



Universitat de Lleida

# TREBALL DE FI DE GRAU



ESCOLA  
POLITÈCNICA SUPERIOR  
UNIVERSITAT DE LLEIDA  
INSPIRING THE FUTURE

**Estudiant:** César Alejandro Crespo Freire

**Titulació:** Grau en Tècniques d'Interacció Digital i de Computació

**Títol de Treball de Fi de Grau:** Renovando Amadeus con IA

**Director/a:** Jordi Vilaplana Mayoral

Presentació

Mes: Juny

Any: 2025

## **Agradecimientos**

Quisiera empezar agradeciendo a mi tutor Jordi Vilaplana, por toda la paciencia que ha tenido a lo largo de mi TFG y por haberme guiado en este proyecto.

Agradecer a la Fundación Esplai y mis compañeros, por la formación que me dieron y el soporte que me brindaron a lo largo de esos 4 meses.

Para finalizar a mi familia por haberme apoyado durante todo este tiempo

## Contenido

<b>1. Introducción</b>	<b>4</b>
<b>2. Motivación</b>	<b>4</b>
<b>3. Objetivos</b>	<b>5</b>
<b>4. Estado del arte</b>	<b>5</b>
<b>5. Planning</b>	<b>7</b>
<b>6. Análisis de requisitos</b>	<b>8</b>
<b>7. Desarrollo</b>	<b>9</b>
7.1. Base de datos	9
7.1.1. Datos de usuarios	9
7.1.1.1. auth_user	10
7.1.1.2. PerfilUsuario	10
7.1.1.3. Planes	10
7.1.1.4. Pagos	10
7.1.1.5. Accesos	10
7.1.2. Bloque de Amadeus	11
7.1.2.1. Vuelos	11
7.1.2.2. Reservas	11
7.1.2.3. ClasesVuelo	11
7.1.2.4. DisponibilidadVuelos	12
7.1.2.5. Tarifas	12
7.1.2.6. Aerolíneas y Aviones	12
7.1.2.7. Países, Regiones y Aeropuertos	12
7.1.2.8. Usuarios y Contacto	12
7.2. Diseño web	12
7.2.1. Proximidad	12
7.2.2. Similitud	13
7.2.3. Énfasis	14
7.2.4. Estética y facilidad de uso	15
7.2.5. Superioridad visual de la imagen	15
7.2.6. Relación coste-beneficio	15
7.2.7. Visibilidad	16
7.2.8. Libertad de interacción	16
7.3. Lógica de la aplicación	17
7.3.1. Controladores (Views en Django)	17
7.3.1.1. AI	17
7.3.1.2. FlyBot (modelo predictivo de precios)	17
7.3.1.3. Simulator	18
7.3.1.4. Pages	18
7.3.1.5. Users	18
7.3.2. Modelos	18

7.3.2.1.	Users	19
7.3.2.2.	Simulador	19
7.3.2.3.	Flybot	20
7.3.2.4.	Otros modelos	20
7.3.3.	Vista	20
7.3.3.1.	Interacción con el simulador Amadeus	20
7.3.3.2.	Autocompletado y sugerencias inteligentes	21
7.3.3.3.	Chatbot inteligente	21
7.3.3.4.	Predicción de precios	22
7.3.3.5.	Estilo y diseño visual	22
7.4.	Análisis y conclusiones	23
<b>Referencias</b>		<b>25</b>

## **1. Introducción**

Hoy en día la inteligencia artificial (IA) [18] se ha convertido en una herramienta fundamental para optimizar procesos, facilitar el aprendizaje, mejorar la eficiencia en diferentes sectores y llegar a hacer cosas que nunca nos hubiéramos imaginado hace unos años, como que un coche se pudiera llegar a conducir solo.

Por otro lado, en el sector del turismo, agencias de viaje y aerolíneas siempre han usado el mismo software para hacer reservas de vuelos, hoteles y coches, llamado Amadeus [15], que fue desarrollado en el 1987 y actualmente es líder en el sector.

Pero a pesar de su gran infraestructura y años en el mercado es un software bastante anticuado y que comparando con otros de la actualidad se hace bastante tedioso y complicado de usarlo si no has tenido una formación previa, donde sus cursos no son muy accesibles para cualquier persona que se quiera preparar en Amadeus. Es en este contexto donde surge la necesidad de una herramienta que simplifique el acceso a la información, facilite la comprensión del sistema y automatice tareas repetitivas o propensas a error a la vez que puedas seguir practicando y aprendiendo de este software.

