UNIVERSITAT DE BARCELONA

Trabajo final de grado

GRADO DE INFORMÁTICA

Facultad de Matemáticas e Informática
Universidad de Barcelona

Plataforma e-learning con generación y evaluación de cuestionarios mediante LLMs

Autor: Laura Bujalance Castell

Director:	Dr. Eloi Puertas i Prats
Realizado en:	Departamento de Matemáticas e Informática
Barcelona,	10 de junio de 2025

Resumen

La inteligencia artificial ha introducido cambios en el ámbito del *e-learning*, aportando nuevas posibilidades para el diseño y la gestión de recursos educativos. A pesar de que las plataformas digitales educativas han facilitado algunas tareas, la elaboración de cuestionarios personalizados y la retroalimentación al alumnado continúan requiriendo una considerable inversión de tiempo por parte de los docentes. En este contexto, los Modelos de Lenguaje de Gran Tamaño (LLMs) permiten automatizar la generación de preguntas y el acompañamiento individual, facilitando la autorregulación del aprendizaje a través de respuestas personalizadas y recomendaciones adaptadas a cada estudiante.

Este Trabajo Final de Grado presenta el desarrollo de una plataforma educativa que integra LLMs, permite a los profesores generar cuestionarios educativos automáticamente y a los estudiantes realizarlos. La plataforma emplea técnicas de ingeniería de *prompts* que permiten a los docentes crear preguntas de opción múltiple, verdadero/falso y desarrollo, a partir de un documento PDF o utilizando el conocimiento del propio modelo, proporcionándole un tema específico, con posibilidad de ajustar el nivel educativo. Una vez que los alumnos realizan los cuestionarios, obtienen retroalimentación de sus respuestas y la puntuación correspondiente, fomentando así la autorregulación del aprendizaje.

La investigación analiza las capacidades y limitaciones de los LLMs en contextos académicos, así como el uso de estas técnicas en el ámbito educativo. Los resultados muestran que la plataforma facilita la creación de cuestionarios personalizados, aunque requiere supervisión docente para garantizar la calidad del contenido, lo que subraya la importancia del diseño adecuado en la generación de preguntas educativas.

Palabras clave: LLM, modelos de lenguaje de gran tamaño, inteligencia artificial generativa, generación automática de preguntas, retroalimentación, ingeniería de *prompts*, plataforma *e-learning*

Resum

La intel·ligència artificial ha introduït canvis en l'àmbit de l'*e-learning*, aportant noves possibilitats per al disseny i la gestió de recursos educatius. Tot i que les plataformes digitals educatives han facilitat algunes tasques, l'elaboració de qüestionaris personalitzats i la retroalimentació a l'alumnat continuen requerint una inversió considerable de temps per part del professorat. En aquest context, els Models de Llenguatge Extensos (LLMs) permeten automatitzar la generació de preguntes i l'acompanyament individual, afavorint l'autoregulació de l'aprenentatge mitjançant respostes personalitzades i recomanacions adaptades a cada estudiant.

Aquest Treball Final de Grau presenta el desenvolupament d'una plataforma *e-learning* que integra LLMs, permet als professors generar qüestionaris educatius automàticament i als estudiants realitzar-los. La plataforma utilitza tècniques d'enginyeria de *prompts* que permeten als docents crear preguntes d'opció múltiple, veritable/fals i desenvolupament, a partir d'un document PDF o utilitzant el coneixement del propi model, proporcionant-li un tema específic, amb possibilitat d'ajustar el nivell educatiu. Una vegada que els alumnes realitzen els qüestionaris, obtenen retroalimentació de les seves respostes i la puntuació corresponent, fomentant així l'autoregulació de l'aprenentatge.

La recerca analitza les capacitats i limitacions dels LLMs en contextos acadèmics, així com l'ús d'aquestes tècniques en l'àmbit educatiu. Els resultats mostren que la plataforma facilita la creació de qüestionaris personalitzats, encara que requereix supervisió docent per a garantir la qualitat del contingut, fet que posa de manifest la importància del disseny adequat en la generació de preguntes educatives.

Paraules clau: LLM, models de llenguatge extensos, intel·ligència artificial generativa, generació automàtica de preguntes, retroalimentació, enginyeria de *prompts*, plataforma *e-learning*

Abstract

Artificial intelligence has introduced changes in the field of e-learning, offering new possibilities for the design and management of educational resources. Although educational digital platforms have simplified certain tasks, the creation of personalized quizzes and the provision of feedback to students still require a considerable investment of time by teachers. In this context, Large Language Models (LLMs) make it possible to automate question generation and individual support, facilitating self-regulated learning through personalized responses and recommendations tailored to each student.

This Final Degree Project presents the development of an e-learning platform that integrates LLMs, enabling teachers to automatically generate educational quizzes and students to complete them. The platform employs prompt engineering techniques that allow educators to create multiple-choice, true/false, and open-ended questions based on a PDF document or using the model's own knowledge, by providing a specific topic and adjusting the educational level. Once students complete the quizzes, they receive feedback on their answers and corresponding scores, thereby promoting self-regulated learning.

The research analyzes the capabilities and limitations of LLMs in academic contexts, as well as the use of these techniques in the educational field. The results show that the platform facilitates the creation of personalized quizzes, although teacher supervision is necessary to ensure the quality of the content, highlighting the importance of proper design in the generation of educational questions.

Keywords: LLM, large language models, generative artificial intelligence, automatic question generation, feedback, prompt engineering, e-learning platform.

Agradecimientos

Quiero comenzar agradeciendo a mi tutor, Eloi Puertas, por acompañarme durante todo el proceso, por su dedicación, su disponibilidad y por ofrecerme herramientas que han sido imprescindibles para avanzar en este proyecto.

A mi pareja, Alex, por su comprensión y afecto, por animarme a seguir adelante incluso cuando yo dudaba. Este logro también es suyo.

A mi madre y a mi hermano, por quererme sin medida y por darlo todo sin esperar nada. Su confianza en mí me ha dado fuerza en los momentos más exigentes.

Índice

1. Introducción

Las plataformas de aprendizaje digital (plataformas *e-learning*) han incorporado gradualmente sistemas de colaboración más avanzados entre docentes y estudiantes. La integración de herramientas de inteligencia artificial (IA) en estos entornos ha permitido el desarrollo de funcionalidades que van desde la generación automática de contenido educativo hasta asistentes virtuales, promoviendo así la autorregulación del aprendizaje [1].

El uso de estas tecnologías permite adaptar los contenidos y las actividades a las particularidades de cada estudiante, facilitando un aprendizaje más personalizado. Además, al automatizar tareas repetitivas y ofrecer respuestas inmediatas, se libera tiempo para que los docentes puedan dedicarse a mejorar la experiencia educativa y acompañar mejor a sus estudiantes [2].

Los Modelos de Lenguaje de Gran Tamaño (LLMs, por sus siglas en inglés) permiten crear cuestionarios con preguntas de distintos tipos a partir del contenido proporcionado por los docentes y adaptar su dificultad según las indicaciones, ofreciendo retroalimención (*feedback*) basado en las respuestas del estudiante [3]. A diferencia de los modelos discriminativos, que requieren conjuntos de datos etiquetados y procesos de entrenamiento complejos, un LLM genera contenido nuevo a partir de patrones aprendidos, lo que facilita su uso sin necesidad de entrenar el modelo desde cero mediante la formulación de instrucciones (*prompts*) que guían la respuesta [4].

Este trabajo final de grado plantea el desarrollo de una plataforma educativa que integra un componente investigativo con una solución tecnológica práctica. Se investigan las capacidades y limitaciones de los LLMs aplicados en el ámbito educativo, así como las estrategias de ingeniería de *prompts* para mejorar la calidad de las preguntas generadas, tanto de contenido específico como del conocimiento previo del modelo. Los resultados de esta investigación ofrecen información sobre la fiabilidad de los modelos y los criterios para valorar la calidad de las preguntas.

En términos de implementación práctica, la plataforma ofrece una experiencia personalizable donde el profesor define parámetros como tema, nivel académico y cantidad de preguntas de opción múltiple, desarrollo y verdadero o falso, obteniendo cuestionarios ajustados a distintos grados de dificultad. El sistema permite cargar documentos PDF para generar preguntas, y ofrece al alumno autoevaluaciones con explicaciones personalizadas basadas en el contenido.

1.1. Motivación

Actualmente, las plataformas educativas han avanzado mucho gracias a la integración de inteligencia artificial, pero todavía hay un margen amplio para mejorar la forma en que los docentes pueden adaptar los recursos a las necesidades de sus estudiantes [1]. Si bien la automatización de la generación y corrección de cuestionarios contribuye a aliviar la carga administrativa y facilita la personalización, la mayoría de soluciones disponibles no permiten al profesorado ajustar fácilmente el tipo, la dificultad o el contenido de las preguntas de forma intuitiva [2].

Este proyecto propone una herramienta donde cualquier docente pueda crear evaluaciones adaptadas a su asignatura sin requerir experiencia técnica avanzada [3]. El sistema se centra en ofrecer una experiencia intuitiva, ya que solo es necesario indicar el tema, el nivel académico y el número de preguntas de cada tipo, permitiendo que la tecnología se adapte a la pedagogía en lugar de condicionar la práctica docente.

El uso de LLMs permite la generación de preguntas personalizadas incluso a partir de documentos PDF, lo que amplía el abanico de recursos y ahorra tiempo a los docentes [5].

Sin embargo, no basta con automatizar la generación, ya que es importante que los profesores puedan guiar el resultado mediante instrucciones, asegurando que los cuestionarios respondan a los objetivos pedagógicos de cada materia [6].

Por otra parte, esta herramienta busca fomentar la autonomía del estudiante mediante la autoevaluación y el *feedback*. Así, cada alumno puede avanzar a su propio ritmo, identificar sus puntos débiles y mejorar de forma progresiva, mientras el profesor puede revisar o ajustar cualquier pregunta generada.

El desarrollo de este proyecto constituye una oportunidad para trasladar a la práctica los conocimientos adquiridos durante la carrera, integrando tecnologías actuales como los modelos de lenguaje y la ingeniería de *prompts*.

1.2. Objetivos del proyecto

A continuación se presentan los objetivos del proyecto, comenzando por los generales que marcan la finalidad principal, seguidos de los específicos que detallan las funciones y desarrollos previstos.

Objetivos Generales

- Crear una plataforma *e-learning* donde los profesores pueden generar cuestionarios educativos mediante inteligencia artificial.
- Permitir a los estudiantes autoevaluarse y recibir *feedback* mediante inteligencia artificial.
- Analizar las capacidades y limitaciones de los LLMs aplicados en educación.
- Profundizar en la ingeniería de *prompts* para mejorar la forma en que los modelos responden.

Objetivos Específicos

- Diseñar una interfaz web utilizando Vue.js y TailwindCSS, con vistas diferenciadas para docentes y estudiantes que sea intuitiva y adaptada a distintas resoluciones.
- Implementar una plataforma que, conectada con un LLM, genere automáticamente preguntas (opción múltiple, verdadero/falso y desarrollo de respuesta corta) a partir de un tema, utilizando técnicas de ingeniería de *prompts* para optimizar la calidad del contenido.
- Desarrollar el *backend* utilizando Python, LangChain y FastAPI para integrar modelos LLM y ofrecer una API REST.
- Evaluar la calidad de las preguntas generadas mediante métricas apropiadas.

2. Conceptos previos

Este apartado establece las bases para comprender el desarrollo de la plataforma educativa propuesta. Se abordan los Modelos de Lenguaje de Gran Tamaño (LLMs), incluyendo su funcionamiento y problemáticas asociadas.

Se explica también el concepto de agentes basados en LLMs y las técnicas de ingeniería de *prompts* para optimizar la interacción con estos modelos. Finalmente, se contextualizan las plataformas de *e-learning*, estableciendo el marco educativo en el que se integra la solución desarrollada.

2.1. Arquitectura de los LLMs

El desarrollo de modelos generativos mediante aprendizaje profundo (*deep learning*) ha permitido la creación de sistemas capaces de producir contenido a partir de grandes volúmenes de datos lo que se conoce como Modelos de Lenguaje de Gran Tamaño (LLM). Entre los primeros avances en este campo destacan las redes generativas antagónicas (GAN, en inglés) [7], que introducen una red generadora y otra discriminadora, como se muestra en la Figura 1. Aunque estas redes demostraron eficacia en la generación de imágenes y también pueden utilizarse para otros tipos de datos, como música y vídeo, presentan limitaciones en el procesamiento del lenguaje natural (PLN), sobre todo en la generación de texto coherente, debido a su limitación para modelar adecuadamente las dependencias secuenciales propias del lenguaje [7].

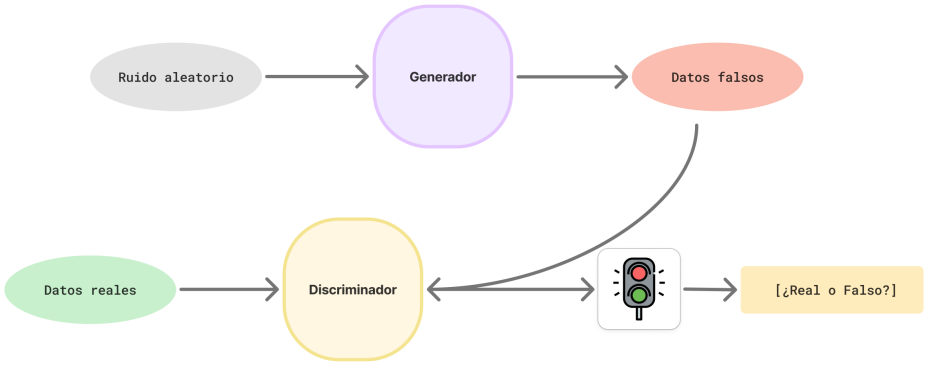

Figura 1: Funcionamiento básico de una red generativa antagónica (GAN)

La arquitectura *Transformer* [8], introducida en 2017, ha cambiado el tratamiento de secuencias en PLN al superar estas limitaciones mediante mecanismos de atención que analizan las relaciones entre todos los elementos de una secuencia de forma paralela, mejorando así la comprensión del contexto.

Su diseño se basa en dos componentes, como se muestra en la Figura 2. El *encoder*, que recibe la secuencia de entrada y la transforma en una representación interna que resume la información y las relaciones presentes en el texto original, y el *decoder*, que utiliza esta representación para generar secuencias de salida manteniendo la coherencia mediante el acceso tanto a la entrada codificada como al texto previamente generado [8].

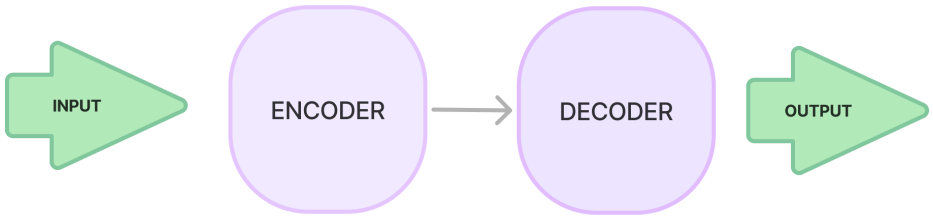

Figura 2: Arquitectura Transformer

Dentro de la familia de modelos basados en *Transformer*, el enfoque que ha revolucionado la generación de lenguaje natural es GPT (Generative Pre-trained Transformer) [9], desarrollado por *OpenAI*. Este modelo utiliza solo el componente *decoder* de la arquitectura *Transformer* y, a diferencia de otros modelos enfocados en comprensión, como BERT [10] de Google, GPT demuestra mejor rendimiento en generación de lenguaje natural.

Este tipo de modelos, entrenados con grandes volúmenes de datos y estructurados sobre arquitecturas como *Transformer*, constituyen la base de los actuales LLMs.

2.2. Procesamiento de texto en LLMs

En el procesamiento del lenguaje natural (PLN), los LLMs trabajan con unidades llamadas *tokens* [11], que pueden ser una palabra completa, una parte de una palabra o incluso un grupo de caracteres, según cómo esté diseñado el sistema. El proceso de dividir el texto en estos fragmentos se conoce como tokenización y es necesario para que el modelo pueda entender o generar texto. La cantidad de *tokens* que un modelo puede procesar varía según la arquitectura, por lo que saber cuántos *tokens* tiene un texto ayuda a gestionar los recursos computacionales.

Durante la generación de texto, estos modelos procesan las secuencias de *tokens* de manera secuencial, seleccionando cada nuevo elemento según el contexto previo acumulado [11]. Esta predicción incluye un nivel de aleatoriedad, regulado por el parámetro *temperature*, que regula el nivel de creatividad de la respuesta. Cuando la temperatura es baja, el modelo selecciona el token más probable, mientras que aumentar la temperatura puede producir resultados más aleatorios [12].

Una vez que el texto ha sido *tokenizado*, cada uno de estos *tokens* se convierte en una representación numérica conocida como *embedding* [13], que permite al modelo operar matemáticamente sobre el lenguaje. Estos *embeddings* son representaciones aprendidas durante el entrenamiento y muestran relaciones semánticas y sintácticas entre palabras. Así, términos con significados o contextos similares quedan cerca unos de otros en el espacio vectorial. A diferencia de los modelos discriminativos, los Transformers generan *embeddings* contextuales, lo que significa que la representación de un token varía según el contexto en el que aparece.

2.3. Problemáticas de los LLMs

Los modelos de tipo *Transformer* presentan algunos desafíos, entre ellos lo que se conoce como *alucinaciones* [11] que son palabras o frases generadas por el modelo que a menudo carecen de sentido o son gramaticalmente incorrectas.

Las *alucinaciones* pueden deberse a una variedad de factores. Algunas de las causas más comunes que pueden generar una salida de baja calidad son las siguientes [11]:

- El modelo no ha sido entrenado con suficientes datos.
- Los datos utilizados para el entrenamiento presentan ruido o son erróneos.
- No se proporciona suficiente contexto o restricciones durante la generación.

Otra cuestión relevante es la discriminación algorítmica, derivada de la presencia de sesgos en los datos de entrenamiento, que puede conducir a conclusiones erróneas para determinados grupos [14].

Igualmente relevante es el *AI alignment*, que ocurre cuando el modelo genera resultados que no se ajustan exactamente a las instrucciones. Además, los modelos pueden experimentar el *AI lock-in problem*, quedándose anclados en un enfoque y reduciendo la diversidad de respuestas [14].

Por otro lado, a medida que se desarrollan modelos cada vez más grandes, la necesidad de GPUs potentes ha aumentado de forma exponencial. La demanda de este tipo de hardware supera con frecuencia la oferta, lo que provoca un aumento sostenido en los precios y convierte al mercado de GPUs en un entorno altamente competitivo [11, 15].

2.4. Agentes

En el contexto de los LLMs, un agente es un sistema capaz de interpretar instrucciones, analizar información disponible y ejecutar una secuencia de acciones que buscan alcanzar un objetivo definido. Estos sistemas combinan la generación de lenguaje natural con procesos de razonamiento, toma de decisiones y, depende el caso, la interacción con herramientas externas como bases de datos. El funcionamiento de un agente suele apoyarse en estrategias como la ingeniería de *prompts* y la generación aumentada por recuperación (RAG, por sus siglas en inglés) [16].

2.5. Prompt engineering

Un *prompt* es la entrada textual que guía el comportamiento del LLM, y su formulación determina directamente la calidad de las respuestas generadas. La ingeniería de *prompts* (*prompt engineering*) se ocupa del diseño de los *prompts* que se proporcionan a los modelos de lenguaje de gran tamaño [17].

La efectividad de un *prompt* depende de su capacidad para comunicar claramente la tarea deseada al modelo. Pequeñas variaciones en la redacción pueden producir diferencias importantes en los resultados, lo que convierte el diseño de *prompts* en un proceso que requiere precisión y conocimiento del comportamiento del sistema [17].

