Universidad Politécnica
de Madrid

Escuela Técnica Superior de Ingenieros Informáticos

Grado en Ingeniería Informática

Trabajo Fin de Grado

Automatización de Procesos con Docker, n8n y Modelos de IA: Generación y Evaluación Automática de Exámenes Basada en RAG y Modelos de IA para Entornos Educativos

Autor: Diego Luján Raboso

Tutora: Adriana Toni Delgado

Madrid, junio 2025

Este Trabajo Fin de Grado se ha depositado en la ETSI Informáticos de la Universidad Politécnica de Madrid para su defensa.

Trabajo Fin de Grado
Grado en Ingeniería Informática
Título: Automatización de Procesos con Docker, n8n y Modelos de IA: Generación y Evaluación Automática de Exámenes Basada en RAG y Modelos de IA para Entornos Educativos
Junio 2025

Autor: Diego Luján Raboso

Tutora:
Adriana Toni Delgado
Lenguajes y Sistemas Informáticos e Ingeniería de Software
ETSI Informáticos
Universidad Politécnica de Madrid

Resumen

Este Trabajo de Fin de Grado presenta el diseño y desarrollo de un sistema automatizado para la generación y evaluación de exámenes, utilizando tecnologías modernas como Docker, n8n, modelos de lenguaje de OpenAI y un sistema RAG (Retrieval-Augmented Generation). El objetivo principal ha sido reducir la carga de trabajo del profesorado en la elaboración y corrección de pruebas, al tiempo que se mejora la experiencia de aprendizaje del alumnado a través de una retroalimentación automatizada y personalizada.

El sistema se ha implementado en forma de un prototipo funcional que simula el entorno de una asignatura ficticia. Este enfoque ha permitido trabajar con un temario inventado y autocontenido, lo cual garantiza que cualquier contenido generado por el sistema se base exclusivamente en la información cargada mediante el sistema RAG, descartando la posibilidad de que el modelo utilice datos aprendidos previamente. Esto ha sido clave para verificar que el modelo accede correctamente al contexto proporcionado por la base vectorial, confirmando el funcionamiento efectivo del sistema RAG.

Para el desarrollo de este sistema se han integrado distintas tecnologías. Por un lado, Docker permite desplegar los componentes del sistema de forma modular y escalable, asegurando su portabilidad. n8n actúa como motor de automatización, con flujos que gestionan tareas como la segmentación del temario, la consulta al sistema RAG o la comunicación con los modelos de IA. Para la generación y evaluación de exámenes se ha utilizado el modelo GPT-4.1-mini de OpenAI, junto con el modelo text-embedding-3-small para la vectorización de los contenidos del temario. Por último, se ha desarrollado una aplicación web con Angular, que sirve de interfaz visual para que el usuario pueda interactuar con el sistema de forma intuitiva y eficaz.

El funcionamiento del sistema ha sido validado mediante diversas pruebas. En el proceso de generación de exámenes se ha comprobado que las preguntas generadas se alinean correctamente con los temas seleccionados, y que el modelo respeta las instrucciones contenidas en los “prompts”. Asimismo, en la evaluación de respuestas, el sistema ha demostrado ser capaz de generar valoraciones realistas, justificadas y coherentes, especialmente en el caso de preguntas de desarrollo, donde se proporciona además un comentario explicativo por cada calificación asignada.

Aunque se han obtenido resultados satisfactorios, también se han identificado diversas limitaciones. El sistema RAG implementado en n8n resulta eficaz para temarios sencillos, pero no está optimizado para gestionar contenidos complejos o muy extensos. Además, el modelo de lenguaje, pese a su rendimiento, sigue presentando algunas limitaciones como la sensibilidad al “prompt” o la posibilidad de generar respuestas incorrectas (“alucinaciones”). Por otro lado, la integración con plataformas educativas como Moodle todavía no ha sido posible, debido a la falta de una API accesible directamente desde n8n.

En cuanto al impacto, en el proyecto se ha reflexionado sobre el uso responsable de la inteligencia artificial en el ámbito educativo, incluyendo una evaluación de su posible huella ecológica y su alineación con los Objetivos de Desarrollo Sostenible (ODS). Se ha destacado que, aunque el sistema actual utiliza modelos ligeros y no tiene un uso intensivo, su escalado podría tener un impacto

considerable en términos de consumo energético, emisiones de carbono o generación de residuos electrónicos.

En resumen, este TFG demuestra la viabilidad de automatizar parte del proceso de evaluación educativa mediante tecnologías actuales, y presenta las bases para su evolución futura hacia sistemas más complejos.

Abstract

This Final Degree Project presents the design and development of an automated system for the generation and evaluation of exams, using modern technologies such as Docker, n8n, OpenAI language models, and a Retrieval-Augmented Generation (RAG) architecture. The main goal of the project is to reduce the workload of teaching staff in exam preparation and grading, while also improving the student experience through fast, personalized, and automated feedback.

The system has been implemented as a functional prototype built around a fictional subject. This approach has enabled the use of a custom, self-contained syllabus, ensuring that all content generated by the system is based solely on the information loaded into the RAG system. This guarantees that the language model does not rely on any previously learned knowledge, thus allowing for proper validation of the RAG-based retrieval mechanism.

Multiple technologies were combined to develop the system. Docker was used to deploy system components in a modular and scalable way. n8n served as the workflow orchestrator, managing tasks such as syllabus segmentation, vector database access, and model interaction. OpenAI's GPT-4.1-mini was used for text generation and exam evaluation, while the text-embedding-3-small model was employed for content vectorization. An Angular-based web application was developed to serve as the user interface, offering a clear and responsive way to interact with the system.

The system's performance was evaluated through several tests. In the exam generation process, it was confirmed that the questions generated always aligned with the selected topics, and that the model adhered to the instructions defined in the prompts. Regarding evaluation, the system provided realistic and coherent feedback, particularly in open-ended questions, where each score was accompanied by a justification comment.

Although the results were positive, several limitations were identified. The RAG system implemented with n8n is suitable for simple syllabi but is not optimized for complex or hierarchical content. Moreover, while the language model delivered good performance, it still exhibits typical limitations such as sensitivity to prompt design and occasional inaccuracies or hallucinations. Additionally, full integration with educational platforms such as Moodle is not yet feasible, as there is no accessible API that allows n8n to retrieve course data automatically.

Finally, the project also reflects on the responsible use of artificial intelligence in educational contexts, analyzing its environmental impact and its relevance to the Sustainable Development Goals (SDGs). Although the current system operates with lightweight models and limited usage, its future scalability may have a greater carbon footprint, energy demand, and electronic waste implications.

In conclusion, this project demonstrates the technical viability of automating parts of the educational assessment process using current AI technologies and provides a foundation for future development.

Tabla de contenidos

1 Introducción

1.1 Planteamiento del problema

¿Cuánto tiempo se invierte en la creación de exámenes que incluyan preguntas variadas y relevantes? ¿Y cuánto más en corregirlos y dar una retroalimentación útil a cada estudiante?

Hoy en día, un gran número de estas tareas siguen siendo manuales. Diseñar preguntas, corregir respuestas y, especialmente, dar un feedback personalizado, sigue siendo un proceso que consume mucho tiempo, sobre todo cuando hay numerosos alumnos. Esto representa una carga importante tanto para el profesorado como para el alumnado:

- **Para los docentes y centros educativos**, implica una gran cantidad de horas de trabajo dedicadas a preparar, corregir y devolver las pruebas.
- **Para los estudiantes**, supone esperar para conocer su nota y recibir comentarios que les ayuden a mejorar, algo fundamental para su avance académico.

Con el desarrollo de herramientas basadas en inteligencia artificial, automatizar parte de este proceso ya no es solo una idea teórica, sino una posibilidad real. Tecnologías como los modelos de lenguaje, los flujos de trabajo automáticos y las arquitecturas en contenedores permiten imaginar sistemas que generen preguntas, corrijan respuestas y ofrezcan feedback sin intervención constante del profesorado.

Aun así, esto no está exento de dificultades. La IA todavía tiene retos importantes como adaptarse bien a diferentes asignaturas, ofrecer resultados fiables o integrarse correctamente en plataformas educativas que ya existen.

Este Trabajo de Fin de Grado surge de esa problemática y busca proponer una solución que aproveche herramientas como Docker, n8n y modelos de inteligencia artificial para automatizar la generación y evaluación de exámenes. El objetivo es explorar una forma más eficiente y moderna de abordar el proceso de evaluación académica, especialmente en contextos con muchos estudiantes y recursos limitados.

1.2 Objetivos del proyecto

El objetivo principal de este Trabajo de Fin de Grado es diseñar y desarrollar un sistema automatizado que facilite tanto la creación como la corrección de exámenes, con el fin de reducir la carga de trabajo del profesorado y mejorar la experiencia educativa de los estudiantes.

Para lograrlo, el sistema combinará diferentes tecnologías:

- **Docker**: Permitirá desplegar de forma sencilla y aislada los distintos componentes del sistema;
- **n8n**: Herramienta de automatización para orquestar los flujos de trabajo;
- **Modelos de lenguaje (LLMs)**: Encargados del procesamiento y generación de preguntas, respuestas y retroalimentación;
- **Angular**: Un framework de desarrollo web que servirá como interfaz para interactuar de forma intuitiva con el sistema.

De forma más específica, en el proyecto se plantean los siguientes objetivos:

- **Automatizar la generación de preguntas** de examen y la **corrección automática de las respuestas**.
- **Implementar un sistema RAG (Retrieval-Augmented Generation)** que permita procesar el temario y mejorar las respuestas generadas por el modelo de lenguaje a partir de documentos o fuentes relevantes.
- **Ofrecer retroalimentación automática** y personalizada al alumnado, para que puedan conocer su rendimiento de forma más rápida y útil.
- **Explorar la posible integración del sistema en plataformas educativas existentes** y su capacidad de adaptarse a diferentes tipos de contenidos y asignaturas.
- **Evaluar el impacto medioambiental y ético** del uso de tecnologías de inteligencia artificial, especialmente en el contexto educativo.

Este conjunto de objetivos busca no solo demostrar la viabilidad técnica del sistema, sino también reflexionar sobre su aplicabilidad real y su potencial impacto a medio y largo plazo.

2 Marco teórico y tecnologías

2.1 Modelos de lenguaje (LLMs)

Los modelos de lenguaje de gran tamaño (LLMs, por sus siglas en inglés) son sistemas de inteligencia artificial diseñados para comprender y generar lenguaje humano. Estos modelos se entrenan con vastas cantidades de datos textuales, lo que les permite realizar tareas como traducción, resumen, generación de texto y respuesta a preguntas [1].

El entrenamiento de un LLM implica el procesamiento de enormes volúmenes de datos textuales mediante técnicas de aprendizaje profundo. Durante este proceso, el modelo aprende a predecir la siguiente palabra en una secuencia, desarrollando una comprensión estadística del lenguaje [2].

Una vez preentrenado, el modelo puede ajustarse (fine-tuning) para tareas específicas, adaptándolo a dominios particulares mediante conjuntos de datos más pequeños y especializados [3].

Los LLMs tienen una amplia gama de aplicaciones en diversos sectores:

- **Educación**: Generación de contenido educativo, asistencia en la redacción y evaluación automatizada de textos.
- **Salud**: Análisis de registros médicos, asistencia en diagnósticos y generación de informes clínicos [1].
- **Atención al cliente**: Implementación en chatbots y asistentes virtuales para proporcionar respuestas precisas y contextuales [1].
- **Desarrollo de software**: Asistencia en la generación de código y documentación técnica [2].

Algunos de los LLMs más destacados incluyen:

- **GPT-4**: Desarrollado por OpenAI, es capaz de comprender y generar texto de alta calidad en múltiples idiomas.
- **Bard**: Modelo de Google diseñado para proporcionar respuestas informativas y creativas.
- **LLaMA**: Modelo de Meta optimizado para eficiencia y rendimiento en tareas de procesamiento de lenguaje natural [2].

A pesar de sus capacidades, los LLMs presentan desafíos, como la generación de información incorrecta o sesgada, conocida como "alucinaciones", debiendo diseñar “prompts” muy específicos y sin margen de duda para el LLM. Además, su implementación plantea cuestiones éticas relacionadas con la privacidad, la seguridad y el impacto en el empleo [3].

2.2 Arquitectura RAG

La arquitectura *Retrieval-Augmented Generation* (RAG) es una técnica que mejora significativamente la precisión y actualidad de las respuestas generadas por modelos de lenguaje de gran tamaño (LLMs). En lugar de depender exclusivamente del conocimiento almacenado durante el entrenamiento del modelo, un sistema RAG incorpora mecanismos de recuperación de información externa que permiten al modelo acceder a una base de conocimiento dinámica y actualizada en tiempo real [4], [5].

Esto es especialmente útil para contextos en los que se requiere información específica o reciente, como en entornos educativos, legales o de atención al cliente. RAG optimiza la salida de un LLM al consultar una base de datos autorizada antes de generar una respuesta, ayudando así a reducir las "alucinaciones" o errores típicos de los modelos generativos [4].

