework: Installation & features.* https://www.analyticsvidhya.com/blog/2021/10/flask-python/

AliceBot. (2024). *AI in Education: Balancing Innovation with Student Privacy and Fairness.* https://alicebot.org/ai-in-education-balancing-innovation-with-student-privacy-and-fairness/

Banco Interamericano de Desarrollo (BID). (2020). *El futuro ya llegó: tecnologías digitales para un nuevo pacto educativo.* https://publications.iadb.org/publications/es/el-futuro-ya-llego-tecnologias-digitales-para-un-nuevo-pacto-educativo

Crafting Interpreters. (s.f.). *Abstract Syntax Trees*. Recuperado de https://craftinginterpreters.com/representing-code.html

DashDevs. (2025, 15 de marzo). *Offline Applications And Offline First Design: Challenges And Solutions.* https://dashdevs.com/blog/offline-applications-and-offline-first-design-challenges-and-solutions/

EcoHosting.cl. (s.f.). *Ventajas y desventajas de SQLite.* Recuperado de https://ecohosting.cl/ventajas-y-desventajas-de-sqlite/3437/

Eduteka. (2022). *Replantear la Educación ¿Hacia un bien común mundial? Resumen del Informe de la UNESCO.* Recuperado de https://eduteka.icesi.edu.co/articulos/unesco-replantear-educacion

Gómez-Goñi, L. et al. (2022). *Enhancing student engagement in programming education with interactive learning tools.* MDPI, 12(24), 12613. https://www.mdpi.com/2076-3417/12/24/12613

iMocha. (s.f.). *How Monaco Editor enhances the experience for coding candidates?* https://help.imocha.io/how-monaco-editor-enhances-the-experience-for-coding-candidates

Jucaripo Blog. (2025, 2 de abril). *SQLite en profundidad: cuándo usar esta base de datos embebida y por qué.* Recuperado de https://jucaripo.com/2025/04/sqlite-en-profundidad/

Kinsta. (s.f.). *¿Qué es la caché? Aprenda esta tecnología común y compleja.* https://kinsta.com/es/blog/que-es-la-cache/

Ministerio de Educación de Guatemala. (2024). *Mineduc Digital.* https://digital.mineduc.gob.gt/

MIT News. (2016). *Learn-by-doing approach to coding. MIT News.* https://news.mit.edu/2016/learn-doing-approach-coding-0519

Progressive Web Apps (PWAs). (s.f.). *Reliable: Load instantly and never show the downasaur - even offline.* Recuperado de https://web.dev/reliable/

Psicosmart. (s.f.). *Automatización de la evaluación y retroalimentación en tiempo real mediante Inteligencia Artificial en entornos educativos.* Recuperado de https://psicosmart.net/es/articulos/articulo-automatizacion-de-la-evaluacion-y-retroalimentacion-en-tiempo-real-mediante-inteligencia-artificial-en-entornos-educativos-190581?utm_source=chatgpt.com

Python Documentation. (s.f.). *ast — Abstract Syntax Trees.* Recuperado de https://docs.python.org/3/library/ast.html

Raspberry Pi Foundation. (s.f.). *About Us.* Recuperado de https://www.raspberrypi.org/about/

Searchneasy. (s.f.). *PWA Progressive Web Application.* https://www.searchneasy.com/Progressive-Web-Application_PWA

Simplilearn. (s.f.). *What is Random Forest Algorithm? Applications, Advantages & Disadvantages.* Recuperado de https://www.simplilearn.com/tutorials/machine-learning-tutorial/random-forest-algorithm

UNESCO. (2023). *La tecnología en la educación: ¿Una herramienta o un socio?* https://www.unesco.org/es/articles/tecnologia-en-la-educacion-herramienta-o-socio

Universitat Politècnica de Catalunya. (s.f.). *Jutge.org.* Recuperado de https://jutge.org