Un *prompt* efectivo suele componerse de varios elementos que pueden combinarse en función del resultado deseado:

- **Instrucción principal:** Define de forma clara y precisa la tarea que debe ejecutar el modelo.
- **Contexto:** Proporciona información relevante para que el modelo comprenda mejor la solicitud.
- **Formato:** Estructura o tipo de respuesta esperada.
- **Restricciones:** Especifican condiciones como la longitud, el vocabulario, el estilo, etc.
- **Rol o perspectiva:** Indican el tono, nivel de especialización o punto de vista desde el cual debe responder el modelo.

La ingeniería de *prompts* es un proceso iterativo, pocas veces se obtiene el resultado deseado en el primer intento, por lo que es necesario probar distintas formulaciones, ajustar el contexto y evaluar el rendimiento del modelo hasta conseguir la respuesta esperada [17].

Para facilitar este proceso, se han desarrollado plantillas y patrones de *prompting* que posibilitan abordar tareas comunes para mejorar la eficiencia en las interacciones con los modelos [17].

Además, cuando se requiere mantener coherencia a lo largo de múltiples intercambios, por ejemplo en conversaciones con el modelo, es importante incorporar de forma adecuada el contexto previo, instrucciones anteriores o información relevante para evitar repeticiones y contradicciones en las respuestas generadas [17].

2.6. Métodos para mejorar la precisión de los LLMs

Además del diseño de *prompts*, existen otras técnicas para adaptar los modelos a tareas específicas, como el ajuste fino (*fine-tuning*) [18], que implica reentrenar el modelo con datos propios de una temática concreta, o el aprendizaje por refuerzo con retroalimentación humana (*Reinforcement Learning with Human Feedback*, RLHF) [18], donde las personas evalúan y ajustan las respuestas del modelo para mejorar su precisión. Otra técnica es la generación aumentada por recuperación (*Retrieval-Augmented Generation*, RAG) [18], que permite al modelo acceder a fuentes externas de información actualizadas. De esta manera, el modelo puede

complementar su conocimiento interno con datos más recientes y ofrecer respuestas más precisas y verificadas.

2.7. Plataformas e-learning

Las plataformas *e-learning* [19] son entornos digitales diseñados para facilitar el acceso a contenidos educativos mediante tecnologías electrónicas. Estas plataformas suelen organizarse a través de sistemas de gestión del aprendizaje (LMS, por sus siglas en inglés), los cuales facilitan administrar cursos, distribuir contenidos, realizar seguimientos del progreso estudiantil y gestionar evaluaciones en línea [1].

La calidad de la experiencia de aprendizaje en estas plataformas depende en gran medida del diseño instruccional, es decir, de cómo se estructuran los contenidos, se gestionan las interacciones y se adaptan los métodos pedagógicos al entorno digital [20].

Las limitaciones más frecuentes de las plataformas *e-learning* se encuentran entre las más frecuentes los problemas técnicos, la falta de interacción directa con los docentes, la necesidad de una alta autodisciplina y las desigualdades en el acceso a dispositivos y conectividad. También pueden surgir dificultades relacionadas con el nivel de competencia digital de algunos usuarios, limitando su capacidad de desenvolverse en estos entornos. [19].

3. Estado del arte

En esta sección se presentan investigaciones recientes que han contribuido al avance en el uso de LLMs en entornos educativos, organizados en tres áreas principales. Primero, se examinan las aplicaciones prácticas de los LLMs en el contexto de generación de preguntas y *feedback* en el ámbito educativo. Después, se incluyen los métodos de *prompt engineering* desarrollados para contextos educativos, incluyendo marcos teóricos y técnicas que han demostrado efectividad en la mejora de la calidad de las respuestas generadas. Finalmente, se explica cómo han integrado estas tecnologías las plataformas *e-learning*.

3.1. Aplicaciones de LLMs en entornos educativos

3.1.1. LLMs para evaluación automática

Estudios comparativos entre GPT-4 y expertos en educación para la evaluación académica de exámenes han mostrado que el modelo puede asignar puntuaciones consistentes siempre que se le proporcione información detallada del curso y una rúbrica clara [21].

No obstante, se ha observado que los LLMs presentan ciertas limitaciones [22]:

- **Dependencia de fuentes públicas:** Un LLM genera respuestas principalmente basándose en información pública accesible porque es con lo que han sido preentrenados.
- **Dificultad con situaciones o debates específicos en clase:** Cuando los exámenes plantean problemas particulares o discusiones internas de clase que no están disponibles públicamente, los LLMs presentan dificultades para responder con precisión.
- **Limitación en interpretación gráfica o experimental:** La inclusión de gráficos, diagramas o datos experimentales representa una dificultad para los LLMs, ya que su capacidad para interpretar estos elementos es limitada.
- **Vulnerabilidad a información falsa o contradictoria:** Introducir afirmaciones incorrectas o engañosas puede confundir a los LLMs, revelando su falta de capacidad crítica.
- **Dificultad en preguntas basadas en probabilidades estadísticas:** Los

modelos pueden mostrar dificultades en responder adecuadamente cuando las preguntas requieren análisis profundo o pensamiento probabilístico preciso.

Por otra parte, se ha visto que, aunque los LLMs pueden identificar correctamente la respuesta a una pregunta de opción múltiple, su desempeño puede verse alterado si en lugar de referirse a las opciones como A, B, C o D, que son las más frecuentes en conjuntos de datos de entrenamiento, se reemplazan por símbolos poco habituales, como Q, R, S o T, algunos modelos tienden a confundirse, incluso cuando reconocen cuál es la opción correcta. Se puede interpretar que, en algunos casos, los LLMs no comprenden completamente la estructura de la pregunta y responden guiados por patrones aprendidos durante el entrenamiento [23].

Además, los modelos de lenguaje actuales no tienen la capacidad de evaluar habilidades interpersonales o sociales como la colaboración en grupo, la negociación o el liderazgo en la toma de decisiones. Estas competencias requieren interacción humana real, algo que los LLMs no pueden observar ni participar. Por esta razón, se recomienda complementar las evaluaciones automatizadas con actividades prácticas, como trabajos en grupo, presentaciones orales o simulaciones, donde los estudiantes deban comunicarse, coordinarse y resolver problemas en conjunto [22].

3.1.2. LLMs en generación de preguntas

En cuanto a la generación de contenido, los LLMs pueden servir como punto de partida para generar preguntas proporcionando buenos resultados. Sin embargo, las preguntas se han de adaptar al contexto específico de cada asignatura, incluyendo terminología, notaciones y enfoques pedagógicos propios. Aunque el modelo puede producir cuestionarios siguiendo estas instrucciones, la revisión y ajuste por parte del profesorado garantizan la calidad técnica y pedagógica [24].

Por otro lado, se ha analizado la capacidad de los modelos de lenguaje para generar preguntas de calidad a partir de textos extraídos de Wikipedia, comparándolas con preguntas hechas por educadores. Para ello, se evaluaron seis aspectos, el tipo de pregunta, la longitud, la cobertura del contexto, la capacidad de ser respondida con el texto, la rareza (si requiere conocimiento no común) y la longitud de la respuesta necesaria [25].

Los resultados muestran que los LLMs tienden a generar preguntas más descriptivas, que requieren respuestas más largas y detalladas. A diferencia de los profesores, no se centran solo en el inicio del texto, sino que distribuyen su atención de forma más equilibrada. Además, muchas de las preguntas no pueden responderse sin

el contexto, lo que pueden ser útiles para evaluar sistemas como RAG o detectar alucinaciones en modelos de lenguaje [25].

De forma complementaria, un estudio reciente utilizó GPT-4o para generar un test completo de opción múltiple para un curso universitario de PLN. Las preguntas se generaron directamente a partir de fragmentos del temario seleccionados por el docente, asegurando su alineación con los objetivos de aprendizaje. Para mantener coherencia y calidad en el formato, se utilizó una plantilla de *prompt* estructurado y se forzó la salida en formato JSON mediante el *framework LangChain* junto con *Pydantic*. El test fue revisado por expertos y aplicado a estudiantes, quienes también evaluaron cada ítem. Los resultados mostraron que las preguntas generadas presentaban un nivel adecuado de dificultad y discriminación, y fueron bien valoradas en términos de claridad y utilidad. No obstante, se identificaron algunos casos de sesgo o ambigüedad, lo que reafirma que es necesaria la supervisión por parte de un experto en el proceso [26].

3.1.3. LLMs para retroalimentación educativa

Respecto ofrecer *feedback*, uno de los aspectos más valorados de los LLM es su capacidad para ofrecer *feedback* de forma inmediata. Algunos estudios, han demostrado que la IA puede imitar patrones discursivos utilizados por docentes humanos, como la reformulación de respuestas, la sugerencia de estrategias de mejora o la confirmación de respuestas correctas. Esta capacidad de actuar como un profesor contribuye a crear una experiencia más interactiva y motivadora para el alumno, especialmente en contextos donde la supervisión directa del profesorado no es continua [27].

Sin embargo, esta capacidad presenta importantes limitaciones, ya que las explicaciones que generan suelen ser demasiado genéricas o repetitivas, y no logran abordar adecuadamente los errores conceptuales específicos de cada estudiante. En este sentido, la técnica RAG ha mostrado buenos resultados al permitir al modelo consultar materiales concretos del curso, aunque persisten limitaciones en la precisión con la que el modelo interpreta y organiza la información del curso, lo que dificulta su uso en sistemas de aprendizaje adaptativo sin la guía de una persona experta [24].

Por el contrario, cuando se utilizan *prompts* bien estructurados, es posible guiar al modelo para que genere *feedback* que no solo indique si una respuesta es correcta o incorrecta, sino que también explique el porqué, ofrezca pistas o proponga ejemplos

complementarios para fomentar el aprendizaje autónomo [27].

3.2. Prompt engineering en educación

3.2.1. Fundamentos del diseño de *prompts* en educación

La ingeniería de *prompts* aplicada a la educación ha desarrollado métodos para optimizar la generación de contenido educativo mediante modelos de lenguaje. Entre los marcos más utilizados destaca TCI + RATEL, que estructura *prompts* combinando componentes básicos como Tarea, Contexto, Instrucciones, con elementos secundarios como Rol, Audiencia, Tono, Ejemplos y Límites. Esta estructura facilita tareas como la planificación de contenidos, el diseño de actividades y la creación de criterios de evaluación [28].

También se han desarrollado marcos como CRISPE, CLEAR y AIPROMT, que actúan como guías para diseñar *prompts* de manera organizada. El marco CRISPE incluye los elementos *capacidad y rol*, *perspectiva*, *instrucción*, *personalidad* y *experimentación*, y propone estructurar las instrucciones en cinco etapas. Por su parte, CLEAR pone énfasis en la *claridad*, la *lógica*, el uso de *ejemplos*, la *evitación de ambigüedad* y la *reflexión*. El modelo AIPROMT plantea siete componentes, *audiencia*, *intención*, *instrucción*, *rol*, *tipo de salida*, *métricas* y *ajustes*. Este último está orientado a usuarios que desean ajustar el modelo en tareas educativas, como la generación de texto, la elaboración de resúmenes o la retroalimentación. [29].

Desde el enfoque docente, se ha planteado un modelo práctico para mejorar el diseño de *prompts* en educación, conocido como IDEA. Este modelo reúne tres propuestas complementarias que buscan ayudar a los profesores a construir mejores instrucciones para modelos de lenguaje. La primera se llama PARTS y propone que todo *prompt* debe incluir una descripción clara de quién habla, cuál es el propósito de la tarea, a quién va dirigida la respuesta, qué tema debe abordar el modelo y qué estructura se espera en la salida. La segunda parte del modelo se basa en principios para escribir buenos *prompts*, como ser concisos, lógicos, explícitos, adaptables y precisos. Finalmente, la tercera propuesta es un proceso de mejora continua llamado REFINE, que consiste en reformular las instrucciones, experimentar con diferentes variantes, observar los resultados, pedir *feedback*, ajustar según lo necesario y volver a evaluar. Este conjunto de estrategias tiene como objetivo formar a los docentes en la creación de *prompts* [30].

3.2.2. Técnicas específicas de prompting

Los *prompts* también se clasifican según el número de ejemplos que incluyen técnicas conocidas como técnica de "shots". El zero-shot prompting no proporciona ejemplos al modelo, mientras que el few-shot prompting incluye algunos ejemplos para guiar la respuesta [29]. La comparativa entre las dos técnicas se muestra en la Figura 3.

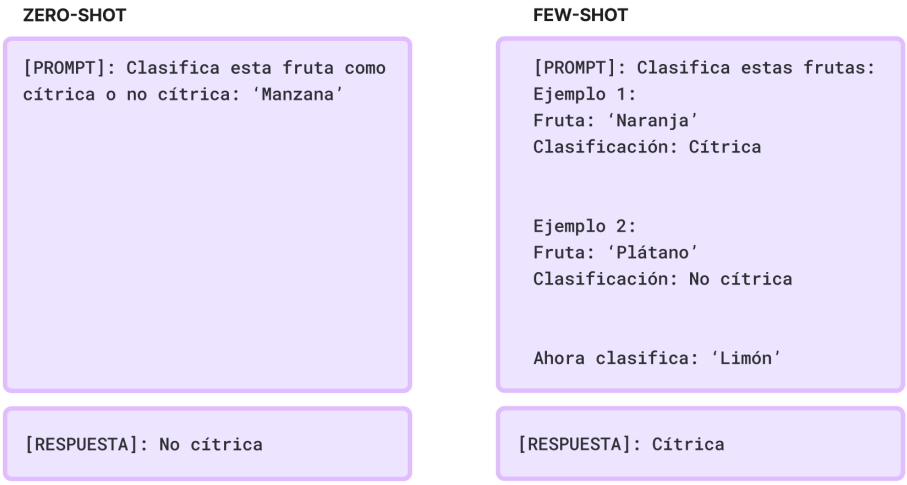

Figura 3: Comparación entre zero-shot y few-shot

El chain-of-thought prompting busca que el modelo razone paso a paso, y ha demostrado mejorar la calidad de las respuestas en tareas que requieren análisis [29, 33]. La comparativa entre respuesta directa (zero-shot) y chain-of-thought se muestra en la Figura 4. Una reciente investigación respalda este enfoque, mostrando que cuando los modelos de lenguaje generan razonamientos paso a paso antes de responder preguntas de opción múltiple, tienden a tener más confianza en sus respuestas. Este fenómeno fue observado en múltiples modelos de lenguaje evaluados con el benchmark MMLU (Massive Multitask Language Understanding), una prueba estandarizada que evalua la capacidad de los modelos para razonar y aplicar conocimientos en una amplia variedad de materias. Los resultados de este estudio indican que el razonamiento previo incrementa la autoconfianza del modelo en sus

respuestas, incluso cuando estas son incorrectas, lo cual plantea desafíos en el uso de estas herramientas [31].

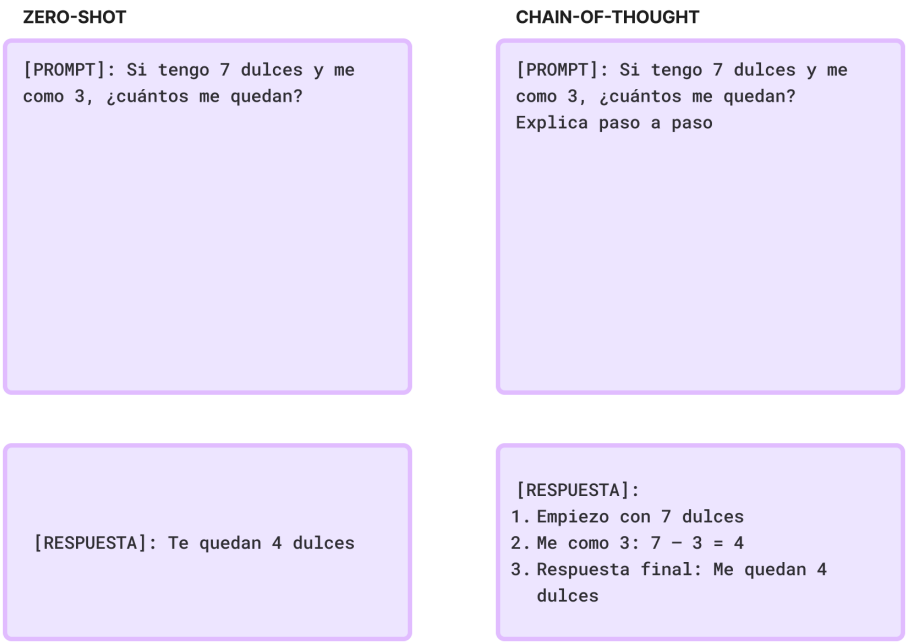

Figura 4: Comparación entre zero-shot y chain-of-thought

Existen técnicas más avanzadas como el *Role-Based Prompting*, que consiste en asignar al modelo un rol según el contexto, como "profesor", "tutor" o "experto en educación". Esta se ha aplicado con éxito en simulaciones de aula y generación de *feedback* y se ha visto que esta estrategia mejora la relevancia, el tono y la alineación de las respuestas con los objetivos pedagógicos [28, 29].

El *prompting* iterativo es otra estrategia utilizada donde el usuario mejora gradualmente sus instrucciones según las respuestas que obtiene del modelo. Esta técnica se aplica en áreas creativas y de formación práctica, como arte, escritura o educación literaria, ya que permite adaptar el modelo a tareas más complejas [29].

En la creación y evaluación automatizada de contenido educativo, como cuestionarios de opción múltiple, los *prompts* se diseñan mediante plantillas que permiten personalizar variables como el número de preguntas, el nivel de dificultad, el tema o el formato de salida. Estas plantillas, integradas mediante el *framework Langchain*, permiten generar instrucciones consistentes y reutilizables, facilitando la interacción con modelos como Gemini o GPT [32].

Una vez generado el cuestionario, es una práctica útil realizar un segundo prompt de evaluación que analiza automáticamente la calidad de las preguntas. Este prompt revisa si las preguntas se repiten, si son adecuadas para el nivel cognitivo definido y si las opciones de respuesta presentan una dificultad equilibrada. Además, se puede evaluar si las preguntas se ajustan al nivel cognitivo esperado del estudiante y propone ajustes cuando detecta desviaciones [32].

3.2.3. Calidad de las respuestas

Respecto al *feedback*, se ha demostrado que los *prompts* bien construidos pueden dar lugar a respuestas automatizadas que muestran un tono empático y constructivo. Incluso se ha observado que, en ciertos casos, estas respuestas superan en calidad a las que ofrecería un experto, lo que pone en evidencia la importancia del diseño del *prompt* como una competencia docente importante para la formación del profesorado [34].

Un estudio reciente evaluó la utilidad real de las preguntas educativas generadas por modelos de lenguaje mediante técnicas de *prompt* engineering combinadas con taxonomías como la de Bloom y niveles de dificultad. Para ello, se empleó un modelo, al que se le proporcionaron instrucciones claras acompañadas de ejemplos siguiendo el few-shot *prompting*. Cada *prompt* incluía cinco pares de contexto y pregunta, con el objetivo de guiar al modelo en la generación de nuevas preguntas que se ajustaran a un tipo taxonómico específico, como recordar o evaluar, o a un nivel de dificultad (fácil, intermedio o avanzado). Los resultados mostraron que un gran porcentaje de las preguntas fueron únicas, y en promedio, consideradas útiles con pocas o ninguna edición necesaria para su uso en el aula [35].

3.3. Plataformas e-learning actuales

La incorporación de la inteligencia artificial en las plataformas *e-learning* está captando cada vez más la atención de la comunidad académica y tecnológica, con un número creciente de investigaciones en los últimos años que se centran en su

aplicación en la enseñanza y el aprendizaje [36].

