Trabajo de Fin de Grado

Gestor web de correcciones de exámenes mediante el uso de la API de ChatGPT

TITULACIÓN: Grado en Ingeniería Informática

AUTOR: Abián Sánchez Martel

TUTORIZADO POR:
Cayetano Guerra Artal

Julio 2024

Resumen

El presente trabajo de fin de título (TFT) se centra en la creación de un asistente de corrección de exámenes integrando inteligencia artificial (IA), en concreto la API de ChatGPT.

Es así como surge **Corrector Web**, una innovadora aplicación web que facilita a los profesores la tarea de evaluar exámenes de manera eficiente y precisa. Está diseñado para trabajar con archivos en formato CSV y XLS, permitiendo la carga directa de los exámenes con las respuestas de los estudiantes.

Una de las principales características de **Corrector Web** es su capacidad para utilizar rúbricas de evaluación. Estas rúbricas permiten a los docentes definir criterios específicos para cada pregunta, asegurándose así una corrección consistente y justa.

El motor de corrección que esta aplicación emplea está impulsado por la tecnología de inteligencia artificial de OpenAI, específicamente el modelo ChatGPT. Este modelo de procesamiento de lenguaje natural permite al asistente interpretar y evaluar las respuestas de los estudiantes con bastante precisión. ChatGPT analiza las respuestas en función de las rúbricas proporcionadas, cuando las haya, asignando puntuaciones y proporcionando justificación de las puntuaciones.

En síntesis, **Corrector Web** es una herramienta innovadora que ofrece una solución moderna y eficiente en la corrección de exámenes. Este sistema no solo ahorra tiempo valioso a los educadores, sino que también garantiza una corrección uniforme y objetiva de los exámenes.

Abstract

The present final degree project (TFT) focuses on the creation of an exam correction assistant integrating artificial intelligence (AI), specifically the ChatGPT API.

This is how **Corrector Web** emerged, an innovative web application that facilitates the task of evaluating exams for teachers in an efficient and accurate manner. It is designed to work with CSV and XLS file formats, allowing direct upload of exams with student responses.

One of the main features of **Corrector Web** is its ability to use evaluation rubrics. These rubrics allow teachers to define specific criteria for each question, thus ensuring consistent and fair grading.

The correction engine used by this application is powered by OpenAI's artificial intelligence technology, specifically the ChatGPT model. This natural language processing model allows the assistant to interpret and evaluate student responses with considerable accuracy. ChatGPT analyzes the responses based on the provided rubrics, if available, assigning scores and providing justification for the scores.

In summary, **Corrector Web** is an innovative tool that offers a modern and efficient solution for exam correction. This system not only saves valuable time for educators but also guarantees uniform and objective grading of exams.

Índice general

Índice de figuras

Capítulo 1

Introducción

> La inteligencia artificial no es nuestra competencia, es nuestra compañera.
>
> Ginni Rometty

En un mundo donde la tecnología avanza a pasos agigantados, podemos observar cómo la inteligencia artificial transforma muchos aspectos de la sociedad, desde los asistentes virtuales en nuestros dispositivos hasta los sistemas de recomendación en plataformas de streaming.

La inteligencia artificial es un campo de la informática que se enfoca en el desarrollo de sistemas y máquinas capaces de realizar tareas que típicamente requieren inteligencia humana, como el reconocimiento de patrones, el procesamiento del lenguaje natural, la toma de decisiones y el aprendizaje automático. Estos sistemas utilizan algoritmos y modelos matemáticos para interpretar datos, aprender de experiencias pasadas y ajustar su comportamiento en función de nuevos inputs. [1]

1.1. Historia y evolución de la Inteligencia Artificial

El término **"inteligencia artificial"** fue acuñado por John McCarthy en 1956 durante la conferencia en Dartmouth College. Este evento es ampliamente considerado como el nacimiento oficial del campo de la IA. Reunió a destacados investigadores como Marvin Minsky, Nathaniel Rochester y Claude Shannon, quienes discutieron cómo hacer que las máquinas usaran el lenguaje, resolvieran problemas y mejoraran a sí mismas. [2]

La conferencia impulsó el desarrollo de los primeros programas de IA y sentó las bases para

futuras investigaciones. Este hito histórico marcó el inicio de un campo que ha evolucionado notablemente.

1.1.1. Primeros Avances (1950-1970)

Durante estas dos décadas, se realizaron avances significativos en áreas como la resolución de problemas y el procesamiento de lenguaje natural. Se desarrollaron programas capaces de resolver problemas matemáticos y jugar ajedrez a nivel básico. Sin embargo, las limitaciones tecnológicas de la época y las expectativas desmesuradas llevaron a un progreso más lento del esperado.

En la década de 1960, Joseph Weizenbaum desarrolló **ELIZA**[3], uno de los primeros sistemas de procesamiento de lenguaje natural, que simulaba una conversación con un psicoterapeuta. **ELIZA** demostraba la capacidad de las computadoras para interactuar con los humanos mediante lenguaje natural, aunque de manera muy limitada y basada en reglas predefinidas.

En la década de 1970, apareció el programa **SHRDLU**[4], desarrollado por Terry Winograd en el MIT, que simulaba la comprensión y manipulación de objetos tridimensionales en un mundo virtual. **SHRDLU** demostraba cómo una computadora podía entender y responder a instrucciones en lenguaje natural en un contexto limitado.

1.1.2. La Primera "crisis de la IA"(1970s-1980s)

A medida que se hizo evidente que las máquinas no podían replicar fácilmente la inteligencia humana, el entusiasmo inicial por la IA se enfrió, dando paso a lo que se conoce como el **"Invierno de la IA"**. Durante este período, hubo una reducción significativa en la financiación y el interés en este campo emergente. Las expectativas sobre las capacidades de la IA no se cumplieron, y las limitaciones tecnológicas y computacionales se volvieron más evidentes, lo que llevó a un escepticismo generalizado sobre el futuro de la inteligencia artificial.

Sin embargo, a pesar del declive en la inversión y el entusiasmo, la investigación en IA no se detuvo por completo. Se centró en áreas específicas como los sistemas expertos, que utilizaban reglas predefinidas para tomar decisiones en dominios limitados. Estos sistemas encontraron aplicaciones prácticas en medicina e industria, demostrando que la IA tenía potencial para resolver problemas complejos dentro de ciertos límites. A medida que avanzaban las décadas

y mejoraban los algoritmos y la capacidad computacional, se sentaron las bases para futuros avances significativos en el campo de la inteligencia artificial.[5]

1.1.3. Renacimiento y Progreso Rápido (1990s-2000s)

El aumento del poder computacional y los avances en el almacenamiento de datos permitieron a los investigadores abordar problemas más complejos. En 1997, el superordenador **Deep Blue**[6] de IBM venció al campeón mundial de ajedrez Garry Kasparov, convirtiendose en el primer sistema en conseguirlo, demostrando el potencial de la IA en tareas específicas. En los años 2000, el auge de Internet y la disponibilidad de grandes conjuntos de datos impulsaron el desarrollo de algoritmos de aprendizaje automático (machine learning).

1.1.4. La Era del Deep Learning (2010s-Presente)

En la última década se ha visto un crecimiento exponencial en la capacidad de las máquinas para aprender y procesar información. El deep learning, una rama del aprendizaje automático que utiliza redes neuronales profundas, ha permitido avances impresionantes en reconocimiento de voz, procesamiento de lenguaje natural y visión por computadora. Empresas como Google, Facebook y OpenAI han liderado el camino en la investigación y aplicación de estas tecnologías .

En 2011, **Watson** de IBM captó la atención mundial al vencer a los campeones humanos en el concurso de televisión "Jeopardy!". Watson utilizó procesamiento de lenguaje natural y aprendizaje automático para interpretar preguntas complejas y buscar respuestas en una vasta base de datos. Este logro destacó el potencial de la IA para manejar grandes volúmenes de información y resolver problemas en tiempo real. Desde entonces, IBM ha aplicado la tecnología de Watson en campos como la medicina, donde ayuda a los médicos a diagnosticar enfermedades y a recomendar tratamientos basados en grandes cantidades de datos clínicos.[7] [8]

1.2. El Impacto de los Modelos de Lenguaje Generativo

En los últimos años, los modelos de lenguaje natural han revolucionado el campo de la IA, destacándose especialmente la serie Generative Pre-trained Transformer **(GPT)** de OpenAI. Lanzado en 2020, demostró su capacidad para generar texto coherente y relevante en diversas

tareas, gracias a sus 175 mil millones de parámetros, marcando así un hito en la inteligencia artificial moderna.

La serie GPT ha mostrado que los modelos de lenguaje pueden comprender y generar texto de manera casi humana, con aplicaciones prácticas en servicio al cliente, generación de contenido y educación. Sin embargo, también ha planteado cuestiones éticas y de seguridad debido a su potencial para generar información falsa o sesgada.[9]

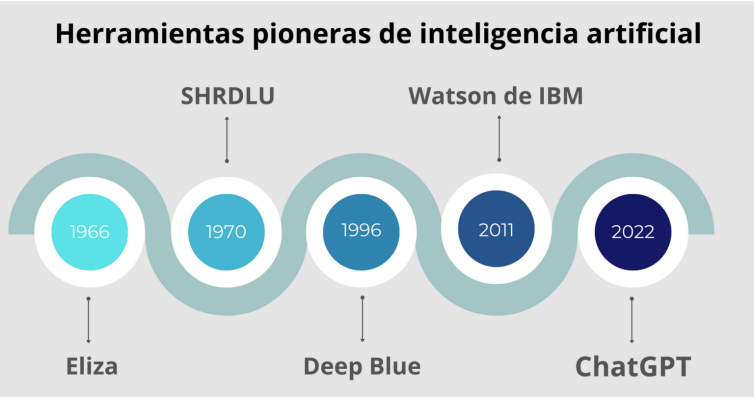


Ilustración 1.1: Herramientas pioneras de la inteligencia artificial [1]

Capítulo 2

Motivación y Objetivos Iniciales

2.1. Estado Actual y Motivación

La práctica de la vida cotidiana prueba que, la inteligencia artificial se utiliza en una amplia gama de aplicaciones, desde asistentes virtuales como Siri y Alexa hasta vehículos autónomos y diagnóstico médico. La IA está transformando industrias enteras, mejorando la eficiencia y creando nuevas oportunidades.

El ascenso de la inteligencia artificial está cambiando radicalmente la dinámica de la evaluación académica y profesional. Gracias a su capacidad para analizar grandes volúmenes de datos de manera eficiente y precisa, la IA automatiza procesos que antes eran laboriosos y propensos a errores humanos. Desde la corrección de exámenes hasta la evaluación de competencias y habilidades en diversos campos, la IA se ha convertido en una herramienta invaluable para garantizar la consistencia y objetividad en las decisiones evaluativas.

En el ámbito educativo, sin embargo, la mayoría de los centros educativos todavía emplean el método tradicional de evaluación. Los profesores enfrentan la ardua tarea de corregir una gran cantidad de exámenes, a menudo de cientos de estudiantes, lo que representa una carga significativa de trabajo que puede consumir mucho tiempo. Esta situación no solo afecta a los docentes, sino que también genera largas esperas para los estudiantes que ansiamos recibir sus calificaciones.

En respuesta a esta problemática, surge **Corrector Web**, que no solo busca optimizar la corrección de exámenes, sino también proporcionar a los educadores herramientas avanzadas para gestionar y mejorar el proceso de evaluación académica.

2.2. Objetivos Iniciales

Este proyecto se enfoca en desarrollar una plataforma de corrección de exámenes mediante inteligencia artificial, a través de herramientas avanzadas para que los educadores gestionen y optimicen el proceso de evaluación.

Los objetivos específicos del proyecto son los siguientes:

- Desarrollar una interfaz web amigable que permita a los docentes gestionar archivos de exámenes y redactar rúbricas.
- Implementar la API de OpenAI para la corrección automática de exámenes basada en las rubricas previamente definidas.
- Realizar pruebas para asegurar la precisión y fiabilidad de las correcciones automáticas.
- Preparar material formativo para usuarios finales.

Capítulo 3

Competencias específicas y aportaciones del trabajo

3.1. Competencias

El desarrollo del presente trabajo ha permitido cumplir exitosamente algunas competencias específicas, las cuales serán mostradas detalladamente y justificadas a continuación.

3.1.1. IS01

"Capacidad para desarrollar, mantener y evaluar servicios y sistemas software que satisfagan todos los requisitos del usuario y se comporten de forma fiable y eficiente, sean asequibles de desarrollar y mantener y cumplan normas de calidad, aplicando las teorías, principios, métodos y prácticas de la ingeniería del software."

Esta competencia se refleja en la necesidad de mejorar la eficiencia y precisión en la corrección de exámenes para reducir la carga de trabajo de los docentes. Para ello, se propusieron soluciones en las fases de análisis y diseño que aseguran una interacción cómoda e intuitiva de los usuarios con la aplicación.

3.1.2. IS04

"Capacidad de identificar y analizar problemas y diseñar, desarrollar, implementar, verificar y documentar soluciones software sobre la base de un conocimiento adecuado de las teorías, modelos y técnicas actuales".

En el proceso de análisis y desarrollo de este proyecto, se aportaron soluciones de software que cumplieron con los propósitos preestablecidos. Se aplicó una metodología ágil para desarrollar diversas tareas, proporcionando soluciones a diferentes objetivos. Además, se realizaron cambios en el diseño de la aplicación para resolver inconvenientes encontrados durante el proceso. Por lo tanto, esta competencia se justifica plenamente.

3.2. Aportaciones

Este asistente de corrección de exámenes presenta aportaciones no solo en el ámbito social, sino también en el tecnológico.

3.2.1. Entorno Social

Eficiencia Laboral: La capacidad de **Corrector Web** para reducir significativamente el tiempo de corrección es una de sus principales ventajas, ya que la inteligencia artificial puede evaluar un gran número de exámenes en una fracción del tiempo requerido por un profesor. Esto libera recursos humanos para que los educadores puedan concentrarse en actividades más interactivas y personalizadas con los estudiantes, mejorando así la calidad de la enseñanza y optimizando la experiencia educativa en general.

Evaluación Imparcial: La **precisión, exactitud y consistencia** de esta aplicación mejora significativamente el proceso de evaluación. La IA puede aplicar criterios de corrección, proporcionados por los profesores, de manera uniforme y consistente, asegurando que todos los exámenes sean evaluados bajo los mismos estándares y disminuyendo el margen de error humano, garantizando que todos los estudiantes sean juzgados bajo parámetros equitativos y uniformes.

Acceso Equitativo: Al mejorar la eficiencia y reducir en parte las cargas administrativas, el asistente de corrección puede contribuir a una distribución más equitativa de recursos educativos. Esto es especialmente relevante en entornos donde el acceso a la educación de calidad puede estar limitado por recursos o infraestructura educativa insuficiente.

3.2.2. Entorno Técnico

Integración Tecnológica: La implementación de IA en la corrección de exámenes no solo mejora la precisión y la velocidad del proceso de evaluación, como se describe en el apartado anterior, sino que también impulsa la adopción de tecnologías avanzadas en el sector académico. Esto fomenta la innovación y la modernización de otros procesos, como el desarrollo de plataformas digitales educativas y herramientas de aprendizaje adaptativo.

