Universidad de Valladolid

ESCUELA DE INGENIERÍA INFORMÁTICA DE SEGOVIA

Grado en Ingeniería Informática de Servicios y Aplicaciones

Generación automática de recursos de aprendizaje utilizando Large Language Models (LLM)

Alumno: Miguel Sánchez-Beato Díaz-Hellín

Tutores: Miguel Ángel Martínez Prieto
Anibal Bregón Bregón

Fecha: 14 de julio de 2025

Generación automática de recursos de aprendizaje utilizando Large Language Models (LLM)

Miguel Sánchez-Beato Díaz-Hellín

14 de julio de 2025

"La educación es lo que queda
cuando uno ha olvidado todo lo que aprendió en la escuela."
— Albert Einstein

"El jardín está, cuando lo miro,
de flores lleno."
— José Corredor-Matheos

Resumen

Los recursos de autoevaluación son una herramienta fundamental para estudiantes y profesores. El proceso de aprendizaje se beneficia ampliamente de la retroalimentación, y estos recursos la aportan de forma instantánea, pero su elaboración manual requiere una gran inversión de tiempo.

Este trabajo explora la generación automática de recursos de aprendizaje utilizando inteligencia artificial (IA), en particular, Grandes Modelos de Lenguaje (LLM). Para ello, se ha diseñado una aplicación web para la asignatura de «Sistemas de Bases de Datos» que utiliza preguntas generadas localmente con Llama 3. La ejecución local de un LLM es una tarea compleja y exigente computacionalmente. Para resolver la complejidad se emplean contenedores (*dockers*) que garantizan su funcionamiento sin problemas de compatibilidad. Para mitigar la exigencia de recursos, se recurre a la cuantización del modelo, reduciendo significativamente su consumo de memoria. Adicionalmente, se realizó un ajuste fino (fine-tuning) a partir de un *dataset* de entrenamiento para especializar el modelo al contenido específico de la asignatura.

El resultado es una plataforma funcional que evalúa al estudiante y le muestra su progreso mediante *dashboards* interactivos. Al responder a cada pregunta tipo test, el sistema la corrige y ofrece una explicación. Además, el estudiante puede solicitar la generación de nuevas preguntas, proceso que se ejecuta en su propio ordenador, permitiéndole continuar con su autoevaluación.

Palabras claves: Large Language Models, generación automática, fine-tuning, autoevaluación educativa, Docker, Node.js, Quasar, Llama 3.

Abstract

Self-assessment resources are a fundamental tool for both students and teachers. The learning process greatly benefits from feedback, and these resources provide it instantly, but their manual creation requires a significant investment of time.

This work explores the automatic generation of learning resources using artificial intelligence (AI), in particular, Large Language Models (LLM). For this purpose, a web application has been designed for the "Database Systemsçourse that uses questions generated locally with Llama 3. The local execution of an LLM is a complex and computationally demanding task. To solve the complexity, containers (dockers) are used, which guarantee its operation without compatibility problems. To mitigate the resource requirements, model quantization is used, significantly reducing its memory consumption. Additionally, a fine-tuning was performed from a training dataset to specialize the model to the specific content of the course.

The result is a functional platform that evaluates the student and displays their progress through interactive dashboards. Upon answering each multiple-choice question, the system grades the response and provides an explanation. Furthermore, the student can request the generation of new questions, a process that runs on their own computer, allowing them to continue their self-assessment.

Keywords: Large Language Models, automatic question generation, fine-tuning, educational self-assessment, Docker, Node.js, Quasar, Llama 3.

Índice general

Índice de figuras

Índice de tablas

Parte I

Descripción del proyecto

Capítulo 1

Introducción

La **inteligencia artificial o *Artificial Intelligence* (IA)** está cada día en más aspectos de nuestras vidas y supone un cambio que afecta a múltiples aspectos de ella como trabajo, educación o tiempo libre.

Podemos describir la IA como el conjunto de capacidades cognoscitivas e intelectuales que puede expresar un sistema informático o conjuntos de algoritmos que crean máquinas que imitan la inteligencia humana para realizar tareas y que son capaces de aprender recopilando información. [58] Estos algoritmos son los que dirigen nuestras redes sociales, nos ayudan a aumentar la productividad en nuestro trabajo y nos aportan nuevas soluciones a problemas.

La irrupción de los Grandes Modelos de Lenguaje o *Large Language Models* (LLMs) en estos últimos años supone una revolución que crea una serie de retos que la sociedad tiene que afrontar para sacar su máximo potencial.

Los LLM forman parte de lo que conocemos como *Inteligencia Artificial Generativa*, que son un tipo de inteligencia artificial que es capaz de producir texto, imágenes, video u otros tipos de respuesta a partir de un comando o *prompt*. La ingeniería de instrucción o *Prompt engineering* consiste en la estructuración de un texto que un modelo de inteligencia artificial generativa puede interpretar. El *prompt* es el texto en lenguaje natural que contiene la orden que le enviamos a la máquina. [57]

Los *Large Language Models* son modelos de lenguaje que habitualmente se considera que deben tener al menos mil millones de parámetros, en inglés un *billion*. [53]. Un modelo de lenguaje o *Language Model* es un modelo probabilístico del lenguaje natural [24] que pretende predecir la siguiente palabra a escribir a partir de una red neuronal que está formada por parámetros y cada palabra o frase tiene unos valores en cada uno de esos parámetros, los cuáles definirán la probabilidad de la siguiente palabra. [56]

Habitualmente se indica el número de parámetros del modelo en el nombre del mismo terminando el nombre en "-Xb" siendo X el número de miles de millones (o *billions*) de parámetros que tiene el modelo.

En general, más parámetros y una base de datos de entrenamiento más grande mejoran las capacidades del LLM. [15] [61] Pero el precio de tener más parámetros es que sea más costoso inferenciar, que es el proceso de generar texto o completar una tarea utilizando el modelo, y entrenarlo. Este coste computacional en la actualidad es un problema, pues se requieren unidades de GPU de última generación y la oferta apenas puede cumplir la demanda [49], y se gasta gran cantidad de energía [44]. Entonces lo que se busca es la optimización entre el número de parámetros y una buena base de datos de entrenamiento. Es tan importante este problema y se

investiga con tanto interés que los LLM de la misma potencia, cada año son diez veces menos costosos de inferenciar [3].

Los LLMs han puesto una presión considerable en la educación y plantea oportunidades y retos que están siendo objeto de estudio. [5] Entre ellos, se destaca la detección de texto generado por IA para asegurar la autenticidad del trabajo del estudiante y la integridad académica como se señala en [39] El estudio [26] identifica algunos problemas clave, como la adaptación de los métodos de enseñanza para aprovechar el potencial de la inteligencia artificial generativa. Además, destaca la importancia de capacitar a profesores para integrar de manera efectiva la IA generativa en el diseño de actividades de aprendizaje y promocionar la conciencia entre los estudiantes acerca del uso de la IA generativa de manera responsable y constructiva en su proceso de aprendizaje. También se destaca la importancia de promover el uso de herramientas basadas en inteligencia artificial por parte de estudiantes y profesores, de forma que, con un uso responsable, todos puedan aprovechar las posibilidades y facilidades que pueden aportar al aprendizaje.

En la actualidad, el modelo de lenguaje más ampliamente reconocido e importante es el *Generative Pre-trained Transformer* (GPT), desarrollado por OpenAI. Desde la primera versión, GPT-1, OpenAI ha ido publicando versiones cada vez más avanzadas hasta la versión más reciente en el momento de escribir esta memoria, GPT-4. Su acceso se encuentra disponible a través de una suscripción mensual, aunque también existe una versión gratuita con ciertas limitaciones de uso y rendimiento.

Para las distintas series GPT de OpenAI se ha ido aumentando el rendimiento y la cantidad de parámetros. El número de parámetros de cada GPT es: GPT-1 tiene 117 millones, GPT-2 tiene 1,5b, GPT-3 y GPT-3.5 tienen 175b de parámetros y por último, aunque no ha sido confirmado por OpenAI, se estima, a partir del tiempo que tardó en entrenarse y su rendimiento, que GPT-4 podría tener más de 1000b. El entrenamiento de GPT se realizó con contenido de internet. [10] [31]

Con tantas posibilidades encima de la mesa y el avance que se realiza constantemente en estas áreas, es imposible no plantearse la capacidad de los *Large Language Models* en el área de la educación, y en este TFG se plantea una aplicación cuyo fin es demostrar que en la actualidad ya se puede aprovechar el potencial de estos modelos para cubrir deficiencias del sistema educativo y ayudar a los estudiantes.

1.1. Planteamiento del problema (*Problem Statement*)

Estructura: (*Problem statement* (Wikipedia))

Un planteamiento del problema constituye una descripción fundamental de un problema que se debe abordar. Busca identificar la brecha entre el problema actual y el objetivo y debe responder a las preguntas ¿qué?, ¿cómo?, ¿cuándo?, ¿dónde? ¿quién? y ¿por qué?.

La estructura que toma un planteamiento de problema ayuda a la comprensión y aprobación del proyecto por parte de los interesados. Entre las distintas opciones que existen, a menudo se suele utilizar una estructura basada en cuatro partes.

La primera es el ideal, y describe el estado deseado o "previsible" del proceso o producto. Identifica los objetivos de las partes interesadas y los clientes, estableciendo el alcance esperado de la solución.

A continuación, se expone la realidad, donde se expone el estado actual del proceso o produc-

to. Se destacan los puntos débiles señalados por las partes interesadas y los clientes, respaldados por la experiencia del equipo del proyecto y los expertos en la materia.

La tercera sección es la de consecuencias, y es donde se evalúan los impactos en el negocio si el problema no se soluciona. Se analizan los costos asociados con la pérdida de dinero, tiempo, productividad, ventaja competitiva, etc., lo cual contribuye a determinar la prioridad del proyecto.

Por último se presenta la propuesta que otorga posibles soluciones después de comprender y aprobar las secciones de ideal, realidad y consecuencias. Se ofrecen opciones para resolver el problema, basadas en sugerencias de las partes interesadas y los clientes, aunque pueden requerir debates adicionales y más investigaciones.

- **IDEAL:** En un entorno educativo ideal, los alumnos dispondrían de métodos de autoevaluación de aprendizaje interactivos, personalizados y adaptables, que permitan a los alumnos caracterizar tanto el profesor como los defectos de su aprendizaje, de acuerdo a sus necesidades individuales. La automatización de estos métodos permitiría enriquecer su proceso de aprendizaje con un esfuerzo asumible por parte de los docentes.

- **REALIDAD:** Actualmente los estudiantes tienen muy pocos medios de autoevaluación y que suelen carecer de interactividad y adaptabilidad. La generación manual de preguntas y la evaluación tradicional no aprovechan las capacidades de los modelos de lenguaje avanzados. La falta de retroalimentación inmediata y la escasa visualización de estadísticas limitan la evolución y aprendizaje de los estudiantes, que tienen pocos datos de su rendimiento antes de realizar el examen.

- **CONSECUENCIAS:** Esta brecha entre el entorno educativo ideal y la realidad pueden causar que los estudiantes experimenten dificultades para autoevaluar su conocimiento y reconocer su comprensión del temario. Además, los estudiantes podrían sentir que no tienen suficientes métodos interactivos para el aprendizaje y falta de motivación.

  A su vez, los profesores se enfrentan a desafíos para ofrecer experiencias de aprendizaje personalizadas y efectivas, además del desarrollo de preguntas tipo test adecuadas para el temario desarrollado en la asignatura.

- **PROPUESTA:** La propuesta de este Trabajo Fin de Grado (TFG) busca la creación de una aplicación interactiva basada en un *Large Language Model* (LLM) que mejore la capacidad autoevaluación de los estudiantes y la evaluación para profesores. La generación automatizada de preguntas tipo test basadas en objetivos de aprendizaje, junto con la implementación de *dashboards* estadísticos, pretende cerrar la brecha entre la realidad educativa actual y el entorno educativo ideal. Esta solución integral tiene como objetivo mejorar la interacción, adaptabilidad y efectividad del proceso de aprendizaje, proporcionando a estudiantes y profesores una herramienta valiosa para la educación.

1.2. Objetivos del trabajo

- **OBJ-1:** Desarrollar una herramienta educativa que aprovecha el potencial de los LLM.

  - **SUBOBJ-1:** Filtrar información y crear una base de datos de entrenamiento para entrenar el LLM.

  - **SUBOBJ-2:** Entrenamiento del LLM a través de un proceso de fine-tuning.
  - **SUBOBJ-3:** Desarrollo de la aplicación educativa que interactúe con el LLM.
  - **SUBOBJ-4:** Creación de los *dashboards* estadísticos.

- **OBJ-2:** Ser capaz de generar preguntas de calidad dado un criterio de aceptación.
- **OBJ-3:** Garantizar una experiencia de usuario accesible e intuitiva.
  - **SUBOBJ-1:** Diseñar una interfaz amigable con navegación sencilla.
  - **SUBOBJ-2:** Asegurar la adaptabilidad a distintos tamaños de pantalla (*responsive design*).
  - **SUBOBJ-3:** Implementar notificaciones claras sobre el estado del sistema (por ejemplo, generación en curso, errores, etc.).
- **OBJ-4:** Diseñar la aplicación con una arquitectura modular y escalable que facilite su ampliación y mantenimiento futuro.

Durante el proceso de desarrollo se validarán y evaluarán los avances de cada uno de estos objetivos.

1.2.1. Restricciones

- **REST-1:** Limitamos el tiempo del TFG a unas 300-360 horas.
- **REST-2:** Los medios informáticos con los que contamos son limitados frente a los altos requisitos que supone el uso de LLMs.
- **REST-3:** Se utilizan modelos y datos de entrenamiento que permitan su uso con fines académicos.

1.3. Estructura de la memoria

La presente memoria está estructurada en 6 capítulos y el apéndice, en los cuales podemos encontrar toda la documentación correspondiente a cada aspecto del proyecto.

1. **Capítulo 1:** El presente capítulo tiene la función de introducir el contexto del proyecto, el problema que trata de resolver, los objetivos que cubrirá y las limitaciones para el desarrollo de este.
2. **Capítulo 2:** Se abarca toda la explicación de la metodología seguida y la planificación temporal y económica para el desarrollo del TFG, además de la comparación entre el tiempo y dinero estimado y el que finalmente se ha requerido.
3. **Capítulo 3:** Se presenta el entorno de negocio de los LLM en la educación y el estado del arte, donde comparamos las propuestas existentes con la nuestra y entre ellas. Finalmente, se proporciona un contexto científico-técnico de los LLM.

4. **Capítulo 4:** Se describen los requisitos de la propuesta, el diseño de la aplicación y sus diagramas, la explicación de cómo se ha desarrollado la aplicación y una serie de pruebas realizadas para comprobar el correcto funcionamiento de la aplicación.

5. **Capítulo 5:** Se presentan resultados obtenidos durante el desarrollo de la aplicación, donde evaluamos las preguntas del modelo

6. **Capítulo 6:** Se exponen las conclusiones del trabajo desde una perspectiva del proyecto y desde la perspectiva personal, y se plantean posibles líneas de trabajo futuro que permitirían, con el tiempo suficiente, hacer una aplicación más completa y valiosa.

7. **Apéndices:** Se proporciona un manual de instalación detallado donde se explican los requisitos previos y los pasos para la ejecución del proyecto, incluyendo la puesta en marcha del modelo LLM mediante un *docker*, así como la configuración del *backend* y del *frontend*. Además, se incluyen soluciones a errores frecuentes que pueden surgir durante la instalación y la ejecución. También se ofrece un manual de usuario con indicaciones para la correcta utilización y sacarle provecho a la aplicación.

A lo largo de toda la memoria, los extranjerismos estarán escritos en cursiva.

Capítulo 2

Planificación

Este capítulo abarca cuatro aspectos fundamentales en la organización para el desarrollo del proyecto, los cuales son la metodología de trabajo seguida, la planificación temporal y económica del proyecto y finalmente un análisis de la diferencia entre la realidad y la planificación temporal y económica.

2.1. Metodología de trabajo

La estrategia de trabajo seguida en este proyecto ha sido ASAP (*Agile Student Academic Projects*), que es una metodología que adapta prácticas ágiles habituales en el sector profesional para alcanzar los objetivos de aprendizaje propios del TFG. [30] Para ello se utiliza una dinámica de trabajo incremental que se basa en la interacción regular de todas las partes y la generación frecuente de feedback.

Con esta metodología es el estudiante el que se convierte en responsable de su aprendizaje, mientras que los profesores asumen funciones de asesoría, orientación, guía o facilitación del proceso, adaptándose así esta metodología perfectamente al desarrollo de un TFG.

ASAP se puede definir en base al conjunto de roles, eventos y artefactos que tienen características en común con la metodología Scrum [45] y requiere de un entorno de trabajo adecuado para el correcto desarrollo de la metodología.

2.1.1. Roles

En el ámbito del desarrollo de un TFG solamente se suele tener en cuenta al estudiante que lo desarrolla y al tutor, pero la realidad es que no son los únicos que influyen durante el proceso. Podemos clasificar y definir de la siguiente manera los distintos roles que influyen en el TFG y su misión dentro del marco ASAP.

Estudiante

El primer y más importante rol es el del estudiante que se ha matriculado en la asignatura "Trabajo Fin de Grado". Es responsable de la realización del TFG. Teniendo en cuenta que la metodología adapta prácticas ágiles habituales del sector profesional, en la empresa sería el desarrollador.

Su tarea consiste en generar un producto de calidad con una documentación, que será la memoria, que contenga toda la información necesaria. Esta documentación cubrirá la explicación del proyecto desde la idea inicial al producto final, detallando aspectos como la planificación seguida, el estado del arte y una explicación de la propuesta y todos los conceptos relevantes para la comprensión del desarrollo del producto.

En ASAP el estudiante tiene que cumplir con distintas responsabilidades para el correcto funcionamiento de la metodología, y las principales son:

- Plantear la planificación del *sprint* a partir de las tareas a realizar.
- Cumplir la planificación del *sprint* completando cada tarea teniendo en cuenta sus criterios de aceptación en el plazo temporal del *sprint*
- Mantener una comunicación fluida con el tutor a través del medio acordado, en este caso, el canal de Microsoft Teams.
- Asistir a las reuniones organizadas.
- Mantener actualizado el *tablero del proyecto*
- Mantener actualizado el *cuaderno de trabajo*,
- Comprometerse con la mejora continua del producto a partir del *feedback* proporcionado por el resto de roles.

Finalmente, el estudiante tras haber terminado el TFG lo expondrá ante el tribunal mediante una presentación y una demostración en caso de haber desarrollado una aplicación.

Tutor

Este rol lo desempeñan uno o varios profesores que tutelan al estudiante. Su equivalente en la empresa sería el *Product Owner* y *Scrum Master*.

Al tutor le corresponden una serie de responsabilidades clave para el éxito del proyecto en el marco de ASAP. En la primera etapa del Trabajo Fin de Grado, el tutor trabaja con el estudiante para plantear el problema y definir los objetivos del proyecto. También en esta primera etapa facilita al estudiante recursos y referencias bibliográficas para establecer una base en el inicio del proyecto.

En el inicio de cada *sprint* participa en la definición de los objetivos del *sprint* junto al estudiante y se asegura de que estén alineados con las metas generales del proyecto. Además, proporciona retroalimentación basada en los resultados al estudiante en las distintas reuniones que se realizan y al final del *sprint*.

El tutor apoya y orienta al estudiante durante todo el desarrollo del Trabajo Fin de Grado, asegurando que se cumplan los estándares de calidad y que se alcancen los objetivos establecidos.

Comunidad

En ocasiones encontramos que pueden surgir colaboraciones durante el desarrollo del TFG en el ámbito de un grupo de investigación, como puede ser la ayuda de profesores, alumnos o expertos.

La comunidad participa en los eventos de comunicación de progresos y es aquí donde pueden dar un *feedback* al estudiante.

Este grupo no tiene unas responsabilidades definidas pero son responsables de participar con el objetivo de crear un entorno crítico y constructivo que ayude al estudiante proporcionándole distintos puntos de vista respecto al producto y oportunidades de mejorarlo.

Tribunal

Este rol corresponde a los profesores que realizan la evaluación del TFG. En el ámbito de la empresa serían el cliente, y mediante la presentación realizada y la documentación entregada por el estudiante evaluarán de forma objetiva el producto entregado.

2.1.2. Eventos

La metodología ASAP requiere la realización de eventos para el desarrollo del TFG y mantenimiento de una interacción continua entre el estudiante y el tutor. Además, facilitan el establecimiento de compromisos por ambas partes y evitar bloqueos por parte del estudiante.

Estos eventos descritos a continuación están inspirados en los eventos de *Scrum*, aunque adaptados al desarrollo del TFG.

Sprint

Es el bloque organizativo principal de ASAP. Para el desarrollo del Trabajo Fin de Grado se divide su desarrollo en cuatro *sprints* de una duración similar (aproximadamente de un mes) comparable a los *sprint* de otras metodologías ágiles.

Cada *sprint* tendrá unos objetivos específicos definidos por las historias de usuario que se abordarán en este período. Estas historias representan hitos que se integran en el producto del desarrollo, añadiendo capas de funcionalidad en cada *Sprint*. La planificación del *sprint* incluye el resto de eventos que nombraremos, es decir, todas las demás reuniones están dentro del contexto de cada *sprint*.

La organización por *sprints* se basa en reducir la incertidumbre del trabajo al tener la capacidad de proponer objetivos más pequeños a corto plazo, permitiendo una construcción incremental del producto.

Al final de cada *sprint* el estudiante obtiene una retroalimentación que aprovecha para mejorar y revisar el proyecto. Es positivo que la duración de cada *sprint* sea similar para tener una frecuencia de retroalimentación aproximadamente constante.

Reunión de inicio

Se trata del primer evento del *sprint* y en él se establecen los objetivos del *sprint* y las historias de usuario necesarios para cumplirlos. Para ello se tiene en cuenta el *feedback* recibido el *sprint* anterior. El estudiante y el profesor pueden acordar cambios en el alcance durante esta reunión. Tras definir el alcance el estudiante deberá establecer las tareas necesarias para alcanzar los objetivos.

Se realiza el primer día del *sprint* y tiene una duración de media hora. En este proyecto se hacía el primer lunes del *sprint* por la mañana.

Reunión de sincronización

Se realiza semanalmente y tiene una duración de quince minutos. En este proyecto se ha realizado los lunes a las 10 de la mañana. En ella el estudiante le comunica al tutor los avances realizados desde la anterior reunión y las tareas que pretende llevar a cabo durante la semana hasta la siguiente reunión. La finalidad de esta reunión es llevar un seguimiento frecuente del progreso realizado, mantener una comunicación fluida y resolver bloqueos en un plazo breve de tiempo.

Reunión de comunicación de progresos

Se realiza al final del *sprint*. En esta reunión los estudiantes realizan una presentación del trabajo realizado hasta el momento. Todos los estudiantes en la reunión asisten a la presentación del resto y actúa también como miembro de la comunidad del proyecto de los compañeros. Esta presentación se debe asemejar al acto de defensa del TFG si estuviese completo el proyecto y al terminar se abre un turno de debate en el que participan todos los asistentes dando un *feedback* acerca del material expuesto.

Retrospectiva

Es el último evento del *sprint*. En la retrospectiva pueden participar todos los asistentes a la reunión de comunicación de progresos pues se realiza después de esta reunión.

Este evento pretende reflexionar acerca de los aspectos positivos y negativos observados en el *sprint* y propuestas de mejora. Para plasmar estos comentarios se utiliza un tablero *online* que todos los usuarios pueden ver actualizado para leer los comentarios del resto de forma anónima.

Al final de la retrospectiva se discuten las aportaciones de forma colectiva para dialogar acerca de la necesidad de hacer cambios en el siguiente *sprint*.

2.1.3. Artefactos

ASAP propone dos artefactos para el seguimiento de los avances del TFG.

Incremento

Basado en los incrementos de la metodología **Scrum** con el mismo nombre. Se trata del trabajo realizado hasta la finalización del *sprint*. Los incrementos son aditivos a los incrementos anteriores y van en línea con los objetivos del proyecto.

Este incremento es revisado por el tutor y a partir de este incremento el profesor aporta un *feedback*.

Retroalimentación

Este otro artefacto consolida el *feedback* que el estudiante ha recibido al finalizar cada *sprint*. Se trata de un documento que incorpora la retroalimentación que aporta la comunidad en la reunión de comunicación de progresos y la generada por el tutor tras una revisión del incremento consolidado en el *sprint*

2.1.4. Entorno de trabajo

Para trabajar con la metodología ASAP requerimos de un entorno tecnológico que incorpore:

Espacio de trabajo compartido

Este espacio de trabajo compartido alojará las versiones actualizadas del producto generado por el estudiante. El tutor también tiene acceso a este repositorio y puede ver, modificar y añadir documentos igual que el alumno.

Para el desarrollo de este TFG se ha utilizado *Microsoft Teams*, y la organización ha consistido en un canal privado en el que están el tutor y el alumno y otro público en el que está también la comunidad. Dentro de cada canal hay unos ficheros, los del canal público son información más general para la comunidad como información general del TFG y en el canal privado se guardan y comparten detalles específicos acerca del desarrollo del TFG con carpetas para la documentación, para los incrementos, para guardar ficheros personales o referencias, además del propio cuaderno de trabajo actualizado.

Tablero del proyecto

Es un tablero **Kanban** [11], que funciona como herramienta visual para organizar y hacer seguimiento de las tareas del proyecto y en particular del *sprint*. En la metodología ASAP, este tablero es clave para poder trabajar de manera ágil e incremental, ya que permite ver de forma sencilla en qué estado se encuentra cada tarea en todo momento.

En este proyecto se ha utilizado *Trello* como herramienta para crear el tablero. La estructura utilizada ha sido la habitual en ASAP, con listas que representan las distintas fases por las que puede pasar una tarea:

- **Backlog del proyecto:** Tareas generales que todavía no se han planificado para un *sprint* concreto. Es el conjunto de tareas pendientes a largo plazo.
- **Objetivo del *sprint*:** Tareas que se han seleccionado para ser completadas durante el *sprint* actual. Son las prioridades a corto plazo.
- **#ToDo:** Tareas del *sprint* que están planificadas para empezar, pero todavía no se han iniciado.
- **#Doing:** Tareas en las que se está trabajando activamente en ese momento.
- **#Blocked:** Tareas que no pueden avanzar por algún impedimento y que requieren resolver un bloqueo para continuar.
- **#Done:** Tareas que ya se han completado y han sido validadas.

