Pontificia Universidad
JAVERIANA
Cali

Prototipo de un sistema inteligente para la retroalimentación continua y la identificación de falencias de aprendizaje en estudiantes de Ingeniería de Sistemas y Computación

Isabella Victoria Henríquez

Trabajo de Grado para optar al título de
Ingeniero(a) de Sistemas y Computación

Director
Dr. Gerardo Mauricio Sarria Montemiranda

Codirectora
Dr. Marcela Valencia Serrano

Pontificia Universidad Javeriana Cali
Facultad de Ingeniería y Ciencias
Ingeniería de Sistemas y Computación

Santiago de Cali, Enero de 2026

Cartas de Aprobación

Santiago de Cali, enero de 2026.

Señores.
PONTIFICIA UNIVERSIDAD JAVERIANA – Seccional CALI
Dr. Gerardo Mauricio Sarria
Director Carrera Ingeniería de Sistemas y Computación
Cali.

Cordial saludo,

Por medio de la presente me permito informarle que la estudiante de Ingeniería de Sistemas y Computación, Isabella Victoria Henríquez (cód: 8961991), trabajó bajo mi dirección en el proyecto de grado titulado *"Prototipo de un sistema inteligente para la retroalimentación continua y la identificación de falencias de aprendizaje"*, el cual se encuentra finalizado y listo para sustentación.

Atentamente,

Dr. Gerardo Mauricio Sarria Montemiranda

Santiago de Cali, enero de 2026.

Señores.
PONTIFICIA UNIVERSIDAD JAVERIANA – Seccional CALI
Dr. Gerardo Mauricio Sarria
Director Carrera Ingeniería de Sistemas y Computación
Cali.

Cordial saludo,

Por medio de la presente me permito informarle que la estudiante de Ingeniería de Sistemas y Computación, Isabella Victoria Henríquez (cód: 8961991), trabajó bajo mi codirección en el proyecto de grado titulado *"Prototipo de un sistema inteligente para la retroalimentación continua y la identificación de falencias de aprendizaje"*, el cual se encuentra finalizado y listo para sustentación.

Atentamente,

Dr. Marcela Valencia Serrano

Carta de Compromiso

Santiago de Cali, enero de 2026.

Señores.
PONTIFICIA UNIVERSIDAD JAVERIANA – Seccional CALI
Dr. Gerardo Mauricio Sarria
Director Carrera Ingeniería de Sistemas y Computación
Cali.

Cordial saludo,

Me permito presentar a su consideración el trabajo de grado titulado *"Prototipo de un sistema inteligente para la retroalimentación continua y la identificación de falencias de aprendizaje"* con el fin de cumplir con los requisitos exigidos por la Universidad para llevar a cabo el proyecto de grado y posteriormente optar al título de Ingeniera de Sistemas y Computación.

Al firmar aquí, doy fe que entiendo y conozco la directriz 35 para los trabajos de grado en los programas académicos de pregrado aprobada el 17 de junio de 2024, donde se establecen los plazos y normas para el desarrollo del anteproyecto y del trabajo de grado.

Atentamente,

Isabella Victoria Henríquez.

Isabella Victoria Henríquez

Resumen

El aprendizaje autorregulado constituye un proceso fundamental en la educación superior, especialmente en carreras con un alto nivel de contenido teórico, como la Ingeniería de Sistemas y Computación. Este proceso implica que el estudiante planifique, supervise y ajuste sus estrategias de aprendizaje con el fin de alcanzar objetivos académicos específicos. Diversos estudios han evidenciado que el fortalecimiento de habilidades autorregulatorias se relaciona con la permanencia y el rendimiento académico en programas de ciencias de la computación. En este contexto, la retroalimentación cumple un papel central, ya que proporciona información que permite al estudiante reconocer discrepancias entre su desempeño actual y los objetivos de aprendizaje, facilitando la corrección de errores conceptuales y el ajuste de estrategias.

La literatura reciente sobre retroalimentación automatizada destaca que los sistemas capaces de ofrecer feedback formativo oportuno y específico pueden apoyar significativamente el proceso de aprendizaje, siempre que este se encuentre alineado con los criterios de la tarea. Asimismo, los sistemas inteligentes, entendidos como sistemas capaces de analizar información y actuar con base en objetivos definidos, representan una oportunidad para ampliar el acceso a retroalimentación continua en escenarios donde la desproporción entre docentes y estudiantes limita el acompañamiento personalizado.

En el contexto específico del curso *Introducción a la Programación* de la Pontificia Universidad Javeriana Cali, ofrecido hasta 2024, los grupos académicos solían estar conformados por más de 25 estudiantes bajo la orientación de un único docente. Esta dinámica dificultaba la posibilidad de ofrecer un acompañamiento individualizado constante y de dedicar tiempo suficiente a la revisión detallada de cada respuesta. Adicionalmente, era frecuente que algunos estudiantes no expresaran sus dudas en clase ni se acercaran a solicitar apoyo por temor, inseguridad o vergüenza, lo que podía limitar la identificación temprana de falencias conceptuales.

A partir de esta problemática, el presente proyecto se centra en el análisis de una tarea específica del curso, orientada al refuerzo de la habilidad para abstraer sistemas, en la cual los estudiantes debían identificar el propósito de un sistema, sus componentes, las interacciones entre estos y la clasificación de variables y constantes. Con base en respuestas reales de estudiantes, se construyó una taxonomía de errores propia del proyecto, fundamentada en los errores conceptuales más recurrentes observados en la ejecución de la tarea. Este enfoque permitió diseñar un prototipo de sistema inteligente capaz de analizar respuestas abiertas de texto, detectar errores básicos y generar retroalimentación automatizada y continua.

La arquitectura del sistema se fundamentó en el modelo C4, utilizando un enfoque híbrido de clasificación que combina técnicas de aprendizaje supervisado (específicamente Máquinas de Vectores de Soporte o SVM) y un clasificador basado en reglas. Además, se integró la API de OpenAI para la generación automatizada de mensajes de retroalimentación pedagógica, basándose en la taxonomía definida para el proyecto.

Los resultados obtenidos durante la fase de entrenamiento mostraron una exactitud del 89.13 % en la clasificación de respuestas, lo que demuestra la viabilidad técnica de la solución. La plataforma, desarrollada sobre Oracle APEX, facilita una interfaz web intuitiva para que estudiantes y docentes gestionen tareas y consulten el historial de errores. Finalmente, la validación funcional confirmó que el prototipo opera de manera integrada, cumpliendo con los requisitos de identificar falencias académicas y brindar apoyo al aprendizaje autorregulado en el contexto de la ingeniería.

Palabras Clave: *Sistema inteligente de tutoría, retroalimentación automatizada, falencias de aprendizaje, ingeniería de sistemas, aprendizaje automático, procesamiento de lenguaje natural, modelo C4, Oracle APEX.*

Abstract

Self-regulated learning is a fundamental process in higher education, particularly in curricula with high theoretical content, such as Systems and Computing Engineering. This process requires students to plan, monitor, and adjust their learning strategies to achieve specific academic goals. Various studies have shown that strengthening self-regulatory skills is directly linked to student retention and academic performance in computer science programs. In this context, feedback plays a central role by providing information that allows students to recognize discrepancies between their current performance and learning objectives, thereby facilitating the correction of conceptual errors and the adjustment of strategies.

Recent literature on automated feedback highlights that systems capable of offering timely and specific formative feedback can significantly support the learning process, provided they are aligned with task criteria. Furthermore, intelligent systems—defined as systems capable of analyzing information and acting based on defined objectives—represent an opportunity to expand access to continuous feedback in scenarios where high student-to-teacher ratios limit personalized guidance.

In the specific context of the *Introduction to Programming* course at Pontificia Universidad Javeriana Cali, offered until 2024, academic groups typically consisted of more than 25 students under the guidance of a single instructor. This dynamic hindered the possibility of offering constant individualized support and dedicating sufficient time to the detailed review of each response. Additionally, students often refrained from expressing doubts in class or seeking support due to fear, insecurity, or embarrassment, which could limit the early identification of conceptual gaps.

To address this problem, this project focuses on the analysis of a specific course task aimed at reinforcing system abstraction skills, where students were required to identify a system's purpose, its components, their interactions, and the classification of variables and constants. Based on real student responses, a project-specific error taxonomy was developed, grounded in the most recurrent conceptual errors observed during task execution. This approach enabled the design of an intelligent system prototype capable of analyzing open-ended text responses, detecting basic errors, and generating automated, continuous feedback.

The system architecture was based on the C4 model, utilizing a hybrid classification approach that combines supervised learning techniques—specifically Support Vector Machines (SVM)—and a rule-based classifier. Additionally, the OpenAI API was integrated for the automated generation of pedagogical feedback messages based on the taxonomy defined for the project.

Results obtained during the training phase showed a classification accuracy of 89.13 %, demonstrating the technical feasibility of the solution. The platform, developed on Oracle APEX, provides an intuitive web interface for students and instructors to manage tasks and consult error history. Finally, functional validation confirmed that the prototype operates in an integrated manner, fulfilling the requirements to identify academic weaknesses and provide support for self-regulated learning in

an engineering context.

Keywords: *Intelligent tutoring system, automated feedback, learning gaps, systems engineering, machine learning, natural language processing, C4 model, Oracle APEX.*

Índice general

Capítulo 1

Introducción

La formación universitaria en carreras con un alto componente teórico y técnico, como lo son la ingeniería de sistemas y las ciencias de la computación, presenta desafíos persistentes en la comprensión de sus temáticas fundamentales. Las dificultades de aprendizaje, especialmente en asignaturas que sientan las bases de la programación, afectan significativamente la motivación de los estudiantes y pueden derivar en la deserción académica [1].

En este contexto, la retroalimentación desempeña un papel central en el proceso de aprendizaje, ya que permite al estudiante reconocer discrepancias entre su desempeño actual y los objetivos académicos esperados, favoreciendo procesos de autorregulación y ajuste estratégico [2]. Sin embargo, en la práctica universitaria, la provisión de retroalimentación efectiva enfrenta limitaciones estructurales. Las clases magistrales, aunque ampliamente utilizadas, no siempre logran captar el interés ni facilitar la comprensión de los conceptos complejos [3]. Como resultado, los estudiantes buscan recursos externos para reforzar su aprendizaje, lo que en ocasiones genera discrepancias entre lo que estudian de manera independiente y los objetivos de la asignatura.

Uno de los factores más relevantes en este proceso es la escasa retroalimentación de manera oportuna, que se ve limitada por la desproporción entre el número de docentes y el de estudiantes, así como por la dificultad de atender de forma individual las necesidades específicas de cada uno por motivos de tiempo. En los últimos años, diferentes investigaciones han explorado el uso de sistemas automatizados para ofrecer retroalimentación formativa en contextos educativos, apoyándose en técnicas de aprendizaje automático y procesamiento de lenguaje natural. Estos trabajos evidencian que es posible clasificar respuestas abiertas y generar sugerencias automáticas; no obstante, también señalan retos importantes como la necesidad de datasets adecuados, la dificultad para capturar errores conceptuales complejos y la integración de criterios pedagógicos sólidos dentro del diseño del sistema [4, 5, 6].

En este contexto, surge la necesidad de desarrollar herramientas tecnológicas que permitan ofrecer un acompañamiento más cercano y adaptado a las necesidades específicas del contexto. Particularmente, los sistemas inteligentes representan una oportunidad para identificar falencias académicas de manera automatizada y generar

sugerencias útiles en tiempo real que refuercen el proceso de aprendizaje.

Este proyecto propone el diseño e implementación de un prototipo de sistema inteligente de tutoría que analiza las respuestas de los estudiantes, detecta errores básicos y proporciona retroalimentación automatizada basada en una tarea del curso de Introducción a la Programación, ofrecido por la Pontificia Universidad Javeriana Cali hasta 2024. El sistema se fundamenta en una arquitectura modular, utilizando el modelo C4 para el diseño. El modelo C4 es una metodología de representación arquitectónica que organiza la descripción de un sistema en cuatro niveles de abstracción —contexto, contenedores, componentes y código— facilitando la comprensión progresiva del sistema según el nivel de detalle requerido [7]. Esta aproximación permite comunicar de manera estructurada la arquitectura del prototipo, especialmente en un proyecto que integra una plataforma web, una API externa y componentes de aprendizaje automático.

Para la identificación de errores comunes, el sistema emplea técnicas de aprendizaje supervisado combinadas con un clasificador basado en reglas, apoyado en una taxonomía de errores construida específicamente para el proyecto a partir del análisis de respuestas reales de estudiantes. De esta manera, se busca no solo automatizar la detección de falencias, sino también estructurar la retroalimentación de acuerdo con categorías conceptuales definidas pedagógicamente.

Este desarrollo no solo busca mejorar el rendimiento académico de los estudiantes mediante el aprendizaje autorregulado, sino también brindar a los docentes una herramienta de apoyo que permita optimizar el proceso de enseñanza.

A continuación, encontrará el documento organizado de la siguiente manera:

- El capítulo 2 presenta el planteamiento del problema.
- El capítulo 3 expone el marco teórico y los trabajos relacionados.
- El capítulo 4 describe la caracterización del conjunto de datos y la taxonomía de errores.
- El capítulo 5 detalla el diseño del sistema inteligente de tutoría.
- El capítulo 6 desarrolla la implementación del prototipo y los resultados del entrenamiento de los modelos de clasificación.
- El capítulo 7 presenta la validación del sistema y los resultados obtenidos.
- Y el capítulo 8 expone las conclusiones y los trabajos futuros.

Capítulo 2

Planteamiento del Problema

2.1. Descripción del problema

En la actualidad, la educación superior enfrenta múltiples desafíos, especialmente en carreras con un alto componente teórico y matemático, como la ingeniería de sistemas y las ciencias de la computación. Estudios recientes han evidenciado que muchos estudiantes experimentan dificultades para comprender los contenidos teóricos en ciencias de la computación, particularmente cuando se trata de conceptos formales o altamente abstractos, debido a altos niveles de exigencia cognitiva y brechas entre los conocimientos previos de los estudiantes y los contenidos impartidos en clase [3]. A lo largo de su formación, estas dificultades pueden generar frustración y desmotivación, e incluso incrementar el riesgo de abandono académico, fenómeno ampliamente documentado en programas de computación [3].

Uno de los principales problemas radica en la falta de adaptación de los métodos de enseñanza a las necesidades individuales de los estudiantes. En muchos contextos universitarios, el modelo predominante continúa siendo la clase magistral, en la cual el docente expone los contenidos a un grupo amplio de estudiantes con escasa interacción con los alumnos. Aunque este modelo permite cubrir temarios extensos, no siempre facilita la identificación temprana de errores conceptuales ni la adaptación a distintos ritmos de aprendizaje [3]. Como resultado, los estudiantes buscan recursos externos para reforzar su aprendizaje, lo que en ocasiones genera discrepancias entre lo que estudian de manera independiente y los objetivos de la asignatura [8].

En este contexto surge la pregunta de cómo apoyar a los estudiantes para que aprendan de manera más efectiva, la cual puede abordarse desde dos aspectos fundamentales:

1. **El método:** ¿qué se debe hacer para aprender de manera efectiva?

2. **El medio:** ¿qué herramientas o tecnologías pueden utilizarse para lograr que el aprendizaje sea efectivo?

Un factor clave que articula ambos aspectos es la retroalimentación. La litera-

tura sugiere que proporcionar retroalimentación inmediata o en tiempo oportuno mejora significativamente la adquisición de conocimientos y habilidades por parte de los estudiantes [5]. En el contexto de este proyecto, la retroalimentación oportuna se entiende como aquella que se proporciona en el momento en que el estudiante la necesita para orientar su proceso de aprendizaje. De acuerdo con la literatura sobre sistemas automatizados de retroalimentación, el momento de entrega constituye un factor determinante para su efectividad, ya que influye tanto en la calidad como en la velocidad con la que el estudiante puede ajustar su comprensión [5]. Dentro de las distintas posibilidades temporales —como la retroalimentación inmediata, a solicitud o al finalizar la tarea—, en este proyecto se prioriza la retroalimentación proporcionada al cierre de la actividad, una vez el estudiante ha completado y enviado su respuesta. Este momento permite que el estudiante disponga de la información necesaria para reflexionar sobre su desempeño, procesar los comentarios recibidos y, si lo considera pertinente, realizar nuevos intentos que favorezcan el fortalecimiento del aprendizaje autorregulado.

No obstante, la cantidad y calidad de la retroalimentación en entornos educativos suelen ser insuficientes debido a la limitación de recursos docentes y a la heterogeneidad de los perfiles de los estudiantes. Investigaciones sobre sistemas automatizados de retroalimentación señalan que, aunque existen múltiples propuestas tecnológicas, todavía persisten retos relacionados con la personalización efectiva, la claridad del mensaje y la alineación pedagógica del feedback [5].

Además, la literatura distingue diferentes formas de retroalimentación, tales como la correctiva, que indica si una respuesta es correcta o incorrecta, y la informativa, que aporta explicaciones adicionales para mejorar la comprensión. Para que la retroalimentación sea pedagógicamente efectiva, debe cumplir ciertos criterios: ser clara, específica, orientada a la tarea, proporcionar información útil para la mejora y evitar la simple entrega directa de la respuesta correcta [9]. Cuando estos criterios no se cumplen, la retroalimentación pierde efectividad y no contribuye de manera significativa al aprendizaje del estudiante.

En este sentido, el uso de tecnologías avanzadas, como los sistemas de retroalimentación automatizada basados en aprendizaje automático, surge como una alternativa para complementar la labor docente, ofreciendo respuestas inmediatas, estructuradas y alineadas con los objetivos de la actividad académica [5].

2.1.1. Formulación del problema

A partir de los elementos expuestos en la descripción del problema, se plantea la siguiente pregunta de investigación: **¿Cómo desarrollar un prototipo de sistema inteligente que ayude a los estudiantes a identificar sus debilidades académicas y brinde retroalimentación para ayudar en su aprendizaje?**

2.1.2. Sistematización del problema

Con el fin de descomponer la pregunta de investigación en elementos abordables, esta sección presenta la sistematización del problema a través de un conjunto de interrogantes que orientan el diseño, desarrollo y validación del sistema propuesto.

- ¿Qué requisitos funcionales son necesarios para un sistema inteligente que identifique falencias académicas y brinde retroalimentación automatizada?
- ¿Cómo diseñar la arquitectura del sistema inteligente mediante el modelo C4 para garantizar la recolección de datos del estudiante, el análisis de respuestas, la identificación de falencias de aprendizaje y la retroalimentación automatizada?
- ¿Cómo implementar un prototipo funcional que analice las respuestas de los estudiantes, identifique errores y brinde retroalimentación automatizada?
- ¿Cómo validar la calidad del módulo de análisis de respuestas y qué hallazgos surgen de las pruebas realizadas en el sistema inteligente?

2.2. Objetivos del Proyecto

2.2.1. Objetivo General

- Desarrollar un prototipo de sistema inteligente que identifique falencias de aprendizaje y brinde retroalimentación continua a estudiantes de Ingeniería de Sistemas y Computación.

2.2.2. Objetivos Específicos

- Identificar los requisitos funcionales para la construcción del prototipo del sistema inteligente de tutoría, definiendo los tipos de datos y los módulos necesarios para detectar las falencias académicas de los estudiantes y generar retroalimentación automatizada.
- Diseñar la arquitectura del prototipo del sistema inteligente de tutoría utilizando el modelo C4, incluyendo los módulos de almacenamiento de datos, análisis de respuestas y generación automatizada de retroalimentación.
- Seleccionar el algoritmo de clasificación más adecuado para la detección de falencias de aprendizaje, a partir de la evaluación y el entrenamiento de al menos dos modelos de aprendizaje supervisado y de la comparación de métricas de desempeño.
- Implementar un prototipo funcional del sistema inteligente de tutoría, basado en el modelo C4, que permita identificar errores básicos, almacenar las respuestas de los estudiantes y generar retroalimentación automatizada.
- Realizar pruebas de aceptación del prototipo y documentar sus resultados, con el fin de evaluar su capacidad de identificar errores, la efectividad de la retroalimentación generada y la percepción de usuario.

2.3. Justificación

La creciente demanda de formación en fundamentos de la programación en entornos universitarios ha resaltado la necesidad de contar con herramientas que ofrezcan retroalimentación automatizada y personalizada para apoyar el aprendizaje de los estudiantes. Estudios recientes han demostrado que los sistemas inteligentes de tutoría tienen un impacto significativo en la mejora del rendimiento académico, al proporcionar asistencia inmediata y adaptada a las necesidades individuales [4]. Estos sistemas no solo permiten identificar errores con precisión, sino que también ofrecen explicaciones detalladas que facilitan su comprensión y corrección.

En el ámbito de la enseñanza de los conceptos necesarios para programar, el uso de retroalimentación automatizada ha mostrado ser especialmente beneficioso para abordar los desafíos que enfrentan los estudiantes al resolver problemas complejos. Investigaciones recientes destacan que la retroalimentación automatizada, cuando se diseña de manera adecuada, puede reducir significativamente las tasas de error y aumentar la retención del conocimiento al proporcionar información oportuna y específica sobre las respuestas incorrectas [6].

El presente proyecto es relevante porque busca desarrollar un sistema inteligente de tutoría que no solo identifique las falencias académicas de los estudiantes, sino que también proporcione retroalimentación continua adaptada al contenido específico del curso de *Introducción a la Programación* de la Pontificia Universidad Javeriana Cali. A diferencia de otros sistemas que suelen apoyarse en formatos cerrados de evaluación, como cuestionarios de opción múltiple, este prototipo se enfoca en el análisis de respuestas abiertas en formato textual, restringidas a un dominio específico, lo que permite una evaluación más flexible, alineada con los contenidos y objetivos del curso.

Además, este proyecto sienta las bases para futuras investigaciones sobre el desarrollo de sistemas de retroalimentación automatizada en el ámbito educativo, destacando la importancia de personalizar la experiencia de aprendizaje mediante tecnologías inteligentes. La implementación del sistema propuesto no solo beneficiará a los estudiantes al proporcionarles asistencia inmediata, sino que también permitirá a los docentes identificar áreas críticas de mejora, optimizando así el proceso de enseñanza-aprendizaje.

2.4. Alcance y Limitaciones

2.4.1. Alcance

El presente proyecto tiene como alcance desarrollar un prototipo funcional de un sistema inteligente de tutoría capaz de identificar falencias académicas en el aprendizaje de la abstracción de sistemas y brindar retroalimentación automatizada. Este sistema permitirá procesar las respuestas de los estudiantes, identificar errores básicos y proporcionar retroalimentación automática con base en los errores detectados. La arquitectura se diseñará siguiendo modelo C4, abarcando los cuatro niveles: el diagrama de contexto, el diagrama de contenedores, el diagrama de componentes y el diagrama de código.

El proyecto se limita a recopilar y documentar los requisitos funcionales del sistema. La base de conocimiento utilizada para evaluar las respuestas de los estudiantes estará constituida por los materiales de la asignatura *Introducción a la Programación* ofrecida por la Pontificia Universidad Javeriana Cali hasta el 2024. Las respuestas al ejercicio serán proporcionadas por el profesor encargado de la materia. Esta delimitación garantiza que el sistema esté alineado con los contenidos específicos del curso y con las necesidades reales de los estudiantes en este contexto educativo.

El proyecto contempla una fase exploratoria en la que se revisarán y evaluarán distintos modelos de clasificación para determinar su viabilidad técnica en función de los requerimientos del sistema. Esta evaluación se basará en criterios como la cantidad limitada de datos disponibles para el entrenamiento, los tipos de errores que se desea clasificar, el formato de las respuestas de los estudiantes y los criterios de evaluación de las tareas. A partir de este análisis, se seleccionarán dos modelos candidatos que se ajusten mejor a dichas condiciones y se procederá a entrenarlos; el modelo con mejor desempeño de clasificación será el que se integre al sistema para la detección de falencias y la generación automatizada de retroalimentación.

En el contexto de este proyecto, las falencias académicas que se pretenden identificar no se abordan como deficiencias cognitivas ni como condiciones clínicas (como la dislexia u otros trastornos del aprendizaje), sino como errores derivados de un proceso de razonamiento incorrecto al resolver la tarea. Esta perspectiva se fundamenta en modelos de aprendizaje que reconocen que el error puede representar una oportunidad para el desarrollo de nuevas comprensiones.

Asimismo, el sistema será validado mediante pruebas de aceptación realizadas por el desarrollador y retroalimentación de un grupo controlado de usuarios del programa de Ingeniería de Sistemas y Computación de la Pontificia Universidad Javeriana Cali, quienes brindarán su percepción sobre la calidad de la retroalimentación generada.

2.4.2. Limitaciones

Tamaño reducido del conjunto de datos

El entrenamiento de los modelos de clasificación supervisada se realizó a partir de un número limitado de respuestas reales de estudiantes. Esta limitación está directamente relacionada con el curso específico seleccionado, los semestres en los que se implementó la actividad, la cantidad de estudiantes matriculados que realizaron la entrega y los periodos académicos en los que el profesor encargado orientó la asignatura. Al tratarse de una actividad diseñada y aplicada por un único docente, el acceso a los datos para el entrenamiento dependió exclusivamente de este contexto, lo que reduce la diversidad de ejemplos disponibles y puede afectar la capacidad de generalización del modelo ante nuevas respuestas.