Desarrollo de Herramientas: La demanda y la implementación de asistentes de corrección basados en IA promueven el desarrollo continuo de herramientas educativas más sofisticadas. Esto incluye mejoras en la interoperabilidad de sistemas, la integración de análisis de datos educativos y la personalización del aprendizaje para adaptarse mejor a las necesidades individuales de los estudiantes.

Capítulo 4

Análisis

En este capítulo se realizará una descripción detallada de los diferentes requisitos y casos de uso del sistema. Además, se explicarán las tecnologías y herramientas empleadas en el desarrollo del proyecto, destacando sus características y, cuando corresponda, la manera en que contribuyen al logro de los objetivos establecidos.

4.1. Requisitos y Casos de Uso

4.1.1. Requisitos Funcionales

- El sistema deberá permitir a los usuarios (profesores) autenticarse mediante email y contraseña.
- Los profesores podrán subir exámenes y visualizar, archivos de tipo “csv, exls”
- Los examenes podrán contener diferentes tipos de preguntas, como opción múltiple, verdadero o falso, preguntas cortas y de desarrollo. Ademas de las respuestas de los estudiantes.
- Los docentes podrán crear y modificar las rúbricas para cada pregunta de un examen.
- El sistema deberá poder evaluar respuestas de exámenes en base a la rúbrica.
- Corrector Web debe corregir el examen de todos los estudiantes.

4.1.2. Requisitos No Funcionales

- La interfaz debe ser intuitiva y fácil de usar.
- El tiempo de respuesta de la interfaz de usuario debe ser menor a 2 segundos para la mayoría de las acciones.

4.2. Casos de Uso

En esta sección se describirán los diferentes casos de uso de **Corrector Web**. Es importante recordar que esta herramienta está diseñada para facilitar la creación, gestión y corrección de exámenes por parte de usuarios registrados. A continuación, se detallarán los actores que interactúan con el sistema, así como los casos de uso que describen sus principales interacciones y opciones disponibles.

4.2.1. Registrar Usuario

Permite a un usuario no registrado crear una cuenta en el sistema.

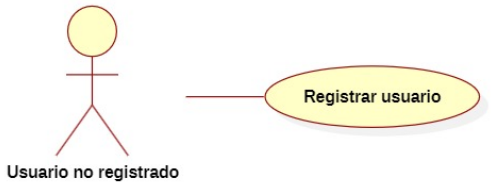


Ilustración 4.1: Caso de Uso Registrar Usuario

Flujo Principal:

1. El usuario proporciona sus datos (nombre, email y contraseña).
2. El sistema verifica que el usuario no esté ya registrado.
3. El sistema guarda los datos, crea una cuenta e inicia la sesión
4. El usuario es redirigido a la creación de examen.

4.2.2. Login

Permite a un usuario registrado iniciar sesión en el sistema

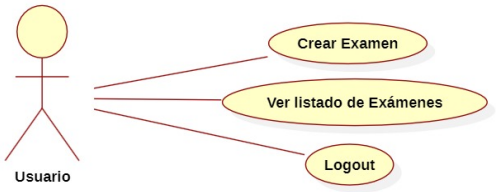


Ilustración 4.2: Caso de Uso Login

Flujo Principal:

1. El usuario proporciona sus datos (email y contraseña).
2. El sistema verifica los datos
3. El sistema inicia la sesión
4. El usuario es redirigido al listado de exámenes.

4.2.3. Crear Examen

Permite a un usuario crear un nuevo examen.

Flujo Principal:

1. El sistema solicita los detalles del examen (Nombre, Fecha, Modelo, Temperatura, subir archivo en formato xls o csv).
2. El usuario ingresa los detalles y guarda el examen.
3. El sistema confirma la creación del examen.
4. El usuario es redirigido al listado de examenes.

4.2.4. Ver Listado de Exámenes

Permite al usuario ver un listado de los exámenes que ha creado.

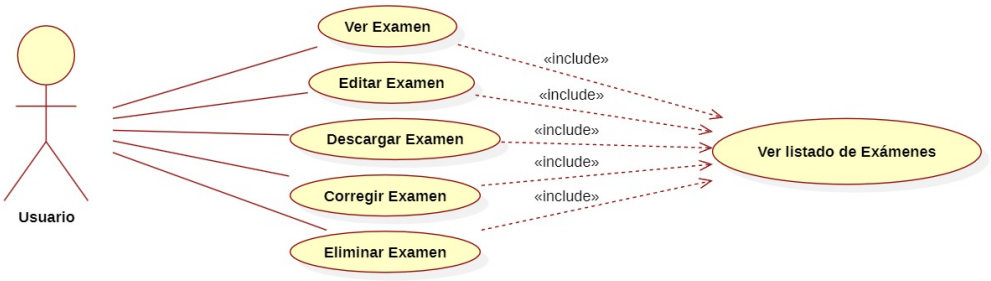


Ilustración 4.3: Caso de Uso Ver Listado de Exámenes

Flujo Principal:

1. El usuario selecciona la opción de ver listado de exámenes.
2. El sistema muestra un listado de los exámenes creados por el usuario.
3. El usuario puede navegar entre las opciones disponibles
   a) Ver Examen
   b) Editar Examen
   c) Descargar Examen
   d) Corregir Examen
   e) Eliminar Examen

4.2.5. Ver Examen

Permite al usuario ver los detalles de un examen específico.

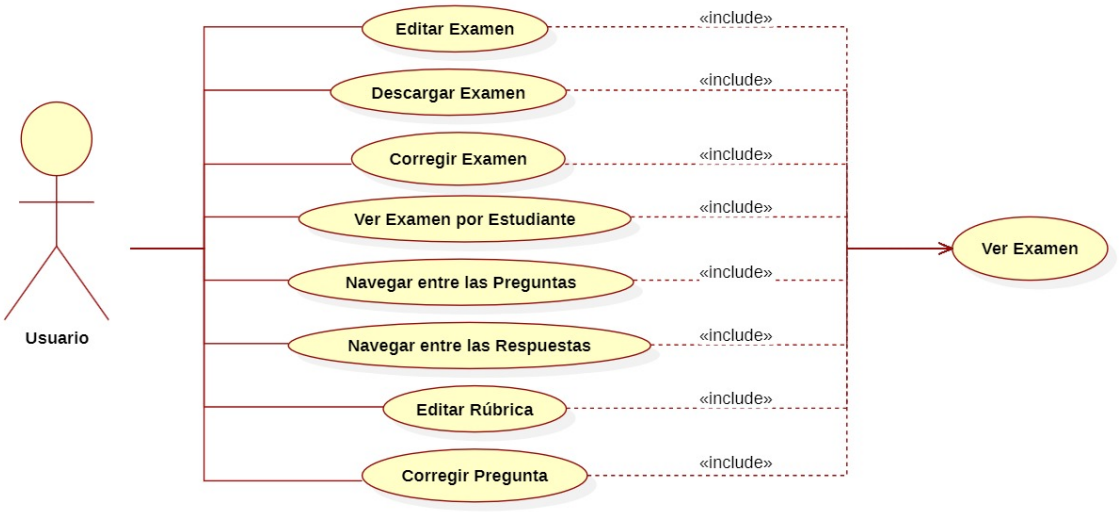


Ilustración 4.4: Caso de Uso Ver Examen

Flujo Principal:

1. El usuario selecciona la opción de ver examen.
2. El sistema muestra los detalles del examen (Nombre, Fecha, Número de preguntas, Número de estudiantes, Modelo, Temperatura, Preguntas, Respuestas, Rubricas, Calificación Justificación)
3. El usuario puede navegar entre las opciones disponibles
   a) Editar Examen
   b) Descargar Examen

c) Corregir Examen
d) Ver Examen por Estudiante
e) Navegar entre las Preguntas
f) Navegar entre las Respuestas (estudiantes)
g) Editar Rúbrica
h) Corregir pregunta

4.2.6. Corregir Examen

Permite al usuario corregir un examen completo.

Flujo Principal:

1. El usuario selecciona la opción de corregir examen.
2. El sistema se conecta con la API de OpenAI, donde se corregirán las preguntas en función de la rubrica correspondiente a cada pregunta, del Modelo y de la temperatura.
3. El sistema muestra un mensaje de éxito.

4.2.7. Descargar Examen

Permite al usuario descargar un examen en formato xls

Flujo Principal:

1. El usuario selecciona la opción de descargar examen.
2. El sistema genera un archivo descargable del examen.
3. El usuario descarga el archivo.

4.2.8. Navegar entre Preguntas del Examen

Permite al usuario navegar entre las preguntas de un examen específico.

Flujo Principal:

1. El usuario selecciona la opción siguiente/anterior para navegar entre preguntas.
2. El sistema permite al usuario ver cada pregunta, una por una.

4.2.9. Navegar entre las Respuestas

Permite al usuario navegar entre las respuestas que, los estudiantes, han realizado en un examen específico.

Flujo Principal:

1. El usuario selecciona la opción siguiente/anterior entre respuestas de estudiantes.
2. El sistema permite al usuario ver cada respuesta de un estudiante, a la pregunta seleccionada

4.2.10. Editar Rúbricas

Permite al usuario editar la rúbrica de calificación de una pregunta del examen.

Flujo Principal:

1. El sistema muestra la rúbrica de la pregunta actual.
2. El usuario hace los cambios deseados y selecciona la opción Actualizar Rúbrica.
3. El sistema guarda los cambios.

4.2.11. Corregir Pregunta

Permite al usuario corregir una sola pregunta de un examen.

Flujo Principal:

1. El usuario selecciona la opción de corregir pregunta.
2. El sistema se conecta con la API de OpenAI y corrige la pregunta, en función de la rúbrica suministrada, modelo y temperatura.
3. Es sistema guarda la calificación y el Razonamiento de la nota asignada.
4. El sistema muestra la calificación y el Razonamiento.

4.3. Tecnologías estudiadas y utilizadas

En este apartado se detallan las tecnologías estudiadas y utilizadas en el desarrollo de **Corrector Web**. Estas herramientas y plataformas no solo permiten la implementación eficiente de las funcionalidades requeridas, sino que también garantizan robustez y escalabilidad

del sistema. A continuación, se describen las tecnologías seleccionadas, junto con su relevancia y aplicación específica en el proyecto.

4.3.1. Python

Python es un lenguaje de programación de alto nivel, conocido por su simplicidad y legibilidad. En el proyecto **Corrector Web**, Python se utilizó para el desarrollo del backend debido a su extensa biblioteca de módulos y la facilidad con la que permite implementar algoritmos complejos de inteligencia artificial.

4.3.2. Jinja

Jinja[10] es un motor de plantillas para Python, diseñado para ser rápido, expresivo y extensible. Esto hace posible separar la lógica de la aplicación del diseño, facilitando la creación de HTML dinámico. Jinja se integra perfectamente con frameworks web como Flask, lo que permite generar páginas web dinámicas con datos provenientes de la lógica de la aplicación. Esto se logra con el uso de variables, expresiones y control de flujo directamente en las plantillas HTML. Jinja ha sido de gran ayuda en el desarrollo de este asistente, ha permitido crear interfaces de usuario dinámicas de manera bastante sencilla.

4.3.3. Flask

Flask[11] es un microframework para aplicaciones web en Python. Su ligereza y flexibilidad lo convierten en una excelente opción para proyectos que requieren una configuración rápida y sencilla. En **Corrector Web**, Flask se usó para crear el servidor web y gestionar las rutas de la aplicación, facilitando la comunicación entre el frontend y el backend.

4.3.4. Bootstrap-Flask

Bootstrap-Flask[12] es una extensión de Flask que integra fácilmente el popular framework de CSS Bootstrap. Esta herramienta se utilizó para diseñar una interfaz de usuario responsiva y atractiva, asegurando que **Corrector Web** sea accesible y fácil de usar.

4.3.5. Pandas

Pandas[13] es una biblioteca de Python especializada en la manipulación y análisis de datos. Se empleó en **Corrector Web** para procesar los archivos XLS que contienen las respuestas de los estudiantes, facilitando la carga y gestión de grandes volúmenes de datos de manera sencilla y eficiente.

4.3.6. Chardet

Chardet [14] es una biblioteca de Python diseñada para la detección automática de codificación de caracteres en archivos y textos. En **Corrector Web** se empleaó para procesar los archivos CSV que contienen las respuestas de los estudiantes.

4.3.7. Werkzeug

Werkzeug[15] es una biblioteca de utilidades para aplicaciones web en Python, que proporciona funciones de enrutamiento y depuración. En el desarrollo de **Corrector Web**, Werkzeug se utilizó para mejorar la seguridad, el manejo de contraseñas, garantizando un entorno de ejecución seguro y confiable.

4.3.8. Blueprint

Blueprint[16] es un componente de Flask que permite dividir una aplicación grande en módulos más pequeños y manejables. En **Corrector Web**, Blueprint se utilizó para organizar el código del proyecto en diferentes módulos, mejorando la estructura y mantenibilidad del código.

4.3.9. Dotenv

Dotenv[17] es una herramienta que carga variables de entorno desde un archivo `.env` a las variables de entorno de Python. En el proyecto, Dotenv se empleó para gestionar configuraciones sensibles como claves de API y credenciales de base de datos, asegurando que no se expongan en el código fuente.

4.3.10. API de OpenAI

La API de OpenAI[18] proporciona acceso a modelos de lenguaje avanzados, como GPT-3, que pueden realizar una amplia gama de tareas de procesamiento de lenguaje natural. Estos modelos están entrenados en grandes volúmenes de datos y son capaces de entender y generar texto de manera coherente y contextual. La API permite a los desarrolladores integrar fácilmente estas capacidades en sus aplicaciones, permitiendo realizar tareas como redacción de textos, traducción, resumen, y mucho más.

4.3.11. ChatGPT

ChatGPT[19] es una implementación específica de los modelos GPT de OpenAI, diseñada para generar texto de manera conversacional. Utilizando la API de OpenAI, ChatGPT puede mantener diálogos, responder preguntas, ofrecer explicaciones y generar texto continuo en función de las entradas proporcionadas por los usuarios. Es una herramienta poderosa para crear asistentes virtuales, bots de servicio al cliente, y otras aplicaciones que requieren una interacción natural y fluida con los usuarios. En el contexto de **Corrector Web**, ChatGPT se utiliza para evaluar exámenes proporcionando retroalimentación detallada y precisa, mejorando así la calidad y eficiencia del proceso de corrección.

4.3.12. MariaDB

MariaDB[20] es un sistema de gestión de bases de datos relacional, derivado de MySQL. Fue la base de datos elegida para almacenar toda la información relacionada con los exámenes y usuarios de **Corrector Web**, debido a su robustez, escalabilidad y alto rendimiento.

4.3.13. SQLAlchemy

SQLAlchemy[21] es un ORM (Object-Relational Mapper) para Python que simplifica las interacciones con bases de datos relacionales. En el proyecto, SQLAlchemy se usó para manejar la persistencia y manejo de datos, permitiendo realizar operaciones CRUD (Crear, Leer, Actualizar, Eliminar) de manera eficiente y segura.

4.3.14. UML

UML (Unified Modeling Language) es un lenguaje de modelado estándar utilizado para especificar, visualizar y documentar los componentes de un sistema. Durante la elaboración de la memoria, UML se utilizó para crear diagramas de casos de uso, ayudando a planificar y comunicar la arquitectura del sistema de manera clara y precisa.