Cada tarea se organiza dentro de estas listas según el estado en el que se encuentra, lo que facilita la gestión del trabajo, la planificación de los *sprints* y la detección de posibles bloqueos.

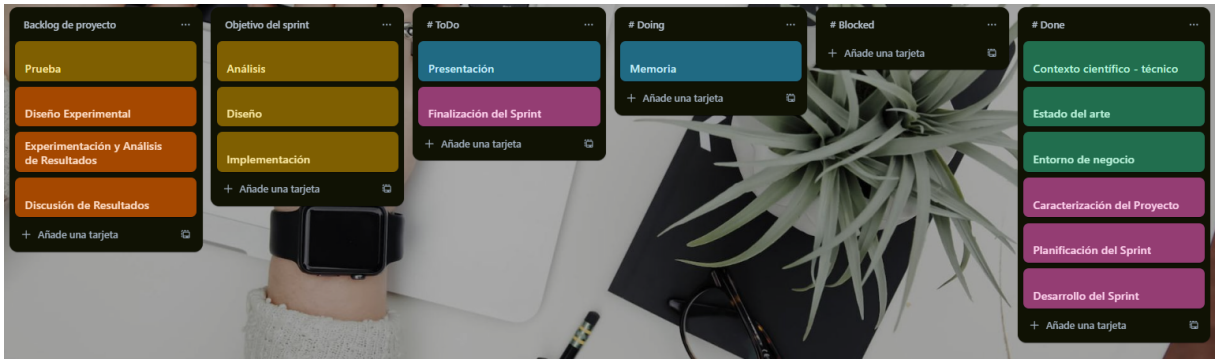

Figura 2.1: Tablero de Trello

Cuaderno de trabajo

Es un documento donde el estudiante registra sus sesiones de trabajo indicando la fecha, el *sprint* al que corresponde, las horas invertidas, la tarea realizada y una breve descripción de esta.

Esta herramienta es esencial para ASAP, pues ayuda a tener un control de las horas dedicadas y del esfuerzo dedicado a cada tarea.

Se realiza una entrega de este cuaderno al final de cada *sprint* y la intención es que sea una rutina en la que el estudiante, tras una sesión de trabajo, apunte las actividades desarrolladas y el tiempo dedicado.

Alumno	Miguel Sánchez-Beato Díaz-Hellín		TFG	Generación automática de recursos de aprendizaje utilizando Large Language Models (LLM)

Fecha	Sprint	Actividad		Tiempo	Descripción
		Tipo	Tarea		
30/11/2023	Sprint #1	Dinámica de Trabajo	Preparar Incremento	0:45:00	Primera reunión del proyecto y del primer sprint
29/11/2023	Sprint #1	Lectura de Documentación		2:00:00	Búsqueda de información acerca de LLM's y finetuning
30/11/2023	Sprint #1	Lectura de Documentación		2:00:00	Búsqueda de información acerca de LLM's y finetuning
01/12/2023	Sprint #1	Lectura de Documentación		1:30:00	Búsqueda de información acerca de LLM's y finetuning
09/12/2023	Sprint #1	Lectura de Documentación		1:30:00	Búsqueda de información acerca de LLM's y finetuning
11/12/2023	Sprint #1	Lectura de Documentación		2:00:00	Búsqueda de información acerca de LLM's y finetuning
13/12/2023	Sprint #1	Dinámica de Trabajo	Reunión Semanal	0:15:00	Reunión de sincronización
19/12/2023	Sprint #1	Dinámica de Trabajo	Reunión Semanal	0:15:00	Reunión de sincronización
25/12/2023	Sprint #1	Lectura de Documentación		1:30:00	Lectura acerca de Mistral 7-b, su finetuning y de capacidad de gpu necesaria
26/12/2023	Sprint #1	Redacción de Memoria		2:00:00	Empezar a trabajar con overleaf y empezar apartados 1 y 3 de la memoria
26/12/2023	Sprint #1	Programación		3:00:00	Descubrir cómo ejecutar y probar a ejecutar LLM's de HuggingFace en google collab
29/12/2023	Sprint #1	Programación		3:00:00	Buscar información de lo de Google Collab y probar a ejecutar en VisualStudio y anaconda con notebooks
30/12/2023	Sprint #1	Redacción de Memoria		1:45:00	Inicio de la redacción del apartado 3 y búsqueda de información y estado del arte.
30/12/2023	Sprint #1	Lectura de Documentación		1:00:00	Buscar modelos de 5b y 7b a ejecutar y ver que hay métodos para simplificar la ejecución de estos
06/01/2024	Sprint #1	Lectura de Documentación		2:00:00	Buscar cómo ejecutar modelos más grandes con menos RAM y GPU
06/01/2024	Sprint #1	Redacción de Memoria		0:45:00	Continuar con la redacción del apartado 3
16/01/2024	Sprint #1	Dinámica de Trabajo	Reunión Semanal	0:15:00	Sincronización
17/01/2024	Sprint #1	Lectura de Documentación		2:00:00	Empezar a trabajar con llama.cpp y leer documentación de dockers
17/01/2024	Sprint #1	Otros		2:30:00	Preparar presentación
18/01/2024	Sprint #1	Otros		1:30:00	Terminar toda la presentación y usar LM Studio
19/01/2024	Sprint #1	Dinámica de Trabajo	Reunión Fin Sprint	2:00:00	Presentaciones del primer sprint y tercer sprint de Clara
31/01/2024	Sprint #2	Dinámica de Trabajo	Preparar Incremento	0:20:00	Reuniónde inicio del segundo sprint
02/01/2024	Sprint #2	Programación		1:00:00	Empezar a utilizar docker desktop en mi ordenador y buscar información de dockers con llama.cpp
03/01/2024	Sprint #2	Programación		2:30:00	Descargar modelo, descargar un docker de github e intentar trabajar con ellos
04/01/2024	Sprint #2	Otros		1:00:00	Realizar la comunicación de inicio
05/02/2024	Sprint #2	Dinámica de Trabajo	Reunión Semanal	0:15:00	Sincronización

Figura 2.2: Cuaderno de trabajo

2.2. Planificación temporal

La planificación inicial del proyecto era de cuatro *sprints*, pero por situaciones académicas han sido cinco, siendo el quinto en el curso siguiente al del inicio del Trabajo Fin de Grado.

La duración de los *sprints* ha sido variable y las horas invertidas en cada uno también dependiendo de mi disponibilidad, teniendo en cuenta otras asignaturas.

La planificación de cada *sprint* se ha realizado al inicio del mismo. Semanalmente los lunes se ha realizado una reunión de sincronización de unos quince minutos. Al final de cada *sprint* se realiza una reunión en la que se exponen los avances y se realiza una presentación del TFG para hacer que esta sea también incremental.

- Primer *sprint*: 4 de Diciembre 2023 - 19 de enero de 2024.

  En este primer *sprint* se realiza una primera aproximación al proyecto, con reuniones muy importantes para terminar de definir el resultado esperado del proyecto. Hay una parte de este primer *sprint* que se dedica a la investigación, documentación y redacción de memoria, en especial del estado del arte, y otra parte más práctica en el que se empieza a probar LLM's en local. Las tareas llevadas a cabo en este *sprint* se pueden ver de forma esquemática en la Figura 2.3 y su planificación temporal en el diagrama de Gantt de la Figura 2.4.

- Segundo *sprint*: 22 de enero de 2024 - 23 de febrero de 2024. Este segundo *sprint* tendrá una organización parecida al primero debido a las pocas horas disponibles en el primer *sprint*. Reuniones para discutir las posibilidades disponibles en cuanto a los LLM y *dockers*, corrección de errores en la memoria y redacción de nuevos capítulos son la parte más teórica de este *sprint*, y en la parte más práctica se realizan unos primeros intentos de ejecución con *dockers* y se prueban nuevos modelos LLM con Hugging Face y Google Collab. En la Figura 2.5 se presenta un esquema de las tareas y en la Figura 2.6 su correspondiente diagrama de Gantt.

- Tercer *sprint*: 26 de febrero - 12 de abril de 2024. En el tercer *sprint* tratamos ya de tener un backend con la lógica correspondiente al LLM ejecutándose en un *Docker* y respondiendo preguntas que se guardan en la base de datos que también creamos en este *sprint*. Este *sprint* está más orientado a la práctica que a la memoria y documentación. El esquema de tareas y el diagrama de Gantt de este *sprint* se pueden consultar en las Figuras 2.7 y 2.8, respectivamente.

- Cuarto *sprint*: 15 de abril - 12 de mayo de 2024. Al inicio del cuarto *sprint* aparece el nuevo modelo Llama 3, y como es la parte más innovadora y valiosa del trabajo, consideramos necesaria la migración a Llama 3 y hacer un *fine-tuning* con un dataset que contenga materiales específicos de la asignatura "Sistemas de Bases de Datos". También se realizan en este *sprint* los diagramas restantes de la aplicación y se revisan. Una visualización detallada de las tareas y su planificación se pueden ver en la Figura 2.9 y la Figura 2.10.

- Quinto *sprint*: 21 de Junio de 2025 - 7 de Julio de 2025. Este último *sprint* tiene una labor de finalizar todo lo que está en proceso y crear la página web con la que interactuarán los usuarios. La redacción y corrección de la memoria y la página web son los dos aspectos más importantes tratados en este quinto *sprint*. El esquema de las tareas realizadas y su

planificación temporal se muestran en el esquema de la Figura 2.11 y en el diagrama de Gantt de la Figura 2.12.

Primer sprint

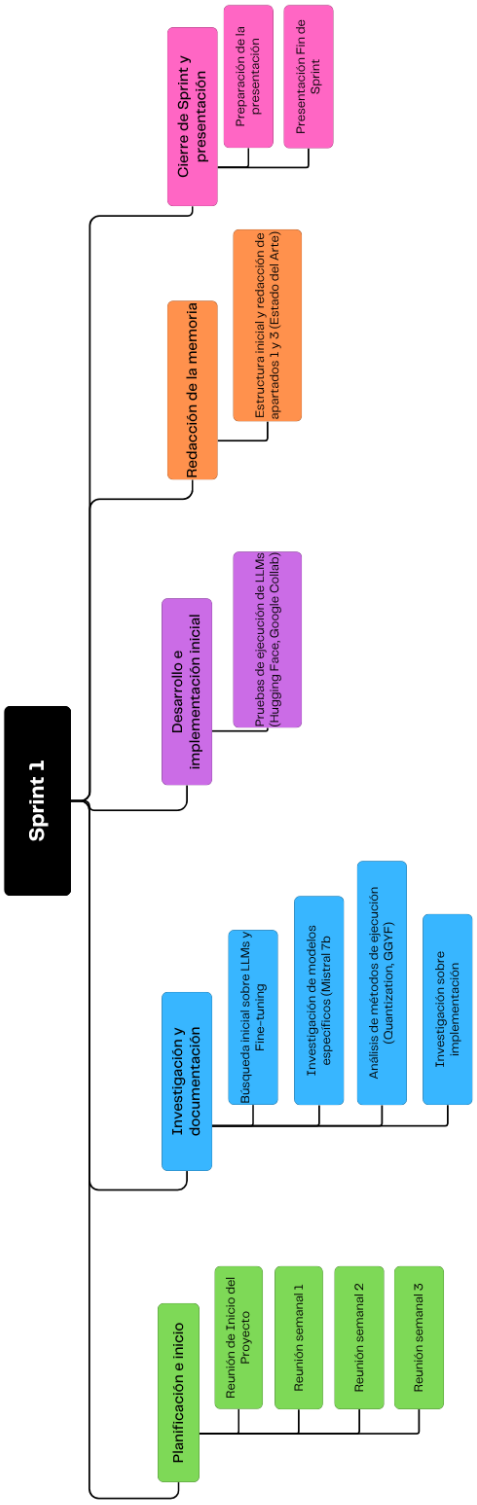

Figura 2.3: Esquema de las tareas del primer *sprint*.

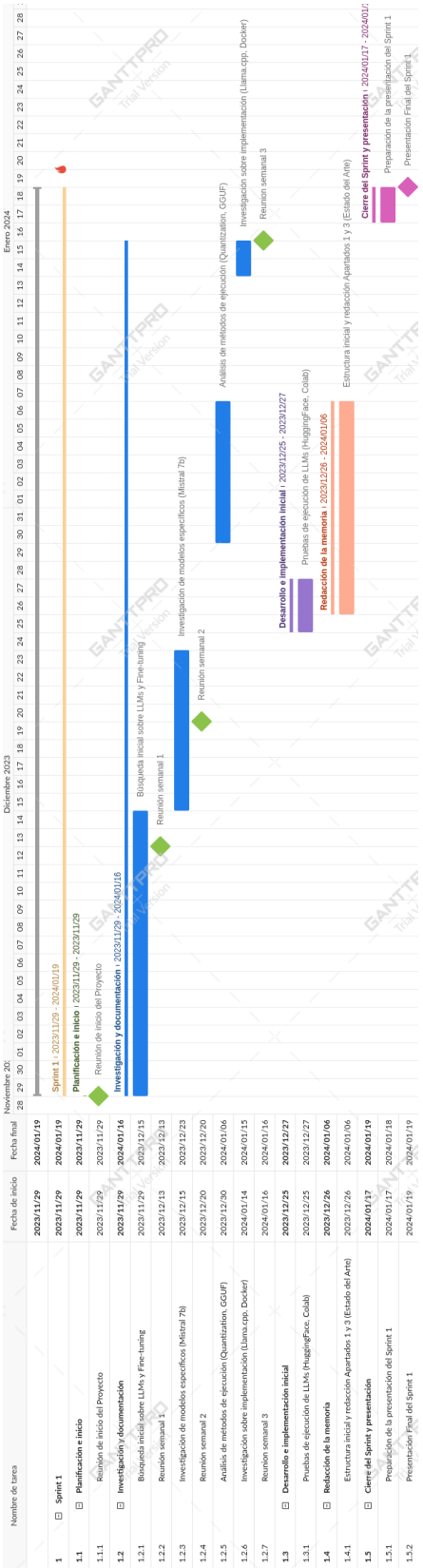

Figura 2.4: Diagrama de Gantt del primer *sprint*.

Segundo sprint

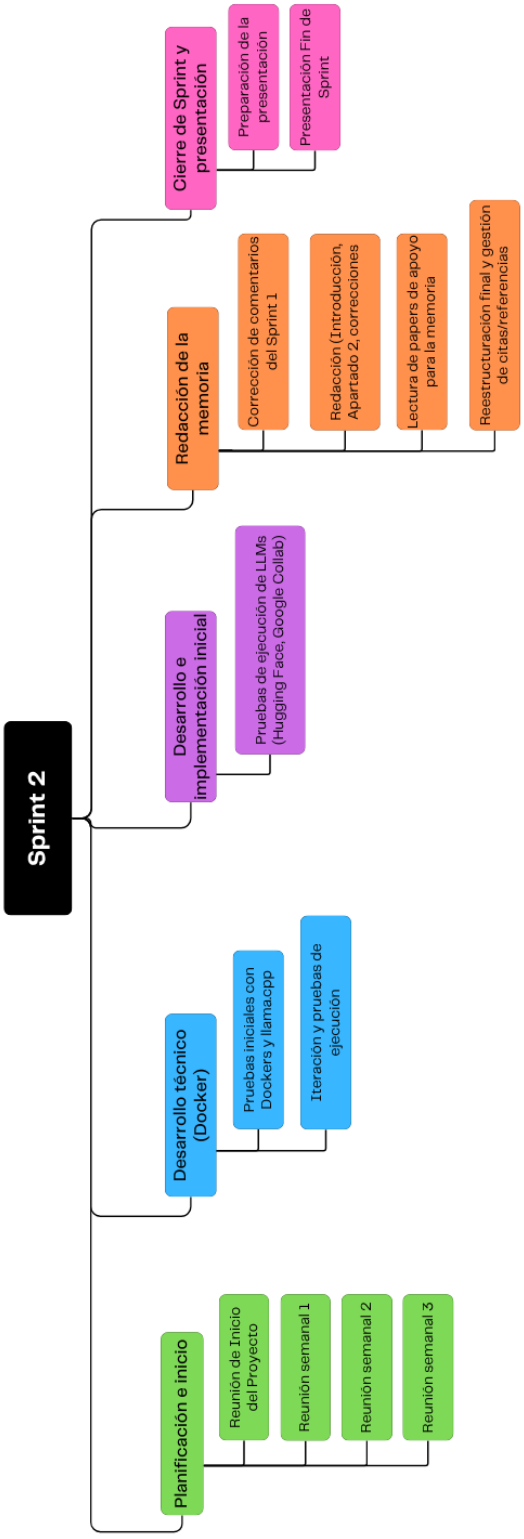

Figura 2.5: Esquema de las tareas del segundo *sprint*.

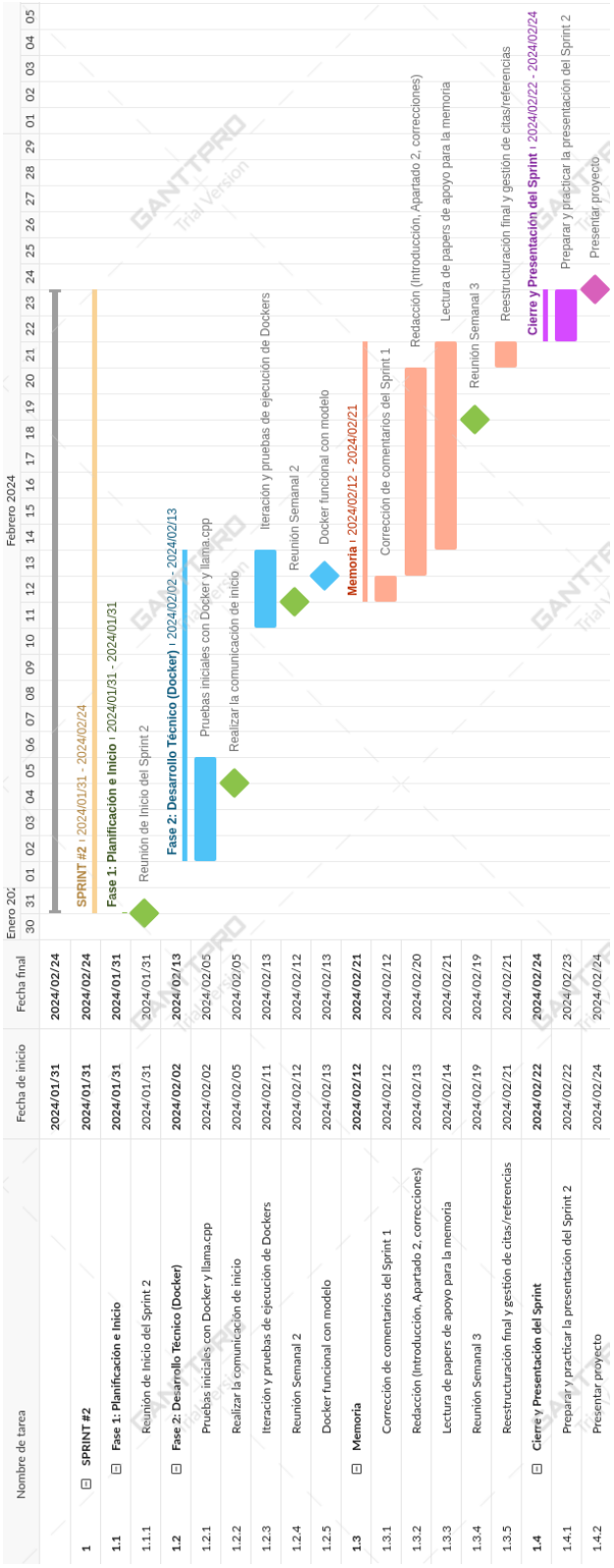

Figura 2.6: Diagrama de Gantt del segundo *sprint*.

Tercer sprint

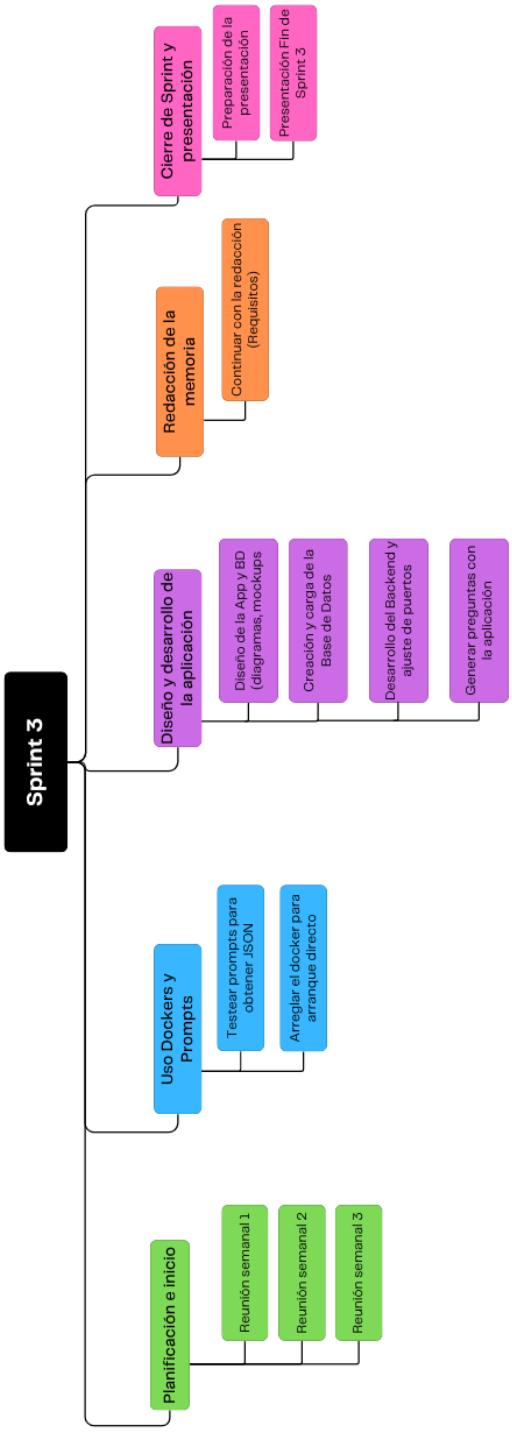

Figura 2.7: Esquema de las tareas del tercer *sprint*.

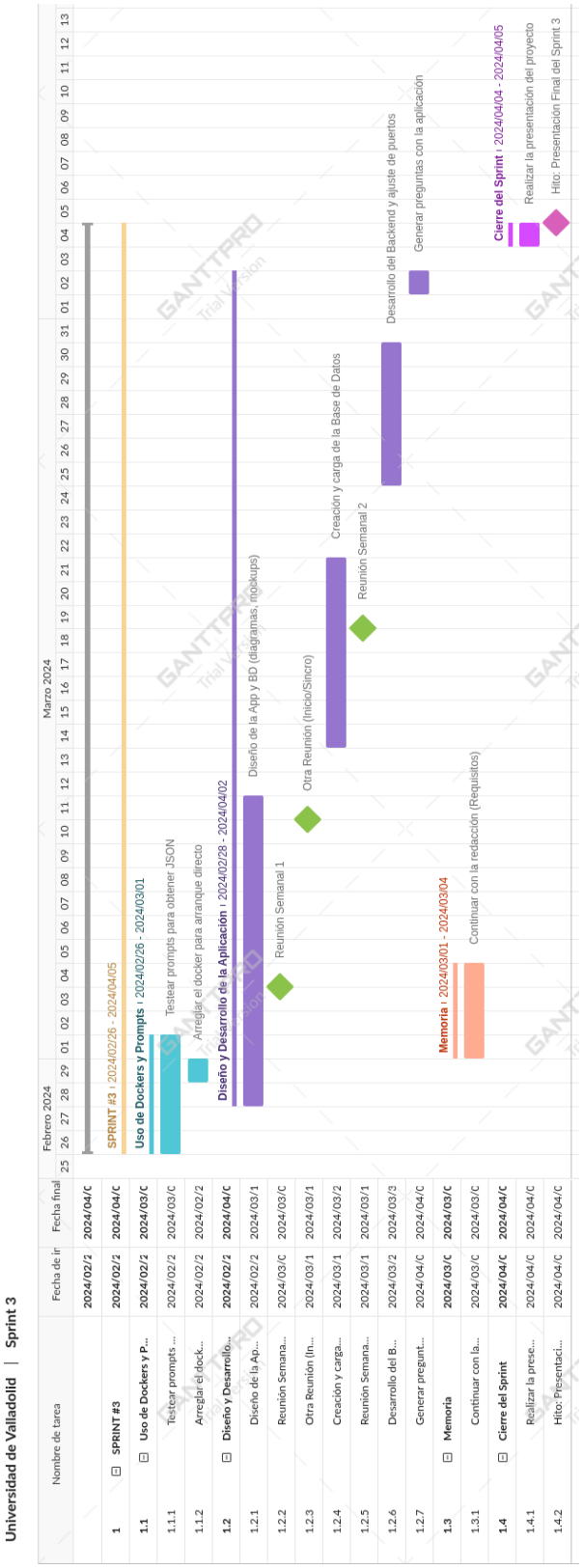

Figura 2.8: Diagrama de Gantt del tercer *sprint*.