Falta de homogeneidad del conjunto de datos por variabilidad en los requerimientos de la tarea

El sistema se desarrolló y evaluó para una única actividad del curso. Los requerimientos de la tarea variaron ligeramente entre semestres; en particular, algunos criterios se incorporaron en los semestres posteriores al 2020-2, lo que limita la uniformidad y consistencia del conjunto de datos y reduce la cantidad de respuestas comparables entre periodos. Adicionalmente, los sistemas propuestos para ser abstraídos variaron en cada semestre, lo que restringe aún más la reutilización de respuestas para el entrenamiento de los modelos.

Marco conceptual pedagógico exclusivo para el proyecto

Las falencias académicas abordadas en este proyecto se limitan a errores conceptuales específicos identificados en el contexto de la tarea, derivados de un razonamiento incorrecto durante su resolución. El sistema no contempla la identificación de deficiencias cognitivas, dificultades de aprendizaje de origen clínico ni factores externos que puedan influir en el desempeño del estudiante, lo cual delimita el tipo de retroalimentación que puede ofrecerse.

Limitaciones tecnológicas y de infraestructura

El prototipo fue implementado utilizando servicios con restricciones propias de entornos de bajo costo o planes gratuitos, como el uso de una base de datos en Oracle Cloud Infrastructure bajo la modalidad *free tier*, el consumo limitado de tokens de la API de OpenAI y las restricciones mensuales de Railway para el alojamiento de la API. Estas restricciones condicionan la escalabilidad del sistema, la frecuencia de uso del módulo de retroalimentación automatizada y su eventual despliegue en un entorno productivo con un mayor número de usuarios.

Validación en un contexto controlado

La validación del sistema se realizó mediante pruebas de aceptación realizadas por el desarrollador y la retroalimentación de un grupo controlado de usuarios del programa de Ingeniería de Sistemas y Computación. Si bien este enfoque permite evaluar la utilidad del prototipo en el contexto definido, los resultados obtenidos no pueden generalizarse directamente a otros cursos, instituciones o poblaciones estudiantiles sin hacer ajustes adicionales.

2.5. Metodología

2.5.1. Tipo de Estudio

Este proyecto se enmarca en un estudio de tipo descriptivo y exploratorio. Desde el enfoque descriptivo, se busca caracterizar las falencias académicas asociadas a la tarea de abstracción de sistemas, así como las necesidades de retroalimentación en un contexto educativo específico. Desde el enfoque exploratorio, se aborda el diseño e implementación de un prototipo funcional de un sistema inteligente de tutoría que analice respuestas abiertas de texto y genere retroalimentación automatizada, como una aproximación inicial a soluciones de mayor alcance para el apoyo al aprendizaje.

2.5.2. Estrategia metodológica

La implementación del proyecto se realizó mediante un enfoque incremental, estructurado en etapas alineadas con los objetivos específicos. Cada etapa incorporó revisiones iterativas que permitieron realizar ajustes progresivos al sistema e investigación constante para la toma de decisiones informada. Estas etapas fueron:

- La exploración y el análisis del material académico
- La identificación de requisitos funcionales
- El diseño de la arquitectura del sistema
- El entrenamiento y la selección del algoritmo de clasificación
- El desarrollo del prototipo funcional
- Validación mediante pruebas unitarias y retroalimentación de usuarios.

Capítulo 3

Marco Teórico y Trabajos Relacionados

3.1. Marco de Referencia

Sistema Inteligente

Un sistema inteligente es una entidad basada en computadora capaz de percibir su entorno, razonar sobre sus entradas y ejecutar acciones para alcanzar objetivos específicos. Estos sistemas se caracterizan por integrar componentes funcionales como bases de conocimiento para la representación de información, mecanismos de inferencia para el razonamiento deductivo o inductivo, y capacidades de aprendizaje que permiten la adaptación a condiciones cambiantes. Al imitar aspectos de la inteligencia humana, estos sistemas permiten automatizar el manejo de tareas complejas y deliberativas. En este proyecto, este concepto fundamenta la creación de un agente capaz de analizar respuestas de estudiantes y mejorar su rendimiento de clasificación con el tiempo para ofrecer apoyo en la toma de decisiones pedagógicas [10].

Aprendizaje autorregulado (SRL)

El aprendizaje autorregulado (SRL, por sus siglas en inglés) se refiere al proceso mediante el cual los estudiantes gestionan activamente su propia experiencia de aprendizaje a través de estrategias metacognitivas, motivacionales y conductuales. Este proceso se desarrolla de manera cíclica a través de etapas que incluyen el establecimiento de objetivos, el empleo de tácticas para lograrlos, el monitoreo constante del progreso y el ajuste de los enfoques según sea necesario. El SRL permite a los estudiantes tomar el control de su educación, facilitando una comprensión más profunda y mejores resultados académicos. Para este prototipo, el SRL es el eje central, ya que el sistema busca proporcionar las herramientas necesarias para que el alumno identifique sus propias falencias y regule su ritmo de estudio [2].

Retroalimentación

En los contextos educativos, la retroalimentación es la información proporcionada a los estudiantes sobre su desempeño o comprensión con el fin de guiar sus futuros procesos de aprendizaje. Una retroalimentación efectiva se caracteriza por resaltar las discrepancias entre el rendimiento actual y los objetivos deseados, pudiendo clasificarse en diferentes tipos según su enfoque: de tarea (información sobre el resultado), de proceso (estrategias para realizar la tarea) o de autorregulación (orientada a la autoevaluación del alumno). Este es un componente crítico del aprendizaje que facilita la reflexión y promueve la mejora continua. En la presente investigación, la retroalimentación es el medio principal para cerrar las brechas de conocimiento identificadas, permitiendo una asistencia inmediata que la docencia tradicional no siempre puede cubrir por limitaciones de tiempo [11].

Aprendizaje automático (Machine Learning)

El aprendizaje automático es una rama de la inteligencia artificial que se centra en desarrollar algoritmos capaces de aprender a partir de la experiencia a través de métodos computacionales. Su característica principal es el uso de datos para identificar patrones y realizar predicciones o tomar decisiones sin ser programados explícitamente para cada tarea [12]. Esta disciplina abarca diversos enfoques técnicos, como el aprendizaje supervisado, el no supervisado y el aprendizaje por refuerzo, los cuales utilizan métricas estadísticas para mejorar su rendimiento conforme aumenta la exposición a los datos. En este proyecto, el aprendizaje automático es la base técnica que permite al sistema clasificar de forma autónoma los errores de los estudiantes a partir de ejemplos previos recopilados en semestres anteriores [13].

Sistema adaptativo

Un sistema adaptativo está diseñado para modificar su comportamiento de manera autónoma en respuesta a cambios en su entorno o en su estado interno. Estos sistemas funcionan mediante bucles de retroalimentación constante que monitorean el rendimiento y ejecutan ajustes dinámicos, asegurando una operación óptima incluso en condiciones de alta variabilidad. Se caracterizan por su capacidad de mantener la estabilidad y la eficiencia a través de una respuesta continua a estímulos externos. Para esta investigación, el concepto de sistema adaptativo justifica la necesidad de una herramienta que no sea estática, sino que ajuste la retroalimentación generada según las respuestas específicas y el nivel de comprensión mostrado por cada estudiante [14].

Modelo C4

El modelo C4 es un enfoque de "mapas para el código"diseñado por Simon Brown para describir la arquitectura de los sistemas de software mediante diferentes

niveles de abstracción. Este modelo permite comunicar el diseño técnico de forma clara tanto a equipos de desarrollo como a otros interesados, utilizando diagramas que van desde una visión general hasta el detalle técnico. Se estructura en cuatro niveles jerárquicos: el diagrama de **Contexto** (nivel 1), que muestra el sistema en relación con sus usuarios y otros sistemas; el de **Contenedores** (nivel 2), que descompone el sistema en aplicaciones y almacenes de datos; el de **Componentes** (nivel 3), que detalla los módulos internos de cada contenedor; y el de **Código** (nivel 4), que representa la implementación de las clases o elementos de programación. En este proyecto, el modelo C4 se utilizó para documentar toda la arquitectura del prototipo, permitiendo una separación clara entre la interfaz web en Oracle APEX, la API de clasificación y los servicios externos de IA, asegurando que cada nivel de la solución sea comprensible y escalable [7]

Procesamiento de Lenguaje Natural (NLP)

El procesamiento de lenguaje natural (NLP) es una subárea de la inteligencia artificial que permite a las computadoras entender, interpretar y generar lenguaje humano, ya sea en forma de texto o voz. Esta tecnología combina la lingüística computacional con modelos de aprendizaje automático para analizar la estructura de las oraciones y extraer el significado semántico subyacente. Sus aplicaciones incluyen el análisis de sentimientos, la traducción automática y, fundamentalmente, la extracción de información en sistemas de interacción humano-computador. En el desarrollo de este prototipo, el NLP es la herramienta tecnológica esencial para procesar las respuestas abiertas de los estudiantes, transformando el texto no estructurado en datos que el sistema puede categorizar como errores específicos [15].

TF-IDF (Term Frequency-Inverse Document Frequency)

TF-IDF es un método de vectorización de texto diseñado para representar documentos en un espacio numérico, permitiendo que algoritmos de aprendizaje automático procesen información textual. Esta técnica se caracteriza por asignar un peso a cada término basándose en su frecuencia dentro de un documento (TF) ponderada por su rareza en todo el conjunto de datos (IDF), lo que permite resaltar las palabras más significativas y reducir la importancia de términos comunes o poco informativos. En el contexto de este proyecto, TF-IDF es la técnica fundamental para transformar las respuestas abiertas de los estudiantes en vectores matemáticos comparables, facilitando así la identificación de términos clave asociados a errores conceptuales [16].

N-gramas

Los n-gramas son secuencias contiguas de n elementos, ya sean palabras o caracteres, extraídas de un cuerpo de texto. Su función principal es modelar patrones del lenguaje al capturar relaciones locales y dependencias de contexto entre los ele-

mentos de una secuencia, lo cual no es posible mediante el análisis de palabras individuales. Técnicamente, el uso de bigramas (n=2) o trigramas (n=3) permite al sistema entender el orden y la estructura de las frases empleadas por los alumnos. Para esta investigación, los n-gramas se justifican como una herramienta de precisión semántica, permitiendo que el clasificador diferencie entre conceptos similares según la combinación de palabras utilizada en la respuesta [17].

Tokenización

La tokenización es el proceso de preprocesamiento de texto que consiste en segmentar una secuencia de caracteres en unidades discretas y mínimas llamadas tokens, tales como palabras o sub-palabras. Esta técnica es fundamental para transformar texto no estructurado en una forma organizada que los algoritmos puedan analizar de manera sistemática. Entre sus características técnicas se incluye la normalización y la eliminación de ruido textual para limpiar la entrada de datos. En el desarrollo de este prototipo, la tokenización es el primer paso esencial del flujo de procesamiento, pues permite descomponer las respuestas de los estudiantes en piezas procesables por los modelos de clasificación y el motor de reglas [18].

Dataset o bases de referencia

Un dataset o base de referencia corresponde a un conjunto estructurado de datos que se utiliza para el análisis, entrenamiento y evaluación de modelos de aprendizaje automático. Estos conjuntos suelen organizarse en forma de tablas o colecciones de registros donde cada instancia representa un ejemplo y cada atributo describe una característica relevante del dominio. Su funcionamiento se basa en proporcionar la base de conocimiento necesaria para que los algoritmos identifiquen patrones; para ello, se dividen habitualmente en conjuntos de entrenamiento, destinados al ajuste de parámetros, y conjuntos de prueba, que permiten validar la capacidad de generalización del modelo. En este proyecto, el dataset cumple un rol central al definir la información de las respuestas de los estudiantes a partir de la cual los modelos aprenden las relaciones entre el texto y los errores conceptuales, siendo el insumo principal para alimentar los algoritmos de clasificación supervisada [19].

Regresión Logística

La regresión logística es un modelo estadístico ampliamente utilizado para tareas de clasificación, especialmente en problemas donde la variable objetivo es categórica y binaria. A diferencia de la regresión lineal, este modelo utiliza una función logística o sigmoide para mapear una combinación lineal de características a un valor de probabilidad entre cero y uno, permitiendo decidir a qué clase pertenece una entrada. Dependiendo de la naturaleza de la variable objetivo, este modelo se clasifica principalmente en tres tipos: la regresión logística binaria, utilizada cuando solo existen dos clases posibles (como correcto o incorrecto); la multinomial, em-

pleada para tres o más categorías sin un orden intrínseco; y la ordinal, diseñada para clasificar datos en categorías con una jerarquía lógica. En el desarrollo de este trabajo, este algoritmo se implementó de manera binaria como uno de los modelos base para el entrenamiento, permitiendo evaluar la viabilidad de clasificar las falencias académicas de los estudiantes mediante un enfoque probabilístico que facilita la interpretación de cómo cada término de la respuesta influye en la detección del error [17].

Máquinas de Vectores de Soporte

Las Máquinas de Vectores de Soporte (SVM) son un tipo de modelo de aprendizaje automático diseñado para tareas de clasificación y regresión mediante la identificación de un hiperplano óptimo. Este modelo funciona buscando la frontera de decisión que maximiza el margen de separación entre diferentes grupos de datos, lo que garantiza una mayor robustez ante nuevos ejemplos. Las SVM pueden emplear diferentes funciones de kernel para transformar datos complejos en espacios de mayor dimensión donde la separación sea posible, siendo especialmente efectivas cuando el volumen de datos es limitado. Existen varios tipos de funciones de kernel, como el lineal, que separa datos con una línea recta en dimensiones simples; el polinomial, para relaciones más complejas; y el de Función de Base Radial (RBF), que permite proyectar los datos a espacios de dimensiones infinitas para separar grupos que no son linealmente divisibles en su estado original. En este prototipo, el modelo SVM es el componente encargado del análisis de secciones de propósito y de interacciones de las respuestas, aprovechando su capacidad para trabajar eficazmente con vectores de texto de alta dimensionalidad. Su uso permite detectar con precisión las categorías de error definidas en la taxonomía, garantizando una clasificación robusta incluso ante la cantidad limitada de datos disponibles para el entrenamiento [20].

Sistemas basados en reglas

Los sistemas basados en reglas son métodos tradicionales de inteligencia artificial en los cuales el conocimiento experto se representa mediante un conjunto de reglas lógicas de la forma “IF... THEN...” que permiten derivar conclusiones o decisiones según las condiciones observadas. Estos sistemas operan mediante un motor de inferencia que procesa una base de conocimientos para evaluar datos de entrada y producir resultados consistentes. Existen dos tipos de razonamiento principales: el encadenamiento hacia adelante, que parte de los datos disponibles para alcanzar una conclusión, y el encadenamiento hacia atrás, que parte de un objetivo para verificar los hechos que lo sustentan. En este proyecto, este enfoque es fundamental para el clasificador basado en reglas, el cual valida los componentes, variables y constantes ingresados por los estudiantes al compararlos con una base de referencia. Su uso permite una lógica explícita y una explicación clara de las decisiones, facilitando la identificación de errores deterministas que no requieren de modelos probabilísticos [21].

API

Una API (Application Programming Interface) es un conjunto de reglas y mecanismos que permite la comunicación e interacción entre diferentes sistemas o aplicaciones de software de manera estandarizada. Estas interfaces definen cómo los componentes deben interactuar, especificando los métodos de solicitud, los formatos de respuesta y los protocolos de seguridad para el intercambio de datos. Según su arquitectura, existen diversos tipos como las APIs REST, que utilizan el protocolo HTTP para operaciones sin estado, y las APIs de tipo SOAP o GraphQL, cada una con diferentes niveles de flexibilidad y complejidad. Este enfoque facilita la integración de servicios independientes, promoviendo arquitecturas modulares y escalables donde cada componente cumple una función específica. En el prototipo desarrollado, la API actúa como el puente de comunicación central que encapsula la lógica de análisis y clasificación de errores, permitiendo que la interfaz en Oracle APEX solicite el procesamiento de respuestas y reciba la retroalimentación generada de forma transparente [22].

Infraestructura en la Nube

La infraestructura en la nube hace referencia al conjunto de recursos computacionales, como servidores, almacenamiento y redes, que se ofrecen de manera remota a través de Internet bajo un modelo de uso flexible y bajo demanda. Este paradigma elimina la necesidad de mantener infraestructura física local, permitiendo el acceso a capacidades de procesamiento escalables mediante modelos de servicio como IaaS (Infraestructura como Servicio), PaaS (Plataforma como Servicio) o SaaS (Software como Servicio). Estos entornos se caracterizan por su alta disponibilidad, redundancia de datos y la capacidad de ajustar los recursos según las necesidades de carga del sistema en tiempo real. Para este proyecto, se utilizó infraestructura en la nube para soportar el despliegue de la solución, empleando servicios de Oracle Cloud para la persistencia de datos y Railway para el alojamiento de la API de clasificación. Esto permitió reducir costos operativos y asegurar que el sistema fuera accesible vía web para estudiantes y docentes sin depender de un servidor local [23].

Requerimientos Funcionales

Los requerimientos funcionales describen las funcionalidades, tareas y comportamientos específicos que un sistema debe proporcionar para satisfacer las necesidades de sus usuarios y cumplir con las condiciones esperadas de operación. Estos requisitos representan las interacciones que el software debe soportar bajo circunstancias normales y constituyen la base fundamental para el diseño, la construcción y la validación de un sistema funcionalmente correcto. Se caracterizan por ser verificables, lo que implica que debe existir un proceso de prueba que permita comprobar su cumplimiento, y deben centrarse en el comportamiento externo observable sin entrar en detalles de implementación interna. En el desarrollo de este sistema inteligente de tutoría, se definieron 18 requerimientos funcionales siguiendo el estándar

IEEE 830-1998, los cuales abarcan desde la autenticación de usuarios y la gestión de tareas por parte del profesor, hasta el procesamiento automático de respuestas y la generación de retroalimentación automatizada para el estudiante [24].

Pruebas de Aceptación

Las pruebas de aceptación representan una fase del aseguramiento de la calidad en el desarrollo de software, en la cual se evalúa el sistema desde una perspectiva funcional para verificar que cumple con los requisitos establecidos y con el comportamiento esperado por el usuario final. Este tipo de pruebas se centra en validar que las funcionalidades operen correctamente conforme a las especificaciones definidas, independientemente de su estructura técnica interna. Generalmente, se dividen en pruebas de aceptación de usuario (UAT), realizadas por los clientes finales, y pruebas de aceptación operativa, que aseguran que el sistema esté listo para su despliegue en un entorno productivo. En este proyecto, las pruebas de aceptación fueron fundamentales para validar la capacidad del prototipo para identificar errores conceptuales de manera precisa. Estas pruebas permitieron documentar los hallazgos de integración y evaluar la percepción de los usuarios sobre la utilidad de la retroalimentación automática generada, asegurando que el sistema responda efectivamente a las necesidades pedagógicas planteadas [25].

Plataforma de hosting

La plataforma de hosting en la nube es un servicio que ofrece recursos computacionales y de infraestructura accesibles a través de Internet, donde los servicios de computación se distribuyen dinámicamente entre múltiples servidores virtuales interconectados. A diferencia del hosting tradicional, este modelo permite escalar recursos según la demanda, mejorando la disponibilidad y ofreciendo una mayor flexibilidad para alojar aplicaciones o servicios de software sin necesidad de administrar hardware físico. Existen diversos tipos de hosting en la nube, incluyendo nubes públicas, privadas e híbridas, cada una con distintos niveles de control y seguridad según el tipo de acceso a los recursos. Para la implementación de este prototipo, se utilizaron plataformas de hosting de bajo costo y modalidad "free tier"para garantizar la viabilidad del despliegue. Específicamente, se empleó Railway para el alojamiento de la API de clasificación y procesamiento, facilitando una URL pública que permite la comunicación continua con la interfaz de usuario y los servicios externos de inteligencia artificial [26].

3.2. Trabajos Relacionados y Diferenciadores del Proyecto

3.2.1. Trabajos Relacionados

NLtoFOL: Intelligent Tutoring System for Natural Language to Logic Conversion

En el ámbito de los sistemas inteligentes de retroalimentación, uno de los trabajos más representativos es NLtoFOL, desarrollado por Perikos et al. en 2017. Este sistema de tutoría inteligente fue diseñado para apoyar a los estudiantes en la conversión de lenguaje natural a lógica de primer orden, incorporando un mecanismo de retroalimentación progresiva que guía al estudiante antes y después de la entrega de la solución. A partir de un esquema de categorización, NLtoFOL clasifica las respuestas según su nivel de completitud y precisión, generando retroalimentación que evoluciona de lo general a lo específico. Los resultados reportados evidencian mejoras significativas en el aprendizaje cuando los estudiantes reciben retroalimentación detallada y oportuna [4].

i-SIDRA: Sistema de Retroalimentación Inteligente en Evaluaciones Cerradas

Otro referente relevante es i-SIDRA, propuesto por Fernández-Alemán et al. en 2016, un sistema orientado a la enseñanza de anatomía del sistema locomotor que emplea redes neuronales para analizar respuestas a cuestionarios de opción múltiple. El sistema genera retroalimentación diagnóstica a partir de los errores identificados y permite a los docentes monitorear el progreso de los estudiantes y detectar conceptos que requieren refuerzo adicional. La evaluación realizada con estudiantes de medicina mostró mejoras significativas en el rendimiento académico del grupo que utilizó la herramienta frente a métodos tradicionales [27].

3.2.2. Diferenciadores del Proyecto

A diferencia de los trabajos anteriores, el presente proyecto se enfoca en el análisis automatizado de respuestas abiertas en formato textual, restringidas a un dominio académico específico y a una única tarea de abstracción de sistemas. El sistema desarrollado no evalúa código ni emplea cuestionarios de respuesta cerrada. En su lugar, implementa un enfoque híbrido para la detección de errores, combinando técnicas básicas de procesamiento de lenguaje natural y clasificación supervisada con un clasificador basado en reglas, con el fin de identificar errores conceptuales definidos exclusivamente en el contexto de la tarea analizada. Adicionalmente, el proyecto propone una taxonomía y un marco pedagógico propios para la clasificación de los errores. De este modo, el prototipo se posiciona como una aproximación complemen-

taria a los sistemas existentes, priorizando la coherencia con los contenidos del curso y la generación de retroalimentación automatizada y contextualizada, sin pretender su generalización a otros dominios o actividades académicas.

Capítulo 4

Caracterización del Conjunto de Datos y de los Errores

4.1. Descripción de la Consigna y de las Respuestas para cada Semestre

La exploración del material tuvo como objetivo comprender el contexto de la tarea aplicada en cada semestre, así como identificar similitudes y variaciones en su formulación que pudieran influir en el análisis posterior de las respuestas de los estudiantes. Para ello, se revisaron los enunciados de la actividad cuando estuvieron disponibles y se analizaron las entregas realizadas por los estudiantes, prestando especial atención a los elementos solicitados, el tipo de sistemas propuestos y la forma en que los estudiantes abordaron la tarea de abstracción de sistemas.

Semestre 2020-2

Instrucciones de la actividad

Para el semestre 2020-2 no se contó con el documento oficial en formato PDF que describiera el enunciado de la tarea. En este caso, la reconstrucción del ejercicio se realizó a partir del análisis de las entregas de los estudiantes y de los fragmentos del enunciado que estos incluyeron en sus documentos de respuesta. La tarea consistía en describir un sistema y abstraer sus principales componentes, para luego pasar a identificar variables y constantes. Los sistemas con los que trabajaron los estudiantes fueron:

- Un juego de buscaminas
- Un parqueadero
- El curso de Introducción a la Programación

- Un torneo de algún deporte que le guste

Análisis de las respuestas de los estudiantes

Durante este semestre se registraron 26 entregas en diferentes formatos incluyendo:

- PDFs con imágenes de la resolución escrita a mano
- PDFs con la resolución escrita digitalmente
- Documentos Excel que presentaban la respuesta en una tabla
- Documentos Word con la resolución escrita digitalmente

  Normalmente contenían listas breves de los 3 requerimientos. Se descartaron 2 entregas escritas a mano debido a la ilegibilidad de la resolución. Este semestre, a causa de las instrucciones, supuso un reto importante para el proyecto, que se abarca más adelante en la siguiente sección.

Semestre 2021-1

En el semestre 2021-1 se dispuso del documento oficial de la Tarea 1 del curso *Introducción a la Programación*, el cual se presenta en la Figura 4.1.

Introducción a la Programación 2021-1
Profesor: Carlos Ramírez **Tarea 1**

Esta es la Tarea 1 del curso *Introducción a la Programación*, 2021-1. La actividad es **individual** y sus soluciones deben ser entregadas a través de Discord a más tardar el día **16 de Febrero a las 21:59pm**. En caso de dudas y aclaraciones puede escribir por el canal `#tareas` en el servidor de *Discord* del curso o comunicarse directamente con el profesor y/o el monitor.

Para cada uno de los siguientes sistemas diga cuáles son sus componentes, cuál es su propósito y cómo interactúan sus componentes. Así mismo, indique algunas constantes y variables para cada sistema.

1. Juego de parqués
2. Un concierto de música organizado en el estadio
3. Un salón de clases en un colegio
4. Un partido de fútbol (o de otro deporte que le guste)

Figura 4.1: Enunciado de la Tarea 1 del curso Introducción a la Programación, semestre 2021-1.

Los sistemas propuestos para el análisis fueron distintos, incluían a:

- Un juego de parqués
- Un concierto de música organizado en el estadio
- Un salón de clase
- Un partido de fútbol o de otro deporte qu el guste

Para cada uno se debía de nombrar el propósito del sistema, una lista de componentes del sistema, las interacciones entre los componentes, y una lista de algunas variables y constantes.