Un sistema RAG opera a través de una secuencia de pasos bien definidos que combinan técnicas de búsqueda semántica con generación de lenguaje natural. A continuación, se describe el proceso:

1. **Ingestión y procesamiento de datos**: En primer lugar, el sistema ingiere documentos o materiales relevantes. Estos documentos se dividen en fragmentos coherentes conocidos como “chunks” (por ejemplo, párrafos o secciones temáticas), y se preparan para su análisis posterior [5].
2. **Generación de embeddings**: Cada fragmento de texto es transformado en un vector numérico conocido como *embedding*. Este vector representa el significado semántico del texto, lo que permite compararlo con otros fragmentos o consultas. Estos embeddings se almacenan en una base de datos vectorial, optimizada para búsquedas de similitud [6].
3. **Conversión de la consulta**: Cuando un usuario realiza una pregunta o consulta, esta también se convierte en un embedding utilizando el mismo modelo. Esto asegura que tanto las preguntas como las respuestas compartan el mismo espacio semántico [6].
4. **Recuperación de información**: Utilizando técnicas de búsqueda por similitud, el sistema compara el embedding de la consulta con los embeddings almacenados, recuperando los fragmentos más relevantes [4].
5. **Generación de la respuesta**: Finalmente, los fragmentos recuperados se combinan con la consulta original para construir un *prompt* enriquecido con un contexto actualizado, relevante y personalizado, que se envía al modelo de lenguaje generativo. Este modelo produce una respuesta informada que integra tanto su conocimiento previo como la información recuperada [5].

El enfoque RAG se ha aplicado con éxito en múltiples dominios. En educación, permite generar evaluaciones y proporcionar retroalimentación personalizada basada en los materiales de estudio. En el sector legal, puede consultar grandes volúmenes de documentación jurídica para apoyar la redacción de informes. En atención al cliente, permite respuestas precisas en tiempo real basadas en bases de datos de productos o servicios [4]–[6].

Un ejemplo representativo de la aplicación de RAG se encuentra en los chatbots implementados en sitios web comerciales, como tiendas en línea. En estos casos, el sistema RAG permite que la inteligencia artificial acceda a información específica proporcionada por la empresa (como catálogos de productos, precios actualizados o respuestas a preguntas frecuentes), lo que permite generar respuestas precisas y contextualizadas para los usuarios.

En el marco de este Trabajo de Fin de Grado, la arquitectura RAG es esencial para automatizar tanto la generación como la corrección de exámenes. El sistema desarrollado aprovecha RAG para generar preguntas y evaluar respuestas abiertas de los estudiantes basándose en el contenido específico de cada asignatura, procesando automáticamente temarios y materiales docentes en tiempo real.

Esta integración de RAG no solo mejora la eficiencia del proceso de evaluación, sino que también aporta mayor precisión, adaptabilidad y escalabilidad a sistemas educativos digitales.

2.3 Automatización de flujos

La automatización se define como la aplicación de tecnología, programas, robótica o procesos para lograr resultados con una intervención humana mínima [7]. Esta práctica busca optimizar la eficiencia operativa al reducir la dependencia de tareas manuales, permitiendo que los recursos humanos se enfoquen en actividades de mayor valor estratégico.

La automatización de procesos implica el uso de tecnologías para ejecutar tareas rutinarias de forma automática. Esto disminuye los plazos y errores, y reduce la carga de trabajo de los equipos, permitiendo que se enfoquen en otros objetivos más estratégicos para la empresa [8]. Al implementar sistemas automatizados, las organizaciones pueden mejorar la precisión y consistencia de sus operaciones, lo que se traduce en una mayor calidad en los resultados y una reducción de los costos operativos.

La implementación de soluciones de automatización ofrece múltiples beneficios, entre los que se destacan:

- **Reducción de errores**: Al minimizar la intervención humana en tareas repetitivas, se disminuye la probabilidad de errores asociados a la fatiga o distracción [8].
- **Mejora en la eficiencia**: Los procesos automatizados suelen ser más rápidos y consistentes, lo que permite una mayor productividad y una mejor utilización de los recursos disponibles [7].
- **Enfoque en tareas estratégicas**: Al liberar a los empleados de tareas rutinarias, estos pueden dedicar más tiempo a actividades que requieren análisis, creatividad y toma de decisiones, contribuyendo al crecimiento y la innovación dentro de la organización [8].

En el marco de este proyecto, la automatización se aplica para optimizar la creación y corrección de exámenes. Con ello, no solo mejora la eficiencia del proceso educativo, sino que también garantiza una mayor equidad y objetividad en la evaluación, al reducir la variabilidad que puede introducirse mediante la corrección manual.

2.4 Aplicaciones web

Una aplicación web es un tipo de software que los usuarios pueden utilizar a través de un navegador web, sin necesidad de instalar nada en sus dispositivos locales. A diferencia de las aplicaciones tradicionales de escritorio, las aplicaciones web están alojadas en servidores y se accede a ellas a través de internet. Esto las hace fácilmente accesibles desde cualquier lugar, siempre que se cuente con una conexión [9].

Estas aplicaciones suelen estar compuestas por dos partes principales: el frontend, que es la interfaz con la que interactúa el usuario, y el backend, que gestiona la lógica de negocio, la base de datos y la comunicación con otros servicios. Ambos componentes se comunican mediante el protocolo HTTP, a través de lo que se conoce como peticiones HTTP [9].

Una petición HTTP (Hypertext Transfer Protocol) es un mecanismo fundamental mediante el cual un navegador web se comunica con un servidor para solicitar recursos o enviar datos. Cada vez que un usuario interactúa con una aplicación web (por ejemplo, al hacer clic en un botón, enviar un formulario o acceder a una página), se realiza una petición HTTP que puede tener distintos métodos como:

- **GET**: Para obtener datos del servidor;
- **POST**: Para enviar datos al servidor (como al iniciar sesión);
- **PUT y DELETE**: Para actualizar o eliminar información respectivamente [10].

Cada petición HTTP incluye elementos como la URL del recurso solicitado, los encabezados (headers), que pueden contener información sobre el tipo de contenido o las credenciales de autenticación, y el cuerpo (body), en el caso de métodos como POST o PUT [10].

El uso de aplicaciones web ofrece múltiples ventajas en términos de accesibilidad, mantenimiento y escalabilidad. Al centralizar el software en un servidor, se simplifica su actualización y gestión, permitiendo que múltiples usuarios puedan acceder simultáneamente desde cualquier dispositivo.

En el contexto de este proyecto, la aplicación web desarrollada con Angular actúa como interfaz para que los profesores puedan generar exámenes, revisar correcciones automatizadas o analizar resultados. Mediante peticiones HTTP, esta aplicación se comunica con los distintos servicios del sistema (como el motor de LLMs o los flujos en n8n), permitiendo una experiencia fluida y dinámica en tiempo real.

2.5 Herramientas utilizadas

Para llevar a cabo el desarrollo del sistema propuesto, se han utilizado distintas herramientas tecnológicas que permiten automatizar procesos, generar contenido con inteligencia artificial y ofrecer una interfaz web moderna e intuitiva. A continuación, se describen en detalle estas herramientas y su papel dentro del proyecto.

2.5.1 Docker + n8n

La combinación de Docker y n8n ha resultado clave para lograr una arquitectura modular, escalable y fácil de mantener. Ambas tecnologías permiten automatizar procesos de forma eficiente, lo que se alinea directamente con los objetivos del trabajo.

Docker es una plataforma que facilita la creación, el despliegue y la ejecución de aplicaciones a través de contenedores. Estos contenedores agrupan todo lo necesario para que una aplicación funcione correctamente, incluyendo el código, las librerías y las dependencias, lo que garantiza que se comporten igual en cualquier entorno, ya sea local o en la nube [11]. Esto resulta especialmente útil en contextos educativos o de investigación, donde las condiciones técnicas pueden variar entre instituciones o servidores. Además, su carácter ligero en comparación a otras soluciones como las máquinas virtuales y su capacidad de aislamiento facilitan el despliegue de varios servicios sin conflictos entre ellos [12].

Sobre esta base se integra n8n, una herramienta de automatización de código abierto que permite diseñar flujos de trabajo conectando diferentes servicios y

aplicaciones de forma visual, es decir, sin necesidad de escribir código. n8n destaca por su flexibilidad: permite construir automatizaciones complejas mediante una interfaz gráfica, pero también ofrece la posibilidad de incluir código personalizado si se requiere un control más detallado [13]. Esta herramienta se convierte así en el "motor" de automatización del sistema, gestionando tareas como la generación de preguntas, el envío de respuestas a modelos de lenguaje o el almacenamiento y presentación de resultados.

Al ejecutarse dentro de un contenedor Docker, n8n puede instalarse y ejecutarse fácilmente en cualquier máquina compatible, lo que simplifica tanto la puesta en marcha como la distribución del sistema. Esta integración también aporta ventajas en términos de mantenimiento y escalabilidad, ya que permite modificar o ampliar partes del sistema sin afectar al resto.

En conjunto, Docker y n8n ofrecen una solución sólida y versátil para automatizar procesos, reduciendo tiempos de desarrollo y facilitando la adaptación del sistema a diferentes contextos. En este proyecto, su uso ha permitido centrarse en la lógica educativa y en la calidad de las evaluaciones, dejando en segundo plano los problemas técnicos de despliegue e integración.

2.5.2 OpenAI LLM

En este proyecto, se ha optado por utilizar modelos de lenguaje de OpenAI debido a su capacidad para generar texto coherente y contextualizado, así como para realizar tareas complejas de procesamiento del lenguaje natural. En particular, se ha empleado el modelo GPT-4.1-mini y el modelo de embeddings text-embedding-3-small, ambos accesibles a través de la API de OpenAI [14], [15]. A pesar de ser modelos de pago, se trata de un precio bastante reducido, y para realizar un sistema aislado de prueba que no requiere una gran carga de peticiones simultaneas resultan ideales.

Los modelos de lenguaje de OpenAI, como GPT-4.1-mini, son redes neuronales entrenadas para predecir la siguiente palabra en una secuencia de texto, lo que les permite generar respuestas coherentes y contextualmente relevantes. GPT-4.1-mini es una versión optimizada del modelo GPT-4.1, diseñada para ofrecer un equilibrio entre rendimiento y eficiencia. Este modelo destaca por su capacidad para seguir instrucciones, generar código y comprender contextos extensos, con una ventana de contexto de hasta 1 millón de tokens [14].

La elección de GPT-4.1-mini en este proyecto se debe a su rendimiento mejorado en tareas de generación de texto y su capacidad para manejar contextos largos, lo que es esencial para procesar documentos educativos y generar retroalimentación detallada. Además, su eficiencia en términos de costo y latencia lo convierte en una opción adecuada para aplicaciones que requieren respuestas rápidas y precisas [14], [16].

Los modelos de embeddings de OpenAI transforman texto en vectores numéricos que capturan el significado semántico del contenido, facilitando tareas como la búsqueda semántica y la recuperación de información. El modelo text-embedding-3-small es una versión eficiente y de alto rendimiento que mejora significativamente a su predecesor, text-embedding-ada-002. Ofrece una mayor precisión en tareas de recuperación multilingüe y en inglés, con una reducción de costos considerable [15].

En el contexto de este proyecto, text-embedding-3-small se utiliza en la implementación del sistema de generación aumentada por recuperación (RAG),

donde las respuestas generadas por el modelo de lenguaje se enriquecen con información relevante extraída de los materiales de estudio [15], [17].

2.5.3 Angular

Angular es un framework para aplicaciones web de una sola página (SPA) desarrollado por Google. Está basado en TypeScript y ofrece una arquitectura orientada a componentes, lo que permite estructurar las aplicaciones de manera modular y reutilizable. Angular incorpora funcionalidades como la inyección de dependencias, el enrutamiento, la vinculación bidireccional de datos y un sistema de plantillas declarativas que mejora la productividad durante el desarrollo [18].

Una de las principales ventajas de Angular es su escalabilidad, ya que permite crear desde aplicaciones pequeñas hasta soluciones empresariales complejas, manteniendo una arquitectura clara y mantenible. Además, cuenta con una comunidad activa y una rica colección de herramientas que facilitan la integración y el despliegue de funcionalidades avanzadas [18].

Para complementar Angular, se ha utilizado PrimeNG, una biblioteca de componentes de interfaz de usuario diseñada específicamente para este framework. PrimeNG proporciona una colección extensa de elementos listos para usar, como botones, formularios, menús y tablas. Estos componentes están optimizados para integrarse con Angular y son altamente personalizables, lo que permite construir interfaces ricas de forma rápida [19].

Además, PrimeNG ofrece temas visuales y plantillas que facilitan el diseño coherente de las aplicaciones, adaptándose a diferentes estilos visuales sin necesidad de desarrollar cada componente desde cero. Su naturaleza modular permite incorporar únicamente los elementos necesarios, lo que mejora el rendimiento y facilita el mantenimiento [19].

Por otro lado, se ha incorporado Tailwind CSS como solución para el diseño visual y la personalización de estilos. Tailwind es un framework de CSS basado en utilidades, lo que significa que se construyen las interfaces directamente en el HTML mediante clases predefinidas. Esto permite un control más preciso y una mayor velocidad a la hora de aplicar estilos, sin depender de hojas de estilo personalizadas extensas [20].

Tailwind se integra en Angular de manera completamente compatible y favorece un desarrollo ágil, manteniendo la consistencia visual en toda la aplicación. Gracias a su enfoque de bajo nivel, es posible diseñar interfaces altamente personalizadas sin sacrificar productividad [20].