Cuarto sprint

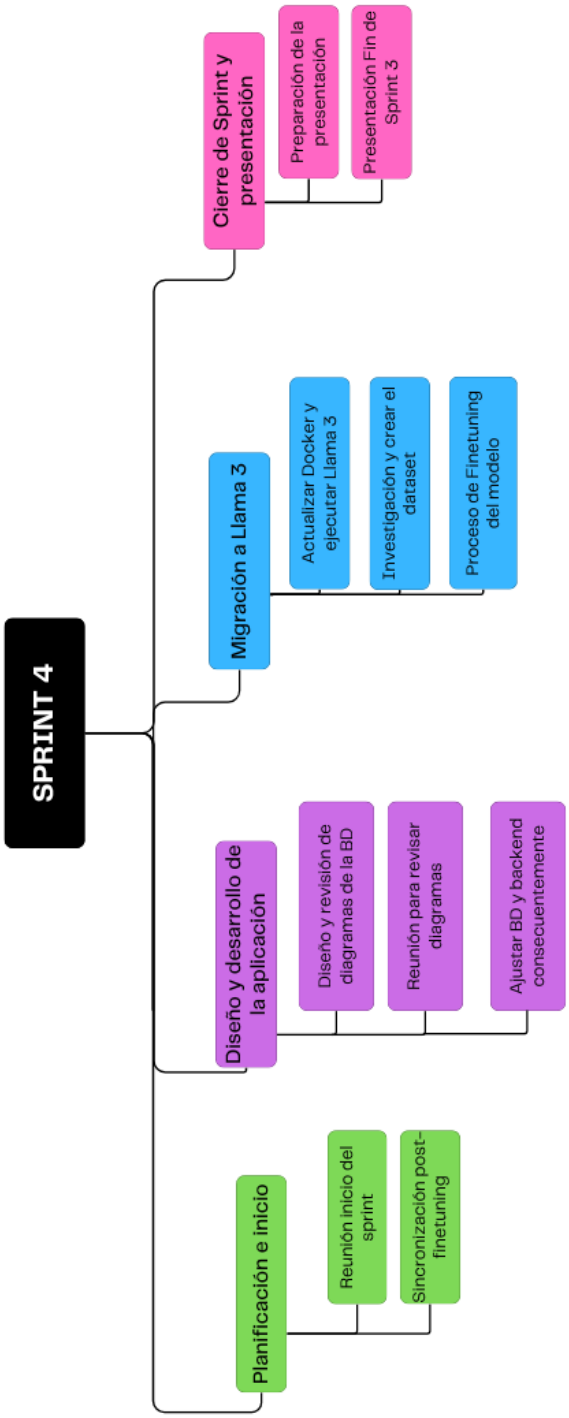

Figura 2.9: Esquema de las tareas del cuarto *sprint*.

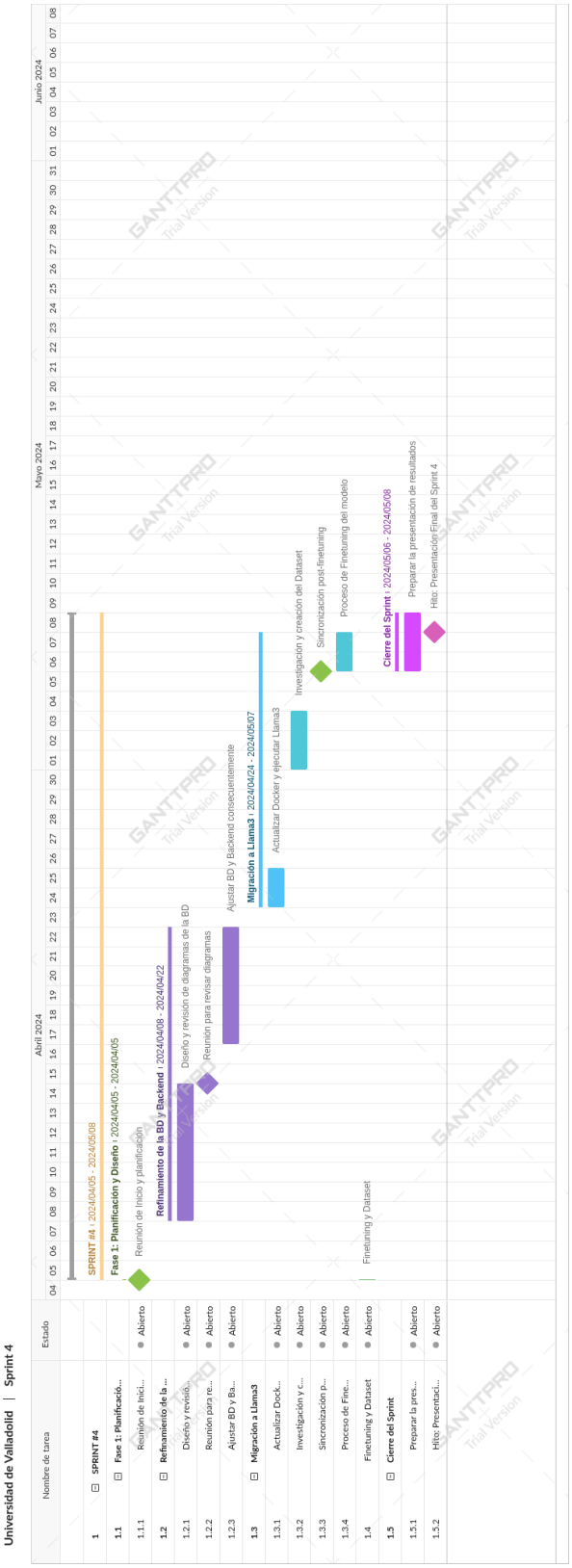

Figura 2.10: Diagrama de Gantt del cuarto *sprint*.

Quinto sprint

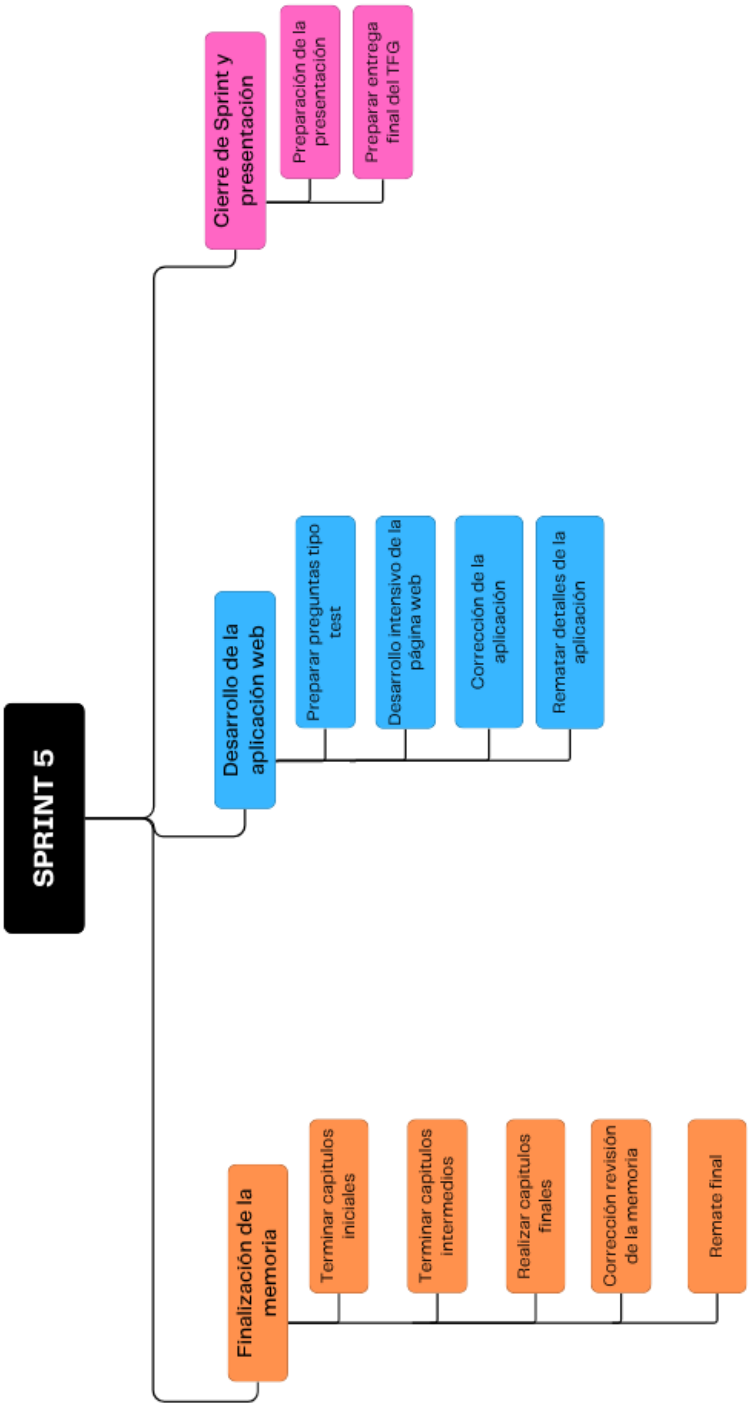

Figura 2.11: Esquema de las tareas del quinto *sprint*.

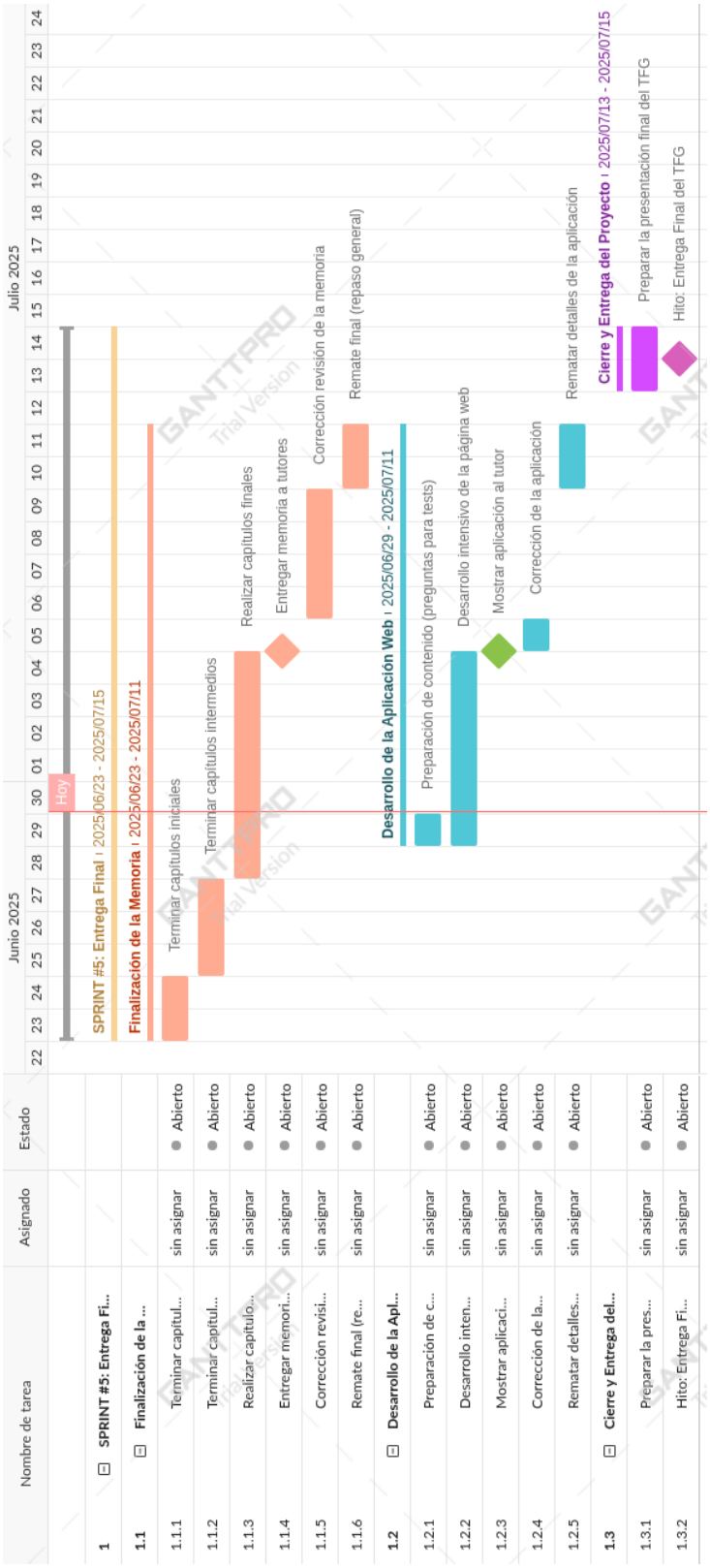

Figura 2.12: Diagrama de Gantt del quinto *sprint*.

2.3. Presupuesto

En esta sección se presenta un análisis detallado del presupuesto necesario para el desarrollo del Trabajo Fin de Grado. El presupuesto es un elemento importante dentro de la planificación del proyecto, ya que permite estimar los recursos que serán necesarios para alcanzar los objetivos planteados. En este caso, se detallan los costes asociados al *hardware*, *software* y recursos humanos.

Hardware

El recurso *hardware* empleado ha sido un ordenador personal con procesador i7 de octava generación, 16GB de memoria RAM y disco SSD, suficiente para ejecutar la aplicación localmente con el modelo Llama.cpp a través de contenedores en Docker Desktop. La conexión a internet se ha realizado mediante red Wi-Fi personal. En caso de querer desplegar la aplicación en un servidor accesible por otros usuarios, se ha estimado el coste de un servidor básico alojado en la nube.

Con respecto al ordenador portátil estimamos una la vida útil de este en cinco -seis años, por lo que la amortización del ordenador sería de aproximadamente un 20 % del coste del ordenador.

El resumen de los costes de *hardware* se muestra en la Tabla 2.1.

Componente	**Coste**	**% uso**	**Coste total**
Ordenador personal	900 €	20 %	180 €
Conexión a internet	25 €/mes	6 meses	150 €
Total			**330 €**

Tabla 2.1: Costes hardware

Software

Todo el *software* utilizado en el proyecto ha sido de libre acceso o de licencia gratuita. Las herramientas utilizadas han sido las siguientes:

- Visual Studio Code para el desarrollo de la aplicación.
- Docker Desktop para la virtualización de contenedores.
- Trello y Microsoft Teams como soporte de la metodología ASAP.
- Overleaf para la redacción de la memoria.
- Draw.io para la elaboración de diagramas.
- GanttPro para la elaboración de diagramas de Gantt.
- Hugging Face para obtener el modelo LLM.
- Windows 11, que no es gratuito pero viene incluido en el precio del ordenador.

- Google Collab en su versión gratuita para realizar pruebas con LLMs y para realizar el *fine-tuning*.
- LLM Studio para el inicio del proyecto para ejecutar en local distintos LLMs.
- Postman para realizar solicitudes HTTP para probar el *backend* y sus *endpoints*.

El resumen de los costes de *software* se muestra en la Tabla 2.2.

Componente	**Coste total**
Visual Studio Code	0 €
Docker Desktop	0 €
Trello	0 €
Microsoft Teams	0 €
Overleaf	0 €
Draw.io	0 €
GanttPro	0 €
Hugging Face	0 €
Windows 11	0 €
Google Collab	0 €
LLM Studio	0 €
Postman	0 €
Total	**0 €**

Tabla 2.2: Costes software

Recursos Humanos

En el proyecto se ha supuesto una dedicación total aproximada de 300 horas, divididas en distintos perfiles que el propio estudiante ha asumido. Para estimar los costes humanos, se ha considerado el sueldo de cada perfil y el tiempo invertido en las tareas que le corresponderían

El desglose de las horas y los perfiles se muestra en la tabla 2.3.

Puesto	**Salario (€/hora)**	**Horas**	**Salario total**
Gestor de proyecto [23]	11,50	120	1.380,00 €
Programador *Full Stack Junior* [17]	10,00	120	1.200,00 €
Ingeniero IA *junior* [18]	13,00	60	780,00 €
Total		300	**3.360,00 €**

Tabla 2.3: Costes de Recursos Humanos

Además, se debe añadir el coste asociado a la Seguridad Social, que estimo en un 30 % sobre el salario bruto, según los datos consultados en la página de la Seguridad Social [46].

El coste total de los recursos humanos, incluyendo la Seguridad Social, se calcula como:

$$\begin{aligned} C_{\text{RRHH}} &= \text{Coste bruto} + \text{Seguridad Social} \\ &= 3{,}360{,}00\ € + 0{,}30 \times 3{,}360{,}00\ € \\ &= 3{,}360{,}00\ € + 1{,}008{,}00\ € \\ &= \boxed{4{,}368{,}00\ €} \end{aligned}$$

Costes de despliegue (opcional)

En caso de querer desplegar la aplicación en un servidor accesible por Internet, se considerarían los siguientes costes opcionales:

- Servidor en la nube: **30 €** mensuales.
- Dominio web: **15 €** anuales.

Coste total

No computamos los costes de despliegue, porque este sería un gasto recurrente que se generará mensualmente después de finalizar la aplicación en caso de desplegarla.

$$\begin{aligned} C_{\text{total}} &= C_{\text{hardware}} + C_{\text{software}} + C_{\text{RRHH}} \\ &= 330{,}00\ € + 0{,}00\ € + 4{,}368{,}00\ € \\ &= \boxed{4{,}698{,}00\ €} \end{aligned}$$

2.4. Gestión de riesgos

En todo proyecto es importante identificar, analizar y planificar posibles riesgos que puedan afectar a su correcto desarrollo. La gestión de riesgos permite anticiparse a posibles problemas y tener planes de contingencia para minimizar su impacto en caso de que ocurran. Se valora cual es la probabilidad de que ocurran y el impacto que tendrían en el desarrollo del proyecto para cuantificar la necesidad de un plan de contingencia. [48]

2.4.1. Gestión de riesgos

A continuación, se detallan los principales riesgos identificados en este Trabajo Fin de Grado:

Riesgo	**Descripción**
R-01	Falta de tiempo para finalizar el desarrollo del proyecto.
R-02	Problemas técnicos con la ejecución de modelos Llama.cpp en *docker*.
R-03	Dificultad para generar preguntas relevantes a través del LLM.
R-04	Desorganización en la gestión de tareas.
R-05	Pérdida de información o versiones desactualizadas.
R-06	Problemas en la comunicación con el tutor.
R-07	Riesgo de que salgan nuevas actualizaciones de LLM y librerias utilizadas que conlleven incompatibilidades y obsolescencia.

Tabla 2.4: Riesgos identificados

Riesgo	**Probabilidad**	**Valor**	**Justificación**
R-01	45 %	3	Riesgo de que surjan otros compromisos y falta de tiempo.
R-02	70 %	4	Los modelos Llama.cpp en *docker* pueden fallar por limitaciones técnicas.
R-03	60 %	4	Puede ser difícil obtener preguntas útiles si se usan prompts no óptimos.
R-04	20 %	2	Riesgo bajo debido al uso constante de herramientas de gestión.
R-05	12 %	2	Riesgo bajo debido a las copias de seguridad y buenas prácticas.
R-06	6 %	1	Riesgo bajo gracias a la rutina de reuniones establecida.
R-07	95 %	5	Riesgo muy alto porque los LLM están en pleno auge y se realizan avances muy importantes constantemente, dejando versiones anteriores obsoletas.

Tabla 2.5: Probabilidad de ocurrencia de cada riesgo

Consideramos una jornada de trabajo de 8 horas y un sueldo por hora promedio de 11,50€ según los perfiles ya considerados.

Riesgo	**Pérdida operacional**	**Impacto en los costes (%)**	**Impacto (1-5)**	**Justificación**
R-01	92 €/día durante 3 días = 276 €	5,88 %	3	Puede provocar retrasos en la entrega final del proyecto, lo que supondría días adicionales de trabajo para ponerse al día con el proyecto.
R-02	Modificación de equipo: 500 €	10,64 %	4	Puede impedir la correcta ejecución de modelos básicos, lo que requeriría reponer o actualizar el hardware.
R-03	92 €/día durante 2 días = 160 €	3,92 %	2	Las preguntas deben de ser útiles y hacer pruebas extra podría suponer un par de días extra de trabajo.
R-04	92 €/día durante 3 días = 276 €	5,88 %	3	Puede generar ineficiencias y pérdidas de tiempo que se podrían solucionar invirtiendo un poco de tiempo en cumplir las tareas que conlleva la metodología ASAP.
R-05	92 €/día durante 5 días = 460 €	9,79 %	3	La pérdida de información crítica podría obligar a rehacer parte del trabajo, suponiendo varios días de esfuerzo adicional.
R-06	92 €/día durante 3 días = 276 €	3,92 %	2	Puede provocar descoordinación, problemas de comunicación y bloqueos en el avance del proyecto, con necesidad de reuniones y reorganización para retomar el ritmo.
R-07	92 €/día durante 5 días = 460 €	9,79 %	3	Puede quedarse obsoleto el trabajo en cuestión de unos meses y necesitaríamos actualizar una o varias partes del proyecto.

Tabla 2.6: Impacto económico de los riesgos

2.4.2. Matriz de probabilidad x Impacto

Una vez tenemos la probabilidad y el impacto, calculamos la exposición, que se obtiene a partir de la fórmula

$$\text{Exposición} = \text{Probabilidad} \times \text{Impacto}$$

Clasificamos la prioridad según el valor de exposición:

- Prioridad alta: 10 <Exposición

- Prioridad media: 5 <Exposición <= 10

- Prioridad baja: Exposición <= 5

Riesgo	**Probabilidad**	**Impacto**	**Exposición**
R-01	3	5	15
R-02	4	4	16
R-03	4	2	8
R-04	2	3	6
R-05	2	3	6
R-06	1	2	2
R-07	5	3	15

Tabla 2.7: Matriz de probabilidad e impacto con prioridades coloreadas

2.4.3. Plan de contingencia

Según la prioridad de la tabla anterior (de mayor a menor) creamos un plan de contingencia:

Riesgo	**Exposición**	**Plan de contingencia**
R-02	16	Realizar pruebas tempranas y buscar soluciones alternativas como modelos más ligeros o ejecución en servidores externos.
R-01	15	Planificar correctamente los *sprints* y priorizar las tareas más importantes para cumplir los objetivos mínimos.
R-07	15	Estar atentos a las actualizaciones previstas para Llama y fijar la versión de las librerías utilizadas en el *docker* para evitar que con las actualizaciones deje de funcionar el código.
R-03	8	Ajustar los *prompts* y realizar varias iteraciones de prueba para mejorar la calidad de las preguntas generadas.
R-04	6	Utilizar de forma constante el tablero del proyecto en Trello y realizar reuniones de sincronización periódicas.
R-05	6	Utilizar un espacio de trabajo compartido bien estructurado en Microsoft Teams y realizar copias de seguridad.
R-06	2	Mantener la rutina de reuniones semanales y comunicarse de forma ágil mediante el canal privado de Teams.

Tabla 2.8: Plan de contingencia por prioridad

2.5. Balance temporal y económico

En esta sección se presenta el balance temporal y económico del proyecto, en el que se compara el tiempo inicialmente estimado con el tiempo realmente dedicado, así como el coste asociado a cada tarea según el esfuerzo invertido.

Balance económico

A partir del tiempo real dedicado a cada tarea y considerando el sueldo anteriormente estimado, se ha calculado el coste económico real del proyecto. Las tareas de Estudio del estado del arte y lectura de artículos, el tiempo de reuniones y la redacción de memoria corresponde al gestor de proyecto, que como ya indicamos, estimamos un sueldo de 11,50€. Las tareas del diseño de la aplicación y del desarrollo del *backend* y *frontend* corresponden al programador *full stack* que estimamos un sueldo de 10€ al ser junior. Las tareas relacionadas con el modelo LLM corresponden al ingeniero en IA, que tiene un sueldo de 13€ al ser también junior.

Tarea	**Horas reales**	**Coste (€)**
Estudio del estado del arte y lectura de artículos	27,17 h	312,42 €
Diseño de la aplicación (diagramas, arquitectura, BD)	33,75 h	337,50 €
Desarrollo del *backend* y *frontend*	74,58 h	745,83 €
Tareas relacionadas con el modelo LLM (*docker*, *prompts*, implementación)	61,50 h	799,50 €
Pruebas y validación	8,25 h	89,25 €
Redacción de la memoria	86,83 h	998,58 €
Total	**341 h**	**3.855,21 €**

Tabla 2.9: Balance económico del proyecto

Y con el 30 % extra por el pago de la seguridad social tenemos que el coste real de recursos humanos ha sido de:

$$\begin{aligned} CR_{\text{RRHH}} &= \text{Coste bruto} + \text{Seguridad Social} \\ &= 3{,}855{,}21\ € + 0{,}30 \times 3{,}855{,}21\ € \\ &= \boxed{5{,}011{,}77\ €} \end{aligned}$$

El coste real de los recursos humanos del proyecto ha resultado ser de 5.011,77€, teniendo en cuenta las horas dedicadas a cada tarea del proyecto.

Ha habido un sobrecosto de 643,77€, y el coste real total ha sido de 5341,77€, pues en *hardware* y *software* el gasto ha sido el presupuestado.

Capítulo 3

Antecedentes

3.1. Entorno de negocio

El entorno de negocio en el que se encuentra este proyecto es el ámbito educativo, tanto instituciones académicas como empresas o plataformas de formación en línea. La aplicación propuesta plantea una herramienta de aprendizaje que integra un LLM entrenado con datos seleccionados manualmente según el conocimiento específico deseado.

En la actualidad, no es habitual el uso inteligencias artificiales orientadas al sector educativo, aunque la evaluación por preguntas tipo test sí es una práctica habitual. Esta situación plantea una gran oportunidad de negocio en el desarrollo de una aplicación que cumplan este fin utilizando un *Large Language Model*.

Podemos encontrar estudios acerca de la generación de preguntas para la educación, su utilidad y cómo plantear los *prompts* para obtener preguntas de máxima calidad. Por ejemplo, en *papers* como [14] [56] se profundiza en este tema.

Además, podemos encontrar estudios acerca de los potenciales beneficios de ChatGPT en la educación y el aprendizaje como es [5], de los cuales se consideran beneficios como la tutoría personalizada a las necesidades individuales de los estudiantes, la calificación automática de las entregas de los estudiantes, la traducción de materiales educativos a distintos idiomas, el aprendizaje interactivo mediante conversaciones con un tutor virtual o la capacidad de adaptar sus métodos de enseñanza según el progreso del estudiante para mejorar su rendimiento académico.

Nos encontramos una demanda creciente de herramientas que mejoren la evaluación del aprendizaje de forma interactiva y con una retroalimentación efectiva, todo esto centrado en una serie de objetivos de aprendizaje definidos.

3.2. Estado del arte

Se estudiarán las distintas herramientas existentes que combinan los Large Language Models (LLM) con el aprendizaje.

3.2.1. Descripción de trabajos relacionados

Khanmigo Education

La plataforma de aprendizaje en línea Khan Academy una de las más importantes a nivel mundial en el ámbito de las ciencias en general, ha creado un modelo de lenguaje en el proyecto Khanmigo Education. Este modelo de lenguaje está basado en GPT-4 y es un asistente virtual de pago al que se le pueden realizar consultas dentro de la plataforma. Este asistente virtual conoce el contexto de la lección en la que se encuentra el estudiante y los ejercicios propuestos y el estudiante le puede realizar preguntas para que le ayude.