Este semestre es particularmente relevante para el proyecto, ya que se recopilaron 29 entregas y las respuestas fueron calificadas por el profesor, incluyendo comentarios breves de retroalimentación. Dichos comentarios permitieron identificar de manera explícita los errores más comunes cometidos por los estudiantes y sirvieron como insumo fundamental para la definición de la taxonomía de errores y para la posterior construcción de los criterios de clasificación utilizados en el sistema.

Semestre 2021-2

Para el semestre 2021-2 también se contó con el documento oficial de la Tarea 1, cuyo enunciado se muestra en la Figura 4.2. En términos generales, la estructura de la actividad se mantuvo similar a la del semestre anterior; sin embargo, los estudiantes parecen haber recibido instrucciones adicionales, puesto que en las respuestas, al momento de derivar constantes y variables, ya lo hacían para cada componente y no para el sistema en general.

Introducción a la Programación 2021-2
Profesor: Carlos Ramírez **Tarea 1**

Esta es la Tarea 1 del curso *Introducción a la Programación*, 2021-2. La actividad es **individual** y sus soluciones deben ser entregadas a través de Discord a más tardar el día **4 de Agosto a las 21:59pm**. En caso de dudas y aclaraciones puede escribir por el canal `#tareas` en el servidor de *Discord* del curso o comunicarse directamente con el profesor y/o el monitor.

Para cada uno de los siguientes sistemas diga cuáles son sus componentes, cuál es su propósito y cómo interactúan sus componentes. Así mismo, indique algunas constantes y variables para cada sistema.

1. Juego de dominó
2. Un punto de vacunación contra el covid-19 en la ciudad
3. El sistema de ascensores de un edificio
4. Una biblioteca

Figura 4.2: Enunciado de la Tarea 1 del curso Introducción a la Programación, semestre 2021-2.

Los sistemas propuestos para el análisis fueron distintos, incluyendo:

- Un juego de dominó
- Un punto de vacunación contra el COVID-19
- El sistema de ascensores de un edificio
- Una biblioteca.

En este semestre se registraron 26 entregas. Aunque el formato de la tarea fue consistente, el cambio en los sistemas a abstraer introdujo nuevas variaciones en las respuestas de los estudiantes. Estas diferencias reforzaron la necesidad de acotar el análisis a un dominio y a una tarea específica, así como de definir errores conceptuales dependientes del contexto de la actividad.

4.2. Formulario Pedagógico Aplicado

Con el fin de comprender la naturaleza de la tarea propuesta a los estudiantes y anticipar los posibles tipos de errores que podían presentarse en su resolución, se diseñó y aplicó un formulario de análisis previo. Este instrumento permitió identificar las dificultades conceptuales y de interpretación más recurrentes asociadas a la actividad de abstracción de sistemas.

El formulario fue desarrollado por Marcela Valencia Serrano, doctora en Psicología y especialista en psicología educativa, quien aportó un enfoque pedagógico fundamentado para la investigación. Posteriormente, se aplicó el formulario con el fin de recopilar información relevante sobre la tarea y los errores observados en los estudiantes. A continuación, se presentan la estructura del formulario y los resultados obtenidos.

Cuadro 4.1: Preguntas y respuestas del formulario de análisis de la tarea

Pregunta	**Respuesta**
¿En qué consiste la tarea?	La tarea consiste en nombrar y analizar los componentes que hacen parte de un sistema, identificar cuál es el propósito del sistema y cómo interactúan entre sí los componentes o cómo un componente afecta a otro. Adicionalmente se solicita nombrar una lista de variables y constantes de los sistemas mencionados.
¿Qué se espera que logren los estudiantes?	Se espera que los estudiantes desarrollen la capacidad de abstracción de sistemas, identificando correctamente los elementos relevantes y su función dentro del sistema.
¿Qué exigencias cognitivas les hace la tarea a los estudiantes?	La tarea exige que los estudiantes describan el sistema, analicen sus elementos, evalúen su relevancia y construyan una representación estructurada del mismo.
¿Qué conocimientos necesita el estudiante para resolver la tarea?	El estudiante debe comprender los conceptos de sistema, variable, constante e instancia de un sistema, así como la estructura básica de un sistema en términos de entradas, procesos y salidas.
¿Cuáles serían los pasos ideales que debe seguir el estudiante para resolver la tarea?	El estudiante debe leer y comprender el enunciado, reconocer o investigar el sistema, identificar sus componentes principales y analizar distintas instancias del sistema para clasificar adecuadamente variables y constantes.
¿Cuáles serían posibles errores en los que puede incurrir el estudiante al ejecutar la tarea?	El estudiante puede no comprender la tarea solicitada, confundir componentes con variables o constantes, presentar descripciones ambiguas o clasificar incorrectamente la información del sistema.

4.3. Marco Pedagógico para el Proyecto

El área de la psicología educativa ha estudiado durante décadas el papel del error dentro del proceso de aprendizaje. A partir de estos estudios, se han propuesto distintas taxonomías que permiten clasificar los errores de los estudiantes según su origen y su función cognitiva. Por ejemplo, en un estudio [28] se distingue entre errores que surgen de un aprendizaje incompleto o impreciso, los cuales se corrigen mediante el enriquecimiento del conocimiento existente, y aquellos que derivan de una comprensión errónea de un concepto fundamental, que requieren un proceso de reestructuración conceptual más profundo. De manera complementaria, en otro artículo [29] señalan que los errores de los aprendices pueden originarse tanto en deficiencias del conocimiento previo (knowledge gaps) como en concepciones erróneas persistentes (misconceptions), enfatizando que la identificación de la causa del error es esencial para diseñar estrategias efectivas de retroalimentación y enseñanza.

Desde el área de la educación en programación, diversos autores han evidenciado que los cursos introductorios presentan altos niveles de dificultad, lo que ha motivado investigaciones orientadas a detectar y clasificar los errores más comunes de los estudiantes. En este sentido, en una importante investigación [30] se identifican una serie de errores recurrentes en la enseñanza de los principios básicos de programación, como la confusión entre operadores aritméticos y relacionales, la mala selección de estructuras de control o la incorrecta inicialización de variables, los cuales reflejan falta de comprensión conceptual o aprendizaje incompleto de los fundamentos del lenguaje. Los autores concluyen que estos errores no solo afectan el rendimiento académico, sino que evidencian dificultades en la construcción de modelos mentales correctos, lo que resalta la importancia de ofrecer retroalimentación oportuna y dirigida para guiar al estudiante en la corrección de sus falencias cognitivas.

Para este proyecto, debido a la necesidad de clasificar los errores de los estudiantes según la naturaleza de sus fallos en la tarea propuesta, se definió una taxonomía propia compuesta por dos niveles: un nivel 1 con tres categorías de origen cognitivo y un nivel 2 con seis subcategorías de error.

Las tres categorías generales (nivel 1) que se definieron para este trabajo se fundamentaron en la integración de enfoques teóricos propuestos por distintos autores, entre ellos Chi y Reason [28] [31]. Mientras que las seis subcategorías de error (nivel 2) se asociaron directamente al contexto de la tarea, siendo identificadas a partir del análisis de las respuestas de los estudiantes y de la retroalimentación proporcionada por el profesor en las entregas del semestre 2021-1.

4.4. Taxonomía de los Errores para el Proyecto

4.4.1. Categorías de Error (Nivel 1)

Las categorías de aprendizaje de nivel 1 permiten identificar el origen cognitivo del error, es decir, la causa subyacente por la cual el estudiante presenta dificultades al resolver una tarea. Estas categorías no describen el error observable en sí mismo, sino el tipo de aprendizaje que lo origina, en concordancia con los enfoques de análisis de errores y cambio conceptual en educación.

1. **Aprendizaje incompleto o impreciso**

   En este tipo de error el estudiante posee un conocimiento parcial del concepto, pero este no es suficiente o no está correctamente consolidado, lo que lo lleva a aplicarlo de manera incorrecta o incompleta. El estudiante reconoce el término o la idea general, pero presenta vacíos conceptuales (*knowledge gaps*) que afectan su desempeño.

   El origen cognitivo más común de este error es una exposición insuficiente al concepto o la falta de variedad de ejemplos que permitan comprenderlo en distintos contextos. Como consecuencia, el estudiante construye una representación frágil del concepto, que funciona solo en situaciones muy específicas.

   Este tipo de error puede superarse mediante la introducción de nueva información y el refuerzo conceptual, ya sea presentando más ejemplos y contraejemplos que clarifiquen la diferencia entre conceptos como variable y constante, o mostrando cómo se comportan en distintos estados del sistema.

2. **Comprensión errónea del concepto**

   En este caso, el estudiante construye un modelo mental incorrecto que considera válido. En otras palabras, cree comprender el concepto, pero su interpretación es errónea.

   El origen cognitivo de este error radica en una conceptualización inicial incorrecta que no ha sido cuestionada ni corregida. Por ejemplo, un estudiante puede creer que las constantes son los componentes del sistema que no cambian, cuando en realidad una constante es la información asociada a un componente que permanece fija una vez se le asigna un valor. Este tipo de error es más profundo que el aprendizaje incompleto, ya que el estudiante no percibe la necesidad de modificar sus conocimientos.

   Superar este error requiere un proceso de reestructuración conceptual, en el cual el estudiante confronte explícitamente su idea previa con la definición correcta. Estrategias como la comparación de conceptos, el uso de ejemplos contradictorios y la retroalimentación guiada son fundamentales para promover este cambio conceptual.

3. **No comprensión de las instrucciones**

   En este caso, el estudiante no logra interpretar correctamente lo que se le solicita en la tarea. Como resultado, desarrolla una respuesta que no corresponde

a los requerimientos planteados, independientemente de que cuente con los conocimientos necesarios.

El origen cognitivo más común de este error se encuentra en dificultades de comprensión lectora, atención limitada al enunciado o una interpretación superficial de la tarea. El estudiante se enfoca en describir el sistema de forma general, pero omite analizar o clasificar los elementos según las categorías solicitadas.

Este tipo de error puede superarse mediante la clarificación de las instrucciones, el uso de ejemplos de respuestas esperadas y el acompañamiento inicial que ayude al estudiante a identificar explícitamente qué debe hacer en cada parte de la tarea.

4.4.2. Subcategorías de Error (Nivel 2)

A partir del análisis detallado de las respuestas de los estudiantes y de la retroalimentación proporcionada por el profesor en el semestre 2021-1, se definieron seis subcategorías de error asociadas directamente al contexto de la tarea de abstracción de sistemas. Cada subcategoría describe un error observable en las respuestas y se vincula con una categoría de error de nivel 1, de acuerdo con su origen cognitivo.

1. **Componentes escritos como variables o constantes**

   En este tipo de error, el estudiante clasifica como variables o constantes elementos que en realidad corresponden a componentes del sistema, confundiendo el objeto con la información asociada a dicho objeto. Por ejemplo, en el sistema del salón de clase, algunos estudiantes identificaron como constantes elementos como *"sillas"*, *"tablero"* o *"escritorios"*. De manera similar, en el sistema de un concierto en un estadio, se clasificaron como constantes elementos como *"el estadio"*, *"la música"* o *"los instrumentos"*.

   Frente a este tipo de respuestas, el profesor proporcionó retroalimentación como: *"Cuando se habla de constantes y variables, se habla de información sobre los componentes"* y *"Deben especificarse como información"*. Este error se asocia a la categoría de **comprensión errónea del concepto**, ya que el estudiante construye un modelo mental incorrecto sobre la naturaleza de las variables y constantes dentro de un sistema.

2. **Datos de componentes escritos como componentes**

   En esta subcategoría, el estudiante identifica como componentes datos que corresponden a variables o constantes asociadas a un componente. Un ejemplo de este error se presenta cuando, en el sistema del concierto en un estadio, un estudiante incluyó como componente la *"capacidad del estadio"*, la cual corresponde en realidad a un dato constante del componente *estadio*.

Este error se clasifica dentro de la categoría de **comprensión errónea del concepto**, dado que el estudiante no diferencia adecuadamente entre los elementos estructurales del sistema y la información que los describe, evidenciando una conceptualización incorrecta del sistema.

3. **Variables o constantes no esenciales para el sistema**

   Esta subcategoría agrupa los casos en los que el estudiante identifica variables o constantes que, si bien pueden cambiar o mantenerse estables, no son esenciales para el funcionamiento del sistema que se está abstrayendo. Por ejemplo, en el sistema de un salón de clases, se incluyeron como variables elementos como *"la información escrita en el tablero"*, *"la disposición de cada persona para aprender"*, *"la manera didáctica"*, *"el ánimo"* o *"el profesionalismo del docente"*. Aunque estos aspectos pueden variar, no constituyen elementos fundamentales para modelar el sistema desde el nivel de abstracción solicitado.

   Este error se asocia a la categoría de **aprendizaje incompleto o impreciso**, ya que el estudiante comprende qué es una variable o una constante, pero presenta dificultades al abstraer el sistema y seleccionar únicamente los elementos esenciales, sin considerar que no es posible modelar un sistema real incluyendo todas sus características.

4. **Falta de información**

   En este tipo de error, el estudiante proporciona una respuesta que no incluye la información necesaria para cumplir con lo solicitado en la tarea, lo que impide comprender adecuadamente el planteamiento realizado. Por ejemplo, en el sistema del concierto, un estudiante definió el propósito como *"Compartir música y disfrutar de ella"*, ante lo cual el profesor comentó: *"Demasiado general. No es claro cuál es el rol de los componentes"*. En otro caso, para el sistema del juego de parqués, un estudiante escribió como propósito *"Superar los obstáculos del tablero y enemigos con todas las fichas y ganar"*, y el profesor señaló específicamente la palabra *"ganar"* con la pregunta: *"¿Y cómo se gana?"*. En ambos casos, aunque existe una intención de responder, la información proporcionada resulta insuficiente para describir el propósito del sistema de acuerdo con los criterios de la tarea.

   Este error se asocia a la categoría de **no comprensión de las instrucciones**, ya que el estudiante no desarrolla el nivel de detalle requerido, omitiendo información explícitamente solicitada en el enunciado.

5. **Ambigüedad en la respuesta**

   La ambigüedad se presenta cuando la respuesta del estudiante no permite comprender con claridad a qué se refiere, debido a formulaciones vagas o poco precisas. Por ejemplo, en el sistema de un partido de fútbol, un estudiante incluyó como variable *"las ocasiones de gol"*, sin que fuera claro si se refería a las oportunidades de gol, a los intentos realizados o a los goles efectivamente anotados. Esta falta de precisión dificulta la interpretación del dato y su correcta clasificación dentro del sistema.

Este tipo de error se asocia a la categoría de **no comprensión de las instrucciones**, dado que, aunque el estudiante puede tener una idea general del concepto, no logra expresarla de forma clara ni detallada según lo solicitado, generando respuestas ambiguas que no permiten evaluar correctamente el sistema abstraído.

6. **Confusión entre variables y constantes**

   Este error ocurre cuando el estudiante clasifica incorrectamente una variable como constante o una constante como variable. Por ejemplo, en el sistema de un partido de fútbol, un estudiante identificó como variable las posiciones en las que jugaría cada jugador, cuando en el contexto planteado estas posiciones permanecen fijas durante el desarrollo del partido y deberían considerarse constantes.

   En este proyecto, la confusión entre variables y constantes se clasifica principalmente como un error de **aprendizaje incompleto o impreciso**. El estudiante reconoce las definiciones básicas de variable y constante, pero no logra abstraer correctamente el sistema ni plantear escenarios que le permitan identificar el comportamiento de los datos a lo largo del tiempo.

4.4.3. Tabla de Resumen de la Taxonomía

Cuadro 4.2: Categorías y subcategorías de error identificadas en la tarea

Categoría del error	**Subcategoría del error**	**Código del error**
Comprensión errónea del concepto	Componentes escritos como variables o constantes	`COMPONENT_AS_DATA`
Comprensión errónea del concepto	Datos de componentes escritos como componentes	`DATA_AS_COMPONENT`
Aprendizaje incompleto o impreciso	Variables o constantes no esenciales para el sistema	`NON_ESSENTIAL_-ABSTRACTION`
No comprensión de las instrucciones	Falta de información	`INSUFFICIENT_INFORMATION`
No comprensión de las instrucciones	Ambigüedad en la respuesta	`AMBIGUOUS_DESCRIPTION`
Aprendizaje incompleto o impreciso	Confusión entre variables y constantes	`VARIABLE_AS_CONSTANT / CONSTANT_AS_VARIABLE`

Capítulo 5

Diseño del Sistema Inteligente de Tutoría

5.1. Requerimientos Funcionales

Para la redacción de los requisitos funcionales de un proyecto de software no existe un estándar universal; sin embargo, a lo largo del tiempo se han propuesto buenas prácticas y distintos marcos orientados a mejorar su claridad y consistencia. En el contexto de este proyecto, se optó por seguir las directrices del estándar IEEE 830-1998, dado que es un referente ampliamente reconocido en el mundo de la ingeniería que reúne múltiples prácticas consolidadas, de las cuales se adoptaron las consideradas más relevantes y adecuadas para el alcance del sistema desarrollado [32].

En particular, para la definición de los requerimientos funcionales del sistema, se tuvieron en cuenta los siguientes lineamientos:

1. **Identificación única**: Todo requisito debe ser identificable de manera unívoca mediante un código o sistema de numeración, lo que facilita su seguimiento, referencia y trazabilidad a lo largo del ciclo de desarrollo.
2. **Uso de lenguaje natural y fórmulas específicas**: Los requisitos deben expresarse como acciones que el sistema realizará, utilizando formulaciones claras como "el sistema deberá...". Aunque pueden emplearse tablas o representaciones gráficas, estas deben complementar y no sustituir la descripción en lenguaje natural.
3. **Evitar la ambigüedad**: Cada requisito debe admitir una única interpretación. Para ello, se recomienda emplear términos precisos, evitar palabras vagas como "adecuado" o "a veces" y, cuando sea necesario, apoyarse en definiciones claras o notaciones formales.

4. **Verificabilidad**: Un requisito funcional es válido únicamente si puede ser comprobado. Esto implica que debe existir un proceso de prueba finito y razonable que permita verificar si el sistema cumple con lo especificado.

5. **Enfoque en el comportamiento externo**: Los requerimientos deben describir comportamientos observables por los usuarios u otros sistemas, sin entrar en detalles sobre la implementación interna o las decisiones de diseño.

6. **Consistencia interna**: El conjunto de requisitos no debe contener contradicciones, ya que especificaciones incompatibles entre sí impiden una implementación correcta del sistema.

7. **Nivel de detalle suficiente**: La redacción debe proporcionar el grado de detalle necesario para que los diseñadores puedan construir el sistema y el equipo de pruebas pueda planificar y ejecutar las verificaciones correspondientes.

8. **Organización lógica**: Los requerimientos funcionales deben estructurarse de manera coherente, ya sea por objetivos, tipos de usuario, estímulos o jerarquías funcionales, de acuerdo con las necesidades del proyecto.

9. **Legibilidad**: El documento debe ser comprensible para lectores con distintos niveles de formación técnica, asegurando que los requerimientos puedan ser interpretados correctamente por todos los interesados en el desarrollo del sistema.

5.1.1. Listado de Requerimientos

Con base a los lineamientos anteriores, se desarrollaron los siguientes requisitos funcionales para el proyecto:

1. **RF-01. Autenticación de usuarios**: El sistema deberá permitir el acceso de estudiantes y profesores mediante autenticación con usuario y contraseña.

2. **RF-02. Gestión de roles y permisos**: El sistema deberá diferenciar las funcionalidades disponibles para estudiantes y profesores, restringiendo el acceso a la información y acciones según el rol asignado.

3. **RF-03. Registro y gestión de tareas**: El sistema deberá permitir que el profesor registre una tarea académica para su análisis, asociándola a un sistema o dominio definido dentro de la plataforma.

4. **RF-04. Carga y visualización del enunciado**: El sistema deberá permitir que el profesor cargue el archivo PDF con las especificaciones de la tarea y deberá permitir su visualización desde la plataforma.

5. **RF-05. Diligenciamiento de respuestas por secciones**: El sistema deberá permitir que el estudiante diligencie su respuesta mediante campos de texto separados por cada sección solicitada en la tarea.

6. **RF-06. Consulta del enunciado durante el diligenciamiento**: El sistema deberá permitir que el estudiante visualice el PDF de la tarea mientras diligencia su respuesta.

7. **RF-07. Envío y almacenamiento de respuestas**: El sistema deberá permitir que el estudiante envíe su respuesta a la tarea y deberá almacenar la información enviada para su consulta posterior.

8. **RF-08. Procesamiento de respuestas**: El sistema deberá procesar las respuestas enviadas por los estudiantes para identificar falencias académicas en el contexto de la tarea definida.

9. **RF-09. Identificación y clasificación de errores**: El sistema deberá identificar errores en la respuesta del estudiante y clasificarlos por categoría y subcategoría de acuerdo con la taxonomía definida para el proyecto.

10. **RF-10. Registro del valor asociado al error**: El sistema deberá registrar, para cada error detectado, el fragmento o valor de la respuesta del estudiante que fue identificado como erróneo.

11. **RF-11. Generación de retroalimentación por error**: El sistema deberá generar retroalimentación automatizada por cada error detectado en la respuesta del estudiante, manteniendo una relación de uno a uno (un error $\rightarrow$ una retroalimentación).

12. **RF-12. Visualización de resultados al estudiante**: La plataforma web deberá mostrar al estudiante los errores detectados en su respuesta, incluyendo su categoría, subcategoría, el valor asociado al error y la retroalimentación generada para cada caso.

13. **RF-13. Consulta de respuestas del estudiante**: La plataforma web deberá permitir al estudiante consultar el historial de todas las respuestas que ha enviado y visualizar únicamente información asociada a su cuenta.

14. **RF-14. Consulta de respuestas por parte del profesor**: La plataforma web deberá permitir al profesor consultar las respuestas enviadas por los estudiantes asociadas a la tarea registrada.

15. **RF-15. Registro de errores y retroalimentación**: El sistema deberá registrar un historial de los errores detectados y de la retroalimentación generada para cada respuesta procesada, con el fin de permitir su consulta posterior.

16. **RF-16. Acceso del estudiante al historial de errores**: La plataforma web deberá permitir al estudiante consultar el historial de errores y retroalimentación asociados únicamente a sus propias respuestas.

17. **RF-17. Acceso del profesor al historial de errores**: La plataforma web deberá permitir al profesor consultar el historial de errores y retroalimentación correspondientes a los estudiantes y respuestas asociadas a la tarea.

18. **RF-18. Página de ayuda y descripción funcional**: La plataforma web deberá incluir una sección de ayuda que describa de forma clara las funcionalidades principales del sistema y el flujo de uso para estudiantes y profesores.

5.2. Diseño de la arquitectura

El diseño de la arquitectura del prototipo se documenta utilizando el modelo C4, un enfoque de descripción arquitectónica , el cual permite representar sistemas de software a distintos niveles de abstracción [7]. Este modelo facilita una comprensión progresiva de la solución, iniciando con una visión general del sistema y avanzando gradualmente hacia representaciones más cercanas a su implementación.

El modelo C4 se compone de cuatro niveles: **Contexto**, **Contenedores**, **Componentes** y **Código**. Cada uno de estos niveles aporta una perspectiva complementaria de la arquitectura, permitiendo describir el sistema sin sobrecargar los diagramas con detalles innecesarios. En este proyecto, el modelo C4 resulta adecuado debido a la integración de múltiples tecnologías y servicios, así como a la necesidad de separar claramente la interfaz de usuario de los procesos automáticos de análisis y generación de retroalimentación.

A continuación, se describen los diagramas correspondientes a cada uno de los niveles del modelo C4 utilizados en el diseño de la arquitectura del prototipo.

5.2.1. Diagrama de contexto del sistema (Nivel 1)

La Figura 5.1 presenta el diagrama de contexto del sistema, el cual muestra una visión general de la solución y delimita claramente su alcance. En este nivel se identifican los actores externos y los sistemas con los que interactúa la plataforma, sin detallar aspectos técnicos ni de implementación.

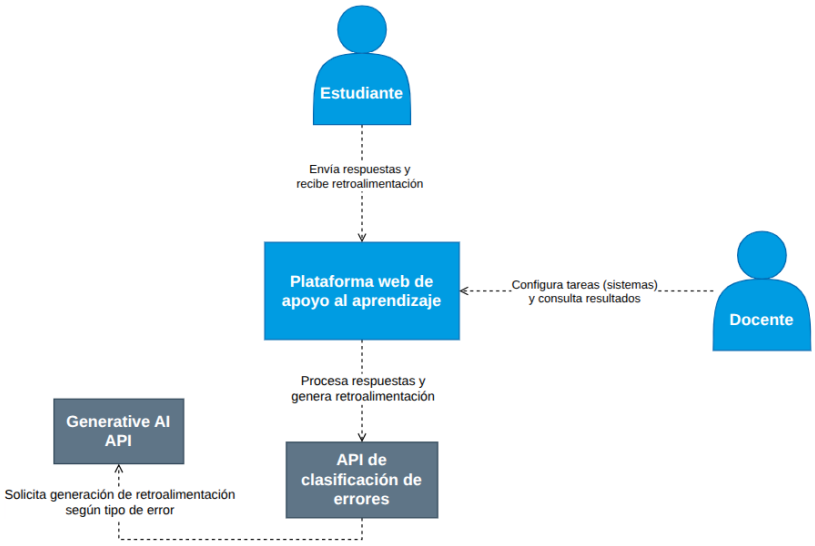

Figura 5.1: Diagrama de contexto del sistema (Nivel 1)

Como se observa en la figura, el sistema es utilizado por dos actores principales: **estudiantes** y **docentes**. Los estudiantes interactúan con la plataforma web para enviar sus respuestas a la tarea y recibir retroalimentación automática sobre su desempeño. Los docentes, por su parte, utilizan la plataforma para configurar las tareas asociadas a distintos sistemas y consultar los resultados obtenidos por los estudiantes.