3 Desarrollo

3.1 Diseño general del sistema

Con el objetivo de mostrar de forma práctica cómo se pueden aplicar los flujos automatizados y la inteligencia artificial en el ámbito educativo, este Trabajo de Fin de Grado presenta un prototipo funcional de un sistema automatizado para la generación de preguntas de examen y la evaluación de las respuestas proporcionadas por el alumnado. Este sistema busca ofrecer una primera aproximación a una solución concreta para los problemas planteados en los apartados anteriores.

El prototipo se basa en una asignatura ficticia sobre los planetas de un sistema solar inventado. Para esta asignatura se ha generado un temario compuesto de 9 temas autocontenidos en los cuales se explican las características de cada uno de los 9 planetas que conforman el sistema solar. Con este temario tan único se ha diseñado un sistema RAG que proporciona contexto al modelo de lenguaje encargado de generar preguntas y evaluar las respuestas.

La razón de usar una asignatura con un temario totalmente ficticio es demostrar el correcto uso del sistema RAG por parte del LLM. Dado que el contenido no existe previamente en el conocimiento del modelo, cualquier pregunta o evaluación relevante sólo puede generarse a partir del contexto proporcionado dinámicamente a través de RAG.

En vistas generales, este sistema consiste principalmente en 3 partes diferenciadas:

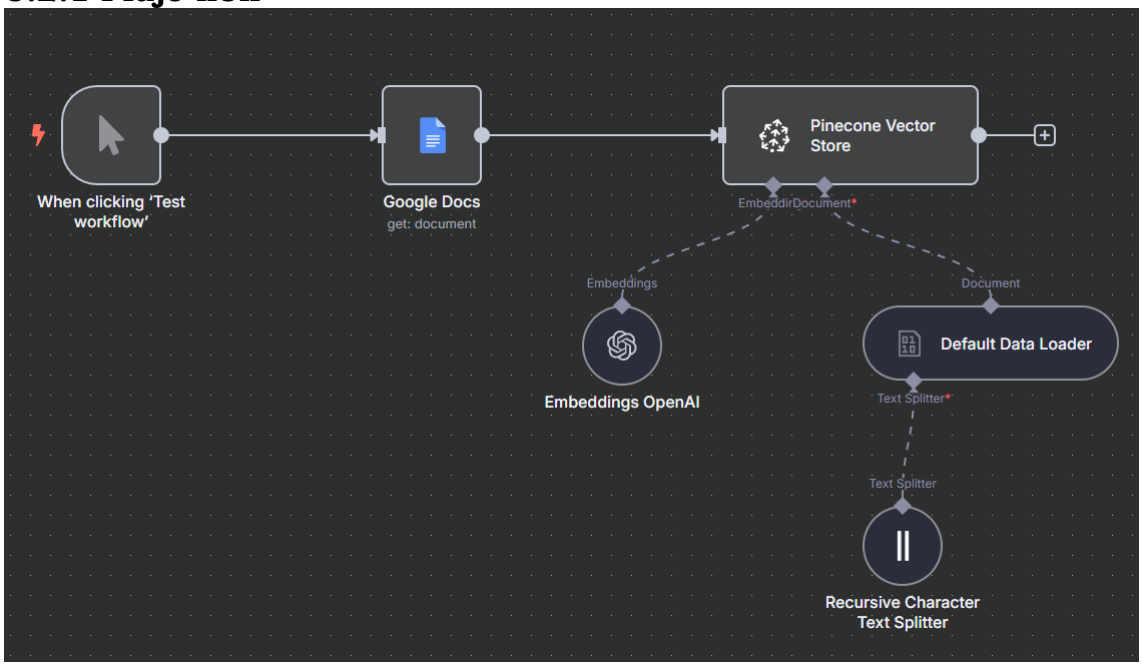
Por un lado, tenemos la aplicación web, desarrollada en el framework de Angular, la cual presenta de una forma intuitiva y atractiva una interfaz pensada para que el usuario interaccione con el sistema. Esta interacción se realiza mediante el uso de formularios reactivos y peticiones HTTP dirigidas a los endpoints de los flujos n8n.

Por otro lado, para que el sistema RAG funcione correctamente, necesita tener acceso a una base de datos vectorial con disponibilidad constante. Por ello, se ha optado por implementarla con Pinecone, una aplicación que dispone de una versión gratuita en la que puedes disponer de una sencilla (pero suficiente) base de datos vectorial diseñada para aplicaciones con IA [21]. Además, Pinecone tiene una integración directa en n8n ya que dispone de nodos concretos para la interacción con bases de datos de este proveedor.

Por último, n8n actúa como un nexo entre el resto de los componentes del sistema a través de flujos. Estos flujos recuperan la información pertinente, la modifica y/o la usa para generar nueva información. De esta manera se puede realizar tareas de manera automatizada como por ejemplo introducir el temario de la asignatura en la base de datos vectorial o interactuar con un LLM.

3.2 Implementación del sistema RAG

3.2.1 Flujo n8n



3.2.1.1 Flujo n8n para establecer un RAG

El sistema RAG constituye el núcleo funcional de este proyecto, ya que es el encargado de proporcionar al modelo de lenguaje el contexto necesario para generar preguntas y evaluar respuestas de forma coherente. La implementación del sistema comienza con la preparación e indexación del temario ficticio, y se realiza a través de un flujo automatizado en n8n.

El objetivo de este flujo es recuperar el contenido del temario de la asignatura, segmentarlo en fragmentos relevantes (chunks), convertir dichos fragmentos en vectores semánticos mediante el modelo de embedding de OpenAI, y almacenarlos en una base de datos vectorial gestionada con Pinecone. De este modo se construye la base de conocimientos del sistema RAG.

Al ser un flujo que se va a ejecutar una única vez (o cuando se quiera hacer algún cambio en el contenido del temario), se ha elegido un "trigger" manual.

3.2.2 Puntos clave

- **Recuperación del temario**: En primer lugar, obtenemos el contenido del temario ficticio el cual tenemos almacenado en un documento de Google Docs. Este temario, se ha diseñado específicamente para el proyecto y será el que se use para todas las pruebas en el sistema.
- **Segmentación del texto**: Una vez recuperado el temario, el siguiente paso es dividirlo en fragmentos (chunks) adecuados para su posterior vectorización. Para ello, se utiliza un nodo de Pinecone con soporte para segmentación personalizada. n8n ofrece 3 "text splitters" como opciones para realizar esta división:
    - "Character Text Splitter": Divide el texto en fragmentos basados en un número fijo de caracteres [22].
    - "Recursive Character Splitter": Realiza una división recursiva del texto, intentando conservar la coherencia semántica al mantener juntos párrafos, oraciones o palabras siempre que sea posible [23].

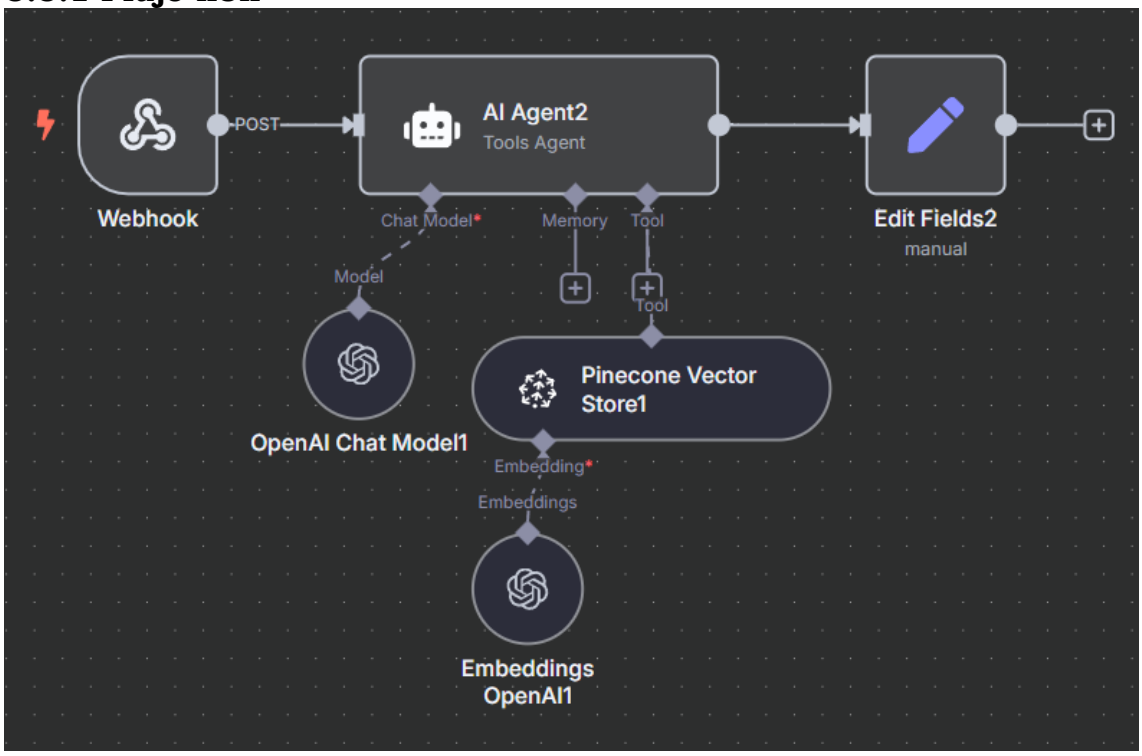
- “Token Splitter”: Divide el texto basándose en el número de tokens (unidades básicas de significado utilizadas por los modelos de lenguaje) especificado [24].

En este caso, se ha optado por el “Recursive Character Splitter”, que permite un mayor control sobre los puntos de corte del contenido. Gracias a que el temario está estructurado por temas autocontenidos y que se ha configurado el nodo de manera que el tamaño de un chunk sea suficiente para contener un tema, cada chunk corresponde directamente a un tema. Esta decisión garantiza una segmentación semántica eficaz, conservando la coherencia del contenido y facilitando su recuperación posterior sin pérdida de información.

- **Vectorización**: Una vez segmentado el temario, cada chunk es transformado en un vector numérico, que representa su significado semántico, utilizando el modelo “text-embedding-3-small” de OpenAI.

3.3 Generación de preguntas de examen

3.3.1 Flujo n8n



3.3.1.1 Flujo n8n para generar preguntas de examen

Para automatizar la generación de preguntas de examen de forma eficiente, se ha desarrollado un flujo en n8n. Este flujo permanece a la espera de peticiones HTTP entrantes desde la aplicación web, gestionadas mediante un nodo de tipo “webhook”. Una vez recibida la solicitud (de tipo POST), el sistema utiliza los datos proporcionados por el usuario para construir dinámicamente un “prompt”. Dicho “prompt” se encuentra parcialmente definido en un nodo de tipo “AI Agent” (nodo principal del flujo), encargado de comunicarse con el modelo de lenguaje. Por último, se adapta la respuesta del modelo y se envía de nuevo a la aplicación web para su visualización.

El objetivo de este flujo es generar preguntas personalizadas, basadas exclusivamente en el temario ficticio que se encuentra almacenado en la base

de datos vectorial del sistema RAG. Para ello, el flujo accede a dicha información a través de una herramienta ("tool") asociada al nodo "AI Agent".

3.3.2 Puntos clave

- **Recepción de datos mediante "webhook"**: El flujo se activa cuando se recibe una petición HTTP de tipo POST desde la aplicación web. En esta solicitud, el usuario especifica parámetros como el número de preguntas de desarrollo y tipo test, el número de opciones para las preguntas tipo test, y los temas concretos sobre los que deben tratar las preguntas.
- **Construcción del "prompt"**: Se genera un "prompt" dinámico en el que se detallan las características específicas del examen que el usuario desea generar. También se le recuerda al modelo de lenguaje que debe extraer la información necesaria únicamente del temario recuperado desde el sistema RAG.
- **"System message"**: Además del "prompt", se incluye un "system message" que define el comportamiento esperado del modelo. En este mensaje se establece que actúe como un asistente experto en la creación de exámenes, se le recuerda reiteradamente la necesidad de consultar el temario antes de generar las preguntas, y se le proporciona un esquema detallado del formato JSON que debe seguir en su respuesta.
- **Configuración del modelo GPT-4.1-mini**: El modelo utilizado para esta tarea es el GPT-4.1-mini de OpenAI, cuya configuración en el nodo incluye una temperatura alta (cercana a 1) para favorecer la generación de preguntas originales y más variadas. Asimismo, se define que el formato de respuesta debe ser un JSON, lo que facilita su posterior tratamiento por parte del sistema.
- **Formato de salida en JSON**: La respuesta generada por el modelo se devuelve en un formato JSON estructurado, con campos previamente definidos en el "system message". Esto permite que la aplicación web trabaje con los datos fácilmente.