No solamente le puede ayudar en los ejercicios, sino que le puede proponer ejercicios similares, enseñar a redactar ensayos, mantener debates académicos, actuar como un personaje histórico para que le entrevistes entre otras muchas acciones ya contempladas en la plataforma. Además, se le pueden hacer consultas de forma personalizada acerca del tema que te interese.

Este modelo de lenguaje también permite a los profesores realizar consultas, como puede ser cómo orientar una clase acerca de un determinado tema, y el asistente virtual realizará una serie de preguntas acerca de la clase y los estudiantes. Según las respuestas dadas, realizará una serie de sugerencias.

En resumen, es una herramienta de tutorización para estudiantes y profesores centrado en el aprendizaje y que acompaña durante el aprendizaje en la plataforma.

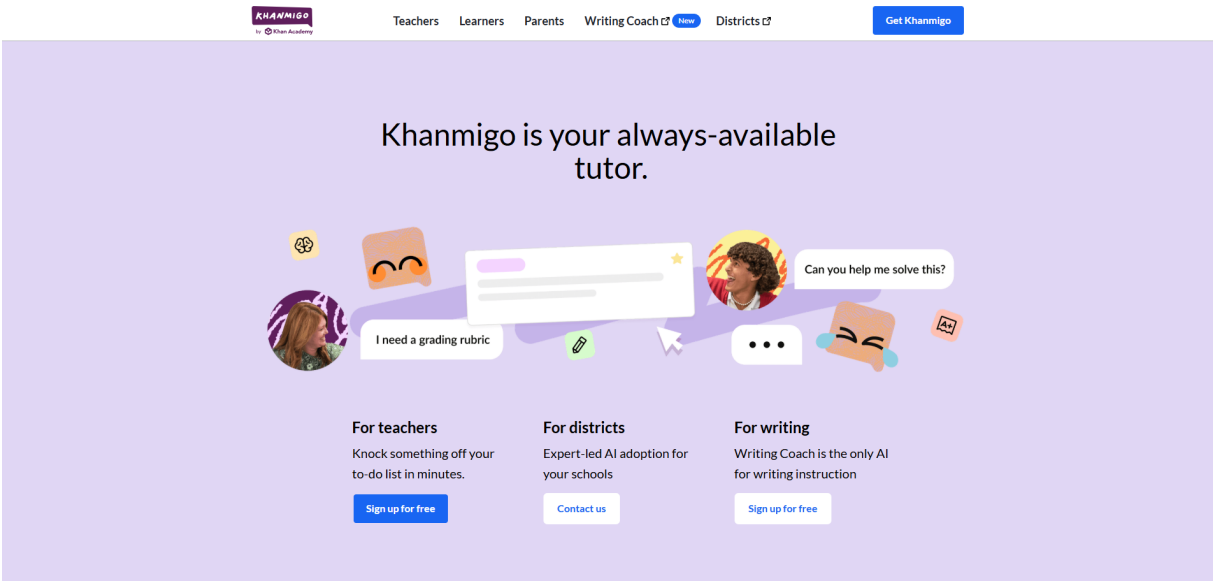

Figura 3.1: Página web de Khanmigo Education

Merlyn Mind

Otra herramienta de aprendizaje de estas características es Merlyn Mind. Esta herramienta se trata de una serie de *Large Language Models* (LLMs) de código abierto y cuyo entrenamiento está realizado por la empresa Merlyn Mind usando como modelo base pythia-12b. De momento tienen tres modelos que están pensados para recibir la información que ocurre en la clase en tiempo real. Uno de ellos está centrado en responder las preguntas que surgen durante el diálogo

en clase, otro clasifica las dudas o preguntas que se tratan en la conversación como adecuadas o inadecuadas en el contexto de la educación en las clases y el último realiza recomendaciones al profesor de actividades y temas interesantes acerca de la conversación que está ocurriendo en la clase.

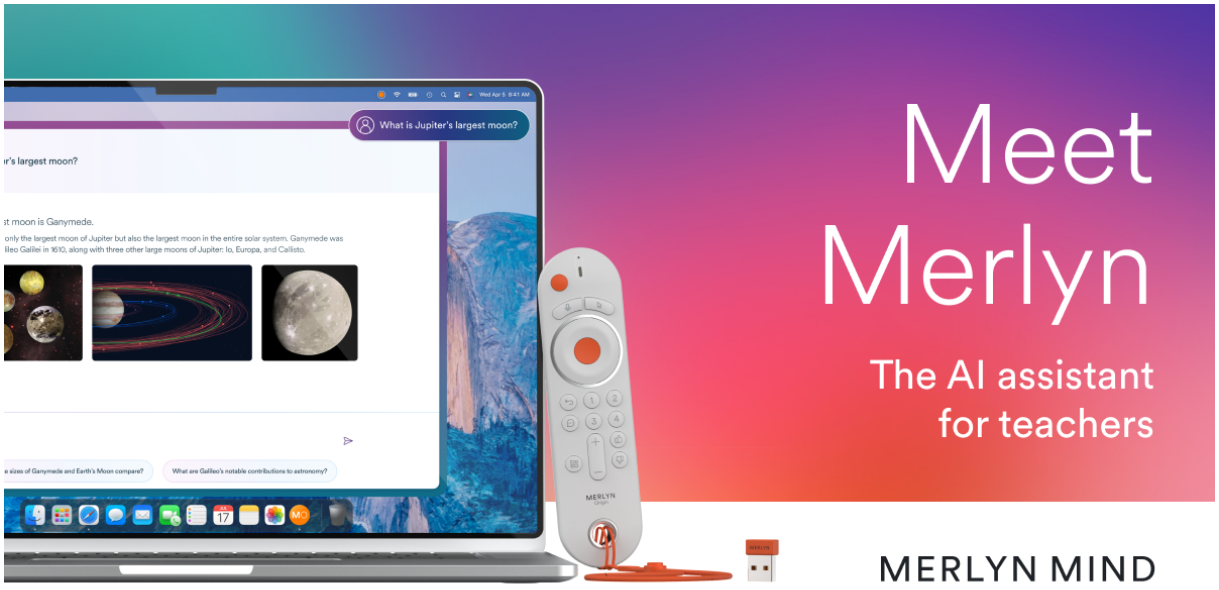

Figura 3.2: Página web de Merlyn Mind

Plataformas de generación de preguntas

Por último, existen varias plataformas web de pago que a partir de un texto dado, se pueden obtener distintos tipos de preguntas. De las cinco encontradas: Quizgecko, Questgen, Quillionz, Yippity y Prepai, las dos más completas y destacables son las dos primeras.

La más eficaz y completa es Quizgecko, la cual funciona muy bien en español, tiene pruebas gratuitas de forma sencilla, tiene preguntas de tipo test, verdadero/falso, rellenar huecos y de respuesta breve, y todas tres niveles de dificultad (Fácil, Medio y Difícil), permite exportar las preguntas en distintos formatos, puede trabajar URLs de documentos en línea y con videos de YouTube y nos permite realizar *flashcards* con las preguntas generadas entre otras funcionalidades.

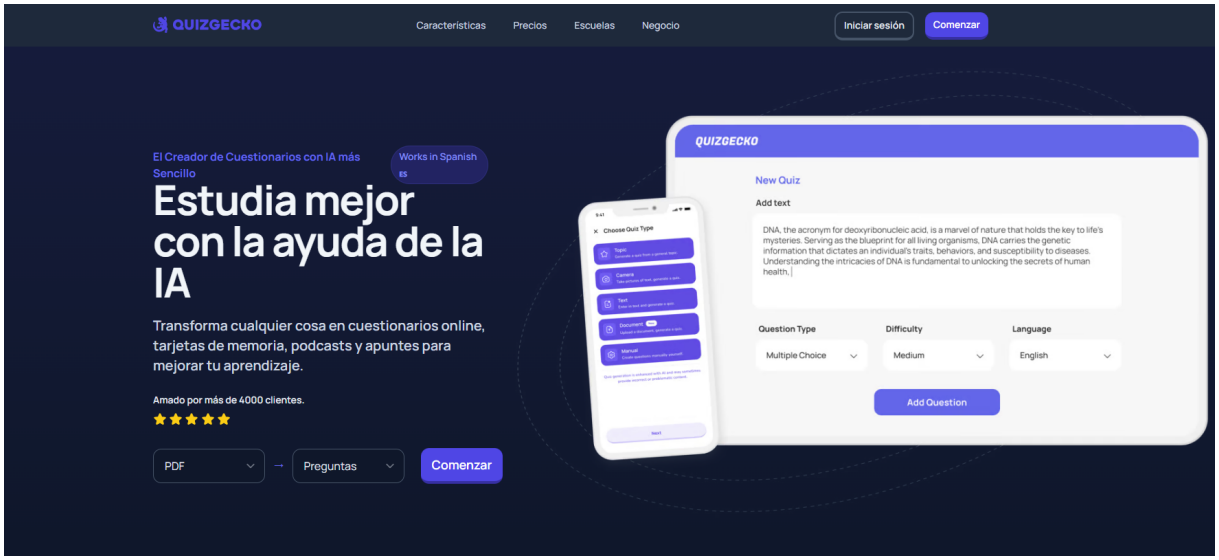

Figura 3.3: Página web de Quizgecko

Questgen es algo diferente, pues a partir del texto dado puede generar cinco tipos de contenido: Preguntas tipo test, preguntas verdadero/falso, de rellenar huecos, preguntas comunes (FAQ) y preguntas cortas. Además tiene dos características destacables. Una es la generación de preguntas similares a una pregunta dada, y la otra es generar preguntas que se focalicen en las diferentes categorías de la Taxonomía de Bloom. "La taxonomía de Bloom para el dominio cognitivo es la clasificación de objetivos más usada y conocida en los entornos educativos del mundo occidental. Bloom definía seis categorías, de progresiva complejidad: Conocimiento, Comprensión, Aplicación, Análisis, Síntesis y Evaluación."(Wikipedia)[59]

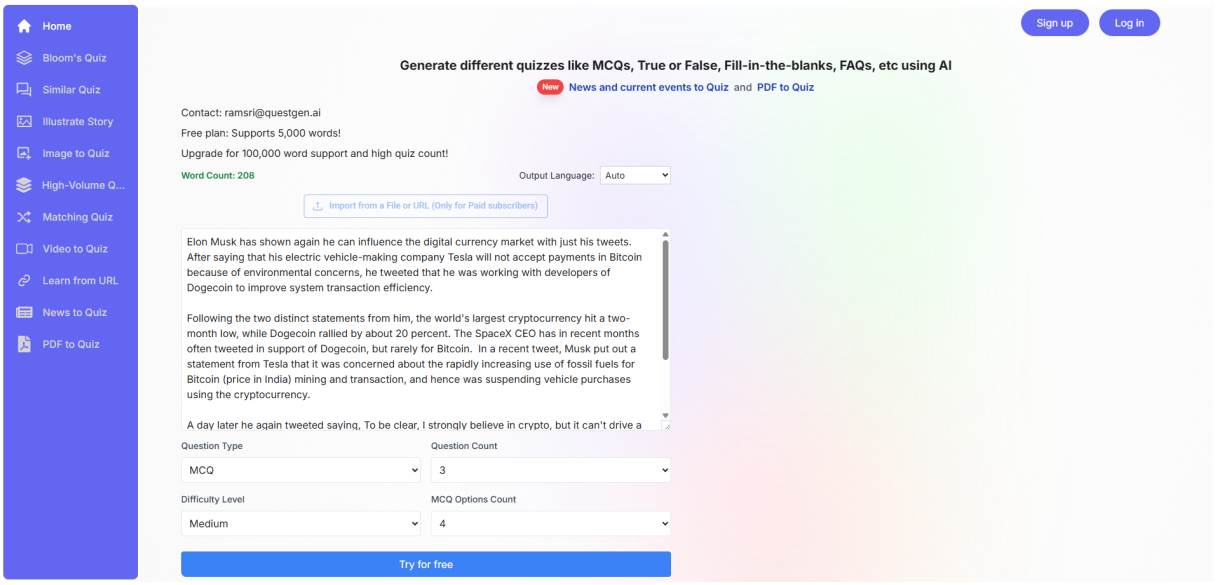

Figura 3.4: Página web de Questgen

Las otras tres alternativas no aportan ninguna novedad a estas ya comentadas y no son tan completas.

3.2.2. Discusión

Ninguna de las aplicaciones encontradas se termina de parecer del todo a este proyecto. Podemos encontrar las diferencias entre todas las aplicaciones en la siguiente tabla 3.1.

Tabla 3.1: Comparación entre Khanmigo Education, Merlyn Mind, Quizgecko, Questgen y TFG

Propuesta	**Khanmigo Education**	**Merlyn Mind**	**Quizgecko**	**Questgen**	**TFG**
Concepto	Apoyo a la plataforma de aprendizaje	LLMs para el aula	Generación de preguntas	Generación de preguntas	Generación de preguntas con explicación + dashboards
Requiere apuntes	No	No	Sí	Sí	No
De pago	Sí	Sí	Sí	Sí	No
Prueba gratuita	No	No	Sí	Sí	Gratuito
LLM basado en	GPT-4	pythia-12b	Desconocido	Desconocido	Llama 3
Tutorización	Sí	Sí	No	No	No
Individual/Aula	Individual	Aula	Individual	Individual	Individual
Rol	Profesor	Asistente del profesor	Generar preguntas	Generar preguntas	Generar preguntas + explicación

Este proyecto totalmente es gratuito para los estudiantes, no requiere de sus apuntes específicos, aporta explicación de las preguntas y muestra *dashboards* de su progreso. Estas son las principales características que aportan valor al proyecto, que se basa en aprender a partir de los objetivos de aprendizaje, historias de usuario y criterios de aceptación de las asignaturas generando preguntas tipo test y con una interfaz amigable y que invite al progreso gracias a sus *dashboards*.

El hecho de que sea aplicable a una asignatura y no a todas es una limitación debida al tiempo disponible, pero con no mucho tiempo más seríamos capaces de generalizar la aplicación para cualquier asignatura. Esto se debatirá en el último capítulo en el apartado de trabajo futuro.

3.3. Contexto científico-técnico

Para comprender la base tecnológica de este proyecto, necesitamos definir y controlar una serie de conceptos clave que van desde la inteligencia artificial en general hasta las técnicas

específicas que permiten ejecutar modelos de lenguaje de gran tamaño en hardware convencional. Esta sección establece una jerarquía entre los campos de la Inteligencia Artificial, el Aprendizaje Automático y el Aprendizaje Profundo; define cómo se mide el rendimiento de los modelos; y finalmente, detalla las técnicas de optimización que han sido cruciales para el desarrollo de la aplicación.

3.3.1. Inteligencia Artificial, Aprendizaje Automático y Aprendizaje Profundo

Para situar tecnológicamente este proyecto, es fundamental entender la relación entre tres campos clave, que es de hecho jerárquica. Partimos de la **Inteligencia Artificial** (IA), la disciplina más amplia que busca dotar a las máquinas de capacidades humanas. Dentro de ella, se encuentra el Aprendizaje Automático o ***Machine Learning*** (ML), que se especializa en algoritmos que aprenden de los datos. Finalmente, como un subcampo del ML, el **Aprendizaje Profundo** o ***Deep Learning*** (DL) utiliza redes neuronales profundas para resolver tareas de alta complejidad.

Los *Large Language Models* (LLMs), que son el núcleo de este trabajo, nacen precisamente del *Deep Learning* [19]. Para clarificar cómo estos conceptos se relacionan y diferencian en alcance y aplicación, la Tabla 3.2 ofrece un resumen comparativo que va de lo general a lo específico.

Tabla 3.2: Comparación entre Inteligencia Artificial, Machine Learning y Deep Learning.

Característica	**Inteligencia Artificial (IA)**	**Machine Learning (ML)**	**Deep Learning (DL)**
Definición	Campo amplio de la informática para crear máquinas inteligentes.	Subcampo de la IA que usa datos para que los sistemas aprendan y mejoren.	Subcampo del ML basado en redes neuronales con múltiples capas.
Alcance	Muy amplio. Incluye lógica, planificación, ML, etc.	Específico. Aprendizaje a partir de datos (supervisado, no supervisado).	Aún más específico. Uso de redes neuronales profundas para tareas complejas como el reconocimiento de patrones.
Ejemplo	Sistemas expertos, robots de ajedrez, asistentes virtuales.	Filtros de *spam*, sistemas de recomendación, previsión de la demanda.	Coches autónomos, reconocimiento facial, traducción automática, LLMs.

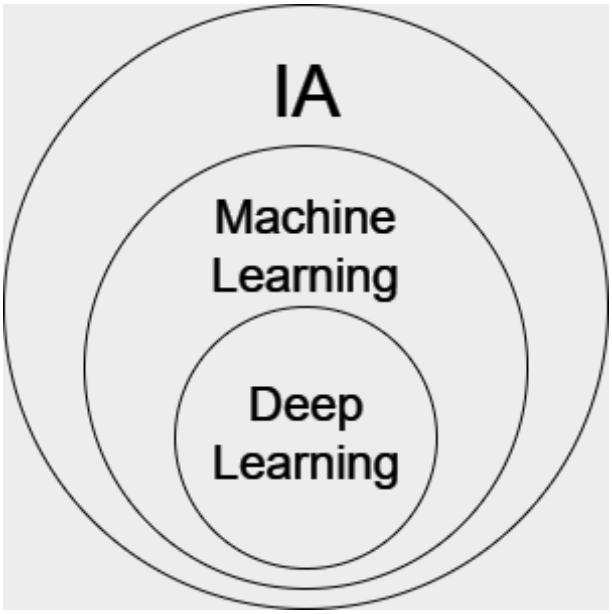

Figura 3.5: Comparativa IA vs ML vs DL

3.3.2. Benchmarking y Comparativa de LLMs

Para evaluar y comparar la efectividad de diferentes LLMs, la comunidad de IA utiliza el **benchmarking**. Un *benchmark* es un conjunto de pruebas y métricas estandarizadas diseñadas para medir el rendimiento de un modelo en diversas capacidades, como el razonamiento, el conocimiento general o la resolución de problemas matemáticos [27]. Esto permite una comparación objetiva entre modelos, como los de la familia Llama de Meta.

La elección del modelo base es una decisión crítica en este proyecto. Para realizar una selección informada, se compararon las versiones Llama 2 de 7 mil millones (7B) de parámetros, Llama 3 de 8 mil millones (8B) de parámetros, y Llama 3.1 de 8 mil millones (8B) de parámetros. La evaluación se basó en un conjunto de *benchmarks* estandarizados, cada uno diseñado para medir una capacidad cognitiva clave:

- **MMLU** (*Massive Multitask Language Understanding*): Mide el conocimiento general y la capacidad de resolución de problemas.
- **GPQA** (*Graduate-Level Google-Proof Q&A*): Evalúa el razonamiento en preguntas de nivel de posgrado en biología, física y química.
- **HumanEval**: Mide la capacidad del modelo para generar código Python funcional a partir de una descripción del problema.
- **GSM-8K** (*Grade School Math 8K*): Pone a prueba la habilidad para resolver problemas matemáticos de varios pasos a nivel de primaria.
- **MATH**: Evalúa el razonamiento matemático con problemas de competiciones de matemáticas de nivel preuniversitario.

Modelo	MMLU	GPQA	HumanEval	GSM-8k	MATH
Llama 2 7B	34.1	21.7	7.9	25.7	3.8
Llama 3 8B	66.6	**34.2**	62.2	79.6	30.0
Llama 3.1 8B	**73.0**	32.8	**72.6**	**84.5**	**51.9**

Tabla 3.3: Comparativa de rendimiento (*benchmarks*) para modelos Llama de 8B de parámetros. Las puntuaciones más altas son mejores.

Los resultados de esta comparativa, que se detallan en la Tabla 3.3, no solo permiten cuantificar el rendimiento de cada modelo, sino que también revelan la rápida evolución en este campo. [33, 34, 32]. Se tuvo que elegir Llama 3 en vez de Llama 3.1 porque en la etapa de testeo dio peores resultados.

Natural Language Processing

Un concepto importante relacionado con los LLMs es el de Procesamiento del Lenguaje Natural o *Natural Language Processing* (NLP), que es la rama de la IA dedicada al análisis y representación computacional de los lenguajes humanos.

Sin embargo, a pesar de los enormes avances, el objetivo de un análisis automático de texto al nivel de la cognición humana sigue presentando desafíos significativos [8]. Esta limitación se manifiesta en dificultades prácticas, especialmente relevantes para el ámbito educativo. Por ejemplo, un modelo puede fallar al interpretar la ambigüedad en la pregunta de un estudiante, confundir el sarcasmo o la ironía con una afirmación literal, o carecer del razonamiento de sentido común necesario para entender el contexto implícito de un problema. Además, la comprensión de jerga muy específica de una materia académica a menudo requiere un entrenamiento explícito que los modelos generalistas no siempre poseen.

En relación con los LLM, el NLP es el campo que aporta las técnicas fundamentales, como la arquitectura *Transformer*, para que estos modelos puedan comprender y generar lenguaje humano de una manera cada vez más efectiva.

Tokens

Los *tokens* son fragmentos de texto que un modelo lee o genera. En esencia, es la longitud básica que utilizan los modelos para calcular la longitud de un texto. En los *Large Language Models* sirven como *input* y como *output* tanto durante el entrenamiento como en la inferencia. Un *token* no necesariamente corresponde a una palabra completa, puede ser una unidad más pequeña, como un carácter o parte de una palabra, o una unidad más grande como una frase completa. La *tokenización* es la división de un texto en *tokens*, y este proceso puede ayudar al modelo a manejar diferentes idiomas, vocabularios y formatos, y a reducir los costos computacionales y de memoria. [21]

El *tokenizador* de los modelos GPT de OpenAI utiliza un método de *tokenización* llamado *Byte-Pair Enconding* (BPE). Es un método que fusiona los pares de caracteres o bytes más frecuentes en un solo *token*, hasta alcanzar el límite de *tokens*. De esta forma se pueden gestionar palabras raras o no vistas anteriormente y crear representaciones más compactas y consistentes

del texto. Aproximadamente cada *token* son unos cuatro caracteres, y en palabras sería un *token* cada aproximadamente tres cuartos de palabra. [21]

El límite de *tokens* en GPT-3.5 es de 4096 y en GPT-4 de 8192, combinando la suma de la pregunta y la respuesta. [21]

Hugging Face

En la actualidad Hugging Face es el repositorio más grande y activo de modelos de inteligencia artificial, cuyos usuarios y contribuidores aportan nuevos modelos a diario a un ritmo exponencial, duplicándose el número de modelos aproximadamente cada 18 semanas [63].

Realmente no hay una convención exacta [22] de como llamar a los modelos puesto que es muy habitual tomar modelos pre-entrenados funcionales con una base sobre la que realizar un entrenamiento específico para la tarea que le queremos otorgar, y esto genera multitud de modelos diferentes sobre una misma base.

También aporta a desarrolladores una API desde la cual se puede hacer uso de sus modelos, foros de la comunidad, documentación, tutoriales y librerías muy relevantes para el desarrollo de la IA.

Transformers

Los *Transformers* son una arquitectura de modelos de aprendizaje automático desarrollada originalmente por investigadores de Google en 2017 [55] que ha revolucionado muchos campos, incluido el procesamiento del lenguaje natural.

Los *Transformers* no solo han supuesto una mejora de rendimiento, sino que también han hecho posible que los modelos actuales puedan trabajar en varios idiomas, resolver diferentes tipos de tareas y adaptarse a distintos contextos sin necesidad de empezar desde cero cada vez [24]. Gracias a esto, los *Transformers* son la base de modelos como GPT-3 y GPT-4, que ya se están utilizando en áreas como la educación [5], el aprendizaje de idiomas [15] o la generación automática de preguntas educativas [56, 14].

La clave del éxito de los *Transformers* radica en que abandonan por completo las redes recurrentes y convolucionales tradicionales, y basan todo su funcionamiento en mecanismos de atención. Esta decisión permite que los *Transformers* puedan procesar las secuencias de entrada de forma mucho más paralelizable, reduciendo significativamente los tiempos de entrenamiento respecto a modelos anteriores. Además, gracias al uso de la atención multi-cabeza (*multi-head attention*), estos modelos pueden aprender a enfocarse simultáneamente en diferentes partes de una secuencia, lo que mejora la captura de relaciones complejas a larga distancia entre las palabras [55].

En sus experimentos originales, los autores del modelo *Transformer* lograron superar el estado del arte en tareas de traducción automática, alcanzando puntuaciones BLEU superiores a las de modelos previos y con un coste computacional mucho menor [55]. Además, el modelo mostró capacidad para interpretar bien otras tareas de procesamiento del lenguaje natural, como el análisis sintáctico.

Hugging Face aporta una librería de *Transformers* muy relevante y utilizada ampliamente por desarrolladores.

3.3.3. Cuantización y `llama.cpp`

Uno de los mayores desafíos para ejecutar modelos de lenguaje de gran tamaño (LLMs) es el alto requerimiento computacional y de memoria que requieren durante la inferencia. Por ejemplo, el modelo LLaMA 3 de 8 mil millones de parámetros ocupa aproximadamente 16 GB en su formato de media precisión (`float16`) [32], lo cual excede la capacidad de muchos ordenadores personales.

Para solventar esta limitación, se recurre a la **cuantización**, una técnica que reduce la precisión numérica de los pesos del modelo (por ejemplo, de 16 bits a 4 o 5 bits), disminuyendo drásticamente su tamaño y el uso de memoria RAM, a cambio de una pérdida de precisión generalmente pequeña [13, 12].

Esquemas de cuantización comunes:

- **Q8, Q5, Q4:** Indican el número de bits por parámetro. Por ejemplo, Q5 representa cada peso con 5 bits, permitiendo hasta 32 valores posibles.
- **Cuantización por bloques:** Agrupa pesos y aplica escalas y desplazamientos locales por bloque, mejorando la precisión respecto a una cuantización global.
- **Cuantización doble:** Técnicas como QLoRA cuantizan también los factores de escala, permitiendo reducir aún más el uso de memoria sin afectar notablemente al rendimiento.

Estas técnicas permiten ejecutar modelos como Llama 3 en ordenadores personales con 8 o 16 GB de RAM. En este proyecto se ha utilizado Llama 3 8B con una cuantización Q5, lo que ha permitido desplegar el modelo localmente sin necesidad de una GPU.