Este proyecto consiste en el desarrollo de una página web que integre inteligencia artificial para cuando estes interactuando con el sistema. Su objetivo principal es servir como asistente inteligente que no solo ayude en la operativa diaria, sino que también funcione como una herramienta de apoyo en el proceso de aprendizaje para nuevos usuarios o estudiantes del sector turístico.

Combinando una interfaz web accesible, junto con la integración de la IA, permitirá una experiencia más fluida y eficaz a la hora de utilizar Amadeus, optimizando así tanto el tiempo como el nivel de comprensión de los usuarios.

## **2. Motivación**

A principios de este año me apunté a un curso de inteligencia artificial, para aprovechar las tardes mientras hacía mis practicas universitarias por la mañana.

Gracias a ese curso pude asistir al Mobile World Congress y vi muchos proyectos increíbles con IA para la salud de las personas, ayudar a personas con diferentes discapacidades y algunos proyectos educativos.

Todo lo que vi me motivo a adentrarme del todo en este mundo y buscar en algún futuro poder participar en algún proyecto de ese tipo y aportar mi granito de arena.

Para finalizar el curso teníamos que hacer un proyecto, aplicando diferentes tipos de inteligencia artificial que habíamos aprendido. No se me ocurría de que podía hacerlo, hasta que un día cenando con un grupo de amigos, uno me comento que

había empezado en una agencia de viaje y me explico lo mal que lo estaba pasando con Amadeus, ahí fue cuando descubrí que era ese Software y que hacía, ahí vi la oportunidad de poder desarrollar algo con lo que poder ayudar a todos los estudiantes de Amadeus y a todos los trabajadores que tienen que pasar horas clicando su código.

### 3. Objetivos

Los principales objetivos de mi TFG son:

Para empezar, realizar una investigación y análisis del funcionamiento del sistema Amadeus, identificando los principales problemas a los que se enfrentan los usuarios y determinando qué procesos pueden beneficiarse del uso de inteligencia artificial. Esto servirá de base para definir las funcionalidades clave de la plataforma.

Después, se diseñará la arquitectura general del proyecto, definiendo tanto la estructura de la base de datos como los flujos de navegación de la aplicación. El desarrollo del backend se realizará utilizando el framework Django [1], aprovechando su robustez, su sistema de administración integrado y su facilidad de integración con modelos de inteligencia artificial.

A la vez se hará el frontend utilizando HTML, CSS [10] y JavaScript [11], con el objetivo de mantener una interfaz sencilla, accesible y fácil de mantener. Se diseñará una interfaz clara e intuitiva, adaptada a usuarios con diferentes niveles de experiencia en el uso de Amadeus.

Después de haber analizado los principales problemas de los usuarios he decidido que las IAs que desarrollare serán:

- Chatbot con la API de open AI [3] o Gemini [4] donde puedas hacer diferentes tipos de consultas.
- Texto predictivo.
- Recomendaciones de código.
- Predicciones en los precios de los vuelos.

### 4. Estado del arte

En este punto voy a explicar brevemente las tecnologías, herramientas y conceptos claves que usare a lo largo del trabajo, con el objetivo de que todos los elementos técnicos involucrados sean comprensibles y estén bien definidos.

- **Aplicación web:** Es un software que se ejecuta en un servidor web y se accede mediante un navegador. Suele combinar código del lado del cliente y

