



**UNIVERSIDAD
DE GRANADA**

**TRABAJO FIN DE GRADO
INGENIERÍA EN INFORMÁTICA**

**Desarrollo de un asistente para el apoyo a
los procesos de enseñanza/aprendizaje de
la programación**

Autor

Óscar Picado Cariño

Directores

David Griol Barres

Zoraida Callejas Carrión



**ESCUELA TÉCNICA SUPERIOR DE INGENIERÍAS INFORMÁTICA Y DE
TELECOMUNICACIÓN**

Granada, 16 Junio de 2025



Desarrollo de un asistente para el apoyo a los procesos de enseñanza/aprendizaje de la programación



<https://github.com/oscarpicadocarino/tfg-app>

Autor

Óscar Picado Cariño

Directores

David Griol Barres

Zoraida Callejas Carrión

Desarrollo de un asistente para el apoyo a los procesos de enseñanza/aprendizaje de la programación

Óscar Picado Cariño

Palabras clave: Modelos de lenguaje, educación, aprendizaje de la programación, inteligencia artificial, prompting, asistente conversacional, LLMs, OpenAI.

Resumen

Este Trabajo Fin de Grado presenta el diseño y desarrollo de un asistente educativo inteligente basado en modelos de lenguaje de gran tamaño (LLMs), enfocado en apoyar el aprendizaje de la programación en el ámbito universitario. La propuesta parte de la necesidad creciente de herramientas que faciliten la comprensión de conceptos complejos y refuercen el aprendizaje autónomo, especialmente en asignaturas técnicas con alta tasa de dificultad y abandono.

El objetivo principal de este proyecto es analizar, comparar e integrar diferentes modelos de lenguaje y técnicas de prompting para evaluar su utilidad pedagógica, su eficiencia en términos de coste computacional y su capacidad para adaptarse a distintos perfiles de usuario. Para ello, se han probado diversos modelos mediante la plataforma Ollama, incluyendo variantes como LLaMA 3, Deepseek, Codestral y otros especializados en generación de código. A través de múltiples experimentos se ha evaluado la calidad, relevancia y adecuación didáctica de sus respuestas en contextos reales de enseñanza-aprendizaje.

El resultado es una aplicación web interactiva que ofrece a alumnado y profesorado un entorno personalizado para el refuerzo de competencias clave en asignaturas como Fundamentos de Ingeniería del Software, Programación y Diseño Orientado a Objetos, y Sistemas Concurrentes y Distribuidos. La herramienta permite crear, editar y resolver actividades interactivas, generar explicaciones adaptadas al nivel del estudiante, responder dudas frecuentes y proponer ejercicios orientados a errores habituales en la programación.

Además, se ha implementado un sistema completo de gestión de usuarios con distintos roles (alumnado, profesorado, administración) y una base de datos que registra las interacciones para su posterior análisis. Esta funcionalidad no solo facilita la evaluación continua del progreso del alumnado, sino que también permite adaptar dinámicamente las actividades generadas a partir de las competencias y resultados de aprendizaje definidos en las guías docentes, aumentando así la eficacia pedagógica del sistema.

Development of an assistant to support the teaching/learning processes of the programming.

Óscar Picado Cariño

Keywords: Language models, education, programming learning, artificial intelligence, prompting, conversational assistant, LLMs, OpenAI.

Abstract

This Bachelor's Degree Final Project (TFG) presents the design and development of an intelligent educational assistant based on large language models (LLMs), aimed at supporting programming learning in the university context. The proposal addresses the growing need for tools that help students understand complex concepts and promote autonomous learning, especially in technical subjects with high levels of difficulty and dropout rates.

The main objective of this project is to analyze, compare, and integrate different language models and prompting techniques in order to evaluate their pedagogical effectiveness, computational cost, and adaptability to different user profiles. Several models were tested using the Ollama platform, including variants such as LLaMA 3, Deepseek, Codestral, and others specialized in code generation. Through extensive experimentation, the quality, relevance, and educational adequacy of the models' responses were assessed in real learning contexts.

The result is an interactive web application that provides both students and instructors with a personalized environment to reinforce key competencies in subjects such as Fundamentals of Software Engineering, Programming and Object-Oriented Design, and Concurrent and Distributed Systems. The assistant allows users to create, edit, and solve interactive activities, generate explanations tailored to the student's level, address frequently asked questions, and propose exercises focused on common programming mistakes.

In addition, a complete user management system has been implemented, supporting multiple roles (students, instructors, administrators), along with a database that records interactions for further analysis. This functionality not only enables continuous assessment of student progress, but also allows the dynamic adaptation of the generated activities based on the learning outcomes and competencies defined in the course syllabi, thereby increasing the pedagogical effectiveness of the system.

Yo, **Óscar Picado Cariño**, alumno de la titulación Grado en Ingeniería Informática de la **Escuela Técnica Superior de Ingenierías Informática y de Telecomunicación de la Universidad de Granada**, con DNI 75930961A, autorizo la ubicación de la siguiente copia de mi Trabajo Fin de Grado en la biblioteca del centro para que pueda ser consultada por las personas que lo deseen.

Fdo: Óscar Picado Cariño

Granada a 16 de Junio de 2025 .

D. **David Griol Barres**, Profesor del Área de Lenguajes y Sistemas Informáticos del Departamento de Lenguajes y Sistemas Informáticos de la Universidad de Granada.

D. **Zoraida Callejas Carrión**, Profesora del Área de Lenguajes y Sistemas Informáticos del Departamento de Lenguajes y Sistemas Informáticos de la Universidad de Granada.

Informan:

Que el presente trabajo, titulado *Desarrollo de un asistente para el apoyo a los procesos de enseñanza/aprendizaje de la programación*, ha sido realizado bajo su supervisión por **Óscar Picado Cariño**, y autorizamos la defensa de dicho trabajo ante el tribunal que corresponda.

Y para que conste, expiden y firman el presente informe en Granada a 16 de junio de 2025.

Los directores:

David Griol Barres

Zoraida Callejas Carrión

Agradecimientos

Quiero expresar mi agradecimiento a todas las personas que me han acompañado y apoyado durante esta etapa tan importante de mi vida, tanto en lo académico como en lo personal.

A mis compañeros de clase, por compartir los retos del camino y por los momentos de aprendizaje y amistad que hemos vivido juntos. Gracias por estar ahí en los días buenos y en los no tan buenos.

A mi familia, por su apoyo incondicional, por creer en mí incluso cuando yo dudaba, y por darme siempre la fuerza para seguir adelante.

Y a mis amigos, por saber escuchar y por ayudarme a mantener el equilibrio entre el estudio y la vida. Gracias por estar siempre ahí.

A todos vosotros, muchas gracias.

Índice general

1. Introducción	17
1.1. Motivación	18
1.2. Objetivos	19
1.3. Presupuesto	20
1.4. Planificación del proyecto	21
2. Estado del Arte	23
2.1. ¿Que es un LLM?	23
2.2. Aprendizaje profundo y NLP	24
2.2.1. Aprendizaje profundo	24
2.3. Prompting	25
2.4. Modelos generativos en educación	25
2.5. Propuesta de valor y diferenciación de la aplicación	26
2.5.1. Análisis DAFO	27
3. Marco Tecnológico y Recursos Computacionales	29
3.1. GitHub	29
3.2. Visual Studio Code	30
3.3. Docker	30
3.4. HTML y CSS	31
3.5. JavaScript	31

3.6. Bootstrap	31
3.7. PHP	31
3.8. MySQL (con phpMyAdmin)	32
3.9. Figma	32
3.10. draw.io	32
3.11. OpenAI y GPT-3.5 Turbo	33
3.12. Hugging Face	34
3.13. Ollama	34
4. Estudio de modelos LLM y técnicas de prompting	35
4.1. Modelos evaluados	35
4.1.1. Pruebas en el servidor de la universidad	37
4.1.2. Exploración de modelos en la nube y análisis de costes	38
4.1.3. Decisión de utilizar la API de OpenAI	39
4.2. Resultados de la comparación	40
4.3. Técnicas de prompting empleadas	41
4.4. Principales conclusiones	42
5. Desarrollo de la Aplicación	43
5.1. Historias de Usuario	44
5.2. Arquitectura del sistema	49
5.3. Diseño de la base de datos	51
5.4. Diseño visual de la interfaz	52
5.4.1. Pantalla de Inicio de Sesión	53
5.4.2. Pantalla de Inicio del Administrador	53
5.4.3. Pantalla de Gestión de Usuarios	54
5.4.4. Pantalla de Gestión de Clases	55
5.4.5. Pantalla de Asignar Alumnos a una Clase	55

5.4.6. Pantalla de Inicio para los Profesores	56
5.4.7. Pantalla de Clase de los Profesores	57
5.4.8. Pantalla de Generar Actividad	57
5.4.9. Pantalla de Gestión de Actividades	58
5.4.10. Pantalla de Errores Comunes	59
5.4.11. Pantalla de Inicio para los Alumnos	59
5.4.12. Pantalla de Inicio de Clase de los Alumnos	60
5.4.13. Pantalla de Chatbot Alumno	60
5.4.14. Pantalla de Clase de los Alumnos	61
5.4.15. Pantalla de Chatbot para Dudas sobre Actividades de los Alumnos	61
5.5. Estructura del proyecto y distribución de ficheros	62
5.6. Implementación del Chatbot	63
5.6.1. Modelo base y justificación de uso	63
5.6.2. Diseño funcional del chatbot	63
5.6.3. Uso de prompting interno y control del contexto	64
5.6.4. Lógica e integración técnica	65
5.6.5. Limitaciones técnicas	66
5.7. Ficheros de configuración y automatización	66
5.7.1. Estructura del archivo docker-compose.yml	66
5.7.2. Automatización del proceso	69
5.7.3. Conclusión	70
5.8. Pruebas realizadas	70
6. Conclusiones y Trabajos Futuros	73
6.1. Conclusiones	73
6.2. Trabajos Futuros	74
A. Manual de Usuario	77

Capítulo 1

Introducción

El aprendizaje de la programación es uno de los pilares fundamentales en la formación de los estudiantes de ingeniería informática. También es uno de los mayores retos tanto para el alumnado como para el profesorado, debido a la complejidad conceptual, la necesidad de pensamiento lógico estructurado y la abstracción de ideas. A lo largo de los últimos años, diversos enfoques pedagógicos han intentado facilitar esta enseñanza, desde metodologías activas como el aprendizaje basado en problemas, hasta herramientas de visualización o plataformas interactivas. Hoy en día, con el crecimiento de la inteligencia artificial generativa y los grandes modelos de lenguaje (Large Language Models o LLMs), se están abriendo nuevas oportunidades para apoyar el proceso de enseñanza y aprendizaje de la programación.

Los LLMs, como ChatGPT, Codellama o DeepSeek-Coder, se entrenan sobre enormes cantidades de texto y código fuente para aprender patrones del lenguaje natural y de la programación. Esta capacidad les permite generar explicaciones, sugerir soluciones, corregir errores, e incluso proponer actividades educativas. Esto los convierte en herramientas para desarrollar asistentes inteligentes capaces de adaptarse al nivel del estudiante, proporcionar feedback inmediato y fomentar el aprendizaje autónomo.

El presente Trabajo de Fin de Grado se centra en el desarrollo de un asistente educativo, integrado en una aplicación web, cuyo objetivo es facilitar el aprendizaje de la programación mediante el uso de LLMs. No se pretende desarrollar un modelo propio desde cero, sino realizar un estudio comparativo de distintos modelos de lenguaje y técnicas de prompting, con el fin de determinar cuál se adapta mejor a los objetivos docentes del contexto universitario.

El asistente se integrará en una aplicación más amplia que servirá como plataforma de apoyo tanto para alumnado como profesorado. Entre sus funcionalidades se incluirán: actividades formativas, materiales didácticos

y apoyo en el aprendizaje autónomo en asignaturas relacionadas con programación. Todo ello en base a los contenidos, competencias y resultados de aprendizaje definidos en las guías docentes de asignaturas de programación, como Fundamentos de Ingeniería del Software, Programación y Diseño Orientado a Objetos, y Sistemas Concurrentes y Distribuidos.

Este enfoque coincide con estudios recientes que analizan cómo la IA generativa puede contribuir a la educación. Por ejemplo, el proyecto TeachYou [11] utiliza LLMs como agentes enseñables, donde el estudiante toma el rol de “profesor” del modelo, promoviendo el aprendizaje mediante la enseñanza. Del mismo modo, existen estudios que muestran cómo adaptar modelos de lenguaje para que adopten comportamientos pedagógicos mediante fine-tuning supervisado [15]. A nivel nacional, diversas contribuciones presentadas en las Jornadas sobre la Enseñanza Universitaria de la Informática (JENUI) demuestran el interés creciente por estas tecnologías, como el uso de bots conversacionales en Telegram [8] o las implicaciones de ChatGPT en la docencia universitaria [10]

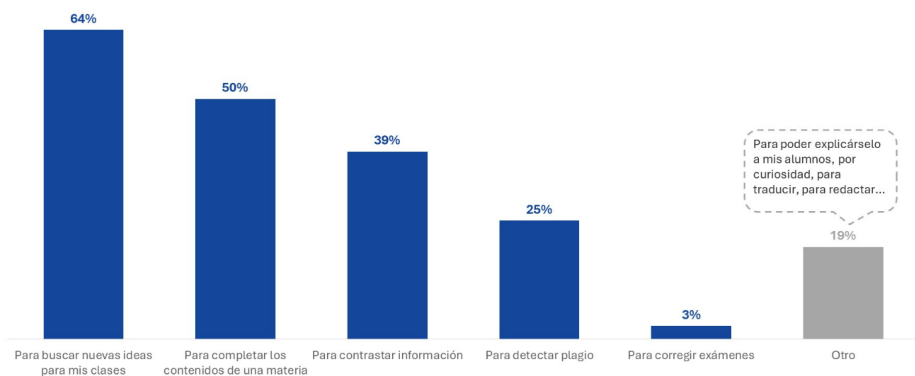
En este trabajo se propone analizar cómo se puede diseñar y optimizar la interacción con LLMs mediante técnicas de prompting, explorar sus capacidades y limitaciones en el entorno educativo, e integrarlos eficazmente en una plataforma que contribuya al aprendizaje de la programación.

1.1. Motivación

En los últimos años el uso de herramientas basadas en inteligencia artificial generativa se ha generalizado entre estudiantes, especialmente en el ámbito de la programación. Si bien estas tecnologías ofrecen un apoyo valioso, muchas veces son empleadas como una solución automática que proporciona respuestas sin fomentar una comprensión profunda. Esta tendencia resulta especialmente preocupante en etapas formativas como el inicio de una carrera en Ingeniería Informática, donde el pensamiento crítico, el razonamiento lógico y la comprensión de los fundamentos son esenciales.

Utilizar la IA sin un conocimiento previo adecuado puede llevar a una dependencia tecnológica que impide desarrollar habilidades clave. Sin embargo, cuando se cuenta con una base sólida, la inteligencia artificial generativa puede convertirse en una gran ayuda para afianzar conocimientos, reducir el tiempo de trabajo...

La utilización de este tipo de Inteligencias Artificiales, no solo ha crecido entre el alumnado, sino también entre los docentes. Una reciente encuesta realizada en 2024, afirma que el 73 % de los profesores han utilizado IA en alguna ocasión, para preparar las clases (64 %) y para complementar los contenidos (50 %). (ver Figura 1.1).



[4]

Figura 1.1: Finalidad del uso de la inteligencia artificial del profesorado

Este trabajo nace de la necesidad de analizar cómo la inteligencia artificial puede ser utilizada de forma responsable en la educación, concretamente en la programación, para potenciar el aprendizaje en lugar de reemplazarlo. Ante esta situación, surge la oportunidad de diseñar soluciones que no solo utilicen la IA como una herramienta de consulta, que también, sino que además se utilice como un recurso pedagógico adaptativo que complemente y enriquezca la labor educativa.

Es por ello que en el proyecto se desarrolla una aplicación educativa que integra un chatbot con funciones didácticas. Este chatbot no ha sido entrenado desde cero, sino que utiliza un modelo de lenguaje preentrenado al que se le proporciona contexto específico de forma interna como errores comunes detectados por docentes universitarios o información extraída de las guías docentes, para así personalizar sus respuestas. Esta estrategia, similar al uso avanzado de prompts, permite que el sistema genere actividades adaptadas al nivel del alumnado y ofrezca apoyo tanto a estudiantes como a profesores.

Frente a herramientas existentes como PRADO, centradas en la gestión de contenidos y tareas, la aplicación desarrollada apuesta por una interacción pedagógica más dinámica y personalizada, que promueve una comprensión activa de los conceptos, fomenta la autonomía del alumno y mejora el acompañamiento docente mediante el uso de IA responsable.

1.2. Objetivos

El objetivo general de este TFG es desarrollar un estudio comparativo entre diferentes modelos de lenguaje (LLMs) y técnicas de prompting para evaluar su efectividad en la generación de actividades educativas interactivas

en el ámbito de la programación, y su integración en una aplicación web que sirva como asistente educativo para estudiantes de ingeniería informática.

A continuación se detallan objetivos más específicos:

- **Explorar y analizar diversos LLMs:** Investigar y probar diferentes modelos de lenguaje como Llama 3, Codellama, Mistral, entre otros, evaluando su capacidad para generar respuestas precisas, contextualizadas y adecuadas para la enseñanza de la programación.
- **Evaluar las técnicas de prompting:** Estudiar diversas estrategias de prompting que permitan optimizar la interacción con los modelos, con el objetivo de generar actividades educativas relacionadas con la programación que sean útiles y comprensibles para los estudiantes.
- **Desarrollar un sistema de actividades interactivas:** Diseñar un sistema basado en IA que genere actividades educativas dinámicas y personalizadas, con niveles de complejidad creciente, que ayuden a los estudiantes a aprender conceptos fundamentales de programación.
- **Integrar el asistente en una aplicación web:** Implementar un chatbot educativo dentro de una aplicación web, proporcionando una interfaz fácil de usar que permita a la comunidad educativa (estudiantes y profesores) interactuar con el asistente.