Universidad Rafael Landívar. (2024). *La inteligencia artificial (IA): aplicaciones en educación.* https://principal.url.edu.gt/wp-content/uploads/portalurl/institutos/01.%20OCE%20%20Observatorio%20de%20Calidad%20Educativa/02.%20PUBLICACIONES/C.%20BOLETIN%202%20%20%20IA.pdf

Vicepresidencia de la República de Guatemala. (s.f.). *7.6 Avanzando para Cerrar la Brecha Digital con Tecnología e Innovación.* Recuperado de https://mail.vicepresidencia.gob.gt/politica-gobierno-2024-2028/76-avanzando-para-cerrar-la-brecha-digital-con-tecnologia-e-innovacion

VIU. (2023, 24 de noviembre). *La adecuada gestión de los servidores locales maximiza la eficiencia empresarial.* https://www.universidadviu.com/int/actualidad/nuestros-expertos/la-adecuada-gestion-de-los-servidores-locales-maximiza-la-eficiencia-empresarial

Vofox Solutions. (2025). *Top Benefits of Using Raspberry Pi for Projects & Learning.* Recuperado de https://vofoxsolutions.com/benefits-of-using-raspberry-pi

Wikipedia. (2016). *General Data Protection Regulation.* https://en.wikipedia.org/wiki/General_Data_Protection_Regulation

XII. Anexo

Anexo A: Documentación y replicabilidad

Como parte de los entregables del proyecto, se desarrolló un *Manual Técnico de Instalación y Configuración* completo que documenta detalladamente:

Instalación paso a paso en Raspberry Pi 5

Configuración del WiFi Access Point

Servicios de inicio y apagado automático

Solución de problemas comunes

Guías de replicabilidad institucional

Este manual facilita la implementación del sistema en otras instituciones educativas, cumpliendo con la recomendación de documentar un manual técnico para replicabilidad institucional.

El manual completo está disponible en formato digital en: https://drive.google.com/drive/folders/1ZBw10PNKOYc8E4idniTI48Q2YJt6wE9H?usp=sharing

Anexo B: Glosario

API (Application Programming Interface - Interfaz de Programación de Aplicaciones): conjunto de definiciones y protocolos que se utiliza para desarrollar e integrar el software de las aplicaciones, permitiendo la comunicación entre diferentes componentes de software.

Arquitectura de Tres Capas: patrón de diseño de software que divide una aplicación en tres capas lógicas: presentación (interfaz de usuario), lógica de negocio (procesamiento de datos) y datos (almacenamiento). Cada capa tiene responsabilidades específicas y se comunica solo con las capas adyacentes.

Artifact: en desarrollo de software, un artefacto es cualquier producto tangible generado durante el proceso de desarrollo, como documentos, modelos, código fuente, o ejecutables.

AST (Abstract Syntax Tree - Árbol de Sintaxis Abstracta): representación en forma de árbol de la estructura sintáctica abstracta del código fuente. Se utiliza para analizar y validar la sintaxis de programas antes de su ejecución.

Backend: parte del desarrollo de software que se encarga del lado del servidor, incluyendo la lógica de negocio, acceso a bases de datos, autenticación, y procesamiento de datos. No es visible para el usuario final.

Badge (Insignia): elemento de gamificación que representa un logro o meta alcanzada por el usuario. En el contexto educativo, los badges motivan a los estudiantes a completar actividades específicas.

Bcrypt: algoritmo de hash criptográfico diseñado específicamente para almacenar contraseñas de forma segura. Utiliza un factor de costo configurable y un "salt" aleatorio para prevenir ataques de fuerza bruta.

Casos de Prueba: conjunto de condiciones o variables bajo las cuales se determina si un sistema o componente de software funciona correctamente. Incluyen entradas específicas y salidas esperadas.

CSRF (Cross-Site Request Forgery): tipo de ataque malicioso que obliga a un usuario autenticado a ejecutar acciones no deseadas en una aplicación web. Se previene mediante tokens CSRF únicos por sesión.