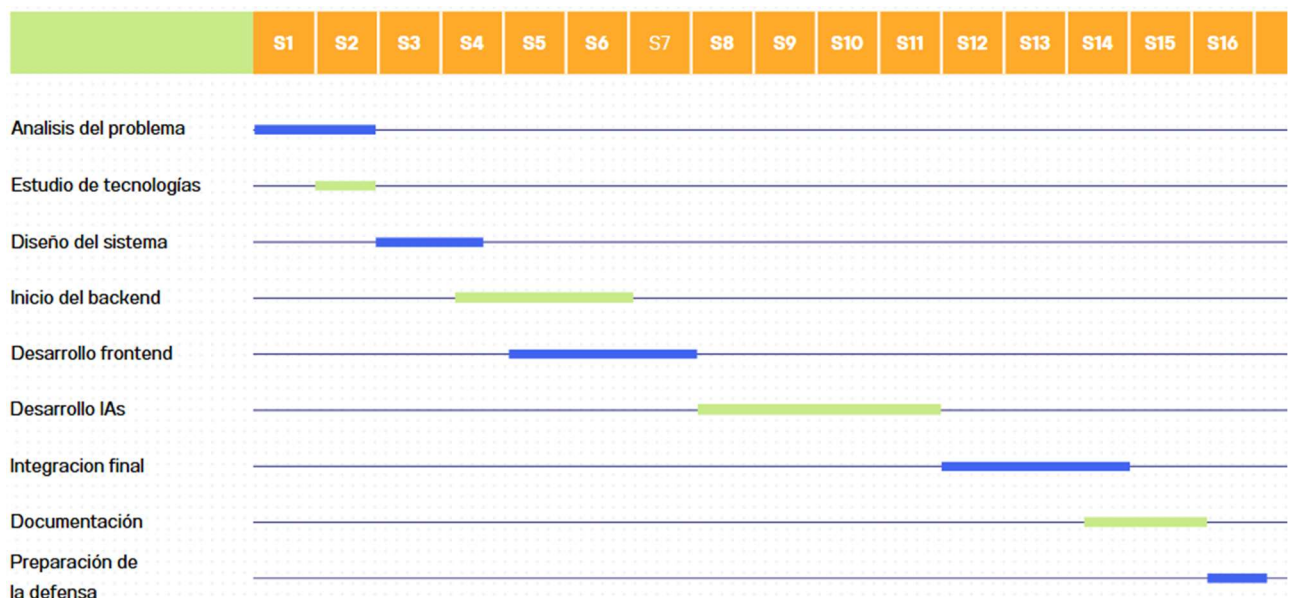
del lado del servidor, y puede requerir autenticación y comunicación con una base de datos.

- **Sitio web:** Conjunto de páginas web relacionadas bajo un mismo dominio. Puede incluir imágenes, texto, videos y funcionalidades interactivas. Puede ir desde páginas estáticas simples hasta plataformas dinámicas complejas.
- **Servidor web:** Software (y a veces también hardware) encargado de procesar las solicitudes HTTP/HTTPS de los clientes (normalmente navegadores) y responder con contenido, como archivos HTML o datos provenientes de una base de datos.
- **Frontend:** Es la parte visible de la aplicación, con la que el usuario interactúa. En mi proyecto, el frontend lo he desarrollado utilizando HTML, CSS y JavaScript puro.
- **Interfaz de usuario (UI):** Punto de contacto entre el usuario y el sistema. El diseño de esta interfaz fue pensado para que el usuario tenga experiencia intuitiva y clara, simulando el entorno de trabajo de Amadeus de forma más simple y asistiendo al usuario durante su uso.
- **Backend:** Es la parte lógica del sistema que se ejecuta en el servidor. Se encarga de procesar peticiones, gestionar la base de datos y servir contenido dinámico. En este proyecto he utilizado el framework **Django** para implementar el backend.
- **Django:** Framework web de alto nivel escrito en Python [2] que fomenta el desarrollo rápido y un diseño limpio. Proporciona herramientas integradas para enrutamiento, modelado de base de datos, interfaces de administración, autenticación, entre otros.
- **Visual Studio Code [14]:** Editor de código ligero pero potente, utilizado durante todo el desarrollo del proyecto. Ofrece soporte para Python, HTML, CSS, JavaScript y control de versiones.
- **Pytest [13]:** Framework de pruebas en Python, empleado para validar funcionalidades del sistema y garantizar la estabilidad del proyecto mediante pruebas automatizadas.
- **Python:** Lenguaje principal del proyecto, utilizado tanto para la lógica del backend como para los módulos de inteligencia artificial desarrollados.
- **JavaScript:** Utilizado en el frontend para manejar la interactividad, responder a eventos del usuario y actualizar dinámicamente la interfaz.
- **HTML5:** Lenguaje de marcado utilizado para estructurar las páginas web de la aplicación.
- **CSS3:** Lenguaje utilizado para definir el estilo visual de las páginas (colores, disposición, fuentes, etc.).