Un ejemplo destacado es *Claude for Education*, una versión del modelo Claude desarrollada por Anthropic para instituciones educativas. Esta plataforma está diseñada para guiar el razonamiento de los estudiantes en lugar de proporcionar respuestas directas, fomentando así el pensamiento crítico. Además, los docentes pueden generar rúbricas personalizadas, ofrecer *feedback* y crear ecuaciones químicas o problemas matemáticos complejos [37].

Claude se encuentra actualmente integrado con *Canvas LMS*, un sistema de gestión de aprendizaje (LMS) de código abierto que facilita la creación de cursos, cuestionarios y permite a los estudiantes acceder al contenido de forma flexible [38].

Por otro lado, Moodle ha comenzado a incorporar IA tanto a través de un subsistema nativo como mediante plugins de terceros. El subsistema nativo se basa en una arquitectura modular que habilita la conexión con distintos proveedores de IA mediante API a través de proveedores ofreciendo la posibilidad de seleccionar el servicio de inteligencia artificial que mejor se ajuste a sus requerimientos técnicos, presupuestarios y de privacidad. Además, los *plugins* también se pueden incorporar mediante un fichero ZIP [39, 1].

Entre los plugins destaca Wooclap, que puede integrarse con Moodle y permite a los docentes crear preguntas interactivas, compartirlas con los estudiantes y sincronizar los resultados en el LMS [40].

Otro caso interesante es *EduPlanner*, un sistema automatizado basado en LLMs que asiste a los docentes en la creación y mejora de planes de clase personalizados. Su arquitectura se compone de múltiples agentes inteligentes que colaboran para evaluar, optimizar y analizar los diseños instruccionales. Para representar el conocimiento del alumnado, utiliza una estructura denominada *Skill-Tree*, un modelo jerárquico que organiza habilidades según su nivel de dependencia y dificultad. De este modo, identifican las competencias dominadas por los estudiantes y aquellas que requieren refuerzo, facilitando así una personalización pedagógica más eficaz [41].

4. Planificación

Para el desarrollo de este proyecto se ha optado por una metodología ágil, ya que ofrece mayor flexibilidad para adaptarse a los cambios que puedan surgir durante el proceso de desarrollo. Esta metodología contrasta con el modelo en cascada, donde no es posible comenzar la implementación hasta haber completado totalmente la fase de diseño.

También se ha aplicado el concepto de Producto Mínimo Viable (MVP), lo que ha permitido contar con una versión funcional desde las primeras etapas del proyecto. Esta elección ha facilitado poder probar el sistema, identificar mejoras y realizar ajustes necesarios de forma iterativa.

A continuación se presenta el diagrama de Gantt del proyecto, donde se muestra la planificación de las diferentes fases durante 16 semanas. Como se puede observar en la Figura 5, inicialmente se ha enfocado el trabajo en establecer la infraestructura técnica y la configuración básica del sistema. Posteriormente, se aborda la funcionalidad principal del proyecto que es la generación automática de preguntas mediante IA y proporcionar el *feedback* correspondiente.

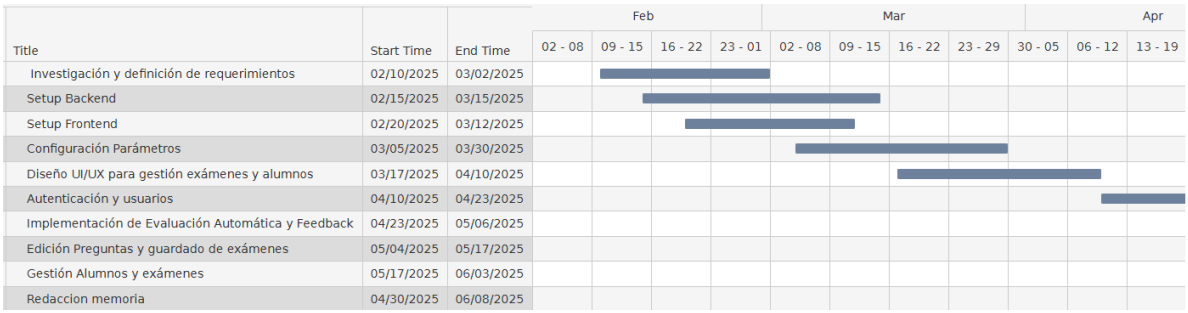

Figura 5: Diagrama de Gantt (Parte 1)

En las últimas semanas, se integró todo el conjunto de funcionalidades necesarias para que los profesores puedan gestionar estudiantes y asignar exámenes. Esto se encuentra en la Figura 6.

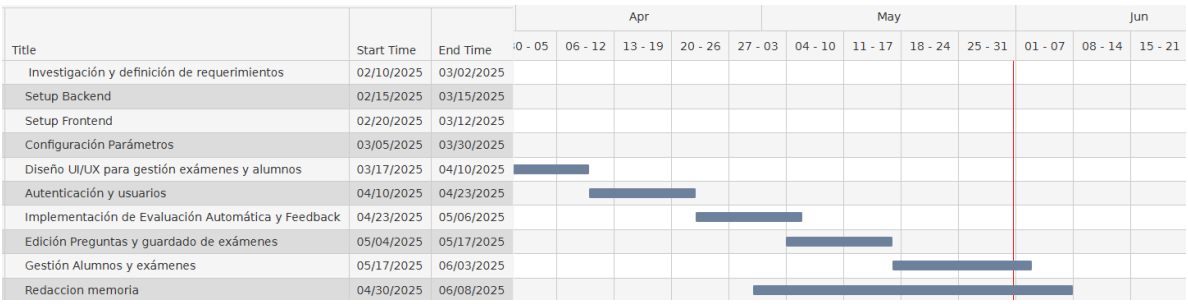

Figura 6: Diagrama de Gantt (Parte 2)

5. Análisis

El desarrollo del proyecto se gestionó mediante un sistema de backlog, tal y como aparece en la Figura 7, que organizaba las historias de usuario según su estado de desarrollo, es decir, las tareas pendientes, en desarrollo, en fase de validación y finalizadas.

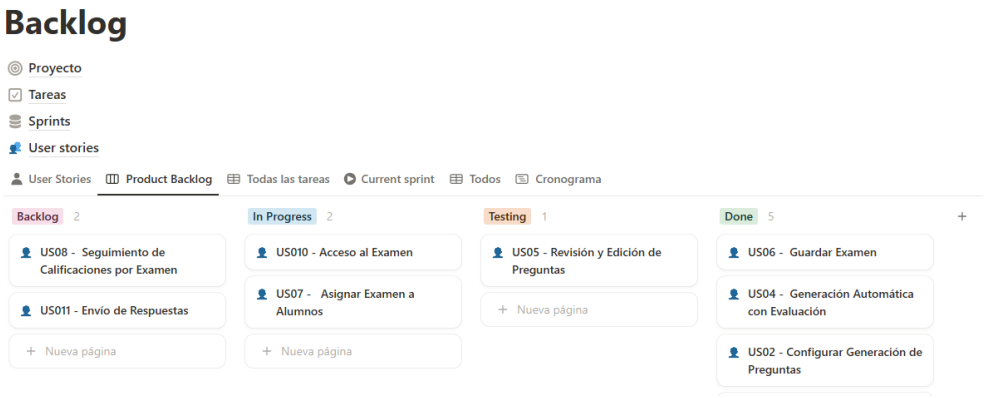

Figura 7: Organización del backlog de desarrollo

A partir de las historias de usuario recogidas en el Anexo historias de usuario 11, se definieron los casos de uso correspondientes para los perfiles de profesor y estudiante, que representan los dos roles existentes en la aplicación.

Casos de uso para profesores. La Figura 8 muestra el diagrama de casos de uso asociados al actor profesor.

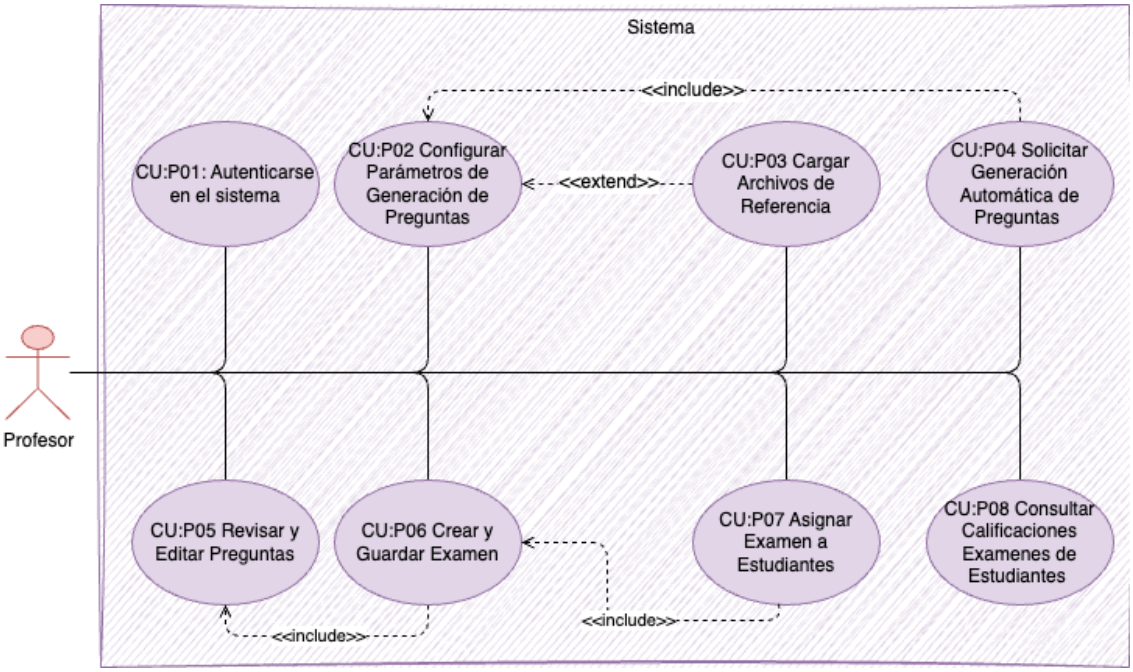

Figura 8: Diagrama de casos de uso para el actor de profesor

El Cuadro 1 describe cada caso de uso, su propósito y la historia de usuario correspondiente.

ID	**Nombre**	**Descripción**	**US**
CU-P01	Autenticarse en el sistema	El profesor inicia sesión o se registra con su rol	US01
CU-P02	Configurar parámetros	Define criterios para la generación automática de preguntas	US02
CU-P03	Cargar archivos	Sube documentos PDF como base para generar preguntas	US03
CU-P04	Generar preguntas	Crea preguntas a partir de los parámetros definidos	US04
CU-P05	Revisar/editar preguntas	Permite modificar o regenerar preguntas generadas	US05
CU-P06	Crear examen	Permite crear y guardar un examen	US06
CU-P07	Asignar examen	Distribuye exámenes a estudiantes o grupos	US07
CU-P08	Ver calificaciones	Ver los resultados de los alumnos en un examen	US08

Cuadro 1: Casos de uso para el perfil de profesor

Casos de uso para estudiantes. De manera similar, la Figura 9 muestra el diagrama de casos de uso para el actor estudiante.

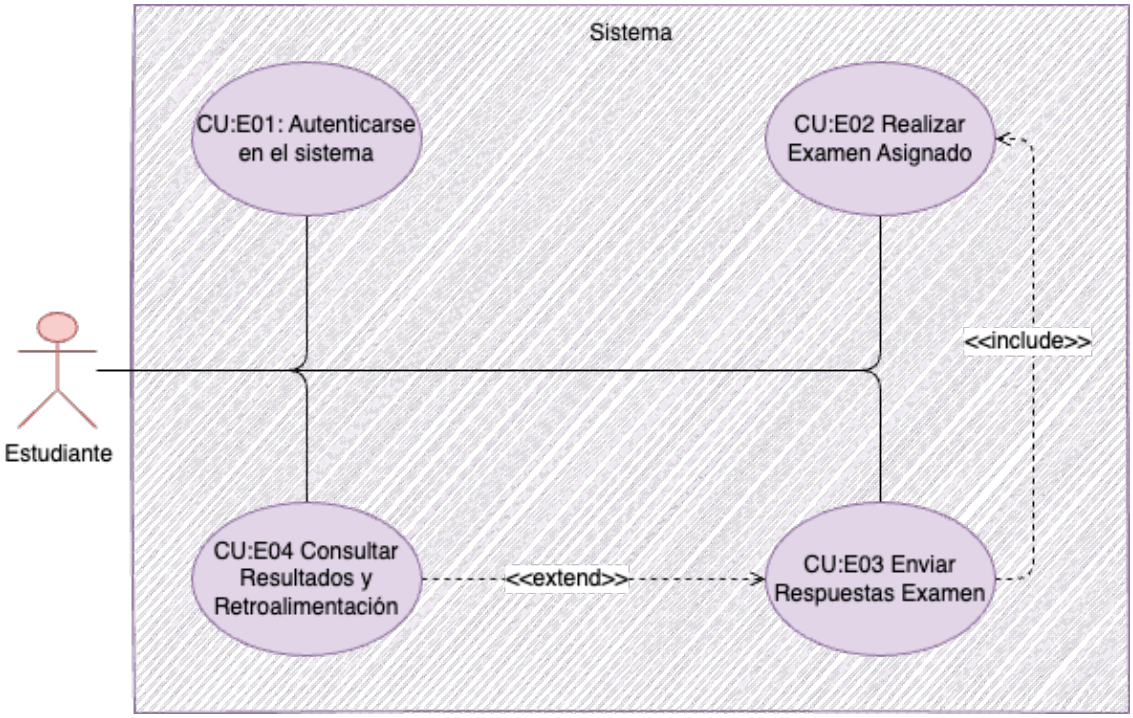

Figura 9: Diagrama de casos de uso para el actor de estudiante

El Cuadro 2 describe cada caso de uso, su propósito y la historia de usuario correspondiente.

ID	**Nombre**	**Descripción**	**US**
CU-E01	Autenticarse	El estudiante inicia sesión o se registra con su rol	US09
CU-E02	Realizar examen	Completa un examen asignado por el profesor	US10
CU-E03	Enviar respuestas	Finaliza y envía el examen para evaluación	US11
CU-E04	Consultar resultados	Visualiza calificación y retroalimentación recibida	US12

Cuadro 2: Casos de uso para el perfil de estudiante

6. Diseño

En esta sección se presenta la arquitectura general del sistema y las decisiones adoptadas respecto a la elección del modelo, el modo de integración y los recursos disponibles.

6.1. Arquitectura

La plataforma adopta una arquitectura de tipo cliente-servidor, compuesta por un *frontend* accesible desde el navegador y/o móvil, y un *backend* encargado de realizar las principales funcionalidades del sistema, tal y como se muestra en Figura 10. Ambas partes se han desarrollado de forma desacoplada, manteniéndose en repositorios independientes, lo que permite una evolución modular del proyecto y facilita la implementación de cambios futuros en cada componente por separado.

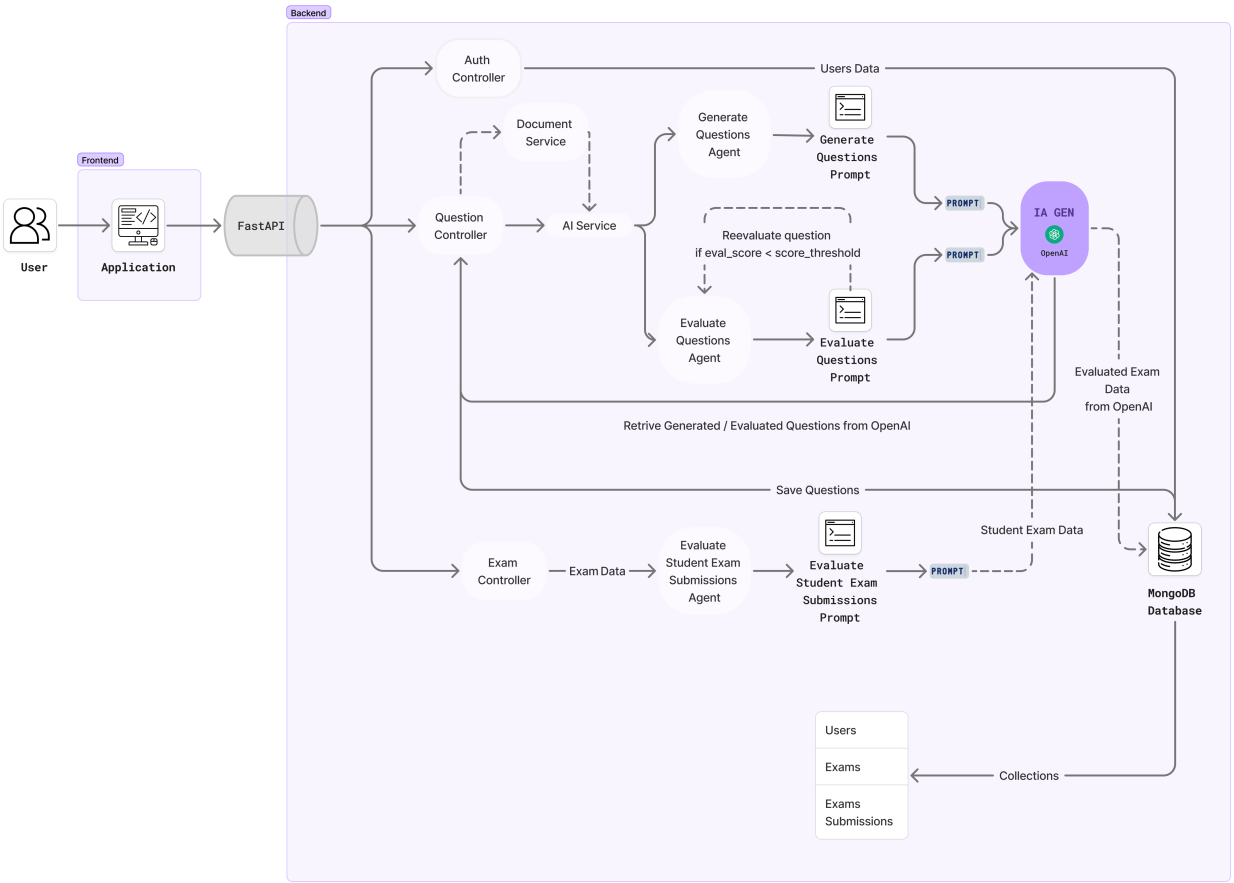

Figura 10: Arquitectura de la plataforma

El *frontend* permite la interacción con el usuario, utilizando *Vue.js* y *TailwindCSS* para crear interfaces adaptadas a distintas resoluciones, con vistas diferenciadas para docentes y estudiantes.

El *backend*, desarrollado en Python con *FastAPI* y *LangChain*, se encarga de recibir las solicitudes, validar el acceso, administrar los *prompts* para el modelo de lenguaje y organizar la información generada. El uso de *LangChain* permite gestionar los *prompts* y las respuestas, facilitando el cambio de modelo de lenguaje o de proveedor en el futuro.

La integración con GPT-4.1 mini se realiza mediante la API de OpenAI, gestionada por *LangChain*. Toda la información sobre usuarios, exámenes, respuestas y evaluaciones se almacena en una base de datos *NoSQL*, lo que facilitando la gestión de datos semiestructurados.

6.1.1. Flujo general de la aplicación

El flujo de funcionamiento de la plataforma se puede dividir en tres etapas principales:

1. **Generación de preguntas:** Cuando el usuario con rol profesor solicita la creación de preguntas desde el formulario, debe definir una serie de parámetros como el nivel educativo, tipo de preguntas, cantidad, dificultad y si desea basarse en un archivo PDF. A partir de estos datos, se activa el agente `Generate Questions Agent`, el cual utiliza uno de los dos *prompts* disponibles: la *prompt* para generar preguntas a partir de un documento PDF, o la *prompt* para generar preguntas a partir de un tema general. La instrucción generada es enviada al modelo LLM (OpenAI), que responde con una lista de preguntas que serán procesadas en la siguiente fase.