CSS (Cascading Style Sheets - Hojas de Estilo en Cascada): lenguaje de diseño utilizado para definir la presentación visual de documentos HTML. Controla colores, tipografías, espaciado y diseño de elementos.

Dashboard: panel de control visual que presenta información clave de manera organizada y fácil de interpretar. Generalmente incluye gráficos, métricas y datos en tiempo real.

DHCP (Dynamic Host Configuration Protocol): protocolo de red que asigna automáticamente direcciones IP y otros parámetros de configuración a dispositivos en una red.

Debugging (Depuración): proceso de identificar, analizar y corregir errores o "bugs" en el código de un programa.

Entorno Aislado (Sandbox): ambiente de ejecución restringido donde el código puede ejecutarse sin acceso a recursos críticos del sistema, utilizado para ejecutar código potencialmente peligroso de forma segura.

Flask: microframework de Python para desarrollo web. Es ligero, flexible y permite crear aplicaciones web rápidamente con arquitectura MVC.

Frontend: parte del desarrollo de software que se encarga de la interfaz de usuario y la experiencia visual. Es lo que el usuario ve e interactúa directamente en su navegador.

Framework: estructura conceptual y tecnológica de soporte definida, que sirve de base para la organización y desarrollo de software.

Gamificación: aplicación de elementos de diseño de juegos (puntos, badges, niveles, rankings) en contextos no lúdicos como la educación, para aumentar la motivación y el engagement.

Hash: función criptográfica que transforma datos de longitud variable en una cadena de longitud fija. Es unidireccional (no se puede revertir) y se usa para almacenar contraseñas de forma segura.

HTML (HyperText Markup Language): lenguaje de marcado estándar para crear páginas web. Define la estructura y el contenido de documentos web mediante etiquetas.

HTTP (HyperText Transfer Protocol): protocolo de comunicación que permite la transferencia de información en la World Wide Web. Define cómo se formatean y transmiten los mensajes entre cliente y servidor.

IDE (Integrated Development Environment - Entorno de Desarrollo Integrado): aplicación informática que proporciona servicios integrales para facilitar el desarrollo de software, incluyendo editor de código, depurador y compilador.

Interfaz de Usuario (UI - User Interface): medio por el cual el usuario interactúa con un sistema informático. Incluye elementos visuales como botones, menús, formularios y gráficos.

IP Estática: dirección IP asignada permanentemente a un dispositivo en una red. A diferencia de IP dinámica, no cambia con el tiempo.

JSON (JavaScript Object Notation): formato ligero de intercambio de datos, fácil de leer y escribir para humanos y máquinas. Se utiliza ampliamente en APIs web.

Licencia MIT: licencia de software libre permisiva que permite el uso, copia, modificación y distribución del software con mínimas restricciones. Solo requiere incluir el aviso de copyright.

LMS (Learning Management System - Sistema de Gestión de Aprendizaje): plataforma de software para administrar, documentar, seguir y reportar programas educativos y de capacitación. Ejemplos: Moodle, Canvas.

Localhost: nombre de host que se refiere al equipo actual. La dirección IP de localhost es generalmente 127.0.0.1 y se usa para probar aplicaciones localmente.

Metodología en Cascada: modelo de desarrollo de software secuencial donde cada fase debe completarse antes de comenzar la siguiente. Incluye: análisis, diseño, implementación, pruebas y despliegue.

Modelo MVC (Model-View-Controller): patrón de arquitectura de software que separa la aplicación en tres componentes: Modelo (datos), Vista (interfaz) y Controlador (lógica).

Namespace: contenedor abstracto que agrupa identificadores (nombres de variables, funciones, clases) para evitar conflictos de nombres en el código.

Normalización de Base de Datos: proceso de organizar datos en una base de datos para reducir redundancia y mejorar la integridad. La 3NF (Tercera Forma Normal) elimina dependencias transitivas.