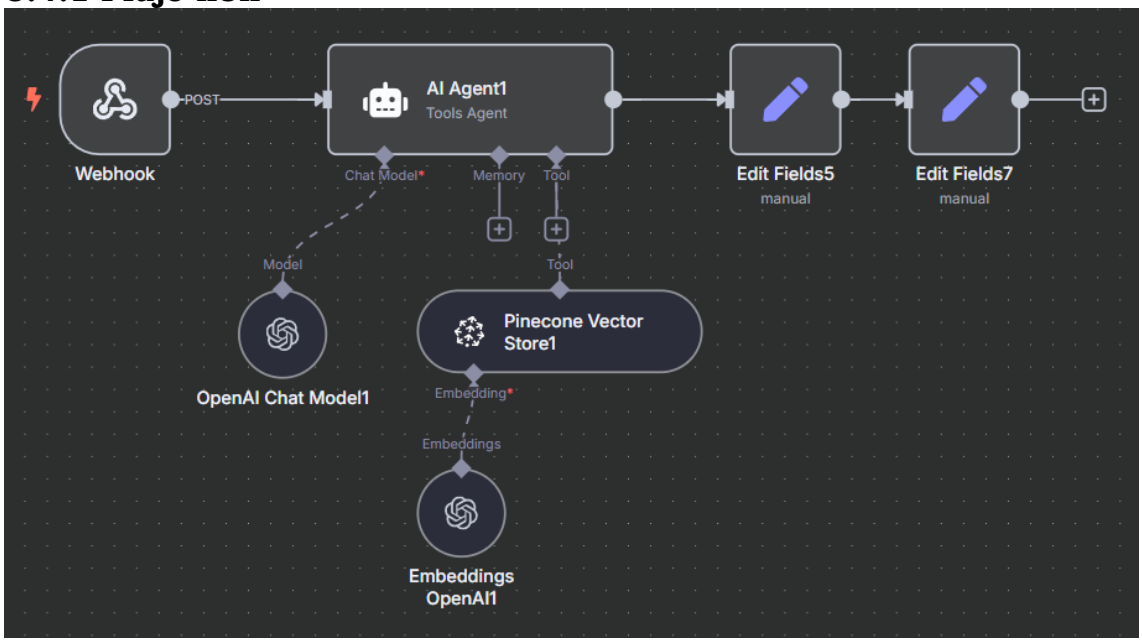
3.3.3 Regeneración

Para ofrecer una experiencia de usuario más dinámica, se da la opción de regenerar las preguntas individualmente tras generar una primera versión del examen. Para ello, al ser un caso muy parecido al general, se ha optado por reutilizar este flujo, ajustando el número de preguntas de desarrollo a 1 y de tipo test a 0 o viceversa.

Con el objetivo de ofrecer una experiencia de usuario más dinámica, el sistema permite regenerar de forma individual cualquiera de las preguntas del examen tras generar una primera versión. Dado que este caso es similar al de la generación completa de un examen, se ha optado por reutilizar el mismo flujo de n8n.

3.4 Evaluación de respuestas de exámenes

3.4.1 Flujo n8n



3.4.1.1 Flujo n8n para evaluar preguntas de examen

Para la evaluación de preguntas de examen y generación la retroalimentación correspondiente, se ha desarrollado un flujo en n8n. Este flujo permanece a la espera de peticiones HTTP entrantes desde la aplicación web, gestionadas mediante un nodo de tipo "webhook". Una vez recibida la solicitud (de tipo POST), el sistema utiliza los datos proporcionados por el usuario para construir dinámicamente un "prompt". Dicho "prompt" se encuentra parcialmente definido en un nodo de tipo "AI Agent" (nodo principal del flujo), encargado de comunicarse con el modelo de lenguaje. Por último, se adapta la respuesta del modelo y se envía de nuevo a la aplicación web para su visualización.

El objetivo de este flujo es generar una evaluación personalizada de las respuestas a las preguntas de exámenes de los estudiantes. Para ello, el flujo debe acceder al temario de la asignatura, el cual es accesible desde una herramienta ("tool") asociada al nodo "AI Agent". En dicha herramienta se interactúa con el sistema RAG.

3.4.2 Puntos clave

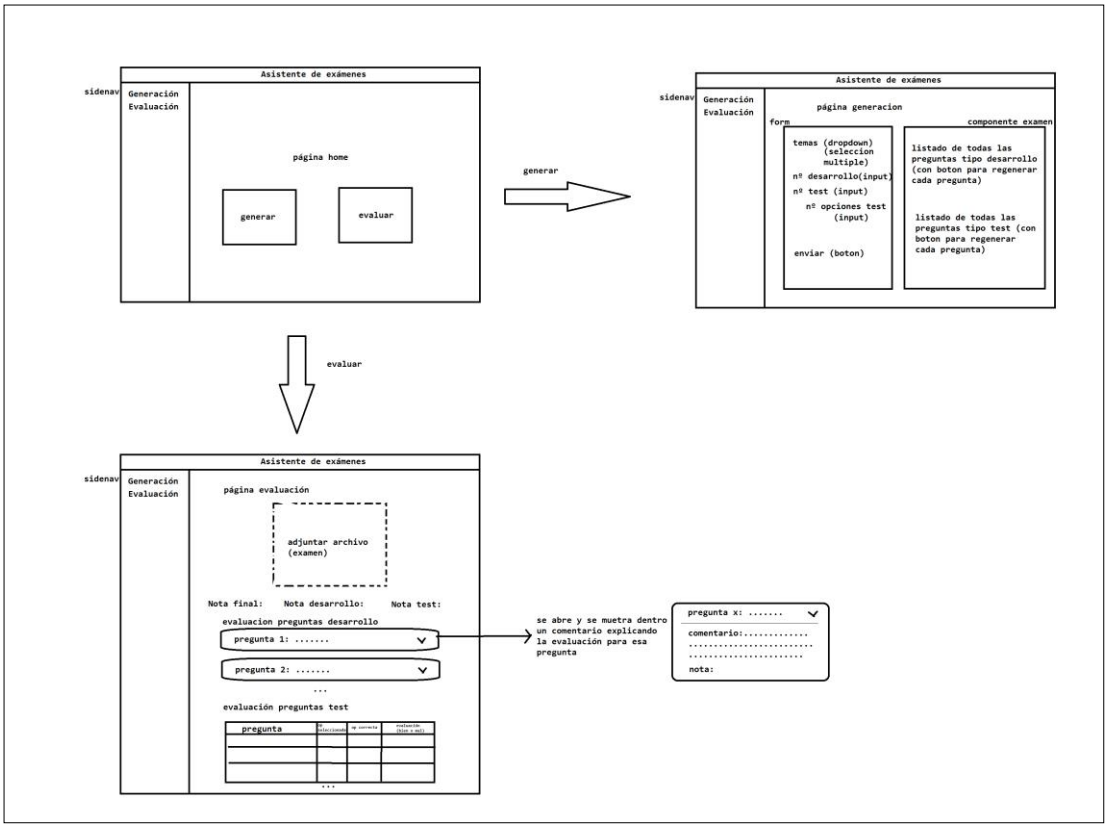
- **Recepción de datos mediante "webhook"**: El flujo se activa cuando se recibe una petición HTTP de tipo POST desde la aplicación web. En esta solicitud, el usuario adjunta el examen con las respuestas y especifica parámetros como el peso en la nota de cada parte (desarrollo y test) en formato de porcentaje.
- **Construcción del "prompt"**: Se genera un "prompt" en el que se detallan los criterios de evaluación que debe seguir el modelo, adaptados dinámicamente según las especificaciones del usuario, junto con el examen que debe ser evaluado y comentado. También se le recuerda al modelo de lenguaje que debe extraer la información necesaria únicamente del temario recuperado desde el sistema RAG.

- **“System message”**: Además del “prompt”, se incluye un “system message” que define el comportamiento esperado del modelo. En este mensaje se establece que actúe como un asistente experto en la correción de exámenes, se le recuerda reiteradamente la necesidad de consultar el temario antes de evaluar cada una de las preguntas del examen, y se le proporciona un esquema detallado del formato JSON que debe seguir en su respuesta.
- **Configuración del modelo GPT-4.1-mini**: El modelo utilizado para esta tarea es el GPT-4.1-mini de OpenAI, cuya configuración en el nodo incluye una temperatura baja (cercana a 0) para favorecer la generación de preguntas originales y más variadas. Asimismo, se define que el formato de respuesta debe ser un JSON, lo que facilita su posterior tratamiento por parte del sistema.
- **Formato de salida en JSON**: La respuesta generada por el modelo se devuelve en un formato JSON estructurado, con campos previamente definidos en el “system message”. Esto permite que la aplicación web trabaje con los datos fácilmente.

3.5 Aplicación web

3.5.1 Estructura general

Para la creación de la interfaz de usuario del sistema se ha desarrollado una aplicación web utilizando el framework Angular, que facilita la construcción de aplicaciones de tipo SPA (Single Page Application). Al tratarse de un sistema con una arquitectura sencilla y centrado en la interacción con flujos definidos en n8n, se ha optado por una implementación puramente frontend, sin backend propio.



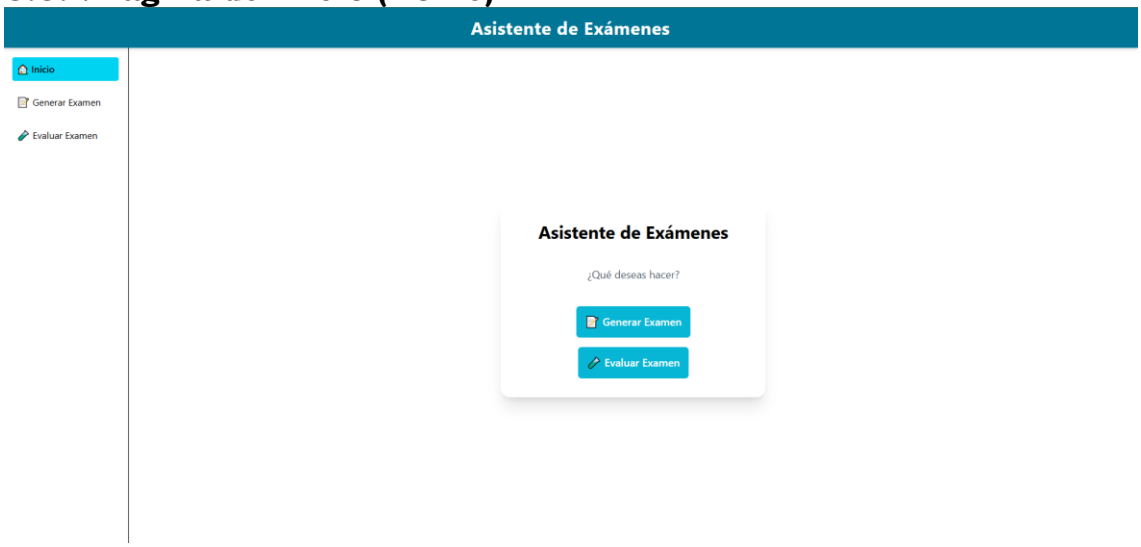
3.5.1.1 Esquema general inicial de la aplicación

La aplicación está diseñada con una estructura visual coherente y reutilizable en todas sus páginas debido al uso de componentes. Está compuesta por una barra lateral de navegación (sidenav) y un encabezado (header) que se mantienen constantes a lo largo de toda la aplicación, mientras que el contenido central se gestiona mediante el componente router-outlet de Angular, que permite cargar dinámicamente las distintas vistas según la ruta en la que nos encontremos.

El estilo visual mantiene una estética homogénea y accesible gracias al uso combinado de PrimeNG, una biblioteca de componentes personalizables, y Tailwind CSS, que permite aplicar estilos de manera rápida, flexible y coherente a través de clases CSS predefinidas.

La aplicación cuenta con tres páginas principales, cada una con funcionalidades específicas relacionadas con los objetivos del sistema: generación y evaluación automática de exámenes.

3.5.2 Página de inicio (Home)



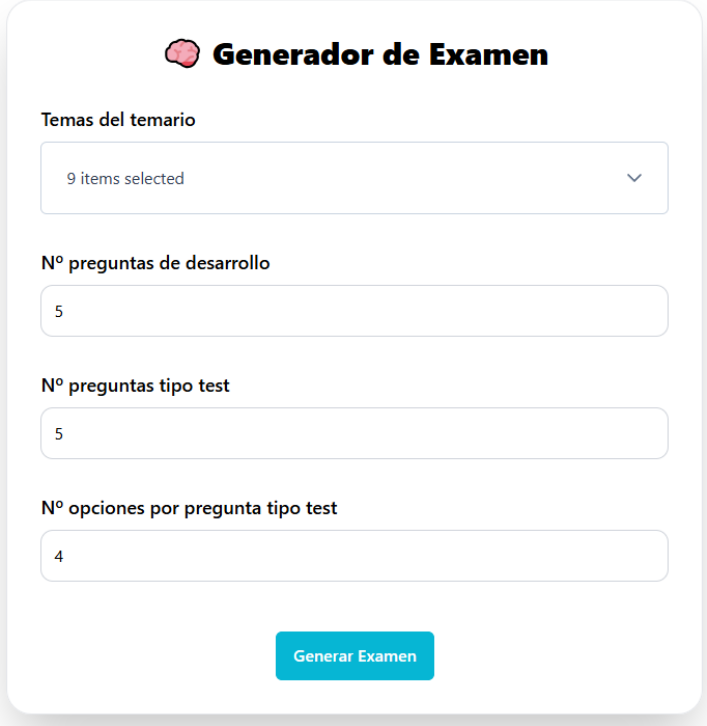
3.5.2.1 Página home de la aplicación web

Se trata de una pantalla sencilla que actúa como punto de entrada a la aplicación. Ofrece una breve presentación del sistema y permite al usuario navegar fácilmente hacia las dos funcionalidades principales: generar examen y evaluar examen.