1.3. Presupuesto

Para calcular el presupuesto del Trabajo de Fin de Grado se ha tenido en cuenta tanto el tiempo de trabajo dedicado como los recursos utilizados durante el desarrollo del mismo.

El proyecto ha sido desarrollado durante un periodo de cuatro meses, con una dedicación media de 20 horas semanales. Esto supone un total de 320 horas de trabajo. Considerando un coste de 12,67 €/hora, basado en el salario medio de un ingeniero informático junior en España [16], el coste asociado al tiempo es de 4.054,40 €.

A nivel técnico, he utilizado mi ordenador personal adquirido en 2021 por 1.000 €. He hecho una estimación de una amortización del 10 % de su valor para este proyecto, lo que representa un coste de 100 €. Además, he contado con acceso al servidor del departamento de la universidad, sin coste adicional.

Concepto	Detalle	Coste
Horas de trabajo	320 horas x 12.67€/horas	4054,40 €
Amortización del equipo	Ordenador personal	100,00 €
Total del proyecto		4174,40 €

1.4. Planificación del proyecto

El desarrollo del presente Trabajo de Fin de Grado se organizó en varias fases con el fin de cumplir con la fecha límite de entrega. Se previó una duración aproximada de cuatro meses. A continuación, se detallan como distribuí las etapas del proyecto:

- **Fase 1: Diseño e implementación de la aplicación web (1-1.5 meses)**

Durante esta primera etapa trabajaré en:

- Análisis de requisitos y definición de funcionalidades de la aplicación.
- Creación de las historias de usuario (HU).
- Diseño de la base de datos y elaboración del diagrama entidad-relación.
- Diseño de la infraestructura técnica necesaria.
- Selección de la paleta de colores y diseño visual.
- Desarrollo e implementación del prototipo funcional de la aplicación web.

- **Fase 2: Estudio de técnicas de prompting y modelos LLM (1-1.5 meses)**

Una vez establecida la base técnica de la aplicación, iniciaré un proceso de estudio y análisis que incluya:

- Revisión de la literatura y documentación técnica sobre prompting y LLMs.
- Pruebas con distintos modelos (Llama 3, Codellama, Mistral, etc.).
- Evaluación de las respuestas generadas por cada modelo ante distintos tipos de prompts educativos.
- Análisis comparativo de resultados para determinar qué modelo y estrategia resultan más efectivos en el contexto del aprendizaje de programación.

■ **Fase 3: Ajustes, pruebas finales y redacción de la memoria (últimas semanas)**

Esta última etapa la reservo para:

- Corrección de errores y mejora de funcionalidades detectadas durante las pruebas.
- Ajustes en la integración del asistente y refinamiento de la interfaz.
- Redacción y revisión del documento de memoria del TFG.
- Preparación de la presentación final del proyecto.

A continuación se presenta el diagrama de Gantt correspondiente a la planificación del proyecto. Este cronograma permite visualizar la distribución temporal de las diferentes fases y actividades, facilitando el seguimiento y control del avance. (ver Figura 1.2).

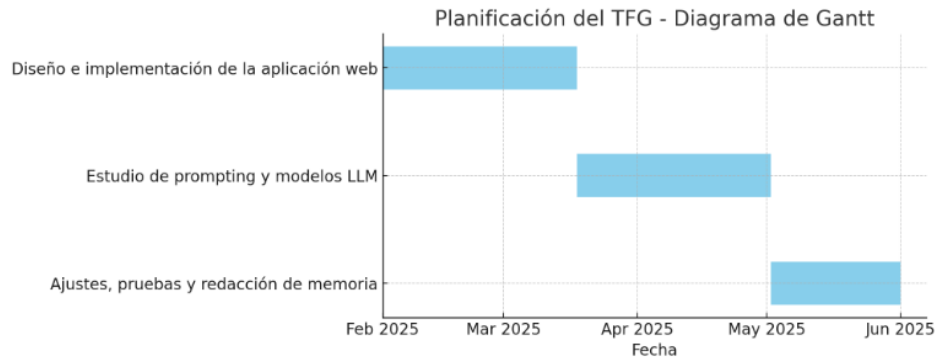


Figura 1.2: Planificación temporal del proyecto

Capítulo 2

Estado del Arte

El objetivo principal de este capítulo es presentar un análisis detallado del estado del arte en el uso de modelos de lenguaje de gran tamaño (LLMs) aplicados al ámbito educativo, con especial énfasis en la generación automática de actividades formativas para asignaturas relacionadas con la programación.

Se introducirán los conceptos clave necesarios para comprender el funcionamiento de los LLMs, incluyendo los fundamentos del aprendizaje profundo, el procesamiento del lenguaje natural (NLP) y las arquitecturas más relevantes como los transformadores. Además, se incluirán temas fundamentales relacionados con la generación automática de contenido, la personalización del aprendizaje y los retos actuales en cuanto a evaluación, sesgos y adaptabilidad de estos modelos al contexto educativo.

2.1. ¿Que es un LLM?

Los modelos de lenguaje de gran tamaño, comúnmente conocidos como LLM (por sus siglas en inglés, Large Language Models), son modelos de aprendizaje profundo diseñados para procesar, comprender y generar lenguaje natural [2, 3, 17]. Se caracterizan por contar con miles de millones de parámetros y por haber sido entrenados con volúmenes masivos de datos textuales procedentes de diversas fuentes, como páginas web, libros, artículos científicos o bases de datos abiertas como Common Crawl ¹ o Wikipedia ².

La arquitectura base de la mayoría de los LLM actuales está fundamentada en los transformadores (transformers), una estructura de redes neu-

¹<https://commoncrawl.org/>

²<https://es.wikipedia.org>

ronales introducida en [18]. Esta arquitectura se compone de dos bloques principales: un codificador (encoder) y un decodificador (decoder), ambos con mecanismos de atención que permiten al modelo identificar relaciones entre palabras, frases y conceptos en una secuencia de texto. A diferencia de las redes neuronales recurrentes (RNN), los transformadores procesan las secuencias de texto de forma paralela, lo que permite una mayor eficiencia computacional y facilita el entrenamiento a gran escala utilizando GPUs [7].

El entrenamiento de los LLM suele realizarse mediante técnicas de aprendizaje no supervisado o auto-supervisado, en las que el modelo aprende a predecir la siguiente palabra en una secuencia o a rellenar partes omitidas del texto. A través de este proceso, los modelos adquieren una comprensión sintáctica y semántica del lenguaje, así como conocimientos generales sobre el mundo. Además, algunos modelos de razonamiento, como DeepSeek, incorporan técnicas de aprendizaje reforzado para mejorar su capacidad de razonamiento y toma de decisiones [21].

Gracias a su escalabilidad y capacidad para capturar patrones complejos del lenguaje, los LLM se han convertido en herramientas fundamentales para tareas de procesamiento del lenguaje natural (NLP), incluyendo la traducción automática, la respuesta a preguntas, la generación de textos y, en el contexto educativo, la creación de contenidos personalizados.

2.2. Aprendizaje profundo y NLP

2.2.1. Aprendizaje profundo

El aprendizaje profundo es una rama del aprendizaje automático que utiliza redes neuronales con múltiples capas para modelar relaciones complejas en grandes volúmenes de datos. Estas redes aprenden representaciones jerárquicas de los datos, lo que les permite abordar tareas complejas, como el reconocimiento de imágenes o el procesamiento del lenguaje natural (NLP).

Una de las áreas donde el aprendizaje profundo ha tenido un impacto particularmente significativo es en el Procesamiento del Lenguaje Natural. Este campo abarca un conjunto de técnicas y enfoques que permiten a las máquinas comprender, interpretar y generar lenguaje como el de los seres humanos [5].

En este contexto, la arquitectura *transformer* ha transformado el campo del NLP al permitir el procesamiento paralelo de secuencias, a diferencia de las redes neuronales recurrentes (RNN), que procesan los datos de manera secuencial. Los transformadores utilizan un mecanismo de atención para identificar las relaciones más relevantes dentro de una secuencia, lo que mejora la captura del contexto a largo plazo [18].

2.3. Prompting

El prompting es una técnica utilizada para interactuar con modelos de lenguaje mediante la formulación de instrucciones o ejemplos que orientan la generación del texto. Su relevancia ha crecido exponencialmente con la popularización de los modelos, ya que permite adaptar el comportamiento a diferentes tareas sin necesidad de realizar un ajuste fino (fine-tuning) en sus parámetros [3].

A lo largo del tiempo se han desarrollado diversas estrategias de prompting:

- **Zero-shot prompting:** se formula una instrucción directa sin ejemplos previos, confiando en el conocimiento general del modelo.
- **One-shot prompting:** se incluye un único ejemplo junto con la instrucción para guiar la generación.
- **Few-shot prompting:** se proporcionan varios ejemplos para mostrar el comportamiento deseado, lo que mejora la comprensión del modelo sobre la tarea [3].
- **Chain-of-thought prompting:** esta técnica añade razonamientos intermedios o pasos de pensamiento para tareas que requieren lógica o razonamiento complejo [20].
- **Instruction prompting:** se basa en modelos previamente ajustados para seguir instrucciones humanas, como InstructGPT o LLaMA 3, lo que mejora la alineación entre los resultados del modelo y las expectativas del usuario [13].

En el presente Trabajo de Fin de Grado, el prompting se emplea para generar automáticamente actividades educativas a partir de conceptos clave extraídos de las guías docentes de varias asignaturas del grado en Ingeniería Informática. El objetivo es evaluar cómo diferentes modelos de lenguaje (LLaMA 3, Codestral, Deepseek-Coder ...), responden a distintas formulaciones de prompts, y cuál de ellos se adapta mejor a la generación de materiales pedagógicos relacionados con programación.

2.4. Modelos generativos en educación

El uso de modelos generativos basados en redes neuronales profundas, especialmente los Modelos de Lenguaje de Gran Tamaño (LLMs), ha transformado la manera en la que se diseña y gestiona el aprendizaje en entornos

educativos. Diversos trabajos recientes han mostrado cómo estas tecnologías pueden apoyar tanto al alumnado como al profesorado mediante la generación automática de contenidos, la personalización del aprendizaje y la retroalimentación inmediata.

Por ejemplo, Kasneci et al. [12] analizan el papel de ChatGPT en la educación superior, destacando su potencial para mejorar la escritura académica, fomentar el aprendizaje autónomo y apoyar el pensamiento crítico, aunque advierten sobre el riesgo de una dependencia tecnológica si no se utiliza de forma adecuada. Del mismo modo, Qadir et al. [14] presentan una revisión exhaustiva sobre el uso de LLMs en educación, subrayando la importancia de diseñar entornos pedagógicos controlados y con validación humana para garantizar su efectividad.

Entre las propuestas más innovadoras se encuentra el enfoque *Teach You* de Xu et al. [22], que utiliza un LLM como agente enseñable, promoviendo el aprendizaje mediante la inversión del rol tradicional estudiante-profesor. Otros trabajos como CodeReviewer [19] exploran cómo utilizar prompting avanzado para generar revisiones automáticas de código con explicaciones pedagógicas, facilitando así la comprensión de errores frecuentes en programación.

A nivel nacional, además de las contribuciones presentadas en las Jornadas de Enseñanza Universitaria de la Informática (JENUI) [1], existen iniciativas como el uso de bots docentes en plataformas como Telegram o estudios que evalúan el impacto de ChatGPT en la preparación de clases y la evaluación automatizada [6].

Estos trabajos reflejan un creciente interés por la integración de LLMs en educación, pero también destacan desafíos clave como la alineación curricular, la supervisión pedagógica del contenido generado o la adaptabilidad al nivel del estudiante. Estas cuestiones son abordadas de forma complementaria en el presente TFG, cuya propuesta concreta se detalla en el siguiente apartado 2.5.

2.5. Propuesta de valor y diferenciación de la aplicación

En el presente trabajo se propone el desarrollo de una aplicación web que integra un modelo de lenguaje de última generación para automatizar la generación de actividades educativas vinculadas con asignaturas del Grado en Ingeniería Informática, especialmente en aquellas relacionadas con la programación.

A diferencia de soluciones genéricas que ofrecen generación de conteni-

do sin supervisión pedagógica, esta aplicación se distingue por incorporar elementos clave que refuerzan su utilidad académica y su adaptación al contexto docente:

- **Alineación curricular:** las actividades se generan a partir del análisis de las guías docentes oficiales, lo que garantiza que cada propuesta esté directamente vinculada con los resultados de aprendizaje y competencias definidos en el plan de estudios. Además, el sistema permite incorporar patrones de errores frecuentes identificados por el profesorado, favoreciendo una enseñanza más significativa basada en la corrección guiada y el razonamiento.
- **Interfaz para docentes:** se ofrece una plataforma interactiva que permite al profesorado editar, validar, clasificar y adaptar las actividades generadas. Esto asegura un control humano sobre la calidad pedagógica del contenido, promoviendo un enfoque colaborativo entre la inteligencia artificial y el criterio docente. También se contemplan funcionalidades para reutilizar actividades y compartirlas entre docentes.
- **Evaluación de modelos:** se realiza comparación sistemática entre distintos modelos de lenguaje (como LLaMA 3, gpt, DeepSeek, entre otros), evaluando su rendimiento según criterios como precisión técnica y adecuación al nivel del estudiante. Esta evaluación continua permite seleccionar el modelo más apropiado para el sistema.

Esta propuesta se diferencia claramente de otras herramientas existentes, como los chatbots generalistas (por ejemplo, ChatGPT), por su enfoque explícitamente pedagógico, la intervención activa del profesorado y su integración con la estructura académica oficial. Asimismo, se plantea como una herramienta escalable, capaz de adaptarse a distintos niveles educativos y áreas de conocimiento, abriendo la puerta a futuras extensiones en otros grados o programas formativos.

2.5.1. Análisis DAFO

A continuación se presenta un análisis DAFO (Debilidades, Amenazas, Fortalezas y Oportunidades) sobre el uso de modelos generativos en el ámbito educativo. Este análisis permite identificar tanto los aspectos internos (fortalezas y debilidades) como los externos (oportunidades y amenazas) que pueden influir en la integración de estas tecnologías en procesos de enseñanza-aprendizaje. Entre los factores considerados se encuentran la accesibilidad, la ética, el potencial de aprendizaje y los riesgos asociados a su

uso. Este enfoque facilita una visión global que contribuye a una implementación más crítica y reflexiva de estas herramientas en contextos educativos (ver Figura 2.1).

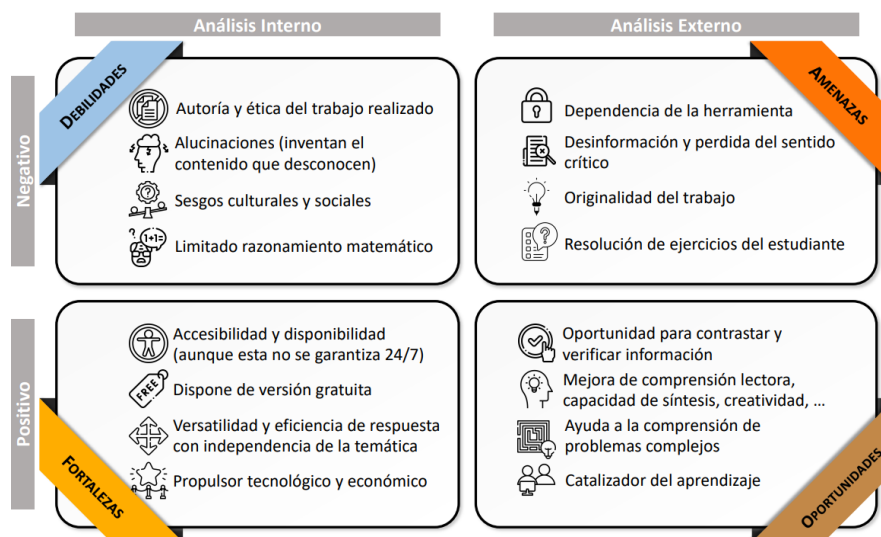


Figura 2.1: Análisis DAFO para la definición de las actividades docentes del profesorado utilizando ChatGPT

[9]

Capítulo 3

Marco Tecnológico y Recursos Computacionales

En este capítulo se describe y analiza el conjunto de tecnologías, herramientas y plataformas utilizadas durante el desarrollo del proyecto. Se ofrece una visión general de los componentes técnicos que han servido como base para la implementación de la aplicación propuesta, incluyendo tanto los modelos de lenguaje como los entornos de desarrollo y servicios asociados. Este análisis permite comprender las decisiones tecnológicas adoptadas, su justificación y la forma en que cada una de ellas contribuye al cumplimiento de los objetivos educativos y funcionales del sistema.

3.1. GitHub

GitHub¹ es una plataforma de alojamiento y gestión de proyectos de software que utiliza un sistema de control de versiones. Proporciona herramientas para la colaboración, seguimiento de problemas y tareas, y facilita la integración continua. En GitHub se pueden alojar y compartir repositorios de código fuente, siendo un repositorio un lugar donde se almacena y organiza el código de un proyecto.

Para el desarrollo de este TFG, se ha creado un repositorio al que se puede acceder a través de <https://github.com/oscarpicadocarino/tfg-app>. Además, también se ha generado un código QR, (ver Figura 3.1), como vía de acceso al proyecto.