La plataforma web de apoyo al aprendizaje actúa como el punto central de interacción y se comunica con una API de clasificación de errores, la cual se encarga de procesar las respuestas enviadas por los estudiantes y generar la retroalimentación correspondiente. A su vez, esta API hace uso de un servicio externo de inteligencia artificial generativa (OpenAI API) para la generación del texto de retroalimentación, el cual se considera externo al sistema principal.

5.2.2. Diagrama de contenedores (Nivel 2)

La Figura 5.2 muestra el diagrama de contenedores, en el cual se describe la estructura tecnológica del sistema y se identifican las principales aplicaciones y servicios que lo conforman.

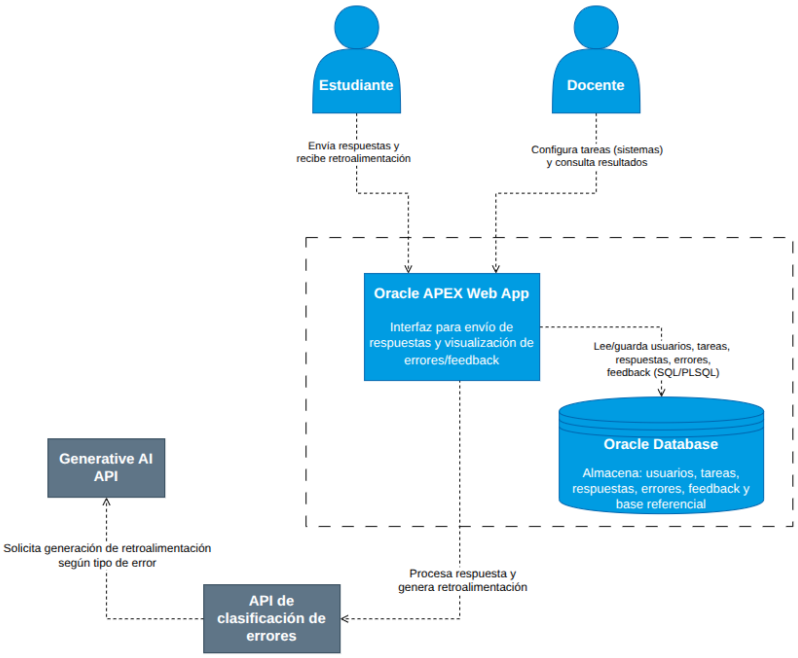

Figura 5.2: Diagrama de contenedores (Nivel 2)

En este nivel se evidencia la separación entre tres contenedores principales. El primero corresponde a la **aplicación web desarrollada en Oracle APEX**, la cual proporciona la interfaz para el envío de respuestas y la visualización de errores y retroalimentación. Este contenedor también se encarga de la interacción con los usuarios y de la orquestación de los procesos de análisis y retroalimentación.

El segundo contenedor corresponde a la **API de clasificación de errores**, implementada como un servicio independiente. Esta API encapsula la lógica de análisis de las respuestas, incluyendo la detección de errores conceptuales y la generación de retroalimentación automática. La comunicación entre la plataforma web y la API se abstrae como un único flujo de procesamiento de respuestas y generación de retroalimentación.

Finalmente, el tercer contenedor corresponde a la **base de datos Oracle**, utilizada para almacenar la información del sistema, incluyendo usuarios, tareas, respuestas, errores detectados y retroalimentación generada. Esta separación permite desacoplar la lógica de negocio de la persistencia de los datos.

5.2.3. Diagrama de componentes (Nivel 3)

La Figura 5.3 presenta el diagrama de componentes, el cual detalla la estructura interna de la **API de clasificación de errores** y los módulos que la conforman.

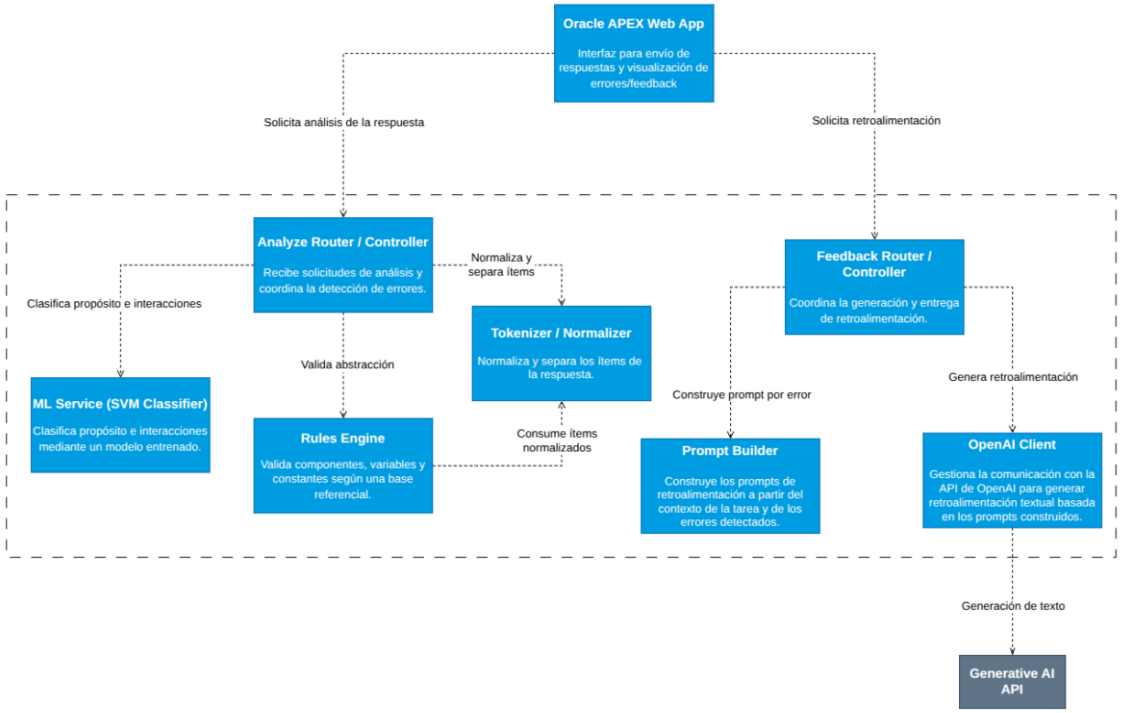

Figura 5.3: Diagrama de componentes (Nivel 3)

Como se observa en la figura, la API se organiza en dos flujos principales: el análisis de las respuestas y la generación de retroalimentación. El análisis es coordinado por el componente **Analyze Router / Controller**, el cual recibe las solicitudes desde la plataforma web y delega el procesamiento a los servicios especializados.

Para las secciones de propósito e interacciones, el análisis se realiza mediante el componente **ML Service (SVM Classifier)**, el cual utiliza un modelo de aprendi-

zaje automático entrenado previamente. En paralelo, la validación de componentes, variables y constantes se realiza a través del componente **Rules Engine**, el cual compara los ítems ingresados por el estudiante contra una base referencial. Para este proceso, el componente **Tokenizer / Normalizer** se encarga de normalizar y separar los ítems antes de su evaluación.

La generación de retroalimentación se gestiona mediante el componente **Feedback Router / Controller**, el cual coordina la construcción de los mensajes de retroalimentación a partir de los errores detectados. Este componente delega la creación de los prompts al **Prompt Builder** y la generación del texto al **OpenAI Client**, encargado de comunicarse con el servicio externo de inteligencia artificial generativa.

5.2.4. Diagrama de código (Nivel 4)

La Figura 5.4 muestra el diagrama de código, el cual representa la organización concreta del código fuente de la API de clasificación de errores y sus dependencias principales, reflejando directamente la implementación real del prototipo.

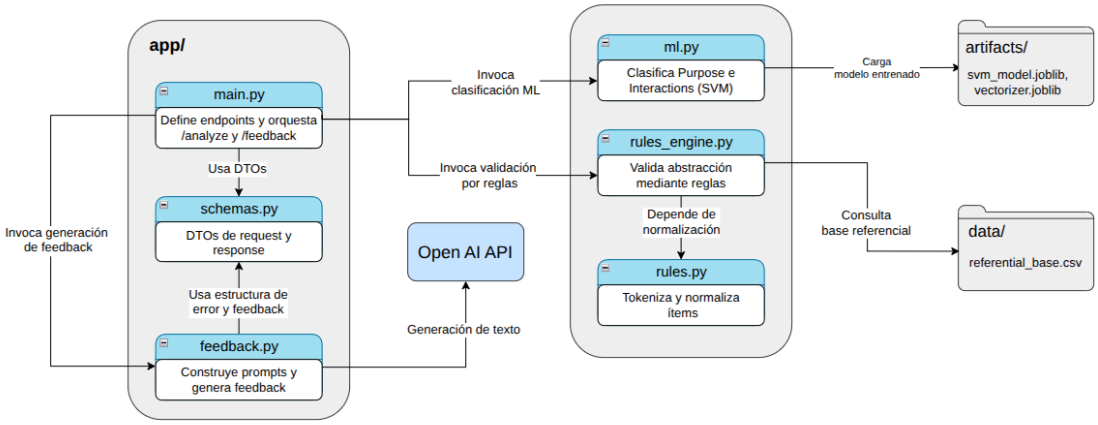

Figura 5.4: Diagrama de código (Nivel 4)

El código se organiza en módulos que corresponden directamente a los componentes definidos en el nivel anterior. El archivo `main.py` define los endpoints y orquesta los flujos de análisis y retroalimentación, apoyándose en `schemas.py` para la estructuración de los datos de entrada y salida. El módulo `ml.py` implementa el servicio de clasificación mediante aprendizaje automático y gestiona la carga de los modelos entrenados desde el directorio de artefactos.

La validación basada en reglas se implementa en `rules_engine.py`, el cual depende de los procesos de normalización definidos en `rules.py` y de la base referencial almacenada en el directorio de datos. Finalmente, la generación de retroalimenta-

ción se concentra en `feedback.py`, donde se construyen los prompts y se gestiona la comunicación con el servicio externo de inteligencia artificial generativa.

Este nivel permite establecer una trazabilidad directa entre el diseño arquitectónico y la implementación del prototipo, mostrando cómo las decisiones de diseño se reflejan en la estructura real del código.

Capítulo 6

Desarrollo del Prototipo

Este capítulo describe en alto nivel las implementaciones realizadas y su proceso, sin profundizar en detalles de código. La implementación completa del sistema puede ser consultada en el repositorio del trabajo de grado [33]

6.1. Proceso de Construcción del Primer Dataset

El primer conjunto de datos se construyó a partir de la extracción manual de las respuestas entregadas por los estudiantes en la actividad de abstracción de sistemas. Este dataset tenía como objetivo inicial servir como base para la identificación y clasificación de errores conceptuales presentes en las respuestas, así como para el posterior entrenamiento de modelos de clasificación. El diseño del conjunto de datos se realizó sin considerar el alto número de errores a clasificar, lo que influyó en el diseño de la solución y propuso cambios importantes. Estas decisiones se explican en la secciones posteriores del capítulo.

Es importante señalar que, para las tareas correspondientes a los semestres 2020-2 y 2021-1, únicamente se consideraron tres sistemas por cada actividad. Esta decisión se tomó debido a que el cuarto sistema era de libre elección por parte de los estudiantes, quienes podían proponer un sistema de su preferencia (por ejemplo, un torneo de un deporte de interés personal). Dado que estas respuestas no compartían un contexto común ni criterios homogéneos de evaluación, su inclusión resultaba poco adecuada para la construcción del dataset, por lo que fueron excluidas del conjunto de datos. Los sistemas que se terminaron seleccionando para la clasificación de las respuestas fueron los siguientes:

1. Un juego de dominó.
2. Un punto de vacunación contra el COVID-19.
3. El sistema de ascensores de un edificio.
4. Un juego de parqués.

5. Un concierto de música organizado en un estadio.
6. Un salón de clases.
7. Un juego de buscaminas.
8. Un parqueadero.
9. El curso de Introducción a la Programación.

A continuación, se presenta la estructura del dataset original, describiendo el propósito de cada una de sus columnas:

Cuadro 6.1: Descripción del dataset

Nombre de la columna	**Descripción**
SEMESTER	Semestre de la tarea a la cual pertenece el sistema
SYSTEM_ID	Identificador del sistema analizado por el estudiante (por ejemplo, biblioteca, parqueadero o concierto), utilizado para contextualizar la respuesta.
RESPONSE_CATEGORY_ID	Categoría de la respuesta evaluada, correspondiente a las secciones de la actividad (Purpose, Components, Variables, Constants o Interactions).
ANSWER_ID	Identificador númerico de la respuesta
ANSWER_TEXT	Texto original escrito por el estudiante como respuesta a la sección correspondiente de la actividad.
ERROR_CATEGORY_ID	Categoría de error nivel 1 según la taxonomía del proyecto.
ERROR_SUBCATEGORY_ID	Subcategoría de error asignada manualmente a la respuesta, utilizada para indicar el tipo de falencia identificada o si la respuesta era correcta.
FEEDBACK	La retroalimentación proporcionada por el docente para el error.

6.1.1. Métricas para el dataset

Para la construcción de este primer dataset se revisaron de manera manual las respuestas de los estudiantes, extrayendo un total de 320 respuestas individuales. Cada respuesta fue analizada y clasificada manualmente con el fin de identificar si correspondía a una respuesta correcta o si presentaba algún tipo de error conceptual. Este proceso permitió asignar etiquetas de error basadas en la taxonomía definida para el proyecto, pero implicó un alto grado de intervención manual y subjetividad en la clasificación.

Las siguientes tablas muestran el resultado de las consultas que se hicieron al dataset para contar la cantidad de respuestas y su distribución.

Cuadro 6.2: Total de filas por sistema

SYSTEM_ID	**TOTAL_FILAS**
ASCENSORES EN UN EDIFICIO	20
BIBLIOTECA	29
BUSCAMINAS	13
CONCIERTO EN UN ESTADIO	45
CURSO INTRO PROG	21
DOMINO	43
PARQUEADERO	23
PARQUES	57
PUNTO DE VACUNACION DE COVID-19	18
SALON DE CLASES DE UN COLEGIO	51

Cuadro 6.3: Total de filas por cada subcategoría de error

ERROR_SUBCATEGORY_ID	**TOTAL_FILAS**
AMBIGUOUS_DESCRIPTION	41
COMPONENT_AS_DATA	41
CONSTANT_AS_VARIABLE	12
DATA_AS_COMPONENT	9
INSUFFICIENT_INFORMATION	17
NON_ESSENTIAL_ABSTRACTION	80
OK	117
VARIABLE_AS_CONSTANT	3

Cuadro 6.4: Distribución de los errores para el sistema de: Ascensores en un Edificio

SYSTEM_ID	**ERROR_SUBCATEGORY_ID**	**TOTAL_FILAS**
ASCENSORES EN UN EDIFICIO	AMBIGUOUS DESCRIPTION	4
ASCENSORES EN UN EDIFICIO	INSUFICIENT INFORMATION	1
ASCENSORES EN UN EDIFICIO	NON_ESSENTIAL_ABSTRACTION	5
ASCENSORES EN UN EDIFICIO	OK	10

Cuadro 6.5: Distribución de los errores para el sistema de: Biblioteca

SYSTEM_ID	**ERROR_SUBCATEGORY_ID**	**TOTAL_FILAS**
BIBLIOTECA	AMBIGUOUS DESCRIPTION	3
BIBLIOTECA	CONSTANT AS VARIABLE	7
BIBLIOTECA	DATA AS COMPONENT	7
BIBLIOTECA	NON ESSENTIAL ABSTRACTION	2
BIBLIOTECA	OK	16

Cuadro 6.6: Distribución de los errores para el sistema de: Juego de Buscaminas

SYSTEM_ID	ERROR_SUBCATEGORY_ID	TOTAL_FILAS
BUSCAMINAS	AMBIGUOUS DESCRIPTION	7
BUSCAMINAS	COMPONENT AS DATA	1
BUSCAMINAS	INSUFICIENT INFORMATION	1
BUSCAMINAS	NON ESSENTIAL ABSTRACTI ON	4

Cuadro 6.7: Distribución de los errores para el sistema de: Concierto en un Estadio

SYSTEM_ID	ERROR_SUBCATEGORY_ID	TOTAL_FILAS
CONCIERTO EN UN ESTADIO	AMBIGUOUS_DESCRIPTION	2
CONCIERTO EN UN ESTADIO	COMPONENT_AS_DATA	18
CONCIERTO EN UN ESTADIO	INSUFFICIENT_INFORMATION	4
CONCIERTO EN UN ESTADIO	NON_ESSENTIAL_ABSTRACTION	3
CONCIERTO EN UN ESTADIO	OK	17
CONCIERTO EN UN ESTADIO	VARIABLE_AS_CONSTANT	1

Cuadro 6.8: Distribución de los errores para el sistema de: Juego de Domino

SYSTEM_ID	ERROR_SUBCATEGORY_ID	TOTAL_FILAS
DOMINO	AMBIGUOUS_DESCRIPTION	3
DOMINO	CONSTANT_AS_VARIABLE	3
DOMINO	DATA_AS_COMPONENT	1
DOMINO	INSUFFICIENT_INFORMATION	4
DOMINO	NON_ESSENTIAL_ABSTRACTION	15
DOMINO	OK	17

Cuadro 6.9: Distribución de los errores para el sistema de: Parqueadero

SYSTEM_ID	ERROR_SUBCATEGORY_ID	TOTAL_FILAS
PARQUEADERO	AMBIGUOUS_DESCRIPTION	7
PARQUEADERO	NON_ESSENTIAL_ABSTRACTION	16

Cuadro 6.10: Distribución de los errores para el sistema de: Juego de Párques

SYSTEM_ID	ERROR_SUBCATEGORY_ID	TOTAL_FILAS
PARQUES	AMBIGUOUS_DESCRIPTION	6
PARQUES	COMPONENT_AS_DATA	12
PARQUES	DATA_AS_COMPONENT	2
PARQUES	INSUFFICIENT_INFORMATION	5
PARQUES	NON_ESSENTIAL_ABSTRACTION	5
PARQUES	OK	26
PARQUES	VARIABLE_AS_CONSTANT	1

Cuadro 6.11: Distribución de los errores para el sistema de: Punto de Vacunación contra el Covid-19 en la ciudad

SYSTEM_ID	**ERROR_SUBCATEGORY_ID**	**TOTAL_FILAS**
PUNTO VACUNACION COVID-19	AMBIGUOUS_DESCRIPTION	5
PUNTO VACUNACION COVID-19	COMPONENT_AS_DATA	1
PUNTO VACUNACION COVID-19	CONSTANT_AS_VARIABLE	2
PUNTO VACUNACION COVID-19	NON_ESSENTIAL_ABSTRACTION	2
PUNTO VACUNACION COVID-19	OK	10

Cuadro 6.12: Distribución de los errores para el sistema de: Salón de Clases de un Colegio

SYSTEM_ID	**ERROR_SUBCATEGORY_ID**	**TOTAL_FILAS**
SALON DE CLASES COLEGIO	AMBIGUOUS_DESCRIPTION	2
SALON DE CLASES COLEGIO	COMPONENT_AS_DATA	9
SALON DE CLASES COLEGIO	DATA_AS_COMPONENT	5
SALON DE CLASES COLEGIO	INSUFFICIENT_INFORMATION	2
SALON DE CLASES COLEGIO	NON_ESSENTIAL_ABSTRACTION	13
SALON DE CLASES COLEGIO	OK	20

Cuadro 6.13: Distribución de los errores para el sistema de: El Curso de Introducción a la Programación

SYSTEM_ID	**ERROR_SUBCATEGORY_ID**	**TOTAL_FILAS**
CURSO INTRO PROG	AMBIGUOUS_DESCRIPTION	4
CURSO INTRO PROG	NON_ESSENTIAL_ABSTRACTION	15
CURSO INTRO PROG	OK	1
CURSO INTRO PROG	VARIABLE_AS_CONSTANT	1

Como resultado de este proceso, se obtuvo un conjunto de datos altamente desbalanceado. La mayoría de las respuestas fueron clasificadas como correctas, mientras que las respuestas que evidenciaban errores específicos eran considerablemente menos frecuentes. Esta desproporción se hizo evidente tanto a nivel global como por sistema y por subcategoría de error, lo que limitaba la representatividad de ciertos tipos de falencias dentro del dataset.

La distribución de las respuestas por sistema muestra una variabilidad significativa en el número total de instancias disponibles para cada contexto. Algunos sistemas, como “Parques” y “Salón de clases de un colegio”, concentran un mayor número de respuestas, mientras que otros, como “Buscaminas” o “Punto de vacunación de COVID-19”, presentan una cantidad notablemente menor. Esta diferencia impacta directamente la cantidad de ejemplos disponibles para identificar patrones de error en cada sistema.

De manera similar, la distribución por subcategoría de error evidencia un fuerte predominio de la clase *OK*, seguida por errores como *NON_ESSENTIAL_ABS-*

TRACTION y *AMBIGUOUS_DESCRIPTION*. En contraste, subcategorías como *VARIABLE_AS_CONSTANT* o *DATA_AS_COMPONENT* aparecen con muy pocas instancias, lo que dificulta el aprendizaje de patrones asociados a estos errores.

Al analizar las tablas individuales por sistema, se observa que el desbalance se mantiene de forma consistente. En sistemas como "Ascensores en un edificio" o "Biblioteca", la mayoría de las respuestas corresponden a la clase correcta, mientras que los errores se distribuyen en pocas instancias entre varias subcategorías. En otros casos, como "Concierto en un estadio" o "Parques", se presenta una mayor diversidad de errores, aunque sigue existiendo una diferencia considerable entre la cantidad de respuestas correctas y aquellas que evidencian falencias específicas.

Este análisis permitió concluir que, si bien el primer dataset ofrecía una visión general de los tipos de errores presentes en las respuestas de los estudiantes, su alto nivel de desbalance y la escasa representación de ciertas subcategorías lo hacían poco adecuado para el entrenamiento de modelos de clasificación supervisada. Estas limitaciones obligaron a replantear la estrategia de construcción de los datos, lo que condujo a la definición de ajustes y a la creación de un nuevo conjunto de datos, descritos en las secciones posteriores.

6.2. Ajustes Realizados al Marco Conceptual del proyecto

A partir de las limitaciones identificadas en el primer conjunto de datos, se tomó la decisión de **reformular el problema de clasificación** y **simplificar la taxonomía operativa de errores** con el fin de:

1. Favorecer la obtención de resultados apropiados en la clasificación de nuevas respuestas, considerando el volumen limitado de datos disponible para el entrenamiento.
2. Mejorar la separabilidad entre clases.
3. Alinear el análisis automático con la naturaleza de cada sección de la respuesta para garantizar que realice un trabajo adecuado.

   En consecuencia, el modelo de clasificación se modificó para un esquema híbrido: **clasificación mediante aprendizaje automático** con un nuevo dataset únicamente para las secciones `PURPOSE` e `INTERACTIONS`, e implementar un **clasificador basado en reglas** con una base de referencia para `COMPONENTS`, `VARIABLES` y `CONSTANTS`.

6.2.1. Nuevo Marco Conceptual Pedagógico

6.2.2. Categorías de Error (Nivel 1)

Las categorías de error definidas en el marco pedagógico del proyecto se mantuvieron conceptualmente consistentes con las establecidas en la fase inicial. No obstante, para la construcción del segundo dataset y de la base referencial únicamente se consideraron dos de estas categorías, dado que las nuevas subcategorías de error definidas (nivel 2) se adscriben exclusivamente a dichos tipos de error:

- **Aprendizaje incompleto o impreciso.**
- **No comprensión de las instrucciones.**

Esta decisión se tomó con base en la naturaleza de los nuevos suberrores identificados, los cuales no introducen un nuevo origen cognitivo del error, sino que se clasifican dentro de estas dos categorías para el contexto específico de la tarea analizada.

6.2.3. Subcategorías de Error (Nivel 2)

Categorias para el Aprendizaje Automático

Para las secciones evaluadas mediante aprendizaje automático, se definió una taxonomía de carácter binario que permite clasificar las respuestas de los estudiantes según su cumplimiento de los criterios de la tarea. Esta taxonomía contempla las siguientes subcategorías de error:

- **Respuesta deficiente**: Corresponde a respuestas que no cumplen con los requerimientos mínimos de la sección evaluada, ya sea porque presentan información insuficiente, formulaciones demasiado generales o descripciones ambiguas que no permiten identificar con claridad el propósito del sistema o la forma en que interactúan sus componentes. En estos casos, aunque el estudiante puede demostrar una comprensión parcial de la tarea, la respuesta no es lo suficientemente clara, completa o precisa para considerarse adecuada.
- **OK**: Agrupa las respuestas correctas, es decir, aquellas que cumplen con los criterios establecidos para la sección evaluada. Estas respuestas presentan información suficiente, clara y coherente, describen adecuadamente el propósito del sistema o las interacciones entre sus componentes y permiten comprender sin ambigüedades la idea planteada por el estudiante.