3.5.3 Página de generación de examen

Esta vista permite al usuario configurar los parámetros del examen que desea generar. Está compuesta por un formulario reactivo con los siguientes campos:

- **Temas del temario** (selector múltiple).
- **Número de preguntas de desarrollo** (campo numérico).
- **Número de preguntas tipo test** (campo numérico).
- **Número de opciones por pregunta tipo test** (campo numérico, con validación mínima de 2).

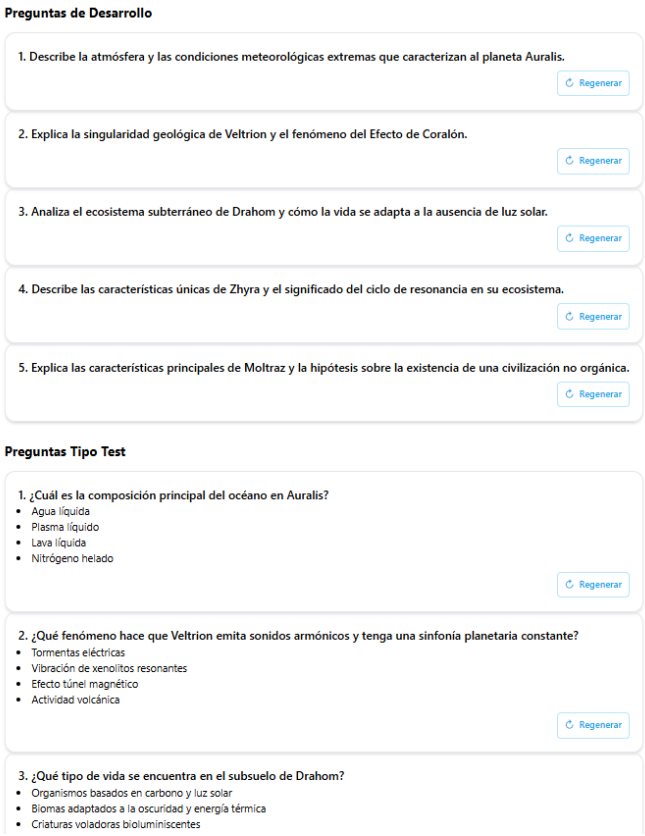


3.5.3.1 Componente: formulario para generar un examen

Todos los campos cuentan con validaciones de tipo required y condiciones mínimas para asegurar la coherencia de los datos introducidos.

Una vez completado el formulario, se realiza una petición HTTP POST al flujo correspondiente de n8n, el cual genera las preguntas solicitadas consultando el temario previamente indexado en la base de datos vectorial. La respuesta de este flujo es un objeto JSON que contiene las preguntas generadas, tanto de desarrollo como de tipo test.

Cuando la respuesta es recibida correctamente, el formulario desaparece y se carga un nuevo componente de visualización del examen, donde se listan las preguntas generadas. Cada pregunta cuenta con la opción de ser regenerada de forma individual, reutilizando el mismo flujo de n8n, pero ajustando los parámetros para solicitar solo una nueva pregunta del tipo correspondiente (desarrollo o test), conservando los temas seleccionados inicialmente.

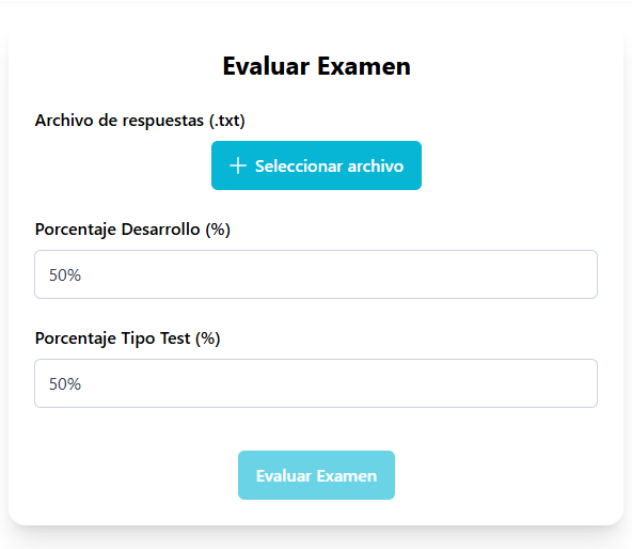


3.5.3.2 Componente: visualizador de las preguntas generadas

3.5.4 Página de evaluación de examen

En esta vista se permite al usuario subir un archivo .txt que contenga las respuestas del alumno al examen generado. Además, se proporciona un formulario con los siguientes campos:

- **Archivo del examen respondido** (input de tipo archivo).
- **Porcentaje de peso para preguntas de desarrollo** (campo numérico).
- **Porcentaje de peso para preguntas tipo test** (campo numérico).

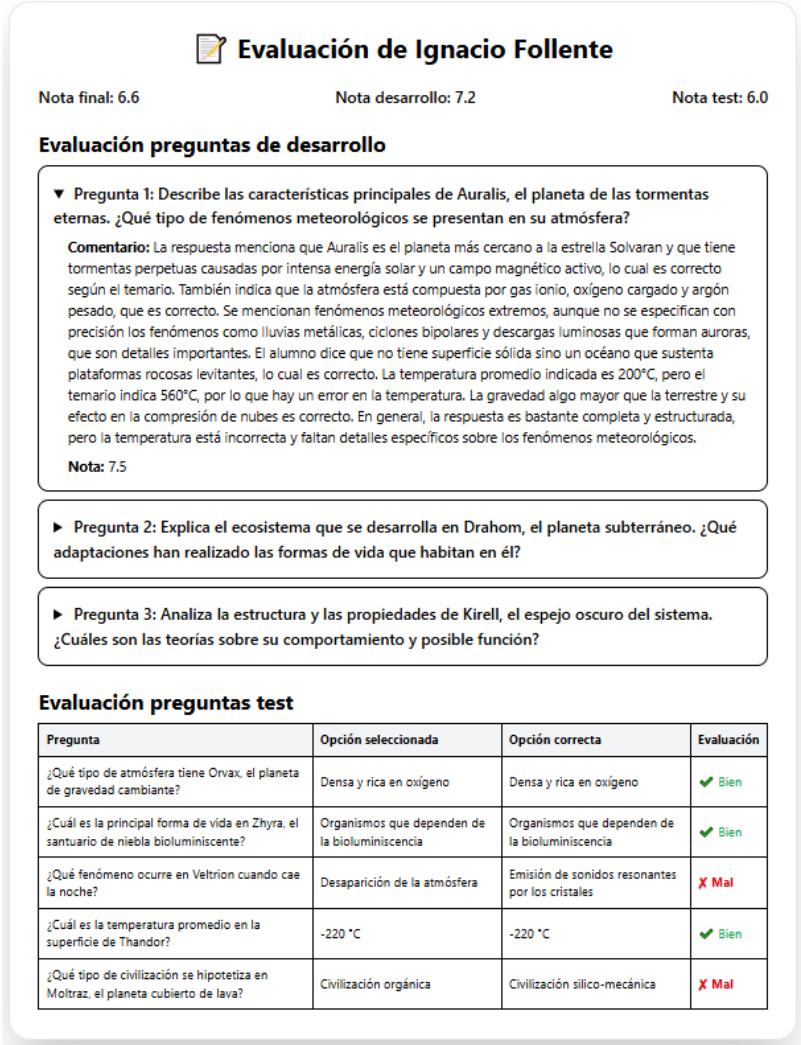


3.5.4.1 Componente: formulario para evaluar un examen

El formulario valida la presencia del archivo y que la suma de los porcentajes sea coherente. Al enviarse, se realiza una petición HTTP POST al flujo de evaluación implementado en n8n, que procesa el contenido del archivo, compara las respuestas con las preguntas originales y genera una evaluación detallada.

La respuesta de este flujo se visualiza en un componente específico que muestra:

- **Nota final** del examen.
- **Notas parciales** para preguntas de desarrollo y tipo test.
- **Evaluación detallada de las preguntas de desarrollo**, incluyendo nota individual y un comentario explicativo por cada una.
- **Tabla de corrección para las preguntas tipo test**, mostrando respuestas correctas, seleccionadas y puntuación obtenida.



Evaluación de Ignacio Follente

Nota final: 6.6 | Nota desarrollo: 7.2 | Nota test: 6.0

Evaluación preguntas de desarrollo

▼ **Pregunta 1: Describe las características principales de Auralis, el planeta de las tormentas eternas. ¿Qué tipo de fenómenos meteorológicos se presentan en su atmósfera?**

Comentario: La respuesta menciona que Auralis es el planeta más cercano a la estrella Solvaran y que tiene tormentas perpetuas causadas por intensa energía solar y un campo magnético activo, lo cual es correcto según el temario. También indica que la atmósfera está compuesta por gas ionio, oxígeno cargado y argón pesado, que es correcto. Se mencionan fenómenos meteorológicos extremos, aunque no se especifican con precisión los fenómenos como lluvias metálicas, ciclones bipolares y descargas luminosas que forman auroras, que son detalles importantes. El alumno dice que no tiene superficie sólida sino un océano que sustenta plataformas rocosas levitantes, lo cual es correcto. La temperatura promedio indicada es 200°C, pero el temario indica 560°C, por lo que hay un error en la temperatura. La gravedad algo mayor que la terrestre y su efecto en la compresión de nubes es correcto. En general, la respuesta es bastante completa y estructurada, pero la temperatura está incorrecta y faltan detalles específicos sobre los fenómenos meteorológicos.

Nota: 7.5

▶ **Pregunta 2: Explica el ecosistema que se desarrolla en Drahom, el planeta subterráneo. ¿Qué adaptaciones han realizado las formas de vida que habitan en él?**

▶ **Pregunta 3: Analiza la estructura y las propiedades de Kirell, el espejo oscuro del sistema. ¿Cuáles son las teorías sobre su comportamiento y posible función?**

Evaluación preguntas test

Pregunta	Opción seleccionada	Opción correcta	Evaluación
¿Qué tipo de atmósfera tiene Orvax, el planeta de gravedad cambiante?	Densa y rica en oxígeno	Densa y rica en oxígeno	✔ Bien
¿Cuál es la principal forma de vida en Zhyra, el santuario de niebla bioluminiscente?	Organismos que dependen de la bioluminiscencia	Organismos que dependen de la bioluminiscencia	✔ Bien
¿Qué fenómeno ocurre en Veltrion cuando cae la noche?	Desaparición de la atmósfera	Emisión de sonidos resonantes por los cristales	✗ Mal
¿Cuál es la temperatura promedio en la superficie de Thandor?	-220 °C	-220 °C	✔ Bien
¿Qué tipo de civilización se hipotetiza en Moltraz, el planeta cubierto de lava?	Civilización orgánica	Civilización silico-mecánica	✗ Mal

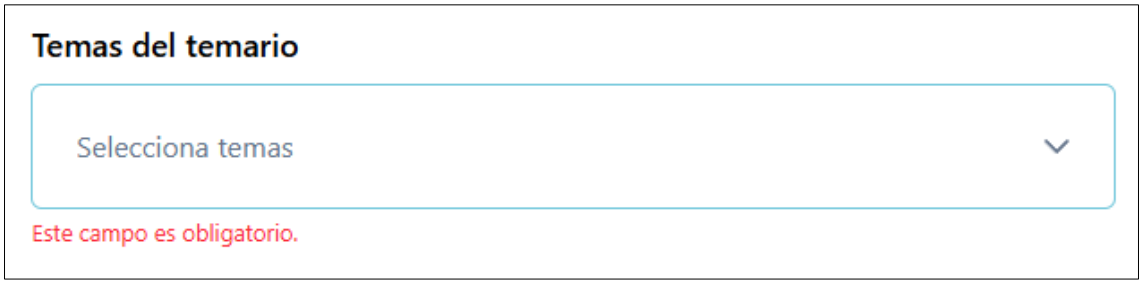
3.5.4.2 Componente: visualizador de la evaluación

3.6 Pruebas del sistema

3.6.1 Validaciones en formularios

Para garantizar una experiencia de usuario coherente y evitar errores en la interacción con la aplicación web, se han implementado distintas validaciones en los formularios. Estas validaciones impiden que el usuario envíe formularios incompletos o incorrectos, ya que el botón de envío permanece deshabilitado hasta que todos los campos requeridos cumplen las condiciones establecidas.

Una de las validaciones más básicas es la de tipo “required”, que obliga al usuario a completar ciertos campos antes de poder continuar. En caso de dejar alguno de estos campos vacío, se muestra un mensaje de advertencia personalizado que informa de manera clara sobre la necesidad de introducir datos válidos.