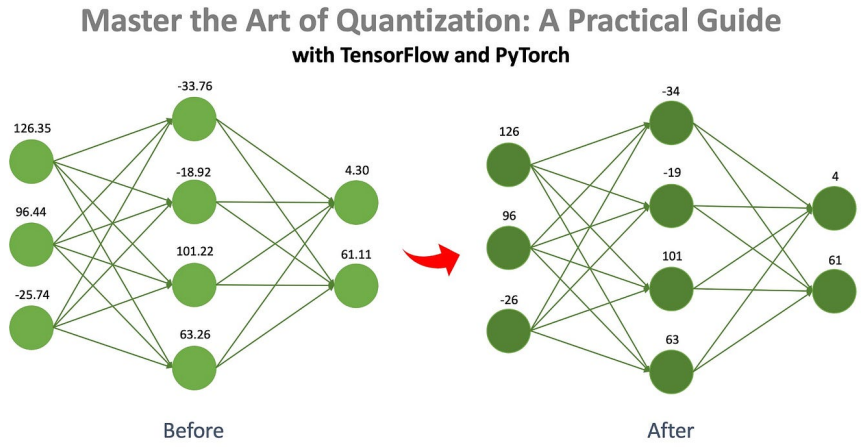

Figura 3.6: Ejemplo visual de cuantización: a la izquierda, un grafo de pesos de los parámetros en precisión flotante; a la derecha, los mismos valores representados en números enteros en menos bits. Imagen adaptada de [25].

Tal como se muestra en la Figura 3.6, la cuantización transforma los valores continuos de los pesos (como `float32` o `float16`) a un conjunto reducido de valores discretos (por ejemplo, `int4` o `int5`). En lugar de almacenar decimales con alta precisión, se aproxima cada peso al valor más cercano dentro del rango permitido por el número de bits. Este proceso se complementa con técnicas de escalado y desplazamiento para minimizar la pérdida de información [25].

Un componente esencial para ejecutar estos modelos cuantizados es **`llama.cpp`**, una implementación optimizada en C/C++ desarrollada por Georgi Gerganov (`ggerganov`) [16]. Esta herramienta permite ejecutar modelos LLaMA directamente en CPU sin necesidad de GPU de alto rendimiento, y ha sido clave para democratizar el uso de LLMs gracias a su alto nivel de eficiencia.

La combinación de cuantización y ejecución mediante `llama.cpp` ha permitido existan proyectos que funcionen localmente en equipos personales. Además, herramientas como *GPU Poor* [7] ayudan a estimar si un modelo puede inferenciarse o hacer *fine-tuning* con éxito según las especificaciones de *hardware* disponibles.

Fine-Tuning y ajuste fino eficiente (PEFT)

El *fine-tuning* (ajuste fino) consiste en adaptar un modelo preentrenado a una tarea concreta reentrenando sus pesos sobre un conjunto de datos específico. Al usar un modelo base ya entrenado, se aprovechan conocimientos previos y se evita entrenar desde cero, ahorrando tiempo, datos y recursos computacionales [44]. De hecho, ajustar un modelo grande suele requerir una fracción de la energía y tiempo de un preentrenamiento completo: un estudio sobre BERT muestra que "un solo pre-entrenamiento consume sustancialmente más energía que el *fine-tuning*" [57].

En el caso de los modelos de lenguaje (LLM), el número de épocas (pasadas completas por el conjunto de datos) suele ser bajo: una, tres, ocho o treinta y una son habituales, frente a proyectos como `ImageNet` que utilizan más de 300. Esto se debe a que repetir los mismos datos muchas veces puede tener un efecto negativo en el rendimiento del modelo [60].

Ventajas del fine-tuning:

- Ahorro en tiempo, energía y recursos computacionales.
- Se requieren menos datos etiquetados específicos.
- Se aprovecha el conocimiento previo del modelo generalista.

Parameter-Efficient Fine-Tuning (PEFT)

El *Parameter-Efficient Fine-Tuning* (PEFT) es una estrategia que mantiene congelada la mayoría de los pesos del modelo y solo entrena un subconjunto reducido de parámetros. En lugar de modificar toda la red neuronal, PEFT introduce capas o adaptadores entrenables que permiten personalizar el modelo de forma más eficiente [20].

Ejemplos de PEFT:

- **LoRA (Low-Rank Adaptation):** Introduce matrices de bajo rango en cada capa del modelo y solo entrena esas matrices adicionales.
- **QLoRA:** Técnica combinada que aplica LoRA sobre un modelo previamente cuantizado a 4 bits (formato NF4), permitiendo afinar modelos muy grandes en una sola GPU [52].

Comparación *Fine-Tuning* completo vs PEFT:

- ***Fine-tuning* completo:** Entrena todos los parámetros. Requiere mucha memoria, tiempo y cómputo. Cada modelo entrenado ocupa tanto como el original.
- **PEFT:** Entrena solo adaptadores o capas específicas. Mucho más eficiente en uso de recursos y fácil de almacenar y desplegar.

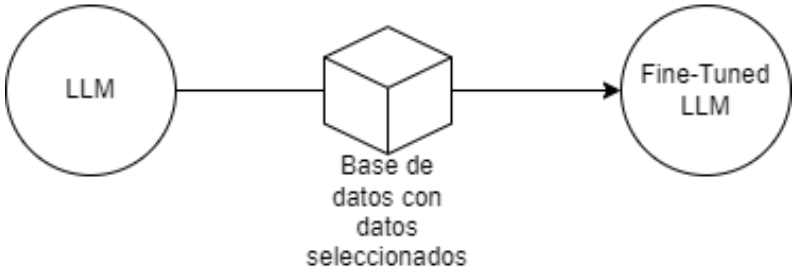

Figura 3.7: Representación visual del ajuste fino

Resumen comparativo

Aspecto	**Fine-Tuning completo**	**PEFT (LoRA / QLoRA)**
Parámetros entrenados	Todos los del modelo	Solo adaptadores
Requisitos de GPU	Muy altos	Bajos o moderados
Tamaño del modelo resultante	Igual que el original	Igual que el original + adaptadores (pequeños)
Tiempo de entrenamiento	Alto	Bajo
Consumo energético	Alto	Bajo
Facilidad para almacenar múltiples versiones	Baja	Alta

Tabla 3.4: Comparación entre *fine-tuning* completo y PEFT

Docker

Docker Desktop es una plataforma de *software* que permite empaquetar software en unidades estandarizadas llamadas contenedores o *dockers*. Estos incluyen todo lo necesario para que el *software* se ejecute; desde un sistema operativo si lo requiere, hasta las bibliotecas, herramientas de sistema, ficheros, código y tiempo de ejecución. [2] Los *docker* son una manera de crear y

distribuir *software* de forma independiente a la máquina del usuario, y aseguran que un contenedor funcional lo será en cualquier ordenador. Esto facilita al usuario la instalación y uso del *software*.

Por lo tanto, es muy conveniente para algo tan complejo como la ejecución de LLMs en local el uso de un *docker* con la configuración correcta para conseguir una mayor portabilidad.

Parte II

Desarrollo de la propuesta

Capítulo 4

Descripción y desarrollo de la propuesta

En este capítulo encontramos dos partes. En la primera se trata la procedencia de los datos usados, el análisis de requisitos y los diagramas realizados. En la segunda parte se explica la idea de la aplicación con *mockups*, se habla de las pruebas necesarias para asegurar que el proyecto funciona correctamente y cumple los requisitos definidos y por último se especifican las optimizaciones realizadas a la aplicación.

4.1. Análisis

4.1.1. Descripción de fuentes de datos

Se han utilizado los objetivos de aprendizaje, historias de usuario y criterios de aceptación de la asignatura de Bases de Datos del grado, impartida por Miguel Ángel Martínez Prieto, tutor del Trabajo Fin de Grado. Estos datos han sido fundamentales para el desarrollo de la estructura de la página y la aplicación se basa en que el estudiante vea su desarrollo en torno a estos criterios de aceptación de cada historia de cada objetivo.

Para el *fine-tuning* se ha utilizado el *dataset* Miguelsbdh/training-tfg [36], creado por mi y que usa como base otro *dataset* muy popular. La base del *dataset* utilizado corresponde al de yahma/alpaca-cleaned [62], que utiliza una versión mejorada de un *dataset* creado por Stanford con el fin de mejorar la calidad de las respuestas de Llama, tanto en contenido como sobre todo en forma. La parte particular de mi *dataset* corresponde a una serie de pares prompt-respuesta creados por mi acerca del contenido del libro «Bases de Datos» de Mercedes Marqués para la *Universitat Jaime I* [29]. El libro se pasó a texto plano, se corrigieron errores de forma manual provenientes de este paso a texto plano, y finalmente se realizó una tarea de separación en las secciones del libro de interés para la asignatura. El *fine-tuning* se realizó con Google Collab, a partir de un cuaderno realizado por Unsloth [54].

Por supuesto, Llama 3 fue entrenado con una fuente de datos filtrados por la empresa Meta como se describe en la página web de Llama 3 [33].

4.1.2. Requisitos

Casos de Uso

- CU-1: El estudiante selecciona un objetivo de aprendizaje y realiza un test asociado a ese objetivo.
- CU-2: El estudiante responde a cada pregunta del test
- CU-3: El estudiante recibe una retroalimentación indicando la respuesta correcta.
- CU-4: El estudiante puede visualizar un resumen de su rendimiento en los *dashboards* estadísticos.
- CU-5: El estudiante puede repetir las preguntas que haya fallado previamente.
- CU-6: El estudiante puede solicitar nuevas preguntas sobre un criterio concreto y estas se generan a través del LLM.
- CU-7: El estudiante puede consultar y visualizar los objetivos de aprendizaje disponibles.

Diagrama de casos de uso

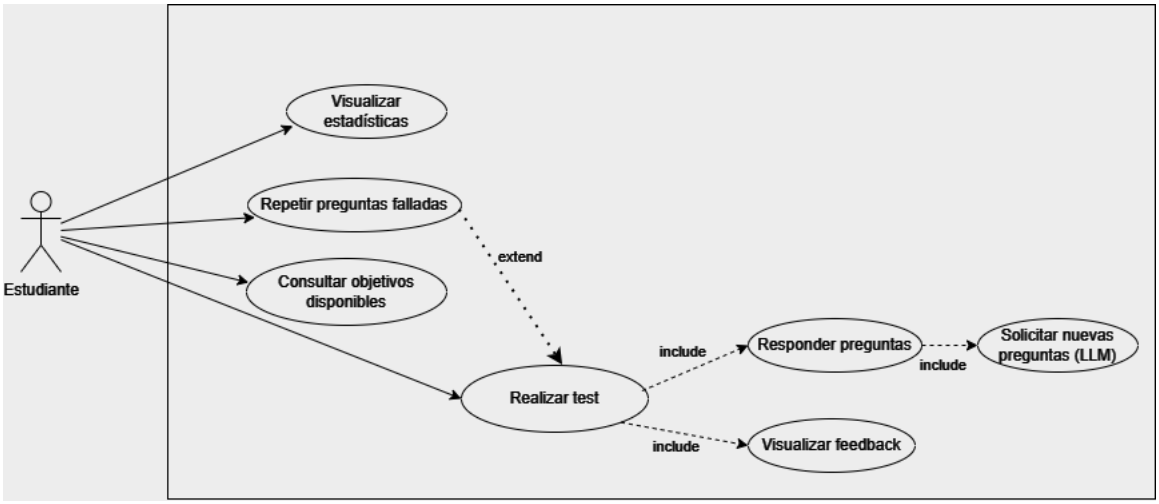

Figura 4.1: Diagrama de casos de usos

Requisitos de usuario

- RU-1: El estudiante podrá conocer los objetivos de aprendizaje disponibles.
- RU-2: El estudiante podrá seleccionar un objetivo de aprendizaje y realizar un test asociado a ese objetivo.
- RU-3: El estudiante podrá elegir una de las respuestas del test para contestar.
- RU-4: El estudiante podrá obtener una retroalimentación inmediata indicando la respuesta correcta al finalizar cada pregunta y su explicación.

- RU-5: El estudiante podrá visualizar su progreso en los *dashboards* estadísticos.
- RU-6: El estudiante podrá repetir las preguntas que haya fallado previamente.
- RU-7: El estudiante podrá solicitar nuevas preguntas sobre un criterio concreto y estas serán generadas por el sistema.

Requisito	CU-1	CU-2	CU-3	CU-4	CU-5	CU-6	CU-7
RU-1							**x**
RU-2	**x**	**x**	**x**				
RU-3		**x**					
RU-4			**x**				
RU-5				**x**			
RU-6					**x**		
RU-7						**x**	

Tabla 4.1: Matriz de trazabilidad entre requisitos de usuario y casos de uso

Requisitos de negocio

- **RN-1:** La aplicación permite realizar *tests* acerca de un objetivo de aprendizaje.
- **RN-2:** Se pueden consultar *dashboards* con las estadísticas de los test realizados.
- **RN-3:** La aplicación debe ofrecer un mecanismo de aprendizaje continuo y personalizado, permitiendo al alumno generar contenido de evaluación nuevo bajo demanda para reforzar áreas de conocimiento específicas.
- **RN-4:** Las preguntas generadas se guardan en la base de datos para poder ser reutilizadas en futuras sesiones.
- **RN-5:** El alumno puede repetir las preguntas falladas para reforzar el aprendizaje.

Requisitos funcionales

- **RF-1:** El sistema deberá exponer un endpoint en la API que, al recibir el ID de un criterio de aceptación, inicie el proceso de generación de una nueva pregunta a través del LLM.
- **RF-2:** El sistema mostrará al terminar cada pregunta cuál era la correcta.
- **RF-3:** El sistema almacenará las nuevas preguntas generadas por el LLM en la base de datos.
- **RF-4:** El sistema actualizará los *dashboards* cada vez que se registren nuevas respuestas.

Requisitos no funcionales

- **RNF-1:** Un sistema que permita dedicar 8GB de RAM exclusivamente al *docker* será capaz de ejecutar la aplicación con el LLM activo.
- **RNF-2:** Se guardará una versión de la base de datos que permita un uso inicial completo de la aplicación.
- **RNF-3:** Debe ser fácilmente portable a otros ordenadores.
- **RNF-4:** El LLM podrá estar alojado en un servidor y este responderá las solicitudes HTTP para generar las preguntas.
- **RNF-5:** El LLM podrá ser ejecutado localmente, a través de un *docker*.
- **RNF-6:** La página web será *responsive*, es decir, se reescalará el tamaño de los elementos según el tamaño de la ventana.

4.2. Diseño

4.2.1. Modelo Entidad-Relación y modelo lógico

Modelo Entidad-Relación

Puesto que nos encontramos con que la aplicación cuenta con una base de datos, que decidimos que sea relacional, podemos plasmar su configuración en un diagrama entidad-relación.

Existen siete entidades: OBJETIVO, HISTORIA, CRITERIO, PREGUNTA, OPCIÓN, INTENTO y USUARIO.

Las entidades de OBJETIVO, CRITERIO e HISTORIA hacen referencia a los objetivos y criterios de aceptación y a las historias de usuario, de las que ya hemos hablado. En cada una de las tablas existentes para estas entidades podremos encontrar todos los objetivos, historias y criterios de la asignatura «Sistemas de Bases de Datos», y esos datos son en principio estáticos y no se modificarán en esta aplicación.

Las entidades PREGUNTA y OPCIÓN tienen una relación de padre-hijo, pues las opciones son dependientes de la existencia de la pregunta.

Por último, las entidades USUARIO e INTENTO tienen también una relación de padre-hijo, pues la existencia de un intento requiere de un usuario que lo realice.

El resto de detalles importantes como las claves primarias, las claves foráneas y los atributos de cada, y las relaciones entre entidades pueden ver en el modelo entidad-relación 4.2 y en el modelo lógico 4.2.1.

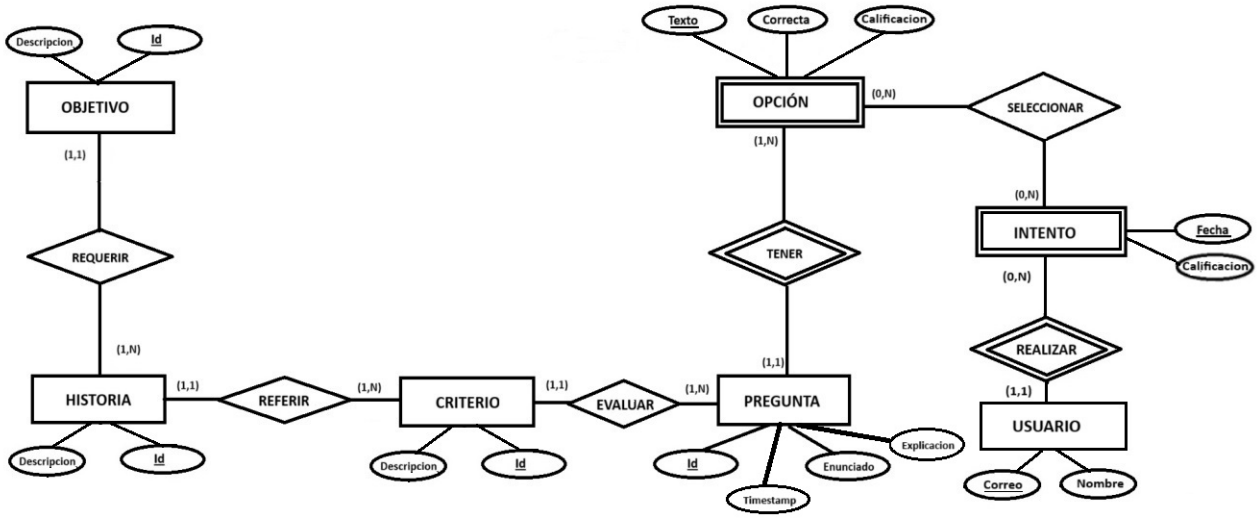

Figura 4.2: Modelo Entidad-Relación

Modelo Lógico

- **OBJETIVO** (Id, Descripcion)
- **HISTORIA** (Id, Descripcion, ObjetivoId)
  - FK: ObjetivoId → OBJETIVO(Id)
- **CRITERIO** (Id, Descripcion, HistoriaId)
  - FK: HistoriaId → HISTORIA(Id)
- **PREGUNTA** (Id, Enunciado, CriterioId, Explicacion, Timestamp)
  - FK: CriterioId → CRITERIO(Id)
- **OPCION** (PreguntaId, Texto, Correcta, Calificacion)
  - FK: PreguntaId → PREGUNTA(Id)
- **INTENTO** (Fecha, UsuarioCorreo, Calificacion)
  - FK: UsuarioCorreo → USUARIO(Correo)
- **USUARIO** (Correo, Nombre)
- **SELECCIONAR** (IntentoFecha, OpcionPreguntaId, OpcionTexto)
  - FK: IntentoFecha → INTENTO(Fecha)
  - FK: (OpcionPreguntaId, OpcionTexto) → OPCION(PreguntaId, Texto)

4.2.2. Arquitectura física

El despliegue del sistema se realizaría en cuatro servidores, y podría desplegarse perfectamente utilizando contenedores. El uso de contenedores permite encapsular los distintos componentes del sistema, facilitando su mantenimiento, escalabilidad y despliegue. Supondremos un despliegue con *dockers* pues es la opción más interesante.

El primero contiene todo lo relacionado con el sistema de gestión de bases de datos (SGBD), encargado de la almacenar de la información del proyecto. En el entorno de desarrollo se ha utilizado **MySQL** a través de **XAMPP**, por lo que el puerto habitual es el `3306`, y se accede a través de `localhost/phpmyadmin`. En producción, este componente podría desplegarse en un contenedor ***(Docker 1)*** escuchando en el puerto `3306` por defecto.

El segundo servidor ejecuta el *backend*, desarrollado con **Node.js**. Este componente gestiona toda la lógica de negocio, las peticiones del *frontend*, y la comunicación con el modelo LLM y la base de datos. En la imagen se representa como ***Docker 2***, y en el entorno de desarrollo se ha configurado para escuchar en el puerto `9001`.

El tercer servidor ***(Docker 3)*** contiene el *frontend*, compuesto por archivos HTML, CSS y JavaScript con la estructura de un proyecto de Quasar y con servidor NGINX. Este servidor actúa como punto de entrada para el usuario y está configurado en el puerto **9000**. Elegimos NGINX porque consideramos que es el más conveniente, ya que está en constante crecimiento y tiene una mejor gestión de subprocesos, por lo que responde mejor a las solicitudes y consume menos memoria RAM.

El cuarto servidor ***(Docker 4)*** ejecuta el modelo Llama 3 cuantizado y con el *fine-tuning*, disponible mediante la API de **llama-cpp-python**, basada en **Python**. Este servicio se comunica con el *backend* a través de HTTP en el puerto 8000. El modelo *fine-tuneado* y cuantizado en formato **gguf** permite su ejecución local con un ordenador personal.

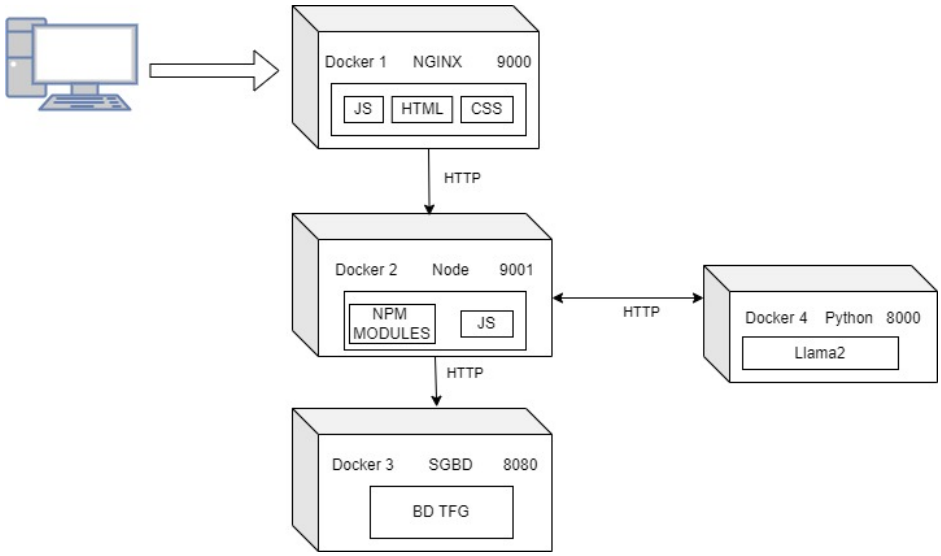

Figura 4.3: Arquitectura física del sistema

En la Figura 4.3 se muestra la interacción entre los distintos componentes, donde todas las comunicaciones se realizan mediante el protocolo HTTP. Este diseño modular permite separar responsabilidades y facilita el mantenimiento y la escalabilidad del sistema.

4.2.3. Arquitectura lógica

Para la implementación de la aplicación web, el servidor, la base de datos y el modelo LLM, se ha utilizado un ecosistema basado en JavaScript, seleccionando herramientas modernas y eficientes tanto para el *backend* como para el *frontend*. El *docker* que contiene al modelo utiliza

Backend: Node.js

El *backend* está hecho en Node.js 4.4, un entorno de ejecución de JavaScript del lado del servidor basado en el motor V8 de Google que se utiliza en Google Chrome [37]. Se eligió Node.js para el desarrollo del *backend* debido a su arquitectura asíncrona y no bloqueante, basada en el motor V8. Esta característica es crucial para el proyecto (**RNF-1**, **RF-1**), ya que permite gestionar eficientemente las peticiones de generación de preguntas al LLM, que es un proceso lento, sin bloquear otras interacciones del usuario, como la consulta de *dashboards* o la respuesta a otras preguntas.

Figura 4.4: Logo de Node.js [9]

Las librerías más importantes utilizadas en el desarrollo del *backend* son:

- **Express.js:** Un *framework* minimalista y flexible que ha servido como base para la construcción de la API REST, gestionando las rutas, peticiones y respuestas.
- **mysql2/promise:** El cliente de MySQL para Node.js, utilizado para establecer la comunicación con la base de datos de manera asíncrona mediante promesas, garantizando un manejo de datos eficiente.
- **Axios:** Cliente HTTP basado en promesas, esencial para la comunicación entre el servidor y el LLM externo, enviando las peticiones para la generación de preguntas.
- **CORS:** *Middleware* para habilitar el Intercambio de Recursos de Origen Cruzado, permitiendo que la aplicación *frontend* pueda realizar solicitudes seguras al *backend*.

Frontend: Quasar Framework

La página web está realizada con Quasar, un *framework* de código abierto basado en Vue.js que permite desarrollar interfaces de usuario de alta calidad y rendimiento [42]. Para el *frontend* se seleccionó Quasar Framework por su ecosistema basado en Vue.js, que facilita el desarrollo de interfaces reactivas. Su amplio catálogo de componentes (`Q-Card`, `Q-Expansion-Item`) permitió implementar rápidamente el diseño visual definido en los *mockups* (Sección 4.2.4). Además, su sistema de *grid* fue fundamental para cumplir el requisito de diseño *responsive* (**RNF-5**), asegurando una experiencia de usuario consistente en diferentes tamaños de pantalla. [41]

Figura 4.5: Logo de Quasar [43]

Las librerías más importantes utilizadas son:

- **Vue.js:** El *framework* progresivo sobre el que se construye Quasar, aportando un sistema reactivo y una arquitectura basada en componentes que facilita la organización del código.
- **Chart.js y vue-chartjs:** Estas librerías han sido fundamentales para la creación de los *dashboards* estadísticos. **Chart.js** es el motor de renderizado de los gráficos, mientras que **vue-chartjs** proporciona los componentes de Vue para integrarlos de manera sencilla y reactiva en la aplicación.
- **Vue Router:** La librería oficial para la gestión de las rutas de la aplicación, permitiendo la navegación entre las diferentes páginas y vistas del sistema.
- **Axios:** Utilizado también en el *frontend* para realizar las peticiones HTTP a la API del *backend*, obteniendo y enviando los datos necesarios para la interacción del usuario.

LLM: Llama.cpp

Para integrar el modelo Llama 3 en la aplicación se ha utilizado **llama.cpp**, en concreto mediante la imagen del contenedor `ghcr.io/abetlen/llama-cpp-python:v0.2.77` [1]. Esta implementación, desarrollada por el usuario `abetlen`, permite la inferencia de llama.cpp a través de una API REST, que viene implementada perfectamente ya en un *docker* y que facilita su integración desde otros servicios como el *backend* de la aplicación.