4.3.15. LaTeX

LaTeX es un sistema de preparación de documentos que permite crear documentos de alta calidad tipográfica. Se utiliza para la elaboración de la presente memoria, asegurando un formato profesional y fácil de leer.

4.3.16. Material Design

Material[22] es un sistema de diseño desarrollado por Google que proporciona una guía completa de experiencia de usuario (UX) y un conjunto de directrices y componentes para crear interfaces de usuario atractivas y consistentes.

Capítulo 5

Diseño

En este capítulo se detalla la arquitectura y estructura de **Corrector Web**, incluyendo diagramas de clases, modelo de datos e interfaz de usuario.

5.1. Arquitectura del Sistema

Corrector Web combina el patrón Modelo-Vista-Controlador (MVC) con un modelo de comunicación Cliente-Servidor. Esta estructura proporciona una clara separación de responsabilidades, mejora la mantenibilidad y escalabilidad del sistema.

5.1.1. Descripción de la Arquitectura

El diseño híbrido de esta herramienta combina lo mejor de ambos paradigmas para crear una aplicación robusta y extensible que facilita la gestión de exámenes y la corrección automática. A continuación, se describirá cada uno de ellos, lo que nos permitirá identificar que el patrón MVC está implementado en el servidor.

5.1.1.1. Modelo-Vista-Controlador (MVC):

- **Modelos:** Definidos con SQLAlchemy, representan las entidades y la estructura de la base de datos. Son responsables de las operaciones de persistencia de datos.

- **Vistas:** Implementadas con plantillas Jinja, generan dinámicamente el contenido HTML que se muestra a los usuarios. Minimizan la lógica de presentación para centrarse en la representación visual.
- **Controladores:** Manejados por Flask, definen y gestionan las rutas de la aplicación, controlan la lógica de negocio y facilitan la interacción entre las vistas y los modelos.

5.1.1.2. Cliente-Servidor:

- **Cliente:** Interactúa con el sistema a través de una interfaz web que utiliza HTML, CSS y JavaScript para la presentación y la interactividad en el navegador.
- **Servidor:** Basado en Flask, procesa las solicitudes HTTP enviadas desde el cliente, ejecuta la lógica de negocio, interactúa con la base de datos MariaDB mediante SQLAlchemy y devuelve las respuestas al cliente.

5.1.2. Diagrama de Arquitectura del Sistema

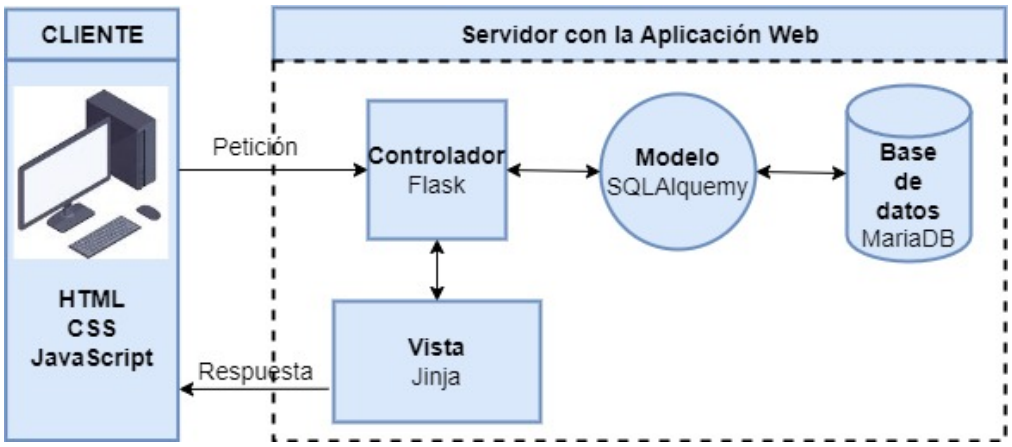


Ilustración 5.1: Diagrama de Arquitectura del Sistema

5.2. Diagrama de Clases

Las clases definen la lógica y propiedades de los objetos en la aplicación. Además, mediante el uso de un mapeador objeto-relacional (ORM) como SQLAlchemy, estas clases también se mapean directamente a las tablas de la base de datos, facilitando la manipulación de datos de manera coherente y eficiente.

Las clases principales en nuestro sistema incluyen ‘User‘, ‘Exam‘, ‘Question‘, ‘Answer‘, ‘Student‘, ‘ExamStudent‘, y ‘Model‘. Estas clases se interrelacionan para proporcionar una estructura robusta para la gestión y corrección automatizada de exámenes.

El siguiente diagrama de clases proporciona una vista detallada de las entidades del sistema, sus atributos y métodos, así como las relaciones entre ellas.

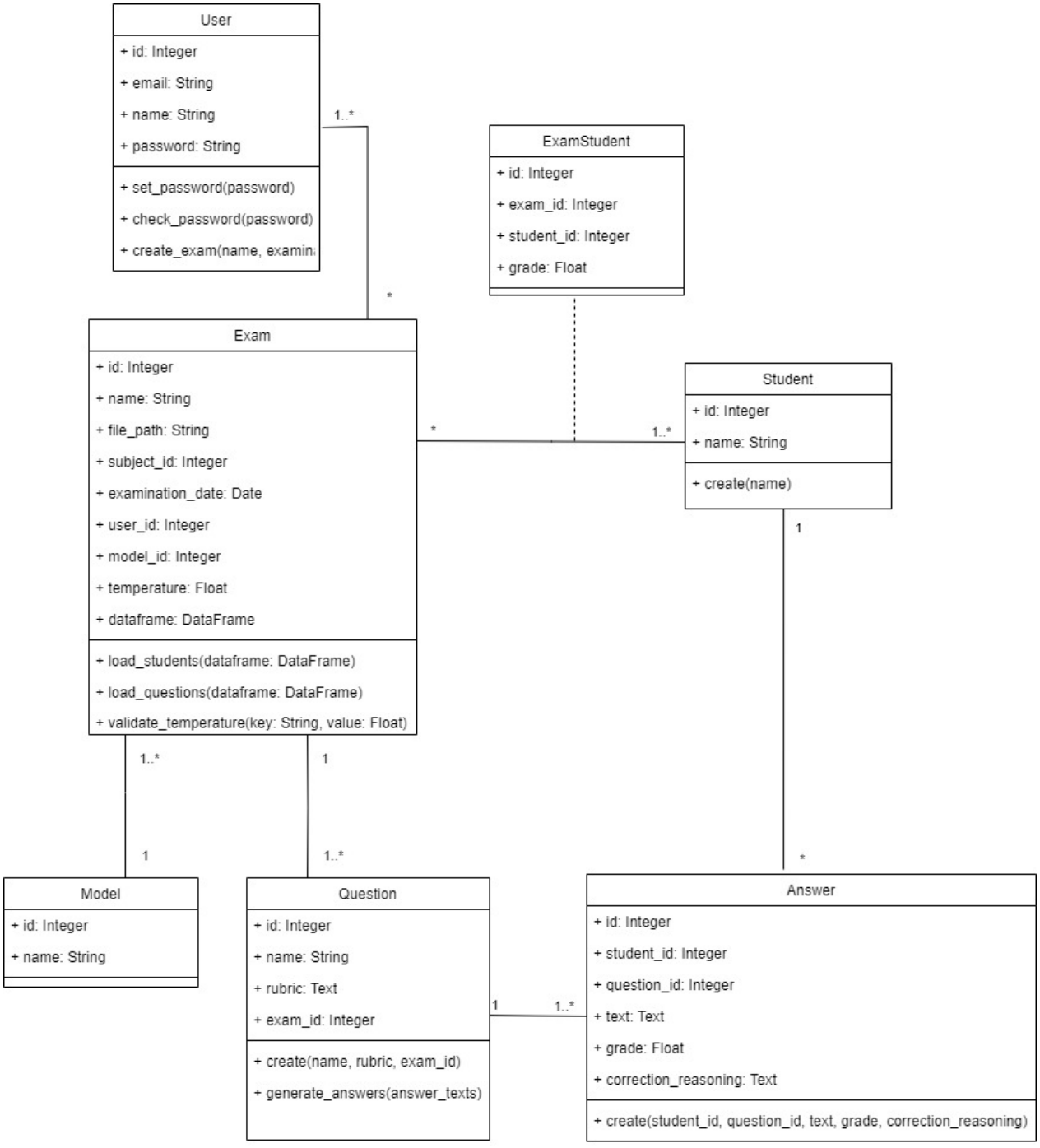


Ilustración 5.2: Diagrama de Clases

5.3. Modelado de Datos

El modelado de datos **Corrector Web** se realiza mediante el uso de SQLAlchemy, recordemos que es un mapeador objeto-relacional (ORM) para Python. SQLAlchemy actúa como un puente entre el diseño orientado a objetos y el modelo relacional de la base de datos. Cada clase en nuestro sistema está directamente vinculada a una tabla en la base de datos, y los atributos de estas clases se corresponden a las columnas de las tablas.

5.3.1. Uso del ORM (SQLAlchemy)

SQLAlchemy proporciona una interfaz de alto nivel para definir, manipular y consultar tablas en la base de datos.

5.3.1.1. Definición de Clases y Tablas

- Las clases ORM definen las tablas en la base de datos.
- Los atributos de clase representan las columnas de las tablas.
- Las instancias de clase corresponden a filas individuales en las tablas.

5.3.1.2. Relaciones entre Entidades

- Las relaciones entre clases en el código se mapean a relaciones entre tablas en la base de datos, tales como uno a muchos, muchos a muchos, y uno a uno.
- SQLAlchemy asegura la integridad referencial y facilita consultas complejas a través de definiciones de relaciones.

5.3.1.3. Persistencia y Manipulación de Datos

- Las operaciones CRUD (Crear, Leer, Actualizar, Eliminar) se simplifican mediante métodos en las clases ORM.
- Las transacciones se manejan automáticamente para asegurar las propiedades ACID (Atomicidad, Consistencia, Aislamiento y Durabilidad).

5.3.2. Modelo Entidad-Relación (ER)

El Modelo Entidad Relación es una representación visual que describe cómo los datos están estructurados y relacionados entre sí en una base de datos. Debido a que esta aplicación trabaja con un **ORM** el diagrama **ER** es muy similar al diagrama de clases.

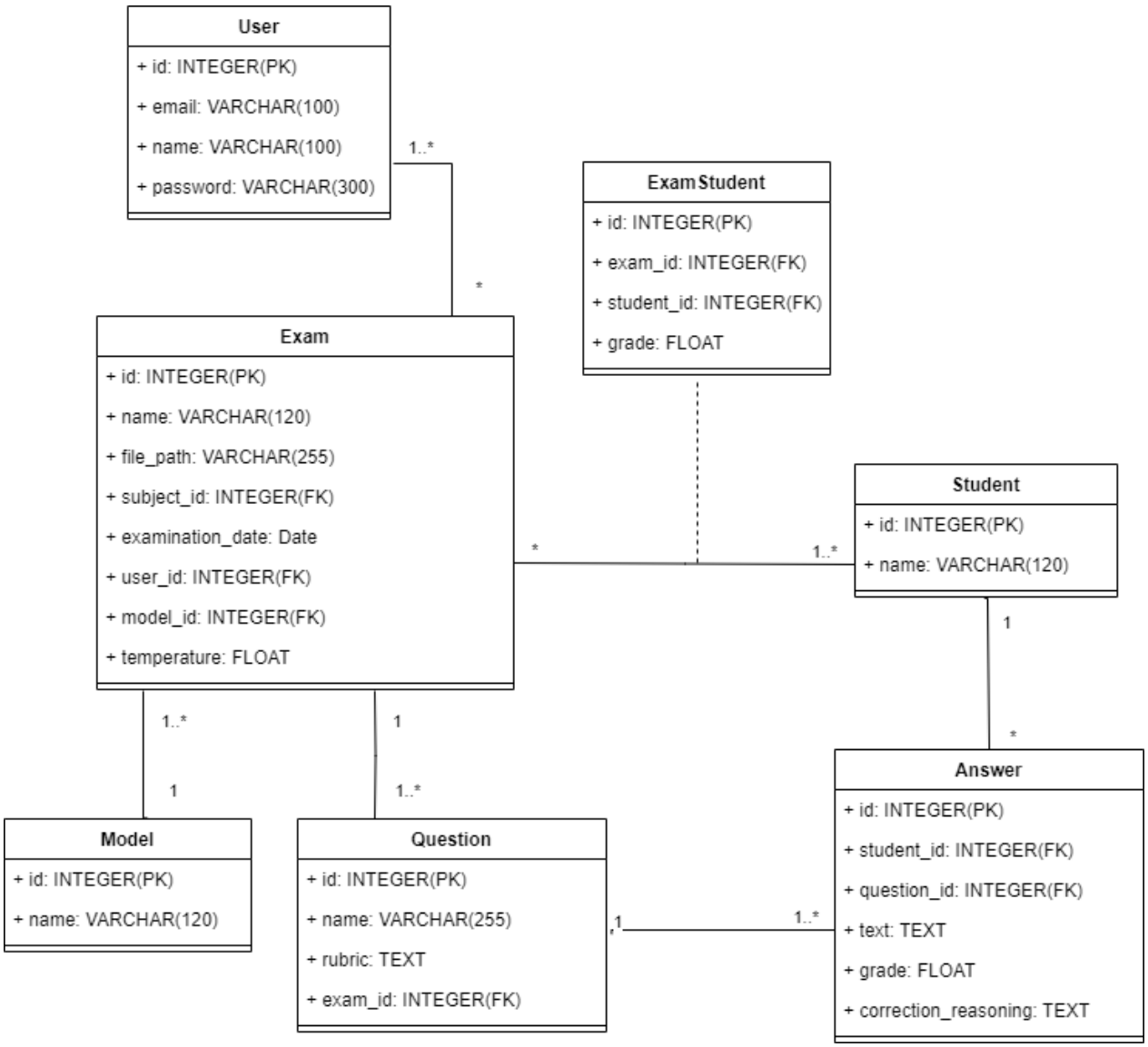


Ilustración 5.3: Diagrama Entidad-Relación (ER)

Debido al uso de Object-Relational Mapping (ORM), es de vital importancia mostrar cómo se estructuran las clases y cómo se mapean a la base de datos, para ello se mostrará un ejemplo.

```python
from flask_sqlalchemy import SQLAlchemy
db = SQLAlchemy(app)

class Exam(db.Model):
    __tablename__ = 'exam'

    id = db.Column(db.Integer, primary_key=True)
    name = db.Column(db.String(120), nullable=False)
    file_path = db.Column(db.String(255))
    subject_id = db.Column(db.Integer, db.ForeignKey('subject.id'))
    examination_date = db.Column(db.Date)
    user_id = db.Column(db.Integer, db.ForeignKey('user.id'), nullable=False)
    user = db.relationship('User', back_populates='exams')
    questions = db.relationship('Question', back_populates='exam', cascade='all, delete-orphan')
    exam_students = db.relationship('ExamStudent', back_populates='exam', cascade='all, delete-orphan')
    model_id = db.Column(db.Integer, db.ForeignKey('model.id'), nullable=False)
    model = db.relationship('Model', backref=db.backref('exams', lazy=True))
    temperature = db.Column(db.Float, nullable=False, default=0.7)
```

Ilustración 5.4: Clase Exam

5.4. Diseño de la Interfaz de Usuario (UI)

La interfaz de usuario es un componente esencial del sistema, ya que es el medio a través del cual los usuarios interactúan con la aplicación. El objetivo principal durante el diseño de la UI fue garantizar que ésta sea intuitiva, amigable y eficiente, permitiendo a los profesores (usuarios) completar sus tareas de manera rápida y sencilla.