Esta categorización binaria se aplica exclusivamente a las secciones `PURPOSE` e `INTERACTIONS`, debido a que la naturaleza abierta de las respuestas hace pertinente el uso de modelos de aprendizaje automático para distinguir entre respuestas adecuadas y respuestas deficientes.

6.2.4. Resumen de la taxonomía de error para aprendizaje automático

La Tabla 6.14 presenta un resumen de la taxonomía de errores utilizada para las secciones evaluadas mediante aprendizaje automático.

Cuadro 6.14: Resumen de la taxonomía de errores para aprendizaje automático

Categoría del error	**Subcategoría del error**	**Código del error**
No comprensión de las instrucciones	Respuesta deficiente	`DEFICIENT_RESPONSE`
Respuesta correcta	OK	`OK`

Categorias para el Clasificador Basado en Reglas

Para las secciones evaluadas mediante un enfoque basado en reglas, se definió una categorización binaria orientada a determinar la corrección de la abstracción

realizada por el estudiante. Esta categorización permite distinguir si los elementos identificados en la respuesta cumplen o no con los criterios conceptuales de la tarea, en función del dominio específico analizado. Las categorías definidas son las siguientes:

- **Abstracción incorrecta del sistema**: Corresponde a respuestas en las que el estudiante presenta errores en la identificación, clasificación o selección de los elementos del sistema, tales como componentes, variables o constantes. Estos errores reflejan una abstracción incorrecta del sistema propuesto, ya sea por confusión conceptual, selección de elementos no esenciales o clasificación errónea de la información. Esta categoría constituye la segunda clasificación de mayor nivel dentro del proyecto y agrupa las distintas subcategorías específicas de error definidas para la tarea.
- **OK**: Agrupa las respuestas en las que el estudiante realiza una abstracción correcta del sistema, identificando adecuadamente los componentes relevantes y clasificando de forma coherente la información asociada como variables o constantes, de acuerdo con los criterios establecidos en el enunciado de la actividad.

La categoría *Abstracción incorrecta del sistema* sirve como base para el funcionamiento del clasificador basado en reglas implementado en el proyecto. A partir de esta clasificación general, el sistema emplea sub–subcategorías de error para identificar con mayor precisión el tipo específico de falencia presente en la respuesta del estudiante, lo que permite generar retroalimentación más focalizada y alineada con la naturaleza del error cometido.

6.2.5. Sub-subcategorias de Error para la Clasificación Basada en Reglas (Nivel 3)

Para refinar la detección de errores dentro de la categoría de *Abstracción incorrecta del sistema*, se definieron subcategorías de nivel 3 que permiten identificar con mayor precisión el tipo específico de confusión cometida por el estudiante al clasificar los elementos del sistema. Estas subcategorías corresponden a las distintas combinaciones posibles de clasificación incorrecta entre componentes, variables y constantes, y constituyen la base operativa del clasificador basado en reglas utilizado en el proyecto.

- **Variable como componente**: Se presenta cuando el estudiante identifica como componente un dato cuyo valor puede variar a lo largo del tiempo o entre distintas instancias del sistema. En este caso, la información asociada a un elemento es tratada erróneamente como una parte estructural del sistema.
- **Constante como componente**: Ocurre cuando el estudiante clasifica como componente un dato que permanece fijo una vez asignado un valor. Aunque

dicho dato describe una característica estable del sistema, no constituye un elemento estructural independiente.

- **Componente como variable**: Este error se produce cuando un elemento estructural del sistema es clasificado como una variable, ignorando que el componente en sí no cambia, sino que puede tener información asociada que varía.
- **Componente como constante**: En este caso, el estudiante identifica un componente del sistema como una constante, confundiendo el elemento estructural con un dato fijo asociado al mismo.
- **Variable como constante**: Se presenta cuando el estudiante clasifica como constante un dato cuyo valor puede cambiar durante la ejecución o entre distintas instancias del sistema, lo que evidencia una abstracción incompleta del comportamiento dinámico del sistema.
- **Constante como variable**: Ocurre cuando un dato que permanece fijo en el contexto del sistema es clasificado como variable, generalmente debido a que el estudiante no considera adecuadamente las restricciones o escenarios que delimitan el comportamiento del sistema.
- **OK**: Agrupa los casos en los que el estudiante clasifica correctamente los componentes del sistema y la información asociada a estos como variables o constantes, de acuerdo con los criterios establecidos para la tarea.

Estas subcategorías permiten una identificación detallada del tipo de error cometido y posibilitan que el sistema genere retroalimentación específica y contextualizada, alineada con la naturaleza de la confusión detectada en la respuesta del estudiante.

6.2.6. Resumen de la Taxonomía para la Clasificación Basada en Reglas

Resumen de la taxonomía de error para clasificación basada en reglas

La Tabla 6.15 presenta un resumen de la taxonomía de errores utilizada para la clasificación basada en reglas. En este esquema, todas las subcategorías y sub–subcategorías de error se agrupan bajo la categoría de *Aprendizaje incompleto o impreciso*. La respuesta correcta se clasifica de manera consistente en todos los niveles como *OK*.

Cuadro 6.15: Resumen de la taxonomía de errores para clasificación basada en reglas

Categoría del error	**Subcategoría del error**	**Sub–subcategoría del error**	**Código del error**
Aprendizaje incompleto o impreciso	Abstracción incorrecta del sistema	Variable como componente	CONSTANT_AS_COMPONENT
Aprendizaje incompleto o impreciso	Abstracción incorrecta del sistema	Constante como componente	CONSTANT_AS_COMPONENT
Aprendizaje incompleto o impreciso	Abstracción incorrecta del sistema	Componente como variable	COMPONENT_AS_VARIABLE
Aprendizaje incompleto o impreciso	Abstracción incorrecta del sistema	Componente como constante	COMPONENT_AS_CONSTANT
Aprendizaje incompleto o impreciso	Abstracción incorrecta del sistema	Variable como constante	VARIABLE_AS_CONSTANT
Aprendizaje incompleto o impreciso	Abstracción incorrecta del sistema	Constante como variable	CONSTANT_AS_VARIABLE
Respuesta correcta	OK	OK	OK

6.3. Proceso de Construcción del Segundo Dataset

A continuación se describe el proceso seguido para la construcción del segundo dataset, el cual fue utilizado para el entrenamiento y evaluación de los modelos de aprendizaje automático elegidos para el proyecto. Este procedimiento utilizó el primer dataset, pero incorpora los ajustes mencionados en la sección anterior, al igual que otros pequeños cambios que mejoraron el diseño.

Es importante recordar que en este segundo conjunto de datos se incluyeron únicamente siete sistemas, dado que los tres sistemas correspondientes a la tarea del semestre 2020-2 no solicitaban la identificación del propósito del sistema ni la descripción de las interacciones entre componentes. Dado que las respuestas de estas dos secciones (`PURPOSE` e `INTERACTIONS`) son las únicas utilizadas para la clasificación basada en aprendizaje automático, dichos sistemas fueron excluidos de este dataset.

El proceso de construcción del conjunto de datos se llevó a cabo siguiendo los pasos que se describen a continuación:

1. Se realizaron consultas PL/SQL sobre el primer dataset para cambiar las subcategorías de error *INSUFFICIENT_INFORMATION* y *AMBIGUOUS_DESCRIPTION* a la categoría *DEFICIENT_RESPONSE*.

2. Se filtró el conjunto de datos para conservar únicamente las filas correspondientes a las secciones `PURPOSE` e `INTERACTIONS`. Los resultados de la consulta se detallan en la tabla:

Cuadro 6.16: Distribución de Errores por Sección (Propósito, Interacciones)

RESPONSE_CATEGORY_ID	**ERROR_SUBCATEGORY_ID**	**TOTAL_FILAS**
INTERACTIONS	DEFICIENT_RESPONSE	10
INTERACTIONS	NON_ESSENTIAL_ABSTRACTION	2
INTERACTIONS	OK	18
PURPOSE	DEFICIENT_RESPONSE	10
PURPOSE	NON_ESSENTIAL_ABSTRACTION	2
PURPOSE	OK	17

3. Se aplicó un segundo filtro para mantener exclusivamente las filas cuya categoría de error correspondiera a *DEFICIENT_RESPONSE* u *OK*, eliminando cualquier otra clasificación que no formara parte del esquema binario definido para este dataset.

4. Una vez obtenido el subconjunto de datos resultante, se evaluó la distribución de respuestas por sistema, por sección y por error, con el fin de . Los resultados se detallan en la tabla:

Cuadro 6.17: Datos de Registros por Categoría y Error

SECCIÓN	SISTEMA	SUBCATEGORIA DEL ERROR	TOTAL FILAS
INTERACTIONS	ASCENSORES	OK	2
PURPOSE	ASCENSORES	OK	2
INTERACTIONS	BIBLIOTECA	OK	1
PURPOSE	BIBLIOTECA	OK	2
INTERACTIONS	CONCIERTO_ESTADIO	DEFICIENT_RESPONSE	3
INTERACTIONS	CONCIERTO_ESTADIO	OK	5
PURPOSE	CONCIERTO_ESTADIO	DEFICIENT_RESPONSE	2
PURPOSE	CONCIERTO_ESTADIO	OK	3
INTERACTIONS	DOMINO	DEFICIENT_RESPONSE	2
INTERACTIONS	DOMINO	OK	2
PURPOSE	DOMINO	DEFICIENT_RESPONSE	2
PURPOSE	DOMINO	OK	1
INTERACTIONS	PARQUES	DEFICIENT_RESPONSE	4
INTERACTIONS	PARQUES	OK	4
PURPOSE	PARQUES	DEFICIENT_RESPONSE	5
PURPOSE	PARQUES	OK	3
INTERACTIONS	PUNTO_COVID	OK	2
PURPOSE	PUNTO_COVID	OK	2
INTERACTIONS	SALON	DEFICIENT_RESPONSE	1
INTERACTIONS	SALON	OK	2
PURPOSE	SALON	DEFICIENT_RESPONSE	1
PURPOSE	SALON	OK	4

Si un sistema no aparece en una sección o subcategoría específica, es porque no tiene registros asociados para esa combinación.

5. Finalmente, se decidió balancear el conjunto de datos agregando respuestas adicionales hasta obtener una distribución uniforme de 32 filas por sistema, correspondientes a 8 respuestas por cada sección (`PURPOSE` e `INTERACTIONS`) y por cada tipo de error (*DEFICIENT* y *OK*). Este balanceo se consideró adecuado para reducir el sesgo del modelo hacia la clase mayoritaria y permitir una comparación más equitativa del desempeño del clasificador entre sistemas y categorías.

Como resultado de este proceso, el conjunto de datos final quedó conformado por un total de 226 filas. Esta cifra se debe a que dos de los sistemas incluidos presentaron una entrada adicional, lo que generó una ligera variación respecto a la distribución ideal, sin afectar de manera significativa la distribución del dataset.

6.4. Descripción de la Base de Referencia para la Clasificación Basada en Reglas

La base de referencia utilizada en el proyecto es un conjunto de datos construido específicamente para la clasificación basada en reglas y para la validación de la abstracción correcta de los sistemas analizados por los estudiantes. Esta base incluye los diez sistemas seleccionados en la exploración inicial y sirve para detectar los errores en las secciones de componentes, variables y constantes.

Los datos de esta base fueron normalizados antes de ser insertados, eliminando artículos, tildes, mayúsculas y otros elementos que puediesen dificultar la comparación entre las respuestas de los estudiantes y los elementos de referencia.

6.4.1. Estructura de la Base de Referencia

Cuadro 6.18: Descripción de las columnas de la base de referencia

Nombre de la columna	**Descripción**
SYSTEM_ID	Identificador del sistema al que pertenece la entrada (por ejemplo, biblioteca, parqueadero, salón de clases). Permite agrupar los elementos de referencia según el dominio específico del sistema abstraído.
RESPONSE_CATEGORY_ID	Indica la sección de la tarea a la que pertenece el elemento de referencia, como componentes, variables o constantes. Esta columna permite diferenciar el tipo de información que se está evaluando.
ITEM_NORMALIZED	Representación normalizada del elemento del sistema. Corresponde al texto procesado y estandarizado (por ejemplo, eliminación de artículos, uso de minúsculas y control de variaciones léxicas) para facilitar comparaciones exactas y por similitud con las respuestas de los estudiantes.
ERROR_SUBCATEGORY_ID	Etiqueta que indica la subcategoría de error asociada al elemento en caso de una abstracción incorrecta. En la base de referencia, este campo se utiliza como parte del esquema de validación y se encuentra alineado con la taxonomía definida para el clasificador basado en reglas.

6.4.2. Distribución de la Base

La tabla a continuación describe la distribución de los datos para cada sistema y el número total de registros.

Cuadro 6.19: Distribución por sistema

Sistema	**Entradas**
Ascensores	34
Biblioteca	56
Buscaminas	51
Concierto en estadio	47
Curso de Intr. a la Prog.	38
Dominó	31
Parqueadero	36
Parqués	30
Punto de vacunación COVID	25
Salón de clases	38
Total general	**386**

6.5. Entrenamiento de los Algoritmos de Clasificación Supervisada

6.5.1. Algoritmos Seleccionados para el Entrenamiento

La elección de los modelos de Máquinas de Vectores de Soporte (SVM) y Regresión Logística para la clasificación de errores en respuestas textuales se fundamenta en su eficacia comprobada en tareas de clasificación automática de texto, particularmente cuando se trabaja con representaciones vectoriales de alta dimensionalidad derivadas de técnicas como TF-IDF y n-gramas.

Algunos autores mencionan que estos algoritmos presentan un desempeño sobresaliente en problemas de clasificación de texto al operar sobre espacios de características dispersos (*sparse*), los cuales son característicos de las representaciones vectoriales del lenguaje natural [34]. El estudio destaca que tanto SVM como la Regresión Logística logran un balance adecuado entre precisión, robustez y capacidad de generalización, superando en muchos casos a métodos más complejos que requieren un mayor ajuste de parámetros o volúmenes de datos más extensos.

En particular, estos modelos resultan apropiados para escenarios de clasificación binaria, como el abordado en este proyecto, donde el objetivo es distinguir entre respuestas correctas y respuestas que evidencian deficiencias conceptuales. Su comportamiento estable frente a conjuntos de datos de tamaño moderado y su interpretabilidad los convierten en alternativas idóneas para su integración en sistemas académicos orientados al análisis automático de respuestas en lenguaje natural [34].

6.6. Implementación de los modelos de clasificación

Esta sección describe de manera general la implementación de los modelos de clasificación supervisada utilizados en el sistema para evaluar respuestas textuales de estudiantes e identificar entre respuestas correctas (OK) y respuestas deficientes ($DEFICIENT\_RESPONSE$).

El propósito de esta sección es presentar el flujo de procesamiento y las decisiones generales de implementación, sin profundizar en detalles de código. La implementación completa de los modelos se encuentra disponible en el repositorio del trabajo de grado [33].

6.6.1. Descripción general de la implementación

Ambos modelos de clasificación fueron integrados dentro de un mismo flujo de procesamiento, permitiendo comparar su desempeño bajo condiciones equivalentes y garantizar consistencia en la evaluación. Este flujo fue diseñado para operar sobre respuestas textuales en lenguaje natural y puede resumirse de la siguiente manera:

- El sistema recibe como entrada un conjunto de datos previamente etiquetado, el cual contiene respuestas textuales asociadas a distintas secciones de la actividad académica.
- Se realizan validaciones básicas sobre la estructura del conjunto de datos para asegurar la coherencia de la información requerida por el proceso de clasificación.
- Las respuestas del estudiante son normalizadas mediante transformaciones estándar de texto, con el objetivo de reducir variaciones superficiales que no aportan significado semántico.
- Se construye una representación textual enriquecida que incorpora explícitamente la categoría de la sección evaluada, permitiendo preservar el contexto de análisis propio del dominio del problema.
- El texto resultante es transformado a una representación numérica utilizando técnicas de vectorización basadas en la relevancia de los términos dentro del conjunto de datos.
- A partir de esta representación, se generan vectores de características que son evaluados por los modelos de clasificación supervisada.
- Cada modelo produce como salida una etiqueta de clasificación asociada a la respuesta del estudiante, la cual es utilizada posteriormente por los módulos de análisis y retroalimentación del sistema.
- Durante el proceso experimental se calcularon métricas de evaluación que permiten analizar el desempeño de los modelos y comparar sus resultados.

6.6.2. Consideraciones de integración en el sistema

La implementación de los modelos de Regresión Logística y SVM fue concebida como un componente independiente dentro de la arquitectura del sistema, lo que permite su reutilización, comparación y eventual reemplazo sin afectar el resto de los módulos. Este diseño facilita la experimentación con distintos enfoques de clasificación y respalda la selección del modelo más adecuado para su integración final en el prototipo.

6.7. Resultados del Entrenamiento de los Modelos

Para la evaluación de los algoritmos de clasificación se utilizó un conjunto de datos real compuesto inicialmente por 224 respuestas de estudiantes, recolectadas a partir de tareas del curso Introducción a la Programación de la Pontificia Universidad Javeriana Cali. Dichas respuestas corresponden a siete sistemas analizados durante los semestres 2021-1 y 2021-2.

La selección de estos periodos académicos se realizó debido a que la consigna aplicada en el semestre 2020-2 presentaba una estructura diferente, enfocada únicamente en la identificación de componentes y la clasificación de variables y constantes, lo cual la hacía incompatible con el esquema de clasificación supervisada propuesto en este proyecto.

Con el fin de facilitar el proceso de entrenamiento y mantener una distribución balanceada entre clases, durante el desarrollo de los modelos se realizó un ajuste controlado del conjunto de datos, incorporando dos instancias adicionales para igualar la cantidad de ejemplos entre OK y DEFICIENT_RESPONSE. Como resultado, los experimentos se ejecutaron sobre un conjunto total de 226 instancias, sin que este ajuste alterara la naturaleza ni las características del conjunto base.

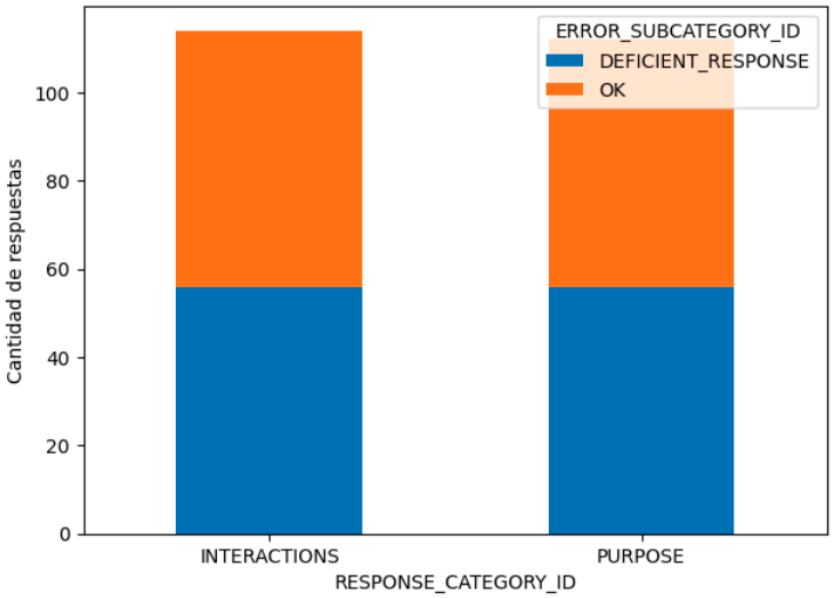

Figura 6.1: Distribución de clases por categoría de respuesta

Para cada sistema se construyeron ocho instancias por sección de texto, distribuidas entre las clases OK y DEFICIENT_RESPONSE para las categorías Purpose e Interactions. La categorización de las respuestas deficientes se basó principalmente en los comentarios y observaciones realizadas por el docente en las tareas calificadas del semestre 2021-1, siendo este el único periodo que contaba con retroalimentación pedagógica explícita.

6.7.1. Limitaciones del estudio

No obstante, es importante reconocer las limitaciones inherentes del conjunto de datos. En primer lugar, el tamaño del dataset es reducido y depende directamente de la cantidad de trabajos entregados por los estudiantes en cada semestre. En segundo lugar, la distribución natural de las respuestas presenta una alta proporción de soluciones correctas, lo que restringe la diversidad de ejemplos correspondientes a falencias de aprendizaje. Finalmente, la identificación de respuestas deficientes se apoya en un único periodo académico con retroalimentación explícita, lo cual limita la generalización de los patrones de error observados.

Estas limitaciones fueron consideradas explícitamente durante el diseño experimental y la interpretación de los resultados, y no invalidan el estudio, sino que delimitan su alcance y contexto de aplicación.

6.8. Representación textual y configuración de los modelos

Las respuestas analizadas corresponden a texto libre, con diferencias en longitud y forma de redacción entre estudiantes. Por esta razón, fue necesario utilizar una técnica de representación que permitiera transformar los textos en una forma numérica adecuada para los modelos de clasificación, sin depender de estructuras sintácticas complejas. Para este propósito se empleó el esquema TF-IDF (Term Frequency–Inverse Document Frequency), el cual es ampliamente utilizado en tareas de clasificación de texto debido a su efectividad y simplicidad.

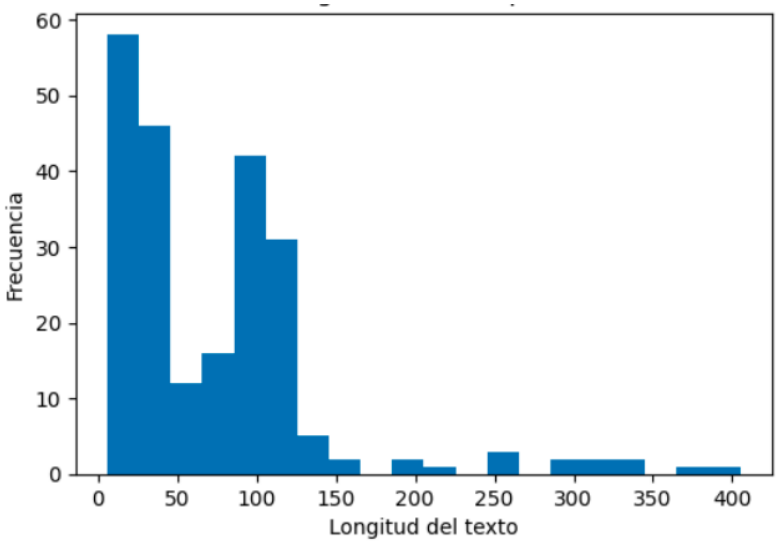

Figura 6.2: Distribución de longitud de las respuestas (caracteres)

La representación TF-IDF se configuró utilizando n-gramas de tamaño uno y dos, con el fin de capturar tanto palabras individuales como combinaciones frecuentes

de términos. Esta configuración resulta especialmente adecuada para textos cortos, donde ciertas expresiones compuestas aportan información relevante que no siempre se obtiene al analizar palabras de forma aislada.

Con el objetivo de conservar el contexto de cada respuesta, se incorporó explícitamente la categoría de la sección (Purpose o Interactions) como un token adicional dentro del texto. Esta estrategia permite que el modelo distinga entre respuestas que describen el propósito del sistema y aquellas que explican sus interacciones, incluso cuando comparten vocabulario similar.

Por otra parte, se decidió no incluir el identificador del sistema como variable de entrada. Esta decisión se tomó para evitar sesgos y problemas de sobreajuste asociados a la memorización de patrones específicos de sistemas con un número reducido de ejemplos. De este modo, el modelo se entrenó para identificar patrones lingüísticos generales relacionados con respuestas correctas o deficientes, independientemente del sistema analizado.

Tanto el modelo de Máquinas de Vectores de Soporte (SVM) como el de Regresión Logística se configuraron como clasificadores binarios, con las clases OK y DEFICIENT_RESPONSE. En ambos casos se utilizó una partición del 80 % de los datos para entrenamiento y el 20 % restante para prueba, aplicando muestreo estratificado con el fin de mantener la proporción entre clases.

6.9. Resultados obtenidos con SVM y Regresión Logística

Bajo la configuración experimental descrita en la sección anterior, ambos modelos evaluados presentaron un desempeño muy similar. Tanto el SVM lineal como la Regresión Logística alcanzaron una exactitud (accuracy) del 89,13 % sobre el conjunto de prueba, lo que indica que aproximadamente nueve de cada diez respuestas fueron clasificadas correctamente. Este resultado sugiere que ambos enfoques son adecuados para la tarea de clasificación binaria planteada en el contexto del proyecto.

Un aspecto particularmente relevante en este estudio es el desempeño de los modelos en la identificación de respuestas deficientes. El recall de la clase DEFICIENT_RESPONSE alcanzó un valor del 86,96 % en ambos enfoques, lo que indica que la mayoría de las respuestas que presentan falencias reales de aprendizaje fueron correctamente identificadas por el sistema. Esta métrica resulta especialmente importante en el contexto educativo del proyecto, ya que el objetivo principal no es únicamente clasificar correctamente, sino detectar oportunamente aquellas respuestas que requieren retroalimentación pedagógica.