Aunque la base del proyecto de Llama.cpp la aporta Gregory Gerganov [16], el proyecto del usuario *Abetlen*, desarrollado en Python, permite desplegarlo en un *docker* y exponer una interfaz HTTP estándar. Esta arquitectura ha sido clave para automatizar la generación de preguntas tipo test en base a los criterios de aceptación definidos en la base de datos.

Figura 4.6: Logo de Llama.cpp [28]

Las razones principales por las que se ha elegido esta solución son:

- **Ejecución local sin dependencias externas:** Permite ejecutar el modelo sin necesidad de conexión a Internet o a servicios en la nube.

- **Compatibilidad con modelos cuantizados:** El formato `gguf` permite reducir significativamente el uso de recursos.

- **API HTTP estándar:** Facilita su consumo desde el *backend* sin necesidad de desarrollos adicionales.

- **Modularidad y portabilidad:** El uso de contenedores (*dockers*) permite replicar el entorno con facilidad.

Queda reflejado todo esto en el diagrama de la arquitectura lógica: 4.7.

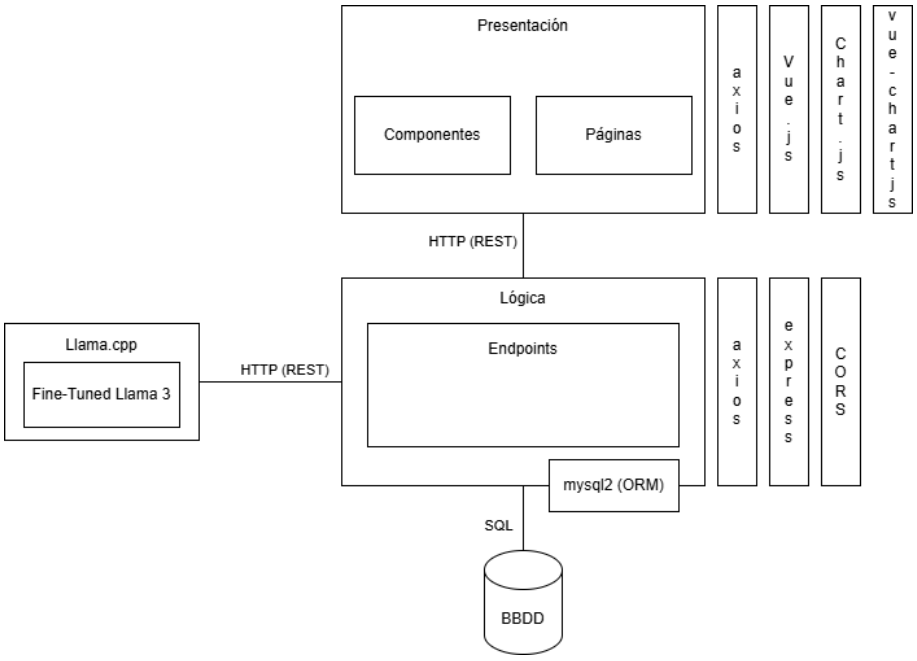

Figura 4.7: Arquitectura lógica del sistema

4.2.4. Mockups de la interfaz

Aquí se presentan los *mockups* diseñados para las principales pantallas de la aplicación, los cuales sirvieron como guía para el desarrollo de la interfaz de usuario. En la página de inicio queremos mostrar el progreso de los objetivos, de las historias y las estadísticas de las preguntas como se puede observar en la Figura 4.8 y en la Figura 4.9.

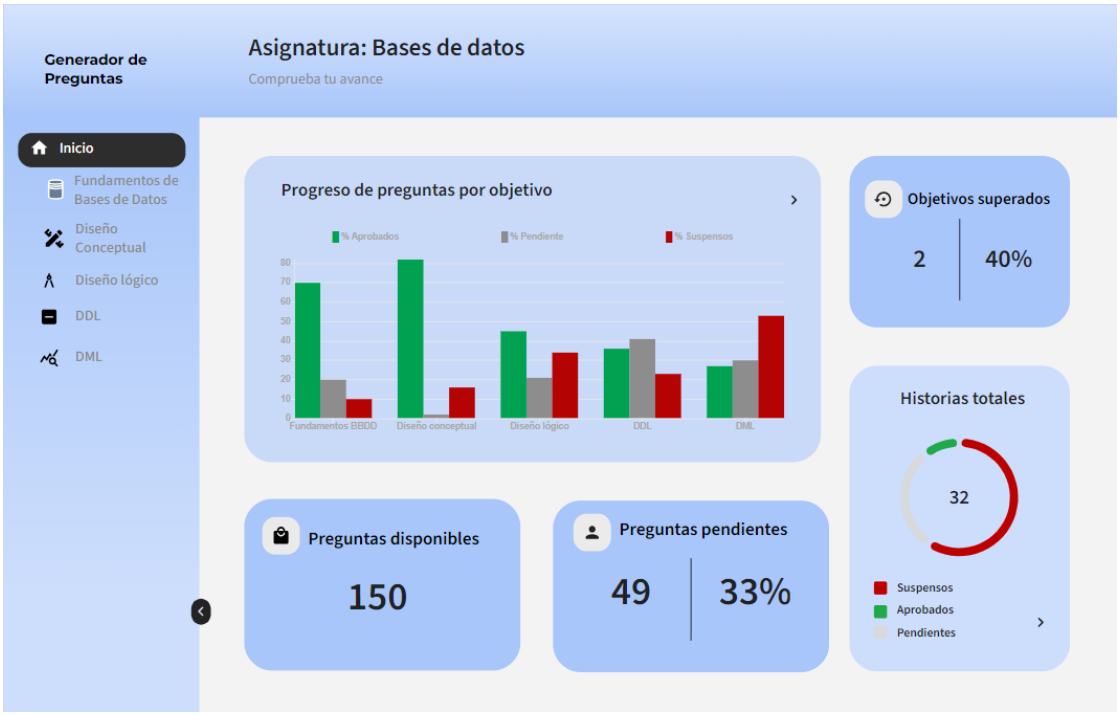

Figura 4.8: *Mockup* de la pantalla de inicio

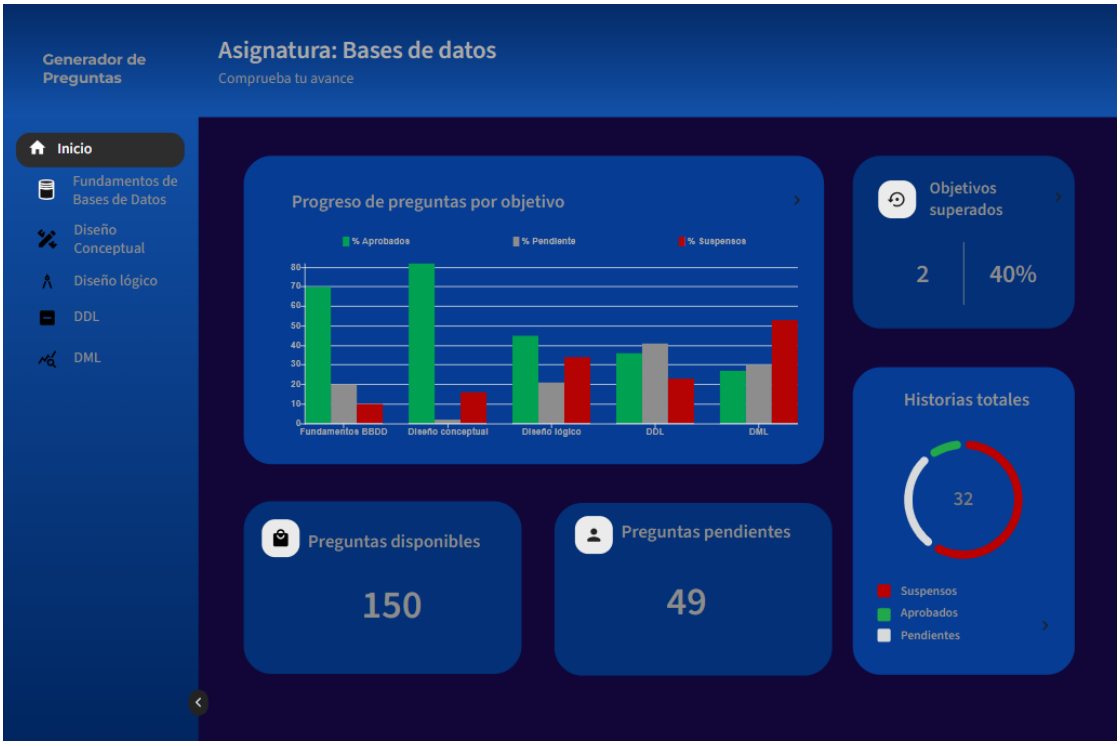

Figura 4.9: *Mockup* de la pantalla de inicio en modo oscuro

Para las páginas destinadas a cada uno de los objetivos de aprendizaje queremos tener la información de cuántas preguntas existen y cuántas hay pendientes para el objetivo y también cada una de sus historias de usuario. También queremos tener gráficas de "donut" en las que veamos el estado de las distintas historias de usuario y criterios de aceptación. Finalmente, queremos que haya botones de evaluar historia que nos lleve a una página donde evaluarnos de las preguntas de la historia y un botón para solicitar más preguntas de esta historia de usuario.

La información específica acerca de cada historia será visible dependiendo de si se extiende la historia de usuario o no, como podemos ver en la Figura 4.10 y en la Figura 4.11.

Figura 4.10: *Mockup* de una historia

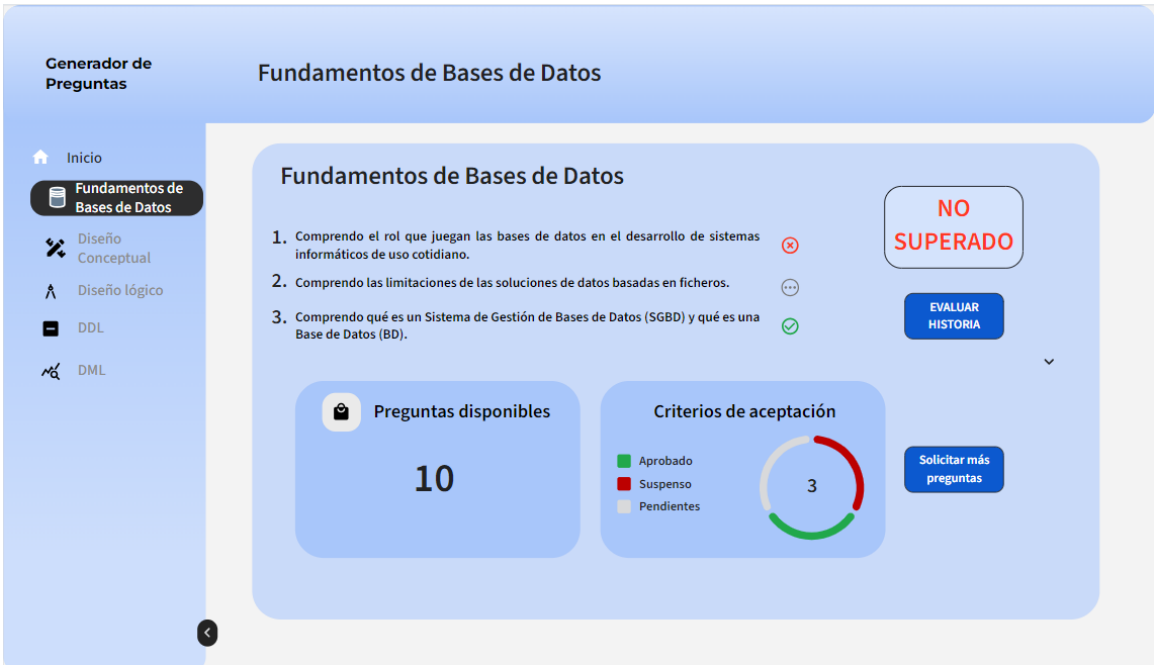

Figura 4.11: *Mockup* de una historia extendida

4.2.5. Diagramas de secuencia

Se presentan dos diagramas de secuencia que representan el flujo de trabajo de dos procesos clave en el funcionamiento de la aplicación

El primero es acerca de la creación de nuevas preguntas mediante el modelo LLM.

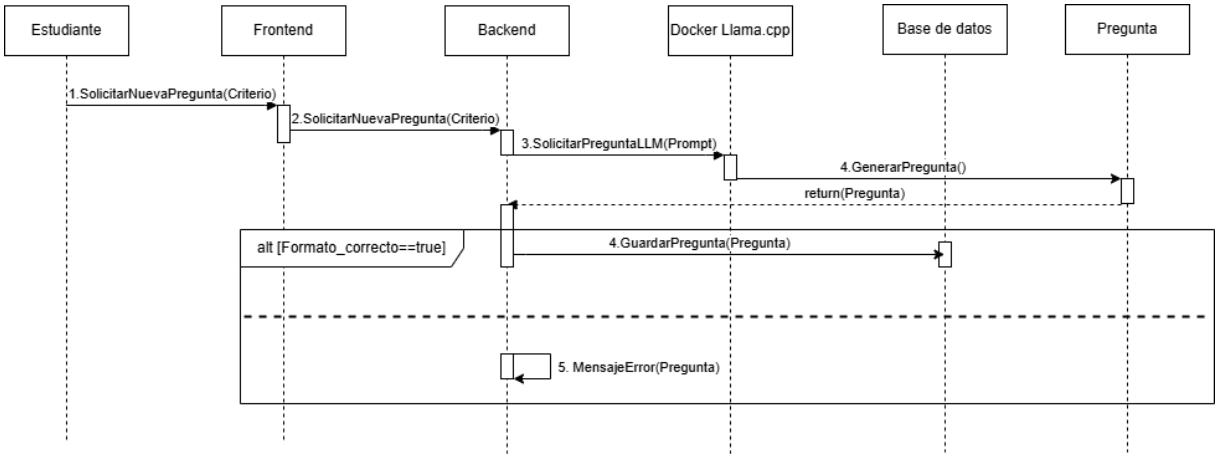

Figura 4.12: Diagrama de secuencia de creación de pregunta

Este diagrama describe el proceso que se ejecuta cuando un estudiante solicita una nueva pregunta basada en un criterio específico. El flujo es el siguiente:

1. El estudiante inicia la solicitud desde la interfaz, indicando el criterio deseado.

2. El *frontend* transmite esta solicitud al *backend*.

3. El *backend* construye el *prompt* correspondiente y realiza una petición al modelo LLM alojado en un *docker* con `llama.cpp`.

4. El modelo genera la pregunta y la devuelve al *backend*, que se encarga de validar su formato.

5. Si el formato es correcto, la pregunta se guarda en la base de datos.

6. Si no lo es, se devuelve un mensaje de error al *backend*.

Este flujo refleja cómo la generación está condicionada a la correcta estructura de la respuesta del modelo, ya que de lo contrario la pregunta no puede almacenarse ni mostrarse al estudiante.

El segundo diagrama de secuencia es acerca de la evaluación de criterios por parte del estudiante.

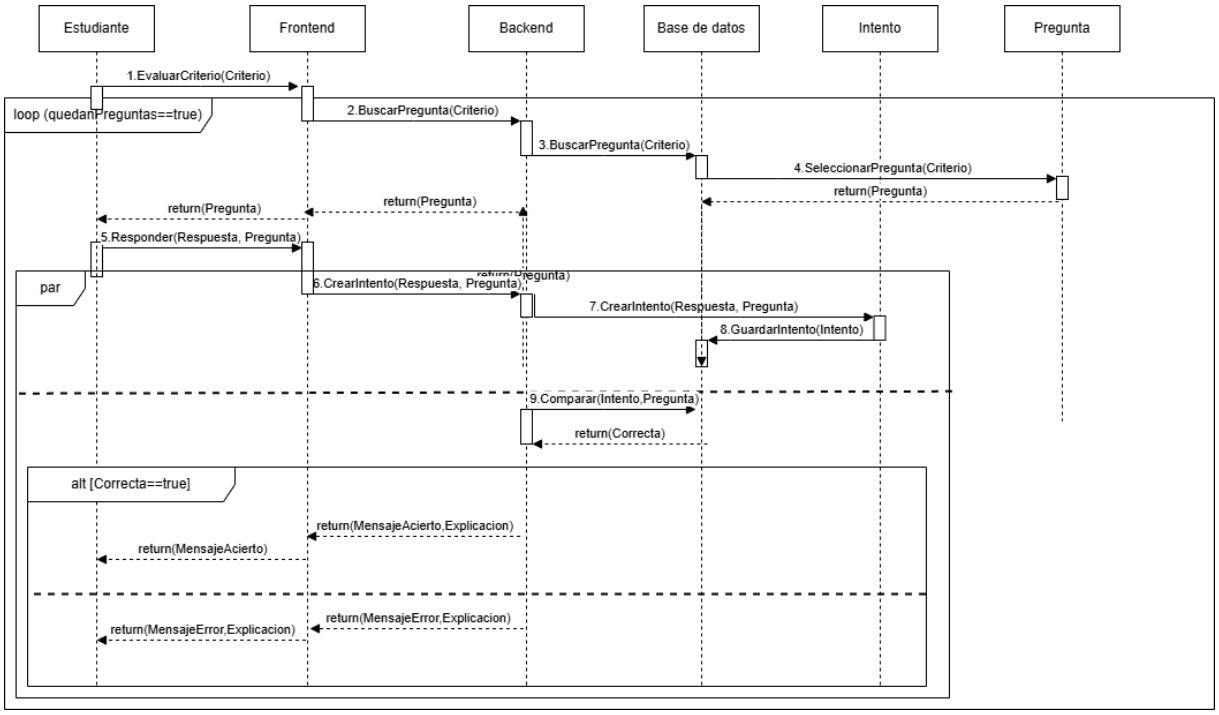

Figura 4.13: Diagrama de secuencia de evaluación de criterio

Este segundo diagrama representa el proceso completo que sigue un estudiante al responder preguntas asociadas a un criterio. Se detalla la lógica de evaluación y el almacenamiento del intento:

1. El estudiante elige evaluar un criterio. Mientras queden preguntas pendientes, el proceso se repite.

2. El *frontend* solicita al *backend* una pregunta correspondiente al criterio.

3. Esta solicitud se reenvía a la base de datos, que selecciona una pregunta disponible y la devuelve.
4. El estudiante responde, y el *frontend* envía esta respuesta junto con la pregunta al *backend*.
5. El *backend* crea un intento, que es guardado en la base de datos.
6. Posteriormente se compara el intento con la respuesta correcta de la pregunta.
7. Si es correcta, se devuelve un mensaje de acierto acompañado de una explicación.
8. Si es incorrecta, se envía un mensaje de error con su explicación correspondiente.

Este diagrama refleja la lógica de validación automática y generación de retroalimentación inmediata, así como la importancia de registrar todos los intentos para permitir seguimiento y estadísticas sobre el progreso del estudiante.

4.2.6. Diagrama de estado

Es importante tener en cuenta que existen distintos tres estados posibles de una pregunta, que corresponden a su estado inicial de generada, y los dos casos posibles, que sea guardada en la base de datos o no, según el formato con el que sea creada por el modelo.

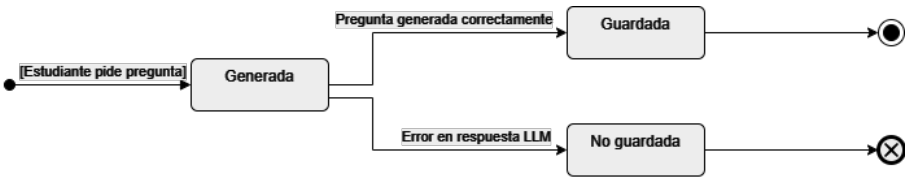

Figura 4.14: Diagrama de estado de las preguntas

4.3. Implementación

Inicialmente se usó Llama 2, posteriormente Llama 3 y al final del proyecto se probó a hacer el cambio a Llama 3.1 para obtener los mejores resultados posibles. Tras hacer el *fine-tuning* de Llama 3.1 [50], haciendo pruebas no ha funcionado mejor, de hecho cumple menos el formato de las preguntas que en el caso de Llama 3, y se guarda un porcentaje inferior de preguntas con este nuevo modelo en comparación con Llama 3. Se probó este cambio y aportó personalmente un aprendizaje pero finalmente no se implementó.

El *backend* tiene *endpoints* para:

- Mostrar todas las historias de usuario de un objetivo de aprendizaje.
- Mostrar todos los criterios de aceptación de una historia de usuario.
- Hacer una solicitud al *docker* para crear una pregunta nueva.
- Hacer las solicitudes al *docker* para crear una pregunta nueva de cada criterio de aceptación de una historia de usuario.

- Comprobar el estado de generación de nuevas preguntas.

- Pedir todas las preguntas existentes de una historia de usuario (o solo las pendientes y falladas).

- Guardar los intentos realizados.

- Varios *endpoints* para tomar todos los datos necesarios para mostrar los *dashboards*.

Esto cubre todas las necesidades que surgen en el uso de la aplicación.

La página web se distribuye principalmente en seis páginas distintas, una de inicio y otras cinco páginas, una de cada objetivo de aprendizaje, tal y como habíamos planificado y expresado en los *mockups*.

La página de inicio vemos que tiene los *dashboards* definidos en el *mockup* correspondiente.

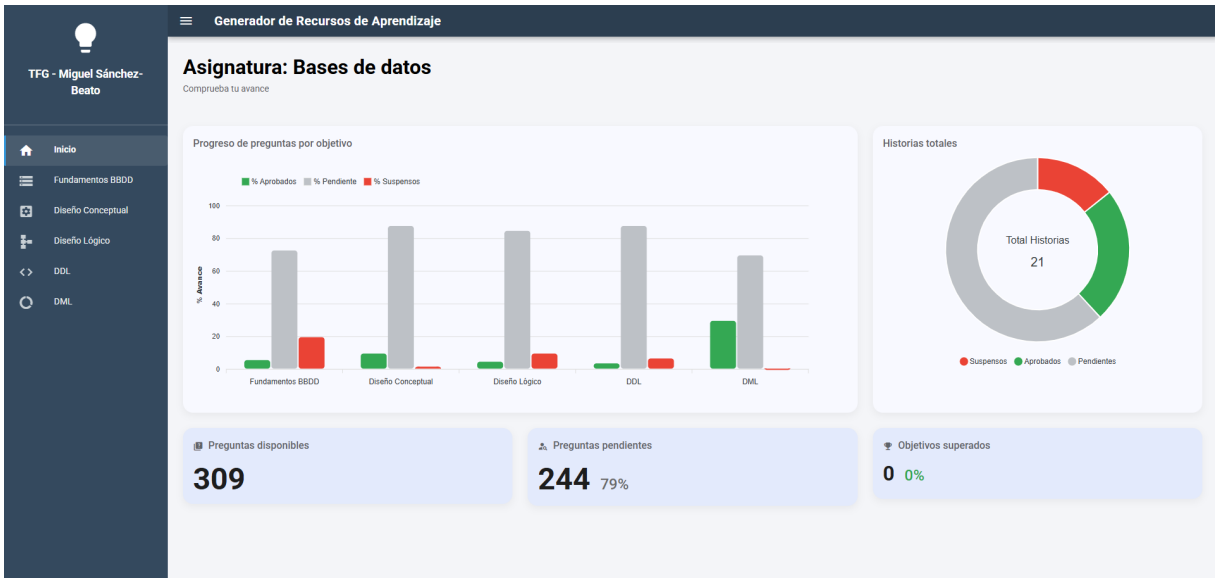

Figura 4.15: Menú de inicio

Cuando vamos a una de las cinco páginas que corresponden a los cinco objetivos de la asignatura «Sistemas de Bases de Datos» podemos encontrar todas las historias de usuario, que son desplegables y en ella encontramos sus criterios de aceptación. Estos datos se cargan de la base de datos de forma dinámica. Contamos con *dashboards* correspondientes tanto a las historias de usuario, a los criterios de aceptación y a las preguntas disponibles, como fue planificado en los *mockups*. Además, existe un botón para generar una pregunta de cada criterio de aceptación de la historia de usuario.