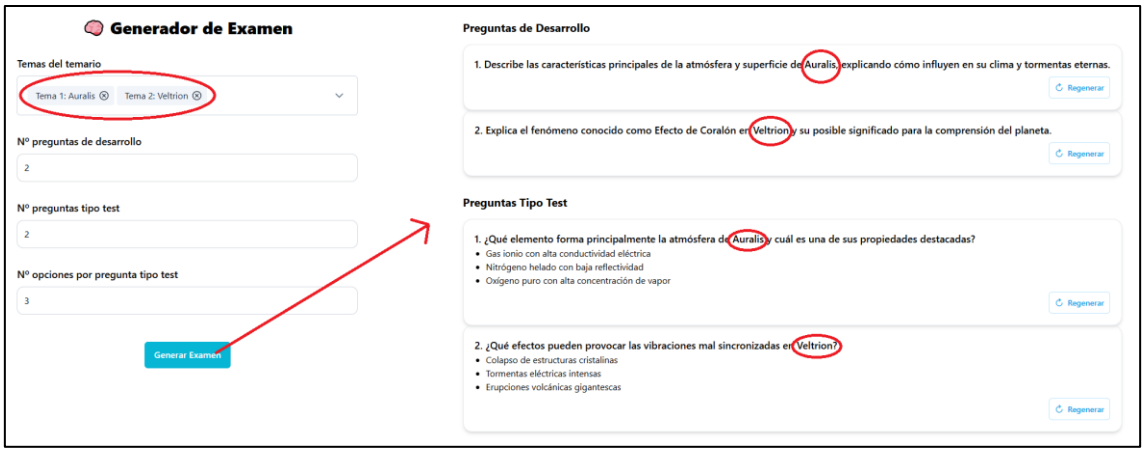
3.6.1.1 Ejemplo de validador "required"

Además, para los campos numéricos se han aplicado validaciones adicionales con el validador min, que establece un valor mínimo permitido. Por ejemplo, en el caso del número de opciones en una pregunta tipo test, el mínimo debe ser 2 para que el formato tenga sentido. Si el usuario introduce un número inferior al permitido, la aplicación muestra un mensaje explicativo, informando del error de manera intuitiva.

3.6.1.2 Ejemplos de validadores "min" en 0 y 2 respectivamente

3.6.2 Respuestas alineadas con el temario

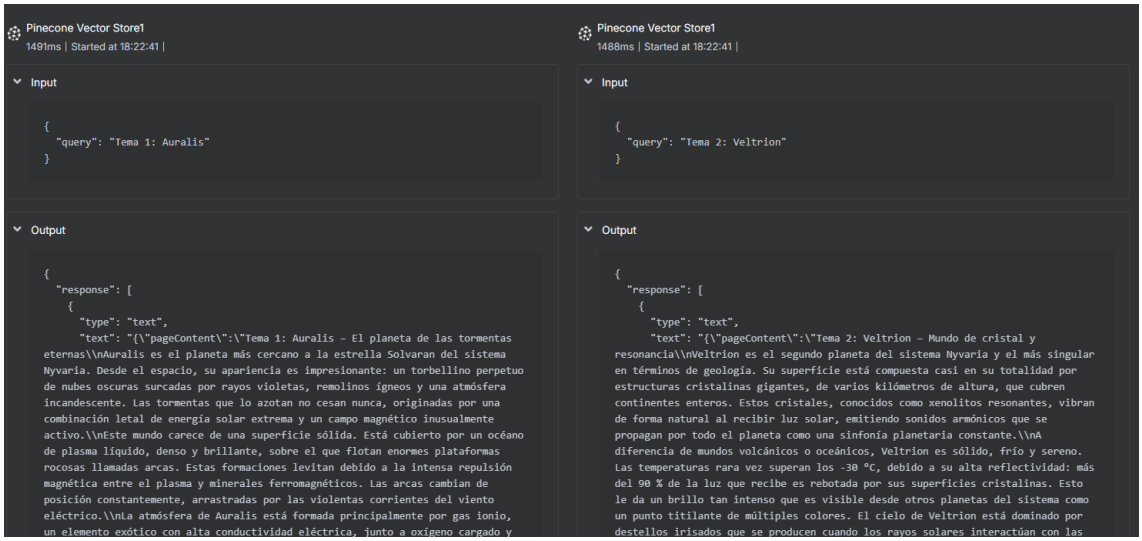
Durante el proceso de generación de exámenes, el usuario puede seleccionar los temas concretos del temario sobre los que desea que traten las preguntas. A lo largo de las pruebas realizadas, se ha comprobado que el sistema respeta con precisión esta selección: las preguntas generadas están siempre relacionadas con los temas indicados.



3.6.2.1 Ejemplo de examen generado con 2 temas

En la ilustración correspondiente se muestra un ejemplo donde el usuario ha seleccionado únicamente los temas Auralis y Veltrion. Como puede observarse, todas las preguntas generadas guardan relación directa con esos contenidos, lo que demuestra que el modelo de lenguaje ha consultado correctamente el sistema RAG y ha generado las preguntas en función del contexto proporcionado.

Adicionalmente, se ha verificado el funcionamiento interno del sistema observando los logs de ejecución en n8n. En ellos puede verse cómo, por cada tema seleccionado, se realiza una consulta individual a la base de datos vectorial de Pinecone. Esta recuperación selectiva garantiza que únicamente se utilice información relevante y específica, reforzando la alineación entre temario y preguntas generadas.



3.6.2.2 Ejemplo de recuperación de temas para generar preguntas de examen

3.6.3 Evaluaciones con criterio

En el proceso de evaluación automática de las respuestas tipo test, el sistema no se limita a asignar una nota, sino que también genera un comentario explicativo para cada pregunta. Este comentario justifica la calificación obtenida indicando qué elementos se han tenido en cuenta en la corrección.

Gracias a este enfoque, el usuario no solo conoce la nota, sino también por qué ha sido evaluado de esa forma, lo cual aporta una retroalimentación formativa que permite revisar los errores cometidos y comprender mejor el contenido evaluado. Esto añade un valor importante a la experiencia educativa, alineándose con los objetivos de mejora del aprendizaje a través del uso de inteligencia artificial.

▼ **Pregunta 2: Explica el ecosistema que se desarrolla en Drahom, el planeta subterráneo. ¿Qué adaptaciones han realizado las formas de vida que habitan en él?**

Comentario: El alumno describe que Drahom no oculta su ecosistema bajo tierra sino que tiene bosques en la superficie, lo cual es incorrecto según el temario, que indica que la superficie es árida y tóxica y que la vida se desarrolla en vastas redes de cavernas subterráneas. El alumno menciona domos internos con atmósferas ricas en humedad y oxígeno, lo cual es correcto. También habla de formas de vida inteligentes que construyen pueblos y aldeas, pero el temario no confirma la existencia de vida inteligente, solo menciona debate sobre posible civilización extinta sin evidencia clara. La respuesta es parcialmente correcta en cuanto a la existencia de ecosistemas subterráneos y adaptaciones a la oscuridad, pero contiene errores importantes sobre la superficie y la vida inteligente. La explicación es coherente pero no precisa en algunos aspectos esenciales.

Nota: 5.0

3.6.3.1 Ejemplo de comentario explicativo en una evaluación

De la misma manera, en las respuestas a las preguntas tipo test, se presenta una tabla en la que se muestran las opciones que el modelo considera correctas a partir del temario recuperado.

Pregunta	Opción seleccionada	Opción correcta	Evaluación
¿Qué tipo de atmósfera tiene Orvax, el planeta de gravedad cambiante?	Densa y rica en oxígeno	Densa y rica en oxígeno	✔ Bien
¿Cuál es la principal forma de vida en Zhyra, el santuario de niebla bioluminiscente?	Organismos que dependen de la bioluminiscencia	Organismos que dependen de la bioluminiscencia	✔ Bien
¿Qué fenómeno ocurre en Veltrion cuando cae la noche?	Desaparición de la atmósfera	Emisión de sonidos resonantes por los cristales	✗ Mal
¿Cuál es la temperatura promedio en la superficie de Thandor?	-220 °C	-220 °C	✔ Bien
¿Qué tipo de civilización se hipotetiza en Moltraz, el planeta cubierto de lava?	Civilización orgánica	Civilización silico-mecánica	✗ Mal

3.6.3.2 Ejemplo de la corrección de las respuestas a preguntas tipo test

4 Resultados y análisis

4.1 Evaluación del rendimiento del sistema

En esta sección se evalúa el rendimiento del sistema propuesto a partir de dos criterios fundamentales: la calidad de las respuestas generadas por el modelo y el tiempo de respuesta medio en cada uno de los principales procesos (generación y evaluación de exámenes). Para ello, se han realizado diversas pruebas funcionales simulando distintos escenarios de uso, con el objetivo de comprobar la robustez, coherencia y eficiencia del sistema en condiciones similares a las que tendría en un entorno real.

4.1.1 Calidad de respuesta

Tras llevar a cabo múltiples pruebas sobre el sistema, se ha observado que las preguntas generadas por el modelo de lenguaje están consistentemente alineadas con el temario ficticio de la asignatura. Además, se ha verificado que el modelo respeta de forma precisa las restricciones de los temas seleccionados por el usuario en el formulario de generación (comportamiento que inicialmente no respetaba en otros modelos como gpt-4o-mini). De esta manera confirmamos que el modelo consulta correctamente el sistema RAG en cada petición y se ajusta a las directrices definidas en el "prompt".

Cabe destacar que, aunque las preguntas son siempre relevantes, no presentan una alta variedad. Esto se debe principalmente a la extensión limitada del temario, que reduce la posibilidad de obtener una mayor diversidad de formulaciones sin caer en repeticiones.

En cuanto a la evaluación de las respuestas, se detectaron inicialmente varios problemas al utilizar modelos como gpt-4o-mini, los cuales tendían a confundir el contenido de las respuestas de los alumnos con información correcta del temario, además de no seguir adecuadamente los criterios de evaluación definidos. Sin embargo, tras adoptar el modelo gpt-4.1-mini, se observó una mejora significativa: las evaluaciones se ajustaron de forma más fiel al contenido del temario, las notas asignadas fueron más coherentes y realistas, y la retroalimentación generada resultó útil y precisa.

Por último, en el caso de las preguntas tipo test, el sistema mostró una alta fiabilidad desde el principio, dado que la corrección en este tipo de formato es más estructurada y menos dependiente de interpretaciones complejas por parte del modelo.

4.1.2 Tiempo de respuesta

El tiempo medio de respuesta del sistema varía según la funcionalidad ejecutada. Para el caso de la generación de preguntas de examen, el tiempo de respuesta oscila entre 5 y 15 segundos. En cambio, en el proceso de evaluación de respuestas, el tiempo medio se sitúa alrededor de los 25 segundos.

Estas diferencias se deben fundamentalmente al número de consultas realizadas a la base de datos vectorial de Pinecone. En la fase de generación, si el usuario selecciona múltiples temas, se requiere recuperar más información del sistema RAG, lo cual genera "prompts" más extensos que el modelo debe procesar. Del mismo modo, en la fase de evaluación, se realiza una consulta al sistema RAG por cada pregunta del examen, lo que incrementa significativamente el tiempo de procesamiento en función del número total de preguntas del examen.

4.2 Posibles mejoras

A pesar de que el sistema desarrollado ha mostrado resultados positivos y funcionales, existen varios aspectos mejorables, tanto a nivel de precisión y eficiencia como en términos de funcionalidades adicionales. Estas mejoras no solo permitirían aumentar la calidad general del sistema, sino también la posibilidad de integrar la aplicación en contextos reales más exigentes, como plataformas de evaluación en línea.

4.2.1 Precisión y eficiencia

Una de las principales áreas de mejora detectadas tiene que ver con la precisión semántica de las respuestas del modelo, especialmente en casos donde los enunciados de los exámenes o las respuestas de los estudiantes pueden resultar ambiguos. Aunque el sistema RAG proporciona un buen nivel de contexto al modelo de lenguaje, se podrían incorporar técnicas más avanzadas de filtrado de contexto o re-rankeo de resultados para garantizar que el contenido recuperado sea el más relevante y útil en cada caso.

Además, en el plano de eficiencia, se ha detectado que el tiempo de respuesta del sistema puede incrementarse de forma considerable cuando aumenta el número de temas seleccionados o la longitud de las respuestas a evaluar. Una posible mejora sería implementar un sistema de caché inteligente para consultas frecuentes o realizar un preprocesamiento de vectores y respuestas más ligero, con el objetivo de reducir la carga computacional y el tiempo de espera del usuario.

Otra opción sería la posibilidad de usar modelos de embedding más eficientes en términos de latencia y coste, o incluso explorar opciones como la reducción dimensional previa al almacenamiento en Pinecone para acelerar las búsquedas sin perder demasiada calidad semántica.

4.2.2 Funcionalidades

En cuanto a la funcionalidad general del sistema, existe margen para introducir diversas características adicionales que enriquecerían la experiencia del usuario. Por ejemplo:

- **Sistema de autenticación y gestión de usuarios**: permitiría guardar historiales de generación y evaluación por estudiante, así como personalizar configuraciones o revisar resultados pasados.
- **Modo profesor / modo alumno**: se podrían separar claramente los roles, permitiendo que el profesorado defina bancos de preguntas, pesos específicos o criterios de evaluación personalizados.
- **Editor visual de exámenes**: una interfaz más rica donde el usuario pueda reorganizar las preguntas, editar textos sugeridos o exportar el examen a diferentes formatos como PDF o Word.
- **Retroalimentación enriquecida**: incorporar explicaciones más pedagógicas, enlaces a recursos de ampliación o comparaciones automáticas con respuestas modelo podría mejorar el valor formativo del sistema.
- **Soporte multiasignatura**: el sistema actual trabaja con un único temario inventado, pero podría ampliarse fácilmente para soportar múltiples asignaturas, cada una con su propia base vectorial y su configuración.

Implementar estas mejoras permitiría acercar el sistema a un producto más maduro y versátil, capaz de adaptarse a entornos reales.