- **JSON [12]**: Formato ligero de intercambio de datos, utilizado para enviar y recibir información entre el cliente y el servidor, especialmente en los módulos relacionados con IA.
- **Anaconda[5]**: Entorno de desarrollo utilizado para gestionar paquetes y entornos virtuales durante el desarrollo de los módulos de inteligencia artificial.
- **Pandas[6]**: Librería de Python para la manipulación y análisis de datos, especialmente útil para trabajar con estructuras como tablas o series temporales.
- **API Gemini [4]**: Interfaz de inteligencia artificial proporcionada por Google, utilizada para generar respuestas y asistir al usuario mediante modelos de lenguaje.
- **Scikit-learn [8]**: Librería de aprendizaje automático en Python, utilizada para entrenar y evaluar modelos predictivos dentro del proyecto.
- **NumPy [7]**: Biblioteca fundamental para el cálculo numérico en Python, empleada para operaciones con matrices y estructuras de datos eficientes.
- **TensorFlow [9]**: Framework de código abierto para el desarrollo y entrenamiento de modelos de machine learning y deep learning.

## 5. Planning

Tenía 4 meses para terminar el proyecto, con lo cual lo organice de la siguiente forma.



**Figura 1:** Planning

## **6. Análisis de requisitos**

El objetivo principal del sistema es ofrecer una herramienta web que permita a estudiantes y futuros profesionales del sector turístico practicar con una simulación del entorno Amadeus, asistida por inteligencia artificial. Esta aplicación busca facilitar el aprendizaje, reducir la curva de entrada y automatizar parte de las consultas y comandos que normalmente deben aprenderse de forma manual.

Para alcanzar este objetivo, se realiza un análisis detallado de los requisitos tanto desde el punto de vista del usuario como del software.

### **Requisitos del usuario**

Los requisitos funcionales que debe permitir el sistema al usuario son los siguientes:

- El usuario puede registrarse y acceder a su cuenta para utilizar el simulador.
- El usuario accede a una landing page donde se muestra información general del proyecto y empresa si no está dentro de ningún usuario registrado.
- El usuario puede escribir comandos en una simulación del entorno Amadeus.
- El sistema ofrece texto predictivo mientras el usuario escribe un comando, facilitando la escritura.
- El sistema muestra un menú de sugerencias de código conforme el usuario va escribiendo.
- El usuario puede hacer clic en un chatbot con interfaz visual para recibir ayuda relacionada con el uso de Amadeus.
- El usuario puede consultar mediante comando las posibilidades de que el precio de un vuelo suba o baje.
- El sistema debe ser accesible desde los principales navegadores en ordenadores de escritorio.
- El sistema debe ser claro, visual e intuitivo para personas en etapa de aprendizaje.

### **Requisitos del software**

Desde el punto de vista técnico, el sistema debe cumplir con los siguientes requisitos:

- El sistema debe guardar y reutilizar búsquedas previas para mejorar el texto predictivo.

- Las consultas realizadas por el usuario se gestionan como datos temporales, sin necesidad de almacenamiento persistente.
- La plataforma debe funcionar correctamente en navegadores como Google Chrome, Microsoft Edge, Mozilla Firefox y Safari en versión escritorio.
- La versión gratuita del sistema debe limitar el número de consultas (máximo 50), mientras que la versión premium debe permitir uso ilimitado.
- El sistema debe ofrecer sugerencias en tiempo real, con un tiempo de respuesta inferior a los 5 segundos.
- La interfaz debe estar optimizada para ordenadores, sin requerimientos específicos para móviles (por ahora).