CorrectorWeb

Ilustración 5.5: Logo de Correctror Web.

5.4.1. Principios de Diseño de UI

Para el diseño de la interfaz de usuario de **Correctror Web**, se utilizó Material Design. Esta elección adopta una serie de principios fundamentales que guían la creación y evolución de la **UI**, asegurando así una experiencia de usuario coherente, intuitiva y eficiente.

5.4.1.1. Consistencia

- **Visual**: Se han utilizado colores, tipografías y estilos de elementos visuales de manera coherente en toda la aplicación, proporcionando así una experiencia uniforme. En este asistente, se ha elegido el azul como color primario para mantener la coherencia visual y facilitar la navegación.
- **Funcional**: Se han mantenido comportamientos y respuestas uniformes para acciones similares, de modo que los usuarios desarrollen una comprensión intuitiva del sistema.

5.4.1.2. Claridad

- **Simplicidad**: Las interfaces han sido diseñadas para ser limpias y minimalistas, donde cada elemento tiene un propósito claro y relevante. Esto reduce el desorden visual y ayuda a los usuarios a concentrarse en sus tareas principales sin distracciones innecesarias.
- **Legibilidad**: Se emplearon tipografías y tamaños de texto fáciles de leer, asegurando contrastes de color adecuados para una mejor visibilidad. Estas elecciones se realizaron para garantizar una legibilidad óptima.

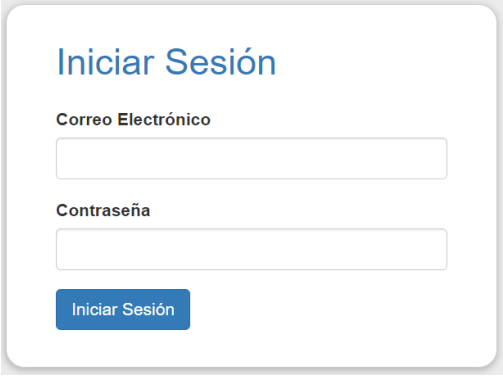


Ilustración 5.6: Ejemplo de Claridad en Corrector Web

5.4.1.3. Feedback Inmediato

- **Visual**: La interfaz proporciona retroalimentación clara e inmediata a las acciones del usuario a través de cambios visuales. Por ejemplo, botones que cambian de color al hacer clic.

Ilustración 5.7: Ejemplo de Botones que cambian de color al interactuar con ellos

- **Mensajes de estado**: Se Informa al usuario sobre el estado de sus acciones, ya sean éxitos, errores o procesos en curso, mediante notificaciones claras y concisas.

Ilustración 5.8: Ejemplo de Feedback Inmediato en Corrector Web

5.4.1.4. Eficiencia

- **Optimización de tareas**: El diseño se enfocó en minimizar el número de pasos necesarios para completar una tarea, optimizando así la eficiencia del usuario. Además, se priorizaron las funciones y accesos más utilizados para que los usuarios puedan realizar sus actividades de manera rápida y sin perder tiempo en procesos innecesarios. Un ejemplo claro de esto se encuentra en las interfaces de **Corrector Web**, donde se muestran de manera clara y accesible todos los botones de acción relevantes.

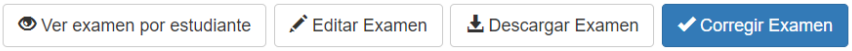


Ilustración 5.9: Ejemplo de Eficiencia en Corrector Web

5.4.2. Accesibilidad

- **Contraste de Color**: Para facilitar la lectura y la comprensión del contenido a usuarios con deficiencias visuales, se ha asegurado que haya suficiente contraste entre el texto y el fondo. Para ello se ha empleado las Directrices de Accesibilidad para el Contenido Web (WCAG)[23] proporcionan estándares de contraste para garantizar que el contenido sea accesible. Según las WCAG, el nivel AA requiere una relación de contraste mínima de 4.5:1 para texto normal y 3:1 para texto grande, mientras que el nivel AAA requiere una relación de contraste mínima de 7:1 para texto normal y 4.5:1 para texto grande. Cumplir con estos niveles asegura que el texto sea legible para personas con baja visión o daltonismo.

En **Corrector Web**, se han seleccionado cuidadosamente colores que ofrecen un alto contraste, alcanzando el nivel AAA; por ejemplo, los textos principales y los botones están diseñados en colores oscuros sobre fondos claros o viceversa. Esto asegura que todos los usuarios puedan leer y comprender fácilmente la información presentada, independientemente de sus capacidades visuales.

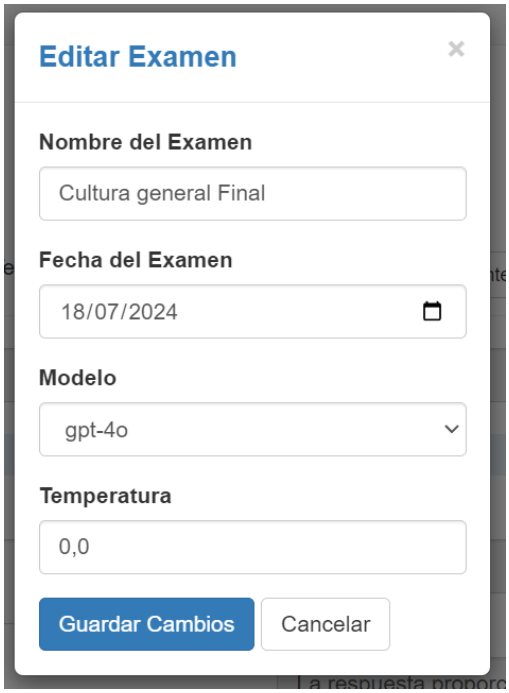


Ilustración 5.10: Ejemplo de Contraste de Color en Corrector Web

5.4.2.1. Control y Libertad del Usuario

- **Deshacer y rehacer**: La UI genera senzación de control al usuario, proporcionando opciones para deshacer algún error y recuperar el control sin penalizaciones severas. Por ejemplo, la edición de exámenes.
- **Navegación**: Se ha diseñado una estructura de navegación que permita al usuario moverse libremente y fácilmente entre las diferentes secciones del sistema.

5.4.2.2. Familiaridad

- **Uso de convenciones**: El uso de patrones y convenciones de diseño familiares para los usuarios, basados en sus experiencias con otras aplicaciones, es fundamental, ya que facilita la rápida adaptación de los usuarios. En este proyecto, se han implementado

estos principios, reduciendo así la curva de aprendizaje y mejorando la experiencia del usuario desde el primer uso.

- **Iconografía universal**: Se han empleado iconos y señales visuales comunes que son fácilmente reconocibles, contribuyendo de esta manera a la reducción de la curva de aprendizaje

Ilustración 5.11: Ejemplo de Iconografía Universal en Corrector Web

5.4.2.3. Prevención de Errores

- **Validación**: se han implementado validaciones en los formularios y entradas de datos para evitar errores antes de que ocurran.
- **Mensajes preventivos**: Se muestran mensajes de advertencias y confirmar acciones destructivas para prevenir errores operacionales.

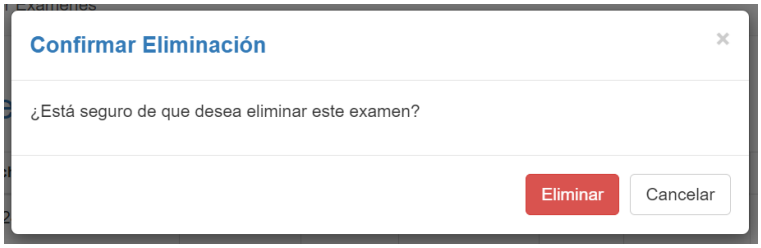


Ilustración 5.12: Ejemplo de Mensajes preventivos en Corrector Web

5.4.3. Prototipos de UI

En este apartado se presentan los prototipos de la interfaz de usuario, destacando las principales pantallas de **Corrector Web**. Estos prototipos ofrecen una visión clara y detallada del diseño y la estructura, permitiendo una comprensión visual de cómo los usuarios interactuarán con la aplicación.

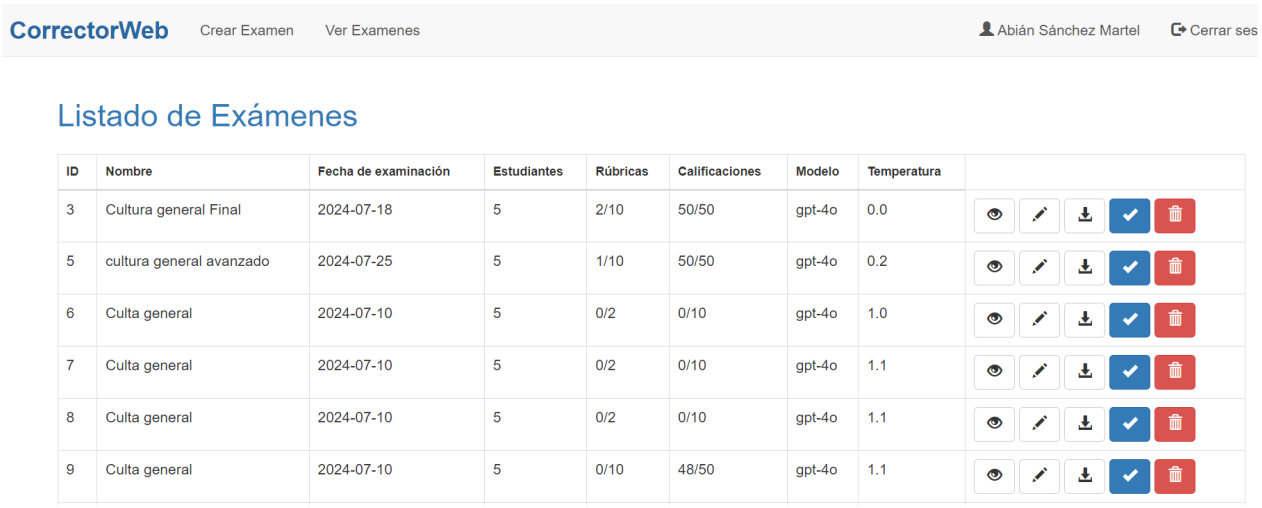


Ilustración 5.13: Pantalla - Ver Examenes

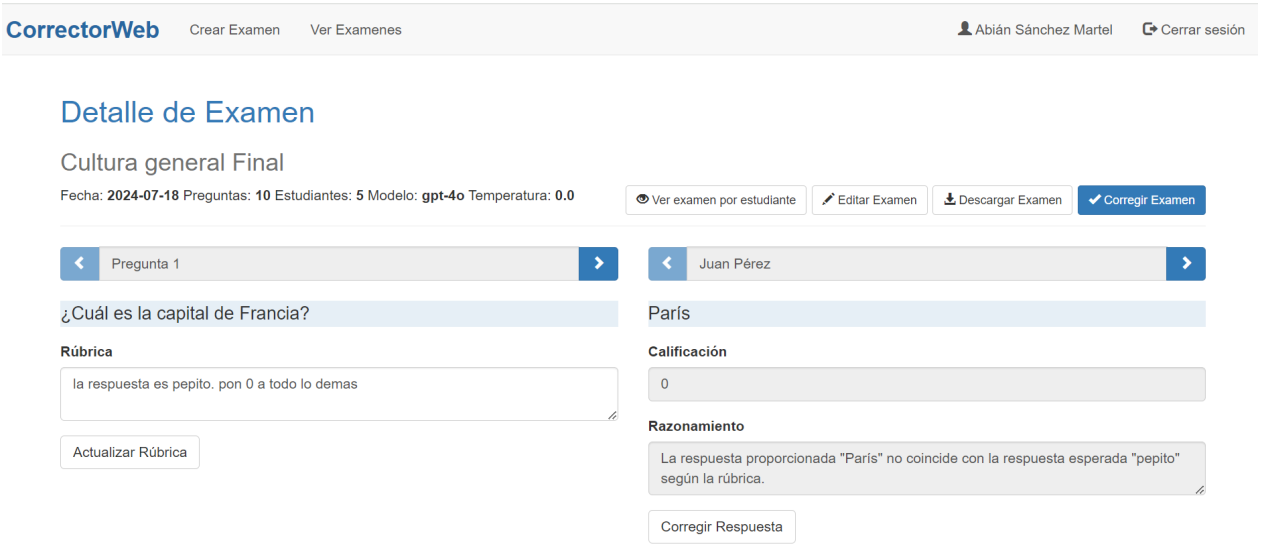


Ilustración 5.14: Pantalla - Ver Examen

Capítulo 6

Desarrollo

En este capítulo, se abordará la metodología de trabajo empleada en el desarrollo de la aplicación. Asimismo, se describirán las distintas etapas del proceso. Esta sección proporcionará una visión integral del enfoque utilizado y de los pasos realizados para garantizar el éxito del desarrollo.

6.1. Metodología de Desarrollo

Para la realización de esta aplicación, se ha seguido una metodología iterativa e incremental. Este enfoque permitió adaptar y mejorar la aplicación de manera continua, incorporando retroalimentación y ajustando los requisitos según las necesidades detectadas a lo largo del proceso. En cada iteración, se añadieron nuevas funcionalidades y se optimizaron las existentes, lo que facilitó la gestión de cambios, priorizando las tareas y logrando resultados rápidos.

De este modo, se pudieron resolver eficazmente los inconvenientes que surgieron durante el desarrollo. Además, esta metodología promovió una mayor flexibilidad y capacidad de respuesta a las demandas del proyecto.

6.2. Primeros pasos

Partiendo de la base de que nunca había creado una página web con Flask, la primera semana del proyecto se reservó para el aprendizaje y familiaridad de las herramientas prin-

cipales a utilizar, para ello se acudió a la documentación oficial de Flask[11] y así poder ver algún ejemplo de implementación. Para la parte de Python se aprovecho el tutorial aportado en la página de educación w3schools.com[24].

En esta fase se creó aplicaciones sencillas y básicas de Flask/Python para conseguir soltura con los mismos.

6.2.1. Configuración del entorno de desarrollo

Para la codificación, se optó por el IDE **Visual Studio Code**. Desde el mismo, se abrió una terminal para crear el módulo del entorno virtual donde se trabajará a partir de ahora para gestionar las dependencias del proyecto. Para ello, se ejecutó los siguiente comandos:

```
CorrectorWeb> py -3 -m venv .venv
```

Ilustración 6.1: Creación entorno virtual

Este comando se utiliza para crear un entorno virtual en Python 3 para una aplicación Flask. El entorno virtual, que se crea en el directorio ".venv", aísla las dependencias del proyecto del resto del sistema, asegurando que las bibliotecas y paquetes necesarios para la aplicación Flask se gestionen de manera independiente.

```
CorrectorWeb> .\.venv\Scripts\activate
```

Ilustración 6.2: Activación entorno virtual

Una vez activado el entorno virtual, cualquier instalación de paquetes o ejecución de scripts utilizará el entorno virtual, asegurando que las dependencias del proyecto no interfieran con otras instalaciones de Python en tu sistema.