4.2.3 Integración en plataformas educativas

Una de las líneas de mejora más interesantes y con mayor impacto potencial del sistema desarrollado sería su integración directa con plataformas educativas ampliamente utilizadas, como Moodle, Canvas, Google Classroom o Microsoft Teams for Education. Esta integración permitiría al sistema aprovechar temarios reales de asignaturas existentes, facilitar el acceso de los docentes y automatizar aún más el proceso de generación y evaluación de exámenes.

Actualmente, el sistema funciona de manera autónoma con un temario estático cargado manualmente desde un documento externo, pero si se pudiera integrar con plataformas de gestión de aprendizaje sería posible:

- Recuperar de forma automática los contenidos del curso (temarios, apuntes, materiales, etc.).
- Adaptar la generación de preguntas al progreso del alumno o a los temas ya cubiertos.
- Asignar y devolver automáticamente los exámenes generados y sus evaluaciones a los estudiantes registrados.
- Aprovechar funcionalidades de la plataforma como la gestión de usuarios, permisos y roles docentes.

Entre las plataformas más compatibles se encuentran:

- **Google Classroom**: gracias a su API bien documentada, que permite acceder a archivos del curso, tareas, materiales y listas de usuarios.
- **Canvas LMS**: que ofrece una API REST robusta y flexible, ideal para integrar sistemas externos mediante herramientas como n8n o servicios intermedios.
- **Microsoft Teams for Education**: que a través de Microsoft Graph API permite consultar cursos, usuarios, tareas y materiales dentro de una infraestructura corporativa o institucional.

En el caso concreto de Moodle actualmente no es posible integrar el sistema desarrollado debido a que no dispone de una API abierta o directamente accesible desde n8n para recuperar temarios u otra información clave de manera sencilla. Aunque Moodle sí ofrece una API web, su uso suele estar restringido por permisos de administrador y una configuración compleja que varía entre instalaciones. Esto limita de momento su integración directa con flujos automatizados.

4.3 Limitaciones actuales

- **Tiempo de respuesta variable y sin control sobre la carga**: El tiempo de respuesta de los flujos depende, entre otros factores, del número de temas seleccionados, la longitud de los textos y la carga que puedan tener tanto el modelo LLM como la base de datos vectorial. No existe actualmente un sistema que permita gestionar esta carga de forma dinámica ni predecir con precisión el tiempo de respuesta. En un entorno con múltiples usuarios concurrentes, este diseño podría generar cuellos de botella o incluso fallos por tiempos de espera excesivos.

- **Limitación por modelo de lenguaje**: Aunque el modelo gpt-4.1-mini ha demostrado un rendimiento notable en la generación y evaluación de exámenes, sigue presentando limitaciones propias de los modelos de lenguaje actuales:
    - Falta de comprensión real: el modelo no "entiende" el contenido de forma humana, sino que predice tokens basándose en patrones. Esto puede generar errores sutiles en razonamiento o evaluación lógica.
    - Sensibilidad al "prompt" y al contexto: cambios mínimos en el "prompt" pueden producir respuestas incoherentes o no alineadas con la intención original del usuario.
    - Alucinaciones: aunque reducidas en comparación con versiones anteriores, pueden seguir produciéndose respuestas incorrectas o inventadas si el modelo interpreta mal la información del sistema RAG.
    - Coste computacional: incluso modelos más ligeros como el 4.1-mini tienen un coste en latencia y recursos, especialmente cuando se hace un uso intensivo en flujos secuenciales.
- **Evaluación subjetiva y dependiente del "prompt"**: Aunque se han logrado resultados coherentes y relevantes en la generación de preguntas y la evaluación de respuestas, la calidad final sigue estando fuertemente influenciada por el diseño del "prompt" y por las respuestas del propio modelo. No se ha implementado ningún sistema de verificación secundaria, re-evaluación o intervención humana, por lo que no es recomendable utilizar el sistema actual como herramienta de evaluación oficial sin supervisión.
- **Simplicidad del sistema RAG creado en n8n**:
  El sistema RAG implementado en este proyecto se ha desarrollado directamente sobre n8n por su accesibilidad y flexibilidad visual. Sin embargo, esta solución es relativamente simple y no escalable cuando se trata de gestionar temarios extensos o con estructuras jerárquicas complejas (por ejemplo, subtemas, referencias cruzadas o contenidos interdependientes). Para casos de mayor complejidad, sería recomendable utilizar soluciones más robustas como LangChain o LlamaIndex (frameworks diseñados específicamente para sistemas RAG más avanzados, que permiten un mejor control sobre la recuperación, segmentación y enriquecimiento del contexto).

5 Impacto y consideraciones éticas

5.1 Impacto medioambiental del uso de la IA

El uso de inteligencia artificial conlleva ciertos impactos medioambientales que conviene tener en cuenta, especialmente cuando se trabaja con modelos de lenguaje de gran tamaño. Aunque este proyecto hace uso de versiones más ligeras y su funcionamiento no es continuo, es importante destacar su potencial huella ecológica si se escalara a un entorno educativo con un gran número de usuarios.

Uno de los principales efectos está relacionado con el consumo energético. Entrenar modelos avanzados de IA, como los de la familia GPT, requiere una enorme cantidad de recursos computacionales. Según estimaciones, el entrenamiento de modelos como GPT-3 puede generar unas 300 toneladas de CO_2, una cantidad comparable a la de más de 100 vuelos transatlánticos [25]. Además, se prevé que los centros de datos asociados al uso de IA podrían representar hasta un 0,5% del consumo energético mundial en los próximos años [26].

A este consumo eléctrico se suma el gasto de agua necesario para mantener refrigerados los servidores. Algunas estimaciones apuntan a que entrenar un solo modelo de gran escala podría requerir cientos de miles de litros de agua, lo que puede suponer un problema si el despliegue de estas tecnologías sigue creciendo sin control [25].

Por otra parte, también hay que tener en cuenta los residuos electrónicos. La constante renovación del hardware (principalmente tarjetas gráficas, servidores y sistemas de almacenamiento) genera una gran cantidad de residuos tecnológicos. Para 2030, se estima que la IA podría contribuir con hasta 2,5 millones de toneladas de residuos electrónicos en todo el mundo [25].

Además, es importante considerar el papel que estas tecnologías pueden tener en relación con los Objetivos de Desarrollo Sostenible (ODS) de la Agenda 2030. Por un lado, el uso responsable de IA está directamente relacionado con el ODS 13: Acción por el clima, que llama a reducir las emisiones y mejorar la eficiencia energética [27]. También guarda relación con el ODS 12: Producción y consumo responsables, ya que la elección de modelos más eficientes y la reducción del consumo energético y de residuos electrónicos pueden contribuir a un sistema más sostenible [27]. Por otro lado, la solución propuesta, al facilitar el acceso a recursos educativos de forma automatizada, puede apoyar el ODS 4: Educación de calidad, al promover una enseñanza más accesible, personalizada y moderna [27]. Estos objetivos no solo orientan el desarrollo tecnológico hacia fines responsables, sino que también refuerzan la importancia de integrar criterios de sostenibilidad en la toma de decisiones técnicas.

En definitiva, aunque este proyecto se encuentra en un entorno de pruebas y utiliza modelos optimizados con un consumo moderado, la implantación a gran escala de soluciones basadas en IA debe ir acompañada de un análisis sobre su sostenibilidad y el uso responsable de los recursos.

5.2 Impacto educativo

La automatización de procesos como la generación de preguntas o la evaluación de respuestas representa un cambio importante en la forma en que se gestionan los procesos educativos.

Aspectos positivos:

- Agiliza tareas repetitivas, liberando tiempo para que el profesorado se centre en tareas de mayor relevancia para la enseñanza.
- Permite generar evaluaciones personalizadas rápidamente.
- Aporta retroalimentación automatizada inmediata, útil para el autoaprendizaje.

Riesgos:

- Depender excesivamente del modelo para evaluar puede ignorar aspectos subjetivos importantes (argumentación, creatividad, contexto).
- Si todos los exámenes se generan de forma algorítmica, podría tenderse a una simplificación de los contenidos o formatos repetitivos.
- La integración de estas herramientas exige conocimientos técnicos y acceso a recursos que no todos los centros educativos o estudiantes poseen.

En conjunto, la IA puede complementar el trabajo docente, pero nunca debe sustituir por completo el juicio profesional del profesorado.

5.3 Privacidad de datos académicos

La recopilación y procesamiento de datos personales en sistemas educativos basados en IA plantea preocupaciones significativas en términos de privacidad y protección de datos. En la Unión Europea, el Reglamento General de Protección de Datos (RGPD) establece directrices claras sobre cómo deben manejarse estos datos [28].

Consideraciones clave:

- **Consentimiento informado**: Los estudiantes deben ser plenamente conscientes de qué datos se recopilan, cómo se utilizan y con qué propósito.
- **Minimización de datos**: Solo deben recopilarse los datos estrictamente necesarios para el funcionamiento del sistema.
- **Seguridad de los datos**: Implementar medidas técnicas y organizativas adecuadas para proteger los datos contra accesos no autorizados o pérdidas.

6 Conclusiones

6.1 Generales

Este Trabajo de Fin de Grado ha permitido diseñar, implementar y analizar un sistema funcional para la automatización de la generación y evaluación de exámenes, utilizando inteligencia artificial, automatización de flujos y desarrollo web. A través de la combinación de herramientas como n8n, OpenAI, Pinecone y Angular, se ha construido un prototipo que demuestra cómo la tecnología puede aplicarse para reducir la carga de trabajo del profesorado y mejorar la experiencia educativa del alumnado.

Uno de los aspectos más destacados del sistema es la incorporación de una arquitectura RAG (Retrieval-Augmented Generation), que ha demostrado ser fundamental para asegurar que las preguntas y evaluaciones generadas estén directamente relacionadas con el contenido del temario. Gracias a esta arquitectura, y al uso de un temario completamente ficticio, ha sido posible comprobar que el modelo de lenguaje realmente consulta el sistema de recuperación de información para generar contenido relevante.

La aplicación web desarrollada en Angular, aunque sencilla, ha sido suficiente para ofrecer una interfaz clara e intuitiva. A través de formularios estructurados, se permite al usuario generar exámenes personalizados y obtener evaluaciones automatizadas con retroalimentación detallada. El uso de tecnologías como PrimeNG y TailwindCSS ha contribuido a crear una experiencia visual coherente y agradable.

En cuanto a los resultados obtenidos, las pruebas realizadas han mostrado un funcionamiento coherente del sistema tanto en la generación de preguntas como en la evaluación automática de respuestas. El uso del modelo GPT-4.1-mini ha permitido lograr un equilibrio adecuado entre calidad de las respuestas, coste y tiempo de respuesta. Además, se ha evidenciado que la automatización mediante n8n es una solución accesible, fácil de mantener y adaptable a otros contextos.

Por otro lado, también se han identificado limitaciones importantes. A nivel técnico, el sistema RAG construido en n8n, aunque funcional, no está diseñado para manejar temarios muy extensos o con estructuras jerárquicas complejas. Asimismo, los modelos de lenguaje, incluso los más avanzados, siguen presentando limitaciones en cuanto a comprensión profunda del contenido, sensibilidad al diseño del “prompt” y generación de resultados inconsistentes en ciertos casos. Además, la falta de integración con plataformas educativas reales, como Moodle, representa una barrera para su integración en contextos educativos reales, dado que no es posible recuperar automáticamente contenidos desde estas plataformas debido a limitaciones en sus APIs.

En resumen, este trabajo ha conseguido demostrar que es posible diseñar un sistema práctico y funcional que automatice parte del proceso de evaluación educativa, reduciendo tiempos, aumentando la objetividad y ofreciendo nuevas oportunidades para innovar en el ámbito de la educación. No obstante, también ha dejado claro que cualquier solución basada en IA debe desarrollarse con precaución, atendiendo tanto a sus limitaciones técnicas como a sus implicaciones éticas, medioambientales y educativas.

6.2 Cumplimiento de objetivos

Los objetivos definidos al inicio de este Trabajo de Fin de Grado se han cumplido de manera satisfactoria. A continuación, se detalla cómo se han abordado y alcanzado cada uno de ellos:

- **Diseñar un sistema automatizado que facilite la creación y corrección de exámenes**: Se ha desarrollado un prototipo funcional que permite generar preguntas a partir de un temario y evaluar las respuestas de los estudiantes de forma automatizada. El sistema combina herramientas modernas de desarrollo web, automatización y procesamiento de lenguaje natural, integrando todas las partes a través de flujos en n8n.
- **Implementar un sistema RAG (Retrieval-Augmented Generation)**: Se ha conseguido construir un sistema RAG sencillo pero eficaz, utilizando una base de datos vectorial en Pinecone junto con modelos de embedding y generación de texto de OpenAI. Esto ha permitido garantizar que el contenido generado por el modelo esté directamente basado en el temario proporcionado, demostrando su utilidad para contextos educativos.
- **Ofrecer retroalimentación automática al alumnado**: A través del flujo de evaluación desarrollado, el sistema no solo asigna una nota, sino que también proporciona comentarios individualizados para las respuestas de desarrollo, lo que aporta un valor añadido para el estudiante.
- **Explorar la integración con plataformas educativas y adaptabilidad a distintos contenidos**: Aunque no se ha logrado una integración completa con plataformas como Moodle debido a limitaciones en sus APIs, sí se ha analizado esta posibilidad y se ha planteado como una mejora viable en futuras versiones. Además, se ha demostrado que el sistema puede adaptarse a otros contenidos simplemente cargando un nuevo temario, gracias a la flexibilidad del sistema RAG.
- **Evaluar el impacto medioambiental y ético del uso de la IA en educación**: En el capítulo correspondiente se ha reflexionado sobre el coste energético de los modelos de IA, su huella de carbono y la gestión de datos personales en contextos educativos. Se ha recordado la necesidad de un uso responsable de estas tecnologías, especialmente en entornos como la educación.
- **Validar el sistema a través de pruebas funcionales**: Se han realizado diversas pruebas tanto de generación como de evaluación, que han permitido observar el comportamiento del sistema, ajustar los modelos utilizados y analizar los resultados. Estas pruebas han mostrado que el sistema responde de forma coherente, precisa y ajustada a las especificaciones dadas por el usuario.