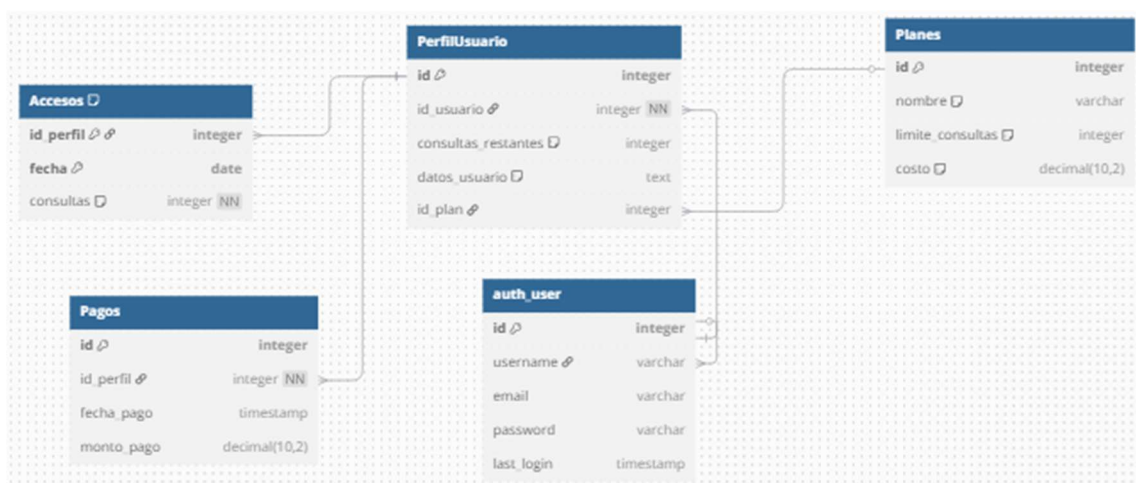
## 7. Desarrollo

### 7.1 Base de datos

Para poder empezar con el proyecto, primero he diseñado la estructura que deberá tener mi base de datos, como se tiene que organizar y relacionar las diferentes tablas entre ellas. Al principio las tablas eran muy diferente, ya que mientras iba planteando todo el proyecto iban apareciendo nuevas necesidades, hasta que obtuve el siguiente resultado. Este modelo permite gestionar toda la información necesaria relacionada con usuarios, vuelos, reservas, precios, accesos, consultas y pagos.

La base de datos se podría dividir en dos bloques, el primer bloque esta más orientado a toda la información relacionada con los usuarios dentro de la web y el segundo bloque estaría relacionado con los datos que se necesitaría para utilizar Amadeus.

#### 7.1.1 Datos de usuarios



## **Figura 2: UML tablas de usuarios**

### **7.1.1.1 auth\_user**

Tabla proporcionada por Django que gestiona el sistema de autenticación de usuarios. Contiene campos como id, username, email, password y last\_login.

### **7.1.1.2 PerfilUsuario**

Esta tabla complementa la información del usuario autenticado. Guarda datos como:

- consultas\_restantes (para gestionar los límites de uso)
- datos\_usuario
- id\_plan, que referencia al plan activo del usuario.

### **7.1.1.3 Planes**

Contiene los diferentes planes disponibles (gratis o premium), con la siguiente información:

- nombre del plan
- limite\_consultas
- costo

### **7.1.1.4 Pagos**

Relacionada con PerfilUsuario, registra los pagos realizados por los usuarios. Incluye campos como:

- fecha\_pago
- monto\_pago

### **7.1.1.5 Accesos**

Lleva el control de las consultas realizadas por cada usuario por día, útil para estadísticas y gestión de límites.

## 7.1.2 Bloque de Amadeus

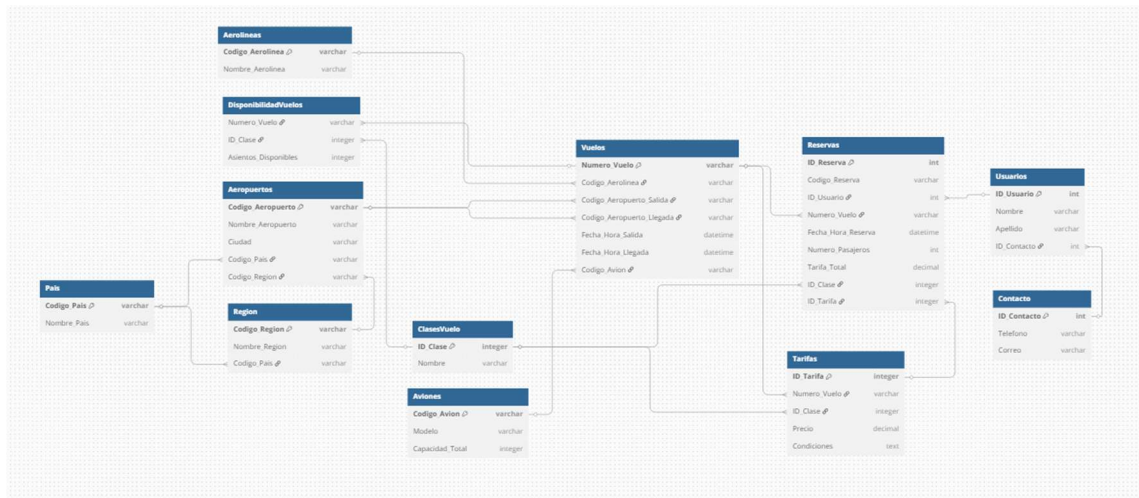


Figura 3: UML Amadeus

### 7.1.2.1 Vuelos

Entidad central que almacena los datos de cada vuelo, como:

- Número de vuelo
- Aerolínea
- Aeropuertos de salida y llegada
- Fechas y horas
- Avión asignado

### 7.1.2.2 Reservas

Tabla que registra las reservas realizadas por los usuarios, incluyendo:

- Número de vuelo
- Fecha y hora de la reserva
- Número de pasajeros
- Tarifa total
- Clase y tarifa asociadas

### 7.1.2.3 ClasesVuelo

Define las clases disponibles (económica o business) para cada vuelo.

#### **7.1.2.4 DisponibilidadVuelos**

Indica cuántos asientos hay disponibles por clase en cada vuelo.

#### **7.1.2.5 Tarifas**

Contiene los precios por clase para cada vuelo y las condiciones asociadas.

#### **7.1.2.6 Aerolíneas y Aviones**

Contienen información de las compañías aéreas y los aviones utilizados en los vuelos, incluyendo modelo y capacidad.

#### **7.1.2.7 Países, Regiones y Aeropuertos**

Conjuntos de tablas que estructuran la geografía de origen y destino de los vuelos. Permiten vincular cada aeropuerto a su país y región correspondiente.

#### **7.1.2.8 Usuarios y Contacto**

Información adicional del usuario final, con campos como nombre, apellido y datos de contacto (teléfono, correo).

### **7.2 Diseño web**

Para el diseño web he aplicado los conceptos claves que he aprendido a lo largo del grado, centrándome sobre todo en la usabilidad y la claridad en la interacción. A continuación, presentare algunos de los principios de diseños más relevantes que se han aplicado.

#### **7.2.1 Proximidad**

Los elementos relacionados los he agrupado de forma visual para facilitar su comprensión. Un ejemplo claro se puede ver en la sección de “Planes Disponibles”, donde las opciones de suscripción están alineadas y uniformemente distribuidas, lo que ayuda al usuario a compararlas rápidamente.

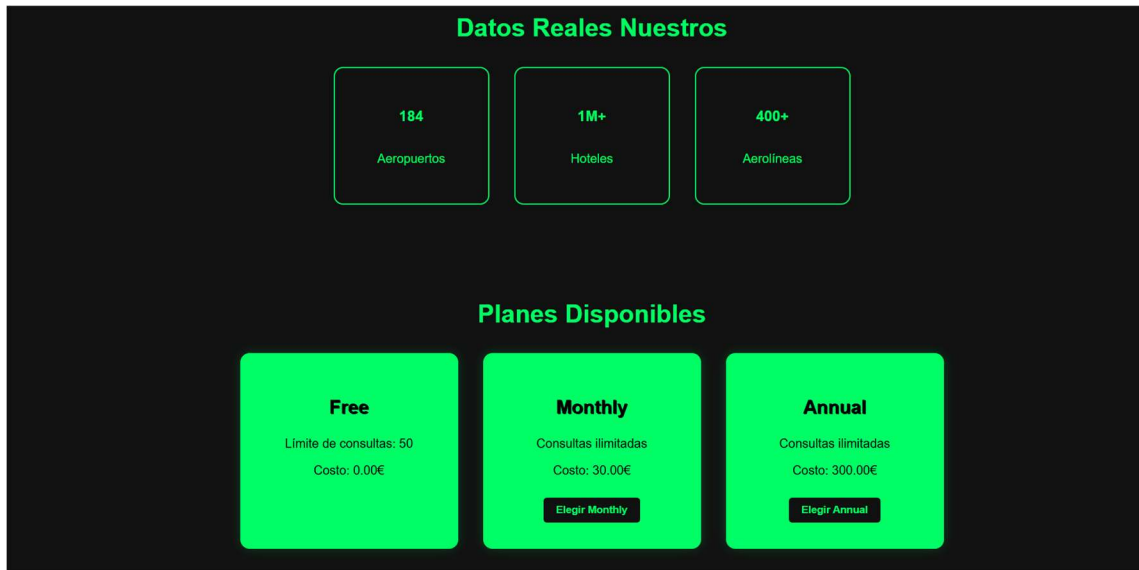


Figura 4: Proximidad

### 7.2.2 Similitud

He utilizado colores, tipografías y estructuras similares en distintos componentes, como las tarjetas de “Datos Reales Nuestros” o los formularios. Esto permite al usuario identificar rápidamente que se trata de elementos del mismo tipo o funcionalidad.