Esto fue de gran ayuda, ya que para el desarrollo del proyecto se trabajó desde casa con un PC de sobremesa y también en un portátil de la biblioteca. Al no tener que preocuparse por las posibles instalaciones de diferentes versiones de Python y librerías, se pudo continuar el trabajo sin interrupciones. Esta configuración también permitió tener las reuniones de seguimiento con el tutor sin inconvenientes.

Para iniciar Flask, se optó por seguir las recomendaciones de la documentación oficial y se evitó crear una instancia global de Flask, ya que esto podría complicar el código a medida que creciera y se volviera más complejo. La solución adoptada fue crear la instancia de Flask dentro de una función conocida como "Factory Method" o "Fábrica de Aplicaciones".

Esta función encapsula los archivos y el código en un paquete llamado **flaskr**. En el fichero `__init__.py` de este paquete, se especifican los ajustes y configuraciones de la aplicación, que será devuelta. Este enfoque no solo organiza mejor el código, sino que también facilita la configuración de la aplicación para diferentes entornos (desarrollo, pruebas, producción).

En resumen, evitar una instancia global de Flask y optar por un patrón de fábrica aumenta la flexibilidad, facilita las pruebas, mejora el manejo de configuraciones y asegura una mayor modularidad y escalabilidad en el desarrollo de aplicaciones Flask. Este método asegura que cada instancia de la aplicación sea independiente, facilitando así las pruebas y la escalabilidad.

```python
import os

from flask import Flask
from flask_bootstrap import Bootstrap
from flask_sqlalchemy import SQLAlchemy

db = SQLAlchemy()

def create_app(test_config=None):
    # create and configure the app
    app = Flask(__name__, instance_relative_config=True)

    Bootstrap(app)

    app.config['SQLALCHEMY_DATABASE_URI'] = 'mysql+pymysql://root:root@127.0.0.1:3306/CorrectorWeb'
    app.config['SQLALCHEMY_TRACK_MODIFICATIONS'] = False
    db.init_app(app)

    from . import auth
    app.register_blueprint(auth.bp)

    from . import exam
    app.register_blueprint(exam.bp)
    app.config['UPLOAD_FOLDER'] = 'uploads'
    app.add_url_rule('/', endpoint='index')

    from . import populate
    with app.app_context():
        db.create_all()
        populate.populate_models()

    return app
```

Ilustración 6.3: Fichero de configuración flask

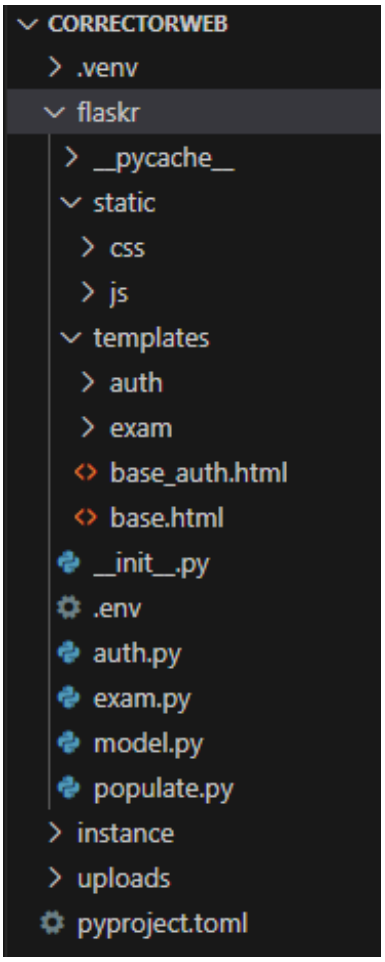


Ilustración 6.4: Paquete flaskr

6.3. Gestión de usuarios

La primera tarea a desarrollar fue la de crear las funcionalidades para permitir registrar usuarios y por consiguiente habilitar las opciones de Login/Logout en el sistema. En un principio se optó por utilizar **SQLite** como base de datos porque ya viene incorporado a Python en el modulo **sqlite3** y no hace falta ninguna configuración del servidor de la base de datos por separado, de esta forma se podía avanzar sin demasiada complejidad.

Flask permite agrupar las vistas relacionadas mediante **Blueprints** y estas a su vez se registran una vez en flask (`__init__.py`). Para la agrupación de vistas de autentificación, se creó las plantillas en el directorio **"/templates/auth"**, siendo estas **"login.html"** y **"register.html"**.

La libreria Jinja2 viene incluida en Flask, y sería la encargada del renderizado de las plantillas, y es especialmente útil por las siguientes características:

1. **Autoescapado**: El concepto se refiere a la función de seguridad que automáticamente escapa caracteres especiales en las plantillas para prevenir ataques de inyección de código, como XSS (cross-site scripting). Esto asegura que el contenido renderizado sea seguro y no ejecute código malicioso.

2. **Sintaxis similar a Python**: Particularmente apropiado para facilitar el aprendizaje y la lógica de control, como bucles y condicionales, se escribe de manera similar a Python, pero utilizando una sintaxis específica de plantillas.

3. **Extensiones de plantillas**: Jinja2 permite la creación de plantillas base y la extensión de estas mediante bloques. Las plantillas hijas pueden definir o sobrescribir bloques de contenido especificados en la plantilla base, permitiendo una estructura modular y reutilizable. Este concepto facilitará el mantenimiento y la organización de la aplicación, ya que permite reutilizar estructuras comunes en múltiples vistas.

Por ello, se crea las plantillas `"base_auth.html"`, que sería la plantilla base para las vistas de Login y Registro de usuarios, y **"base.html"**, se encargaría de la vista común para toda la aplicación.

A continuación se muestra la primera aproximación de la interfaz.

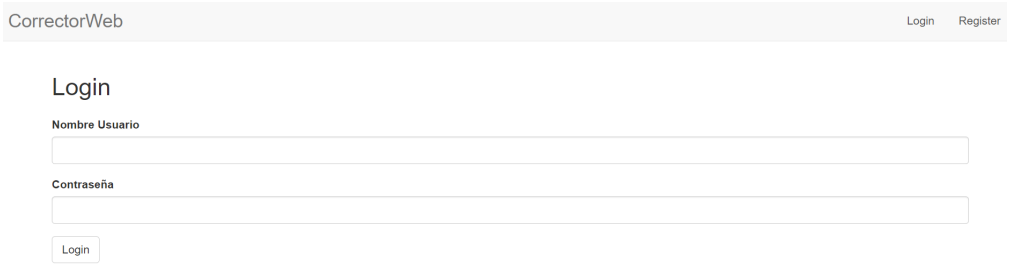


Ilustración 6.5: Plantilla base extendida con plantilla Login

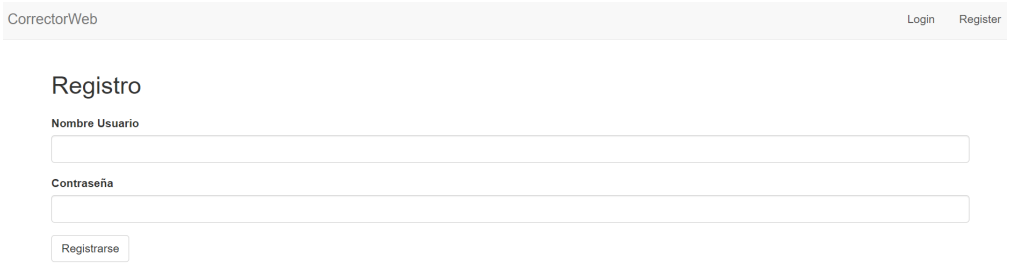


Ilustración 6.6: Plantilla base extendida con plantilla Register

6.3.1. Control

En el apartado de control, se engloban todas las funciones propias de la gestión de usuarios en un mismo fichero, **"auth.py"**, encargado de comprobar la correcta interacción con el usuario, para ello se realizan las siguientes controles:

6.3.1.1. Login

1. Tanto el campo **"Nombre Usuario"** como el de **"Contraseña"** no pueden quedar vacíos.
2. El Nombre de Usuario es incorrecto (no se encuentra en la Base de datos).
3. La Contraseña es incorrecta (la contraseña asociada al Nombre de Usuario no coincide).

6.3.1.2. Registro

1. Los campos **"Nombre Usuario"** y **"Contraseña"** deben estar cumplimentados.
2. El Nombre de Usuario se encuentra en la Base de datos (Usuario ya registrado).

6.3.2. Pruebas Realizadas

Una vez implementadas estas funcionalidades, se llevaron a cabo diversas pruebas para asegurar su correcto funcionamiento. Para el registro de usuarios, se verificó que todos los campos necesarios (nombre de usuario y contraseña) fueran validados correctamente, que no se permitiera registrar usuarios con nombres de usuario duplicados y que se recibiera un mensaje de confirmación tras un registro exitoso.

En cuanto al login, se realizaron pruebas para asegurar que los usuarios pudieran autenticarse correctamente utilizando sus credenciales. Se probaron escenarios de login exitoso y fallido, verificando que el sistema respondiera adecuadamente a intentos con contraseñas incorrectas o nombres de usuario inexistentes. También se probó el proceso de logout para confirmar que los usuarios se desloguearan correctamente y que las sesiones fueran gestionadas de manera segura.

6.4. Manejo de Exámenes

El manejo de exámenes es un componente crucial del asistente de corrección de exámenes. En esta sección se describen las actividades relacionadas con la definición de la estructura de los exámenes y la visualización de las respuestas de los estudiantes.

6.4.1. Definición de la Estructura de los Archivos

Uno de los pasos más cruciales en el desarrollo del asistente de corrección de exámenes fue la definición de la estructura de los archivos que se subirían a la aplicación. Estos archivos contienen toda la información relevante sobre los exámenes, incluyendo las preguntas y las respuestas de los estudiantes.

La estructura de los archivo XLS incluye las siguientes columnas:

- **Nombre Estudiante**: El nombre y apellido del estudiante.

- **Preguntas**: Cada columna subsecuente representa una pregunta del examen. Estas columnas contienen las respuestas dadas por los estudiantes.

	A	B	C	D	E
1		Pregunta 1	Pregunta 2	Pregunta 3	Pregunta 4
2	Alumno 1	Respuesta	Respuesta	Respuesta	Respuesta
3	Alumno 2	Respuesta	Respuesta	Respuesta	Respuesta
4	Alumno 3	Respuesta	Respuesta	Respuesta	Respuesta
5	Alumno 4	Respuesta	Respuesta	Respuesta	Respuesta

Ilustración 6.7: Estructura de los archivos

6.4.2. Visualización de Exámenes y Respuestas

Una vez definida la estructura de los archivos XLS, se procedió a desarrollar la funcionalidad para la carga de exámenes y la visualización de la información contenida en estos archivos. Esta etapa fue esencial para permitir a los profesores revisar las respuestas de los estudiantes de manera clara y organizada.

6.4.2.1. Carga de Exámenes

Se desarrolló una interfaz de carga de archivos que permite a los profesores subir los archivos XLS que contienen los exámenes. Para la gestión de estos archivos, se utilizó la librería Pandas, la cual facilita la manipulación y lectura de datos en formato XLS. Además, se realizó un control de errores para asegurar que el formato del archivo fuese correcto, que se subiera un archivo y que este no estuviera vacío.

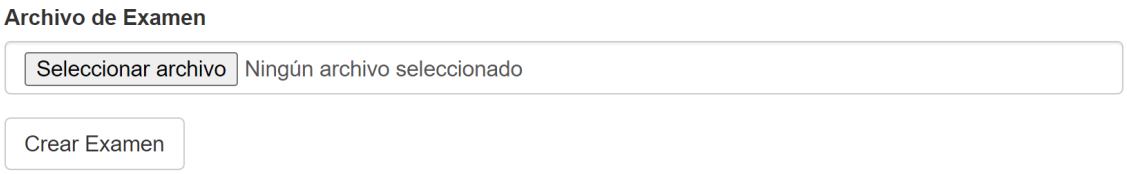


Ilustración 6.8: Subir archivo - Primera Interfaz de usuario

6.4.2.2. Vista de Exámenes

Una vez cargado un archivo XLS, se implementó una funcionalidad que muestra una pregunta del examen y la respuesta de un estudiante a la vez.

6.4.2.3. Navegación por Estudiantes y Preguntas

Se diseñó una interfaz que permite a los profesores navegar fácilmente entre las preguntas del examen y los estudiantes(respuestas). A continuación, se muestra la primera interfaz para la vista de exámenes.

6.4.2.4. Pruebas

Se realizaron diversas pruebas para asegurar que todas las funcionalidades trabajaran correctamente. Se comprobó que:

- Los archivos XLS fueran cargados correctamente.
- El formato de los archivos fuera el adecuado.
- Se evitara la carga de archivos vacíos.
- Las preguntas y respuestas fueran visualizadas correctamente.
- La navegación entre estudiantes y preguntas fuera fluida.

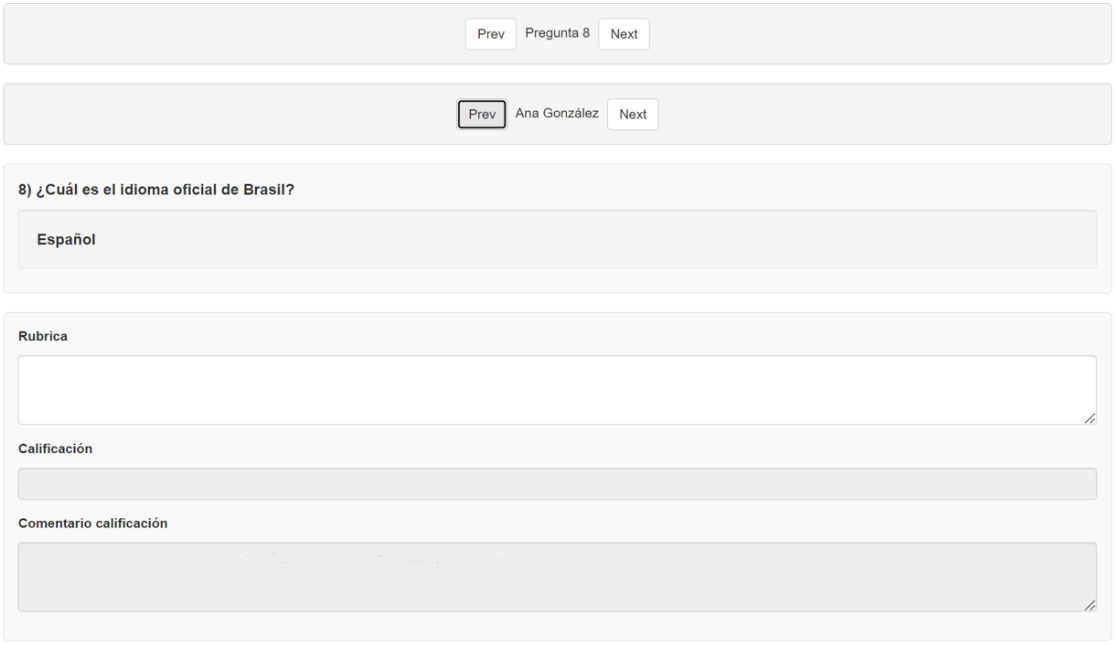


Ilustración 6.9: Ver Examen - Primera Interfaz de usuario

- Durante la navegación, no se intentara acceder a un elemento que no existiera, evitando así posibles errores.