2. **Evaluación de preguntas:** Una vez recibidas las preguntas generadas por la IA, el sistema activa el agente `Evaluate Questions Agent`, que utiliza una *prompt* específica para evaluar la calidad pedagógica de cada pregunta. Si alguna de ellas obtiene una puntuación inferior a un umbral mínimo definido (`score_threshold`), se genera una nueva versión de la misma hasta un máximo de tres intentos. Si tras estos intentos la calidad sigue siendo insuficiente, la pregunta se excluye del conjunto final y no se muestra al docente.

3. **Creación y realización del examen:** Una vez validadas las preguntas, el docente puede crear un examen seleccionando las que considere adecuadas y asignándolas a un grupo de estudiantes. Las preguntas seleccionadas se almacenan en la colección `Exams` de la base de datos MongoDB. Cuando un estudiante completa el examen, sus respuestas son procesadas por el agente `Evaluate Student Exam Submissions Agent`, que utiliza una *prompt* específica para evaluar las respuestas de desarrollo y generar *feedback* inmediato. Además, se ejecuta una segunda *prompt* para generar un comentario global sobre el desempeño del estudiante en el examen. Los resultados se almacenan en la colección `Exam Submissions`.

Cada una de estas etapas está implementada por una *prompt* específica, que se describe en detalle en la sección de implementación.

6.2. Modelos locales vs acceso por API

La ejecución de modelos de lenguaje (LLMs) puede realizarse de forma local o mediante servicios de terceros a través de API. Ejecutar modelos localmente requiere contar con una GPU de alto rendimiento y suficiente capacidad de almacenamiento de disco, CPU y RAM.

Se llevó a cabo una prueba inicial de ejecución local utilizando Ollama, una herramienta que ofrece descargar y ejecutar modelos de lenguaje en entornos locales. Para esta prueba se utilizó el modelo LLaMA 3.1 8B. El 8B significa que el modelo tiene 8 mil millones de parámetros. Estos parámetros son valores numéricos que el modelo aprende durante su entrenamiento. Cuantos más parámetros tiene un modelo, más complejo y preciso puede ser, pero también necesita más recursos para funcionar bien. Aunque el modelo 8B no es de los más grandes, se recomienda usar una GPU con al menos 8–12 GB de memoria VRAM [42].

El portátil usado cumplía con los requisitos mínimos, pero el rendimiento fue muy limitado, el modelo respondía con lentitud y en ocasiones se bloqueaba. Por eso, se optó por utilizar modelos a través de API.

6.3. Comparativa de modelos LLM

La selección de modelos Cuadro 3 responde a una revisión de las principales opciones disponibles en abril de 2025, considerando su accesibilidad mediante API, su rendimiento y su capacidad de razonamiento.

Todos ellos son modelos de lenguaje de última generación, desarrollados por compañías como OpenAI, Google DeepMind y Anthropic. Además, se ha priorizado la inclusión de modelos que tienen una integración técnica documentada.

Modelo	Razonamiento	Velocidad	Coste aprox. (USD)	Entrada / Salida
GPT-4.1 [43]	Alto	Media	\$2.00 / \$8.00	Texto, imagen / Texto
GPT-4.1 mini [43]	Medio	Alta	\$0.40 / \$1.60	Texto, imagen / Texto
GPT-4o [43]	Muy alto	Muy alta	\$2.50 / \$10.00	Texto, imagen
Gemini 1.5 Pro [44]	Muy alto	Alta	\$2.50 / \$10.00	Texto, imagen, audio, video / Texto
Claude 3.5 Sonnet [45]	Alto	Alta	\$3.00 / \$15.00	Texto, imagen / Texto

Cuadro 3: Comparativa de modelos de lenguaje con acceso a API (abril 2025)

6.4. Coste y modelo seleccionado

Durante la fase inicial de pruebas e integración de la plataforma, se optó por utilizar el modelo Gemini 1.5 Pro de Google ofrece una cuota gratuita para desarrollo y pruebas, lo que permitió realizar evaluaciones preliminares sin incurrir en costes económicos hasta agotar ese límite.

Posteriormente, se migró al modelo GPT-4.1 Mini de OpenAI por su equilibrio entre coste y rendimiento, facilidad de integración mediante API y control detallado del consumo, ya que permite recargar los créditos. Además, gran parte de los artículos revisados en el estado del arte utilizan ChatGPT como base experimental, lo que refuerza su elección como modelo de referencia. Toda esta migración de un modelo a otro no tuvo ningún impacto negativo en el desarrollo, ya que el uso de *Langchain* permite desacoplar el proveedor de IA de la implementación, lo que lo hace agnóstico de un modelo u otro.

Al trabajar con modelos mediante API, hay que tener en cuenta las limitaciones de uso impuestas por el proveedor, conocidas como rate limits. Estas definen cuántas peticiones por minuto (RPM, por sus siglas en inglés) o cuántos *tokens* por minuto (TPM, por sus siglas en inglés) pueden procesarse, dependiendo del modelo utilizado y del nivel de uso alcanzado por la cuenta [46].

Además, si se utiliza la API de procesamiento por lotes (Batch API), también se debe considerar el límite de cola de procesamiento por lotes (batch queue limit), que se calcula en función del número total de *tokens* de entrada en espera para un modelo determinado [46]. En el caso de este proyecto, se ha alcanzado el *Tier 1* de OpenAI, al haber superado el umbral de gasto mínimo de 5 dólares estadounidenses [46]. Para el modelo GPT-4.1 mini, esto implica un coste de \$0.40 por millón de *tokens* de entrada y \$1.60 por millón de *tokens* de salida, así como un límite de hasta 500

peticiones por minuto (RPM) y 30 000 *tokens* por minuto (TPM) [47]. Los niveles de uso disponibles y sus condiciones de acceso se presentan en el Cuadro 4.

Tier	**Condición de acceso**	**Límite de uso mensual (USD)**
Gratis	Usuario en una región permitida	100
Tier 1	Pago mínimo de 5 USD	100
Tier 2	Pago de 50 USD y 7+ días desde el primer pago	500
Tier 3	Pago de 100 USD y 7+ días desde el primer pago	1 000
Tier 4	Pago de 250 USD y 14+ días desde el primer pago	5 000
Tier 5	Pago de 1 000 USD y 30+ días desde el primer pago	200 000

Cuadro 4: Tiers de uso de la API de OpenAI según consumo y antigüedad de la cuenta [46]

A su vez, los límites técnicos para el modelo GPT-4.1 mini se encuentran en el Cuadro 5, donde se detalla la capacidad de peticiones por minuto (RPM), *tokens* por minuto (TPM) y la cola de procesamiento por lote.

Tier	**RPM**	**TPM**	**Batch queue limit**
Gratis	No disponible	No disponible	—
Tier 1	500	30 000	90 000
Tier 2	5 000	450 000	1 350 000
Tier 3	5 000	800 000	50 000 000
Tier 4	10 000	2 000 000	200 000 000
Tier 5	10 000	30 000 000	5 000 000 000

Cuadro 5: Límites técnicos de uso para el modelo GPT-4.1 mini por nivel de suscripción [47]

7. Implementación

La arquitectura se organiza en dos componentes principales. Por un lado, el *backend*, encargado de la lógica de negocio, la integración con servicios de IA y la persistencia de datos. Por otro, el *frontend*, que ofrece una interfaz amigable para docentes y estudiantes.

Primero se describe la estrategia de generación de *prompts* y su diseño, que orienta a los LLM en la creación automática de preguntas, la evaluación de las preguntas, la generación de *feedback* de las preguntas de desarrollo y el análisis del desempeño en los exámenes. A continuación, se detalla la arquitectura del *backend*, incluyendo aspectos como la autenticación, la integración con LLMs, el procesamiento de archivos PDF y el uso de bases de datos NoSQL. Posteriormente, se aborda el desarrollo del *frontend*, con las tecnologías empleadas y las vistas de la aplicación.

7.1. Prompts generadas

Los *prompts* permiten dar instrucciones al LLM. Su diseño se ha basado en las buenas prácticas de ingeniería de *prompts*, algunas de las cuales se mencionan en el estado del arte, y otras han sido extraídas y aplicadas de la fuente [17].

Los *prompts* se encuentran almacenadas en archivos `.st` proporcionan la orientación pedagógica necesaria para que el LLM genere contenido dentro de una estructura predefinida. Estas *prompts* especifican el tono educativo, los criterios de evaluación, el enfoque temático y las características específicas que debe tener el contenido generado.

Paralelamente, *LangChain* define la estructura formal de las salidas (*output*) mediante métodos como *with_structured_output()*, que guían al modelo a generar respuestas con un formato, por ejemplo en JSON. Mediante la clase `PromptTemplate` de *LangChain* se construyen instrucciones dinámicas que combinan texto fijo con variables que toman valores específicos durante la ejecución. Esto permite crear comandos personalizados para el modelo que determinan no solo qué debe responder, sino también cómo debe estructurar su respuesta.

Todas las *prompts* se encuentran en Anexo 11 con su respectivo *output* esperado.

7.1.1. Prompt para generar preguntas a partir de un PDF

El *prompt* utilizado para la generación automatizada de preguntas a partir de documentos PDF, se encuentra en el Anexo Prompt para generar las preguntas a partir de un PDF y el *output* correspondiente muestra el formato de salida de la respuesta generada por la IA. En este caso se espera que nos devuelva una lista de preguntas con la pregunta, el tipo, las opciones, la respuesta correcta y la explicación. Las preguntas posibles a generar son preguntas de opción múltiple, verdadero falso y desarrollo de respuesta corta.

Este *prompt* incorpora las siguientes técnicas de *prompt engineering*:

- **Role-Based Prompting**: Se establece el rol que debe asumir el modelo, el cual debe actuar como "un especialista en didáctica educativa" para establecer el estilo de comunicación y la perspectiva educativa que debe adoptar el modelo en sus respuestas.
- **Template-Based Prompting**: El *prompt* se construye utilizando una plantilla con variables dinámicas (como {`file_content`}, {`topic_questions`}, {`nivel_educativo`}, etc.), que se rellenan automáticamente en tiempo de ejecución mediante *LangChain*.
- **Persona-Based Prompting**: Se define un perfil profesional que incluye conocimientos especializados y experiencia en el diseño de evaluaciones educativas. Para ello, se indica como "especialista en didáctica educativa", "con amplia experiencia" y "en la creación de evaluaciones de alta calidad".
- **Scenario-Based Prompting**: El diseño del *prompt* parte de un contexto definido, un profesor entrega un PDF con el contenido y el contenido de él es el único material que pueden utilizar. Esto ayuda al modelo a centrarse en ese contenido y evitar información externa o irrelevante.
- **Conditional Prompting**: Se implementa lógica condicional como indicar que "Cuando la pregunta sea una definición, usa el término a definir en la pregunta", y "Si es necesario usarlas, asegúrate de que estén redactadas con la mayor claridad posible". De esta forma el modelo adapta su comportamiento.
- **Task Decomposition Prompting**: La tarea se divide en subtareas definidas separando las diferentes responsabilidades que son análisis del contenido, clasificación por tipos de pregunta (selección múltiple, verdadero/falso, respuesta corta), distribución por niveles de dificultad (fácil, medio, difícil), aplicación

de criterios por tipo de pregunta, y generación de respuestas correctas. Esto se aplica para guiar al modelo en la respuesta.

- **Constraint-Based Prompting**: El *prompt* establece restricciones que acotan el comportamiento del modelo durante la generación de preguntas. Estas incluyen mantener la coherencia temática (todas las preguntas deben basarse únicamente en el contenido del PDF), respetar una estructura homogénea en las opciones de respuesta, evitar pistas evidentes en las opciones incorrectas y distribuir aleatoriamente los enunciados verdaderos y falsos en las preguntas de tipo verdadero/falso.

- **Contextual Prompting**: El *prompt* proporciona un marco contextual amplio que incluye el escenario educativo (profesor que proporciona PDF como único material), las características del contenido, el propósito evaluativo, y las especificaciones del nivel educativo. Aplicando esta técnica, el modelo puede generar preguntas más contextualizadas y adecuadas al ámbito educativo.

7.1.2. Prompt para generar preguntas con temática libre

A diferencia del enfoque orientado a la generación de preguntas a partir de un PDF proporcionado por el docente, donde las preguntas deben de generarse exclusivamente del archivo, la generación de preguntas a partir de un tema no se encuentran condicionadas por un material de referencia concreto, por lo que la formulación del *prompt* delimitar el alcance indicando "Analiza el siguiente tema...", "No te salgas del área temática indicadaz "Adecúa el lenguaje y el enfoque al nivel educativo solicitado". El *prompt* se encuentra en el Anexo Prompt para generar las preguntas con temática libre.

7.1.3. Prompt para evaluar las preguntas

Una vez se han generado las preguntas, el *prompt* se utiliza para evaluar la calidad de las preguntas generadas automáticamente a partir del *prompt* anterior antes de mostrarlas al docente. Esto se hace en "Listado de preguntas generadas por IA: {preguntas_generadas}". El *prompt* se encuentra en el Anexo Prompt para evaluar las preguntas con su respectivo *output* esperado que es igual al anterior pero con un {`eval_score`} para almacenar la puntuación.

Este *prompt* incorpora las siguientes técnicas de *prompt engineering*:

- **Técnicas comunes**: Se mantienen las técnicas de Role-Based Prompting, Scenario-Based Prompting, Template-Based Prompting y Contextual *Prompting* descritas en la sección anterior.
- **Task Decomposition Prompting**: La tarea de evaluación se divide en cinco dimensiones ponderadas que conforman el total de 10 puntos, claridad y redacción (0–2), relevancia temática (0–2), nivel cognitivo requerido (0–2), calidad pedagógica del tipo de pregunta (0–2), y valor didáctico (0–2).
- **Constraint-Based Prompting**: Se imponen restricciones, una pregunta sin relación con el contenido debe puntuarse con 0, y se penaliza la redundancia o baja calidad.
- **Prompt Chaining**: Es una técnica que el resultado de un *prompt* se pasa a la siguiente *prompt*. Por lo tanto, el proceso se organiza en dos fases, en la primera se generan las preguntas y luego esta *prompt* las utiliza.

7.1.4. Prompt para evaluar pregunta desarrollo

Una vez que el estudiante ha respondido a la pregunta de desarrollo de respuesta corta, estas preguntas se envían al LLM para analizar y proporcionar un *feedback* al estudiante sobre su respuesta. El *prompt* tiene la pregunta, la explicación de la pregunta de que nos ha devuelto el primer *prompt* al generar las preguntas y la respuesta del estudiante. El *prompt* se encuentra en Prompt para feedback preguntas desarrollo y el *output* esperado es si es correcta o no y el porqué.

Por otro lado, el resto de preguntas (opción múltiple y verdadero/falso) ya tienen la explicación y respuesta del *prompt* de generar preguntas que es la opción que se mostrará directamente al estudiante.

Este *prompt* incorpora las siguientes técnicas de *prompt engineering*:

- **Técnicas comunes**: Se mantienen Role-Based Prompting, Scenario-Based Prompting, Template-Based Prompting, Contextual Prompting, implementadas de forma análoga a los casos anteriores.
- **Task Decomposition Prompting**: En este caso, se indican los distintos aspectos que debe contener la evaluación: una valoración, la respuesta, el análisis de su alineación con los elementos esperados, el reconocimiento de aciertos y si es incorrecta, indique sugerencias de mejora.

- **Constraint-Based Prompting**: Se imponen restricciones como "No más de 100 palabras".
- **Direct Instruction Prompting**: Se proporcionan instrucciones sobre qué tiene que hacer: "Tu tarea es analizar una respuesta dada por un estudiantez "Debes decidir si la respuesta es correcta o incorrecta".
- **Question-Based Prompting**: Se presenta la pregunta original, el marco de referencia para la evaluación y finalmente la respuesta escrita, facilitando así la contextualización para la evaluación.

7.1.5. Prompt para obtener feedback del examen

Una vez que el estudiante ha finalizado el examen completo, se activa una *prompt* diseñada para dar *feedback* global de todo su examen. Proporciona un análisis más amplio de sus fortalezas, debilidades y recomendaciones de mejora. El *prompt* utilizado se encuentra en el Anexo Prompt para feedback del examen y el *output* esperado en un texto de máximo 200 palabras con comentarios estructurados y constructivos.

Este *prompt* incorpora las siguientes técnicas de *prompt engineering*:

- **Técnicas comunes**: Se mantienen Role-Based Prompting, Scenario-Based Prompting, Template-Based Prompting y Contextual Prompting, implementadas de forma análoga al caso anterior.
- **Task Decomposition Prompting**: La tarea se descompone en subtareas, análisis de respuestas del estudiante, identificación de fortalezas y debilidades, sugerencias específicas de mejora, y estructuración de comentarios organizados y constructivos.
- **Constraint-Based Prompting**: Se imponen restricciones estrictas sobre el formato (solo texto, máximo 200 palabras), el tono (constructivo y profesional), la estructura (párrafos claros), el enfoque (contenido y comprensión), y la objetividad (evitar juicios de valor).
- **Direct Instruction Prompting**: Proporciona instrucciones específicas sobre la tarea ("analizar el examen realizado", "proporcionar comentarios detallados", "identificar puntos fuertes y áreas de mejora") y sobre el formato de salida esperado.

- **Prompt Chaining**: Esta técnica recibe como entrada tanto las respuestas del estudiante como el examen generado previamente.

7.2. Backend

El *backend* expone una *API REST* desarrollada con *FastAPI*, encargada de gestionar todas las operaciones del sistema. Gracias al uso de *Pydantic*, se consigue un tipado automático y robusto de los modelos de entrada y salida, garantizando una validación eficaz de los datos a lo largo de toda la aplicación.

En cuanto a la seguridad, se ha implementado un sistema basado en *JWT* (*JSON Web Tokens*), utilizando tanto *tokens* de acceso como *tokens* de refresco de sesión, junto con un sistema de control de acceso basado en roles. Todos los *endpoints* de la *API REST* validan y autorizan el acceso a los recursos en función de la autenticación y los permisos del usuario.

Además, desde el *frontend* nunca se envían identificadores de usuario en el cuerpo de las peticiones, sino que estos se extraen del *token JWT* mediante el sistema de dependencias de *FastAPI* (`current_user`). Este enfoque elimina la posibilidad de suplantación de identidad, ya que el servidor confía únicamente en la información verificada criptográficamente del *token*.

Por último, para mejorar la seguridad en el almacenamiento y transmisión de los *tokens* de refresco, estos se envían al cliente mediante *cookies HTTP-only*, lo que impide su acceso desde el entorno *JavaScript* del navegador. Esta medida mitiga vulnerabilidades frecuentes como los ataques de *Cross-Site Scripting (XSS)* y *Cross-Site Request Forgery (CSRF)*.

La aplicación organiza su lógica de negocio a través de los siguientes *routers* (también conocidos como *controllers*):

- **Auth Router:** Gestiona la autenticación de usuarios implementando registro, inicio de sesión, refresco de *tokens JWT* y cierre de sesión.
- **Exam Router:** Controla todas las operaciones relacionadas con exámenes, envío de respuestas y visualización de resultados según role de usuario.
- **Question Router:** Maneja la generación de preguntas mediante IA, permitiendo a profesores crear preguntas basadas en temas específicos o contenido de archivos PDF.

- **User Router:** Gestiona la información del usuario actual que ha iniciado sesión.

- **Question_service**: Gestiona la generación y almacenamiento de preguntas, incluyendo la validación y extracción de contenido de archivos PDF y la integración con servicios de IA para generar preguntas de calidad.

- **Exam_service**: Administra el ciclo de vida completo de los exámenes, desde la creación y modificación hasta la recuperación con diferentes niveles de detalle según el rol del usuario.

- **Auth_service**: Proporciona funcionalidades de autenticación y gestión de sesiones mediante *tokens* JWT, incluyendo registro, inicio de sesión y renovación de *tokens*.