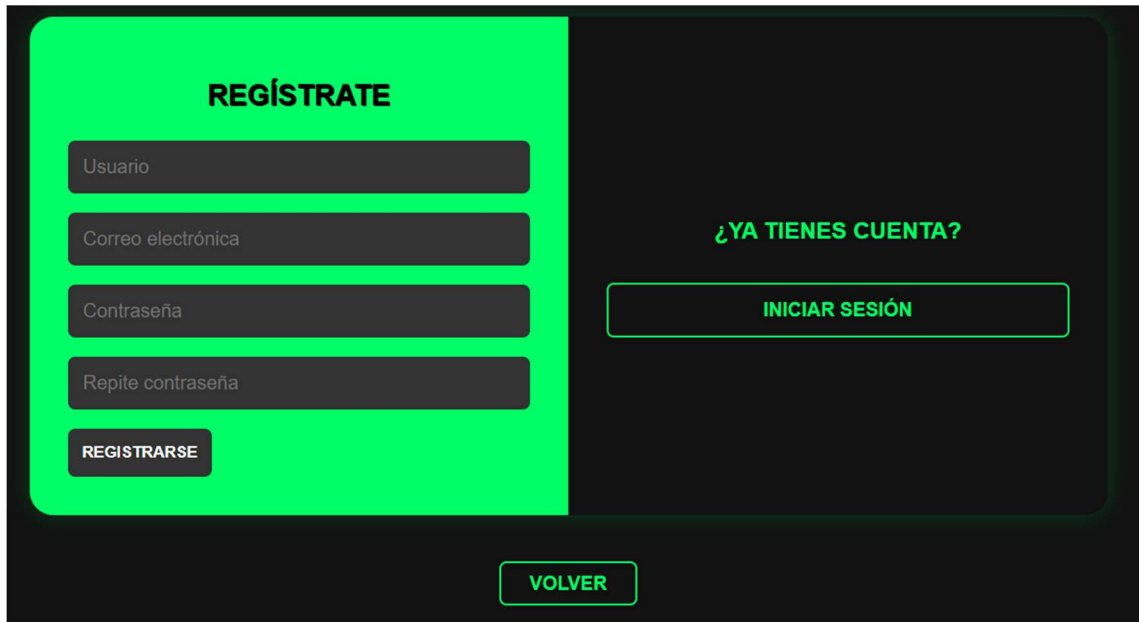
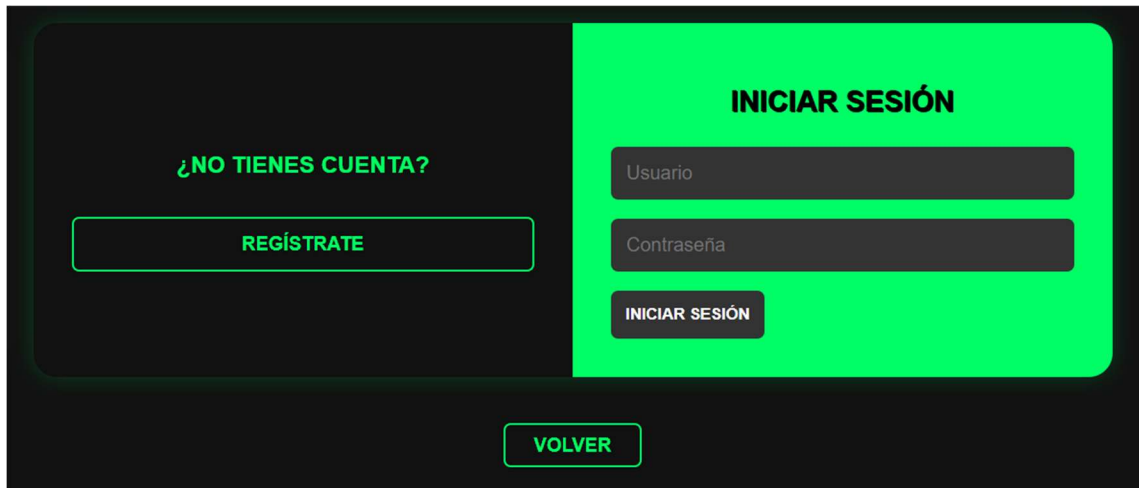