En la Figura 6.3 se presenta una comparación visual de las métricas principales obtenidas por ambos modelos, incluyendo *accuracy*, *precision* y *recall*. Como puede observarse, las diferencias entre SVM y Regresión Logística son mínimas, lo que

refuerza la idea de que ambos modelos aprenden fronteras de decisión similares bajo las condiciones del experimento.

```
Train size: 180 | Test size: 46
=== RESULTADOS SVM (Linear) ===
Accuracy: 0.8913
Recall (DEFICIENT_RESPONSE): 0.8696
```

```
Train size: 180 | Test size: 46
******** Resultados Logistic Regression *********
Accuracy: 0.8913
Recall (DEFICIENT_RESPONSE): 0.8696
```

Figura 6.3: Comparación de métricas entre modelos

Para complementar el análisis de la exactitud, se examinaron las matrices de confusión obtenidas para cada modelo. En ambos casos, la distribución de verdaderos positivos, verdaderos negativos, falsos positivos y falsos negativos fue equivalente, lo que evidencia un comportamiento consistente entre los dos clasificadores. En la Figura 6.4 se presenta la matriz de confusión correspondiente al modelo SVM, mientras que la Figura 6.5 muestra la obtenida para el modelo de Regresión Logística.

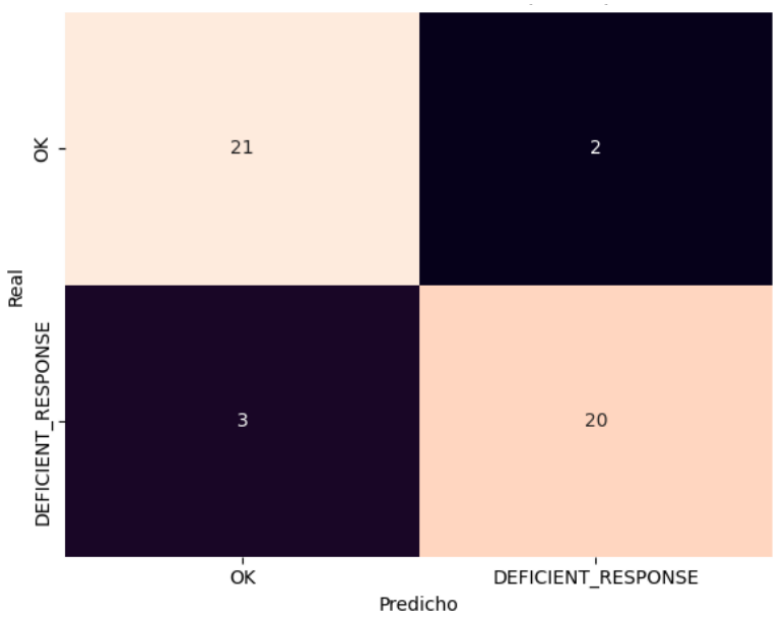

Figura 6.4: Matriz de confusión para el modelo SVM (Linear)

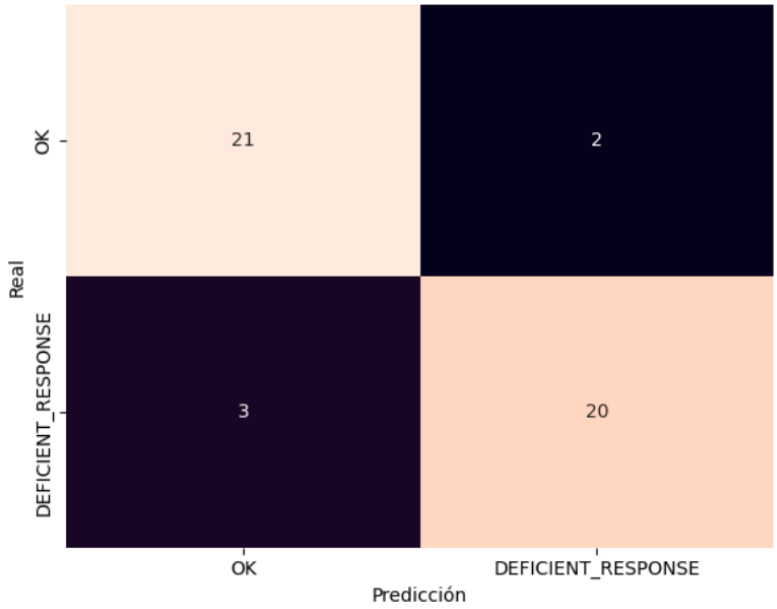

Figura 6.5: Matriz de confusión para el modelo de regresión lineal

El bajo número de falsos positivos observados en en las anteriores dos figuras sugiere que el sistema rara vez clasifica como deficientes respuestas que en realidad son correctas. Este comportamiento es deseable, ya que evita generar retroalimentación innecesaria o confusa para los estudiantes cuando su respuesta es adecuada.

Estos resultados son coherentes con lo reportado en la literatura para problemas de clasificación binaria de texto corto utilizando representaciones TF-IDF y conjuntos de datos de tamaño reducido, donde modelos lineales suelen presentar desempeños comparables [34]. En este sentido, los resultados obtenidos validan la viabilidad técnica del enfoque propuesto y proporcionan una base sólida para la selección del modelo que será integrado al prototipo.

6.9.1. Justificación de la selección del modelo SVM

A pesar de que ambos algoritmos evaluados presentaron métricas de desempeño equivalentes, se decidió seleccionar el SVM lineal como clasificador final del sistema. Esta decisión se fundamenta tanto en criterios teóricos como en consideraciones de diseño del prototipo.

Desde el punto de vista teórico, los SVM han demostrado ser particularmente robustos en tareas de clasificación de texto con alta dimensionalidad y pocos datos, características propias del conjunto analizado. Desde la perspectiva del sistema implementado, el SVM se ajusta mejor a los requerimientos del prototipo, ya que proporciona una decisión de clasificación directa, sin depender del uso explícito de probabilidades o umbrales de confianza, los cuales no forman parte del flujo de retroalimentación implementado [34].

Finalmente, la elección del SVM permite una proyección futura del sistema, faci-

litando su extensión hacia esquemas de clasificación multiclase en caso de incorporar nuevos tipos de errores o subcategorías de falencias de aprendizaje. En este sentido, aunque ambos modelos demostraron ser técnicamente viables, el SVM lineal fue seleccionado por su solidez, coherencia con la arquitectura propuesta y alineación con los objetivos del proyecto.

6.10. Desarrollo en Oracle APEX: Interfaz y Base de Datos

6.10.1. Base de Datos

El desarrollo de la plataforma web del prototipo se llevó a cabo utilizando Oracle APEX, abordando dos componentes principales: la construcción de la base de datos y el diseño de la interfaz de usuario. Esta elección permitió implementar de manera ágil un sistema funcional, integrando persistencia de datos, lógica de negocio y visualización, todo dentro de un mismo entorno de desarrollo.

Para la implementación de la base de datos, en primer lugar se diseñó y creó un esquema relacional que soporta el flujo completo del sistema, desde el registro de usuarios hasta el almacenamiento de errores y retroalimentación generada. Las tablas fueron creadas mediante sentencias SQL, asegurando integridad referencial y una estructura coherente con los requisitos funcionales definidos. La Figura 6.6 presenta el diagrama entidad–relación (ERD) que resume esta estructura.

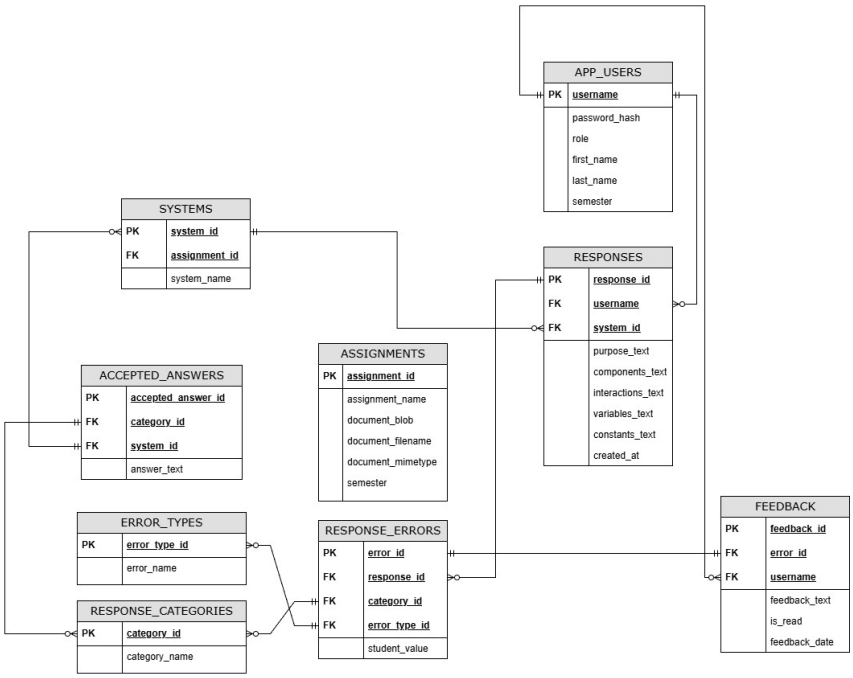

Figura 6.6: Modelo Entidad Relación de la base de datos

A continuación, se describen las principales tablas del sistema:

- **APP_USERS:** almacena la información de los usuarios de la plataforma, incluyendo credenciales, rol (estudiante o profesor) y semestre.
- **ASSIGNMENTS:** contiene los metadatos de las tareas académicas, así como el archivo PDF asociado a cada una.
- SYSTEMS: representa los sistemas que los estudiantes deben analizar en cada tarea, vinculados a un enunciado específico.
- **RESPONSES:** almacena las respuestas textuales enviadas por los estudiantes para cada sistema y sección de la tarea.
- **RESPONSE_CATEGORIES:** define las secciones evaluadas de la tarea (propósito, componentes, interacciones, variables y constantes).
- **ERROR_TYPES:** contiene los tipos de error definidos en la taxonomía del proyecto.
- **RESPONSE_ERRORS:** registra los errores detectados en cada respuesta, relacionando la respuesta del estudiante, la categoría evaluada y el tipo de error identificado.
- **FEEDBACK:** almacena la retroalimentación generada automáticamente para cada error, permitiendo su posterior consulta por estudiantes y profesores.

6.10.2. Desarrollo de la Interfaz

Oracle APEX permitió implementar rápidamente pantallas funcionales sin necesidad de construir manualmente toda la lógica de presentación. Una de las ventajas principales de la herramienta es la amplia disponibilidad de documentación y tutoriales oficiales, lo que facilitó la incorporación de funcionalidades específicas. Un ejemplo de ello fue la visualización del archivo PDF de la tarea dentro de la plataforma, implementada siguiendo como referencia el tutorial PDF Viewer in Oracle APEX publicado por Oracle Developers, adaptándolo a las necesidades del proyecto. [35]

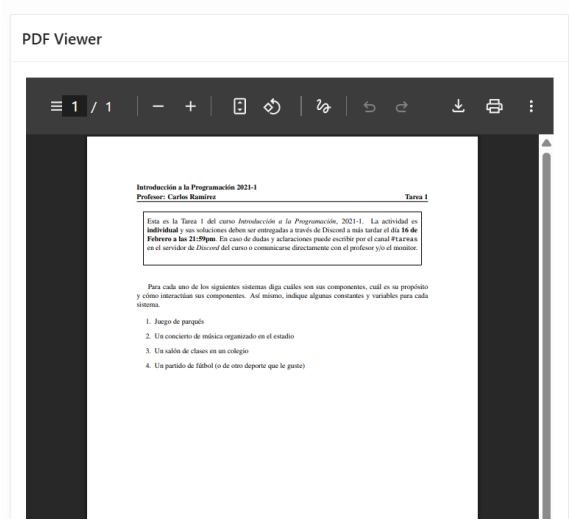

Figura 6.7: Visualizador de PDFs integrado a la interfaz

Adicionalmente, APEX permitió enriquecer la experiencia del usuario mediante la incorporación de código JavaScript y CSS personalizado. Un caso relevante fue la implementación de un mecanismo de guardado automático del contenido ingresado en los campos del formulario, incluso cuando el estudiante no había enviado aún su respuesta. Esta funcionalidad permitió que el usuario pudiera navegar entre distintos sistemas sin perder la información previamente escrita, mejorando significativamente la usabilidad de la plataforma. Asimismo, se utilizó CSS personalizado para mejorar la apariencia visual de la aplicación, como en el diseño del banner principal de la interfaz.

Figura 6.8: Banner diseñado con CSS

En conjunto, el desarrollo en Oracle APEX permitió construir un prototipo funcional de manera eficiente, integrando base de datos, lógica de negocio, servicios externos y una interfaz intuitiva. El resultado del desarrollo se puede visualizar en el capítulo de Validación del Sistema y Resultados.

6.11. Desarrollo de la API e Integración con Oracle APEX

6.11.1. Despliegue Local de la API

El desarrollo de la API constituyó una de las etapas centrales del proyecto, dado que esta actúa como el componente encargado de procesar las respuestas de los estudiantes, aplicar los mecanismos de clasificación definidos y generar la retroalimentación automatizada que posteriormente es consumida por la plataforma web.

El primer paso de la implementación consistió en definir el entorno tecnológico adecuado para la construcción del servicio. Se optó por el uso de Python debido a su amplio soporte para tareas de procesamiento de lenguaje natural, aprendizaje automático y desarrollo de servicios web. En particular, se seleccionó el framework FastAPI por su ligereza, facilidad para la definición de endpoints REST y su adecuada integración con servicios externos.Adicionalmente, se utilizó Uvicorn para ejecutar la aplicación y exponer la API como un servicio web accesible, tanto durante las pruebas locales como en el proceso de despliegue.

Una vez definido el entorno, el desarrollo de la API se realizó de manera incremental. Inicialmente, se creó una estructura básica del proyecto y se implementó un endpoint de verificación (`/health`) cuyo propósito era confirmar que el servicio se encontraba activo y accesible. Este endpoint permitió validar tempranamente la correcta configuración del entorno y sirvió como punto de partida para las pruebas iniciales de comunicación, tal como se ilustra en la Figura 6.9.

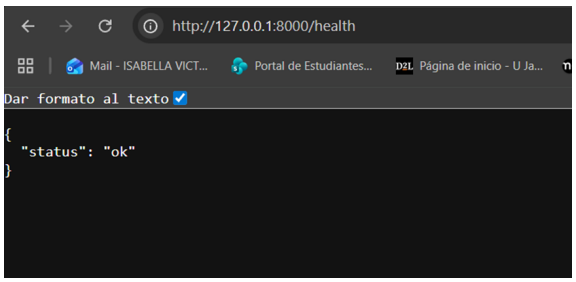

Figura 6.9: Endpoint Health

6.11.2. Implementación del Modelo de Clasificación por Aprendizaje Automatizado

A continuación se desarrollaron los archivos encargados de la clasificación mediante aprendizaje automático. Tal como se mencionó en la sección anterior, este componente fue implementado con el objetivo de desacoplar el proceso de entrenamiento del proceso de inferencia, permitiendo cargar de forma independiente los artefactos generados durante el entrenamiento del modelo. En particular, el sistema

utiliza un vectorizador previamente entrenado en un cuaderno Jupyter de Google Colab —el algoritmo implementado se puede revisar en el repositorio del proyecto [33]—y almacenado (*vectorizer.joblib*) junto con el modelo SVM resultante del proceso de entrenamiento, los cuales se cargan una única vez al iniciar la aplicación.

La ventaja que esto representa es que, ante una modificación del conjunto de datos de entrenamiento y un nuevo proceso de reentrenamiento del modelo, los artefactos se pueden actualizar sin la necesidad de alterar la lógica de la API ni interrumpir el flujo general del sistema. De esta manera, el clasificador puede evolucionar y adaptarse a nuevos datos manteniendo la estabilidad de la integración con la plataforma web.

Para la clasificación se incorporaron validaciones específicas para manejar adecuadamente casos en los que el estudiante no proporciona texto en alguna de las secciones evaluadas mediante aprendizaje automático. En estas situaciones, el sistema evita clasificar la respuesta como deficiente, dado que no existe información para realizar una evaluación válida. Esta decisión previene la generación de retroalimentación incorrecta y asegura que el análisis se realice únicamente cuando hay contenido que pueda ser interpretado por el modelo.

Cuando ya se contaba con la lógica, se procedió a definir el endpoint principal de análisis (***/analyze***) y se realizaron realizar pruebas POST en el entorno local. Como se observa en la Figura 6.9, la respuesta retornada incluye los resultados del modelo de aprendizaje automático para las secciones de propósito e interacciones, indicando explícitamente la clase predicha para cada una de ellas. Esta verificación fue fundamental para asegurar que la lógica de clasificación supervisada estaba correctamente integrada al endpoint y producía resultados coherentes con la taxonomía definida.

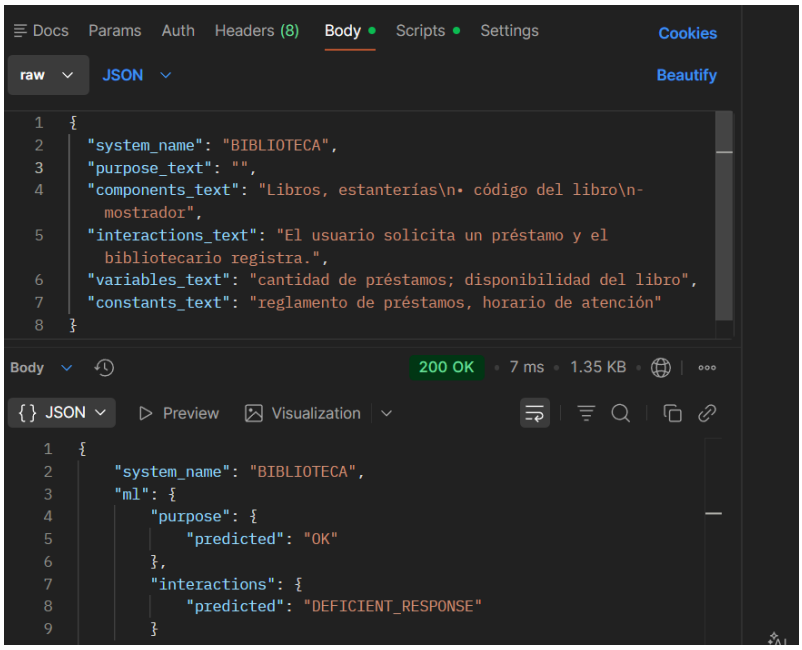

Figura 6.10: Prueba de clasificación con aprendizaje automático en Postman

6.11.3. Implementación del Modelo de Clasificación Basado en Reglas

Una vez validados los mecanismos de clasificación mediante aprendizaje automático, se procedió al desarrollo del módulo de clasificación basada en reglas, encargado de analizar las secciones de componentes, variables y constantes. Este enfoque se fundamenta en la comparación directa entre los elementos ingresados por el estudiante y la base de referencia construida previamente, permitiendo identificar errores específicos de abstracción del sistema.

Previo a la aplicación de las reglas, las respuestas del estudiante pasan por un proceso de tokenización y normalización. En esta etapa, los textos correspondientes a componentes, variables y constantes se separan utilizando delimitadores comunes como comas, saltos de línea y punto y coma. Posteriormente, cada token es normalizado eliminando artículos, tildes, mayúsculas y otros elementos lingüísticos que puedan afectar la comparación textual.

Para la comparación entre los tokens del estudiante y los elementos de la base de referencia descrita en el capítulo 6.4, se implementaron dos mecanismos complementarios: comparación exacta y comparación por similitud. En el caso de la comparación por similitud, se utilizó un enfoque de coincidencia difusa (fuzzy matching) con un umbral del 85 %. La elección de este umbral responde a una decisión pedagógica y técnica: dadas las limitaciones inherentes de la base de referencia y la naturaleza abierta de las respuestas, se priorizó la flexibilidad del sistema, prefiriendo clasificar como correctos elementos que podrían no coincidir exactamente con la referencia, antes que marcar como incorrectos elementos que conceptualmente son válidos pero están expresados de forma distinta. Este enfoque busca reducir falsos negativos y favorecer el aprendizaje, alineándose con el objetivo del sistema de apoyar al estudiante y no penalizar variaciones razonables en la formulación de sus respuestas.

Es importante resaltar que este diseño responde al carácter del sistema como herramienta de apoyo al aprendizaje. El prototipo **no pretende reemplazar la función del profesor en la retroalimentación ni en la evaluación formal**, sino ofrecer un acompañamiento adicional que permita al estudiante reflexionar sobre su respuesta incluso cuando no cuenta con la retroalimentación inmediata del docente.

Con base en estos principios, se definió un conjunto claro y explícito de reglas para la clasificación, estructuradas según la sección de la tarea en la que aparece cada token.

6.11.4. Reglas del Clasificador Basado por Reglas

A) Si algo aparece en VARIABLES:

Para cada token t en VARIABLES del estudiante:

1. Si $t \in$ OK_COMPONENTS(system) $\rightarrow$ sub-error = COMPONENT_AS_VARIABLE
2. Else if $t \in$ OK_CONSTANTS(system) $\rightarrow$ sub-error = CONSTANT_AS_VARIABLE
3. Else if $t \in$ OK_VARIABLES(system) $\rightarrow$ OK
4. Else $\rightarrow$ UNKNOWN_OR_NONESSENTIAL

B) Si algo aparece en CONSTANTS:

Para cada token t en CONSTANTS del estudiante:

1. Si $t \in$ OK_COMPONENTS(system) $\rightarrow$ COMPONENT_AS_CONSTANT
2. Else if $t \in$ OK_VARIABLES(system) $\rightarrow$ VARIABLE_AS_CONSTANT
3. Else if $t \in$ OK_CONSTANTS(system) $\rightarrow$ OK
4. Else $\rightarrow$ UNKNOWN_OR_NONESSENTIAL

C) Si algo aparece en COMPONENTS:

Para cada token t en COMPONENTS del estudiante:

1. If $t \in$ OK_COMPONENTS(system) $\rightarrow$ OK
2. Else if $t \in$ OK_VARIABLES(system) $\rightarrow$ (opcional) VARIABLE_AS_COMPONENT
3. Else if $t \in$ OK_CONSTANTS(system) $\rightarrow$ (opcional) CONSTANT_AS_COMPONENT
4. Else $\rightarrow$ UNKNOWN_OR_NONESSENTIAL.

Estas reglas permiten identificar de forma precisa el tipo de error de abstracción cometido por el estudiante, asociándolo a una sub–subcategoría específica cuando se pueda. Por ejemplo, si un elemento como *"código del libro"* pertenece a la categoría de constantes del sistema biblioteca y el estudiante lo clasifica como variable, el sistema detecta el caso como **CONSTANT_AS_VARIABLE** y genera una retroalimentación explícita indicando que se trata de una constante y no de una variable. No obstante, se decidió no incluir una clasificación automática para los casos *UNKNOWN* o *NONESSENTIAL*, dado que estas situaciones no pueden validarse de manera confiable únicamente mediante reglas, puesto que distinguir entre un elemento no esencial y no contemplado en la base de referencia requiere un juicio contextual que excede el alcance del clasificador basado en reglas.

6.11.5. Despliegue para URL Pública

Se procedió a desplegar la API en un entorno accesible públicamente. Para este propósito se utilizó Railway, un servicio de alojamiento que ofrece una configuración sencilla y un plan gratuito adecuado para prototipos académicos y pruebas de integración.

El despliegue permitió generar una URL pública asociada a la API, replicando el comportamiento previamente validado en el entorno local. Una vez completado el proceso de *deployment*, se realizaron pruebas adicionales utilizando Postman, esta vez consumiendo el endpoint `/analyze` a través de la URL pública. Estas pruebas confirmaron que la API respondía correctamente a solicitudes externas, procesando las entradas enviadas y retornando las predicciones de clasificación y los resultados de las reglas de abstracción de manera consistente con los resultados obtenidos en el entorno local.

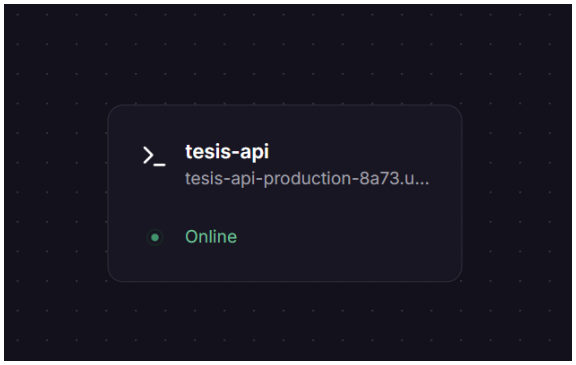

Figura 6.11: Servicio activo y desplegado en Railway

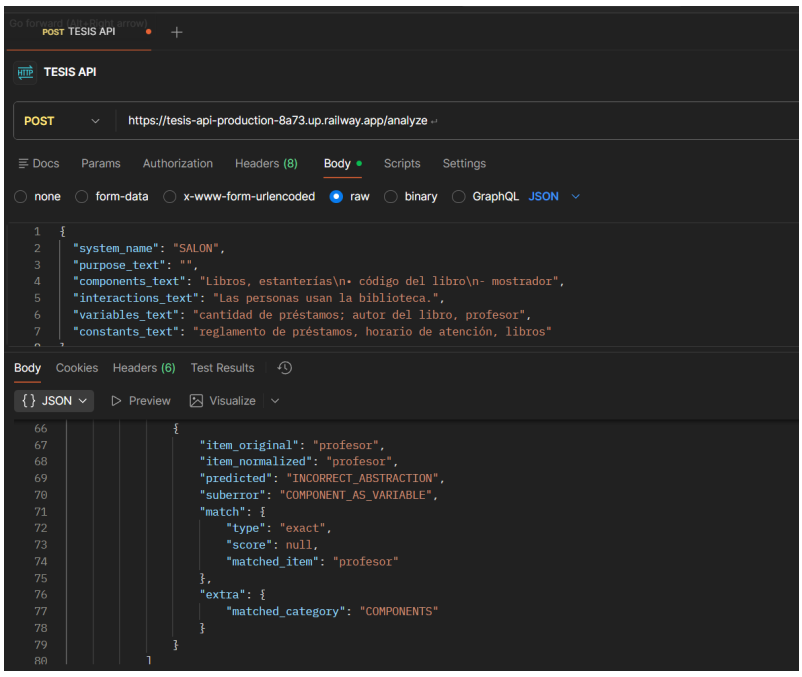

Figura 6.12: Prueba de la clasificación basada en reglas en Postman

6.11.6. Implementación de la Generación Automatizada de Retroalimentación con OpenAI API

Para la implementación del módulo de generación automatizada de retroalimentación se utilizó la API de OpenAI, la cual permite generar texto de manera controlada a partir de instrucciones específicas. En primer lugar, se realizó la creación de una clave de acceso (*API key*) asociada al proyecto, necesaria para autenticar las solicitudes realizadas desde la API desarrollada y garantizar un uso controlado del servicio.