6.5. Integración de MariaDB

El siguiente paso en el desarrollo del proyecto fue la integración de **MariaDB** como base de datos principal. Esta integración proporciona un entorno más robusto y escalable, mejorando significativamente la capacidad de manejo de datos, la seguridad y el rendimiento de la aplicación.

6.5.1. Instalación y configuración

1. Se procedió a descargar **MariaDB** desde la página web oficial e instalarla en el sistema.
2. Para la gestión de las tablas y los atributos de la base de datos se usó la herramienta **HeidiSQL**[25] debido a su compatibilidad con **MySQL** y sus derivados como es el caso de **MariaDB**.
3. Se agregó la librería de MariaDB en el proyecto ejecutando el siguiente comando en la terminal de **Visual Studio**

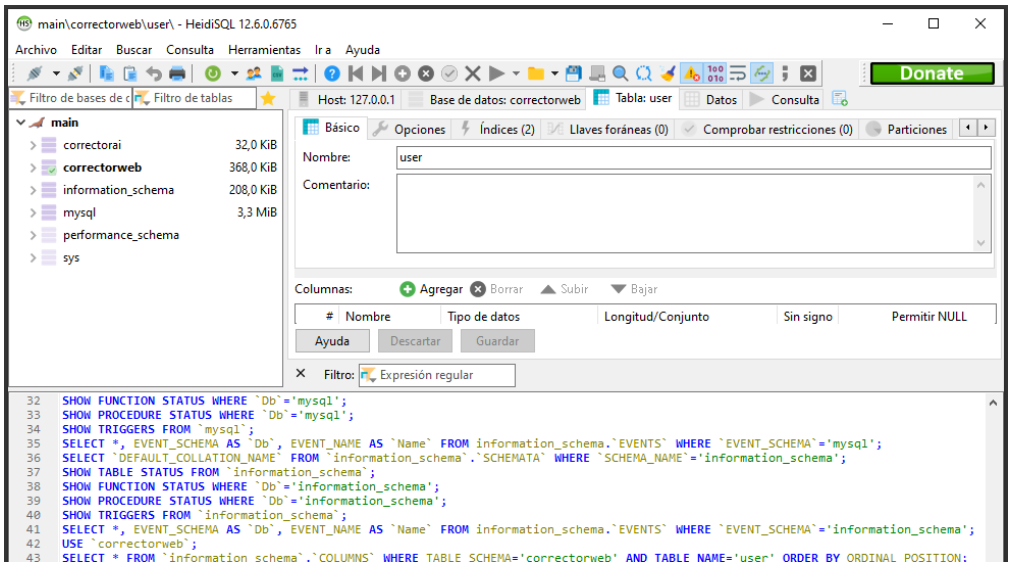


Ilustración 6.10: HeidiSQL - Herramienta de gestión de BD con SQL

```
CorrectorWeb> pip install mariadb
```

Ilustración 6.11: Instalación de MariaDB

Una vez instalada y agregada al proyecto la base de datos, lo primero que se hizo fue una prueba de conexión para asegurarnos de que todo funcionara correctamente.

6.5.2. Gestión de Usuarios en MariaDB

Durante este proceso, se realizaron los siguientes pasos:

- **Creación de la tabla de usuarios**: Lo primero que se realizó fue crear la tabla de usuarios con sus respectivos campos.
- **Modificación del Módulo de Autenticación**: Se ajustó el módulo de autenticación para interactuar con MariaDB en lugar de SQLite. Esto incluyó la conexión a la nueva base de datos y la actualización de las consultas SQL correspondientes.

Durante la integración de MariaDB, se realizaron diversas pruebas para asegurar la correcta migración y funcionamiento del sistema:

- **Pruebas de Conectividad**: Se verificó que la aplicación pudiera conectarse a MariaDB sin problemas y que las consultas fueran ejecutadas correctamente.
- **Pruebas de Integridad de Datos**: Se comprobó que todas las transacciones y operaciones de manejo de datos se completaran sin errores, asegurando que no hubiera

pérdida o corrupción de datos.

- **Almacenamiento Seguro de Contraseñas**: Durante este proceso, se identificó que las contraseñas de los usuarios se almacenaban en texto plano. Para solucionar este problema, se implementó el hashing de contraseñas utilizando las funciones **check_password_hash** y **generate_password_hash** de la librería Werkzeug.
- **Pruebas de Seguridad**: Se realizaron pruebas exhaustivas para asegurar que las contraseñas hasheadas funcionaran correctamente durante el proceso de login y que no hubiera vulnerabilidades de seguridad.

6.6. Estructuración de la información

En esta etapa del desarrollo de **Corrector Web**, se decide dotar de estructura a la información manejada por la aplicación. Los elementos clave fueron: preguntas, estudiantes, respuestas, rúbricas, calificación y justificación de la calificación.

6.6.1. Estructuración de Datos

Para estructurar eficientemente estos datos y gestionar la lógica de la aplicación, considerando que la persistencia de datos se implementaría más adelante, se optó por utilizar clases.

Se utilizó una estructura básica de clases en Python, como se puede apreciar en el capítulo de diseño, en el apartado del modelo de clases. Cada una de estas clases encapsula los atributos y comportamientos necesarios para gestionar la información relacionada con los exámenes y su corrección. Esta estructura sentó las bases para el desarrollo posterior del conjunto de tablas y campos que se utilizarán en la base de datos.

Posteriormente, se refactorizaron las funciones de la aplicación para que operaran con esta nueva arquitectura de datos. La legibilidad del código y su escalabilidad se beneficiaron significativamente al adoptar un enfoque más orientado a la programación orientada a objetos.

A partir de esta nueva estructura surge la creación de exámenes, que posee nombre, fecha y el archivo XLS, con el que se inicializarán todas las clases necesarias para el correcto funcionamiento de la aplicación.

También se permite la edición y visualización de rúbricas.

6.6.2. Pruebas y mejoras

Dado que se cambió por completo la organización del sistema, fue necesario comprobar el correcto funcionamiento del sistema casi en su totalidad, exceptuando la autenticación. Se realizaron pruebas exhaustivas para garantizar que las modificaciones no introdujeran errores.

6.7. Integración de la API de OpenAI

Para integrar la API de OpenAI en **Corrector Web**, es fundamental establecer un sistema de comunicación eficaz mediante solicitudes HTTP. Este proceso requiere algunos pasos previos importantes:

6.7.0.1. Configuración Inicial

Para poder realizar solicitudes a la API de OpenAI, es necesario disponer de una cuenta en la plataforma. Una vez registrada la cuenta, se debe generar una clave API que será utilizada en cada una de las peticiones a la interfaz. Esta clave es un identificador único que autentica cada solicitud y asegura que las interacciones con la API estén autorizadas.

6.7.0.2. Realización de Solicitudes

Cada vez que se realiza una llamada a la API de OpenAI, se utiliza la clave API generada para autenticar la petición. Es importante destacar que cada llamada a la API implica un coste para la cuenta asociada a la clave.

Al establecer la comunicación con la interfaz de OpenAI, se deben especificar el modelo que se desea utilizar, ya que esto influye tanto en el coste como en la calidad de la respuesta. Además, se incluyen dos mensajes en la llamada: uno que define el comportamiento de la IA (en este caso, como asistente de calificación), y otro que contiene la pregunta, la rúbrica y la respuesta que se desea corregir.

```python
client = OpenAI(api_key=openai_api_key)

def grade_answer(q_data):
    messages = [
        {
            "role": "system",
            "content": "Eres un asistente de calificación. " \
                       "Vas a calificar respuestas sobre 10 basándote en la rúbrica proporcionada. " \
                       "Devuelves las respuestas en formato 'Calificación: X.XX\n Razonamiento: ...'"
        },
        {
            "role": "user",
            "content": (
                f"Pregunta: {q_data['question']}\n"
                f"Respuesta: {q_data['answer_text']}\n"
                f"Rúbrica: {q_data['rubric']}"
            )
        }
    ]

    chat_completion = client.chat.completions.create(
        messages=messages,
        model="gpt-4o",
    )
```

Ilustración 6.12: Llamada a la interfaz de OpenAI

Se implementa la funcionalidad de corrección de preguntas en el sistema. Este proceso puede realizarse con o sin el uso de rúbricas de evaluación, dependiendo de las necesidades específicas del examen y de las preferencias del profesor. Además, el sistema genera un razonamiento explicativo que acompaña a cada calificación.

Ilustración 6.13: Corrección de Preguntas

6.7.1. Comprobación y Mejoras

Durante el proceso de integración de ChatGPT en el sistema, se realizaron pruebas detalladas utilizando una variedad de preguntas, tanto de respuesta simple como de desarrollo. En estas pruebas se incluyeron preguntas con y sin rúbrica de calificación. En general, ChatGPT mostró un rendimiento adecuado y consistente en la mayoría de los casos, gestionando tanto las respuestas como las evaluaciones según los criterios esperados.

Sin embargo, se identificaron algunas particularidades en el comportamiento del sistema. En situaciones donde dos estudiantes proporcionaron respuestas incorrectas identicas a una misma pregunta, las calificaciones asignadas por ChatGPT fueron diferentes. Además, se observaron diferencias notables entre el modelo Turbo 3 y el modelo Turbo 4, con el último ofreciendo mejoras en la calidad y coherencia de las respuestas.

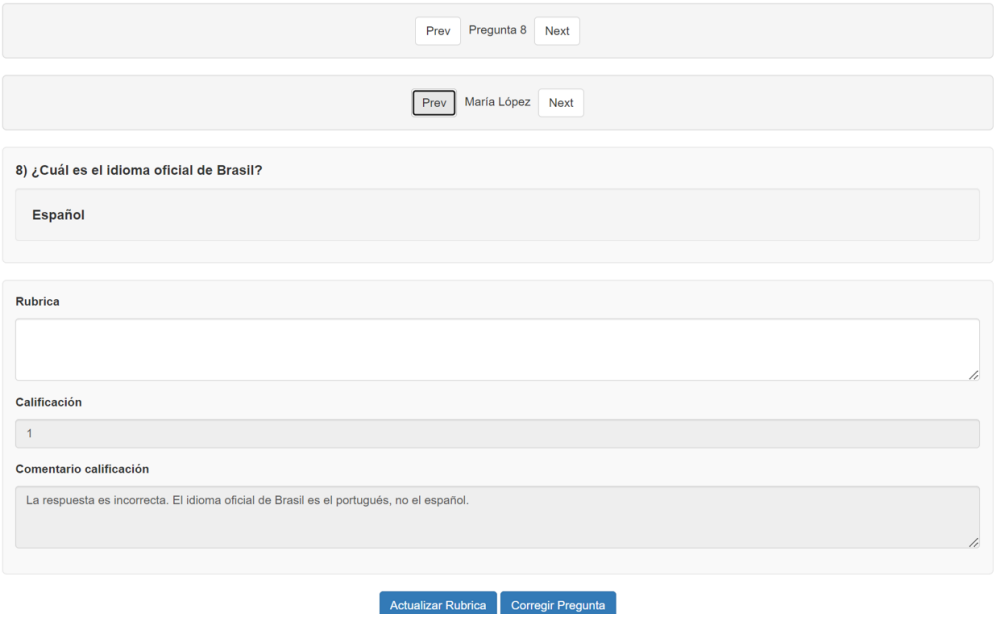


Ilustración 6.14: Respuesta incorrecta estudiante 1

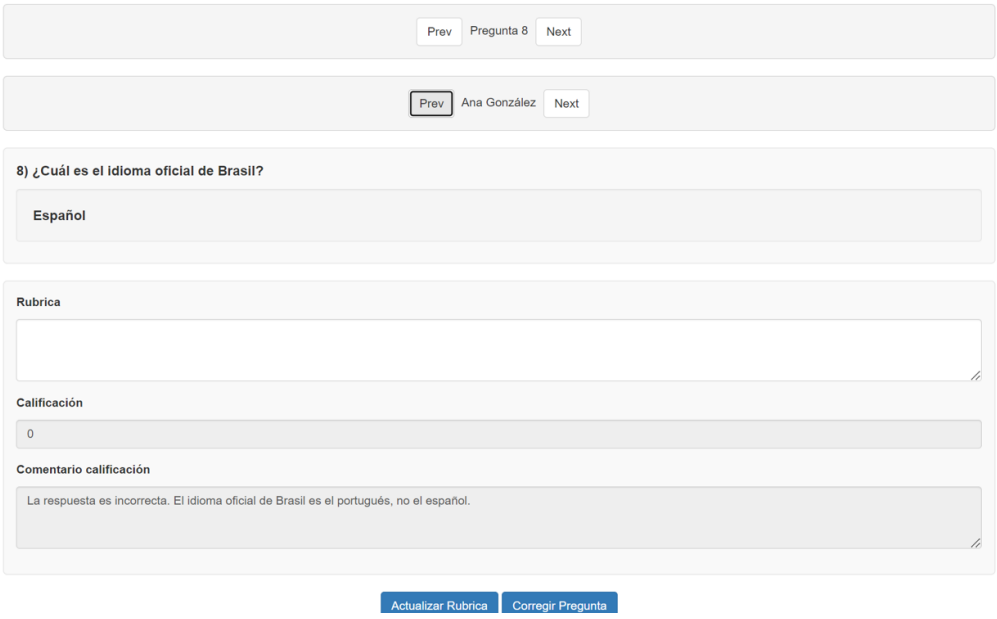


Ilustración 6.15: Respuesta incorrecta estudiante 2

Este comportamiento resalta la necesidad de ajustes para asegurar que la calificación sea

uniforme y equitativa, independientemente de la formulación de las respuestas. Con el objetivo de optimizar esta funcionalidad, se incorporó a la realización de peticiones un parámetro nuevo, la temperatura.

La **temperatura** en modelos de lenguaje controla la creatividad y variabilidad de las respuestas: valores bajos (cerca de 0) generan respuestas más predecibles y coherentes, mientras que valores altos (hasta 2) permiten mayor diversidad y creatividad en el texto generado.

Con esto en mente se decide que tanto la temperatura como el modelo sean parámetros que pueda escoger el usuario para cada examen. Se modifica la interfaz de usuario para incluir estos valores en la creación de examen, y se modifica la comunicación de la interfaz para que tanto el modelo como la temperatura se pasen como variables.

```
chat_completion = client.chat.completions.create(
    messages=messages,
    model=model,
    temperature=temperature
)
```

Ilustración 6.16: Temperatura y Modelo como variables

6.8. Persistencia de datos

Para lograr una capa de abstracción sobre las consultas a la BD se optó por incorporar **SQLAlchemy** a la aplicación, esto se logra con las capacidades del ORM de la misma, permitiendo definir modelos de datos de manera declarativa, realizar consultas complejas y aprovechar las relaciones entre tablas de forma intuitiva a partir de objetos Python en lugar de escribir consultas SQL de forma manual.

6.8.1. Integración de sqlalchemy

Las clases creadas fueron convertidas a un modelo de datos relacional utilizando SQLAlchemy, como se mencionó en el capítulo de Diseño en el apartado Modelado de Datos.