En conjunto, el sistema desarrollado cumple con todos los objetivos planteados y sirve como base para futuras mejoras o ampliaciones. Además de demostrar su viabilidad técnica, el proyecto ha permitido reflexionar sobre cómo herramientas de IA y automatización pueden transformar procesos tradicionales en educación, abriendo nuevas posibilidades, pero también exigiendo nuevas responsabilidades.

7 Bibliografía

[1] Amazon Web Services, “¿Qué es un LLM (modelo de lenguaje de gran tamaño)?”, AWS, [Online]. Disponible: https://aws.amazon.com/es/what-is/large-language-model/

[2] IBM, “¿Qué son los grandes modelos de lenguaje (LLM)?”, IBM, [Online]. Disponible: https://www.ibm.com/es-es/think/topics/large-language-models

[3] Cloudflare, “¿Qué son los modelos LLM? | Grandes modelos de lenguaje”, Cloudflare, [Online]. Disponible: https://www.cloudflare.com/es-es/learning/ai/what-is-large-language-model/

[4] Amazon Web Services, “¿Qué es la generación aumentada por recuperación (RAG)?”, AWS, [Online]. Disponible: https://aws.amazon.com/es/what-is/retrieval-augmented-generation/

[5] Oracle, “¿Qué es la generación aumentada de recuperación (RAG)?”, Oracle, [Online]. Disponible: https://www.oracle.com/es/artificial-intelligence/generative-ai/retrieval-augmented-generation-rag/

[6] Google Cloud, “¿Qué es la generación aumentada por recuperación?”, Google Cloud, [Online]. Disponible: https://cloud.google.com/use-cases/retrieval-augmented-generation?hl=es

[7] IBM, “¿Qué es la automatización?”, IBM, [Online]. Disponible: https://www.ibm.com/es-es/topics/automation

[8] DocuWare, “¿Qué es la automatización de procesos?”, DocuWare, [Online]. Disponible: https://start.docuware.com/es/blog/automatizacion-de-procesos

[9] Amazon Web Services, “¿Qué es una aplicación web?”, AWS, [Online]. Disponible: https://aws.amazon.com/es/what-is/web-application/

[10] Kinsta, “¿Qué es una petición HTTP?”, Kinsta, [Online]. Disponible: https://kinsta.com/es/base-de-conocimiento/que-es-una-peticion-http/

[11] IBM, “¿Qué es Docker?”, IBM. [Online]. Disponible: https://www.ibm.com/es-es/topics/docker

[12] Red Hat, “¿Qué es Docker y cómo funciona?”, Red Hat. [Online]. Disponible: https://www.redhat.com/es/topics/containers/what-is-docker

[13] ComunicaWeb, “¿Qué es n8n? Automatización Open-Source”. [Online]. Disponible: https://comunica-web.com/blog/marketing-digital/que-es-n8n/

[14] OpenAI, “Introducing GPT-4.1 in the API,” 2024. [Online]. Disponible: https://openai.com/index/gpt-4-1/

[15] OpenAI, “New embedding models and API updates,” 2024. [Online]. Disponible: https://openai.com/index/new-embedding-models-and-api-updates/

[16] OpenAI, “GPT Guide.” [Online]. Disponible: https://platform.openai.com/docs/guides/gpt

[17] OpenAI, “Embeddings Guide.” [Online]. Disponible: https://platform.openai.com/docs/guides/embeddings

[18] Angular, “What is Angular?”, [Online]. Disponible: https://angular.dev/overview

[19] PrimeNG, “PrimeNG - Angular UI Component Library”, [Online]. Disponible: https://primeng.org/

[20] Tailwind CSS, “Tailwind CSS - Rapidly build modern websites without ever leaving your HTML”, [Online]. Disponible: https://tailwindcss.com/

[21] Pinecone, “Get Started”, [Online]. Disponible: https://docs.pinecone.io/guides/get-started/overview

[22] n8n Docs, “Character Text Splitter node”, [Online]. Disponible: https://docs.n8n.io/integrations/builtin/cluster-nodes/sub-nodes/n8n-nodes-langchain.textsplittercharactertextsplitter/

[23] n8n Docs, “Recursive Character Text Splitter node”, [Online]. Disponible:

https://docs.n8n.io/integrations/builtin/cluster-nodes/sub-nodes/n8n-nodes-langchain.textsplitterrecursivecharactertextsplitter/

[24] n8n Docs, “Token Splitter node”, [Online]. Disponible: https://docs.n8n.io/integrations/builtin/cluster-nodes/sub-nodes/n8n-nodes-langchain.textsplittertokensplitter/

[25] AFP, “El impacto medioambiental de la inteligencia artificial en cinco cifras,” Ojo al Clima, 2024. [Online]. Disponible: https://ojoalclima.com/articles/el-impacto-medioambiental-de-la-inteligencia-artificial-en-cinco-cifras

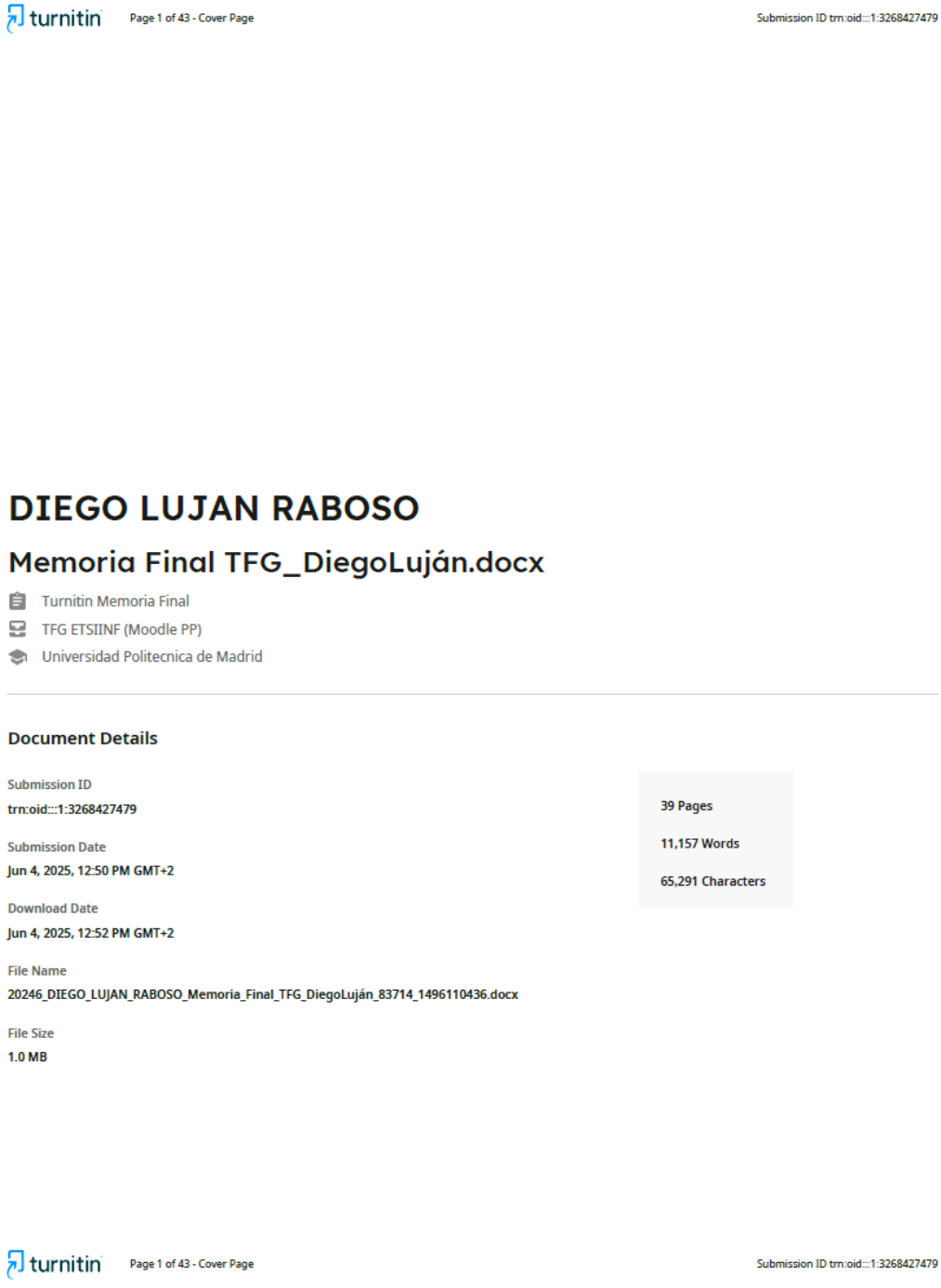
[26] K. Hao, “AI’s carbon footprint is bigger than you think,” MIT Technology Review, May 20, 2025. [Online]. Disponible: https://www.technologyreview.com/2025/05/20/1116327/ai-energy-usage-climate-footprint-big-tech/

[27] Naciones Unidas, “Objetivos de Desarrollo Sostenible,” UN.org, 2024. [Online]. Disponible: https://www.un.org/sustainabledevelopment/es/objetivos-de-desarrollo-sostenible/

[28] Agencia Española de Protección de Datos, “Adecuación al RGPD de tratamientos que incorporan Inteligencia Artificial,” AEPD, [Online]. Disponible: https://www.aepd.es/guias/adecuacion-rgpd-ia.pdf

8 Anexos

8.1 Informe de originalidad de Turnitin



turnitin Page 1 of 43 - Cover Page Submission ID trn:oid:::1:3268427479

DIEGO LUJAN RABOSO

Memoria Final TFG_DiegoLuján.docx

Turnitin Memoria Final

TFG ETSIINF (Moodle PP)

Universidad Politecnica de Madrid

Document Details

Submission ID
trn:oid:::1:3268427479

Submission Date
Jun 4, 2025, 12:50 PM GMT+2

Download Date
Jun 4, 2025, 12:52 PM GMT+2

File Name
20246_DIEGO_LUJAN_RABOSO_Memoria_Final_TFG_DiegoLuján_83714_1496110436.docx

File Size
1.0 MB

39 Pages

11,157 Words

65,291 Characters

turnitin Page 1 of 43 - Cover Page Submission ID trn:oid:::1:3268427479

4% Overall Similarity

The combined total of all matches, including overlapping sources, for each database.

Filtered from the Report

- Bibliography
- Quoted Text

Exclusions

- 1 Excluded Source

Top Sources

0% Internet sources
0% Publications
4% Submitted works (Student Papers)

turnitin Page 2 of 43 - Integrity Overview Submission ID trn:oid:::1:3268427479

Top Sources

0% Internet sources
0% Publications
4% Submitted works (Student Papers)

Top Sources

The sources with the highest number of matches within the submission. Overlapping sources will not be displayed.

1	Student papers	Universidad Politécnica de Madrid 2%
2	Student papers	Universitat Politècnica de València <1%
3	Student papers	Universidad Europea de Madrid <1%
4	Student papers	Instituto Superior de Artes, Ciencias y Comunicación IACC <1%
5	Student papers	Colegio Universitario de Estudios Financiero <1%
6	Student papers	Centro Europeo de Postgrado - CEUPE <1%
7	Student papers	Universidad TecMilenio <1%
8	Student papers	Universidad Manuela Beltrán <1%
9	Student papers	Consorcio CIXUG <1%
10	Student papers	Universidad Anahuac México Sur <1%
11	Student papers	Universidad Autonoma De Guadalajara A.C. <1%

turnitin Page 3 of 43 - Integrity Overview Submission ID trn:oid:::1:3268427479

12 Student papers
Universidad Rey Juan Carlos <1%

13 Student papers
Universidad San Marcos <1%

14 Student papers
Universidad de Burgos UBUCEV <1%

turnitin Page 4 of 43 - Integrity Overview Submission ID trn:oid:::1:3268427479

Este documento esta firmado por

	**Firmante**	CN=tfgm.fi.upm.es, OU=CCFI, O=ETS Ingenieros Informaticos - UPM, C=ES
	Fecha/Hora	Wed Jun 04 18:02:50 CEST 2025
	Emisor del Certificado	EMAILADDRESS=camanager@etsiinf.upm.es, CN=CA ETS Ingenieros Informaticos, O=ETS Ingenieros Informaticos - UPM, C=ES
	Numero de Serie	561
	Metodo	urn:adobe.com:Adobe.PPKLite:adbe.pkcs7.sha1 (Adobe Signature)