Luego se construyeron y ajustaron los archivos encargados de la integración entre el sistema de análisis y la API de OpenAI. Se incorporó una estructura de solicitud que encapsula el tipo de error detectado, el valor incorrecto identificado y el contexto de la tarea, lo que permite que la retroalimentación generada sea específica para cada error. Adicionalmente, se implementó la lógica para generar retroalimentación de forma independiente para cada error detectado, manteniendo una relación uno a uno entre los errores y la retroalimentación.

Finalmente, una vez integrada la API de OpenAI dentro del flujo general del sistema, se realizaron pruebas utilizando Postman para validar la correcta generación de retroalimentación ante distintos escenarios de error. Estas pruebas permitieron verificar que, a partir de los resultados del módulo de clasificación, el sistema era capaz de producir mensajes de retroalimentación automatizados, coherentes y diferenciados según el tipo de error, completando así el ciclo de análisis, clasificación y apoyo al aprendizaje propuesto por el prototipo.

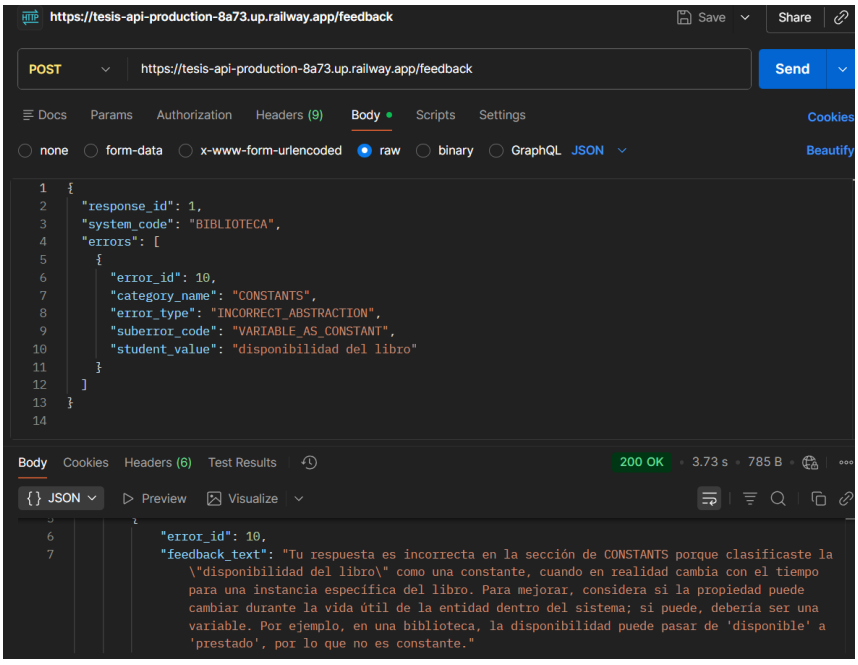

Figura 6.13: Prueba de generación de retroalimentación automatizada en Postman

6.11.7. Características de la Retroalimentación

En el diseño de la retroalimentación generada por el sistema se tuvieron en cuenta distintos enfoques teóricos sobre cómo debe presentarse el feedback para que sea pedagógicamente efectivo. La literatura coincide en que la retroalimentación no debe limitarse únicamente a indicar si una respuesta es correcta o incorrecta, sino que debe aportar información que ayude al estudiante a comprender sus errores y a mejorar su desempeño futuro. En particular, los sistemas de retroalimentación automatizada deben equilibrar claridad, utilidad y orientación al aprendizaje, evitando tanto la ambigüedad como la entrega directa de respuestas.

De acuerdo con un estudio publicado en 2021 [5], existen diferentes tipos de retroalimentación. En este proyecto se optó por abarcar principalmente dos: la retroalimentación correctiva y la retroalimentación informativa. La retroalimentación correctiva indica explícitamente si una respuesta o acción es correcta o incorrecta, lo cual resulta útil para que el estudiante reconozca de manera inmediata la presencia de un error. Por su parte, la retroalimentación informativa proporciona información adicional sobre la tarea o el concepto evaluado sin sugerir directamente una acción específica, con el objetivo de promover una comprensión más profunda del tema. Este enfoque resulta especialmente pertinente para abordar errores asociados al aprendizaje incompleto o impreciso, donde el estudiante presenta vacíos conceptuales que pueden ser subsanados mediante la introducción de ejemplos, aclaraciones o definiciones complementarias [9].

La selección de estos tipos de retroalimentación se alinea con las categorías de error de nivel 1 definidas en el marco pedagógico del proyecto. En los casos de aprendizaje incompleto o impreciso, el sistema prioriza una retroalimentación de carácter informativo, orientada a reforzar conceptos clave. En contraste, cuando se detectan errores asociados a la no comprensión de las instrucciones, la retroalimentación tiene un carácter más correctivo, enfocándose en señalar qué aspecto de la consigna no fue atendido y qué se esperaba en la respuesta, dado que el problema no radica en el conocimiento conceptual sino en la interpretación de la tarea.

Adicionalmente, se consideraron las recomendaciones presentadas en el estudio [36], el cual identifica tres estrategias clave para la presentación del feedback: retroalimentación binaria (indicar si una respuesta es correcta o incorrecta), retroalimentación específica al error (señala de forma explícita el error cometido y el punto de la respuesta donde este ocurre) y retroalimentación orientada a la solución (ofrece pistas, orientaciones o estrategias para que el estudiante pueda corregir su error sin proporcionar directamente la respuesta correcta).

El estudio señala que la retroalimentación puramente binaria puede generar confusión y frustración en los estudiantes. En consecuencia, el sistema desarrollado prioriza una combinación de retroalimentación específica al error y orientada a la mejora, identificando claramente dónde se encuentra la falla y ofreciendo pistas o explicaciones que ayuden al estudiante a reflexionar sobre su respuesta. Este enfoque busca apoyar el aprendizaje autorregulado y posiciona al sistema como una herramienta complementaria al rol del docente, sin pretender reemplazar la retroalimentación humana.

6.11.8. Integración en Oracle Apex

La integración de la API desarrollada con la plataforma web se realizó utilizando Oracle APEX, el cual se ejecuta sobre una base de datos autónoma en Oracle Cloud Infrastructure. Este entorno presenta restricciones de seguridad que limitan, por defecto, la comunicación con servicios externos. Por esta razón, el primer paso para permitir la integración consistió en autorizar explícitamente la URL pública de la API desplegada, habilitando así que la aplicación APEX pudiera realizar solicitudes HTTP salientes hacia dicho servicio. Esta autorización fue un requisito indispensable para garantizar una comunicación segura y controlada entre la plataforma y la API de análisis.

Una vez habilitado el acceso a la URL externa, se procedió a implementar los procesos necesarios para realizar la comunicación entre APEX y la API. En términos generales, se desarrollaron procesos en el servidor que se encargan de construir el objeto JSON con la información ingresada por el estudiante —incluyendo el sistema seleccionado y los textos correspondientes a cada sección de la tarea— y enviarlo mediante una solicitud HTTP al endpoint de análisis. Como respuesta, la API retorna un objeto JSON que contiene los resultados de la clasificación, tanto del enfoque basado en aprendizaje automático como del clasificador basado en reglas.

Los procesos en Oracle APEX corresponden a bloques de código en PL/SQL que se ejecutan en momentos específicos del ciclo de vida de una página o de la aplicación, permitiendo realizar operaciones como validaciones, llamadas a servicios externos, almacenamiento de información y control del flujo de la aplicación [37].

En el contexto de este proyecto, se definieron procesos independientes en Oracle APEX con el objetivo de estructurar de manera clara y modular el flujo de análisis, clasificación y retroalimentación de las respuestas de los estudiantes. Cada proceso cumple una función específica dentro del ciclo de vida de la página y se ejecuta en momentos controlados del envío del formulario, permitiendo una correcta interacción entre la interfaz de usuario, la API externa y la base de datos.

En el contexto de este proyecto, se definieron procesos separados para: invocar la API en los endpoints de análisis o feedback, capturar y almacenar la respuesta JSON recibida y guardar en base de datos los errores detectados y la retroalimentación generada.

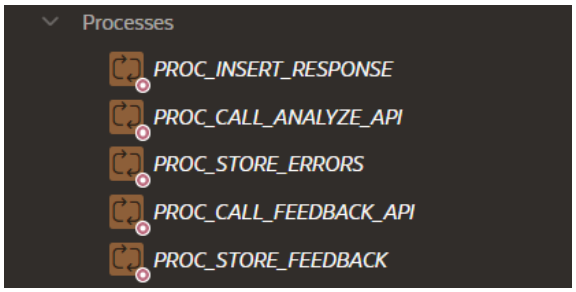

Figura 6.14: Procesos para el análisis y la retroalimentación

- **PROC_INSERT_RESPONSE**: Este proceso es el encargado de almacenar la respuesta original del estudiante en la base de datos. Recopila la información ingresada en los campos de texto del formulario, junto con el usuario autenticado y el sistema seleccionado, y la inserta en la tabla `RESPONSES`.
- **PROC_CALL_ANALYZE_API**: Este proceso se encarga de la comunicación con la API externa de análisis. A partir de la información previamente almacenada, construye el objeto JSON que contiene el nombre del sistema y los textos ingresados por el estudiante, y realiza una solicitud HTTP al endpoint público `/analyze`. Como resultado, recibe un JSON con la clasificación obtenida mediante aprendizaje automático y reglas.
- **PROC_STORE_ERRORS**: Una vez recibida la respuesta de la API, este proceso se encarga de interpretar el JSON retornado y registrar los errores detectados en la tabla `RESPONSE_ERRORS`. Para cada sección evaluada y cada ítem identificado como erróneo, se almacena el tipo de error, la categoría asociada y el valor ingresado por el estudiante.
- **PROC_CALL_FEEDBACK_API**: Este proceso tiene como función invocar el endpoint público `/feedback` que se conecta con API de generación de retroalimentación basada en OpenAI. Para cada error previamente registrado, construye un nuevo JSON que incluye el tipo de error, la sección afectada y el valor incorrecto. La respuesta recibida contiene el texto de feedback generado de manera automática.
- **PROC_STORE_FEEDBACK**: Finalmente, este proceso se encarga de guardar la retroalimentación generada en la tabla `FEEDBACK`. Asocia cada mensaje de feedback con el error correspondiente y con el usuario que realizó la respuesta, permitiendo su posterior visualización en la interfaz.

El flujo completo de integración inicia cuando el estudiante envía su respuesta desde la interfaz web. En ese momento, APEX ejecuta el proceso de llamada a la API, recibe los resultados de clasificación y retroalimentación, y procede a almacenarlos de forma estructurada en la base de datos. Posteriormente, esta información es utilizada para mostrar al estudiante los errores detectados y la retroalimentación correspondiente, así como para permitir al profesor consultar los resultados consolidados. De esta manera, la integración entre Oracle APEX y la API externa constituye el eje central que articula la interacción entre el usuario, el análisis automatizado y la presentación de resultados dentro del sistema.

Con el fin de facilitar la depuración durante el desarrollo, se incorporó en la página del formulario un elemento adicional, visible únicamente para el desarrollador, que permite visualizar el contenido completo del JSON recibido desde la API.

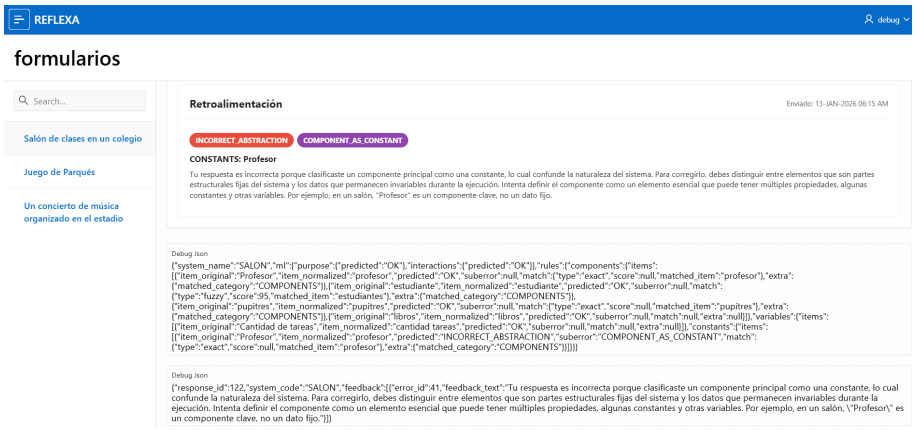

Figura 6.15: Visualización JSON por parte del sistema

Capítulo 7

Validación del Sistema y Resultados

7.1. Plan de Pruebas

El propósito del plan de pruebas fue verificar el correcto funcionamiento del prototipo desarrollado, asegurando que las funcionalidades definidas en los requerimientos funcionales se ejecutaran de manera adecuada y coherente con los objetivos del proyecto. Dado que el sistema corresponde a un prototipo académico, las pruebas se enfocaron en validar el comportamiento funcional del sistema y la correcta integración entre sus componentes, sin aplicar métricas cuantitativas formales de desempeño.

A través de estas pruebas se buscó comprobar que el flujo de uso para estudiantes y profesores se ejecutara de manera consistente, que las respuestas enviadas fueran procesadas correctamente y que los errores y la retroalimentación generada se almacenaran y presentaran de forma adecuada.

7.2. Alcance de las Pruebas

El alcance de las pruebas incluyó la validación integral del sistema bajo condiciones controladas, evaluando las principales funcionalidades implementadas. En particular, se analizaron los siguientes aspectos:

- Autenticación de usuarios y gestión de roles.
- Registro y gestión de tareas académicas por parte del profesor.
- Diligenciamiento y envío de respuestas por parte del estudiante.
- Procesamiento de las respuestas y detección de errores conceptuales.

- Clasificación de errores por categoría y subcategoría.
- Generación de retroalimentación automatizada por cada error detectado.
- Visualización de resultados y retroalimentación en la plataforma web.
- Consulta del historial de respuestas, errores y retroalimentación.
- Persistencia de la información en la base de datos.

7.3. Validación de Requisitos

La validación del sistema se realizó con base en los requerimientos funcionales definidos en la Sección 5.1.1. Cada requisito fue evaluado mediante la ejecución de escenarios de prueba diseñados para verificar su cumplimiento.

La Tabla 7.1 presenta el resumen de la validación de los requerimientos funcionales del prototipo, indicando el criterio de éxito esperado y la forma en que cada requisito fue comprobado durante las pruebas.

Cuadro 7.1: Validación de Requisitos Funcionales del prototipo

Requisito	**Nombre**	**Criterio de Éxito**	**Validación**
RF-01	Autenticación de usuarios	Acceso exitoso con credenciales válidas y bloqueo con credenciales inválidas.	Se probó inicio de sesión con usuarios de rol estudiante y profesor; se validaron errores por credenciales incorrectas.
RF-02	Gestión de roles y permisos	Cada rol visualiza únicamente funcionalidades autorizadas.	Se verificó que estudiantes no acceden a vistas de profesor y viceversa, mediante navegación y accesos directos.
RF-03	Registro y gestión de tareas	El profesor registra tareas asociadas a un sistema/dominio.	Se creó una tarea con sistema definido y se verificó disponibilidad para estudiantes.
RF-04	Carga y visualización del enunciado	PDF cargado y visible desde la plataforma.	Se cargó archivo PDF y se validó su visualización y descarga desde la interfaz.

Requisito	**Nombre**	**Criterio de Éxito**	**Validación**
RF-05	Diligenciamiento por secciones	Campos separados por sección permiten registrar respuesta completa.	Se ingresó texto en cada sección (propósito, componentes, interacciones, variables, constantes) y se validó persistencia.
RF-06	Consulta del enunciado durante diligenciamiento	El estudiante visualiza el PDF sin salir del formulario.	Se validó apertura/visualización del enunciado mientras se diligencian respuestas.
RF-07	Envío y almacenamiento de respuestas	Respuesta enviada queda almacenada para consulta posterior.	Se envió respuesta y se verificó su registro en base de datos y su consulta en historial.
RF-08	Procesamiento de respuestas	El sistema procesa respuestas e identifica falencias en el contexto de la tarea.	Se ejecutó el análisis sobre respuestas de prueba y se verificó generación de salida (errores/OK).
RF-09	Identificación y clasificación de errores	Errores clasificados por categoría y subcategoría según taxonomía.	Se probaron casos con errores intencionales y se validó la clasificación retornada por la API.
RF-10	Registro del valor asociado al error	Cada error almacena el fragmento/valor marcado como erróneo.	Se verificó que los valores detectados se almacenan y se presentan en la interfaz.
RF-11	Retroalimentación por error	Un error produce exactamente una retroalimentación asociada.	Se verificó relación 1:1 error–retroalimentación en BD y en vista de resultados.
RF-12	Visualización de resultados al estudiante	Se muestran errores, categorías, subcategorías, valores y feedback.	Se revisó pantalla de resultados con múltiples errores y retroalimentaciones por respuesta.
RF-13	Consulta de respuestas del estudiante	El estudiante consulta solo su historial de respuestas.	Se validó filtrado por usuario y bloqueo de acceso a respuestas de terceros.
RF-14	Consulta de respuestas por profesor	El profesor consulta respuestas asociadas a la tarea.	Se verificó consulta por tarea y visualización de respuestas de estudiantes vinculados.

Requisito	Nombre	Criterio de Éxito	Validación
RF-15	Registro de errores y retroalimentación	Historial persistente por respuesta procesada.	Se validó almacenamiento de errores y feedback vinculados a cada respuesta (trazabilidad).
RF-16	Historial de errores para estudiante	El estudiante consulta errores/feedback solo de sus respuestas.	Se probó acceso a historial y se validó restricción por usuario.
RF-17	Historial de errores para profesor	El profesor consulta errores/feedback de estudiantes y tareas asociadas.	Se validó consulta agregada por tarea y acceso a detalles por estudiante.
RF-18	Página de ayuda	Existe una sección que describe flujo y funcionalidades.	Se verificó disponibilidad de la ayuda y coherencia con el flujo real de uso.

7.4. Escenarios de Prueba

Se diseñaron escenarios de prueba representativos que cubren los flujos principales del sistema, con el objetivo de verificar el cumplimiento de los requerimientos funcionales definidos. Cada escenario fue construido para evaluar funcionalidades específicas desde la perspectiva de los distintos roles de usuario y asegurar la correcta integración entre los componentes del sistema.

Los escenarios considerados fueron los siguientes:

- **Acceso por rol:** inicio de sesión como estudiante y profesor, validando la autenticación y la visualización de menús y permisos correspondientes a cada rol (RF-01, RF-02).

Figura 7.1: Página de inicio al acceder como estudiante

Figura 7.2: Página de inicio al acceder como profesor

- **Gestión de tarea:** registro de una tarea académica por parte del profesor, así como la carga y visualización del archivo PDF con el enunciado de la actividad (RF-03, RF-04).

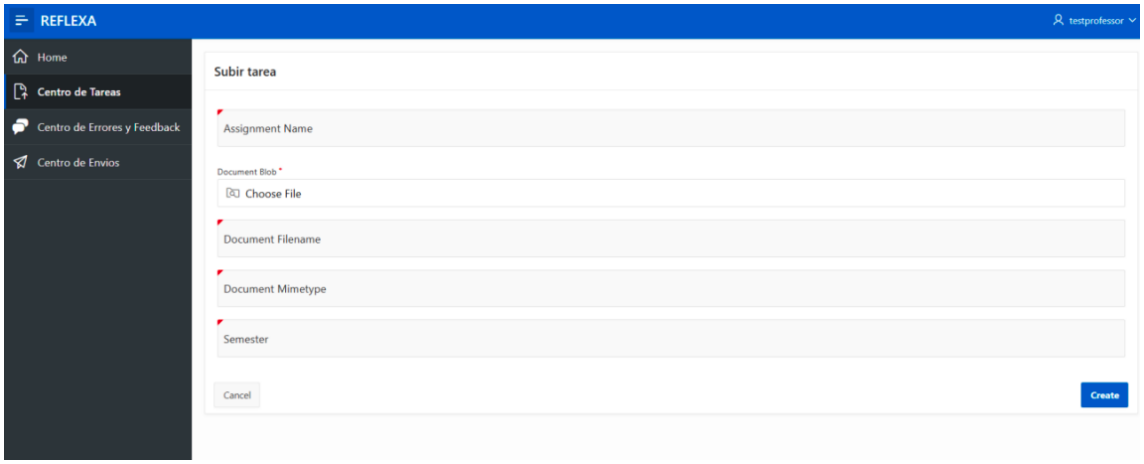

Figura 7.3: Vista para registrar tarea para el rol profesor

- **Diligenciamiento completo:** ingreso de texto por secciones solicitadas en la tarea y consulta simultánea del enunciado durante el diligenciamiento de la respuesta (RF-05, RF-06).

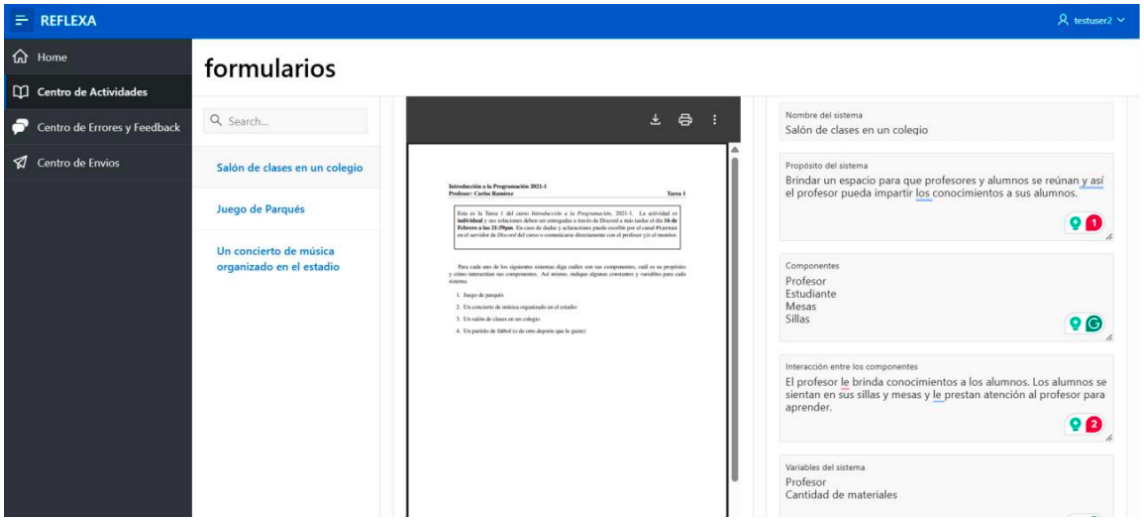

Figura 7.4: Diligenciamiento de la tarea para revisión

- **Envío y persistencia:** envío de la respuesta por parte del estudiante y verificación de su almacenamiento y disponibilidad en el historial de respuestas (RF-07, RF-13).

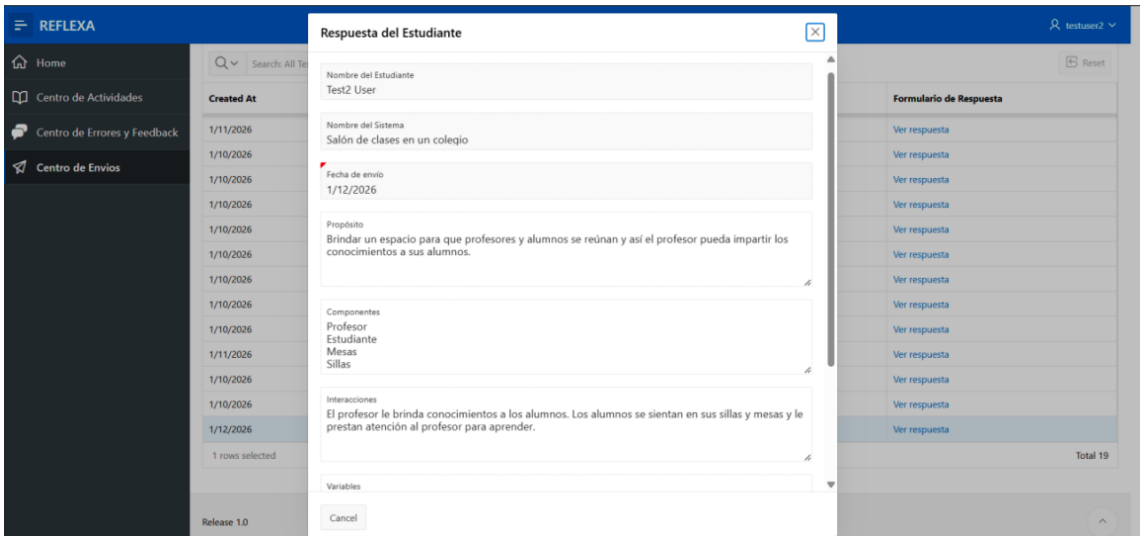

Figura 7.5: Respuesta enviada y cargada correctamente en la plataforma

- **Procesamiento con casos controlados:** envío de respuestas con errores intencionales para validar el procesamiento automático, la identificación y clasificación de errores, así como el registro del valor asociado a cada error detectado (RF-08, RF-09, RF-10).