- **Ai_service**: Implementa la integración con servicios de IA para generar y evaluar preguntas, validar respuestas cortas y proporcionar retroalimentación personalizada sobre exámenes.

- **User_service**: Maneja operaciones como el registro, inició de sesión relacionadas con el usuario.

- **Exam_submission_service**: Gestiona el proceso de envío y evaluación de respuestas de exámenes, incluyendo la validación automatizada y generación de retroalimentación mediante IA.

7.2.1. Extracción de contenido

El proceso de extracción de contenido se inicia cuando la plataforma recibe un archivo PDF a través del *endpoint* diseñado para la generación automática de preguntas a través de una solicitud de tipo *multipart/form-data*, lo que permite la transmisión de archivos binarios junto con el resto de parámetros para la generación de cuestionarios. Como primer filtro, se implementa una validación para comprobar que el archivo proporcionado es de la extensión correspondiente (PDF) para proporcionar respuesta al usuario en caso de que el archivo no es del tipo esperado.

Una vez se valida, el sistema procede a la lectura completa del contenido binario del archivo, almacenándolo temporalmente en memoria como un flujo de bytes. Posteriormente, mediante la creación de un objeto *BytesIO*, se transforma dicho flujo de bytes en un objeto de entrada compatible con la biblioteca *PDFMiner*. La función *extract_text* de *PDFMiner* [48] extrae el contenido textual.

Tras la extracción, el texto obtenido suele presentar imperfecciones derivadas del propio proceso, como líneas vacías, espacios en blanco o saltos de línea irregulares. Por este motivo, se implementa una fase adicional de limpieza, para la eliminación de líneas vacías y la homogeneización de los espacios.

Tras este proceso, el texto limpio se usa únicamente como contexto para la generación de preguntas. El archivo PDF no se almacena en la plataforma tras el proceso de extracción. Únicamente se conserva el texto necesario durante el tiempo necesario para la generación de las preguntas.

7.2.2. Integración con LLM

La integración de LLMs en la plataforma se ha planteado un patrón de diseño *Factory*, que centraliza la lógica de creación de los diferentes clientes LLM disponibles. Inicialmente, se diseñó para Gemini y después se implementó para *OpenAI*. La clase *LLMFactory* actúa como único punto de acceso para la creación de instancias, recibiendo parámetros como el proveedor seleccionado, el modelo concreto y configuraciones como el parámetro *temperature*. El parámetro del valor se ha establecido igual a 0.7 se considera intermedio para favorecer la creatividad y diversidad de las respuestas generadas.

7.2.3. Base de datos NoSQL con MongoDB

Para el almacenamiento de datos se ha utilizado *MongoDB*, una base de datos NoSQL orientada a documentos que ofrece un modelo de datos flexible, ideal para representar distintos tipos de preguntas (opción múltiple, verdadero/falso, respuesta corta) sin necesidad de redefinir esquemas rígidos. Gracias a su soporte nativo para datos anidados, estructuras complejas como arrays de opciones se representan de forma natural mediante documentos *JSON*, por ejemplo:

```
{
  "question": "¿Pregunta?",
  "type": "multiple_choice",
  "options": [
    {
      "text": "Opción 1",
      "is_correct": true
    },
    ...
```

```
  ],
  "difficulty": "medium"
}
```

Por último, su integración con *Python* mediante la biblioteca asíncrona *motor* permite realizar operaciones de base de datos sin bloquear el sistema, encajando perfectamente con el enfoque asincrónico de *FastAPI*. Las principales colecciones utilizadas en la base de datos son:

- **Users**: almacena la información de los usuarios registrados en la plataforma.
- **Exams**: guarda los exámenes creados por los profesores.
- **Exam_submissions**: contiene las respuestas enviadas por los estudiantes para cada examen.

7.2.4. Validación y Manejo de Errores

El sistema implementa diferentes mecanismos de validación para garantizar la integridad de datos a través de validadores *Pydantic* con el uso de `model_validator`, manejo de errores en código mediante el uso de bloques `try-except`, gestión de excepciones personalizadas y transformación en respuestas *HTTP* estandarizadas para proporcionar respuestas consistentes al frontal de la aplicación.

7.3. Frontend

El frontal de la aplicación se ha desarrollado empleando un *stack* basado en Vue3, TypeScript y TailwindCSS para ofrecer una experiencia fluida tanto a estudiantes como a profesores.

7.3.1. Gestión de Estado con Pinia

Para la gestión global del estado de valores se ha implementado Pinia. Las stores de Pinia se organizan por dominio funcional (exámenes, preguntas, usuario, auth), implementando acciones asíncronas para operaciones de API y proporcionando estado reactivo a los componentes.

7.3.2. TailwindCSS y Shadncn-vue

La interfaz de usuario se ha construido utilizando TailwindCSS como framework CSS, aplicando clases directamente en los elementos HTML para construir componentes consistentes, pero además para el uso de componentes más complejos se ha usado la librería UI de Shadcn-vue

7.3.3. Servicios Axios

La comunicación con el backend se ha encapsulado en servicios específicos, que abstraen las llamadas HTTP mediante Axios. Cada servicio implementa métodos para operaciones CRUD sobre su dominio específico, gestionando errores y transformando datos. Además, esta abstracción añade automáticamente las cabeceras `Authorization: Beare JWT` a todas las peticiones segurizadas que lo requieran.

7.3.4. Validación y manejo de errores

Los datos proporcionados en los formularios por parte de los usuarios se validan mediante esquemas Zod, garantizando su integridad antes de ser enviados al backend para su procesamiento. Esta validación proporciona seguridad y documentación implícita de los contratos de API. Gracias a Zod, se muestran mensajes de error asociados a los campos del formulario (inputs) para dar *feedback* al usuario. Por otro lado, hay un componente llamado Toast de la librería de componentes Shadcn-vue que proporciona *feedback* visual al usuario mediante pop-ups, mostrando mensajes de éxito o error según el resultado de la acción realizada.

7.4. Vistas

En este apartado podemos visualizar explicadas las vistas principales y las de profesor y estudiante.

Inicio de sesión La pantalla de inicio de sesión cuenta con un formulario que incluye campos para el correo electrónico y la contraseña, así como un botón para iniciar sesión. En caso de que el usuario no disponga de una cuenta, se proporciona un enlace para registrarse. La Figura 11 muestra el formulario de acceso correspondiente al perfil de profesor, como parte del flujo de autenticación de la plataforma.

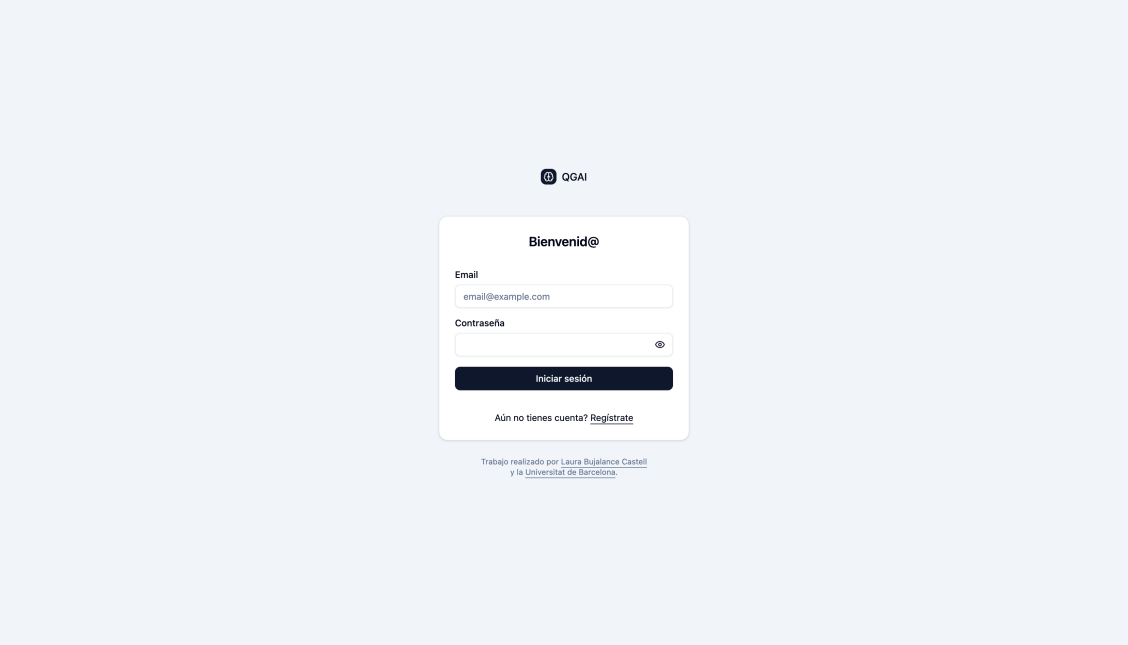

Figura 11: Inicio de sesión de la plataforma

Registro La vista de registro permite crear una cuenta seleccionando el rol de profesor o alumno. El formulario incluye nombre, apellidos, correo electrónico, contraseña y su confirmación. Al hacer clic en "Regístrate", se valida la información antes de enviarla. La Figura 12 muestra esta pantalla.

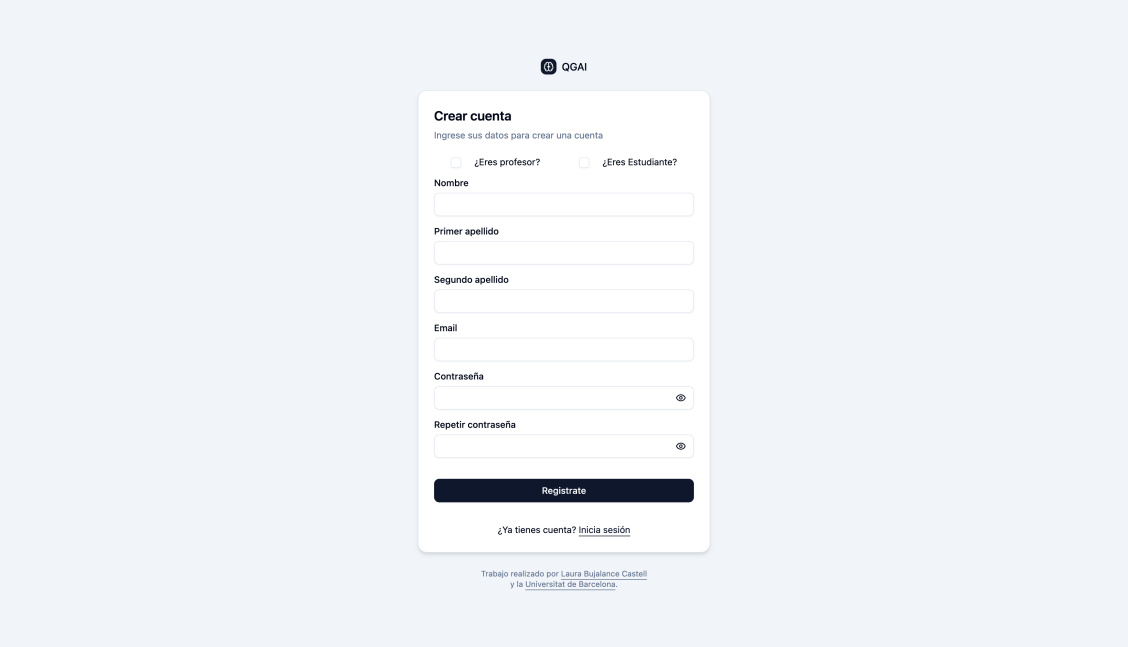

Figura 12: Formulario de registro de usuario

Pantalla de inicio del profesor Esta vista se muestra tras iniciar sesión con un perfil de profesor. Presenta un resumen general de las acciones disponibles, como generar preguntas, crear exámenes o consultar resultados.La Figura 15 muestra esta pantalla, donde además se puede observar que la interfaz es completamente *responsive*.

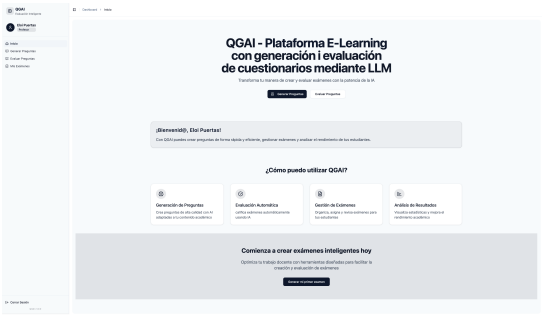

Figura 13: (a) Versión escritorio

Figura 14: (b) Versión móvil

Figura 15: Pantalla de inicio del perfil docente en distintos dispositivos

7.4.1. Vistas del profesor

Generar preguntas Esta vista está diseñada para configurar los parámetros necesarios para la generación de preguntas utilizando inteligencia artificial. Los campos del formulario están directamente relacionados con los inputs requeridos en el *prompt* de generación de preguntas, como el tema, el nivel educativo, el tipo de pregunta (opción múltiple, verdadero/falso o respuesta corta) y la dificultad. Además, es posible añadir detalles adicionales, como indicaciones específicas del profesor, lo que facilita la personalización de las preguntas. Además, se ofrece la opción de subir un archivo PDF para que las preguntas sean generadas a partir del contenido de dicho documento.

Una vez que todos los campos han sido completados, el usuario solo debe hacer clic en "Generar preguntas"para activar el proceso de creación. Además de los parámetros de entrada, la interfaz muestra información sobre el modelo de inteligencia artificial que se está utilizando para generar las preguntas. La Figura 16 muestra esta pantalla.

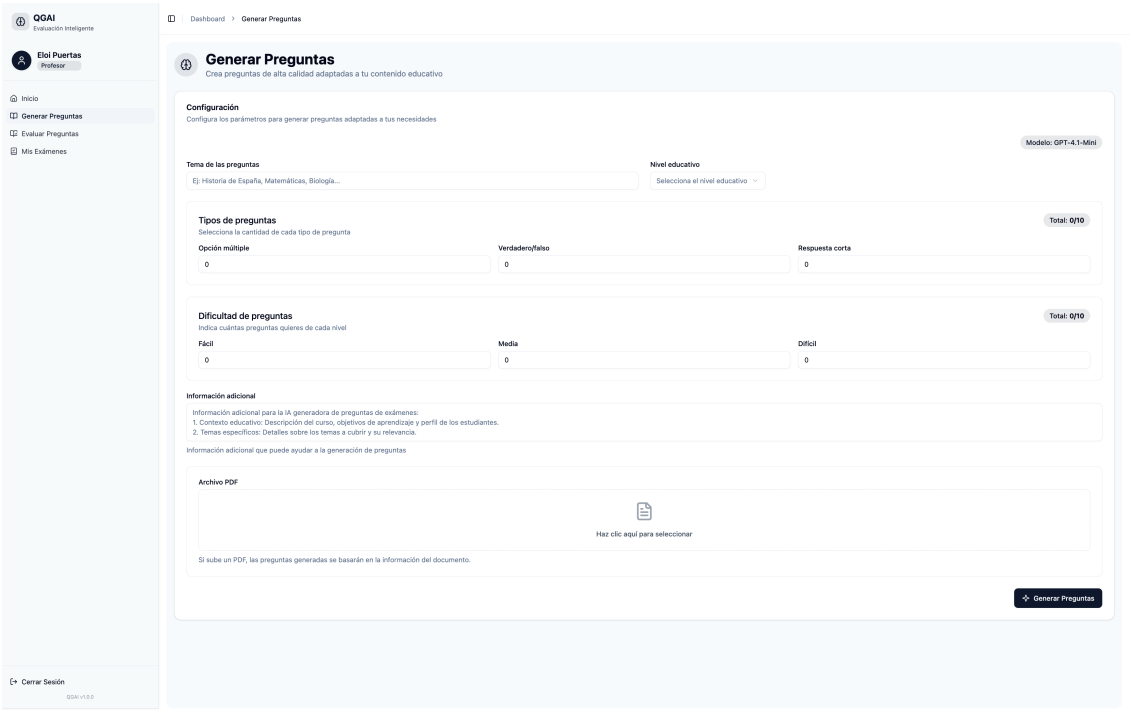

Figura 16: Vista de generación de preguntas con configuración personalizada

Evaluar las preguntas generadas Esta vista muestra las preguntas generadas automáticamente junto con la evaluación que les ha asignado la IA a partir del *prompt* de evaluación de preguntas. Cada pregunta incluye un selector con opciones para regenerarla, editarla (en especial en preguntas abiertas) o ajustarla antes de guardarla. Además, se incluye una pestaña de detalles del examen desde la cual se puede proceder al guardado final. La Figura 17 muestra esta pantalla.

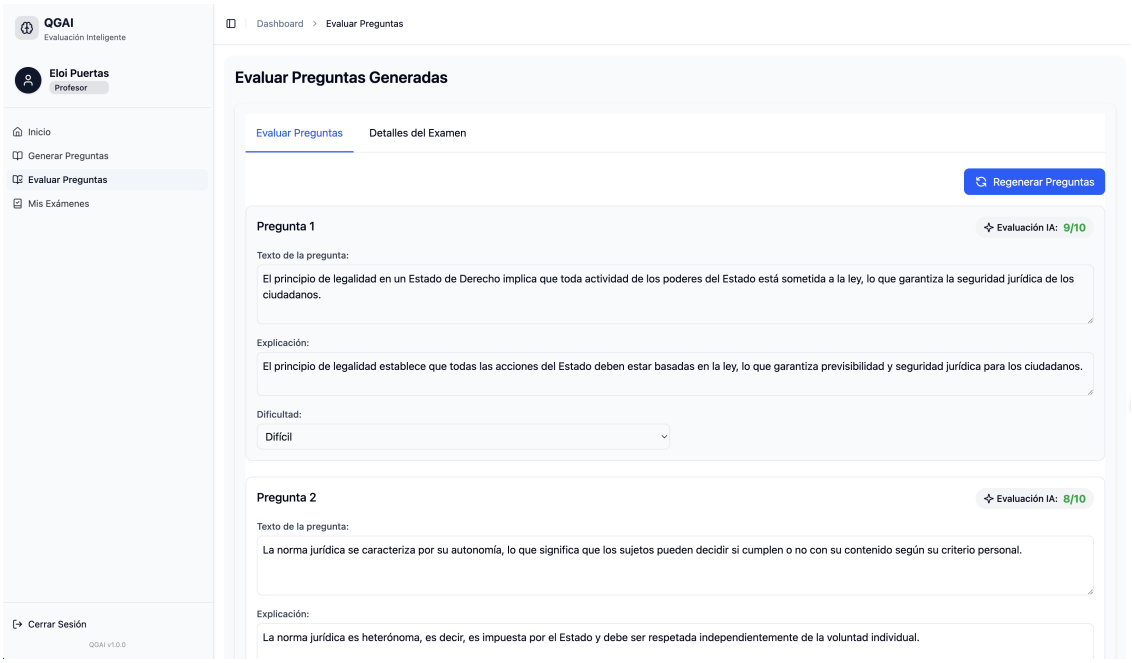

Figura 17: Evaluar preguntas generadas

Guardado del examen Esta vista posibilita realizar la configuración final del examen antes de proceder con su guardado. En este paso, se debe especificar el título del examen y asignar a los estudiantes que participarán en él. Esta asignación puede hacerse de dos maneras, mediante la importación de un archivo en formato .csv que contenga las direcciones de correo de los estudiantes o añadiendo los correos manualmente, uno por uno. En caso de intentar guardar el examen sin haber asignado al menos un estudiante, el sistema mostrará un mensaje emergente indicando que es necesario introducir al menos una dirección de correo electrónico para poder continuar.

Además, si es necesario realizar algún ajuste en las preguntas del examen, se ofrece la opción de regresar a la vista de preguntas y realizar modificaciones. Una vez que todo esté configurado correctamente, el usuario puede hacer clic en "Guardar examen"para completar el proceso. La Figura 18 ilustra cómo se presenta esta pantalla, mostrando las opciones disponibles para finalizar la configuración del examen.