¹<https://github.com/>



Figura 3.1: Código QR para acceso al proyecto en GitHub

3.2. Visual Studio Code

Visual Studio Code (VS Code)² es un editor de código fuente desarrollado por Microsoft, altamente utilizado en el desarrollo de software debido a su versatilidad, velocidad y extensibilidad. VS Code es un editor ligero que ofrece soporte para múltiples lenguajes de programación. A través de una amplia variedad de extensiones, es posible personalizar el entorno de desarrollo, agregar herramientas de depuración, control de versiones y compatibilidad con contenedores como Docker.

Para el desarrollo de este TFG, se utiliza VS Code como principal entorno de desarrollo debido a su integración con GitHub, su compatibilidad con lenguajes de programación relevantes para el proyecto, y su capacidad de personalización a través de extensiones, como Python, Flask y Docker. Además, su interfaz es intuitiva y permite una gestión eficiente del código.

3.3. Docker

Docker³ es una plataforma de contenedores que permite desarrollar, empaquetar y ejecutar aplicaciones de manera aislada y consistente en cualquier entorno. Docker utiliza contenedores, que son entornos ligeros y autónomos que incluyen todo lo necesario para ejecutar una aplicación, como el código, las bibliotecas y las dependencias. Esto asegura que una aplicación se ejecute de la misma manera en diferentes sistemas operativos y plataformas, evitando los típicos problemas de incompatibilidad entre entornos de desarrollo y producción.

En el marco de este TFG, Docker se utiliza para crear contenedores que facilitan la implementación y ejecución del chatbot educativo, asegurando que todos los componentes del sistema (como el backend, la base de datos y las dependencias del modelo de lenguaje) se ejecuten de forma consistente en distintos entornos. Esta solución también facilita el despliegue y la escalabilidad de la aplicación en servidores, contribuyendo a una gestión más eficiente de los recursos computacionales.

²<https://code.visualstudio.com/>

³<https://www.docker.com/>

3.4. HTML y CSS

HTML (HyperText Markup Language) es el lenguaje estándar para la creación de páginas web. Se utiliza para estructurar el contenido de un sitio web, permitiendo definir elementos como encabezados, párrafos, imágenes, enlaces, formularios, entre otros. HTML establece la base sobre la que se construye la apariencia y el comportamiento de una página web.

CSS (Cascading Style Sheets) es un lenguaje utilizado para definir la presentación visual de los elementos HTML. Permite controlar el diseño, los colores, las fuentes y el espacio entre los elementos, ofreciendo una gran flexibilidad para crear sitios web atractivos y responsivos.

En el desarrollo de este TFG, CSS se utiliza para definir la apariencia y disposición de los elementos en la interfaz web de la aplicación.

3.5. JavaScript

JavaScript es un lenguaje de programación orientado a eventos que se ejecuta en el navegador del usuario. Se utiliza para dotar de interactividad a las páginas web, permitiendo realizar acciones dinámicas sin necesidad de recargar la página.

En el desarrollo de este TFG, JavaScript se ha empleado para gestionar la interacción en tiempo real entre el usuario y el chatbot educativo. A través de JavaScript se manejan eventos como el envío de preguntas, la recepción de respuestas mediante WebSockets, y la actualización dinámica de la interfaz sin interrupciones, mejorando así la experiencia de uso.

3.6. Bootstrap

Bootstrap⁴ es un framework de código abierto basado en HTML, CSS y JavaScript, diseñado para facilitar el desarrollo de interfaces web responsivas y adaptables. Bootstrap proporciona una serie de componentes predefinidos, como botones, formularios, modales y barras de navegación, que permiten crear aplicaciones web con un diseño atractivo y funcional de manera rápida.

Para este TFG, Bootstrap se utiliza para garantizar que la interfaz de usuario del chatbot educativo sea accesible desde diferentes dispositivos, como computadoras de escritorio y dispositivos móviles. También para añadir diferentes íconos haciéndola así accesible.

3.7. PHP

PHP⁵ es un lenguaje de programación del lado del servidor utilizado para desarrollar aplicaciones web dinámicas. Es especialmente adecuado para interactuar

⁴<https://getbootstrap.com/>

⁵<https://www.php.net/>

con bases de datos y generar contenido web dinámico. PHP es ampliamente utilizado en el desarrollo de aplicaciones web debido a su facilidad de integración con tecnologías como MySQL.

En este TFG, PHP se utiliza para crear la lógica del backend del chatbot educativo y de la aplicación, gestionando las solicitudes del usuario y las interacciones con la base de datos.

3.8. MySQL (con phpMyAdmin)

MySQL⁶ es un sistema de gestión de bases de datos relacional (RDBMS) que se utiliza para almacenar y gestionar datos de manera estructurada. Se basa en el lenguaje SQL (Structured Query Language) para crear, leer, actualizar y eliminar registros en bases de datos. MySQL es ampliamente utilizado en aplicaciones web debido a su fiabilidad, escalabilidad y facilidad de uso.

phpMyAdmin⁷ es una herramienta de administración de bases de datos MySQL basada en web. Permite a los usuarios gestionar bases de datos, ejecutar consultas SQL y realizar tareas administrativas como la creación de tablas y la importación/exportación de datos, todo a través de una interfaz gráfica.

En este TFG, phpMyAdmin se utiliza para gestionar la base de datos que almacena la información sobre las interacciones del chatbot educativo, los recursos didácticos y los usuarios.

3.9. Figma

Figma⁸ es una herramienta de diseño colaborativo basada en la web, ampliamente utilizada para crear interfaces de usuario, prototipos y diseños visuales. Su principal ventaja radica en su capacidad para permitir la colaboración en tiempo real entre varios diseñadores y desarrolladores.

Figma es particularmente útil para crear y ajustar el diseño de la interfaz web de aplicaciones como la de este TFG, ya que permite diseñar pantallas interactivas y visualizar la experiencia de usuario antes de su implementación. Además, la plataforma soporta la creación de prototipos que pueden simular interacciones y flujos de navegación.

3.10. draw.io

draw.io⁹ es una herramienta gratuita y de código abierto para la creación de diagramas y esquemas, utilizada principalmente para ilustrar arquitecturas de sistemas, diagramas de flujo, diagramas de clases, y diagramas ER (Entidad-Relación).

⁶<https://www.mysql.com/>

⁷<https://www.phpmyadmin.net/>

⁸<https://www.figma.com/>

⁹<https://app.diagrams.net/>

En este TFG, se utiliza draw.io para crear esquemas visuales que representan la arquitectura del sistema, así como los diagramas ER que detallan las relaciones entre las diferentes entidades de la base de datos. draw.io permite crear diagramas detallados que pueden ser exportados en varios formatos y ser fácilmente integrados en documentos y presentaciones.

3.11. OpenAI y GPT-3.5 Turbo

OpenAI¹⁰ es una organización líder en el desarrollo de modelos de inteligencia artificial avanzados, entre los cuales se encuentra la familia de modelos GPT (Generative Pre-trained Transformer). GPT-3.5 Turbo es una de las versiones optimizadas de esta serie, diseñada para ofrecer un alto rendimiento en tareas de generación de texto natural, manteniendo un equilibrio entre velocidad, coste y capacidad de respuesta.

En el contexto de este Trabajo de Fin de Grado (TFG), se ha utilizado el modelo GPT-3.5 Turbo para generar contenidos educativos adaptados, asistiendo en la creación de actividades didácticas a través de un chatbot. Este modelo se integra mediante llamadas a la API de OpenAI, que permiten enviar mensajes en lenguaje natural y recibir respuestas coherentes, contextualizadas y adaptadas al nivel educativo deseado.

El uso de GPT-3.5 Turbo ha permitido automatizar la generación de ejercicios, explicaciones, ejemplos y propuestas didácticas de forma dinámica y personalizada, reduciendo significativamente el tiempo de elaboración de material por parte del profesorado. Entre las funcionalidades más destacadas dentro del proyecto se encuentran:

- Generación automática de actividades a partir de indicaciones del usuario.
- Adaptación del contenido según nivel de dificultad o materia.
- Respuestas explicativas y contextualizadas sobre conceptos educativos.
- Asistencia a docentes en la planificación y creatividad de recursos.

Esta integración se ha llevado a cabo utilizando el lenguaje PHP en el backend, que gestiona las peticiones a la API de OpenAI y devuelve las respuestas generadas por el modelo al entorno web del chatbot. De esta manera, se consigue una experiencia interactiva y enriquecedora dentro de la interfaz del asistente educativo.

Además, se ha tenido en cuenta la optimización del uso de tokens y la eficiencia de las consultas para controlar el coste asociado al uso del modelo, garantizando un equilibrio entre la calidad de las respuestas y la sostenibilidad computacional del sistema.

¹⁰<https://openai.com/>

3.12. Hugging Face

Hugging Face¹¹ es una plataforma líder para el desarrollo y experimentación con modelos de inteligencia artificial de código abierto. Permite acceder y probar modelos preentrenados directamente desde su hub, facilitando tareas como clasificación de texto, generación de lenguaje natural, análisis de sentimientos, entre otras.

Durante el desarrollo del TFG, se ha utilizado Hugging Face para probar diversos modelos de lenguaje de última generación, entre ellos:

- LLaMA 3
- Mistral
- Gemma
- CodeLlama
- Qwen
- DeepSeek-RL

Estas pruebas han permitido comparar capacidades y limitaciones entre modelos, evaluar tiempos de respuesta, calidad del contenido generado y su idoneidad para entornos educativos. Hugging Face ha sido útil también por su comunidad activa y recursos disponibles (datasets, espacios de inferencia, notebooks, etc.).

3.13. Ollama

Ollama¹² es una herramienta que permite ejecutar modelos de lenguaje de gran tamaño localmente, en entornos personales o de desarrollo, sin necesidad de conectarse a la nube. Su principal ventaja es la posibilidad de probar modelos open-source sin incurrir en costes por uso de APIs externas.

En el marco del TFG, Ollama se ha utilizado para experimentar de forma local con modelos como:

- LLaMA 3
- Mistral
- Gemma
- CodeLlama
- Qwen
- DeepSeek-RL

Gracias a Ollama ha sido posible evaluar la viabilidad de estos modelos en mi ordenador de forma local y analizar sus capacidades en la generación de actividades educativas.

¹¹<https://huggingface.co/>

¹²<https://ollama.com/>

Capítulo 4

Estudio de modelos LLM y técnicas de prompting

Antes de abordar el desarrollo del chatbot educativo, se ha llevado a cabo un análisis exploratorio y comparativo de distintos modelos de lenguaje de gran tamaño con el objetivo de seleccionar el más adecuado en términos de rendimiento, integración técnica y calidad de las respuestas generadas en el contexto educativo.

4.1. Modelos evaluados

Durante la fase preliminar del proyecto, se realizó un estudio exploratorio y comparativo de diferentes modelos de lenguaje de gran tamaño, con el objetivo de identificar cuál ofrecía un mejor equilibrio entre calidad de respuesta, adecuación al contexto educativo, facilidad de integración en el sistema y rendimiento técnico en diferentes entornos.

Este estudio incluyó tanto modelos propietarios como modelos de código abierto. En primer lugar, se centró en aquellos modelos que podían ejecutarse de forma local, con el fin de evitar costes externos y garantizar una mayor privacidad en el procesamiento de datos. Para ello se utilizó el gestor de modelos **Ollama**, una herramienta que permite descargar y ejecutar localmente versiones optimizadas de diversos LLMs.

La ejecución local de modelos ofrecía la ventaja de no depender de servicios externos (como APIs en la nube), pero también presentaba importantes limitaciones, sobre todo relacionadas con los recursos computacionales disponibles. Las pruebas iniciales se realizaron en un equipo personal sin tarjeta gráfica dedicada, con 8 GB de RAM, lo que restringía considerablemente la capacidad de ejecución de modelos grandes.

Modelos ejecutados localmente con Ollama

Los modelos probados mediante Ollama fueron variantes compactas, principalmente de alrededor de 7 mil millones de parámetros (7B), ya que son los únicos que pueden ejecutarse con cierta fluidez en entornos sin GPU potente. La RAM necesaria para cargar cada modelo oscilaba entre los 3.8 GB y los 5.0 GB, lo que ya suponía un límite considerable para la máquina utilizada. Los modelos probados fueron los siguientes:

- **deepseek-r1** (requiere aprox. 4.7 GB de RAM): Especializado en tareas de programación. Generaba muchas respuestas en inglés o en chino, incluso cuando se especificaba claramente el idioma español en el prompt.
- **codestral** (12 GB de RAM): Modelo enfocado en la generación de código fuente. Mostró escasa capacidad para dar explicaciones o razonar pedagógicamente.
- **qwen:7b** (4.5 GB de RAM): Modelo multilingüe desarrollado por Alibaba. Aunque generaba respuestas gramaticalmente correctas, no lograba mantener coherencia a largo plazo ni seguir indicaciones estructuradas.
- **codellama** (3.8 GB de RAM): Variante de LLaMA centrada en código. Ágil en tareas técnicas simples, pero sin capacidades educativas destacables.
- **gemma** (5.0 GB de RAM): Modelo generalista más reciente. Las respuestas eran más naturales, aunque seguía presentando dificultades para estructurar actividades pedagógicas.
- **mistral** (4.1 GB de RAM): Uno de los modelos más rápidos. Su velocidad era buena, pero sus respuestas eran demasiado generales o poco útiles en un contexto docente.
- **llama3** (4.7 GB de RAM): Prometedor en cuanto a generación de texto coherente, pero al ejecutarse sin GPU, no podía responder de forma fluida a entradas largas o tareas complejas.

Aunque estos modelos ofrecían un punto de partida interesante y presentaban ventajas como la gratuidad, el control local y la facilidad de prueba, los resultados obtenidos fueron en general insuficientes para los objetivos del proyecto. Las principales limitaciones observadas fueron:

- **Problemas de idioma:** Algunos modelos, como se ha explicado anteriormente, generaban sistemáticamente respuestas en inglés o chino, pese a usar prompts en español.
- **Falta de coherencia pedagógica:** La mayoría estaban diseñados para generar texto o código de forma generalista, pero no para seguir estructuras pedagógicas o formular actividades formativas completas.
- **Limitaciones por recursos del sistema:** Al no disponer de una GPU ni suficiente RAM, los modelos más grandes no podían cargarse o tardaban en generar respuestas, incluso bloqueándose en algunos casos.

Intentos adicionales mediante Hugging Face

También se exploró el uso de modelos desde Hugging Face, utilizando las versiones disponibles para descarga e integración en local. Sin embargo, los problemas persistieron: tiempos de respuesta excesivos, cargas fallidas por falta de RAM, y poca adaptabilidad del contenido generado. Algunos modelos directamente no podían ser iniciados por superar la memoria máxima disponible, y los que sí lo hacían presentaban una latencia elevada que impedía una experiencia fluida.

Estos inconvenientes motivaron la búsqueda de una solución más robusta: la ejecución de modelos más grandes en un entorno con recursos computacionales más potentes, lo que se abordó en la siguiente fase con el servidor de la universidad.

4.1.1. Pruebas en el servidor de la universidad

Para superar las limitaciones de hardware experimentadas durante las pruebas locales, se solicitó acceso a un servidor de la universidad, en concreto al servidor del departamento de Lenguajes y Sistemas de Información, equipado con tarjetas gráficas de alto rendimiento. Este entorno ofrecía una capacidad computacional significativamente mayor, lo cual permitía probar modelos más complejos y con mayor número de parámetros. La conexión al servidor se realizaba mediante protocolo SSH, y en el directorio personal se creó un script personalizado que permitía descargar y ejecutar modelos desde la plataforma Hugging Face.

En esta fase se decidió no utilizar Ollama, ya que no es una práctica habitual en entornos compartidos de producción o investigación donde se promueven herramientas más estandarizadas como las bibliotecas oficiales de Hugging Face. Por tanto, todas las pruebas se realizaron usando directamente el ecosistema de *transformers* y *accelerate*, respetando las buenas prácticas de uso en sistemas multiusuario.

Se repitieron las pruebas con algunos modelos ya utilizados en local, como LLaMA 3, pero en versiones más potentes. Concretamente, se probaron modelos de 13 mil millones de parámetros (13B), como:

- **LLaMA 3 13B (Meta)**
- **Mistral 13B (Mistral AI)**

Gracias a la tarjeta gráfica disponible en el servidor, una **NVIDIA GeForce RTX 3090** con 24 GB de VRAM, fue posible cargar y ejecutar estos modelos sin bloqueos, obteniendo resultados más consistentes y respuestas más extensas y razonadas que en las pruebas anteriores realizadas en local.

El rendimiento también mejoró notablemente: los tiempos de respuesta eran menores y la interacción más fluida. Sin embargo, a pesar de estas ventajas, los modelos seguían presentando limitaciones importantes desde el punto de vista educativo.

- Las respuestas, aunque técnicamente correctas, seguían sin ajustarse adecuadamente a los objetivos pedagógicos.

- No eran capaces de generar actividades educativas completas, con estructura clara y criterios didácticos definidos.
- No comprendían bien el rol del usuario (profesor/alumno) ni se adaptaban al tipo de interacción esperado en un entorno formativo.

Además, se detectó una limitación crítica: aunque existían modelos de 65B y 70B parámetros que podrían ofrecer un rendimiento significativamente mejor, estos requerían más de una GPU en paralelo para poder ser cargados en memoria. En la mayoría de los casos, estos modelos necesitan técnicas como *model parallelism* o sistemas distribuidos, y generalmente requieren tarjetas como las NVIDIA A100 de 80 GB de VRAM o múltiples GPUs unidas mediante interconexión de alta velocidad (como NVLink).