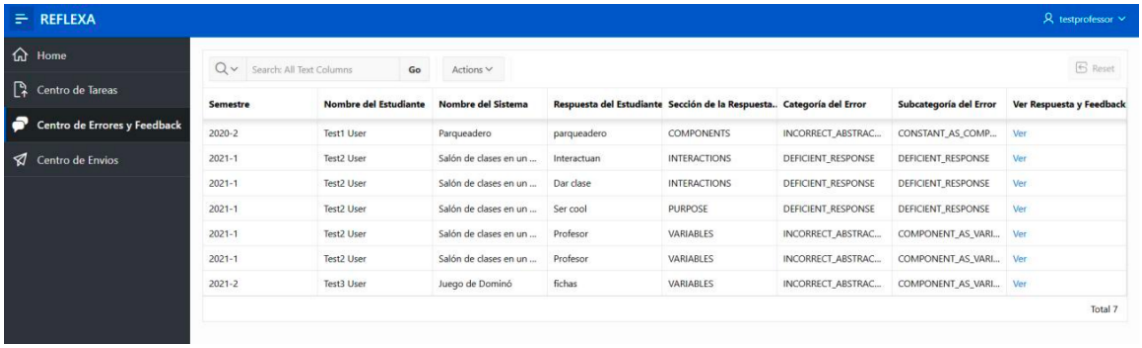

Figura 7.6: Vista de la clasificación de errores en un envío

- **Retroalimentación por error:** procesamiento de respuestas con múltiples errores para validar la generación de retroalimentación individual por error y la trazabilidad entre errores y feedback generado (RF-11, RF-12, RF-15).

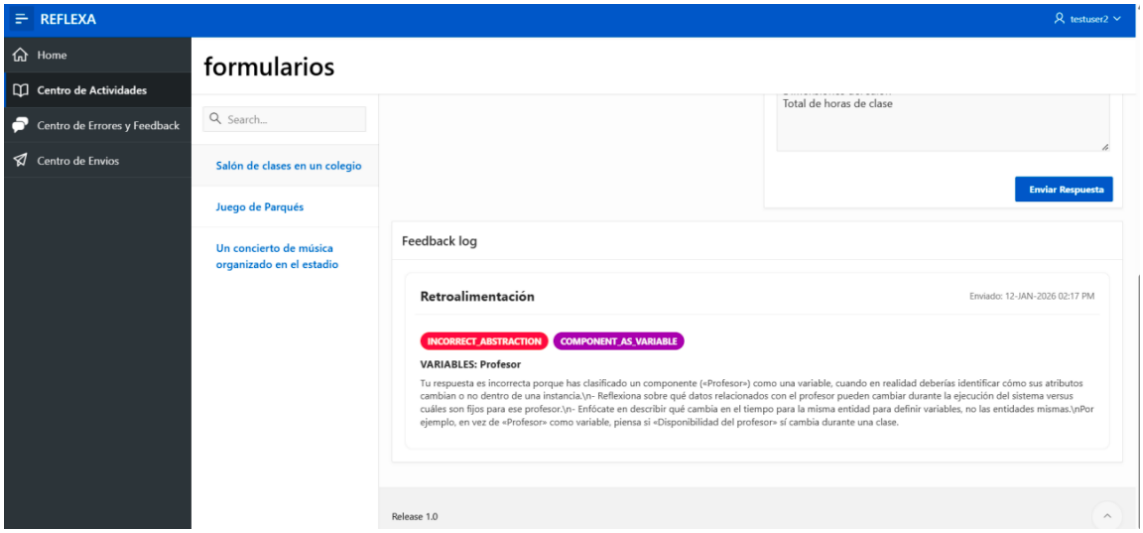

Figura 7.7: Vista de retroalimentación final generada

- **Consulta por profesor:** revisión de respuestas enviadas por los estudiantes, filtradas por tarea, y acceso al historial de errores y retroalimentación correspondiente (RF-14, RF-17).

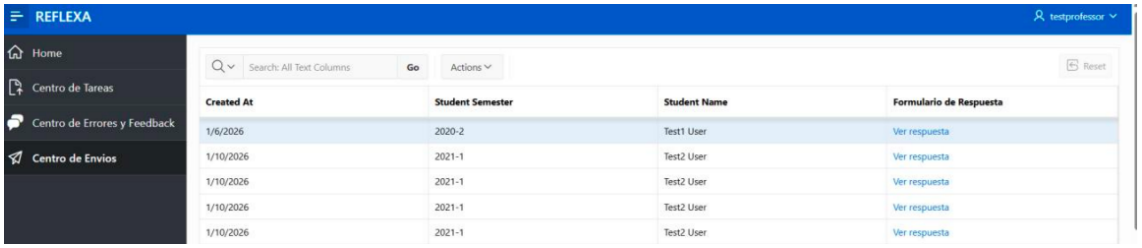

Figura 7.8: Vista de acceso a todos los envíos por parte del profesor

- **Sección de ayuda:** validación del contenido de la sección de ayuda y su correspondencia con el flujo real de uso del sistema para estudiantes y profesores (RF-18).

Figura 7.9: Sección de ayuda se presenta al usuario en la página principal

Cada escenario documentó la entrada proporcionada por el usuario, el comportamiento esperado del sistema, la salida observada tanto en la interfaz como en los registros persistentes, y el estado final de la prueba, clasificado como *aprobado* o *ajuste requerido*.

7.5. Criterios de Evaluación

La evaluación de las pruebas se realizó de manera cualitativa, verificando el cumplimiento de los requisitos funcionales y el comportamiento esperado del sistema. Para cada escenario ejecutado se tuvieron en cuenta los siguientes criterios:

- Ejecución correcta de la funcionalidad evaluada.
- Coherencia del flujo de interacción dentro de la plataforma.
- Correcta generación y visualización de los resultados.
- Almacenamiento adecuado de la información procesada.
- Correspondencia entre el comportamiento observado y el requisito funcional definido.

7.6. Validación de Integración

Finalmente, se ejecutaron pruebas de extremo a extremo para validar la correcta integración entre la plataforma web, la API de clasificación de errores y la base de datos. El flujo validado durante estas pruebas fue el siguiente:

1. El estudiante diligenció y envió una respuesta desde la plataforma web.
2. La plataforma web envió la información a la API de clasificación.
3. La API procesó la respuesta, identificó los errores y generó la retroalimentación correspondiente.
4. La retroalimentación retornó a la plataforma web para su visualización.
5. Se verificó el almacenamiento correcto de la respuesta, los errores detectados y la retroalimentación en la base de datos.

Estas pruebas permitieron comprobar que el sistema operó de forma integrada, sin pérdida de información y manteniendo la trazabilidad entre las respuestas enviadas, los errores detectados y la retroalimentación generada.

7.7. Resultados de las Pruebas

La ejecución de los escenarios de prueba definidos permitió verificar el comportamiento del prototipo en los principales flujos de uso del sistema. En términos generales, las funcionalidades implementadas operaron de acuerdo con lo esperado y de forma coherente con los requerimientos funcionales establecidos para el proyecto.

Durante las pruebas de acceso y gestión de roles, se constató que el sistema diferencia correctamente entre estudiantes y profesores, habilitando únicamente las funcionalidades correspondientes a cada tipo de usuario. Asimismo, los escenarios asociados a la gestión de tareas confirmaron que los profesores pueden registrar tareas académicas, cargar el enunciado en formato PDF y que dicho enunciado se encuentra disponible para los estudiantes durante el diligenciamiento de sus respuestas.

En los escenarios relacionados con el diligenciamiento y envío de respuestas, se verificó que el estudiante puede ingresar información en cada una de las secciones solicitadas, consultar el enunciado de manera simultánea y enviar la respuesta sin inconvenientes. Las respuestas enviadas quedaron almacenadas correctamente y disponibles en el historial del usuario, manteniendo la trazabilidad entre la tarea y la respuesta asociada.

Las pruebas de procesamiento con casos controlados evidenciaron que el sistema identifica y clasifica los errores conceptuales definidos para la tarea, registra el valor asociado a cada error detectado y genera retroalimentación individual por cada uno de ellos. La retroalimentación generada fue presentada de forma clara en la interfaz y quedó almacenada junto con los errores correspondientes, permitiendo su consulta posterior.

Desde la perspectiva del profesor, los escenarios de consulta permitieron verificar que es posible revisar las respuestas enviadas por los estudiantes, filtrarlas por

tarea y acceder al historial de errores y retroalimentación, lo cual facilita el análisis del desempeño académico de los estudiantes. De igual manera, la sección de ayuda mostró coherencia con el flujo real de uso del sistema y proporcionó una guía adecuada para los distintos roles.

Finalmente, las pruebas de integración confirmaron que la comunicación entre la plataforma web, la API de clasificación de errores y la base de datos se ejecuta de manera correcta. El flujo completo de envío, procesamiento, generación de retroalimentación y almacenamiento de la información se mantuvo consistente, sin pérdida de datos ni inconsistencias en los registros.

En conjunto, los resultados obtenidos evidencian que el prototipo funciona de manera integrada y cumple con los requerimientos funcionales definidos, validando su viabilidad como herramienta de apoyo al proceso de retroalimentación académica en el contexto planteado.

7.7.1. Retroalimentación por Parte de los Usuarios

Cinco usuarios participaron en la validación del prototipo. Cuatro de ellos eran estudiantes del programa de Ingeniería de Sistemas y Computación de la Universidad Javeriana Cali, quienes interactuaron directamente con la plataforma enviando respuestas y evaluando su funcionamiento general. Adicionalmente, una estudiante del programa de Psicología de la Universidad Javeriana Cali analizó la retroalimentación generada por el sistema bajo el contexto específico de la tarea, aportando una perspectiva centrada en los aspectos pedagógicos de la retroalimentación. En general, la herramienta fue percibida como útil para identificar errores en las respuestas y orientar al estudiante durante el proceso de aprendizaje; sin embargo, se formularon dos recomendaciones principales para la pedagogía y la experiencia de usuario.

La primera recomendación estuvo relacionada con el contenido de la retroalimentación generada. Los usuarios señalaron que, en algunos casos, la retroalimentación brindaba información demasiado directa, lo que podía interpretarse como una entrega explícita de la respuesta correcta. Por esta razón, se sugirió modificar el enfoque del feedback para que actúe como una guía orientadora, ayudando al estudiante a reconocer el error y reflexionar sobre su respuesta, sin proporcionar directamente la solución. Este ajuste permitiría reforzar el aprendizaje activo y el razonamiento del estudiante.

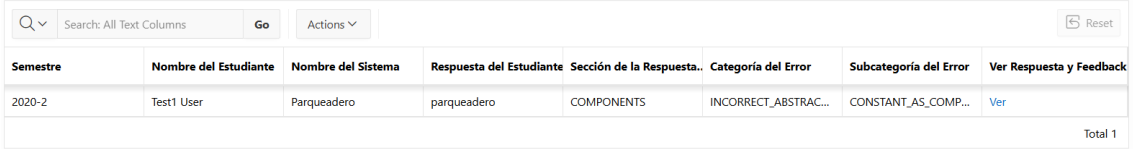

Figura 7.10: Ejemplo de los códigos que se muestran en la interfaz

La segunda recomendación se centró en la interfaz de usuario. Se sugirió que el sistema no muestre los códigos internos de las categorías de error, subcategorías o secciones evaluadas, ya que estos no resultan intuitivos para los usuarios. En su lugar, se recomendó utilizar descripciones en lenguaje natural que expliquen claramente el tipo de error y la sección de la tarea correspondiente, de modo que la información sea más comprensible y cercana para el estudiante.

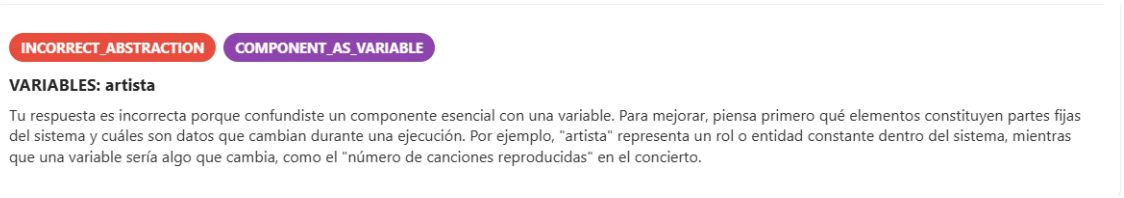

Figura 7.11: Ejemplo de la retroalimentación sugiriendo la respuesta

Capítulo 8

Conclusiones y Trabajos Futuros

8.1. Conclusiones

En el presente trabajo se desarrolló un prototipo funcional orientado al apoyo del proceso de retroalimentación académica mediante la identificación automática de errores conceptuales en respuestas textuales de estudiantes. A lo largo del proyecto se abordaron de manera integral las etapas de definición de requerimientos, diseño arquitectónico, implementación y validación funcional del sistema.

Una de las principales contribuciones del proyecto es la propuesta de una arquitectura modular que separa claramente la interfaz de usuario, la lógica de procesamiento y la generación de retroalimentación. El uso del modelo C4 permitió documentar esta arquitectura de forma progresiva, facilitando la comprensión de la solución desde una visión general hasta su implementación concreta, y evidenciando la correcta integración entre la plataforma web, la API de clasificación y la base de datos.

El prototipo implementado demostró ser capaz de procesar respuestas académicas estructuradas por secciones, identificar errores conceptuales según una taxonomía definida y generar retroalimentación automatizada de manera individual por cada error detectado. Asimismo, se garantizó la trazabilidad entre las respuestas enviadas, los errores identificados y la retroalimentación generada, permitiendo su consulta tanto por parte de estudiantes como de profesores.

Las pruebas funcionales realizadas permitieron verificar el correcto funcionamiento de las principales funcionalidades del sistema y la coherencia de los flujos de interacción para los distintos roles de usuario. Los resultados obtenidos evidencian que el sistema opera de manera integrada y consistente, cumpliendo con los requerimientos funcionales establecidos y validando la viabilidad de la solución propuesta dentro del contexto académico planteado.

Finalmente, este trabajo sienta una base sólida para el desarrollo de soluciones más avanzadas de retroalimentación automatizada en entornos educativos. Si

bien el prototipo presenta limitaciones propias de su alcance académico, los resultados obtenidos confirman el potencial de la combinación de técnicas de clasificación automática y generación de lenguaje natural como herramientas de apoyo al aprendizaje y a la labor docente.

8.2. Trabajos Futuros

A partir del desarrollo del prototipo y de las pruebas realizadas, se identifican diversas líneas de trabajo futuro que permitirían ampliar el alcance del sistema y mejorar su robustez, precisión y aplicabilidad en contextos académicos más amplios.

En primer lugar, una posible extensión del proyecto consiste en la incorporación de **métricas cuantitativas de evaluación**, tanto para el proceso de clasificación de errores como para la calidad de la retroalimentación generada. Esto permitiría medir de forma objetiva el desempeño del sistema, comparar distintos enfoques de clasificación y analizar la efectividad de la retroalimentación desde la perspectiva del aprendizaje del estudiante.

Asimismo, el sistema podría ampliarse para soportar un **mayor número de tareas y dominios académicos**, incorporando nuevas bases referenciales y taxonomías de errores. De esta manera, la solución podría adaptarse a otros cursos o áreas del conocimiento, fortaleciendo su generalización más allá del contexto específico en el que fue validado el prototipo.

Otra línea de trabajo relevante corresponde a la **mejora del proceso de retroalimentación**, explorando estrategias pedagógicas más avanzadas. Esto incluiría la personalización de la retroalimentación según el nivel del estudiante, el historial de errores previos o el progreso a lo largo del curso, así como la incorporación de recomendaciones adaptativas que orienten al estudiante hacia recursos adicionales de aprendizaje.

Desde el punto de vista técnico, se podrían realizar mejoras orientadas a la **optimización del rendimiento y la escalabilidad** del sistema, especialmente para escenarios con un mayor número de usuarios concurrentes. Esto implicaría evaluar alternativas de arquitectura, mecanismos de procesamiento asíncrono y estrategias de cacheo para reducir tiempos de respuesta en entornos de mayor carga.

Finalmente, como trabajo futuro se plantea la posibilidad de realizar **validaciones con usuarios reales en contextos educativos formales**, involucrando a estudiantes y docentes en escenarios de uso prolongado. Este tipo de evaluación permitiría analizar el impacto del sistema en el proceso de aprendizaje, así como recoger retroalimentación cualitativa que contribuya a refinar tanto la funcionalidad como el enfoque pedagógico de la solución.

Estas líneas de trabajo evidencian que el prototipo desarrollado constituye una base sólida sobre la cual es posible construir soluciones más completas y avanzadas, orientadas al apoyo del aprendizaje y a la retroalimentación automatizada en contextos académicos.

Bibliografía

[1] M. N. Giannakos, I. O. Pappas, L. Jaccheri, *et al.*, "Understanding student retention in computer science education: The role of environment, gains, barriers and usefulness," *Education and Information Technologies*, vol. 22, pp. 2365–2382, 2017.

[2] P. H. Winne, "Bootstrapping learner's self-regulated learning," *Psychological Test and Assessment Modeling*, vol. 52, no. 4, pp. 472–490, 2010.

[3] A. Filos-Ratsikas, "Why do students face difficulties in theoretical computer science? a case study," *ACM Transactions on Computing Education*, pp. 1–27, 2021. Article X.

[4] I. Perikos, F. Grivokostopoulou, and I. Hatzilygeroudis, "Assistance and feedback mechanism in an intelligent tutoring system for teaching conversion of natural language into logic," *International Journal of Artificial Intelligence in Education*, vol. 27, pp. 475–514, 2017.

[5] G. Deeva, D. Bogdanova, E. Serral, *et al.*, "A review of automated feedback systems for learners: Classification framework, challenges and opportunities," *Computers & Education*, vol. 162, p. 104094, 2021.

[6] Q. Hao, D. H. Smith, L. Ding, A. Ko, *et al.*, "Towards understanding the effective design of automated formative feedback for programming assignments," *Computer Science Education*, vol. 32, no. 1, pp. 105–127, 2022.

[7] S. Brown, *The C4 Model*. O'Reilly Media, 2022.

[8] W. C. Brandt, "Measuring student success skills: A review of the literature on self-directed learning," tech. rep., National Center for the Improvement of Educational Assessment, Dover, NH, March 31 2020.

[9] E. Kulikova, A. Malinin, and M. Saburova, "A review of automated feedback systems for learners: Classification framework, challenges and opportunities," *International Journal of Educational Technology in Higher Education*, vol. 14, no. 24, pp. 1–20, 2017.

[10] M. Molina, "What is an intelligent system?," *arXiv preprint arXiv:2009.09083v3*, December 2022. Version 3, updated December 2022.

[11] C. Evans, "Making sense of assessment feedback in higher education," *Review of Educational Research*, vol. 83, no. 1, pp. 70–120, 2013.

[12] Z. H. Zhou, *Machine Learning*. Singapore: Springer, 2021.

[13] T. M. Mitchell, "Machine learning," *Annual Review of Computer Science*, vol. 4, no. 1, pp. 417–433, 1990.

[14] A. Shukla, R. Tiwari, and R. Kala, "Adaptive systems," in *Towards Hybrid and Adaptive Computing*, vol. 307 of *Studies in Computational Intelligence*, Berlin, Heidelberg: Springer, 2010.

[15] C. Stryker and J. Holdsworth, "What is NLP (Natural Language Processing)?." `https://www.ibm.com/think/topics/natural-language-processing`, 2024. Accessed: Dec. 22, 2025.

[16] scikit-learn developers, "Tf–idf term weighting." `https://scikit-learn.org/stable/modules/feature_extraction.html#tf-idf-term-weighting`, 2025. Accessed: Dec. 22, 2025.

[17] Daniel Jurafsky and James H. Martin, *Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition, with Language Models*. 3 ed., 2026.

[18] A. A. Awan, "What is Tokenization?." `https://www.datacamp.com/blog/what-is-tokenization`, 2024. Accessed: Dec. 22, 2025.

[19] Kaggle, "Datasets Documentation." `https://www.kaggle.com/docs/datasets`, 2025. Accessed: Dec. 22, 2025.

[20] E. Kavlakoglu, "¿Qué son las máquinas de vector de soporte (SVM)?." `https://www.ibm.com/mx-es/think/topics/support-vector-machine`, 2025. Accessed: Dec. 22, 2025.

[21] C. Grosan and A. Abraham, "Rule-based systems," in *Rule-Based Expert Systems*, Springer, 2011. DOI:10.1007/978-3-642-21004-4_7.

[22] Amazon Web Services, "What is an API (Application Programming Interface)?." `https://aws.amazon.com/what-is/api/`, 2025. Accessed: Dec. 22, 2025.

[23] Google Cloud, "What is Cloud Computing?." `https://cloud.google.com/learn/what-is-cloud-computing`, 2025. Accessed: Dec. 22, 2025.

[24] M. Pavlenko, "Functional Requirements in Software Development: Types and Best Practices." `https://www.altexsoft.com/blog/functional-requirements/`, 2023. Accessed: Dec. 22, 2025.

[25] IBM, "Functional testing." `https://www.ibm.com/think/topics/functional-testing`, 2024. Accessed: Dec. 22, 2025.

[26] DigitalOcean, "What is Cloud Hosting? A Comprehensive Guide to Hosting Services in the Cloud." `https://www.digitalocean.com/resources/articles/cloud-hosting`, 2024. Accessed: Dec. 22, 2025.

[27] J. L. Fernández-Alemán, L. López-González, O. González-Sequeros, *et al.*, "The evaluation of i-sidra – a tool for intelligent feedback – in a course on the anatomy of the locomotor system," *International Journal of Medical Informatics*, vol. 94, pp. 172–181, 2016.

[28] M. T. H. Chi, "Three types of conceptual change: Belief revision, mental model transformation, and categorical shift," *International Handbook of Research on Conceptual Change*, vol. 1, pp. 61–82, 2008.

[29] K. Luneta and J. P. Makonye, "Learner errors and misconceptions in elementary analysis: A case study of a grade 12 class in south africa," *Acta Didactica Napocensia*, vol. 3, no. 3, pp. 35–46, 2010.

[30] L. E. N. Díaz and R. E. L. Martínez, "Identificación de errores en conceptos básicos de principios de programación," *IE Revista de Investigación Educativa de la REDIECH*, vol. 13, p. e1222, 2022.

[31] J. Reason, *Human Error*. Cambridge: Cambridge University Press, 1990. Capítulos 3 y 4.

[32] IEEE Computer Society, "Ieee recommended practice for software requirements specifications," 1998. Reaffirmed 2009.

[33] I. Victoria, "Repositorio para el trabajo de grado." `https://github.com/victor1ahermn/trabajo_de_grado_IVH`, 2025. Repositorio GitHub.

[34] B. Oancea, "Text classification using machine learning methods," *arXiv preprint arXiv:2502.19801*, 2024.

[35] L. Bracken, "Pdf viewer in oracle apex." `https://www.youtube.com/watch?v=PoAl_TAOTxA`, 2021. Oracle Developers, video de YouTube, noviembre de 2021.

[36] S. D'Antoni *et al.*, "Automated feedback generation for open-ended questions: Insights from fine-tuned llms," *Proceedings of the International Conference on Educational Data Mining*, pp. 1–10, 2015.

[37] Oracle Corporation, *About Page Processes*. Oracle, 2023. Oracle APEX 23.2 Documentation.

[38] Oracle Corporation, *Oracle APEX App Builder User's Guide*. Oracle Corporation, 2023.

[39] Oracle Corporation, "Documentación de Oracle Cloud Infrastructure." `https://docs.oracle.com/es-ww/iaas/Content/GSG/Concepts/baremetalintro.htm`, 2025. Accessed: Dec. 22, 2025.

[40] Oracle Corporation, "¿Qué es Oracle Autonomous AI Database?." `https://docs.oracle.com/es-ww/iaas/autonomous-database-serverless/doc/autonomous-intro-adb.html`, 2025. Accessed: Dec. 22, 2025.

[41] Google Research, "Colaboratory Preguntas Frecuentes." `https://research.google.com/colaboratory/faq.html`, 2025. Accessed: Dec. 22, 2025.

[42] Python Software Foundation, *Tutorial de Python*. Python Software Foundation, 2025.

[43] OpenAI, "OpenAI API Reference – Introduction." `https://platform.openai.com/docs/api-reference/introduction`, 2025. Accessed: Dec. 22, 2025.

[44] P. Powell and I. Smalley, "What is Integration Testing?." `https://www.ibm.com/think/topics/integration-testing`, 2025. Accessed: Dec. 22, 2025.