La conversión a un ORM facilitó la interacción con la base de datos, permitiendo una gestión más eficiente y coherente de los datos, así como la realización de consultas y transacciones de manera más intuitiva y sencilla.

En esta etapa, se realizaron pruebas completas de todo el sistema para garantizar su correcto funcionamiento y estabilidad. Estas pruebas abarcaron todos los aspectos de la apli-

cación, desde la funcionalidad básica de las clases y el ORM hasta la integración completa del sistema. Se evaluaron diversos casos de uso para asegurar que cada componente interactuara correctamente y que no hubiera problemas en el flujo de datos.

La estabilidad y fiabilidad del sistema fueron confirmadas gracias a estos rigurosos procesos de prueba, garantizando una experiencia de usuario fluida y sin errores.

6.8.1.1. Cambio en Registro e Inicio de sesión

En esta etapa también se realiza un cambio la tabla de usuario: se renombró el campo username a name y se agregó el campo **email**. Ahora, al registrarse, el usuario debe introducir su nombre, correo electrónico y contraseña. De igual forma, para iniciar sesión, el usuario debe ingresar su correo electrónico y contraseña en lugar de su nombre de usuario.

6.8.2. Corrección de exámenes

Anteriormente se había desarrollado la corrección de preguntas de exámenes, la siguiente implementación fue el desarrollo de la corrección de un examen entero.

Inicialmente se enviaba cada pregunta del examen junto con sus respectivas respuestas de manera secuencial y se persistía la respuesta obtenida de ChatGPT. El tiempo de corrección de un examen podía alcanzar el minuto y medio, dependiendo del número de preguntas y el tipo de respuestas.

Por esta razón, se decidió realizar una optimización, cambiando el envío secuencial por llamadas en paralelo usando hilos.

```
with concurrent.futures.ThreadPoolExecutor() as executor:
    executor.map(grade_answer, questions_data)
```

Ilustración 6.17: Corrección de respuestas por hilos

El tiempo de ejecución se redujo a una tercera parte. Por ejemplo, un examen con 50 preguntas de respuestas simples que antes de la optimización tardaba 1 minuto, ahora solo toma 15 segundos.

6.9. Incorporación de archivos CSV

Al incorporar la posibilidad de usar archivos CSV y no solo XLS, se encontró el problema de que necesitamos saber como está codificado el archivo. Para esto se usó la biblioteca **chardet**.

```python
def read_csv_data(file_path):
    try:
        with open(file_path, 'rb') as f:
            raw_data = f.read()
            result = chardet.detect(raw_data)
            encoding = result['encoding']

        df = pd.read_csv(file_path, encoding=encoding)
        return df
    except Exception as e:
        raise ValueError(f"Error reading CSV file: {str(e)}")
```

Ilustración 6.18: Lectura de archivos CSV

6.10. Mejora de la interfaz de usuario

Una vez se tenía casi toda la funcionalidad desarrollada se trabaja sobre la interfaz de usuario para que sea amigable, eficiente, sencilla y en la medida de lo posible accesible.

Lo primero que se hizo fue asegurarse de que la elección de colores y el contraste con el fondo estuvieran dentro de los parámetros requeridos por el estándar WCAG para que la aplicación fuera más accesible.

Se cambiaron las interfaces de registro e inicio de sesión, eliminando términos en inglés como *login* y *logout* por términos en español para asegurar la coherencia en toda la aplicación. También se les dio otro aspecto para mejorar la usabilidad y la experiencia del usuario.

La interfaz de visualización del examen, que antes tenía una orientación vertical, fue modificada a una visualización horizontal. Esto resultó en una interfaz mucho más amigable e intuitiva, la cual se puede observar en el capítulo de Diseño, en la sección de prototipos de la UI (Figura 5.14).

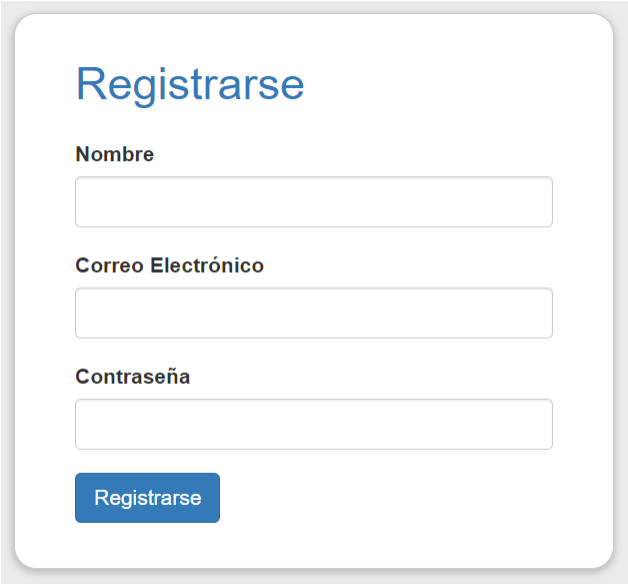


Ilustración 6.19: UI - Registrarse

6.10.1. Creación de vista - ver examen por estudiante

Para mayor facilidad de visualización de datos, se decide crear una vista en la que se pueda ver el examen completo de cada estudiante, se puede navegar entre estudiantes:

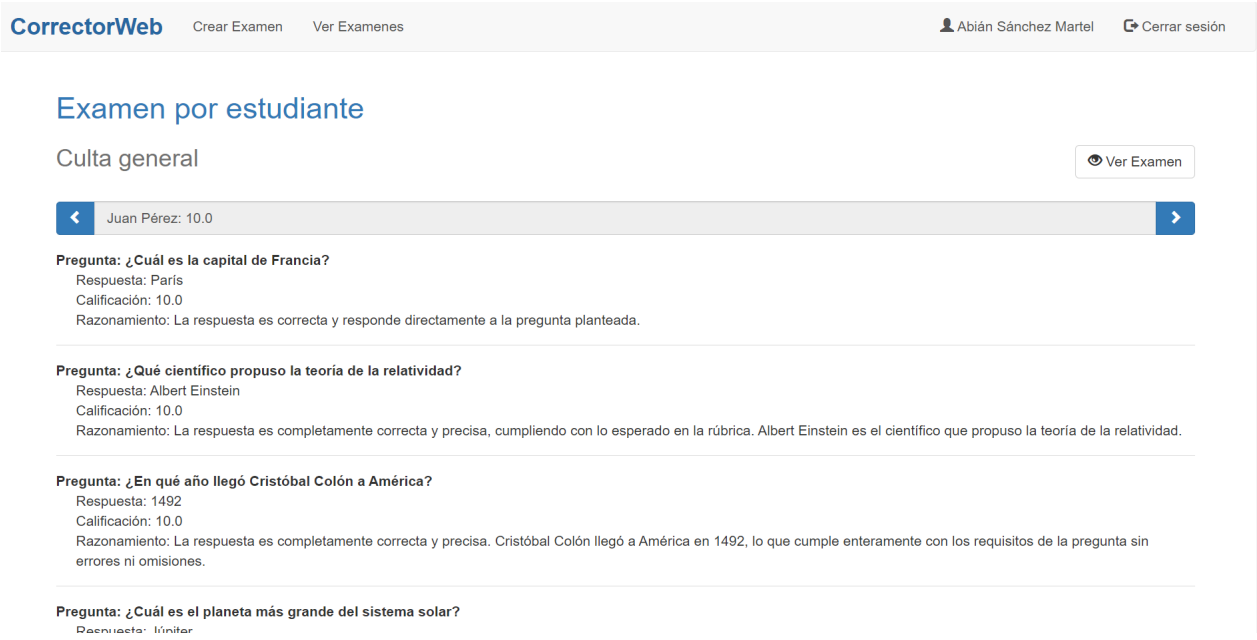


Ilustración 6.20: UI - Examen por estudiante

6.11. Descarga de los resultados de la corrección del examen

Por último, pero no menos importante, se genera un archivo de resultados de los exámenes corregidos. Este archivo mantiene la misma estructura que el archivo de entrada, pero después de cada pregunta se agrega la calificación y la justificación correspondiente.

	A	B	C	D	E	F
1	**Estudiante**	**¿Cuál es la capital de Francia?**	**Calificación**	**Razonamiento**	**¿Qué científico propuso la teoría de la relatividad?**	**Calificación**
2	Juan Pérez	París	10	La respuesta es correcta y responde directamente a la pregunta planteada.	Albert Einstein	10
3	María López	Berlín	0	La respuesta proporciona una capital incorrecta sin relación con la pregunta. La capital de Francia es París, mientras que Berlín es la capital de Alemania.	Isaac Newton	1
4	Ana González	París	10	La respuesta es correcta y cumple plenamente con lo solicitado en la pregunta. París es efectivamente la capital de Francia.	Thomas Edison	0
5	Laura Rodríguez	París	10	La respuesta es correcta y directa. La capital de Francia es realmente París, cumpliendo plenamente con la pregunta formulada.	Albert Einstein	10
6						

Ilustración 6.21: Resultado examen corregido

6.12. Pruebas Finales

Finalmente, se llevaron a cabo las últimas pruebas para verificar que todo el sistema operara de manera óptima y conforme a los requisitos establecidos. Estas pruebas incluyeron una serie de escenarios de uso para asegurar que cada componente del sistema, desde la interfaz de usuario hasta el procesamiento de datos y la generación de informes, funcionara sin errores. Se comprobó la integridad de los datos, la precisión de las calificaciones y la coherencia en la presentación de resultados. Este proceso de validación final garantizó que el sistema no solo cumpliera con las expectativas funcionales, sino que también ofreciera una experiencia de usuario fluida y eficiente.

Capítulo 7

Conclusiones

El desarrollo de esta aplicación web ha sido una experiencia enriquecedora y desafiante, especialmente en el uso de la API de OpenAI. A lo largo del proyecto, se ha adquirido un conocimiento profundo sobre diversas tecnologías y metodologías de desarrollo de software, desde la creación de una aplicación con Flask hasta la integración de inteligencia artificial.

7.1. Framework Flask y Python

La modularidad de Flask ha permitidod dividir el proyecto en componentes manejables, mejorando la claridad y la organización del código. Además, la combinación de Flask con otras tecnologías como Bootstrap, HTML, CSS y JavaScript, resultó en una interfaz de usuario atractiva y funcional.

Gracias al glosario de documentación oficial y a la enorme ayuda de la comunidad en internet, que planteaba problemas y soluciones similares a los que yo enfrentaba, he podido sortear diversos obstáculos en el desarrollo.

El desarrollo con Flask y Python ha demostrado ser una elección acertada para este proyecto, todo gracias al consejo del tutor, pues nunca había programado una aplicación web con Flask. La experiencia adquirida en la creación de esta aplicación web ha sido satisfactoria, proporcionando habilidades prácticas y conocimientos técnicos que serían útiles en futuros trabajos.

7.2. API de OpenAI

La incorporación de la IA al proyecto ha resultado en una herramienta muy potente y fácil de implementar, sin embargo, el factor de aleatoriedad de la misma ha resultado en un quebradero de cabeza cuando se quiere que los resultados sean consistentes para las mismas respuestas.

No fue hasta dar con la solución de ajustar el atributo "Temperature"de la API de OpenAI para controlar el grado de aleatoriedad en las respuestas generadas por la IA. Un valor más bajo de "Temperature"hace que la IA sea más determinista y menos creativa, produciendo respuestas más consistentes y predecibles. Ajustar y restringir este parámetro permitió reducir significativamente la variabilidad en las correcciones, logrando un equilibrio adecuado entre precisión y flexibilidad.

7.2.1. Modelos

Durante el desarrollo de la API se fueron probando diferentes modelos para ajustar los resultados deseados y al final viendo la amplia gama de respuestas obtenidas se planteó disponer de varios a elección del usuario según sus necesidades, los modelos elegidos serían: **"gpt-4o"**, **"gpt-4"**, **"gpt-3.5-turbo"**, **"text-davinci-003"** y **"text-curie-001"**

Esto plantea la tesitura de si merece la pena o no actualizar la aplicación a medida que aparezcan nuevos modelos, pues **"gpt-4o"** apareció este año 2024, **"gpt-4"** salió en 2023 y el resto se lanzaron durante el 2022. Puesto que, lo esperado es que los nuevos modelos sean más eficientes pero a su vez serían más costosos.

Habría que añadir el trabajo de investigación y feedback de los usuarios para ver si el propósito de la aplicación se ve cubierto con unos modelos concretos. También queda resaltar que la curva de aprendizaje para los usuarios novatos se ve ampliada a medida que tenga que aprender conceptos nuevos.

7.3. Valoración final

Visto la enorme aceptación de la IA como herramienta en el entorno social y profesional, cabe esperar que más desarrolladores se involucren en el aprendizaje y creación de proyectos que incorporen asistencia por IA. Esto conllevaría a la automatización de tareas tediosas o

a la simplificación de adaptación a un medio informático cuando parte de la complejidad se ve cubierta con el apoyo de la inteligencia artificial.

Capítulo 8

Trabajos Futuros

Con vistas al futuro se ha decidido describir y planificar una serie de cambios y mejoras que no se recogían en el propósito de este proyecto pero que se podrían implementar en esta aplicación. Estas mejoras están orientadas a aumentar la funcionalidad y la eficiencia tanto para los usuarios como para los administradores, ofreciendo una experiencia más completa y datos más detallados.

8.1. Usuarios

8.1.1. Identificador único para los alumnos:

Añadir en la carga de exámenes un identificador tipo **DNI** o nº de referencia en el centro educativo que diferencie a los alumnos en el caso de que el conjunto de Nombre/Apellidos tenga una coincidencia de 2 o más alumnos.

8.1.2. Incorporar Campo de Asignaturas:

Añadir un campo específico para las asignaturas, permitiendo así asociar exámenes a una asignatura y a un profesor correspondiente.

8.1.3. Vista Profesor/Asignatura:

Desarrollar una vista dedicada que permita a los profesores de la misma asignatura compartir y acceder a los datos de los exámenes. Esto facilitará la colaboración y el intercambio de información entre los profesores.

8.1.4. Estadísticas de Notas de los Alumnos/Asignatura:

Implementar una sección de estadísticas donde los usuarios puedan ver y analizar las notas de los alumnos por asignatura. Esto proporcionará una visión más detallada del rendimiento académico.

8.1.5. Estadísticas de Uso de IA:

Incorporar estadísticas relacionadas con el uso de la inteligencia artificial en la aplicación. Los usuarios podrán ver cómo y con qué frecuencia se utiliza la IA para mejorar la experiencia educativa.

8.1.6. Opciones de Idioma:

Añadir opciones de idioma para que los usuarios puedan interactuar con la aplicación en su lengua preferida. Esto aumentará la accesibilidad y la comodidad de uso para una audiencia más amplia.

8.2. Administrador

8.2.1. Incorporar el Rol de Administrador:

Crear un rol de administrador con permisos especiales para gestionar y supervisar todas las funcionalidades de la aplicación. Esto incluirá la capacidad de realizar configuraciones avanzadas, supervisar el uso global de la aplicación y eliminar usuarios del sistema.