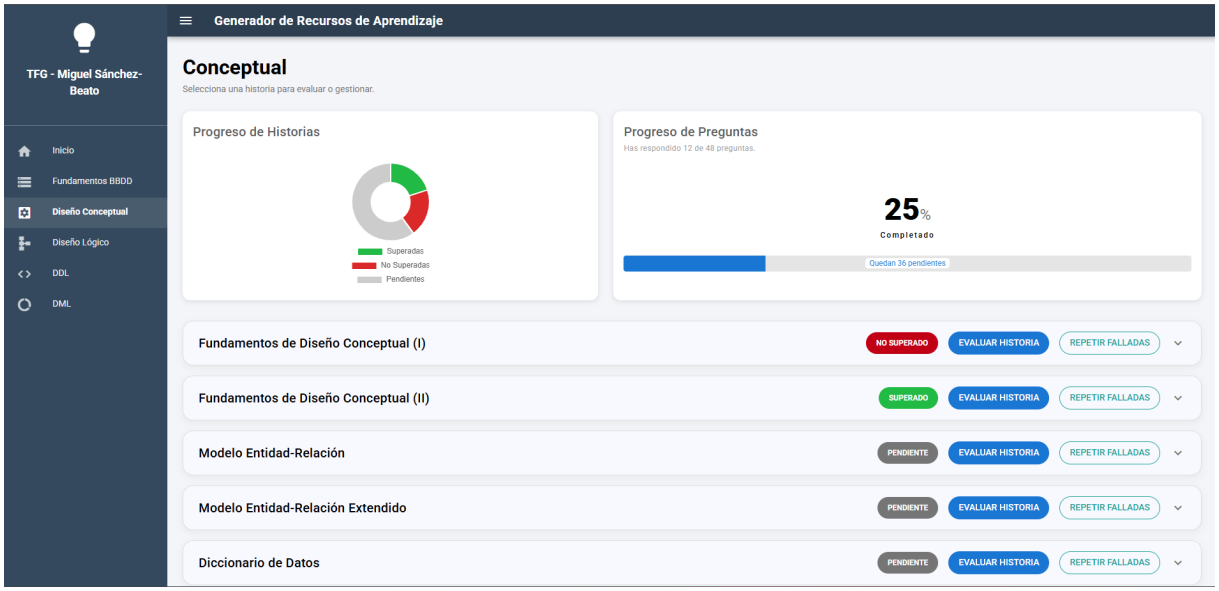

Figura 4.16: Menú historia de usuario

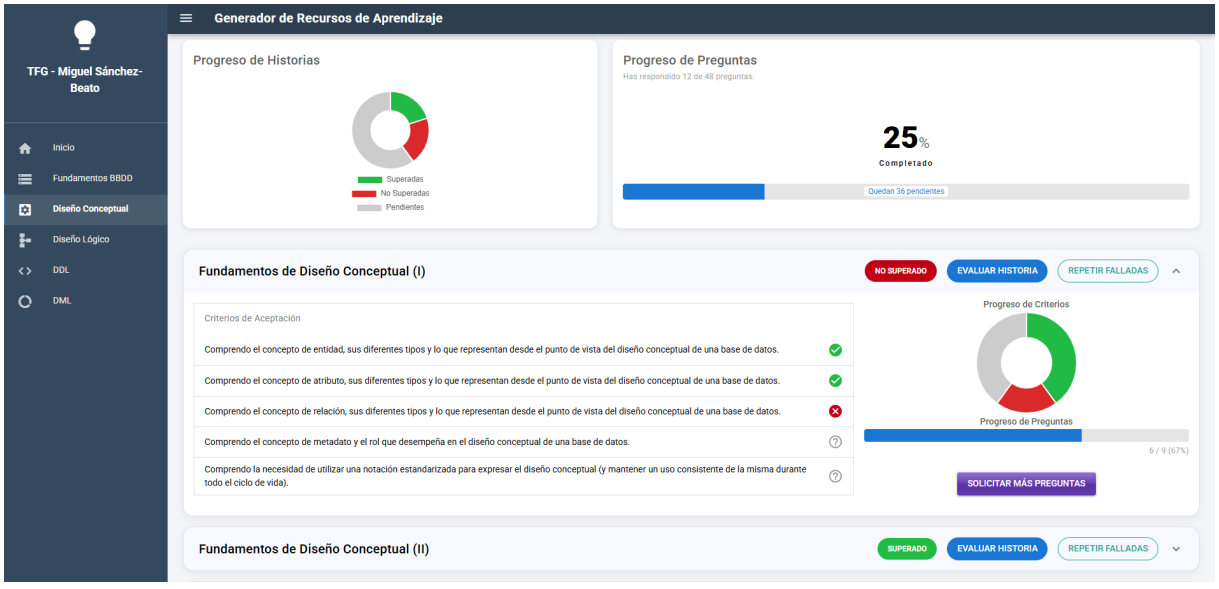

Figura 4.17: Menú historia de usuario extendido

Al pulsar el botón de evaluar historia, vamos a la siguiente página, en la que podemos ver las distintas opciones de las preguntas correspondientes a los criterios de aceptación de esa historia de usuario.

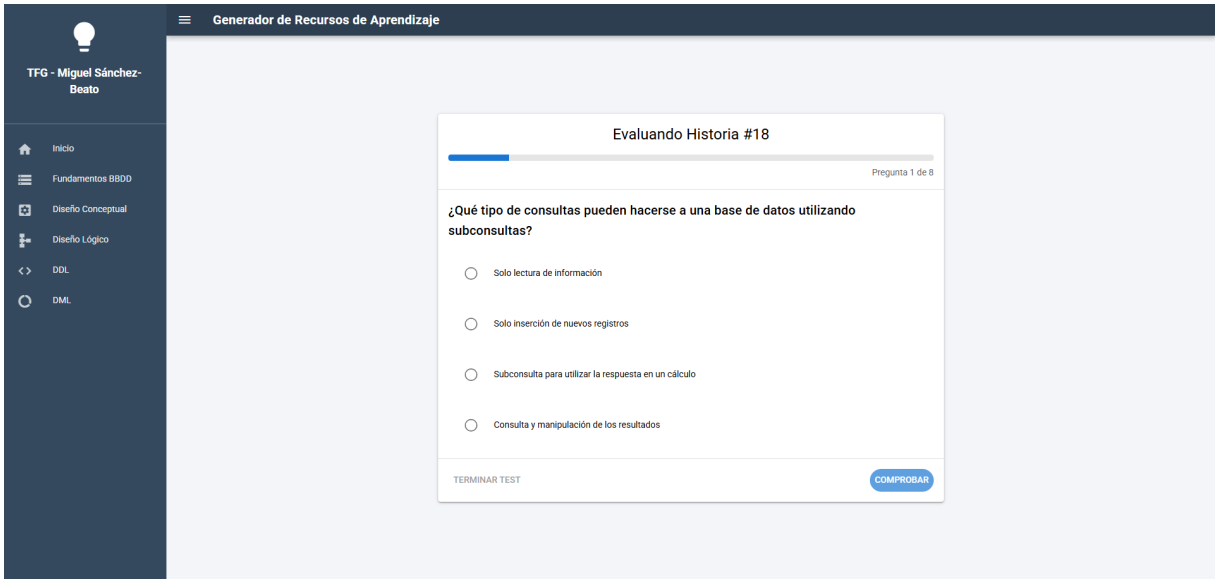

Figura 4.18: Pregunta

Al fallar se colorea de rojo la pregunta elegida, y en verde la correcta. La explicación aparece en caso de acertar o fallar indiferentemente.

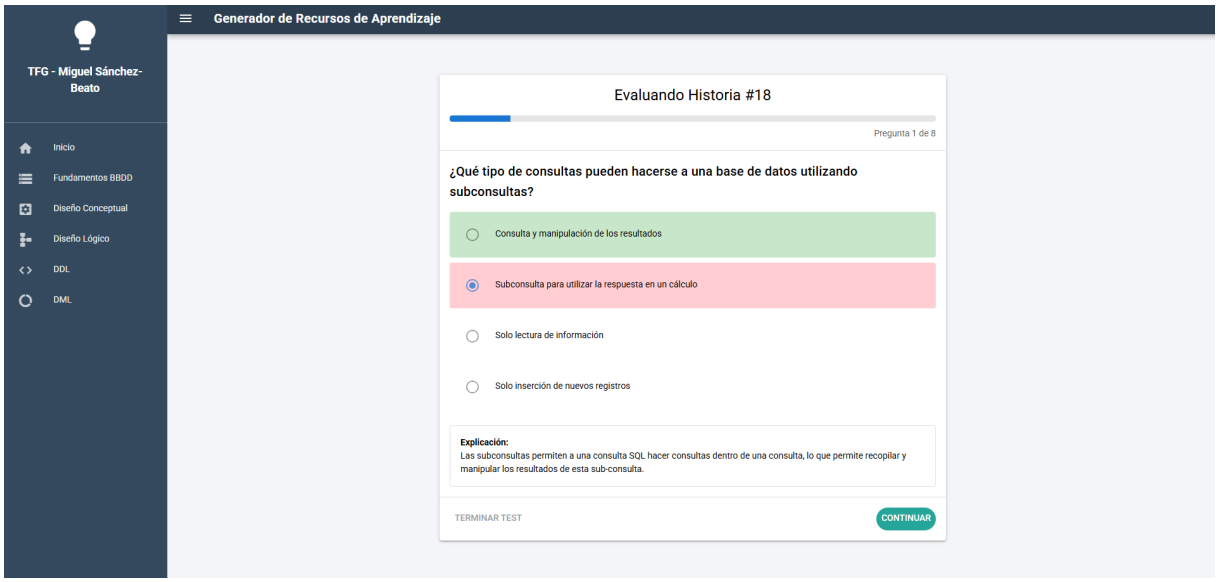

Figura 4.19: Pregunta fallada

Cuando se pulsa el botón «TERMINAR TEST» o terminamos todas las preguntas de la historia de usuario aparece una página que resume los resultados del test, nos notifica que se ha guardado el intento correctamente.

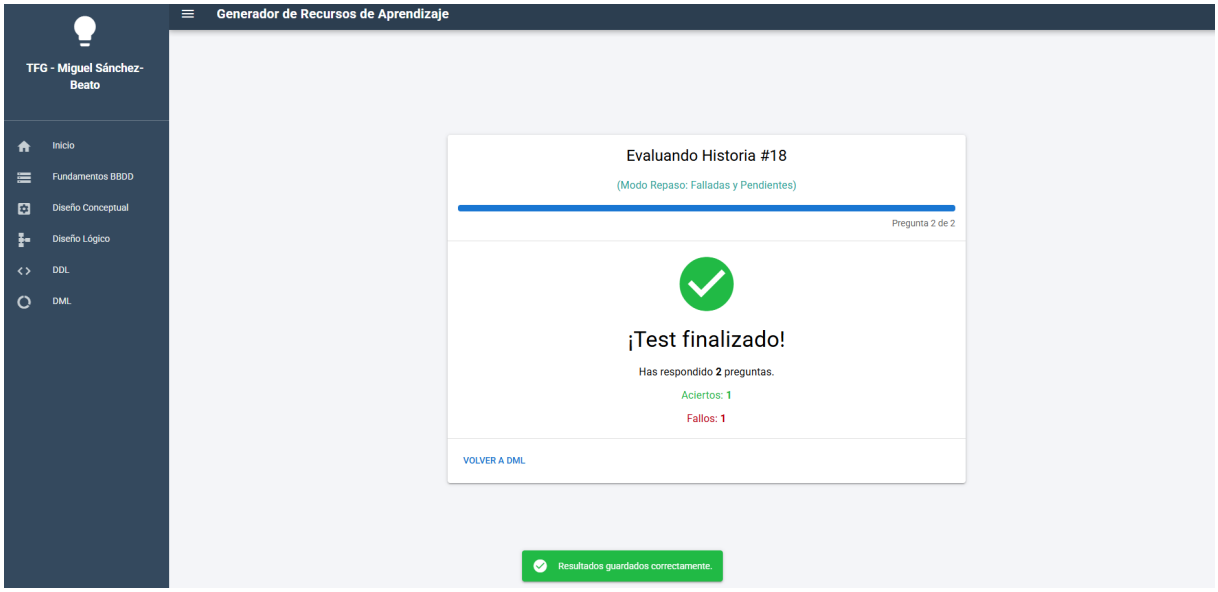

Figura 4.20: Resultados fin del test en modo repaso

Por último, existen dos tipos de notificación más a parte de la verde que confirma que todo ha salido correctamente. Una notificación azul que indica que se están generando preguntas, y otra roja que indica que algo no ha salido correctamente. La azul aparece al darle al botón de «SOLICITAR MÁS PREGUNTAS», y además asegura que no se puedan pedir más preguntas mientras está generando preguntas nuevas.

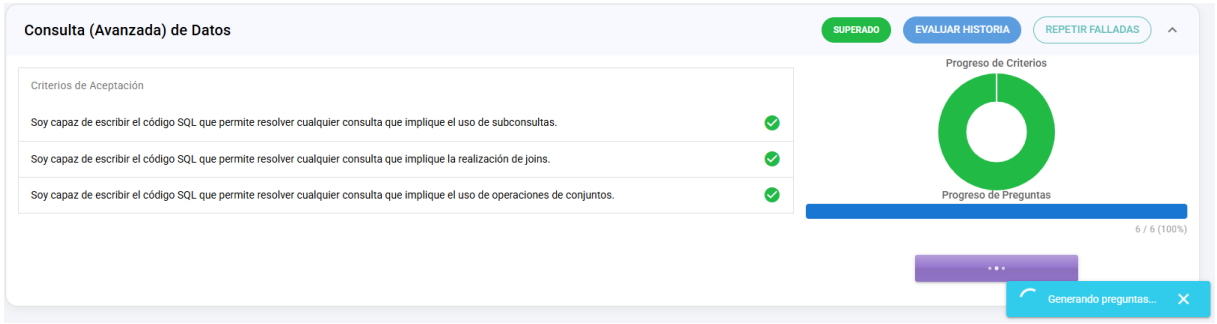

Figura 4.21: Notificación de generando preguntas

Al forzar un error, que se puede conseguir forzando un fallo del *docker* aparece la notificación roja.

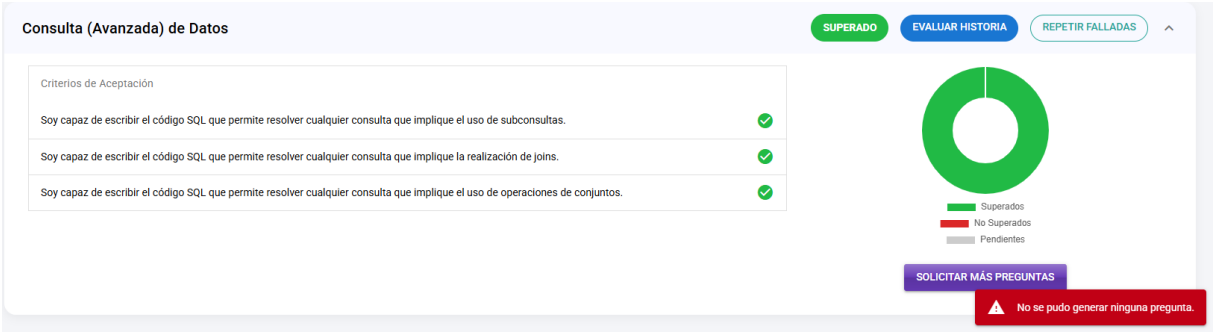

Figura 4.22: Ejemplo de error

4.4. Pruebas

Se utilizó la herramienta Postman para realizar pruebas de integración entre los distintos servicios. Se verificó la correcta comunicación entre el *frontend* y el *backend*, y entre el *backend* y la API del LLM. Estas pruebas fueron cruciales para asegurar que los *endpoints* de la API (descritos en la sección 4.3) respondían correctamente a las peticiones HTTP y manejaban los datos de forma adecuada, por ejemplo, al guardar una pregunta generada (**RF-3**).

Pruebas Funcionales y de Sistema

Una vez integrados los componentes, se llevaron a cabo pruebas funcionales *end-to-end* para validar los casos de uso principales. Estas pruebas manuales cubrieron:

- *Realización de tests (**CU-1**, **CU-2**, **CU-3**)*: Se simuló el flujo completo de un estudiante respondiendo un test, comprobando la correcta evaluación de las respuestas y la presentación de la retroalimentación.
- *Visualización de progreso (**CU-4**)*: Se validó que los datos mostrados en los *dashboards* eran precisos y se actualizaban correctamente tras realizar un test (**RF-4**).
- *Modo repaso (**CU-5**)*: Se comprobó que la funcionalidad para repetir preguntas falladas seleccionaba únicamente las preguntas correspondientes.
- *Generación de preguntas (**CU-6**)*: Se verificó que el botón *"Solicitar más preguntas"* iniciaba el proceso correctamente y que las notificaciones de estado (*generando*, *error*, *éxito*) funcionaban como se esperaba. Además, se verificó que no se pueda pulsar el botón de generar más preguntas mientras generamos nuevas preguntas.

4.5. Optimización

La optimización de la aplicación forma parte del desarrollo, y consiste en mejorar el resultado con los recursos disponibles. Las principales optimizaciones del proyecto se pueden dividir en las tres capas del sistema, *frontend*, *backend* y LLM.

En la página web también hay optimizaciones visuales, como la posición y tamaño de los *dashboards*, los colores de multitud de elementos, el formato menú desplegable o el tamaño de

letra. Es normal durante el proceso de desarrollo *frontend* realizar este tipo de cambios para obtener un mejor resultado, más atractivo visualmente.

Optimización del Modelo (LLM)

En la parte correspondiente al **LLM**, el ajuste de *prompts* para obtener las mejores soluciones con un formato reconocible ha sido una parte importante del proyecto, que se puede denominar como *Prompt Engineering*. La selección del mejor modelo que se puede ejecutar con los recursos disponibles, la realización de un *fine-tuning* óptimo junto a la decisión del *dataset* base de *fine-tuning* utilizado, y la mejora del *endpoint* para que reconozca formatos más variados y así se guarden más preguntas han sido optimizaciones muy importantes en esta parte del proyecto.

También podemos considerar la cuantización y el uso de llama.cpp optimizaciones, aunque es una decisión que se tomó al inicio del proyecto y es la base del proyecto.

Optimización del Backend

En el servidor, la principal optimización se centró en la gestión de la concurrencia.

El uso de `async/await` con librerías como `mysql2/promise` y `axios` para las **operaciones asíncronas** garantiza que las operaciones de larga duración (consultas a la base de datos y, especialmente, peticiones al LLM) no bloqueen el hilo principal de `Node.js`. Esto permite que la aplicación continúe respondiendo a otros usuarios mientras se genera una pregunta. Está bloqueada la generación de preguntas hasta que se terminan de generar las preguntas solicitadas para no sobrecargar el *docker*, pero durante la creación de preguntas se puede realizar cualquier otra actividad en la aplicación.

Optimización del Frontend

Para garantizar una experiencia de usuario fluida, se implementó la **carga asíncrona de componentes**. Como se mencionó, los *dashboards* no se cargan hasta que son necesarios (por ejemplo, al expandir una historia de usuario). Se utilizan pantallas de carga (*spinners*) para indicar al usuario que los datos se están obteniendo. Estas esperas son muy pequeñas, pero se tienen que gestionar adecuadamente pues podrían no actualizarse los datos de forma correcta y no mostrarse el *dashboard* hasta que realizas otra opción en la página, como podría ser cerrar y abrir de nuevo la historia de usuario.

Parte III

Resultados

Capítulo 5

Experimentación y evaluación

Se han realizado tres estudios distintos acerca de la parte de LLMs. Los objetivos a estudiar han sido: la calidad de las preguntas generadas por Llama 3 antes y después del *fine-tuning*, hacer una estimación del tiempo que tarda el modelo en generar las preguntas y por último qué porcentaje de preguntas con problemas de formato produce.

5.1. Diseño experimental

Para la comparación de la calidad, se ha realizado un documento en el que se presentan 30 preguntas (15 antes del *fine-tuning* y 15 después) de 8 criterios de aceptación diferentes, junto a sus opciones, la opción correcta y la explicación que da Llama 3 a la respuesta de la pregunta. Se limitó el análisis a este número de preguntas debido al considerable esfuerzo manual que requiere la valoración detallada de cada ítem. Las preguntas se presentaron en un documento que incluía el enunciado, las opciones, la respuesta correcta y la explicación proporcionada por el modelo. Cada pregunta fue evaluada según cuatro métricas:

- La pregunta es adecuada para el criterio a valorar: Sí/No.
- Valoración de la pregunta del 1 al 5.
- La respuesta me parece correcta: Sí/No. Se tiene en cuenta que solamente haya una respuesta que sea interpretable como correcta.
- Valoración de la explicación del 1 al 5.

Los encargados de evaluar las preguntas ha sido Miguel Ángel Martínez Prieto, como experto en bases de datos y profesor de la asignatura "Sistemas de Bases de Datos", Miguel Sánchez-Beato Díaz-Hellín, como estudiante que cursó la asignatura con matrícula de honor y creador del proyecto, y el modelo ChatGPT 4o, valorando 10 preguntas cada uno. En el experimento no se indicaba qué preguntas correspondían al antes y al después del *fine-tuning*.

El segundo experimento ha consistido en estudiar el tiempo que tarda el modelo en generar preguntas. Se le han pedido dos tandas de preguntas, una de 34 preguntas y otra de 282 preguntas. En la primera tanda no se ha repetido un criterio dos preguntas seguidas, y en la segunda tanda se ha pedido 3 veces seguidas cada criterio, para ver si esto afecta a la velocidad de generación de preguntas.

En el tercer y último experimento se han estudiado las 316 preguntas del experimento dos y hemos documentado cuales son los errores principales que nos encontramos para intentar darle una solución.

5.1.1. Resultados

El **primer experimento**, acerca del *fine-tuning*, nos deja los siguientes resultados: **Antes del *fine-tuning***:

- El 100 % de las preguntas son adecuadas para el criterio de aprendizaje.
- La valoración media de las preguntas (del 1 al 5) es de 3.8
- En el 46,67 % la respuesta es correcta.
- La valoración media de las explicaciones (del 1 al 5) es de 3.6

Después del *fine-tuning*:

- El 100 % de las preguntas son adecuadas para el criterio de aprendizaje.
- La valoración media de las preguntas (del 1 al 5) es de 4,07
- En el 73,33 % la respuesta es correcta.
- La valoración media de las explicaciones (del 1 al 5) es de 3,6

El **segundo experimento**, acerca del tiempo que tarda en generarse una pregunta, ha aportado los siguientes resultados: Las primeras 34 preguntas tardaron 1 hora 27 minutos y 47 segundos, es decir, cada pregunta unos 154 segundos, que son 2 minutos y 34 segundos. La segunda tanda, de 282 preguntas tardó 6 horas, 39 minutos y 41 segundos, es decir, unos 85 segundos por pregunta, que son 1 minuto y 25 segundos.

El **tercer experimento** estudia las 316 preguntas anteriores y hemos encontrado que:

- 240 preguntas se guardan completas y en español. (75,95 %)
- 26 preguntas están en inglés
- 50 preguntas no tenían un formato compatible con la lógica del *backend*.

La mayoría de las preguntas que no se guardaban se debe a errores en el formato, y otro porcentaje importante estaba en inglés. En algunas ocasiones se daba que no creaba una explicación, otras veces no indicaba la letra de la respuesta correcta sino el texto de la opción correcta, o no indicaba donde empezaba la pregunta o las posibles respuestas. Para el resto de casos, solía ser que establecía la separación de cada parte de la pregunta de forma inesperada, es decir, no seguía el patrón indicado en el *prompt*.

5.2. Discusión de resultados

El **primer experimento**, confirma que el proceso de *fine-tuning* tuvo un impacto positivo y significativo. Aunque la valoración de la explicación no varió, la calidad media de las preguntas mejoró de forma muy importante, pues el porcentaje de respuestas consideradas correctas aumentó del 47 % al 73 %.. La pregunta en todos los casos la consideramos adecuada para el criterio a evaluar, y la valoración de la explicación no ha mejorado.

El dato de que el 26,66 % de las respuestas no son del todo correctas después del *fine-tuning* resulta aun preocupante, y sería interesante conseguir ejecutar mejores modelos que puedan aparecer en un futuro o ya existentes más potentes, o alternativamente construir un *dataset* que mejore más las respuestas del modelo, por ejemplo, con más información acerca de bases de datos.

Del **segundo experimento** se extraen dos conclusiones claras: Por una parte, podemos afirmar que es más rápido generar varias preguntas del mismo criterio seguidas a ir alternando de criterio. Por la otra, en la aplicación el tiempo estimado de generación de una pregunta será de 2 minutos y 35 segundos, porque cuando se solicitan preguntas de una historia de usuario, se pide solamente una de cada criterio.

Por último del **tercer experimento** podemos concluir que sin haber realizado cambios se podía esperar un 75 % de preguntas correctamente guardadas, y no es suficientemente buen dato. Las modificaciones más importantes que se pueden realizar estudiando los fallos del modelo y que han sido implementados con éxito son indicar en el *prompt* que la pregunta tipo test sea en español, y estudiar los distintos casos en los que no conseguía guardar la pregunta y darle solución a los más claros. Con esta metodología actualmente guardamos más del 90 % de las preguntas generadas, dato mucho más positivo.

Capítulo 6

Conclusiones y trabajo futuro

6.1. Conclusiones

Este capítulo final presenta las reflexiones extraídas tras la finalización del TFG considerando distintos aspectos desde una perspectiva técnica y personal. Se evalúa el cumplimiento de los criterios, la influencia de la metodología ASAP, la calidad del producto final y el trabajo futuro.

6.1.1. Perspectiva del proyecto

El producto final ha cumplido las expectativas, logrando implementar todas las funcionalidades planificadas. A lo largo de su desarrollo, el proyecto ha sido adaptado progresivamente a los avances en IA y LLMs para conseguir un resultado que no esté obsoleto tecnológicamente después del largo tiempo de desarrollo. Con respecto a los objetivos planteados en un inicio, recuperamos el enunciado de cada uno y juzgamos si se ha cumplido adecuadamente o no:

- **OBJ-1:** Desarrollar una herramienta educativa que aprovecha el potencial de los LLM.

Este objetivo **se ha cumplido de forma sobresaliente**. Existe una aplicación web completamente funcional que genera preguntas con un LLM y que pueden utilizar estudiantes para autoevaluarse y aprender.

- **OBJ-2:** Ser capaz de generar preguntas de calidad dado un criterio de aceptación

El objetivo **se ha cumplido de forma notable a causa de las limitaciones del *hardware***. Hemos generado más de seiscientas preguntas durante el proceso de desarrollo de la aplicación que tienen una calidad razonable y permiten tras el estudio de los apuntes de la asignatura afianzar los conceptos. Con este modelo no todas las preguntas son de una calidad ideal. Una alternativa sencilla para que tenga mejores preguntas la aplicación es conseguir prestado una tarde un servidor con GPU potente y utilizar un modelo más moderno y con más parámetros para generar una cantidad de preguntas suficientes para que la aplicación sea funcional

- **OBJ-3:** Garantizar una experiencia de usuario accesible e intuitiva.

Este objetivo **se ha alcanzado plenamente**. La página web es clara, visual e intuitiva, incorporando los *dashboards* que más información nos aportan y que se actualizan en tiempo real. En la página al estudiante le queda claro cómo está progresando en la asignatura y cuales son los distintos objetivos de aprendizaje, historias de usuario y criterios de aceptación que debe dominar.

- **OBJ-4:** Diseñar la aplicación con una arquitectura modular y escalable que facilite su ampliación y mantenimiento futuro.

El diseño del sistema **cumple con este requisito fundamental**. El proyecto está construido de tal forma que las ampliaciones que queramos realizar en un futuro sean lo más sencillas posibles de implementar. Está programado de forma que la aplicación es modular, de modo que podríamos modificar una de sus piezas, como podría ser, con especial interés, el *docker* con Llama 3, para utilizar un LLM más potente (o más modesto) sin realizar ninguna adaptación extra.

Desde un punto de vista técnico, el *backend* es completo y contempla todos los *endpoints* necesarios para esta aplicación, con comunicación con el *frontend* y con el *docker*. El uso de *dockers* en la parte de Llama.cpp ha funcionado tal y como se esperaba, reduciendo la complejidad de la ejecución local de un modelo de lenguaje, que en un principio es muy alta, al mínimo.