Aunque el servidor de la universidad dispone de varias tarjetas gráficas, como la **NVIDIA GeForce RTX 3090**, **NVIDIA GeForce RTX 2080 Ti**, y una **Tesla V100-PCIE-32GB**, las pruebas realizadas en esta fase se llevaron a cabo utilizando exclusivamente la RTX 3090. Esta tarjeta, con 24 GB de VRAM, permitió ejecutar modelos de hasta 13 mil millones de parámetros de forma estable, pero resultó insuficiente para probar modelos más grandes, como los de 65B o 70B parámetros, que requieren cantidades de memoria superiores o incluso varias GPUs trabajando en paralelo mediante técnicas de paralelización de modelo (*model parallelism*) y comunicación de alta velocidad (como NVLink).

Esto limitó las pruebas a modelos intermedios, que si bien ofrecían una mejora notable respecto a los modelos ejecutados en local, no alcanzaban el nivel de calidad requerido para satisfacer los objetivos pedagógicos del proyecto.

Estas observaciones consolidaron la necesidad de recurrir a un modelo más potente.

4.1.2. Exploración de modelos en la nube y análisis de costes

Tras las limitaciones observadas en la ejecución local y en el servidor universitario, se llevó a cabo una investigación sobre alternativas viables para ejecutar modelos más potentes mediante servicios en la nube. Durante esta fase se descubrió que algunas empresas como OpenAI¹, Anthropic² o Cohere³ permiten acceder a modelos avanzados de lenguaje mediante APIs externas.

En concreto, se analizó el ecosistema de OpenAI, que ofrece acceso a varios modelos de diferentes capacidades mediante su plataforma de desarrollo. Entre los modelos más destacados se encuentran:

- **GPT-3.5 Turbo:** Modelo optimizado en coste-rendimiento, diseñado para uso general y disponible con acceso rápido mediante API.
- **GPT-4 y GPT-4 Turbo:** Modelos más avanzados, con mejor razonamiento y comprensión contextual, pero con un coste por token significativamente superior.

¹<https://openai.com/api>

²<https://www.anthropic.com/>

³<https://cohere.com/>

Durante las pruebas se utilizó temporalmente el modelo GPT-4 Turbo, y se comprobó que ofrecía una mejora apreciable en la calidad y precisión de las respuestas. Sin embargo, su uso sostenido no era viable dentro de los recursos asignados al proyecto, debido a su precio.

Según la documentación oficial de OpenAI:

- **GPT-3.5 Turbo:** \$0.0015 por 1.000 tokens de entrada y \$0.0020 por 1.000 tokens de salida.
- **GPT-4 Turbo:** \$0.01 por 1.000 tokens de entrada y \$0.03 por 1.000 tokens de salida⁴.

Esto implica que el coste por interacción con GPT-4 Turbo puede ser hasta 10 veces superior al de GPT-3.5 Turbo. Por tanto, se optó por este último, ya que proporcionaba una excelente relación entre coste, rendimiento y facilidad de integración.

Este análisis económico fue determinante para la elección final, y se consideró una parte esencial de la planificación del proyecto, asegurando la viabilidad de uso del asistente tanto en pruebas como en un hipotético despliegue en la vida real.

4.1.3. Decisión de utilizar la API de OpenAI

Después de realizar múltiples pruebas con modelos de código abierto en local y en el servidor de la universidad, y tras analizar las distintas opciones de uso de modelos en la nube, se tomó la decisión de utilizar el modelo GPT-3.5 Turbo de OpenAI como motor principal del asistente conversacional del proyecto.

Esta elección fue el resultado de un proceso riguroso de evaluación, en el que se valoraron factores técnicos, pedagógicos y económicos. Si bien los modelos de código abierto ofrecían la ventaja de ejecutarse localmente sin coste por uso, sus capacidades se vieron limitadas por el hardware disponible, especialmente en un entorno personal sin GPU dedicada. Incluso con el uso de una GPU más potente como la NVIDIA RTX 3090 en el servidor universitario, solo fue posible ejecutar modelos de hasta 13 mil millones de parámetros, lo que excluía modelos más avanzados que podrían haber ofrecido una mayor calidad de respuesta.

Al mismo tiempo, se llevó a cabo una investigación sobre los modelos accesibles mediante plataformas en la nube. Esta exploración reveló que empresas como OpenAI, Anthropic y Cohere ofrecen APIs para acceder a modelos de lenguaje de última generación. De entre todas ellas, OpenAI destacó no solo por su documentación detallada y su facilidad de integración, sino también por la posibilidad de elegir entre distintos modelos según necesidades de rendimiento y coste.

Durante esta fase se probaron dos modelos principales de OpenAI: **GPT-3.5 Turbo** y **GPT-4 Turbo**. Si bien GPT-4 Turbo ofrecía una calidad ligeramente superior en términos de razonamiento y profundidad en las respuestas, su coste por uso era significativamente mayor. Según la información oficial de la propia plataforma, el precio por 1.000 tokens en GPT-4 Turbo es de \$0.01 para tokens de

⁴Precios extraídos de la documentación oficial: <https://openai.com/api/pricing>

entrada y \$0.03 para tokens de salida, frente a los \$0.0015 y \$0.0020 respectivamente de GPT-3.5 Turbo⁵. Este diferencial implica que, en escenarios de uso intensivo, el coste operativo del modelo GPT-4 Turbo podría multiplicar por diez el de GPT-3.5 Turbo.

Además del coste, se tuvo en cuenta el tiempo de respuesta, aspecto especialmente importante en aplicaciones interactivas como un chatbot educativo. En este sentido, GPT-3.5 Turbo demostró un rendimiento notable, respondiendo en tiempo real o con apenas unos segundos de latencia, lo cual resultaba más que suficiente para el propósito del proyecto. Por otro lado, su integración técnica mediante API REST resultó sencilla y directa, lo que permitió incorporarlo fácilmente al backend de la aplicación sin necesidad de infraestructura adicional ni reentrenamiento.

Otro factor clave fue la capacidad del modelo para adaptarse mediante *prompting* avanzado. A pesar de no haber sido entrenado específicamente para tareas educativas, GPT-3.5 Turbo respondió correctamente a instrucciones complejas diseñadas para contextualizarlo como tutor virtual, generador de actividades o asistente pedagógico, lo cual permitió personalizar su comportamiento sin recurrir a técnicas costosas como el *fine-tuning*.

En resumen, la decisión de utilizar GPT-3.5 Turbo fue tomada tras valorar la combinación de:

- **Calidad y coherencia de las respuestas**, incluso en contextos técnicos y pedagógicos.
- **Adaptabilidad** del modelo mediante instrucciones internas personalizadas (*system prompts*).
- **Velocidad de respuesta** adecuada para un entorno interactivo.
- **Facilidad de integración** en una aplicación web ya existente.
- **Coste razonable y controlado** para su uso en entorno académico o de prototipado.

Esta decisión no solo mejoró la calidad del asistente desarrollado, sino que también aseguró su viabilidad en un posible despliegue futuro, garantizando sostenibilidad, rendimiento y personalización en un entorno educativo real.

4.2. Resultados de la comparación

Durante el desarrollo del TFG se llevó a cabo una evaluación comparativa de distintos modelos de LLMs, con el objetivo de analizar su idoneidad en contextos educativos vinculados al aprendizaje de la programación.

Para facilitar la interpretación de los resultados y su posterior consulta, se ha elaborado una hoja de cálculo, en la que se recopilan las respuestas obtenidas por cada modelo ante una serie de prompts diseñados para generar explicaciones, actividades o ejercicios didácticos. Dicha hoja incluye, además, una anotación en verde

⁵<https://openai.com/api/pricing>

donde se explica qué cambio o estrategia se ha aplicado al prompt en cada caso, con el objetivo de mejorar la calidad de la respuesta. A continuación se presenta el prompt enviado y en columnas las respuestas generadas por cada modelo.

La hoja de cálculo está organizada en dos bloques diferenciados:

- En la primera parte se recogen los resultados obtenidos con modelos open-source ejecutados localmente mediante **Ollama**, así como en el servidor de la universidad. Aquí se incluyen modelos como LLaMA 3, Codestral, DeepSeek y otros. En esta fase se observa que, a pesar de ciertos ajustes en el prompt, muchas de las respuestas generadas por estos modelos presentan una falta de coherencia significativa o no responden adecuadamente a lo solicitado, lo que limita su aplicabilidad directa en contextos educativos.
- En la segunda parte se presentan los resultados obtenidos utilizando modelos de OpenAI a través de su API, concretamente **gpt-3.5-turbo** y, en menor medida, **gpt-4-turbo**. En este caso, las respuestas muestran una mayor calidad, precisión y alineación con el objetivo educativo de cada prompt. Sin embargo, pese a que GPT-4 ofrece una leve mejora en la calidad de las respuestas, su uso fue descartado debido a su coste significativamente superior. Por tanto, se optó por utilizar **gpt-3.5-turbo** como motor principal del asistente educativo desarrollado.

La hoja de cálculo puede consultarse en el siguiente enlace: **Google Sheets - Comparación de respuestas de LLMs**.

En general, los modelos open-source ofrecen un buen punto de partida y la ventaja de poder ejecutarse localmente, pero presentan limitaciones importantes a la hora de generar actividades educativas completas, coherentes y adaptadas a un contexto pedagógico concreto. En cambio, los modelos de OpenAI, especialmente **gpt-3.5-turbo**, ofrecieron un equilibrio óptimo entre coste, calidad y facilidad de integración, por lo que se seleccionó como solución definitiva para el asistente propuesto.

4.3. Técnicas de prompting empleadas

La personalización del comportamiento del modelo se logró mediante el diseño de un *prompting* interno avanzado. Este sistema permite:

- Definir roles específicos (tutor, generador de actividades, etc.).
- Adaptar la respuesta al tipo de usuario (alumno o profesor).
- Incluir competencias, objetivos de aprendizaje y contenidos programáticos en el contexto.
- Controlar la estructura de las actividades generadas.
- Restringir comportamientos no deseados, como dar la solución directa al alumno.

Esta estrategia se convirtió en una alternativa eficaz al *fine-tuning*, permitiendo adaptar un modelo generalista a un dominio educativo concreto de forma dinámica.

4.4. Principales conclusiones

El estudio comparativo de modelos LLM realizado en esta fase del proyecto ha sido clave para comprender las capacidades, limitaciones y requisitos técnicos de distintas arquitecturas aplicadas al ámbito educativo. A lo largo de las pruebas, se han explorado múltiples enfoques: desde la ejecución local de modelos compactos, pasando por la utilización de servidores con GPU, hasta el análisis de soluciones en la nube.

Este análisis no solo permitió constatar las diferencias de rendimiento entre modelos, sino que también evidenció la necesidad de adoptar un enfoque más realista y orientado a la finalidad del proyecto. Mientras que muchos modelos generaban texto coherente o fragmentos de código funcionales, pocos eran capaces de formular actividades pedagógicas completas, adaptadas al nivel del usuario y a objetivos de aprendizaje específicos. De este modo, quedó claro que no basta con que un modelo sea técnicamente sólido o capaz de generar respuestas sintácticamente correctas; es indispensable que entienda la intención educativa subyacente y sea capaz de estructurar salidas útiles en ese contexto.

Por otra parte, las limitaciones impuestas por el hardware (especialmente al trabajar sin GPU o con recursos limitados) influyeron notablemente en la calidad y fluidez de las respuestas generadas por los modelos locales. Aunque herramientas como Ollama facilitaron la experimentación, las condiciones de ejecución afectaron directamente a la experiencia del usuario y a la viabilidad de las tareas propuestas. Del mismo modo, incluso en entornos más potentes, como el servidor de la universidad, surgieron cuellos de botella relacionados con la imposibilidad de ejecutar modelos de gran tamaño sin técnicas de paralelización avanzadas.

A nivel metodológico, este estudio permitió identificar criterios de evaluación relevantes para seleccionar un modelo en función del contexto: precisión en el idioma, coherencia pedagógica, tiempo de respuesta, capacidad de integración, y adaptabilidad a distintos niveles de complejidad. Esta reflexión condujo a una visión más estratégica sobre el papel del prompting y la ingeniería de instrucciones como complemento indispensable para guiar a los modelos hacia respuestas más útiles y alineadas con los objetivos del proyecto.

Finalmente, el análisis de modelos en la nube demostró que, aunque el coste es un factor a considerar, estos servicios ofrecen una alternativa viable para obtener una calidad de respuesta significativamente superior. Modelos como GPT-4 Turbo ofrecieron resultados notablemente mejores en cuanto a estructura, claridad y adecuación al contexto educativo, lo que justificó su elección para las fases posteriores del proyecto, especialmente en la generación de actividades, casos prácticos y explicaciones personalizadas.

En resumen, esta fase permitió no solo comparar modelos en términos cuantitativos, sino también profundizar en aspectos cualitativos relacionados con la naturaleza del contenido educativo. La experiencia obtenida sirvió como base sólida para orientar las decisiones técnicas y metodológicas del desarrollo del chatbot, poniendo en valor la importancia del entorno de ejecución, la ingeniería de prompts y la calidad de los modelos seleccionados.

Capítulo 5

Desarrollo de la Aplicación

Este capítulo describe en detalle el proceso de desarrollo de la aplicación web diseñada como parte del Trabajo de Fin de Grado. La aplicación tiene como objetivo principal servir de asistente educativo inteligente para estudiantes y docentes del ámbito de la ingeniería informática, facilitando el aprendizaje y la enseñanza en asignaturas técnicas como Fundamentos de Programación, Fundamentos de Ingeniería del Software y Sistemas Concurrentes y Distribuidos. La finalidad de esta herramienta es generar recursos didácticos personalizados, apoyar el aprendizaje autónomo de los estudiantes y ofrecer al profesorado una plataforma dinámica para el seguimiento de los estudiantes.

La aplicación integra un chatbot basado en LLMs, específicamente utilizando GPT-3.5 Turbo de OpenAI, accedido mediante su API oficial. Este modelo ha sido empleado para la generación automática de contenido educativo, resolución de dudas y explicación de conceptos técnicos, adaptándose al nivel y contexto del usuario.

A través de este chatbot, el usuario (dependiendo de su rol dentro de la aplicación) puede realizar preguntas sobre conceptos clave, resolver dudas, solicitar actividades educativas y recibir retroalimentación inmediata.

Todos los usuarios de la aplicación podrán iniciar y cerrar sesión de forma segura, accediendo a una interfaz personalizada según su rol dentro del sistema. A continuación, se describen las funcionalidades principales disponibles para cada tipo de usuario:

- **Interfaz para estudiantes:** El estudiante dispone de un menú lateral siempre visible que le permite acceder rápidamente a las distintas funcionalidades de la aplicación, como el chatbot, la página de inicio o la opción de cerrar sesión.

En la pantalla de inicio se muestra un listado con todas las clases en las que el estudiante está matriculado. Desde ahí podrá acceder a cada una de ellas y consultar los recursos disponibles. Dentro de una clase, el estudiante podrá:

- Visualizar las actividades publicadas por el docente.

- Interactuar con el chatbot, el cual estará contextualizado según la asignatura seleccionada previamente, ofreciendo respuestas y recursos ajustados a esa materia.
- **Interfaz para docentes:** De forma similar a los estudiantes, los docentes disponen de un menú lateral permanente que facilita el acceso a las distintas secciones de la aplicación.

En la pantalla de inicio se presenta un listado con todas las clases en las que el docente imparte docencia. Dentro de cada clase, el profesorado tiene acceso a un conjunto de herramientas orientadas a la gestión de actividades y errores comunes:

- Generar actividades. Estas actividades pueden ser generadas automáticamente por el chatbot, mediante la selección de distintas opciones configurables, como el tipo de actividad o los errores comunes en los que se desea basar.
- Gestión de errores comunes: el docente puede visualizar, editar, añadir y eliminar errores comunes. Estos errores se muestran en un listado que puede filtrarse por temas específicos.
- Gestión de actividades: las actividades pueden visualizarse, editarse, eliminarse o publicarse. La aplicación permite distinguir entre actividades ya publicadas (visibles para el alumnado) y aquellas que aún no lo están.
- **Interfaz para el administrador:** El administrador también cuenta con un menú lateral desde el que puede acceder rápidamente a las distintas funciones del sistema.

En su pantalla de inicio se muestran las herramientas de gestión necesarias para administrar correctamente la plataforma:

- Gestión de usuarios: permite visualizar un listado de todos los usuarios del sistema. Desde este apartado se pueden eliminar usuarios existentes, así como crear o editar usuarios mediante formularios sencillos.
- Gestión de clases: se proporciona acceso a un listado de las clases registradas en el sistema, con opciones para eliminar, editar o crear nuevas clases.
- Asignación de alumnos a clases: se muestra un listado de los estudiantes existentes junto a un menú desplegable que permite seleccionar la clase a la que se desea asignarlos.

5.1. Historias de Usuario

Durante el proceso de desarrollo de la aplicación, se ha seguido una metodología centrada en el usuario con el objetivo de garantizar que el sistema cumpla con las necesidades reales de los distintos perfiles que interactúan con él. Para ello, se han definido una serie de Historias de Usuario (HU) que recogen, de forma sencilla

y estructurada, los requisitos funcionales desde el punto de vista de los usuarios finales, así como los requisitos no funcionales.

Las historias de usuario permiten comprender el comportamiento esperado del sistema ante diferentes situaciones, reflejando tanto las funcionalidades principales como los flujos de uso asociados a cada rol (estudiante, docente y administrador).

Estas historias han servido como base para el diseño e implementación de la interfaz de usuario, la lógica del backend y las funcionalidades específicas asociadas a cada perfil.