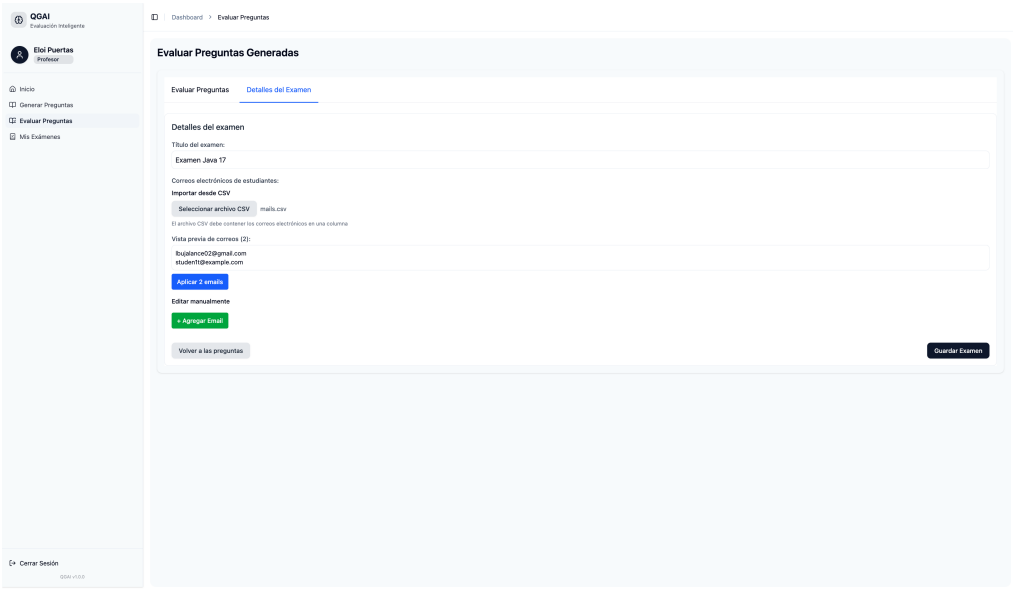

Figura 18: Vista guardado del examen

Exámenes creados por el profesor Desde la sección "Mis exámenes", el profesor accede a una vista que muestra un resumen de los exámenes creados. Se presentan indicadores KPI, como el número de alumnos asignados, el número de exámenes completados y la nota media de la asignatura calculada a partir de los exámenes respondidos. Además se muestra la fecha de creación del examen y se ofrece la posibilidad de editar la lista de estudiantes mediante el botón "Editar", que permite añadir o quitar alumnos.

Además, se visualiza un listado de estudiantes que ya han respondido, junto con la fecha de entrega y una acción para ver sus respuestas. Podemos ir a “Ver examen” para consultar el examen creado con las respuestas generadas por la IA.

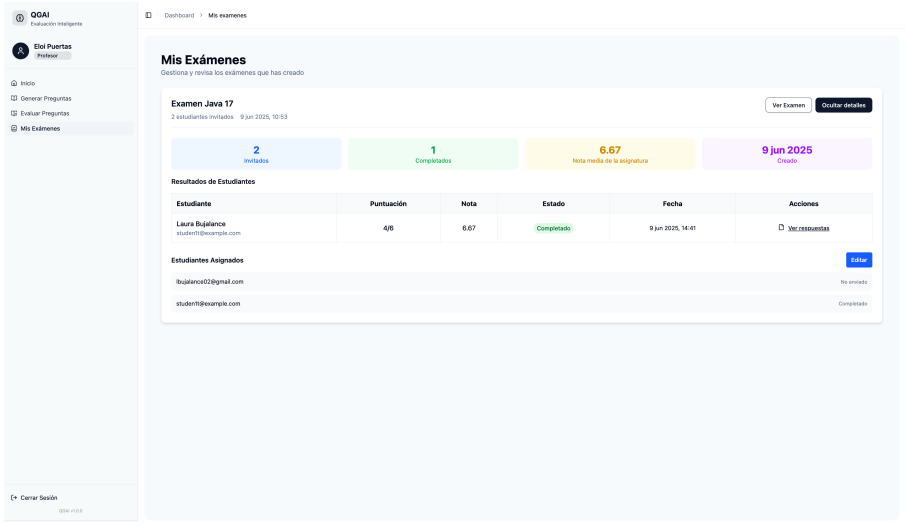

Figura 19: Vista de exámenes creados

Visualizar examen creado En esta pantalla que se accede desde “Mis exámenes”se visualiza el contenido completo del examen generado. Se muestran todas las preguntas junto con su respuesta correcta, la explicación proporcionada por la IA y el nivel de dificultad asignado (fácil, medio o difícil). La Figura 20 muestra esta pantalla.

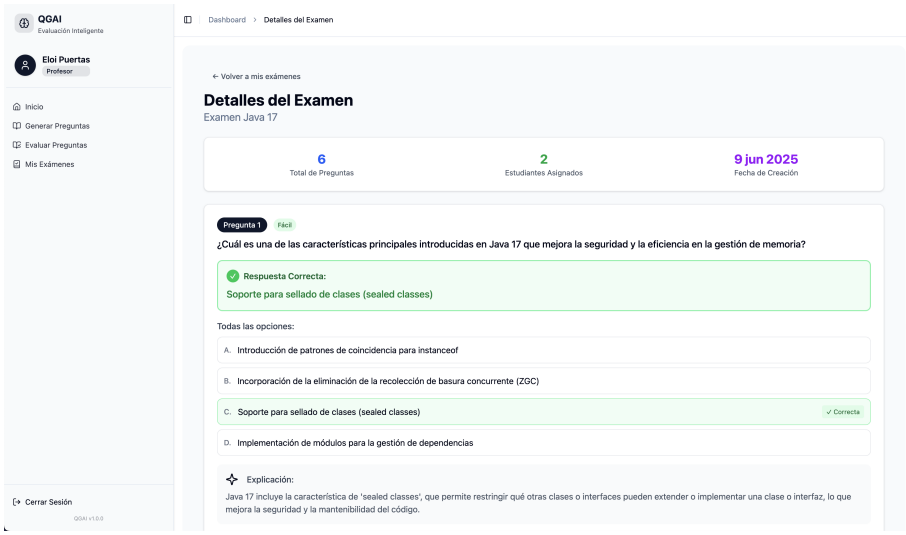

Figura 20: Vista de visualización del examen generado

Resultados del alumno Desde la sección "Mis exámenes", al seleccionar la acción "Ver respuestas" de un estudiante, accede a una pantalla donde se muestran los resultados individuales del alumno. En ella se pueden revisar las respuestas introducidas, el *feedback* generado automáticamente por la IA para cada pregunta, la puntuación obtenida (considerando que cada pregunta vale 1 punto) y la nota total sobre 10. Además, se incluye un *feedback* general del examen, generado a partir del *prompt* específico para ello. La Figura 21 muestra esta pantalla.

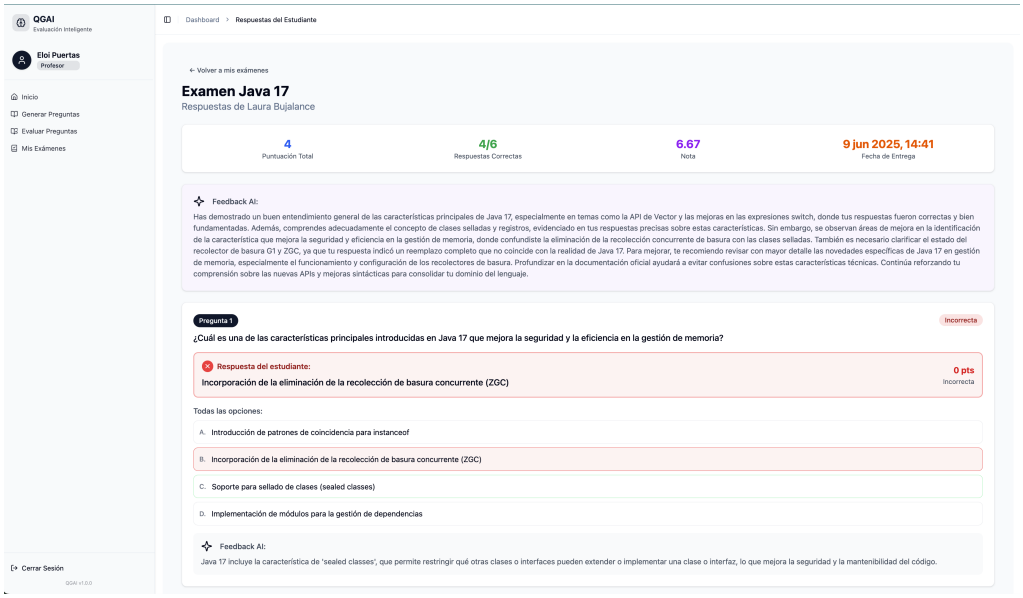

Figura 21: Vista de los resultados detallados de un alumno

7.4.2. Vistas del estudiante

Pantalla de inicio del alumno Esta vista se muestra tras iniciar sesión con un perfil de estudiante. Ofrece acceso directo a los exámenes asignados, así como al historial de evaluaciones realizadas y sus resultados. La interfaz está diseñada para ser simple, clara y centrada en facilitar la navegación del alumno. La Figura 22 muestra esta pantalla.

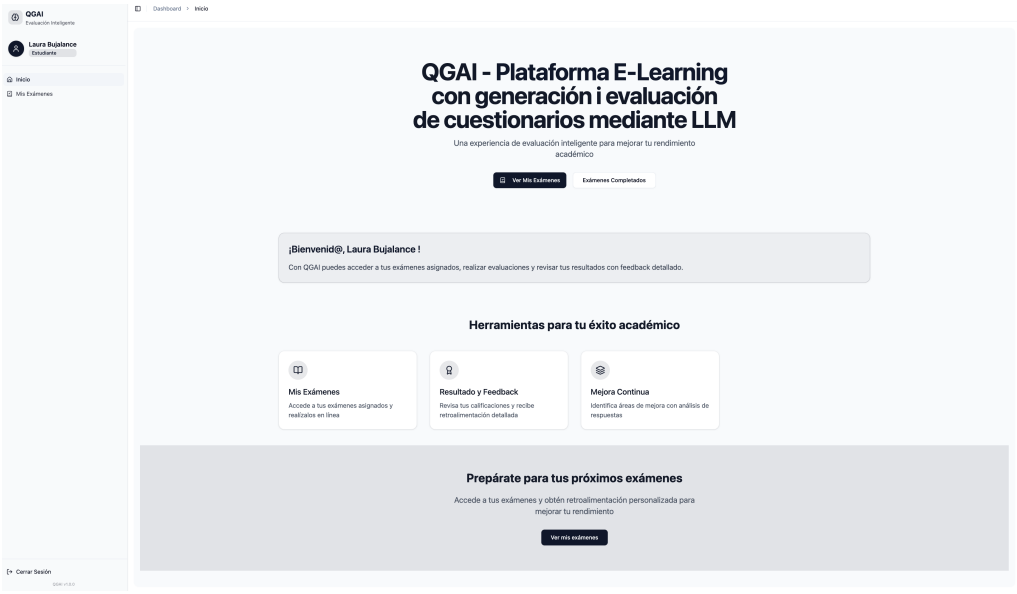

Figura 22: Pantalla de inicio del perfil estudiante

Vista examen alumno Esta vista se encuentra en la pestaña "Mis exámenes". Muestra al estudiante una lista de los exámenes que tiene asignados, diferenciando entre los que ya ha completado y los que aún están pendientes. En estos últimos aparece un botón para comenzar la evaluación, identificado como "Realizar examen". La Figura 25 muestra ambas pantallas.

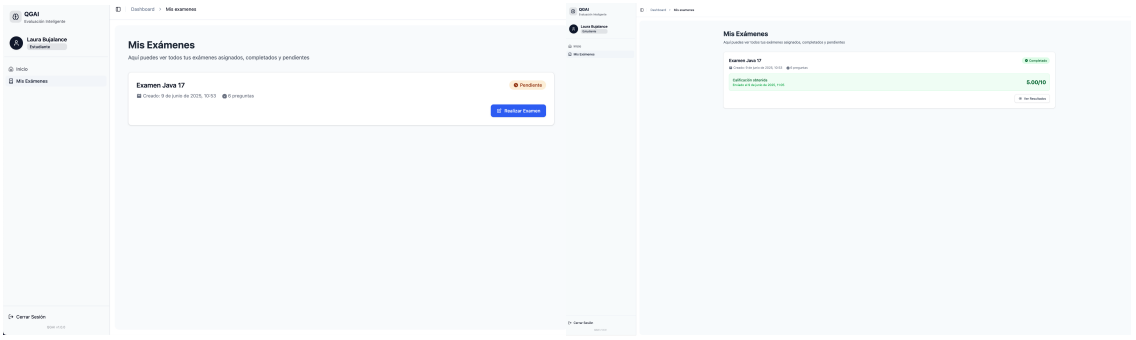

Figura 23: (a) Exámenes pendientes Figura 24: (b) Exámenes completados

Figura 25: Vista de los exámenes asignados al alumno

Vista realizar examen Desde la pestaña “Mis exámenes”, al seleccionar “Realizar examen”, el estudiante accede a la interfaz de resolución del examen. Esta vista permite responder a preguntas de opción múltiple y verdadero/falso mediante selección directa, y proporciona un campo de texto para las preguntas de desarrollo. La pantalla también muestra un temporizador con el tiempo restante para completar el examen y un botón para enviar las respuestas una vez finalizado. La Figura 26 muestra esta vista.

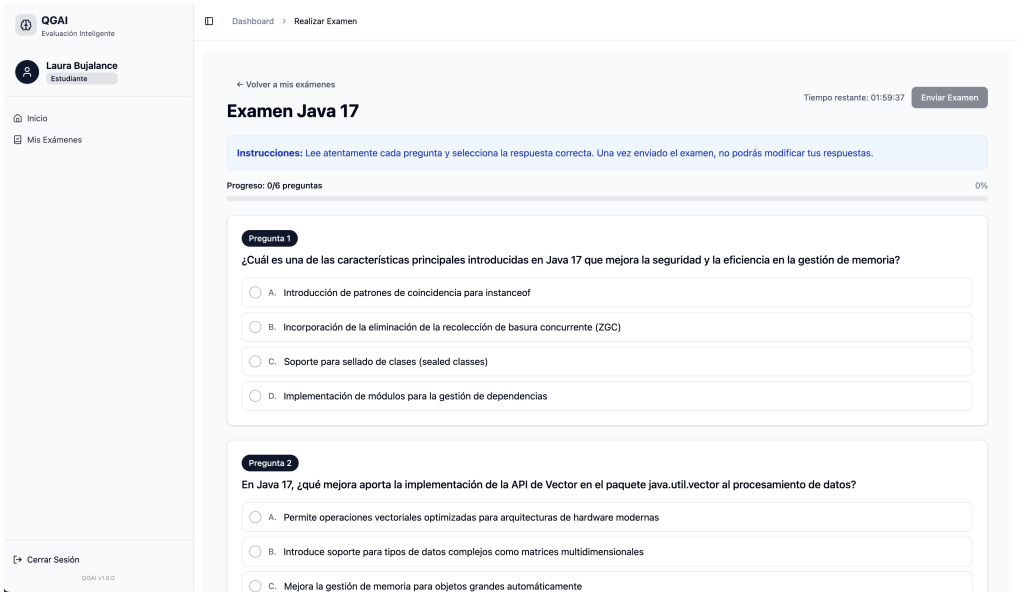

Figura 26: Vista de realización del examen por parte del alumno

Vista visualizar resultados examen Desde la pestaña“ Mis exámenes”, al seleccionar “Ver resultados”, el estudiante accede a la vista con los resultados detallados del examen. Se muestra el *feedback* de todo el examen generado por la IA, así como la respuesta correcta y el *feedback* específico para cada pregunta. Además, se presenta la puntuación obtenida y la nota final sobre 10. La Figura 27 muestra esta pantalla.

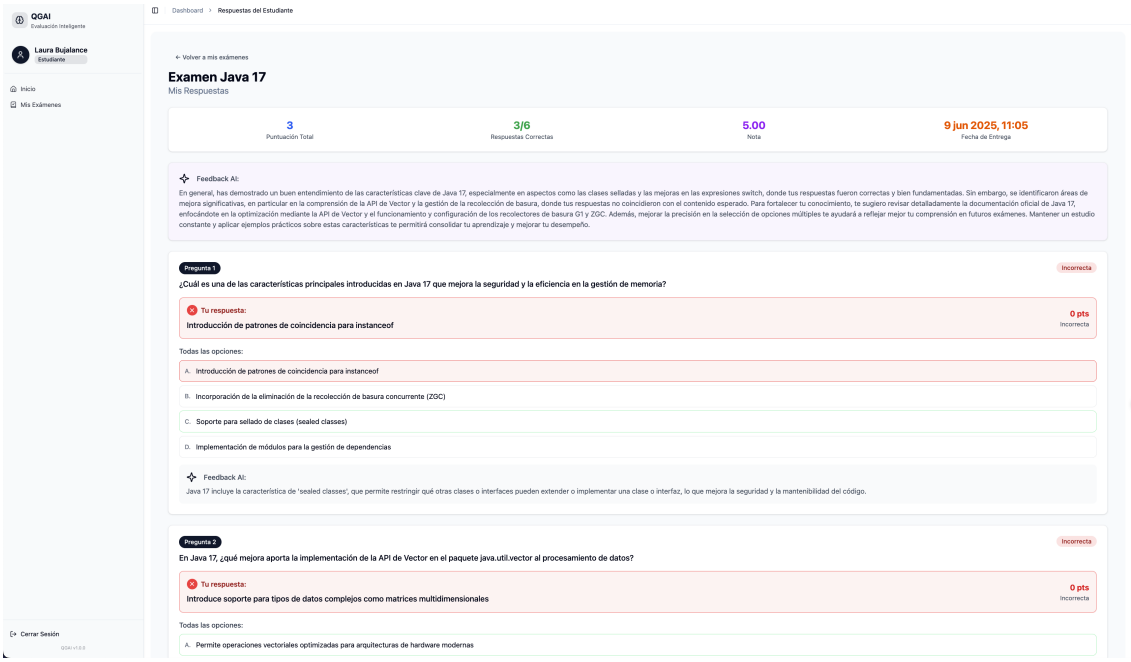

Figura 27: Vista de resultados del examen desde el perfil del alumno

8. Despliegue local

Para ejecutar el proyecto en local de forma rápida, se ha optado por el uso de contenedores *Docker*. Tanto el *frontend* como el *backend* cuentan con su propio archivo `Dockerfile` encargado de la creación del contenedor de la aplicación y la base de datos *MongoDB* se configura directamente en el archivo `compose.yml` el cual se encarga a través de un único archivo de levantar todos los servicios de manera conjunta, lo que permite tener todo el entorno configurado y listo con una sola instrucción. Además, este enfoque permite mantener una estructura clara del despliegue y se podrá usar para una futura integración en entornos de producción en plataforma de servicios en la nube como puede ser Azure mediante su servicio de contenedores serveless *Azure Container App* que permite desplegar aplicaciones *dockerizadas* mediante *Github*. En este Anexo manual de usuario 11 se detalla como ejecutar el despliegue en local usando `Dockerfile` y `compose.yml`.

9. Resultados

El diseño de las diferentes *prompts* ha ido progresivamente mejorando mediante un proceso iterativo de prueba y error, partiendo de la versión inicial (v0.1) hasta alcanzar las versiones actuales (v2.0). En estas versiones más avanzadas, se ha incorporado el uso de *DeepEval* [49] para un *testing* más elaborado.