Esta aplicación la podrían utilizar los estudiantes de la asignatura de `Sistemas de Bases de Datos` y podría ayudarles a comprender la asignatura y a enfrentarse a las preguntas tipo test del examen con más confianza y práctica.

La planificación temporal y metodología ASAP se han desarrollado con éxito durante el proyecto a pesar de los inconvenientes que han surgido durante el desarrollo del mismo. Una metodología bien estructurada y resiliente como ASAP ha ayudado a que se puedan mitigar los efectos de haber dejado en pausa un año el proyecto y retomarlo de forma que se termine en el plazo deseado después de una nueva planificación temporal, pudiendo mantener los objetivos del quinto *sprint* que se hablaron un año antes del fin del proyecto.

6.1.2. Perspectiva personal

Lo que más destaco personalmente del desarrollo de este TFG es que me ha dado una perspectiva y una opinión acerca del uso de la inteligencia artificial generativa en el proceso del aprendizaje. La educación al igual que el mundo laboral se está adaptando a las nuevas tecnologías y este tipo de proyectos plantean posibilidades e ideas a profesores y alumnos. Aportar herramientas y complementos a las clases tradicionales es difícil pues requiere de mucho tiempo, y la inteligencia artificial generativa en el estado actual es capaz de complementar el trabajo del profesor y ayudar al alumno con *feedback* prácticamente automático.

El desarrollo de una aplicación con sus distintas capas ha sido un proceso interesante, en especial por la importancia de la asincronía en el proyecto, pero para mí lo valioso de la aplicación es el trabajo realizado con *dockers* y con LLMs. Son herramientas muy valiosas hoy en día y tenía gran interés por ambas, y he cumplido satisfactoriamente con los objetivos relacionados con estos dos aspectos del Trabajo Fin de Grado.

Además, como ya se ha comentado a lo largo de la memoria, el haber dejado durante un año el Trabajo Fin de Grado por situaciones académicas no ha sido algo ideal pero sí de lo que he aprendido. Todos los proyectos son susceptibles a cambios en las fechas y vivirlo en la práctica me ha hecho estar preparado para afrontar más y distintos inconvenientes propios del desarrollo software, y de la gestión de proyectos en general.

6.2. Trabajo futuro

Este proyecto está abierto a múltiples vías de trabajo futuro. La que más interés me genera ahora que he terminado el proyecto sería generalizar la aplicación a más asignaturas. Un modelo más potente podría no requerir un *fine-tuning* para generar preguntas de calidad y podría valer para la mayoría de asignaturas de casi cualquier carrera. Podría abrirse un apartado en la página web donde importar un documento en el que se detallen los objetivos de aprendizaje, historias de usuario y criterios de aceptación, o incluso delegar en el propio LLM la generación de estos a partir de una guía docente. La página web podría adaptarse a cualquier asignatura añadiendo algunos *endpoints* relativamente sencillos.

También podría implementarse el rol de profesor y que cada alumno tenga su usuario, pues ya existe en la base de datos esta tabla de usuarios. El profesor así podría ver cuales son los puntos débiles de la clase, generar él las preguntas y filtrarlas para que sean de calidad, o introducir él sus propias preguntas si así lo desea.

Otra idea que he tenido mostrando el trabajo a familia y amigos es que la aplicación podría salir del ámbito académico y entrar en el ámbito de la formación profesional. En la mayoría de empresas grandes se da formación a los empleados en distintas materias en forma de cursos, que suelen ser evaluados con exámenes tipo test. Con un modelo potente, podríamos generar multitud de buenas preguntas diferentes para que haya variedad, pues hay, por ejemplo, MOOCs con las respuestas ya filtradas en internet, como pasa con algunos cursos creados por las universidades que otorgan créditos optativos a los estudiantes que lo cursan [4]. También permitiría hacer más formaciones de más temáticas diferentes de forma muy sencilla y barata para el empresario, que tendría el mismo rol que el profesor en la propuesta anterior, y los trabajadores tendrían el rol equivalente al del alumno.

El cambio común a todas estas propuestas es el uso de un modelo más actual y preciso. Esto se podría obtener a través de la capacidad de computación de un servidor con una GPU potente, o utilizando *endpoints* por ejemplo de OpenAI con su modelo GPT-4o, que aunque cuesta dinero según el número de *tokens* del mensaje, para uso no intensivo puede resultar muy barato [38] y sencillo de implementar con resultados óptimos.

Parte IV

Apéndices

Apéndice A

Manual de Instalación

A.1. Requisitos previos

Para poder ejecutar el proyecto de forma local es necesario cumplir con los siguientes requisitos:

- Docker Desktop instalado (versión recomendada 20.10 o superior).
- Node.js instalado (versión recomendada 18.0 o superior).
- npm instalado.
- Mínimo 8 GB de memoria RAM disponible para poder ejecutar el modelo LLM en local.
- Procesador Intel i7 de 7º generación o similar mínimo.
- Conexión a Internet para la descarga inicial de imágenes *docker* y dependencias.

A.2. Clonado del repositorio

El proyecto está alojado en un repositorio de GitHub. Para clonar el repositorio, se debe ejecutar el siguiente comando:

```
git clone Miguelsbdh/TFG
```

Una vez clonado, la estructura del proyecto será la siguiente:

```
cliente/
servidor/
README.md
tfg.sql
```

A.3. Carga de la base de datos

En la carpeta del proyecto encontramos el documento `tfg.sql`. Se recomienda el uso de XAMPP aunque se puede realizar con otras herramientas si son más familiares para el usuario. Dentro de XAMPP se deben activar los módulos de Apache y MySQL.

Una vez hecho esto, acudimos a la base de datos e introducimos la siguiente sentencia sql:

```
-- Crea el usuario para conexiones locales
CREATE USER 'tfg'@'localhost' IDENTIFIED BY 'tfg';
CREATE USER 'tfg'@'%' IDENTIFIED BY 'tfg';
GRANT ALL PRIVILEGES ON tfg.* TO 'tfg'@'localhost';
GRANT ALL PRIVILEGES ON tfg.* TO 'tfg'@'%';
FLUSH PRIVILEGES;
```

Así, en caso de querer utilizar *docker* o *localhost* no vamos a tener ningún problema.

Posteriormente, creamos una base de datos con la siguiente sentencia:

```
CREATE DATABASE tfg;
```

Por último, entramos dentro de esa base de datos y le damos al botón de **IMPORTAR**, que se suele encontrar en la parte superior. Al pulsar el botón, seleccionamos el documento `tfg.sql`, que estará en la carpeta del proyecto, y le damos a importar.

A.4. Instalación de dependencias

Desde la carpeta raíz del proyecto, se deben instalar las dependencias necesarias para el *backend* y el *frontend*. En Visual Studio abrimos la carpeta del proyecto y creamos dos consolas, una para el *backend* y otra para el *frontend*. Escribimos lo siguiente en cada una:

Consola para el backend

```
cd servidor
npm install
```

Consola para el frontend

```
cd cliente
npm install
```

A.5. Ejecución del modelo LLM con Docker

El modelo Llama 3 con el *fine-tuning* y cuantizado lo podemos descargar en *Hugging Face*: Miguelsbdh/llama-3-finetuned-bases-de-datos

Basta con descargar el archivo `llama-3-finetuned-bases-de-datos-unsloth.Q5_K_M.gguf`.

Para ejecutar Llama 3 con *llama.cpp* es necesario iniciar un docker con el siguiente comando:

```
docker run --rm -it -p 8000:8000 \
-v /c/llama.cpp/models:/models \
-e MODEL=/models/llama-3-finetuned-bases-de-datos-unsloth.Q5\_K\_M.gguf \
ghcr.io/abetlen/llama-cpp-python:v0.3.1
```

Notas importantes:

- Si es la primera vez que se ejecuta este contenedor, Docker Desktop descargará automáticamente la imagen. No es necesario realizar un *docker pull* manual.
- La ruta `/c/llama.cpp/models` debe ser modificada según la ubicación local donde se encuentre almacenado el modelo.

A.6. Arranque del backend

Desde la carpeta `"servidor"`, ejecutar el siguiente comando:

```
npm start
```

El *backend* quedará escuchando en el puerto correspondiente, el 9001.

A.7. Arranque del frontend

Desde la carpeta `cliente`, ejecutar el siguiente comando:

```
npm run dev
```

El *frontend* se desplegará en el navegador en la dirección `http://localhost:9000`.

A.8. Solución a errores frecuentes

- **Problemas con *docker*:** Verificar que la carpeta y el modelo existen y están en la ruta indicada. También es importante comprobar que Docker Desktop esté correctamente instalado y en ejecución. El error más común es que si no está abierta la aplicación de Docker Desktop no se puede ejecutar el `docker run`.
- **Conflictos de puertos:** Si el puerto 8000 ya está en uso, se deberá liberar o modificar el puerto en la ejecución del *docker*.
- **Error al instalar dependencias:** Verificar la versión de Node.js y utilizar la recomendada para evitar conflictos.
- **Fallo en la conexión *backend* – modelo:** Asegurarse de que el contenedor esté corriendo y que la configuración de la URL en el *backend* apunte correctamente al puerto 8000.

Apéndice B

Manual de Usuario

B.1. Introducción

Esta aplicación web está diseñada para que estudiantes de la asignatura *Sistemas de Bases de Datos* puedan autoevaluarse mediante preguntas tipo test generadas automáticamente. El objetivo es proporcionar una herramienta que permita a los estudiantes repasar y comprobar el nivel de dominio de los distintos objetivos de aprendizaje de la asignatura.

La aplicación está pensada para un uso sencillo e intuitivo, sin necesidad de registro.

B.2. Estructura y Navegación

La aplicación consta de seis páginas principales:

- Una **página de inicio** con indicadores de progreso (*dashboards*).
- Cinco páginas específicas, cada una correspondiente a un **objetivo de aprendizaje** de la asignatura.

B.2.1. Página de Inicio

En esta página el usuario puede visualizar de forma global:

- El progreso por objetivos de aprendizaje.
- El progreso por historias de usuario.
- La cantidad total de preguntas respondidas, pendientes y falladas.
- El porcentaje de objetivos superados.

Estos datos permiten al usuario visualizar su progreso en la asignatura y decidir de qué objetivo de aprendizaje evaluarse a continuación

B.2.2. Páginas de Objetivos

Cada página correspondiente a un objetivo contiene:

- Todas las **historias de usuario** relacionadas, que son desplegables.
- Dentro de cada historia, los **criterios de aceptación** y el número de preguntas disponibles.
- Un botón para **evaluar** una historia de usuario.
- Un botón para **solicitar más preguntas**, que genera nuevas preguntas tipo test para los criterios de aceptación seleccionados. Una vez pulsado se guardarán las preguntas a un ritmo estimado de dos minutos y medio por criterio.

B.2.3. Evaluación de Historias de Usuario

Al comenzar la evaluación, el usuario responde a una serie de preguntas tipo test, cada una con:

- **Cuatro opciones de respuesta** (solo una correcta).
- Un sistema visual de corrección instantánea:
  - La respuesta seleccionada se muestra en **rojo** si es incorrecta.
  - La respuesta correcta se muestra en **verde**.
- Una explicación de la respuesta correcta, independientemente de si la respuesta ha sido acertada o no.

Al finalizar el test:

- Se muestra una **página resumen** con los resultados del intento.
- Se notifica que el intento ha sido guardado correctamente.

El usuario puede repetir las preguntas falladas o pendientes tantas veces como desee. Si se evalúa de nuevo de todas y falla alguna que acertó, aparecerá en la lista de preguntas falladas.

B.2.4. Notificaciones del Sistema

- **Notificación verde:** Operación realizada con éxito.
- **Notificación azul:** Se están generando nuevas preguntas. Mientras esta notificación esté activa, no se pueden solicitar más preguntas.
- **Notificación roja:** Se ha producido un error. Normalmente asociado a problemas de comunicación con la base de datos o el contenedor.

B.3. Instrucciones de Uso Paso a Paso

1. Acceder a la página web de la aplicación.
2. Consultar el progreso general en la página de inicio.
3. Seleccionar uno de los objetivos de aprendizaje.
4. Desplegar una historia de usuario y revisar los criterios de aceptación.
5. Solicitar más preguntas para los criterios de aceptación. (Opcional)
6. Pulsar el botón **Evaluar historia** para comenzar el test.
7. Responder las preguntas tipo test seleccionando una de las cuatro opciones.
8. Revisar los resultados al finalizar el test.
9. Repetir las preguntas falladas o pendientes si se desea.

B.4. Recomendaciones de Uso

- Si se solicita la generación de nuevas preguntas, esperar a que desaparezca la notificación azul antes de pedir nuevas preguntas. La aplicación se puede seguir utilizando sin problema.
- Si alguna pregunta no tiene sentido o es incorrecta, se puede acudir a la base de datos y borrar esa pregunta.

B.5. Errores Comunes y Soluciones

Problema	**Posible causa**	**Solución**
Error al generar preguntas	El *docker* no está en ejecución o no ha terminado de arrancar	Comprobar que el *docker* está funcionando y esperar unos cinco minutos desde su inicio
No se generan todas las preguntas solicitadas	Fallo puntual en la generación o en el modelo LLM	Solicitar más preguntas de nuevo para completar los criterios
La página no carga	Error de conexión entre la aplicación y la base de datos	Asegurar que está disponible el servidor y la base de datos.

Tabla B.1: Errores comunes y soluciones

Bibliografía

[1] abetlen. *llama-cpp-python: Python bindings for llama.cpp*. https://github.com/abetlen/llama-cpp-python. 2023.

[2] Amazon Web Services, Inc. *Contenedores de Docker | ¿Qué es Docker? | AWS*. Amazon Web Services, Inc. URL: https://aws.amazon.com/es/docker/.

[3] Guido Appenzeller. «Welcome to LLMflation – LLM inference cost is going down fast». en-US. En: *Andreessen Horowitz* (nov. de 2024).

[4] *Apuntes Variados de la Escuela Superior de Informática - UGR*. https://wuolah.com/apuntes/apuntes-variados?communityId=13824&f_course=4. Wuolah, 2025.

[5] David Baidoo-Anu y Leticia Owusu Ansah. «Education in the Era of Generative Artificial Intelligence (AI): Understanding the Potential Benefits of ChatGPT in Promoting Teaching and Learning». En: *Journal of AI* 7.1 (2023, 31 diciembre), págs. 52-62.

[6] P. Bourque y R.E. Fairley. *Guide to the Software Engineering Body of Knowledge, Version 3.0*. ACM-IEEE. 2014.

[7] Rahul Chand. *A guide to LLMs for the GPU poor*. https://rahulschand.github.io/gpu_poor/. 2024.

[8] Kamal R. Chowdhary. *Natural Language Processing*. Springer, 2020, págs. 603-649. DOI: 10.1007/978-81-322-3972-7_19. URL: https://doi.org/10.1007/978-81-322-3972-7_19.

[9] Colaboradores de Wikipedia. *Node.js*. https://es.wikipedia.org/wiki/Node.js. Wikipedia, La Enciclopedia Libre. Dic. de 2024.

[10] Colaboradores de Wikipedia. *Transformador generativo preentrenado*. 2024, 5 febrero. URL: https://es.wikipedia.org/wiki/Transformador_generativo_preentrenado.

[11] Erika Corona, Filippo Eros Pani et al. «A review of lean-kanban approaches in the software development». En: *WSEAS transactions on information science and applications* 10.1 (2013), págs. 1-13.

[12] Tim Dettmers et al. «GPTQ: Accurate Post-Training Quantization for Generative Pre-trained Transformers». En: *arXiv preprint arXiv:2210.17323* (2022).

[13] Tim Dettmers et al. «LLM.int8(): 8-bit Matrix Multiplication or Transformers at Scale». En: *arXiv preprint arXiv:2208.07339* (2022).

[14] S. Elkins et al. «How Useful Are Educational Questions Generated by Large Language Models?» En: *Artificial Intelligence in Education*. Ed. por N. Wang et al. Vol. 1831. Communications in Computer and Information Science. Springer, Cham, 2023. DOI: 10.1007/978-3-031-36336-8_83. URL: https://doi.org/10.1007/978-3-031-36336-8_83.

[15] Y. Gao et al. «An Investigation of Applying Large Language Models to Spoken Language Learning». En: *Applied Sciences* 14.1 (2023, 26 diciembre), pág. 224. DOI: 10.3390/app14010224. URL: https://doi.org/10.3390/app14010224.

[16] Gerganov Gergő. *llama.cpp*. 2023.

[17] Glassdoor. *Sueldo: Desarrollador Full Stack Junior en España 2025*. Sitio web. 2025. URL: https://www.glassdoor.es/Sueldos/desarrollador-full-stack-junior-sueldo-SRCH_KO0,31.htm.

[18] Glassdoor. *Sueldo: Ingeniero Junior AI en España 2025*. Sitio web. 2025. URL: https://www.glassdoor.es/Sueldos/ingeniero-junior-ai-sueldo-SRCH_KO0,19.htm.

[19] Ian Goodfellow, Yoshua Bengio y Aaron Courville. *Deep Learning*. http://www.deeplearningbook.org. MIT Press, 2016.

[20] Edward J. Hu, Yelong Shen, Phillip Wallis et al. «LoRA: Low-Rank Adaptation of Large Language Models». En: *arXiv preprint arXiv:2106.09685* (2021).

[21] Mark Humor. «Understanding "tokens" and tokenization in large language models». En: *Medium* (2024, 8 febrero). URL: https://blog.devgenius.io/understanding-tokens-and-tokenization-in-large-language-models-1058cd24b944.

[22] Wei Jiang. «Exploring Naming Conventions (and Defects) of Pre-trained Deep Learning Models in Hugging Face and Other Model Hubs». En: *arXiv.org* (2023, 2 octubre). URL: https://arxiv.org/abs/2310.01642.

[23] Jooble. *Gestor de proyectos salario - Comprueba gestor de proyectos salario promedio en Jooble*. Sitio web. 2025. URL: https://es.jooble.org/salary/gestor-de-proyectos.

[24] Dan Jurafsky y James H. Martin. *Speech and Language Processing*. 2024, 3 febrero. URL: https://web.stanford.edu/~jurafsky/slp3/3.pdf.

[25] Jan Marcel Kezmann. *Master the Art of Quantization – A Practical Guide*. https://medium.com/@jan_marcel_kezmann/master-the-art-of-quantization-a-practical-guide-e74d7aad24f9. Accessed: 2025-07-10. 2024.

[26] Yu-Ju Lan y Nian-Shing Chen. «TeachersÁgency in the Era of LLM and Generative AI: Designing Pedagogical AI Agents». En: *Educational Technology & Society* 27.1 (2024, enero), págs. I-XVIII. DOI: 10.30191/ETS.202401_27(1).PP01. URL: https://doaj.org/article/006f91ff651b41df9f447e428007e4c1.

[27] Percy Liang et al. «Holistic Evaluation of Language Models». En: *arXiv preprint arXiv:2211.09110* (2022).

[28] *Llama.CPP Benchmark - OpenBenchmarking.org*. https://openbenchmarking.org/test/pts/llama-cpp&eval=6ec440762e0ebc1af4c8a53ca2fae5a7403dd9fd.

[29] Mercedes Marqués. *Bases de Datos*. Castelló de la Plana: Publicacions de la Universitat Jaume I, 2011. ISBN: 978-84-693-0146-3.

[30] Miguel A. Martínez-Prieto et al. «Una metodología basada en prácticas ágiles para la realización de Trabajos Fin de Grado». En: *Actas de las Jornadas de Enseñanza Universitaria de la Informática (JENUI)* vol. 8 (2023), págs. 367-374. URL: https://aenui.org/actas/pdf/JENUI_2023_047.pdf.

[31] S. McAleese. *Retrospective on 'GPT-4 PredictionsÁfter the Release of GPT-4*. 2023, 17 marzo. URL: https://www.lesswrong.com/posts/iQx2eeHKLwgBYdWPZ/retrospective-on-gpt-4-predictions-after-the-release-of-gpt.

[32] Meta. *Meta-Llama-3-8B Model Card*. https://huggingface.co/meta-llama/Meta-Llama-3-8B. 2024.

[33] Meta AI. *Introducing Meta Llama 3: The most capable openly available LLM to date*. 2024. URL: https://ai.meta.com/blog/meta-llama-3/.

[34] Meta AI. *Introducing Meta Llama 3.1: Our most capable, efficient, and open models to date*. 2024. URL: https://ai.meta.com/blog/meta-llama-3-1/.

[35] Miguelsbdh. *llama-3-finetuned-bases-de-datos*. https://huggingface.co/Miguelsbdh/llama-3-finetuned-bases-de-datos. 2025.

[36] Miguelsbdh. *training-tfg Dataset*. https://huggingface.co/datasets/Miguelsbdh/training-tfg. 2025.

[37] Node.js Foundation. *Node.js — the V8 JavaScript Engine*. 2024. URL: https://nodejs.org/en/learn/getting-started/the-v8-javascript-engine.

[38] OpenAI. *Precios de la API de OpenAI*. 2025. URL: https://openai.com/es-ES/api/pricing/.

[39] Michael S. Orenstrakh. «Detecting LLM-Generated Text in Computing Education: A Comparative Study for ChatGPT Cases». En: *arXiv.org* (2023, 10 julio). URL: https://arxiv.org/abs/2307.07411.

[40] R. Pressman. *Ingeniería del Software*. McGraw-Hill, 2014.

[41] Quasar Framework. *Quasar Components*. 2024. URL: https://quasar.dev/components.

[42] Quasar Framework. *Why Quasar?* 2024. URL: https://quasar.dev/introduction-to-quasar.

[43] *Quasar on Best of JS*. https://bestofjs.org/projects/quasar.

[44] F. Rocha. *Costos ambientales de la Inteligencia Artificial*. 2023, 13 junio. URL: https://mitsloanreview.mx/colaborador/costos-ambientales-de-la-inteligencia-artificial/.

[45] Ken Schwaber y Jeff Sutherland. *La Guía Scrum: La Guía Definitiva de Scrum: Las Reglas del Juego*. 2020, noviembre. URL: https://scrumguides.org/docs/scrumguide/v2020/2020-Scrum-Guide-Spanish-European.pdf.

[46] Seguridad Social. *Bases y Tipos de Cotización - Ministerio de Inclusión, Seguridad Social y Migraciones*. Sitio web. 2025. URL: https://www.seg-social.es/wps/portal/wss/internet/Trabajadores/CotizacionRecaudacionTrabajadores.

[47] Adi Shamir. «How to share a secret». En: *Communications of the ACM* 22 (1979).

[48] Steve J Simister. «Usage and benefits of project risk analysis and management». En: *International Journal of Project Management* 12.1 (1994), págs. 5-8. ISSN: 0263-7863. DOI: https://doi.org/10.1016/0263-7863(94)90003-5. URL: https://www.sciencedirect.com/science/article/pii/0263786394900035.

[49] R. Sole. *Se avecina una nueva escasez de gráficas gaming... ahora por la IA*. 2023, 3 noviembre. URL: https://hardzone.es/noticias/tarjetas-graficas/escasez-gpu-inteligencia-artificial/.

[50] Miguel Sánchez-Beato. *Llama-3.1_TFG*. https://huggingface.co/Miguelsbdh/final_model_quantized. Hugging Face. 2025.

[51] Unsloth Team. *Fine-Tuning Guide*. https://docs.unsloth.ai/get-started/fine-tuning-guide. 2024.

[52] Unsloth Team. *LoRA Hyperparameters Guide*. https://docs.unsloth.ai/get-started/fine-tuning-guide/lora-hyperparameters-guide. 2024.

[53] Teradata. *Large Language Model*. 2023, 18 octubre. URL: https://www.teradata.com/insights/ai-and-machine-learning/large-language-model.

[54] UnslothAI. *Fine-tuning Llama 3.1 (8B) with Alpaca Dataset*. https://colab.research.google.com/github/unslothai/notebooks/blob/main/nb/Llama3.1_(8B)-Alpaca.ipynb. 2025.

[55] Ashish Vaswani. «Attention is All You Need». En: *arXiv.org* (2017, 12 junio). URL: https://arxiv.org/abs/1706.03762.

[56] Z. Wang et al. «Towards Human-Like Educational Question Generation with Large Language Models». En: *Artificial Intelligence in Education. AIED 2022*. Ed. por M.M. Rodrigo et al. Vol. 13355. Lecture Notes in Computer Science. Springer, Cham, 2022. DOI: 10.1007/978-3-031-11644-5_13. URL: https://doi.org/10.1007/978-3-031-11644-5_13.

[57] Wikipedia. *Ingeniería de Instrucciones*. 2024, 21 febrero. URL: https://es.wikipedia.org/wiki/Ingenier%C3%ADa_de_instrucciones.

[58] Wikipedia. *Inteligencia Artificial*. 2024, 20 febrero. URL: https://es.wikipedia.org/wiki/Inteligencia_artificial.

[59] Colaboradores de Wikipedia. *Benjamin Bloom*. 2023, 25 diciembre. URL: https://es.wikipedia.org/wiki/Benjamin_Bloom#:~:text=La%20taxonom%C3%ADa%20de%20Bloom%20para,%2C%20An%C3%A1lisis%2C%20S%C3%ADntesis%20y%20Evaluaci%C3%B3n.

[60] Fuzhao Xue et al. «To Repeat or Not To Repeat: Insights from Scaling LLM under Token-Crisis». En: *Advances in Neural Information Processing Systems (NeurIPS)*. 2023. URL: https://proceedings.neurips.cc/paper_files/paper/2023/hash/b9e472cd579c83e2f6aa3459f46Abstract-Conference.html.

[61] Fuzhao Xue et al. «To Repeat or Not To Repeat: Insights from Scaling LLM under Token-Crisis». En: *Proceedings of the 37th Conference on Neural Information Processing Systems (NeurIPS 2023)*. 2023. URL: https://proceedings.neurips.cc/paper_files/paper/2023/file/b9e472cd579c83e2f6aa3459f46aac28-Paper-Conference.pdf.

[62] yahma. *Alpaca-Cleaned Dataset*. https://huggingface.co/datasets/yahma/alpaca-cleaned. 2023.

[63] Xia Yang. «Navigating Dataset Documentations in AI: A Large-Scale Analysis of Dataset Cards on Hugging Face». En: *arXiv.org* (2024, 24 enero). URL: https://arxiv.org/abs/2401.13822.