A continuación, se presentan todas las historias de usuario:

Identificador: HU. 1	Título: Inicio de sesión de usuarios
Descripción: Como usuario (profesor, alumno o administrador), quiero poder iniciar sesión con mi correo y contraseña para acceder a la plataforma de manera segura.	
Requisitos funcionales: <ul style="list-style-type: none"> ■ El sistema debe permitir el registro de nuevos usuarios con correo y contraseña (administrador). ■ El sistema debe validar credenciales al iniciar sesión. ■ Debe haber control de acceso según el rol (profesor, alumno o administrador). ■ El sistema debe mantener la sesión iniciada hasta que el usuario cierre sesión manualmente o expire. 	
Requisitos no funcionales <ul style="list-style-type: none"> ■ El sistema debe encriptar las contraseñas en la base de datos. ■ Tiempo de respuesta para el login inferior a 3 s. ■ Interfaz accesible y responsive para cualquier dispositivo. 	
Observaciones: Se debe considerar un sistema de recuperación de contraseña en fases posteriores.	

Identificador: HU. 2	Título: Cierre de sesión
Descripción: Como usuario (profesor, alumno o administrador), quiero poder cerrar sesión para proteger mi cuenta al salir de la aplicación.	
Requisitos funcionales: <ul style="list-style-type: none"> ■ El sistema debe ofrecer un botón “Cerrar sesión” visible en todas las vistas principales. ■ Al pulsar “Cerrar sesión”, la sesión debe invalidarse y redirigirse al formulario de inicio de sesión. 	
Requisitos no funcionales <ul style="list-style-type: none"> ■ Tiempo de respuesta inferior a 2 s. ■ Confirmación visual de que la sesión se ha cerrado correctamente 	
Observaciones:	

Identificador: HU. 3	Título: Gestión de usuarios por parte del administrador
Descripción: Como administrador, quiero poder crear, editar y eliminar usuarios (profesores y alumnos) para controlar el acceso a la plataforma.	
Requisitos funcionales: <ul style="list-style-type: none"> ■ El sistema debe permitir crear nuevos usuarios indicando nombre, correo, contraseña y rol. ■ El sistema debe permitir editar datos de usuarios existentes (nombre, correo y rol) ■ El sistema debe permitir eliminar usuarios, solicitando confirmación previa. 	
Requisitos no funcionales <ul style="list-style-type: none"> ■ Validación de correos únicos. ■ Confirmación antes de acciones destructivas. ■ Interfaz clara y coherente con el resto de la plataforma 	
Observaciones:	

Identificador: HU. 4	Título: Gestión de asignaturas por parte del administrador
Descripción: Como administrador, quiero poder crear, editar y eliminar asignaturas para mantener actualizada la web para nuevas asignaturas presentes en el catálogo académico.	
Requisitos funcionales: <ul style="list-style-type: none"> ■ Crear asignaturas con campos: nombre, código, descripción, competencias y resultados de aprendizaje. ■ Editar datos de asignaturas existentes. ■ Eliminar asignaturas, mostrando advertencia si tienen clases asociadas. 	
Requisitos no funcionales <ul style="list-style-type: none"> ■ Mensajes de advertencia claros al eliminar asignaturas con dependencias. ■ Interfaz intuitiva para formularios. 	
Observaciones:	

Identificador: HU. 5	Título: Gestión de clases por parte del administrador
Descripción: Como administrador, quiero crear, editar y eliminar clases para organizar los grupos de alumnos.	
Requisitos funcionales: <ul style="list-style-type: none">■ Crear clases con campos: nombre de clase, curso y asignatura asociada.■ Editar datos de clases existentes■ Eliminar clases, solicitando confirmación previa.	
Requisitos no funcionales <ul style="list-style-type: none">■ Validar que cada clase tenga una asignatura asociada.■ Interfaz consistente con gestión de asignaturas.	
Observaciones:	

Identificador: HU. 6	Título: Asignación de usuarios a clases
Descripción: Como administrador, quiero asignar y desasignar alumnos y profesores a clases para configurar quién participa en cada grupo.	
Requisitos funcionales: <ul style="list-style-type: none">■ Selección múltiple para añadir alumnos y profesor a una clase.■ Opción para eliminar alumnos o profesor de una clase.■ Evitar duplicados en la asignación	
Requisitos no funcionales <ul style="list-style-type: none">■ Confirmación visual al desasignar un usuario.	
Observaciones:	

Identificador: HU. 7	Título: Visualización de clases impartidas (profesor)
Descripción: Como profesor, quiero ver las clases que imparto para acceder a su gestión.	
Requisitos funcionales: <ul style="list-style-type: none">■ Mostrar listado de clases asignadas al profesor.■ Enlazar cada clase con su panel de gestión (Actividades, Crear Actividad, Errores comunes).	
Requisitos no funcionales <ul style="list-style-type: none">■ Interfaz intuitiva y limpia para la correcta visualización de las clases.	
Observaciones:	

Identificador: HU. 8	Título: Gestión de errores comunes (profesor)
Descripción: Como profesor, quiero ver, crear, editar y eliminar la lista de errores comunes de una asignatura para usarlos en la generación de tareas.	
Requisitos funcionales: <ul style="list-style-type: none"> ■ Mostrar listado de errores comunes por asignatura. ■ Formulario para crear y editar errores comunes (texto descriptivo). ■ Opción para eliminar errores con confirmación. 	
Requisitos no funcionales <ul style="list-style-type: none"> ■ Interfaz de lista clara y filtrable por temas. 	
Observaciones:	

Identificador: HU. 9	Título: Generación automática de tareas (profesor)
Descripción: Como profesor, quiero generar tareas a través de un chatbot seleccionando errores comunes para agilizar la creación de actividades.	
Requisitos funcionales: <ul style="list-style-type: none"> ■ Mostrar lista de errores comunes. ■ Botón “Generar tarea” que envía la selección al chatbot. ■ Mostrar actividad generada en un editor antes de guardarla. 	
Requisitos no funcionales <ul style="list-style-type: none"> ■ Tiempo de respuesta inferior a 10 s. 	
Observaciones: Las tareas generadas deben alinearse con competencias y resultados de la asignatura.	

Identificador: HU. 10	Título: Gestión de actividades de la clase (profesor)
Descripción: Como profesor, quiero ver, publicar, editar y eliminar actividades de una clase para mantener una gestión de las actividades en cada clase.	
Requisitos funcionales: <ul style="list-style-type: none"> ■ Listado de actividades con estado (borrador/publicada). ■ Botones para publicar, editar y eliminar cada actividad (con confirmación). 	
Requisitos no funcionales <ul style="list-style-type: none"> ■ Interfaz intuitiva y coherente con las demás vistas de gestión. ■ Notificación visual al publicar. 	
Observaciones: .	

Identificador: HU. 11	Título: Visualización de clases matriculadas (alumno)
Descripción: Como alumno, quiero ver las clases de las que formo parte para poder acceder a sus contenidos.	
Requisitos funcionales: <ul style="list-style-type: none">▪ Listado de clases matriculadas en la pantalla de inicio.▪ Cada clase debe ser un enlace a su sección correspondiente.	
Requisitos no funcionales <ul style="list-style-type: none">▪ Interfaz intuitiva y limpia para la correcta visualización de las clases.	
Observaciones:	

Identificador: HU. 12	Título: Chatbot de asignatura (alumno)
Descripción: Como alumno, quiero interactuar con un chatbot dentro de cada asignatura para resolver dudas sobre las asignaturas correspondientes.	
Requisitos funcionales: <ul style="list-style-type: none">▪ Entrada de texto libre en la vista de asignatura.▪ Mostrar respuestas generadas por el chatbot en un estilo de chat.	
Requisitos no funcionales <ul style="list-style-type: none">▪ Interfaz tipo app de mensajería.	
Observaciones:	

5.2. Arquitectura del sistema

La Figura 5.1 muestra el flujo general del sistema de la principal funcionalidad de este, desde la incorporación inicial de las guías docentes hasta la interacción con el usuario y la generación de respuestas mediante el modelo de lenguaje.

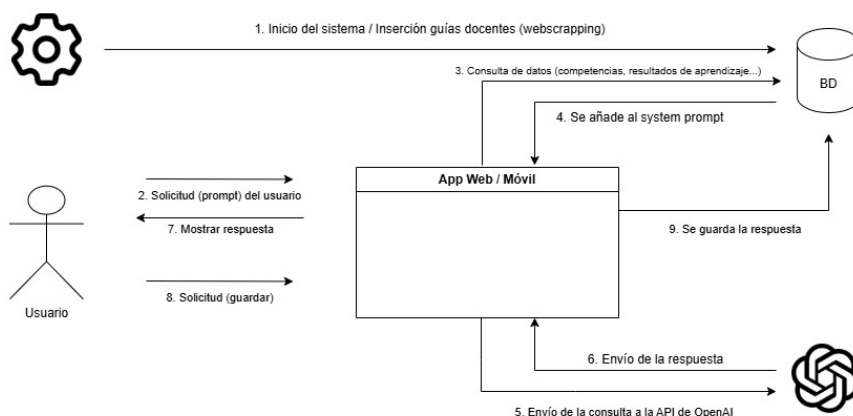


Figura 5.1: Diagrama de arquitectura del sistema

A continuación se describen los pasos que sigue el sistema:

1. Inicio del sistema / Inserción de guías docentes (web scraping):

Antes de que los usuarios puedan interactuar con el sistema, se realiza un proceso automatizado de extracción de datos desde las guías docentes oficiales. Este proceso utiliza técnicas de *web scraping* para almacenar en la base de datos información clave como:

- Competencias de cada asignatura.
- Resultados de aprendizaje.
- Contenidos.

2. Solicitud (prompt) del usuario:

El usuario accede a la aplicación web y realiza una consulta en lenguaje natural, por ejemplo: “*Generame una actividad de nivel intermedio de unos 15 minutos*”.

3. Consulta de datos a la base de datos:

La aplicación identifica la asignatura correspondiente a la consulta del usuario y recupera de la base de datos los datos educativos relevantes (competencias, resultados de aprendizaje, etc.).

4. Generación del *system prompt*:

La aplicación utiliza los datos recuperados para generar un *system prompt* que contextualiza al modelo de lenguaje. Esto permite guiar al modelo para que responda teniendo en cuenta el marco académico de la asignatura.

5. Envío de la consulta a la API de OpenAI:

El *system prompt* y el mensaje del usuario se envían a través de la API de OpenAI, en específico al modelo *gpt-3.5-turbo*

6. Recepción de la respuesta del modelo:

El modelo procesa la solicitud y devuelve una respuesta generada de forma contextualizada.

7. Visualización de la respuesta al usuario:

La respuesta se muestra al usuario de forma clara y estructurada en la interfaz de la aplicación.

8. Solicitud de guardado (opcional):

El usuario puede optar por guardar la interacción para su posterior revisión.

9. Almacenamiento de la respuesta en la base de datos:

Finalmente, la respuesta y los datos asociados a la interacción se almacenan en la base de datos para fines de seguimiento, personalización o evaluación.

Este flujo garantiza que las respuestas generadas estén alineadas con el contenido curricular y contribuyan al aprendizaje autónomo y contextualizado del estudiante.

5.3. Diseño de la base de datos

El diseño de la base de datos constituye un componente fundamental en el desarrollo del sistema educativo propuesto. Su principal objetivo es garantizar una organización estructurada, coherente y eficiente de la información relacionada con las asignaturas, los usuarios (alumnos, profesores y administradores), las clases, las actividades formativas, las entregas y los errores comunes detectados durante el aprendizaje. Esta arquitectura facilita la gestión académica, permite realizar un seguimiento detallado del progreso del alumnado y proporciona soporte a la generación automatizada de actividades y retroalimentación personalizada.

La estructura se ha diseñado conforme a los requerimientos funcionales del sistema y aplicando principios de normalización para evitar redundancias y asegurar la integridad referencial. En total, se han definido ocho tablas principales:

- **usuarios:** almacena la información básica de todos los perfiles registrados en el sistema. Cada usuario tiene un rol definido en el campo `tipo_usuario`, que puede ser `alumno`, `profesor` o `admin`.
- **asignaturas:** contiene los datos esenciales de cada asignatura, como el nombre y la descripción. Está relacionada con otras entidades como `clases`, `errores_comunes` y `guias_docentes`, lo que permite integrar aspectos curriculares y pedagógicos.
- **clases:** representa los grupos formativos concretos. Cada clase está vinculada a una asignatura y a un profesor responsable (`id_profesor`). Esta relación permite gestionar múltiples clases para una misma asignatura.
- **alumnos_clases:** define la relación muchos a muchos entre alumnos y clases, permitiendo que un alumno esté matriculado en varias clases y que una clase tenga múltiples alumnos.
- **actividad:** almacena las actividades formativas asociadas a una clase específica. Incluye información como el título, contenido, estado (`borrador` o `publicada`) y la fecha de creación.
entregas: registra el trabajo entregado por los estudiantes para cada actividad. Esta tabla asocia a un alumno con una actividad concreta e incluye el código entregado y la fecha de envío, facilitando así la evaluación individualizada.
- **errores_comunes:** permite documentar patrones de errores frecuentes cometidos por los estudiantes, organizados por tema y asignatura. Esta información es clave para generar actividades de refuerzo y retroalimentación personalizada.
- **guias_docentes:** contiene los elementos curriculares asociados a cada asignatura, incluyendo competencias, resultados de aprendizaje, y programas teórico y práctico. Aunque esta información no está accesible directamente para los estudiantes, sirve como base para que el sistema genere actividades alineadas con los objetivos académicos.

El modelo de datos diseñado proporciona una base robusta y escalable que respalda las funcionalidades clave del sistema: gestión de usuarios y clases, generación automatizada de actividades, seguimiento del aprendizaje y retroalimentación continua. A continuación, se presenta el diagrama entidad-relación que refleja las relaciones existentes entre las entidades mencionadas (ver figura 5.2).

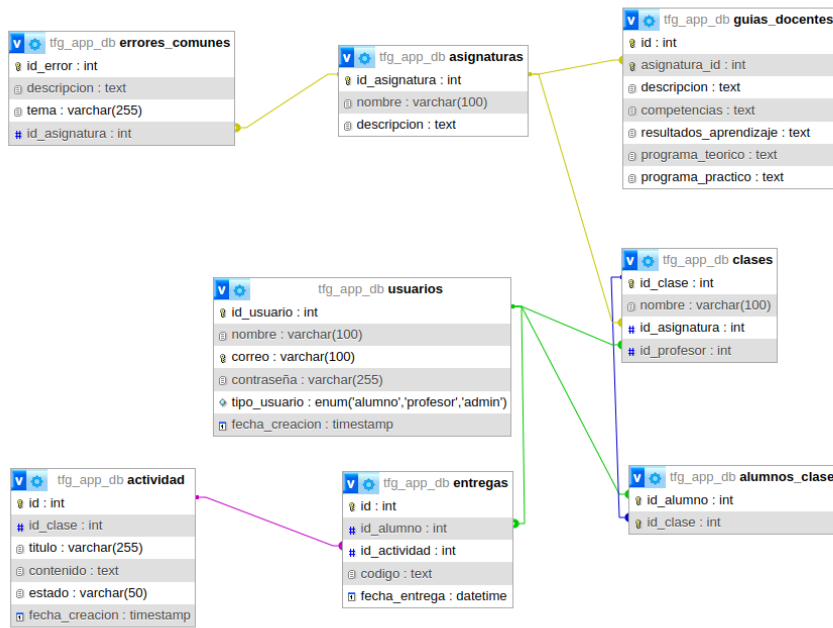


Figura 5.2: Inicio de Sesión para todos los usuarios

5.4. Diseño visual de la interfaz

El diseño visual de la interfaz de usuario se ha desarrollado utilizando la herramienta Figma, con el objetivo de crear una experiencia de uso intuitiva, clara y accesible para los usuarios. Este diseño busca reflejar la estructura funcional de la aplicación, facilitando la navegación y el acceso a las principales funcionalidades del sistema.

A lo largo de esta sección se presentan diferentes capturas del prototipo de Figma, que muestran las pantallas clave de la aplicación, como el formulario de inicio de sesión, el panel de administración, y las vistas específicas para estudiantes y docentes. Estas interfaces han sido diseñadas con un enfoque minimalista, cuidando aspectos como la jerarquía visual, el contraste, la disposición de los elementos y la consistencia del estilo.

A continuación, se muestran las pantallas más representativas del diseño inicial:

5.4.1. Pantalla de Inicio de Sesión

Esta vista permite al usuario autenticarse mediante su correo electrónico y contraseña. Se ha diseñado con un enfoque limpio y funcional, utilizando un contraste alto para el botón de acción principal y márgenes amplios para mejorar la legibilidad (ver Figura 5.3).

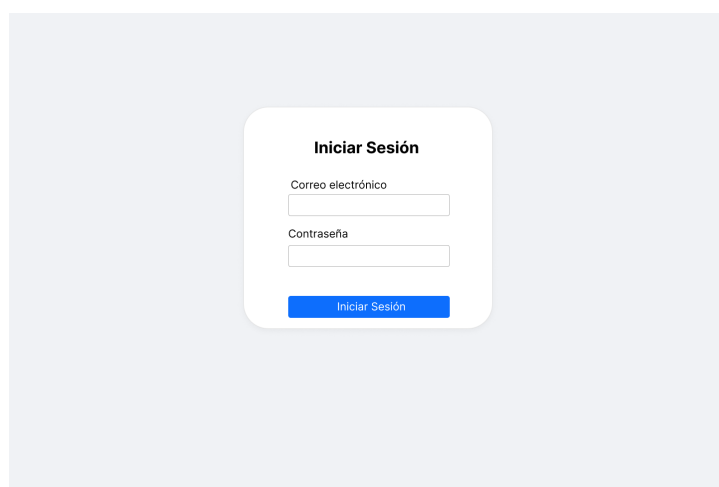


Figura 5.3: Inicio de Sesión para todos los usuarios

5.4.2. Pantalla de Inicio del Administrador

Esta vista sirve como el punto de entrada principal para los administradores del sistema. Presenta un menú lateral claro y bien estructurado para acceder rápidamente a las funciones clave, como gestión de usuarios, clases y asignación de alumnos. El diseño se enfoca en la simplicidad y eficiencia, con un enfoque en la navegación intuitiva y una organización lógica de las opciones para optimizar el flujo de trabajo diario del administrador (ver Figura 5.4).