DeepEval es una herramienta que evalúa las respuestas de los LLMs aplicando métricas que se pueden configurar. Se utilizaron preguntas sobre la asignatura de derecho, dirigidas a estudiantes de primer año de universidad. Se evaluó la relevancia de las respuestas proporcionadas por el modelo respecto a las preguntas originales mediante la métrica `AnswerRelevancyMetric` con GPT-4.1 mini.

Cada test se formuló incluyendo el contexto general, la pregunta esperada y la respuesta generada por el sistema. La evaluación asignaba una puntuación de relevancia a cada caso, considerando exitosas aquellas que superaban un umbral de 0.5. En todas las pruebas realizadas se obtuvo la puntuación máxima (1.0), lo que demuestra que las respuestas generadas eran coherentes con las preguntas planteadas.

Además, antes de que las preguntas lleguen al profesor, pasan por el *prompt* de evaluación que le asigna una puntuación automáticamente a la calidad de las preguntas generadas por el modelo. Esta *prompt*, contiene una rúbrica que considera varios aspectos que suman puntos hasta obtener un total de 10. Se ha podido observar que las preguntas de nivel "fácil" tienden a recibir puntuaciones más bajas, lo cual sugiere que el modelo tiene más dificultad para generar preguntas simples que mantengan un nivel pedagógico adecuado. Por el contrario, las preguntas de nivel "difícil" obtuvieron puntuaciones más altas, lo que indica que el LLM es capaz de generar contenido más con mayor coherencia.

A pesar de los buenos resultados, se detectaron algunos patrones que se repiten con frecuencia en las respuestas, como por ejemplo que la respuesta correcta suele estar siempre en primer lugar. Este tipo de problemas muestra que todavía hay margen para mejorar el diseño de los *prompts*. Asimismo, estas observaciones refuerzan la necesidad de mantener una supervisión por parte del docente para supervisar el contenido generado.

En las primeras versiones del sistema, también se observó que, en el caso de las preguntas de desarrollo, el modelo a veces generaba enunciados que hacían referencia a textos o fragmentos que no estaban presentes en la pregunta ni en el archivo proporcionado por ejemplo expresiones como "en función de este texto", dando

a entender que el contenido estaba presente. Cuando se incorporó la *prompt* de evaluación, se solucionó este problema porque todas aquellas que tienen una baja calificación y se regeneran automáticamente hasta alcanzar el umbral definido antes de mostrarlas al profesor.

Se detectó también un inconveniente al trabajar con documentos PDF utilizados para generar las preguntas. Si contienen un alto número de imágenes, es posible que no se puedan leer completamente, lo que puede afectar la precisión de las preguntas generadas.

Por último, el diseño del *prompt* para la generación de feedback del examen global, devolvió en las primeras iteraciones respuestas excesivamente largas. Para solucionar este problema se puso en el *prompt* una limitación en el número de caracteres, obteniendo así una respuesta más corta.

10. Conclusiones

La arquitectura que se ha desarrollado ha funcionado bien y ha sabido adaptarse a lo que necesitan tanto los profesores como los estudiantes. A partir de la integración de tecnologías como *FastAPI*, *LangChain*, *Vue.js* y *MongoDB*, se ha podido construir una plataforma estable, fácil de mantener y con potencial para seguir creciendo. *LangChain*, en particular, ha sido muy útil para conectar con diferentes proveedores de modelos proporcionando más flexibilidad al sistema.

Para los profesores, la plataforma permite generar preguntas de forma rápida y les da una primera idea sobre la calidad gracias a evaluación automática. Además, ir modificando los *prompts* de forma iterativa ha ayudado a ajustar el comportamiento del modelo según lo que se buscaba. Aun así, se ha comprobado que dejar todo el proceso sin intervención del docente no es recomendable, ya que el modelo sigue cometiendo errores y necesita supervisión para garantizar buenos resultados.

En el caso de los estudiantes, la experiencia también ha sido positiva. La herramienta les permite autoevaluarse de manera fácil, y reciben *feedback* tanto para cada pregunta como para el examen completo. Además, cuando se trata de preguntas desarrollo, el sistema ofrece comentarios personalizados fomentando la autorregulación del alumno.

Más allá de lo técnico, este proyecto también ha sido una buena oportunidad para poner en práctica muchos de los conocimientos adquiridos a lo largo de la carrera. Aunque ya se han conseguido buenos resultados, todavía hay aspectos por mejorar y probar. Eso abre una línea de trabajo interesante para el futuro y da la posibilidad de seguir aumentando el valor educativo de la plataforma.

11. Trabajo futuro

Una primera mejora que incorporar sería implementar técnicas RAG para que el modelo pueda consultar materiales del curso relacionadas con el tema introducido antes de generar las preguntas, esto haría que las respuestas fueran mucho más precisas y se podría incluso referenciar la pregunta de qué contenido ha sido extraído para proporcionar más fiabilidad.

Otra posibilidad sería ofrecer al profesorado la posibilidad de en un examen generado poner sus propias preguntas de manera manual y ofrecer la posibilidad de que el *prompt* evaluador califique estas preguntas.

Actualmente, todas las preguntas generadas por la plataforma tienen el mismo valor dentro del examen. Una mejora sería permitir que el profesorado asigne una puntuación personalizada a cada pregunta, ya sea manualmente o mediante alguna regla establecida en función del nivel de dificultad.

Finalmente, sería útil incorporar la opción de que el docente revise y edite el *feedback* generado por la IA, tanto en las respuestas individuales como en el genérico del examen, para reforzar determinados conceptos o introducir observaciones específicas según el grupo de estudiantes.

Referencias

[1] Rosero Guevara J, Guevara Aulestia D. Exploración del Uso de la Inteligencia Artificial en la Eficiencia de Entornos Virtuales LMS de E-Learning para la Educación Secundaria. Ciencia Latina Revista Científica Multidisciplinar.[Internet] 2025 Jan;8(6). Disponible en: https://www.researchgate.net/publication/387764471_Exploracion_del_Uso_de_la_Inteligencia_Artificial_en_la_Eficiencia_de_Entornos_Virtuales_LMS_de_E-Learning_para_la_Educacion_Secundaria

[2] Chango Sangucho MD, Cóndor Bassantes ST, Bautista Cevallos SG, Ilaquiche Llano PB. Transformando el aprendizaje: el impacto de la inteligencia artificial en la educación contemporánea. Revista Imaginario Social [Internet]. 2025 Jan 6 [citado 2025 May 20];8(1). Disponible en: https://www.revista-imaginariosocial.com/index.php/es/article/view/260

[3] Ilarionov O, Krasovska H, Domanetska I, Fedusenko O. Features of the practical use of LLM for generating quiz. IT&I [Internet]. 2024 [citado el 3 de junio de 2025];275–84. Disponible en: https://ceur-ws.org/Vol-3909/Paper_22.pdf

[4] Stryker C, Scapicchio M. What is generative AI? [Internet]. Ibm.com. 2025 [citado el 5 de junio de 2025]. Disponible en: https://www.ibm.com/think/topics/generative-ai

[5] Bernal M. Revolutionizing eLearning Assessments: The Role of GPT in Crafting Dynamic Content and Feedback. Journal of Artificial Intelligence and Technology.[Internet] 2024 May 6; Disponible en: https://www.researchgate.net/publication/380381495_Revolutionizing_eLearning_Assessments_The_Role_of_GPT_in_Crafting_Dynamic_Content_and_Feedback

[6] ElSayary A, Kuhail MA, Hojeij Z. Examining the Role of Prompt Engineering in Utilizing Generative AI Tools for Lesson Planning: Insights From Teachers' Experiences and Perceptions. Strzelecki A, editor. Human Behavior and Emerging Technologies. [Internet] 2025 Jan;2025(1). Disponible en: https://www.researchgate.net/publication/390156158_Examining_the_Role_of_Prompt_Engineering_in_Utilizing_Generative_AI_Tools_for_Lesson_Planning_Insights_From_Teachers'_Experiences_and_Perceptions

[7] Hagos DH, Battle R, Rawat DB. Recent advances in generative AI and Large Language Models: Current status, challenges, and perspectives [Internet]. arXiv [cs.CL]. 2024. Disponible en: http://arxiv.org/abs/2407.14962

[8] Vaswani A, Shazeer N, Parmar N, Uszkoreit J, Jones L, Gomez AN, et al. Attention is all you need [Internet]. arXiv [cs.CL]. 2017. Disponible en: http://arxiv.org/abs/1706.03762

[9] Radford A, Narasimhan K, Salimans T, Sutskever I. Improving Language Understanding by Generative Pre-Training [Internet]. 2018. Disponible en: https://cdn.openai.com/research-covers/language-unsupervised/language_understanding_paper.pdf

[10] Devlin J, Chang M-W, Lee K, Toutanova K. BERT: Pre-training of deep bidirectional Transformers for language understanding [Internet]. arXiv [cs.CL]. 2018. Disponible en: http://arxiv.org/abs/1810.04805

[11] Phoenix J. Prompt engineering para inteligencia artificial generativa: cómo elaborar preguntas para obtener respuestas fiables y relevantes con la IA.[Internet] Marcombo; 2025. Disponible en: https://elibro-net.sire.ub.edu/es/ereader/craiub/281774

[12] Guía de Ingeniería de Prompt [Internet]. Promptingguide.ai. [citado el 25 de abril de 2025]. Disponible en: https://www.promptingguide.ai/es

[13] Punyakeerthi BL. *Token Embeddings - Punyakeerthi BL - Medium* [Internet]. Medium. 2024 [cited 2025 May 17]. Available from: https://medium.com/@punya8147_26846/token-embeddings-3b249565fe60

[14] Walter Y. Embracing the future of Artificial Intelligence in the classroom: the relevance of AI literacy, prompt engineering, and critical thinking in modern education. Int J Educ Technol High Educ [Internet]. 2024;21(1). Disponible en: https://educationaltechnologyjournal.springeropen.com/articles/10.1186/s41239-024-00448-3

[15] Kolbe M. The rise of GPUs: How they became essential for large language models [Internet]. Medium. 2024 [citado el 24 de mayo de 2025]. Disponible en: https://medium.com/@manikolbe/the-rise-of-gpus-how-they-became-essential-for-large-language-models-02f0c27f68ea

[16] Geissler F, Roscher K, Trapp M. Concept-guided LLM agents for Human-AI safety codesign [Internet]. arXiv [cs.SE]. 2024. Disponible en: http://arxiv.org/abs/2404.15317

[17] Geroimenko V. The Essential Guide to Prompt Engineering: Key Principles, Techniques, Challenges, and Security Risks. [Internet] Cham: Springer; 2025. Disponible en: https://doi.org/10.1007/978-3-031-86206-9

[18] Stryker C, Scapicchio M. What is generative AI? [Internet]. Ibm.com. 2024. Disponible en: https://www.ibm.com/think/topics/generative-ai

[19] Raj R, Mathew AT, Ouseph M, Philiphose LS, Joseph S. The Impact of E-Learning Platforms.[Internet] International Research Journal on Advanced Engineering and Management. 2024 Dec;2(12):3511–3518. Disponible en: https://www.researchgate.net/publication/386899062_The_Impact_of_E-Learning_Platforms

[20] Center for Research Evaluation, University of Mississippi: Review of Literature on Student Outcomes in Online Learning [Internet]. 2024 [cited 2025 May 10]. Disponible en: https://www.ibo.org/globalassets/new-structure/research/pdfs/dp-online-learning-literature-review.pdf

[21] Flodén J. Grading exams using large language models: A comparison between human and AI grading of exams in higher education using ChatGPT. Br Educ Res J [Internet]. 2025;51(1):201–24. Disponible en: http://dx.doi.org/10.1002/berj.4069

[22] Larsen SK. Creating large language model resistant exams: Guidelines and strategies [Internet]. arXiv [cs.CL]. 2023. Disponible en: http://arxiv.org/abs/2304.12203

[23] Wiegreffe S, Tafjord O, Belinkov Y, Hajishirzi H, Sabharwal A. Answer, assemble, ace: Understanding how LMs answer multiple choice questions [Internet]. arXiv [cs.CL]. 2024. Disponible en: http://arxiv.org/abs/2407.15018

[24] Lohr D, Berges M, Chugh A, Kohlhase M, Müller D. Leveraging large language models to generate course-specific semantically annotated learning objects [Internet]. arXiv [cs.AI]. 2024. Disponible en: http://arxiv.org/abs/2412.04185

[25] Zhang Y, Liu X, Sun Y, Alharbi A, Alzahrani H, Alomair B, et al. Can LLMs design good questions based on context? [Internet]. arXiv [cs.CL]. 2025. Disponible en: http://arxiv.org/abs/2501.03491

[26] Wróblewska A, Grabek B, Świstak J, Dan D. Evaluating LLM-generated QA test: A student-centered study [Internet]. arXiv [cs.CL]. 2025 [citado el 7 de junio de 2025]. Disponible en: http://arxiv.org/abs/2505.06591

[27] Ma A, Martínez ER. Retroalimentación formativa con inteligencia artificial generativa: Un caso de estudio. Wimb Lu [Internet]. 2024 Dec 18;19(2):1–20.Disponible en: https://www.researchgate.net/publication/391177895_Retroalimentacion_formativa_con_inteligencia_artificial_generativa_Un_caso_de_es

[28] Nazari M, Saadi G. Developing effective prompts to improve communication with ChatGPT: a formula for higher education stakeholders. Discover Education. [Internet] 2024 Apr 29;3(1). Disponible en: https://link.springer.com/article/10.1007/s44217-024-00122-w

[29] Lee D, Palmer E. Prompt engineering in higher education: a systematic review to help inform curricula.[Internet] *International Journal of Educational Technology in Higher Education*. 2025 Feb 9;22(1). Disponible en: https://educationaltechnologyjournal.springeropen.com/articles/10.1186/s41239-025-00503-7

[30] Park, J. y Choo, M. (2024). Generative AI Prompt Engineering for Educators: Practical Strategies. [Internet] Disponible en: https://www.damiantgordon.com/WebLessons/PromptEngineering/Papers/Generative_AI_Prompt_Engineering_for_Educators_Practical_Strategies.pdf

[31] Fu T, Conde J, Martínez G, Grandury M, Reviriego P. Multiple choice questions: Reasoning makes large language models (LLMs) more self-confident even when they are wrong [Internet]. arXiv [cs.CL]. 2025. Disponible en: http://arxiv.org/abs/2501.09775

[32] Pawar P, Dube R, Joshi A, Gulhane Z, Patil R. Automated generation and evaluation of MultipleChoice quizzes using langchain and Gemini LLM. En: 2024 International Conference on Electrical Electronics and Computing Technologies (ICEECT). [Internet] IEEE; 2024. p. 1–7. Disponible en: https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=10739326

[33] Woo, D. J., Wang, D., Yung, T., Guo, K. (2024). Effects of a Prompt Engineering Intervention on Undergraduate Students' AI Self-Efficacy, AI Knowledge and Prompt Engineering Ability: A Mixed Methods Study. [Internet] Disponible en: https://arxiv.org/abs/2408.07302

[34] Ko, Y., et al. (2024). Evaluating Prompt-Generated Feedback for Teacher Education: A Comparative Study with Human Experts. [Internet] Journal of Intelligence, 6(2), 35. Disponible en: https://www.mdpi.com/2673-2688/6/2/35

[35] Elkins S, Kochmar E, Cheung JCK, Serban I. How useful are educational questions generated by large language models? [Internet]. arXiv [cs.CL]. 2023. Disponible en: http://arxiv.org/abs/2304.06638

[36] Shamkuwar, M., Jindal, P., More, R., Patil, P., Mahamuni, P. (2023). Artificial intelligence and higher education: a systematic visualizations based review. Journal of e-Learning and Knowledge Society, [Internet] 19(3), 36–42. Disponible en: https://doi.org/10.20368/1971-8829/1135857

[37] Introducing Claude for education [Internet]. Anthropic.com. [citado el 5 de mayo de 2025]. Disponible en: https://www.anthropic.com/news/introducing-claude-for-education

[38] Studio - Educacion Superior [Internet]. Instructure. [citado el 5 de mayo de 2025]. Disponible en: https://www.instructure.com/es/educacion-superior/productos/canvas/canvas-studio

[39] AI Subsystem [Internet]. Moodledev.io. [citado el 4 de mayo de 2025]. Disponible en: https://moodledev.io/docs/5.0/apis/subsystems/ai

[40] Moodle plugins directory: Wooclap [Internet]. Moodle.org. [citado el 4 de mayo de 2025]. Disponible en: https://moodle.org/plugins/mod_wooclap

[41] Zhang X, Zhang C, Sun J, Xiao J, Yang Y, Luo Y. EduPlanner: LLM-based multi-agent systems for customized and intelligent instructional design [Internet]. arXiv [cs.AI]. 2025. Disponible en: http://arxiv.org/abs/2504.05370

[42] Llama 3.1 requirements [Internet]. Llama Ai Model. 2024 [citado el 10 de mayo de 2025]. Disponible en: https://llamaimodel.com/requirements/

[43] OpenAI platform [Internet]. Openai.com. [citado el 20 de abril de 2025]. Disponible en: https://platform.openai.com/docs/models/compare

[44] Precios de la API para desarrolladores de Gemini [Internet]. Google AI for Developers. [citado el 20 de abril de 2025]. Disponible en: https://ai.google.dev/gemini-api/docs/pricing?hl=es-419

[45] Pricing [Internet]. Anthropic. [citado el 8 de junio de 2025]. [citado el 20 de abril de 2025]. Disponible en: https://docs.anthropic.com/en/docs/about-claude/pricing

[46] OpenAI platform [Internet]. Openai.com. [citado el 20 de abril de 2025]. Disponible en: https://platform.openai.com/docs/guides/rate-limits/overview

[47] OpenAI platform [Internet]. Openai.com. [citado el 8 de junio de 2025]. Disponible en: https://platform.openai.com/docs/models/gpt-4.1

[48] Pdfminer [Internet]. PyPI. [citado el 8 de junio de 2025]. Disponible en: https://pypi.org/project/pdfminer/

[49] B G, Purwar A. Evaluating the efficacy of open-source LLMs in enterprise-specific RAG systems: A comparative study of performance and scalability [Internet]. arXiv [cs.IR]. 2024. Disponible en: http://arxiv.org/abs/2406.11424

Anexos

Anexo A: Manual de Usuario

Dentro del repositorio/carpeta de "qgai-backend"se encuentra el mismo archivo en formato .md de nombre README_DESPLIEGUE.md"

Requisitos previos

- Docker
- Docker Compose

Configuración inicial Hay dos maneras de iniciar el proyecto:

- Clonando el repositorio (rama master)
- Descomprimiendo el archivo zip `qgai-app.zip`

```
mkdir qgai
cd qgai
git clone https://github.com/lbujalance/qgai-backend
git clone https://github.com/lbujalance/qgai-frontend
cd qgai/qgai-backend
```

2. Configurar las variables de entorno

- Para el backend: Crea un archivo `.env.docker` en la carpeta `qgai-backend/` a partir del `.env.docker.example`: **Solo modificar:**

  AI_API_KEY=your-ai-api-key-here

- Para el frontend: Es necesario modificar el archivo `.env.example` quitándole la extensión `.example` y renombrarlo a `.env`

  VITE_APP_QGAI_API_URL=http://localhost:8000/api

3. Despliegue con Docker Compose

1. Construir e iniciar los contenedores:

   ```
   docker compose up -d
   ```

2. Para reconstruir los contenedores después de cambios:

   ```
   docker compose up -d --build
   ```

3. Para detener los servicios:

```
docker compose down
```

4. Acceso a la aplicación Una vez desplegada, puedes acceder a:

- Frontend: http://localhost:80
- Backend API: http://localhost:8000
- Documentación API: http://localhost:8000/docs
- MongoDB: `mongodb://localhost:27018` (puerto expuesto)

Servicios desplegados

- **MongoDB**: Base de datos NoSQL
- **Backend (qgai-backend)**: API REST en FastAPI (Python)
- **Frontend (qgai-frontend)**: Interfaz de usuario en Vue.js

Gestión de datos persistentes Los datos de MongoDB se almacenan en un volumen Docker para persistencia entre reinicios:

```
volumes:
  mongodb_data:
    driver: local
```

Anexo B: Historias de Usuario

US01: Inicio de Sesión como Profesor

Como profesor, quiero iniciar sesión o registrarme seleccionando mi rol, para acceder a las funciones de generación y gestión de exámenes.
Criterios de aceptación:

- Formulario de inicio de sesión con usuario y contraseña.
- Opción de registro con rol "Profesor".
- Validación de los datos introducidos.