8.2.2. Ver Estadísticas Globales de Uso de IA:

Proporcionar a los administradores acceso a estadísticas globales sobre el uso de la inteligencia artificial por parte de todos los usuarios. Esto permitirá una mejor comprensión del impacto y la eficiencia de las funcionalidades de IA.

8.2.3. Coste de Consultas de IA en el Tiempo:

Implementar un sistema de seguimiento y reporte del coste de las consultas de IA a lo largo del tiempo. Esto ayudará a los administradores a gestionar los recursos y a evaluar la rentabilidad y el costo-beneficio del uso de IA en la aplicación.

Capítulo 9

Manual de Usuario

9.1. Inicio de sesión

Escriba su Correo electrónico y la contraseña asociada en los campos especificados. Una vez completado pulse el botón **Iniciar sesión**.

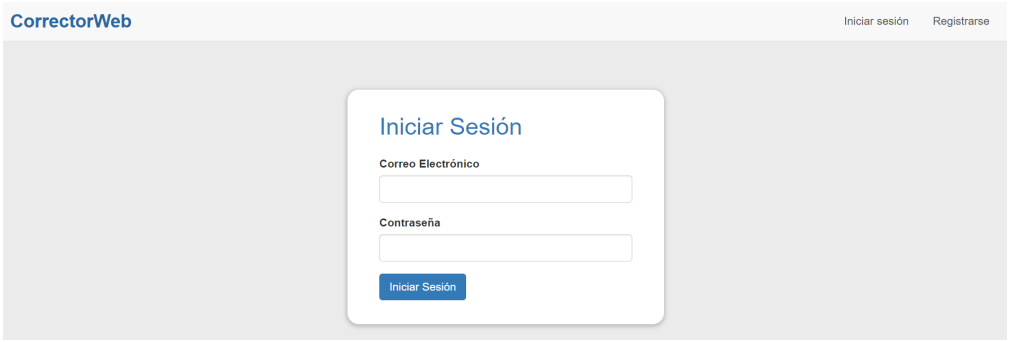


Ilustración 9.1: Iniciar sesión

9.2. Registro de usuario

En caso de no estar dado de alta en la aplicación, pulse el botón **Registrarse"** situado a la derecha del menú superior de la página.

Rellene los campos con su Nombre, Correo electrónico y contraseña de acceso. Continue pulsando el botón **Registrarse**.

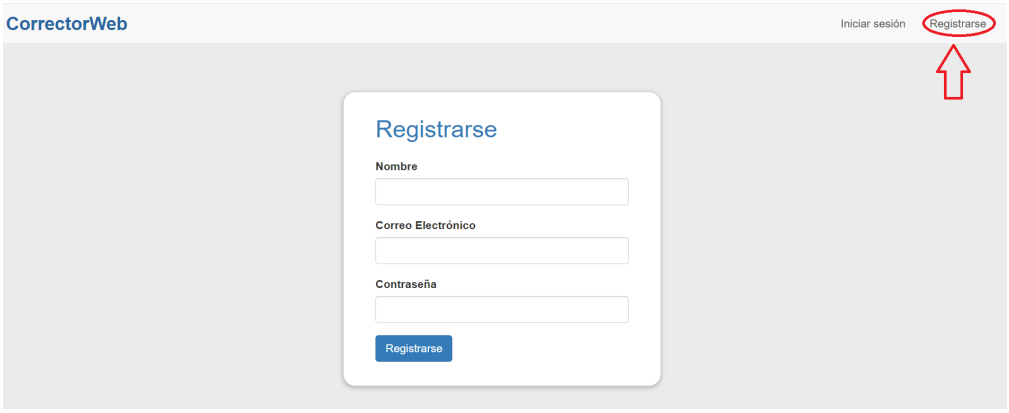


Ilustración 9.2: Registro de nuevo usuario

9.3. Lista de exámenes

La página de inicio muestra la lista de exámenes creados.

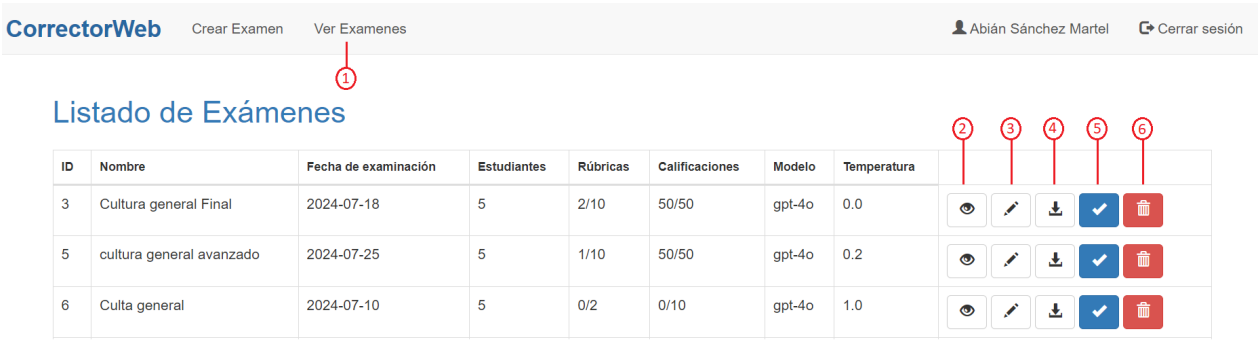


Ilustración 9.3: Página inicial de usuario con la lista de exámenes creados

① Redirecionar a la página para crear un nuevo examen.

② Acceder a la vista de examen.

③ Editar atributos del examen (**Nombre**, **Fecha**, **Modelo** y **Temperatura**).

④ Descargar el examen junto a las calificaciones en formato **".xlsx"**.

⑤ Corregir examen con las **Rúbricas**, **Modelo** y **Temperatura** ya establecidos.

⑥ Eliminar examen.

9.4. Crear examen

Rellene los campos y seleccione un archivo con los resultados. Pulse **"Crear Examen"** para continuar.

Ilustración 9.4: Página de creación de nuevo examen

① Nombre del examen.

② Fecha de realización del examen.

③ Selector de **Fecha** para rellenar el campo.

④ Desplegable con los modelos disponibles.

Los modelos a elegir son:

1. *gpt-4o*
2. *gpt-4*
3. *gpt-3.5-turbo*
4. *text-davinci-003*
5. *text-curie-001*

⑤ Temperatura a aplicar con el modelo elegido.

⑥ Seleccionador del fichero contenedor con las **Preguntas**, **Resultados** y **Alumnos**

9.5. Ver examen

Rellene el campo **Rúbrica** con el criterio de evaluación que desee que corrija la IA. Luego pulse **"Actualizar Rúbrica"** para fijarla.

Pulse el botón **"Corregir Respuesta"** para que evalue la IA y muestre la calificación y el razonamiento a la misma.

Pulse el botón **"Corregir examen"** para evaluar todas las respuestas de loa alumnos con la Rúbrica asociada a cada pregunta.

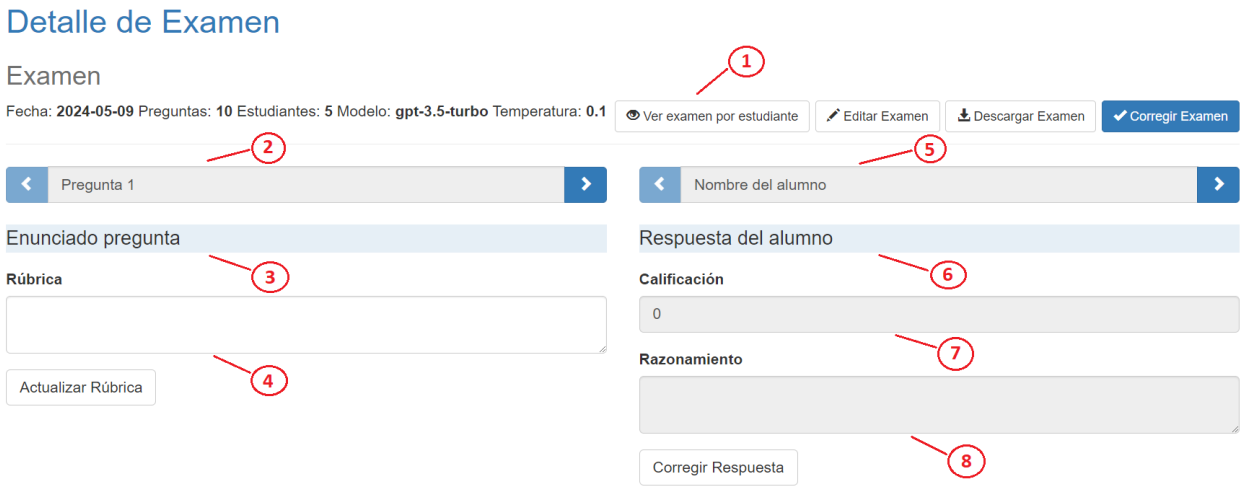


Ilustración 9.5: Página de detalles del examen

① Redireciona a la página visualización de todo el examen alumno por alumno con los campos: **Pregunta**, **Respuesta**, **Calificación** y **Razonamiento**.

② Seleccionador de pregunta.

③ Enunciado de la pregunta seleccionada.

④ Especificación de los criterios de evaluación de la pregunta.

⑤ Seleccionador de alumno.

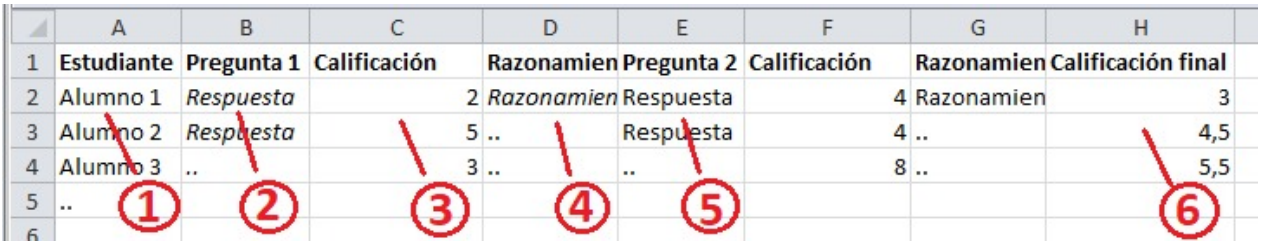
⑥ Respuesta del alumno a la pregunta seleccionada.

⑦ Calificación de la IA.

⑧ Texto explicativo por parte de la IA para adjudicar la calificación.

9.6. Descargar examen

Al descargar el examen corregido se genera un archivo **Excel**.



	A	B	C	D	E	F	G	H
1	**Estudiante**	**Pregunta 1**	**Calificación**	**Razonamien**	**Pregunta 2**	**Calificación**	**Razonamien**	**Calificación final**
2	Alumno 1	*Respuesta*	2	*Razonamien*	Respuesta	4	Razonamien	3
3	Alumno 2	*Respuesta*	5	..	Respuesta	4	..	4,5
4	Alumno 3	..	3	..	..	8	..	5,5
5	..							
6								

Ilustración 9.6: Archivo excel generado con las calificaciones del examen

Este archivo contiene la lista de:

① **Alumnos**.

② **Respuestas** a la **Pregunta**.

③ **Razonamientos** generado por la IA.

④ **Calificaciones** dadas por la IA

⑤ Concatenación de los campos de las siguientes preguntas.

⑥ **Calificación Final** con la media aritmética.

Bibliografía

[1] Juan Carlos Mejía Llano. La historia de la inteligencia artificial: Avances y herramientas pioneras. https://www.juancmejia.com/transformacion-digital/la-historia-de-la-inteligencia-artificial-avances-y-herramientas-pioneras/, 2023. Acceso el 2 de Julio de 2024.

[2] Stanford Encyclopedia of Philosophy. Artificial intelligence. https://plato.stanford.edu/entries/artificial-intelligence/, 2018. Acceso el 2 de Julio de 2024.

[3] Massachusetts Institute of Technology (MIT). Eliza. https://www.masswerk.at/elizabot/, 2024. Acceso el 2 de Julio de 2024.

[4] Terry Winograd. Procedures as a representation for data in a computer program for understanding natural language. https://hci.stanford.edu/winograd/shrdlu/, 2024. Acceso el 2 de Julio de 2024.

[5] gamco. ¿qué es el invierno de la inteligencia artificial? https://gamco.es/glosario/el-invierno-de-la-inteligencia-artificial/, 2024. Acceso el 2 de Julio de 2024.

[6] IBM. Deep blue. https://www.ibm.com/history/deep-blue, 2024. Acceso el 2 de Julio de 2024.

[7] IBM. Ibm watson to watsonx. https://www.ibm.com/watson?mhsrc=ibmsearch_a&mhq=watson, 2024. Acceso el 2 de Julio de 2024.

[8] New York times. Computer wins on 'jeopardy!': Trivial, it's not. https://www.nytimes.com/2011/02/17/science/17jeopardy-watson.html, 2024. Acceso el 2 de Julio de 2024.

[9] Open AI (a través de colaboración). Language models are few-shot learners. https://arxiv.org/abs/2005.14165, 2020. Acceso el 2 de Julio de 2024.

[10] Pallets Projects. Jinja. https://jinja.palletsprojects.com/, 2024. Acceso el 15 de Julio de 2024.

[11] Pallets Projects. Flask. https://flask.palletsprojects.com/, 2023. Acceso el 2 de Julio de 2024.

[12] Bootstrap-Flask. Bootstrap-flask documentation. https://bootstrap-flask.readthedocs.io/, 2023. Acceso el 2 de Julio de 2024.

[13] Pandas Development Team. pandas: powerful python data analysis toolkit. https://pandas.pydata.org/, 2023. Acceso el 2 de Julio de 2024.

[14] Python Software Foundation. Chardet: The universal character encoding detector. https://chardet.readthedocs.io/, 2024. Acceso el 25 de Julio de 2024.

[15] Pallets Projects. Werkzeug. https://werkzeug.palletsprojects.com/, 2023. Acceso el 2 de Julio de 2024.

[16] Pallets Projects. Flask blueprints. https://flask.palletsprojects.com/en/2.0.x/blueprints/, 2023. Acceso el 2 de Julio de 2024.

[17] dotenv linter. dotenv: .env file parser for python. https://pypi.org/project/python-dotenv/, 2023. Acceso el 2 de Julio de 2024.

[18] OpenAI. Openai api. https://beta.openai.com/docs/, 2023. Acceso el 2 de Julio de 2024.

[19] OpenAI. Chatgpt. https://www.openai.com/chatgpt, 2023. Acceso el 2 de Julio de 2024.

[20] MariaDB Foundation. Mariadb. https://mariadb.org/, 2023. Acceso el 2 de Julio de 2024.

[21] SQLAlchemy. Sqlalchemy documentation. https://www.sqlalchemy.org/, 2023. Acceso el 2 de Julio de 2024.

[22] Google. Material design. https://m2.material.io/, 2024. Acceso el 15 de Julio de 2024.

[23] World Wide Web Consortium (W3C). Web content accessibility guidelines (wcag) 2.1. https://www.w3.org/TR/WCAG21/, 2018. Acceso el 2 de Julio de 2024.

[24] Refsnes Data AS. W3schools. https://www.w3schools.com/python/. Acceso el 2 de Febrero de 2024.

[25] Ansgar Becker. Heidisql. https://www.heidisql.com/, 2023. Acceso el 2 de Julio de 2024.