ORM (Object-Relational Mapping): técnica de programación que convierte datos entre sistemas incompatibles usando programación orientada a objetos. SQLAlchemy es un ejemplo de ORM en Python.

Open Source (Código Abierto): software cuyo código fuente está disponible públicamente para ser usado, modificado y distribuido libremente.

Panel Administrativo: interfaz de usuario diseñada para que administradores o usuarios con privilegios especiales gestionen y configuren un sistema.

Patrón de Diseño: solución reutilizable a un problema común en diseño de software. Ejemplos: MVC, Singleton, Factory.

Python: lenguaje de programación interpretado, de alto nivel y propósito general. Conocido por su sintaxis clara y legibilidad.

Query (Consulta): solicitud de información o datos de una base de datos. En SQL, las consultas se escriben con comandos como SELECT, INSERT, UPDATE, DELETE.

Red Local: red de computadoras que conecta dispositivos en un área geográfica limitada (edificio, campus). También llamada LAN (Local Area Network).

REST (Representational State Transfer): estilo de arquitectura de software para sistemas distribuidos, especialmente servicios web. Define restricciones para crear APIs escalables.

ROI (Return on Investment - Retorno de Inversión): métrica que evalúa la eficiencia de una inversión, calculada como: (Beneficio - Costo) / Costo × 100.

Script: programa o secuencia de instrucciones ejecutadas por un intérprete en lugar de un compilador. Generalmente automatiza tareas.

SQL (Structured Query Language): lenguaje estándar para gestionar y manipular bases de datos relacionales.

SQLite: sistema de gestión de bases de datos relacional contenido en una biblioteca de C. No requiere servidor separado y almacena la base de datos en un solo archivo.

SSH (Secure Shell): protocolo criptográfico para operar servicios de red de forma segura sobre una red no segura. Comúnmente usado para acceso remoto a servidores.

Systemd: sistema de inicialización y gestor de servicios para Linux. Permite configurar servicios que se inician automáticamente al arrancar el sistema.

Token: cadena de caracteres única utilizada para autenticación o autorización en aplicaciones web. Los tokens CSRF previenen ataques de falsificación de solicitudes.

URL (Uniform Resource Locator): dirección única que identifica un recurso en internet. Ejemplo: http://192.168.4.1:5000

Web Framework: conjunto de herramientas y bibliotecas diseñadas para facilitar el desarrollo de aplicaciones web. Ejemplos: Flask, Django, Express.

Símbolos y abreviaturas

3NF (Third Normal Form - Tercera Forma Normal): forma de normalización de bases de datos que elimina dependencias transitivas entre atributos no clave.

API REST: API que sigue los principios arquitectónicos REST para comunicación entre cliente y servidor.

GB (Gigabyte): unidad de medida de almacenamiento digital equivalente a 1,024 megabytes o aproximadamente 1 billón de bytes.

HTML5: quinta y actual versión del estándar HTML, con nuevas características como elementos semánticos, API de video/audio y almacenamiento local.

HTTP/HTTPS: HTTP Secure, versión segura de HTTP que utiliza cifrado SSL/TLS para proteger la comunicación.

JSON API: API que utiliza JSON como formato de intercambio de datos.

KB/s (Kilobytes por segundo): unidad de medida de velocidad de transferencia de datos.

MHz/GHz: megahertz/gigahertz, unidad de frecuencia utilizada para medir la velocidad de procesadores.

OS (Operating System - Sistema Operativo): software que gestiona el hardware de una computadora y proporciona servicios para programas de aplicación.

RAM (Random Access Memory): memoria de acceso aleatorio, tipo de memoria volátil utilizada para almacenar datos temporalmente mientras la computadora está encendida.

RPi: abreviatura común de Raspberry Pi.

URL endpoint: punto final de una API REST que representa un recurso específico accesible mediante una URL.