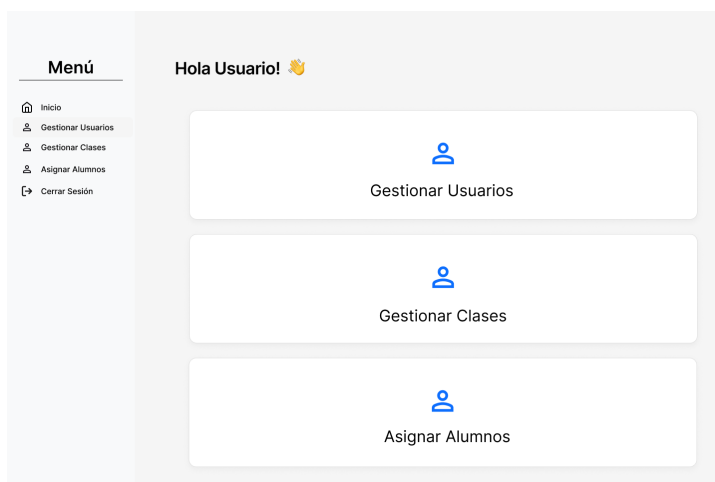


Figura 5.4: Pantalla principal (home/inicio) del administrador del sistema

5.4.3. Pantalla de Gestión de Usuarios

Esta vista facilita la administración de usuarios, mostrando una tabla estructurada con detalles como *Nombre*, *Correo*, *Tipo de Usuario* y *Fecha de Creación*. Incluye acciones rápidas para editar o eliminar usuarios, destacadas en amarillo y rojo respectivamente. Además, permite añadir nuevos usuarios mediante un botón verde, manteniendo la coherencia visual en la navegación (ver Figura 5.5).

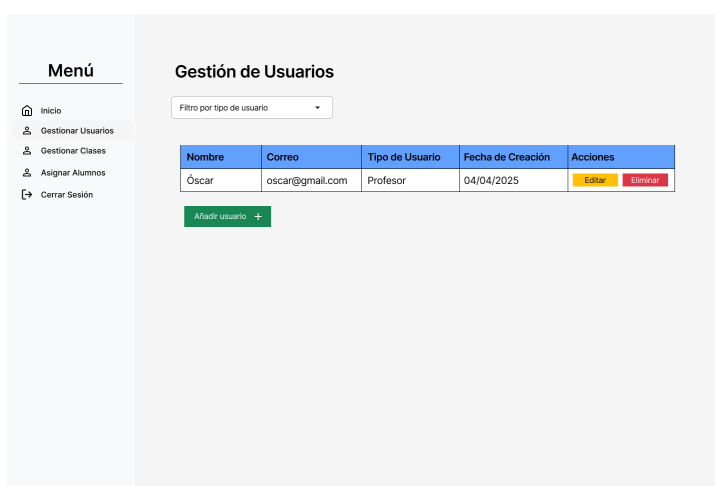


Figura 5.5: Pantalla para la Gestión de los Usuarios en la app

5.4.4. Pantalla de Gestión de Clases

Esta vista permite al usuario gestionar las clases existentes. Incluye una tabla organizada con las columnas *Nombre Clase*, *Asignatura*, *Profesor* y *Acciones*. Los botones de acción como *Editar*, *Eliminar* y *Ver*, están claramente diferenciados mediante colores para una rápida identificación. También se ofrece la opción de añadir nuevas clases con un botón verde para facilitar la expansión del contenido (ver Figura 5.6).

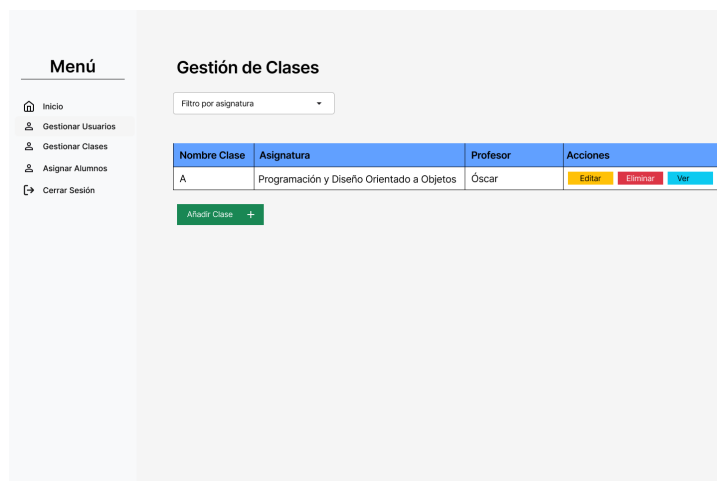


Figura 5.6: Pantalla para la Gestión de las Clases en la app

5.4.5. Pantalla de Asignar Alumnos a una Clase

Esta vista permite al usuario asignar alumnos a una clase específica. Incluye dos menús desplegables para seleccionar la asignatura y la clase correspondiente, seguidos de una lista de alumnos disponibles para asignar. El diseño es limpio y funcional, con botones de acción bien diferenciados para facilitar la interacción rápida y precisa. El botón principal para confirmar la asignación se destaca en azul para atraer la atención del usuario (ver figura 5.7).

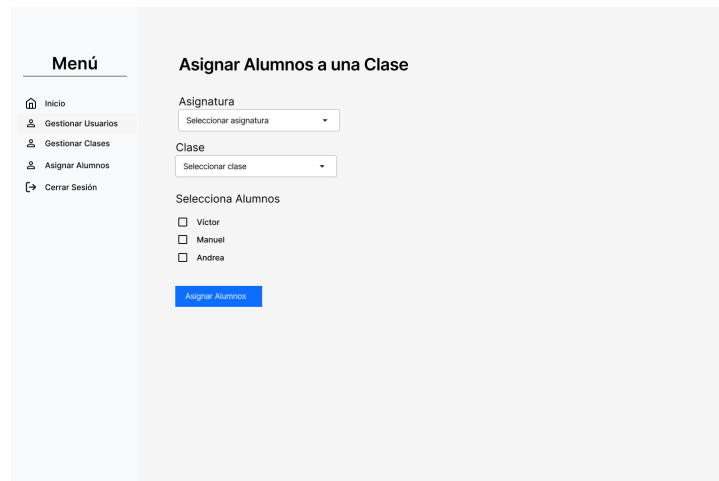


Figura 5.7: Pantalla para la Asignar Alumnos a una Clase

5.4.6. Pantalla de Inicio para los Profesores

Esta vista sirve como el punto de acceso principal para los profesores del sistema. Presenta un menú lateral sencillo con las opciones esenciales, como *Inicio* y *Cerrar Sesión*, manteniendo el enfoque en la facilidad de uso. La pantalla principal muestra las asignaturas que el profesor imparte, organizadas para facilitar el acceso rápido a cada curso y optimizar la gestión académica diaria (5.8).

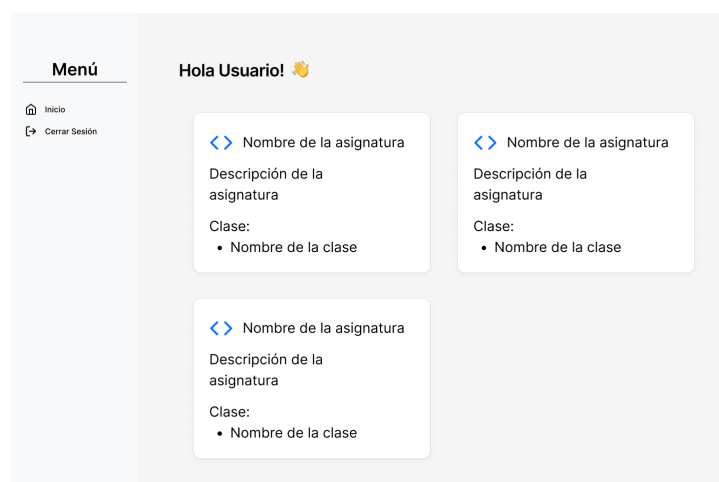


Figura 5.8: Pantalla principal (home/inicio) de los profesores

5.4.7. Pantalla de Clase de los Profesores

Esta vista muestra el nombre de la clase en la parte superior y ofrece acceso directo a las funciones clave: *Generar Actividad*, *Gestionar Actividades* y *Errores Comunes*. El menú lateral incluye opciones básicas como *Inicio* y *Cerrar Sesión*, manteniendo una estructura clara y funcional (ver Figura 5.9).

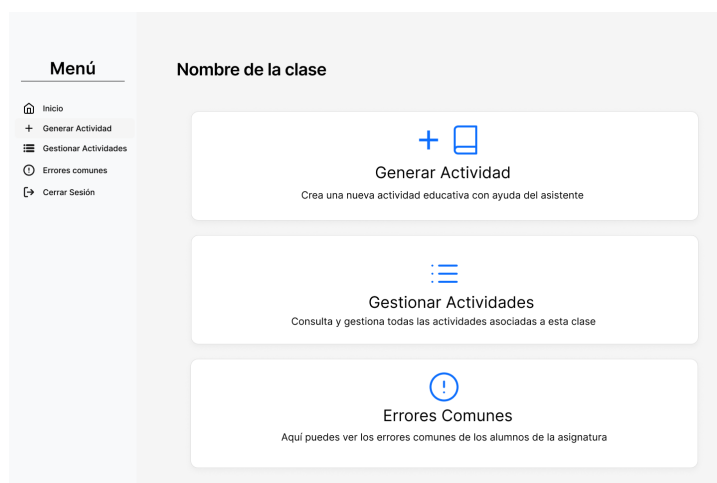


Figura 5.9: Pantalla de Inicio de cada Clase de los Profesores

5.4.8. Pantalla de Generar Actividad

Esta vista permite a los profesores hablar con el chat para la creación de las actividades a través de IA. Los profesores podrán marcar los *Errores Comunes* que quieren fortalecer en los alumnos en dicha actividad (ver Figura 5.10).

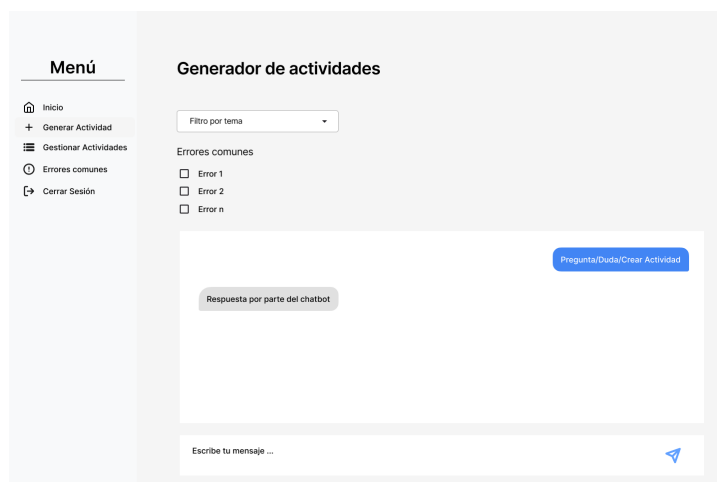


Figura 5.10: Pantalla para la Generación de Actividades por Parte de los Profesores

5.4.9. Pantalla de Gestión de Actividades

Esta vista permite al profesorado visualizar y administrar todas las actividades de la clase. Se presenta una tabla con información relevante como *Título*, *Estado* (publicada o borrador) y *Acciones* para *editar*, *eliminar*, *ver* o *cambiar su estado*. El diseño favorece una gestión rápida y clara (ver Figura 5.11).



Figura 5.11: Pantalla para la Gestión de Actividades por parte de los Profesores

5.4.10. Pantalla de Errores Comunes

Esta vista muestra una tabla con los errores comunes de la asignatura, organizados por tema. Incluye un menú desplegable para filtrar por tema y botones para editar o eliminar cada error. También se ofrece la opción de añadir nuevos errores comunes (ver Figura 5.12).

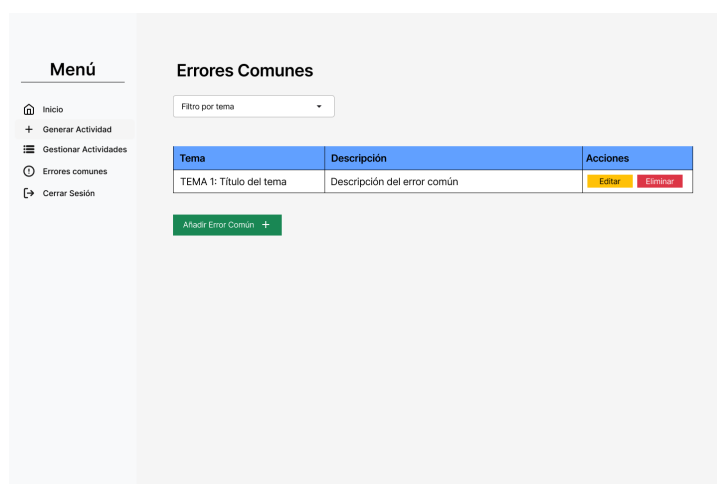


Figura 5.12: Pantalla para la Gestión de los Errores Comunes de una Asignatura

5.4.11. Pantalla de Inicio para los Alumnos

Esta vista sirve como punto de acceso principal para el alumnado. El menú lateral incluye opciones como *Inicio*, *Chatbot* y *Cerrar Sesión*. En la pantalla principal se muestran las clases en las que está inscrito, facilitando el acceso a sus recursos y actividades (ver Figura 5.13).

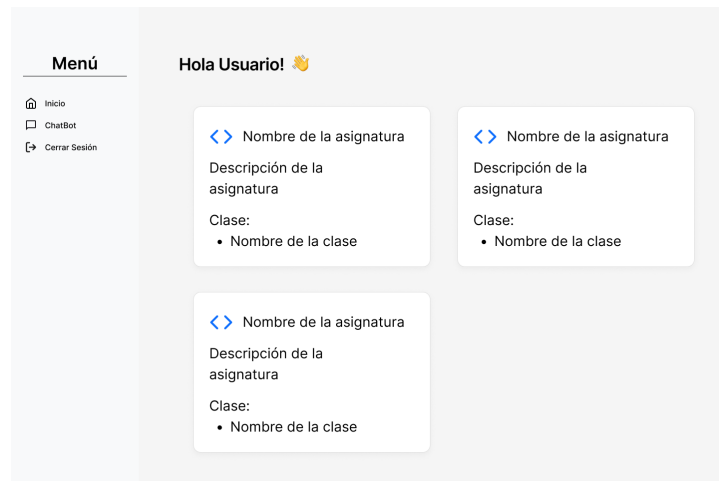


Figura 5.13: Pantalla principal (home/inicio) de los Alumnos

5.4.12. Pantalla de Inicio de Clase de los Alumnos

En esta pantalla se muestran las distintas opciones de los alumnos para cada clase: Interactuar con el chatbot para dudas sobre la asignatura o ver las actividades publicadas por el profesor en esta asignatura (ver Figura 5.14).

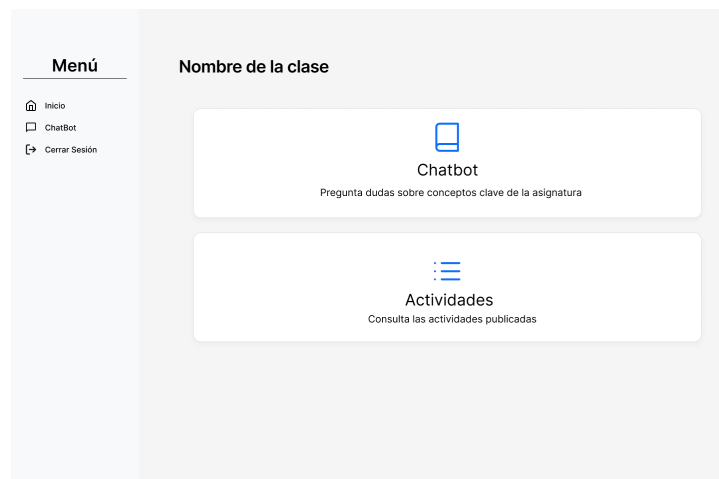


Figura 5.14: Pantalla de Inicio de cada Clase de los Alumnos

5.4.13. Pantalla de Chatbot Alumno

Interfaz de mensajería para la pregunta al chatbot y generación de respuestas de este de dudas generales sobre conceptos de la asignatura a la que pertenece la clase (ver Figura 5.15).

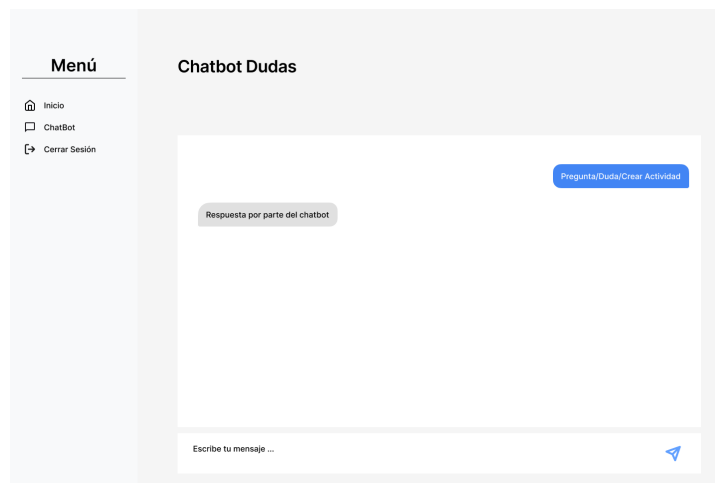


Figura 5.15: Pantalla del Chatbot para Dudas de los Alumnos

5.4.14. Pantalla de Clase de los Alumnos

En esta pantalla se muestra el título de la clase y asignatura. A continuación, se presenta una tabla con las actividades, donde el alumno puede ver el título y acceder a cada actividad (ver Figura 5.16).