US02: Configurar Generación de Preguntas

Como profesor, quiero definir un tema, nivel educativo, número y tipo de preguntas con su dificultad, para generar preguntas adaptadas.
Criterios de aceptación:

- Campo de texto para tema.

- Selector de nivel educativo (Primaria, Secundaria, Bachillerato, etc.).
- Inputs para número de preguntas por tipo (opción múltiple, V/F, desarrollo respuesta corta).
- Inputs para distribución de dificultad (fácil, medio, difícil).
- Campo de texto opcional para información adicional.
- Ver el modelo utilizado
- Botón "Generar preguntas" habilitado solo si se cumplen los requisitos mínimos, título, nivel educativo y cantidad de preguntas, dificultad (en función del número de preguntas).

US03: Subir Fichero

Como profesor, quiero subir un PDF, para generar preguntas basadas en su contenido.
Criterios de aceptación:

- Permite un fichero PDF.
- Posibilidad de continuar sin subir archivos.
- Validación del formato PDF.
- Mostrar un mensaje informativo "Si sube un pdf, las preguntas generadas saldrán a partir de la información del mismo".

US04: Generación Automática con Evaluación

Como profesor, quiero que las preguntas se generen automáticamente según mis configuraciones y que sean evaluadas por calidad antes de mostrarse.
Criterios de aceptación:

- Primera *prompt* genera preguntas según los datos.
- Segunda *prompt* evalúa las preguntas según métricas preestablecidas.
- Puntuación visible de la puntuación generada por cada pregunta.
- Diferenciación entre tipos de pregunta:
  - **Opción múltiple**
  - **Verdadero/Falso**
  - **Desarrollo respuesta corta**
- En el panel izquierdo "Mis exámenes"poder ver los exámenes.

US05: Revisión y Edición de Preguntas

Como profesor, quiero revisar, editar o regenerar las preguntas generadas, para revisar el examen.
Criterios de aceptación:

- Vista previa editable de cada pregunta.
- Selector de preguntas y botón "Regenerar seleccionadas".

US06: Guardar Examen

Como profesor, quiero guardar un conjunto de preguntas como un examen, para reutilizarlo en futuras evaluaciones.
Criterios de aceptación:

- Botón "Guardar Examen" tras la edición de preguntas.
- Guardado con título.

US07: Asignar Examen a Alumnos

Como profesor, quiero asignar un examen a uno o varios estudiantes, para que ellos puedan evaluarse.
Criterios de aceptación:

- Permitir asignar alumnos a través de un Input.
- Permitir subir alumnos por fichero CSV.
- Visualización del estado de entrega por alumno (pendiente, completado, corregido).

US08: Seguimiento de Calificaciones por Examen

Como profesor quiero ver la lista de alumnos con sus calificaciones, para hacer seguimiento del progreso.
Criterios de aceptación:

- Al seleccionar un examen, se muestra la lista de alumnos asignados junto con sus calificaciones.

US09: Inicio de Sesión como Estudiante

Como estudiante, quiero iniciar sesión o registrarme seleccionando mi rol, para poder acceder a mis exámenes y responder.
Criterios de aceptación:

- Formulario de inicio de sesión con usuario y contraseña.
- Opción de registro con rol "Estudiante".
- Validación de datos obligatoria.

US10: Acceso al Examen

Como estudiante, quiero acceder al examen que me asignó el profesor, para completarlo dentro del sistema.
Criterios de aceptación:

- Panel izquierdo con vista a "Hacer examen".
- Visualización clara de preguntas por tipo:
  - **Opción múltiple**
  - **Verdadero/Falso**

- **Desarrollo respuesta corta**

- Vista de la puntuación de cada pregunta.

US11: Envío de Respuestas

Como estudiante, quiero enviar mis respuestas al terminar el examen, para ser evaluado.
Criterios de aceptación:

- Botón "Enviar examen".
- Confirmación de envío.

US12: Ver Resultados y Feedback

Como estudiante, quiero ver mi calificación final y una explicación de los aciertos y errores, para aprender del proceso.
Criterios de aceptación:

- Resumen con puntuación total y por pregunta.
- *Feedback* automático.

Anexo C: Prompts

Prompt para generar las preguntas a partir de un PDF

> Actúa como un especialista en didáctica educativa con amplia experiencia en la creación de evaluaciones de alta calidad para distintas etapas educativas. **Tarea:** Analiza el siguiente tema y diseña una batería de preguntas tipo test destinadas a evaluar la comprensión, el análisis crítico y la aplicación práctica de los contenidos. **Datos:** El profesor ha proporcionado su propio PDF con el tema del cual has de generar las preguntas, por lo que no es necesario que lo analices. No te salgas del tema del PDF proporcionado. Todas las preguntas deben de estar relacionadas con el tema del pdf y no debe haber preguntas fuera de él. **Todas las preguntas deben poder contestarse con el pdf relacionado**, ya que es la única fuente de información de estudio del estudiante. **Texto extraído del PDF:** {file_content} **Criterios que debes seguir para generar las preguntas: Preguntas de opción múltiple:**
>
> - Las preguntas deben tener entre 4 posibles respuestas, de las cuales solo una debe ser correcta.
> - Debes alternar la posición de la respuesta correcta entre las opciones para evitar patrones predecibles.
> - Las opciones incorrectas no deben dar pistas fáciles a la respuesta correcta a través de la gramática, asociaciones de palabras comunes o relaciones demasiado simples con la pregunta.
> - Las opciones "ninguna de las anteriores" y "todas las anteriores" deben usarse con cuidado y solo ocasionalmente ser la respuesta correcta.

- Cuando la pregunta sea una definición, usa el término a definir en la pregunta, no como una de las opciones.
- Todas las opciones de respuesta para una misma pregunta deben tener la misma estructura.
- Procura que las opciones tengan una extensión parecida; evita que la correcta sea siempre la más larga o corta.
- La pregunta debe ser clara, sin palabras innecesarias ni dobles sentidos.
- Solo debe haber una única respuesta claramente correcta.
- La respuesta no debe depender de otra pregunta del examen.
- La dificultad debe ser apropiada para el nivel educativo.
- Cada pregunta debe estar relacionada con un objetivo específico del temario.
- En ciencias o matemáticas, las opciones deben ser técnicamente correctas.
- Las opciones incorrectas deben ser plausibles para quienes no dominan el tema.
- Siempre que sea posible, fomenta la comprensión y aplicación del conocimiento.

Preguntas de verdadero/falso:

- La respuesta debe colocarse aleatoriamente entre las opciones a,b,c,d sin seguir un patrón predecible. No debe repetirse sistemáticamente en la misma posición. La posicion de la respuesta correcta debe variar entre las preguntas.
- Si hay relación causa-efecto, varía si la segunda parte es verdadera o falsa.
- Distribuye aleatoriamente las respuestas verdaderas y falsas.
- Usa errores comunes como afirmaciones falsas plausibles.
- La afirmación debe poder responderse de forma inequívoca como “Verdadero” o “Falso”.
- Evita afirmaciones con doble negación o frases excesivamente complejas.
- Sé cauteloso con afirmaciones absolutas como “siempre” o “nunca”.

Preguntas de respuesta corta:

- Usa un verbo claro que indique el tipo de respuesta (p.ej., Analizar, Comparar, Evaluar).
- Delimita adecuadamente el enfoque de la pregunta.
- Fomenta el pensamiento de orden superior (análisis, síntesis, aplicación).
- La pregunta debe estar vinculada con los objetivos de aprendizaje.
- Evita preguntas compuestas o que dependan de textos externos.
- Usa lenguaje claro y accesible.
- Proporciona el contexto necesario dentro de la propia pregunta.

Datos:

- Tema: {topic_questions}
- Nivel educativo: {nivel_educativo}
- Número total de preguntas: {total_num_questions}
- Distribución por tipo:
  - Preguntas de opción múltiple: {num_questions_multiple}
  - Preguntas de verdadero/falso: {num_questions_true_false}
  - Preguntas de respuesta corta: {num_questions_short_answer}
- Distribución por dificultad:
  - Fáciles: {num_questions_easy}
  - Media dificultad: {num_questions_medium}
  - Difíciles: {num_questions_hard}
- **Debes añadir obligatoriamente las respuestas para las preguntas de respuesta corta.**
- Consideraciones adicionales del profesor: {input_profe}

Instrucciones de estilo:

- Usa un lenguaje adecuado al nivel educativo del lector.
- Varía los subtemas para evitar repeticiones.
- Mantén una redacción neutral y culturalmente inclusiva.

Instrucciones finales:

- Revisa al final del ejercicio que las respuestas correctas estén distribuidas de forma equilibrada y sin patrones. Si es necesario, redistribúyelas.

El modelo debe responder con un JSON que cumpla la siguiente estructura:

```
{
  "questions": [
    {
      "question": "Texto de la pregunta",
      "type": "multiple_choice" | "true_false" | "short_answer",
      "options": [
        {"text": "Texto opción A", "is_correct": true/false},
        ...
      ],
      "correct_answer": "Texto o valor correcto",
      "explanation": "Justificación breve de la respuesta correcta",
      "difficulty": "easy" | "medium" | "hard",
      "skill": "tema o habilidad relacionada"
    },
    ...
  ]
}
```

Prompt para generar las preguntas con temática libre

Actúa como un especialista en didáctica educativa con amplia experiencia en la creación de evaluaciones de alta calidad para distintas etapas educativas. **Tarea:** Analiza el siguiente tema y diseña una batería de preguntas tipo test destinadas a evaluar la comprensión, el análisis crítico y la aplicación práctica de los contenidos. **Criterios que debes seguir para generar las preguntas: Pregunta de opción múltiple:**

- Las preguntas deben tener entre 4 posibles respuestas, de las cuales solo una debe ser correcta.
- La respuesta debe colocarse aleatoriamente entre las opciones a,b,c,d sin seguir un patrón predecible. No debe repetirse sistemáticamente en la misma posición. La posicion de la respuesta correcta debe variar entre las preguntas.
- Las opciones incorrectas no deben dar pistas fáciles a la respuesta correcta a través de la gramática, asociaciones de palabras comunes o relaciones demasiado simples con la pregunta.
- Las opciones "ninguna de las anteriores" y "todas las anteriores" deben usarse con cuidado y solo ocasionalmente ser la respuesta correcta.
- Cuando la pregunta sea una definición, utiliza el término a definir en la pregunta, no como una de las opciones.
- Todas las opciones de respuesta para una misma pregunta deben tener la misma estructura (por ejemplo, todas deben ser frases o todas oraciones).
- Procura que las opciones de respuesta tengan una extensión parecida. No hagas que la correcta sea siempre la más larga o la más corta.
- La pregunta debe ser fácil de entender, sin palabras innecesarias ni dobles sentidos.
- Solo debe haber una única respuesta claramente correcta.
- La respuesta no debe depender de la respuesta dada a otra pregunta del examen.
- La dificultad debe ser apropiada para el nivel educativo al que va dirigida.
- Cada pregunta debe estar directamente relacionada con un objetivo específico del temario.
- En ciencias o matemáticas, asegúrate de que las opciones numéricas o fórmulas sean correctas técnicamente.
- Las opciones incorrectas deben parecer posibles para quienes no dominan el tema.
- Siempre que sea posible, fomenta preguntas que requieran comprensión o aplicación del conocimiento.

Pregunta de verdadero/falso:

- Debes alternar entre afirmaciones verdaderas y falsas, asegurando que la proporción sea equilibrada.
- Si la afirmación establece una relación (causa-efecto, premisa-conclusión), varía si la segunda parte es verdadera o falsa.
- Distribuye aleatoriamente el orden de las afirmaciones verdaderas y falsas, evitando patrones predecibles.
- Utiliza errores comunes o creencias populares como afirmaciones falsas plausibles.
- La pregunta debe responderse inequívocamente con "Verdadero" o "Falso".
- Evita detalles innecesarios en las afirmaciones verdaderas solo para hacerlas más precisas.
- Procura no utilizar doble negación. Si es necesario, asegúrate de que estén redactadas con la mayor claridad posible.
- Evita afirmaciones excesivamente largas o complejas con múltiples informaciones.
- Sé cauteloso con afirmaciones absolutas como "todos", "ninguno", "siempre" o "nunca".
- Las afirmaciones con "a veces", "a menudo" o "generalmente" pueden ser verdaderas incluso con excepciones.

Pregunta de respuesta corta:

- Utiliza un verbo de tarea preciso que indique claramente el tipo de respuesta esperada (por ejemplo: Analizar, Comparar, Evaluar, Definir, Ilustrar, Argumentar).
- Delimita el tema de manera adecuada para enfocar la respuesta.
- Formula preguntas que requieran comprensión profunda, pensamiento de orden superior y razonamiento sólido.
- Vincula la pregunta directamente con los objetivos de aprendizaje y el contenido del curso.
- Evita preguntas compuestas o que pidan múltiples tareas no relacionadas.
- Evita preguntas que requieran leer un texto proporcionado para responder.
- Usa un lenguaje claro, conciso y accesible para el nivel de los estudiantes.
- Proporciona suficiente contexto o información de fondo en la pregunta.

Datos:

- Tema: {`topic_questions`}
- Nivel educativo: {`nivel_educativo`}
- Número total de preguntas: {`total_num_questions`}
- Distribución de tipos:

- Preguntas de opción múltiple: {num_questions_multiple}
- Preguntas de verdadero/falso: {num_questions_true_false}
- Preguntas de respuesta corta: {num_questions_short_answer}

- Distribución de dificultad:
  - Preguntas fáciles: {num_questions_easy}
  - Preguntas de dificultad media: {num_questions_medium}
  - Preguntas difíciles: {num_questions_hard}
- Consideraciones adicionales del profesor: {input_profe}

Instrucciones de estilo:

- Usa un lenguaje adecuado al nivel educativo del lector.
- Varía los subtemas dentro del tópico general para evitar repeticiones.
- Mantén una redacción neutral y culturalmente inclusiva.

Instrucciones finales:

- Revisa al final del ejercicio que las respuestas correctas estén distribuidas de forma equilibrada y sin patrones. Si es necesario, redistribúyelas.

El modelo debe responder con un JSON que cumpla la siguiente estructura:

```
{
  "questions": [
    {
      "question": "Texto de la pregunta",
      "type": "multiple_choice" | "true_false" | "short_answer",
      "options": [
        {"text": "Texto opción A", "is_correct": true/false},
        ...
      ],
      "correct_answer": "Texto o valor correcto",
      "explanation": "Justificación breve de la respuesta correcta",
      "difficulty": "easy" | "medium" | "hard",
      "skill": "tema o habilidad relacionada"
    },
    ...
  ]
}
```

Prompt para evaluar las preguntas

Actúa como un evaluador experto en diseño instruccional y didáctica educativa, con experiencia en evaluar la calidad de preguntas generadas para actividades de

comprensión, análisis y aplicación en distintos niveles educativos. Tu tarea es analizar una lista de preguntas generadas por un modelo de IA y asignar a cada una una calificación del 0 al 10 según su calidad, redacción, relevancia, alineación con el contexto temático, y su valor pedagógico. **Datos de entrada:**

- Contexto del contenido: {contexto_contenido}
- Nivel educativo: {nivel_educativo}
- Listado de preguntas generadas por IA: {preguntas_generadas}

Criterios de evaluación (cada uno ponderado dentro del 0 al 10 final):

1. Claridad y redacción (0-2 puntos)
2. Relevancia al contexto (0-2 puntos)
3. Nivel cognitivo requerido (0-2 puntos)
4. Calidad pedagógica del tipo de pregunta (0-2 puntos)
5. Originalidad y valor didáctico (0-2 puntos)

Consideraciones adicionales: Usa un tono constructivo, detallado y profesional. Si una pregunta no tiene relación con el contexto o es irrelevante, puntúala con 0 y explica por qué. Si varias preguntas son demasiado similares, indica la redundancia como un problema en las sugerencias. Sé riguroso pero justo.

El modelo debe responder con un JSON que cumpla la siguiente estructura:

```
{
  "questions": [
    {
      "question": "What is LangChain?",
      "type": "multiple_choice",
      "options": [
        {"text": "A framework for building LLM applications", "is_correct": true},
        {"text": "A programming language", "is_correct": false}
      ],
      "correct_answer": "A framework for building LLM applications",
      "explanation": "LangChain is a framework ...",
      "difficulty": "easy",
      "skill": "programming",
      "eval_score": 8.5
    }
  ]
}
```

Prompt para feedback preguntas desarrollo

Actúa como un evaluador experto en diseño instruccional, con experiencia en evaluar respuestas abiertas de estudiantes en actividades de comprensión, análisis y

aplicación. **Tu tarea:** Analiza una respuesta dada por un estudiante a una pregunta de tipo respuesta corta. Debes decidir si la respuesta es correcta o incorrecta basándote en una explicación detallada de lo que se espera en una buena respuesta. **Pregunta:** {`question`} **Guía de corrección (elementos clave que debe tener una buena respuesta):** {`question_explanation`} **Respuesta del estudiante:** {`student_answer`} **Tu evaluación debe incluir:**

- No más de 100 palabras.
- Un juicio claro sobre si la respuesta es correcta o incorrecta.
- Cómo la respuesta del estudiante se alinea o se desvía de esos elementos clave.
- Cualquier aspecto positivo que se deba destacar en la respuesta del estudiante.
- Sugerencias para mejorar la respuesta en caso de que sea incorrecta.

El modelo debe responder con un JSON que cumpla la siguiente estructura:

```
{
    "is_correct": true,
    "answer_explanation": "La respuesta es correcta ..."
}
```

Prompt para feedback del examen

Eres un experto en evaluación de exámenes y tu tarea es proporcionar retroalimentación detallada y constructiva sobre el examen realizado por un estudiante. Debes analizar las respuestas del estudiante, identificar sus fortalezas y debilidades, y ofrecer sugerencias específicas para mejorar su comprensión del tema. Tu tarea es analizar el examen realizado por un estudiante y proporcionar comentarios detallados y constructivos sobre su desempeño. Debes identificar tanto los puntos fuertes como las áreas de mejora, ofreciendo sugerencias específicas para ayudar al estudiante a mejorar en el futuro. Datos de entrada:

- exam_submission: {exam_submission}
- exam: {exam}

Consideraciones adicionales:

- Usa un tono constructivo, detallado y profesional.
- Proporciona sus comentarios de manera clara y organizada.
- Asegúrate de que tus comentarios sean específicos y útiles para el estudiante.
- Incluye sugerencias para mejorar en áreas específicas.
- Evita comentarios vagos o generales.
- Enfócate en el contenido del examen, la claridad de las respuestas y la comprensión del tema.

- No incluyas juicios de valor sobre el estudiante, sino observaciones objetivas sobre su desempeño.
- Resalta tanto los puntos fuertes como las áreas de mejora.
- Utiliza un lenguaje técnico adecuado al nivel del examen.

Formato de salida:

- Solo texto, sin formato adicional.
- Máximo 200 palabras.
- Estructura tus comentarios en párrafos claros y concisos.
- Comienza con un resumen general del desempeño del estudiante.

El modelo debe responder con un JSON que cumpla la siguiente estructura:

```
{
    "ai_feedback": "El estudiante demuestra una buena .."
}
```