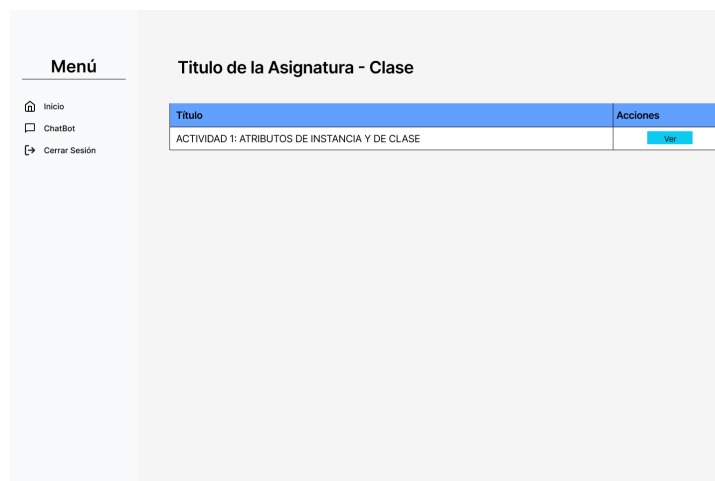


Figura 5.16: Pantalla de Actividades de cada Clase de los Alumnos

5.4.15. Pantalla de Chatbot para Dudas sobre Actividades de los Alumnos

En esta pantalla se muestra el *Título* de la actividad que el alumno está resolviendo. A continuación, se presenta el enunciado de la actividad y justo debajo de

este un cuadro de código donde el estudiante podrá resolver la actividad planteada. Finalmente se puede entregar la tarea o solicitar ayuda al chatbot (ver Figura 5.4.15).

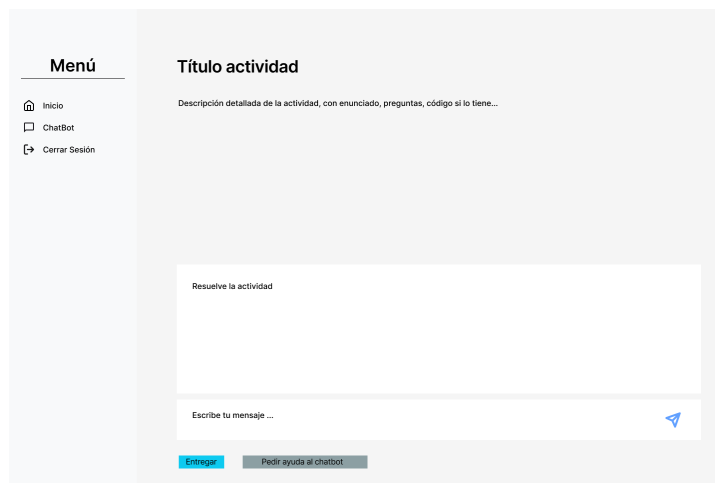


Figura 5.17: Pantalla de Inicio de cada Clase de los Alumnos

5.5. Estructura del proyecto y distribución de ficheros

La estructura del proyecto se ha mantenido simple y funcional, adaptada a las necesidades de una aplicación web desarrollada en PHP, con soporte para contenedores mediante Docker. A continuación se describe la organización de los principales directorios y archivos:

- **TFG-APP/**: Directorio raíz del proyecto.
 - **app/**: Contiene todos los ficheros PHP necesarios para la lógica de la aplicación, así como los scripts de conexión con la base de datos, las vistas y los controladores.
 - **index.php**: Punto de entrada principal de la aplicación.
 - **conexion.php**: Script de conexión a la base de datos.
 - **clases.php**, **usuarios.php**, etc.: Ficheros con la lógica de cada módulo funcional.
 - **docker-compose.yml**: Archivo de configuración principal para levantar los servicios necesarios (servidor web y base de datos) mediante Docker.
 - **Dockerfile**: Define la imagen personalizada que se utilizará para construir el contenedor de la aplicación PHP.
 - **README.md**: Documentación básica del proyecto, incluyendo instrucciones de instalación, uso y despliegue.

Esta estructura permite una rápida puesta en marcha del entorno de desarrollo mediante Docker, facilitando el despliegue y garantizando la portabilidad del sistema entre diferentes entornos.

5.6. Implementación del Chatbot

Una vez seleccionado el modelo GPT-3.5 Turbo de OpenAI como resultado del análisis comparativo descrito en el capítulo anterior, se procedió a su integración dentro de la aplicación web desarrollada para el proyecto. Este apartado describe en detalle el proceso de implementación del chatbot educativo, así como su funcionamiento técnico y pedagógico dentro del sistema.

Uno de los pilares fundamentales de este trabajo es el desarrollo e integración de un asistente conversacional basado en modelos de lenguaje (chatbot), con el objetivo de apoyar los procesos de enseñanza y aprendizaje de la programación. Aunque el modelo base empleado ha sido GPT-3.5 Turbo de OpenAI, el valor diferencial del proyecto se sitúa en el diseño pedagógico, la contextualización del modelo y su integración en una aplicación web orientada al entorno académico universitario.

5.6.1. Modelo base y justificación de uso

Para el desarrollo del asistente se ha optado por utilizar el modelo GPT-3.5 Turbo de OpenAI, por su capacidad para generar texto coherente, responder preguntas técnicas y adaptarse a distintos estilos comunicativos. Este modelo permite la generación de explicaciones, propuestas de actividades, corrección de errores y resolución de dudas técnicas relacionadas con programación, siendo clave para el tipo de tareas educativas planteadas.

El uso del modelo se realiza a través de su API oficial, permitiendo controlar los mensajes enviados y recibidos. Aunque no se ha realizado un entrenamiento adicional del modelo (*fine-tuning*), sí se ha empleado una estrategia de *prompting* avanzada para personalizar el comportamiento del asistente en función del contexto educativo de cada asignatura.

5.6.2. Diseño funcional del chatbot

El chatbot ha sido integrado en la aplicación web como una interfaz de tipo mensajería, accesible tanto para estudiantes como para profesores. La interacción se realiza mediante una caja de texto libre, y las respuestas generadas se presentan de forma estructurada y contextualizada.

Cada mensaje enviado al modelo incluye un conjunto de instrucciones diseñadas para guiar su comportamiento. Estas instrucciones varían según el perfil del usuario e incluyen información como:

- La asignatura sobre la que se está trabajando.
- Los objetivos de aprendizaje definidos en la guía docente.

- Los errores comunes más frecuentes recopilados por los docentes.
- El tipo de actividad deseada (explicación, ejercicio, retroalimentación, etc.).

Gracias a este enfoque, el modelo puede generar respuestas adaptadas sin haber sido entrenado específicamente para el proyecto, lo que permite mantener flexibilidad y control mediante el diseño de los *prompts*.

5.6.3. Uso de prompting interno y control del contexto

Este apartado es clave para comprender cómo el modelo responde de manera precisa y adaptada a las preguntas realizadas por los usuarios.

Para ajustar el comportamiento del modelo a los distintos perfiles y situaciones educativas, se emplea una técnica conocida como *prompting interno* o *system prompt*. Esta técnica consiste en incluir un mensaje oculto al usuario, pero visible para el modelo, que se envía al inicio de cada conversación o iteración. Dicho mensaje contiene instrucciones que guían al modelo sobre cómo debe comportarse y responder en un contexto determinado.

En este proyecto, el *prompting interno* se construye a partir de una serie de datos extraídos mediante *web scraping*, una técnica utilizada para obtener información estructurada desde páginas web. La fuente ha sido la página oficial de la Universidad de Granada¹, concretamente las guías docentes de las asignaturas integradas en la aplicación. De cada guía se han obtenido los siguientes apartados clave:

- **Título:** Se compara con el título almacenado en la aplicación para validar la asignatura seleccionada.
- **Competencias:** Capacidad, conocimientos y habilidades que el estudiante debe adquirir durante la asignatura.
- **Resultados de aprendizaje:** Objetivos que el estudiante debe ser capaz de alcanzar al finalizar la asignatura o unidad.
- **Programa de contenidos:** Títulos de los temas, tanto de teoría como de prácticas.

Estos datos se recargan automáticamente al iniciar la aplicación para garantizar que se mantengan actualizados, incluso si las guías docentes cambian a lo largo del tiempo.

Además de la información contextual, el *prompt* interno también define el rol que debe asumir el modelo, por ejemplo: “Actúa como un asistente educativo especializado en *nombre_asignatura*”.

El contenido del *prompt* también varía en función del tipo de usuario que interactúa con el asistente:

- **Profesorado:** En este caso, cuando se solicita la generación de una actividad, el *prompt oculto* incluye instrucciones explícitas para estructurar dicha actividad de forma pedagógica y completa, siguiendo el siguiente formato:

¹<https://www.ugr.es/estudiantes/grados/grado-ingenieria-informatica/>

- **Propósito de la actividad:** Explicación del objetivo general.
 - **Objetivo de aprendizaje:** Concepto o habilidad que se desea reforzar.
 - **Enunciado de la actividad:** Descripción del ejercicio o problema.
 - **Código base:** Fragmento de código si aplica.
 - **Nivel de complejidad:** Básico, intermedio o avanzado.
 - **Tiempo estimado:** Duración aproximada para realizar la actividad.
 - **Evaluación:** Indicadores para comprobar si se ha logrado el aprendizaje.
- **Alumnado:** Para este perfil, el *prompt* incluye restricciones destinadas a fomentar el pensamiento crítico y evitar respuestas directas o soluciones completas a los ejercicios. Se emplean frases como: “Bajo ningún concepto”, “ni siquiera por petición posterior”, “no resuelvas directamente ningún ejercicio”... Esta estrategia busca evitar que el estudiante use el chatbot como un simple solucionador automático, promoviendo así el aprendizaje autónomo.

Este diseño permite simular un comportamiento especializado, equivalente al de un modelo entrenado específicamente para un entorno educativo, pero con menor coste computacional y mayor adaptabilidad. Aunque el usuario final no ve estas instrucciones, son fundamentales para asegurar que el asistente se alinee con los objetivos pedagógicos definidos.

Además, este enfoque actúa como una primera barrera de protección frente a intentos de *jailbreaking*, es decir, intentos deliberados por parte del usuario de manipular o desactivar las restricciones del modelo a través de su entrada. Al mantener el *prompting* interno bien estructurado y adaptado al contexto, se dificulta que el modelo sea inducido a comportarse de forma inapropiada o contraria a los fines educativos. No obstante, aunque esta técnica reduce significativamente los riesgos, es importante tener en cuenta que ningún sistema basado en LLMs está completamente libre de vulnerabilidades.

5.6.4. Lógica e integración técnica

Desde el punto de vista técnico, el chatbot se ha integrado mediante llamadas HTTP a la API de OpenAI, gestionadas desde el *backend* de la aplicación, implementado en PHP. Cada mensaje del usuario se acompaña de un historial de conversación y el *prompt* interno correspondiente, que define el contexto y las instrucciones del asistente.

La lógica de la aplicación distingue entre tipos de usuario (estudiantes y profesores) y adapta los *prompts* en función del perfil. Por ejemplo:

- Para los estudiantes, el asistente ofrece explicaciones, ayuda contextual y propuestas de ejercicios personalizados.
- Para el profesorado, el chatbot puede generar actividades didácticas nuevas, editar contenidos o sugerir ejercicios basados en errores comunes detectados.

A nivel de arquitectura, el chatbot se ejecuta dentro de un contenedor Docker, lo que facilita su integración y despliegue en distintos entornos de manera controlada y replicable.

5.6.5. Limitaciones técnicas

Aunque el uso de modelos externos como GPT-3.5 Turbo ofrece importantes ventajas en cuanto a potencia y versatilidad, también plantea ciertas limitaciones:

- Dependencia de un servicio de terceros.
- Control limitado sobre la generación exacta del contenido.
- Posibles sesgos presentes en el modelo.
- Necesidad de supervisión por parte del profesorado para validar la calidad pedagógica de las respuestas.

Por esta razón, todo contenido generado por el asistente puede ser editado y validado por los docentes antes de ser compartido con el alumnado, garantizando así la adecuación didáctica de las actividades y explicaciones ofrecidas.

5.7. Ficheros de configuración y automatización

Docker es una solución eficaz para la creación, despliegue y ejecución de aplicaciones en contenedores, permitiendo la administración de múltiples servicios de manera eficiente y aislada. Para lograr una automatización adecuada en este proyecto, se emplea **Docker Compose**, una herramienta que facilita la configuración y el manejo de entornos que requieren de varios contenedores, como es el caso de una aplicación web que interactúa con una base de datos.

El archivo `docker-compose.yml` presentado en este proyecto tiene como objetivo definir y configurar los servicios necesarios para el correcto funcionamiento de la aplicación web, la base de datos y la interfaz gráfica para la administración de la base de datos (phpMyAdmin). A continuación, se explica de manera detallada el contenido de este archivo y cómo contribuye a la automatización del proceso.

5.7.1. Estructura del archivo `docker-compose.yml`

El archivo `docker-compose.yml` se estructura en varias secciones clave, cada una de las cuales tiene una función específica en la configuración del entorno de contenedores. Las secciones más relevantes son:

Definición de servicios

La sección `services` agrupa todos los servicios que serán ejecutados en contenedores. En este archivo se han definido cuatro servicios esenciales para el proyecto: la aplicación web, la base de datos, phpMyAdmin y un scraper para extracción de datos web.

Servicio web (Aplicación Web) Este servicio representa la aplicación web que será accesible para los usuarios a través de un navegador. En la configuración del servicio `web`, se especifican varios parámetros importantes:

```
web:
build: .
container_name: tfg_app
volumes:
- ./app:/var/www/html
ports:
- "8080:80"
depends_on:
- db
```

- **build:** La opción `build`: indica que el contenedor será generado a partir del `Dockerfile` ubicado en el directorio actual. Esto permite personalizar la imagen del contenedor para ajustarse a las necesidades del proyecto.
- **container_name:** Establece el nombre del contenedor, lo que facilita su identificación y gestión.
- **volumes:** El volumen `./app:/var/www/html` monta el directorio local `./app` al directorio `/var/www/html` dentro del contenedor. Esto asegura que los cambios realizados en el código de la aplicación local se reflejen de inmediato en el contenedor, facilitando el desarrollo.
- **ports:** El puerto 80 dentro del contenedor se mapea al puerto 8080 de la máquina local, lo que permite acceder a la aplicación a través de la URL `http://localhost:8080`.
- **depends_on:** Esta opción indica que el servicio `web` depende del servicio `db` (base de datos) para su correcto funcionamiento, asegurando que la base de datos esté disponible antes de iniciar la aplicación.

Servicio db (Base de datos MySQL) Este servicio configura la base de datos MySQL, que es esencial para almacenar los datos de la aplicación. Su configuración es la siguiente:

```
db:
image: mysql:8.0
container_name: tfg_app_mysql
restart: always
environment:
MYSQL_ROOT_PASSWORD: root
MYSQL_DATABASE: tfg_app_db
MYSQL_USER: usuario
MYSQL_PASSWORD: password
ports:
- "3307:3306"
volumes:
- db_data:/var/lib/mysql
```

- **image:** Se utiliza la imagen oficial de MySQL versión 8.0 para crear el contenedor de la base de datos.
- **container_name:** Define el nombre del contenedor como `tfg_app_mysql`.
- **restart:** Configura el contenedor para que se reinicie automáticamente en caso de fallo, asegurando su disponibilidad.
- **environment:** Establece las variables de entorno necesarias para la configuración de MySQL, tales como la contraseña del usuario root (`MYSQL_ROOT_PASSWORD`), el nombre de la base de datos (`MYSQL_DATABASE`), y el usuario y contraseña de acceso a la base de datos (`MYSQL_USER` y `MYSQL_PASSWORD`).
- **ports:** El puerto 3306 dentro del contenedor se mapea al puerto 3307 de la máquina local, lo que permite la conexión externa a la base de datos a través del puerto 3307.
- **volumes:** Define el volumen persistente `db_data:/var/lib/mysql` para almacenar los datos de la base de datos de forma que no se pierdan al reiniciar el contenedor.

Servicio phpmyadmin (Interfaz de administración de la base de datos) Este servicio permite acceder a la base de datos a través de una interfaz gráfica. La configuración es la siguiente:

```
phpmyadmin:
image: phpmyadmin/phpmyadmin
container_name: tfg_app_phpmyadmin
restart: always
ports:
- "8081:80"
environment:
PMA_HOST: db
PMA_USER: usuario
PMA_PASSWORD: password
```

- **image:** Utiliza la imagen oficial de phpMyAdmin para crear el contenedor.
- **container_name:** El contenedor se llamará `tfg_app_phpmyadmin`.
- **restart:** Al igual que el servicio de base de datos, el contenedor de phpMyAdmin se reiniciará automáticamente en caso de fallos.
- **ports:** El puerto 80 del contenedor se mapea al puerto 8081 de la máquina local, permitiendo acceder a phpMyAdmin a través de la URL `http://localhost:8081`.
- **environment:** Configura las variables necesarias para conectar phpMyAdmin con el servicio de base de datos, como el nombre del host (`PMA_HOST: db`), el usuario (`PMA_USER: usuario`) y la contraseña (`PMA_PASSWORD: password`).

Servicio scraper (Extracción de datos web) Este nuevo servicio ejecuta un scraper para la extracción automatizada de datos desde la web. Su configuración se basa en un contexto de construcción distinto, con dependencias y volúmenes específicos para su correcto funcionamiento:

```
scraper:
build:
context: ./webscrapping
container_name: tfg_app_scraper
depends_on:
- db
volumes:
- ./webscrapping:/app
```

- **build.context:** Indica que la imagen del contenedor se construirá a partir del Dockerfile ubicado en el directorio `./webscrapping`.
- **container_name:** Define el nombre del contenedor como `tfg_app_scraper`.
- **depends_on:** Establece que el servicio `scraper` depende del servicio `db`, asegurando que la base de datos esté disponible antes de ejecutar el scraper.
- **volumes:** Monta el directorio local `./webscrapping` dentro del contenedor en la ruta `/app`, permitiendo mantener el código del scraper sincronizado con el contenedor.

Definición de volúmenes

En el archivo también se define un volumen persistente para almacenar los datos de la base de datos MySQL. Este volumen asegura que la información no se pierda al reiniciar los contenedores.

```
volumes:
db_data:
```

5.7.2. Automatización del proceso

El uso de Docker Compose permite automatizar la creación y gestión de los servicios definidos en el archivo `docker-compose.yml`. Esto facilita la configuración del entorno de desarrollo, ya que basta con ejecutar un único comando para iniciar todos los servicios necesarios:

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

Este comando construye y ejecuta los contenedores en segundo plano, garantizando que la aplicación, la base de datos y la interfaz de administración estén disponibles para su uso.

5.7.3. Conclusión

La utilización de Docker y Docker Compose en este proyecto ha proporcionado una solución eficiente y escalable para la automatización y gestión de un entorno de desarrollo compuesto por múltiples servicios. Este enfoque ha permitido garantizar la portabilidad entre sistemas, la consistencia en la configuración de los contenedores, y una administración simplificada tanto en fase de desarrollo como en posibles escenarios de despliegue futuro.

Durante las pruebas realizadas, la aplicación ha demostrado un comportamiento estable y funcional. El uso de contenedores ha permitido ejecutar el backend, el frontend, la base de datos y la integración con el modelo de lenguaje de manera coordinada, sin conflictos de dependencias ni errores de configuración. Todas las funcionalidades clave (registro e inicio de sesión, gestión de usuarios y clases, generación de actividades educativas y chatbot integrado) se han probado de forma exitosa dentro del entorno definido por Docker Compose.

Además, esta arquitectura facilita la escalabilidad y el mantenimiento del sistema, permitiendo que otros desarrolladores o docentes puedan replicar el entorno completo con un solo comando, sin necesidad de configurar manualmente cada componente.

En resumen, la estrategia de contenerización adoptada no solo ha contribuido a mejorar la eficiencia del desarrollo, sino que también sienta una base sólida para futuras ampliaciones o despliegues del sistema en entornos reales.

5.8. Pruebas realizadas

A continuación se presentan una serie de capturas de pantalla que muestran las funcionalidades principales del chatbot educativo implementado. Estas funcionalidades reflejan cómo tanto el profesorado como el alumnado pueden interactuar con el sistema en distintos contextos del proceso de enseñanza/aprendizaje.

En la Figura 5.18 se muestra la interfaz desde la que el profesorado puede generar nuevas actividades. En esta pantalla, al seleccionar un error común previamente identificado y añadir un prompt adicional, el chatbot genera automáticamente una propuesta de actividad didáctica adaptada al contexto. Además, se habilita la opción de guardar y publicar la actividad, facilitando así su reutilización o validación posterior.

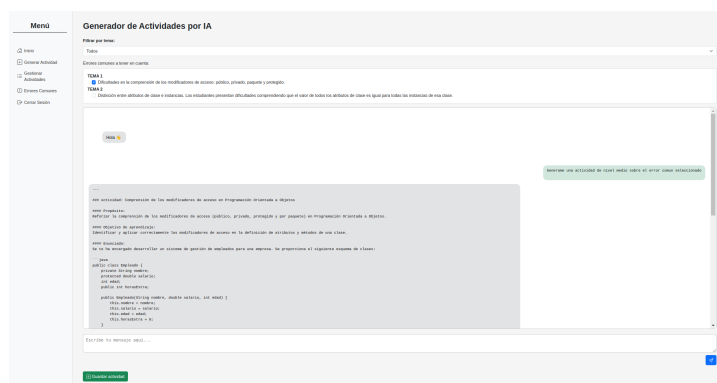


Figura 5.18: Interfaz de generación de actividades por parte del profesorado

Por otro lado, en la Figura 5.19, se observa cómo un alumno puede consultar dudas relacionadas con conceptos teóricos. El chatbot no solo proporciona una explicación adaptada al nivel del estudiante, sino que también sugiere preguntas y pequeños ejercicios con el fin de fomentar el razonamiento y reforzar el aprendizaje activo.

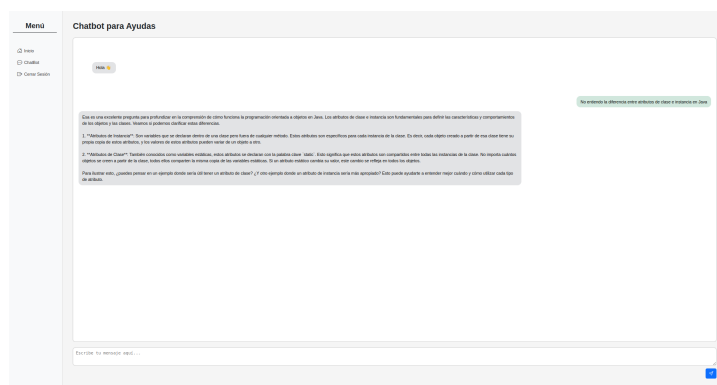


Figura 5.19: Interacción del alumno con el chatbot para resolver dudas teóricas

Finalmente, la Figura 5.20 muestra un caso en el que el estudiante consulta directamente al chatbot durante la resolución de una actividad. El sistema responde de forma coherente, ofreciendo orientación y explicaciones que le ayudan a comprender el error cometido y avanzar hacia la solución correcta. Esta funcionalidad contribuye a personalizar la experiencia de aprendizaje y reducir la frustración ante bloqueos puntuales.

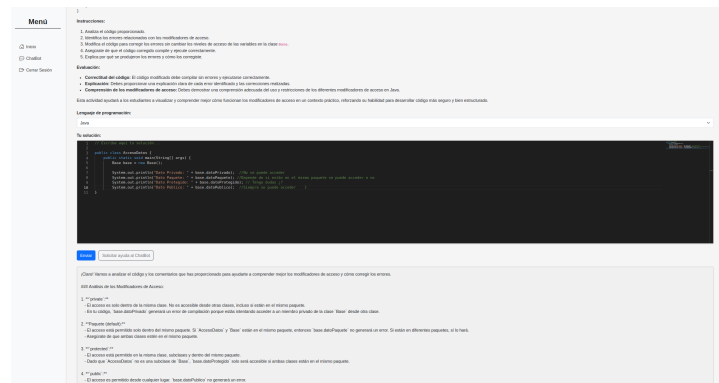


Figura 5.20: Asistencia del chatbot durante la resolución de actividades

Capítulo 6

Conclusiones y Trabajos Futuros

En esta capítulo presentaremos las conclusiones alcanzadas a lo largo de este trabajo. Además, examinaremos las perspectivas de mejora y futuros avances.

6.1. Conclusiones

A lo largo del desarrollo de este Trabajo Fin de Grado se ha llevado a cabo un análisis profundo sobre la integración de modelos de lenguaje de gran tamaño (LLMs) en el contexto educativo, concretamente en el ámbito de la enseñanza/aprendizaje de la programación. El objetivo principal ha sido diseñar y desarrollar un asistente educativo conversacional, accesible a través de una aplicación web, que permita apoyar tanto a estudiantes como a profesores mediante la generación de actividades personalizadas, explicaciones, y resolución de dudas.

Los resultados obtenidos han puesto de manifiesto diversas conclusiones relevantes:

- **Viabilidad del uso de LLMs en educación:** Se ha demostrado que los modelos de lenguaje pueden ser herramientas eficaces para generar contenido educativo adaptado a distintos niveles de dificultad y perfiles de usuario, siempre que se utilicen técnicas adecuadas de *prompting* y contextualización. Esto abre nuevas posibilidades en la personalización del aprendizaje, especialmente en entornos con gran heterogeneidad entre estudiantes.
- **Importancia del prompting avanzado:** Una de las principales claves del éxito del asistente ha sido el uso de técnicas avanzadas de *prompting*, que han permitido definir roles, controlar el contexto, adaptar las respuestas al tipo de usuario (alumno o docente) y evitar comportamientos no deseados como la resolución automática de tareas sin explicación. Este tipo de estrategias resulta esencial para alinear el comportamiento del modelo con los objetivos pedagógicos definidos.

- **Elección de modelo óptimo:** Tras una comparación detallada entre modelos open-source y servicios comerciales, se concluyó que *GPT-3.5 Turbo* ofrecía el mejor equilibrio entre coste, rendimiento, calidad de respuesta e integración técnica. Los modelos locales, si bien útiles para pruebas y con potencial para una futura adopción más autónoma, presentaron importantes limitaciones técnicas (mayor necesidad de recursos computacionales) y pedagógicas (menor coherencia o profundidad en las respuestas).
- **Funcionalidad completa de la plataforma:** Se ha implementado una aplicación web funcional que permite a los usuarios registrarse, gestionar clases, generar y resolver actividades, y comunicarse con un chatbot educativo. Esta plataforma representa una alternativa pedagógica más dinámica y adaptativa en comparación con herramientas tradicionales centradas únicamente en la gestión de contenidos, como los LMS convencionales.
- **Impacto pedagógico:** La aplicación no solo ofrece una solución tecnológica, sino que introduce un enfoque metodológico centrado en la autonomía del aprendizaje, la retroalimentación inmediata y la adaptación al ritmo del estudiante. Este planteamiento puede contribuir a mejorar la comprensión, la motivación y la confianza del alumnado, especialmente en asignaturas de programación, que tradicionalmente presentan altos índices de dificultad y abandono.

A pesar de los logros alcanzados, también se han identificado algunas limitaciones, como la necesidad de una supervisión constante del contenido generado por los LLMs, especialmente en contextos de evaluación o cuando se abordan conceptos sensibles o ambiguos. Además, la dependencia de servicios en la nube plantea desafíos relacionados con la privacidad, escalabilidad y sostenibilidad a largo plazo.

En conjunto, este TFG no solo ha cumplido con los objetivos planteados inicialmente, sino que ha puesto en evidencia el potencial real de los LLMs en entornos educativos universitarios cuando se aplican con un diseño instruccional adecuado. Asimismo, sienta las bases para futuras líneas de trabajo, como la integración de analítica de aprendizaje para personalizar aún más las actividades, la implementación de mecanismos de evaluación automática formativa, o la expansión de la herramienta a otras áreas del conocimiento más allá de la programación.

6.2. Trabajos Futuros

Si bien el asistente desarrollado representa una contribución significativa al ámbito de la educación asistida por IA, también abre numerosas vías de mejora y ampliación que podrían explorarse en trabajos futuros:

- **Mejora de la personalización del aprendizaje:** Una línea de trabajo interesante sería incorporar técnicas de aprendizaje automático que permitan adaptar dinámicamente el nivel de las actividades generadas según el progreso y rendimiento del estudiante a lo largo del tiempo.
- **Soporte multilingüe:** Aunque el modelo utilizado ofrece compatibilidad básica con el idioma español, sería valioso trabajar en su adaptación a otros

idiomas, así como en la mejora de su comprensión semántica de enunciados académicos específicos.

- **Evaluación empírica en entornos reales:** Un paso esencial en trabajos futuros será realizar estudios de campo con estudiantes y profesores reales, para evaluar cuantitativamente el impacto del asistente en la mejora del aprendizaje, la satisfacción del usuario y la eficiencia docente.
- **Despliegue con modelos open-source optimizados:** A medida que se desarrollen modelos open-source más ligeros y precisos, podría ser interesante explorar su integración para reducir costes y aumentar la independencia tecnológica del sistema.
- **Interacción multimodal:** Como evolución natural del chatbot textual, se podrían explorar interfaces multimodales que incluyan voz, imágenes o código visualizado, mejorando la accesibilidad y enriqueciendo la experiencia de usuario.

En definitiva, el presente proyecto representa solo el inicio de un campo con enormes posibilidades. La convergencia entre inteligencia artificial y educación promete transformar radicalmente los procesos de enseñanza y aprendizaje, y trabajos como este permiten sentar las bases para una integración responsable, efectiva y centrada en el estudiante.

Apéndice A

Manual de Usuario

La presente sección ofrece una visión general del uso funcional de la aplicación por parte de los distintos perfiles de usuario: administrador, profesor y alumno. Aunque el diseño visual de las distintas pantallas se detallará más adelante en el apartado 5.4, aquí se presenta una descripción resumida de las acciones principales y el flujo de navegación.

Perfiles de usuario

- **Administrador:** Gestiona usuarios y clases. Puede crear, editar y asignar estudiantes o profesores a clases a través de formularios específicos.
- **Profesor:** Puede visualizar las clases asignadas, crear actividades (manualmente o con el apoyo del asistente), revisar actividades generadas y acceder a recursos educativos.
- **Alumno:** Visualiza las clases a las que está inscrito, resuelve actividades publicadas por su profesor y puede interactuar con el asistente para resolver dudas o reforzar contenidos.

Flujo de uso general

1. El usuario accede a la aplicación mediante la pantalla de inicio de sesión.
2. Una vez autenticado, es redirigido a su panel correspondiente según su rol.
3. Desde el panel principal, puede acceder a las secciones de clases, actividades o asistente.
4. Las funcionalidades disponibles se adaptan al rol del usuario, proporcionando acceso a herramientas específicas.

Para una descripción visual y detallada de las pantallas y menús de la aplicación, véase el apartado 5.4.

Bibliografía

- [1] AENUI. Actas de las jornadas de enseñanza universitaria de la informática (jenui 2024). https://aenui.org/actas/indice_e.html, 2024. Último acceso: 14 de junio de 2025.
- [2] Amazon Web Services. ¿qué es un modelo de lenguaje de gran tamaño (llm)?, n.d. Accedido el 4 de mayo de 2025.
- [3] Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D. Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901, 2020.
- [4] CECE. Finalidad del uso de la ia en la educación, 2025. Accedido: 2025-05-01.
- [5] Diabolocom. ¿qué es el procesamiento del lenguaje natural (nlp)?, n.d. Accedido el 4 de mayo de 2025.
- [6] Ana;Martínez de Pisón Francisco Javier;Sáenz de Cabezón Eduardo Divasón, José;Romero. Modelos de inteligencia artificial para asesorar el proceso evaluador de trabajos informáticos complejos. In *Actas de las JENUI - Vol. 6 (2021)*, pages 55 – 62, Valencia, jul 2021.
- [7] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*, 2020.
- [8] José Luis Suárez Díaz, Noelia Rodríguez Barroso, and Guillermo Gómez Trenado. Bot de telegram interactivo como herramienta de apoyo para la enseñanza. In *Actas de las JENUI 2024*, pages 117–124, 2024.
- [9] Julio Alberto;Martín Baos José Ángel;P. Romero Francisco;Serrano Guerrero Jesús Jiménez, Luis;López Gómez. Chatgpt: reflexiones sobre la irrupción de la inteligencia artificial generativa en la docencia universitaria. In *Actas de las JENUI - Vol. 8 (2023)*, pages 113 – 120, Granada, jul 2023.
- [10] Luis Jiménez, José Antonio López Gómez, José Ángel Martín Baos, Francisco P. Romero, and Julio Serrano Guerrero. Chatgpt: reflexiones sobre la irrupción de la inteligencia artificial generativa en la docencia universitaria. In *Actas de las JENUI 2023*, pages 113–120, 2023.

- [11] Hyungju Jin, Sungjin Lee, Hyeonsu Shin, and Juho Kim. Teach ai how to code: Using large language models as teachable agents for programming education. *arXiv preprint arXiv:2309.14534*, 2023.
- [12] Enkelejda Kasneci, Katharina Sessler, Stefan Küchemann, Maria Bannert, Daria Dementieva, Frank Fischer, Torben Günther, and Gjergji Kasneci. Chatgpt for good? on opportunities and challenges of large language models for education. *Nature Human Behaviour*, 7:1–13, 2023.
- [13] Long Ouyang, Jeff Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, et al. Training language models to follow instructions with human feedback. *arXiv preprint arXiv:2203.02155*, 2022.
- [14] Junaid Qadir et al. Large language models (llms) for education: A survey. *Computers & Education: Artificial Intelligence*, 4:100152, 2023.
- [15] Ethan Ross, Yash Kant Kansal, James Renzella, Alex Vassar, and Adam Taylor. Supervised fine-tuning llms to behave as pedagogical agents in programming education. *arXiv preprint arXiv:2502.20527*, 2025.
- [16] Talent.com. Salario medio para ingeniero informático junior en españa, 2025, 2025. Accedido: 2025-05-01.
- [17] Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, Aurelien Rodriguez, Armand Joulin, and Edouard Grave. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*, 2023.
- [18] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. *NeurIPS*, 30, 2017.
- [19] Zixuan Wang et al. Codereviewer: Prompting chatgpt for automated code review with explainable feedback. *arXiv preprint arXiv:2303.04124*, 2023.
- [20] Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed H Chi, Quoc V Le, and Denny Zhou. Chain of thought prompting elicits reasoning in large language models. *arXiv preprint arXiv:2201.11903*, 2022.
- [21] Kaiwen Xu, Pengfei Yao, Zhaocheng Deng, Deyu Yu, and Haizhou Huang. Deepseek: Learning to reason with knowledge from language models. *arXiv preprint arXiv:2303.18223*, 2023.
- [22] Yuhuai Xu et al. Teachyou: Teaching llms via human-like conversations. *arXiv preprint arXiv:2307.05072*, 2023.