Figura 5: Similitud 1



**Figura 6:** Similitud 2

### 7.2.3 Énfasis

Utilice el color verde brillante para resaltar elementos importantes, como botones, títulos, contadores y enlaces. También se aplican efectos como el cambio de tamaño y sombreado al hacer hover. El asistente virtual, por ejemplo, se muestra en la esquina inferior derecha con la imagen de un robot, que si le das clic se te abre un chat con él, donde la caja de la conversación se puede cambiar de tamaño y mover alrededor de la pantalla.



## Figura 7: Enfasis

### 7.2.4 Estética y facilidad de uso

El diseño mantiene un estilo minimalista con una paleta de colores oscura (fondo negro) y detalles en verde y blanco, lo cual mejora el contraste y facilita la lectura. Se ha mantenido coherencia en el estilo visual en todas las secciones (landing page, simulador, perfil, contacto, etc.), lo que ayuda a que el usuario no se sienta desorientado al navegar.

### 7.2.5 Superioridad visual de la imagen

En la landing page, se utilizan imágenes llamativas (como la del avión al atardecer) para captar la atención del usuario y transmitir de forma inmediata el enfoque del proyecto. Además, en la sección de demostración del simulador, se incluye una captura visual del entorno tipo Amadeus para mejorar la comprensión.

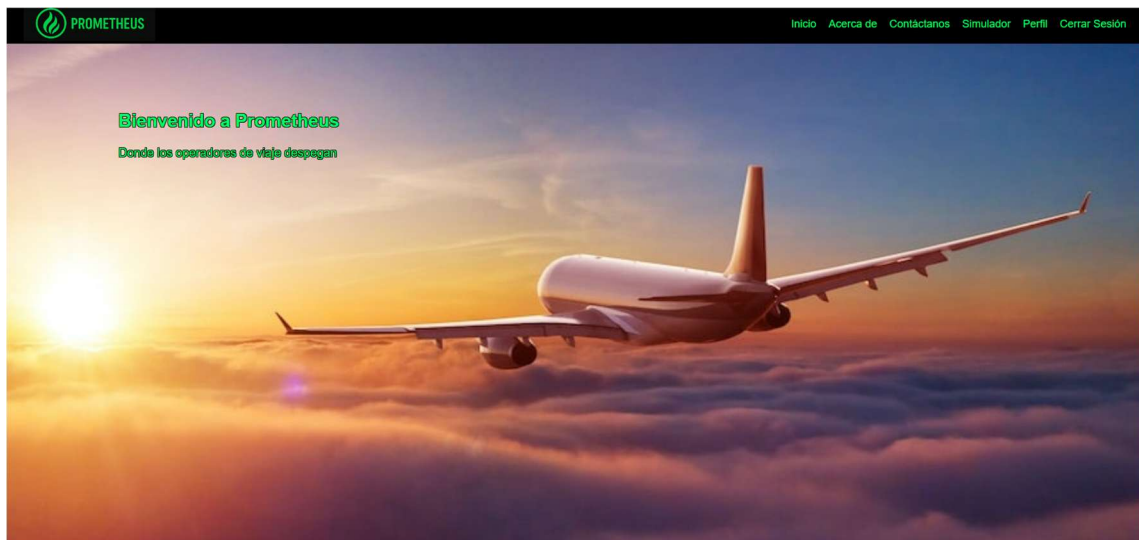


Figura 8: Superioridad visual de la imagen

### 7.2.6 Relación coste-beneficio

La aplicación está pensada para que el usuario pueda comenzar a utilizarla sin necesidad de registrarse, accediendo a contenido informativo o demostrativo. Para acceder al simulador, sí se requiere registro, pero el plan gratuito ofrece 50 consultas, y hay planes premium para uso intensivo. Además, la carga es rápida gracias a la estructura ligera en HTML, CSS y JavaScript.



**Figura 9:** Relación coste-beneficio

### 7.2.7 Visibilidad

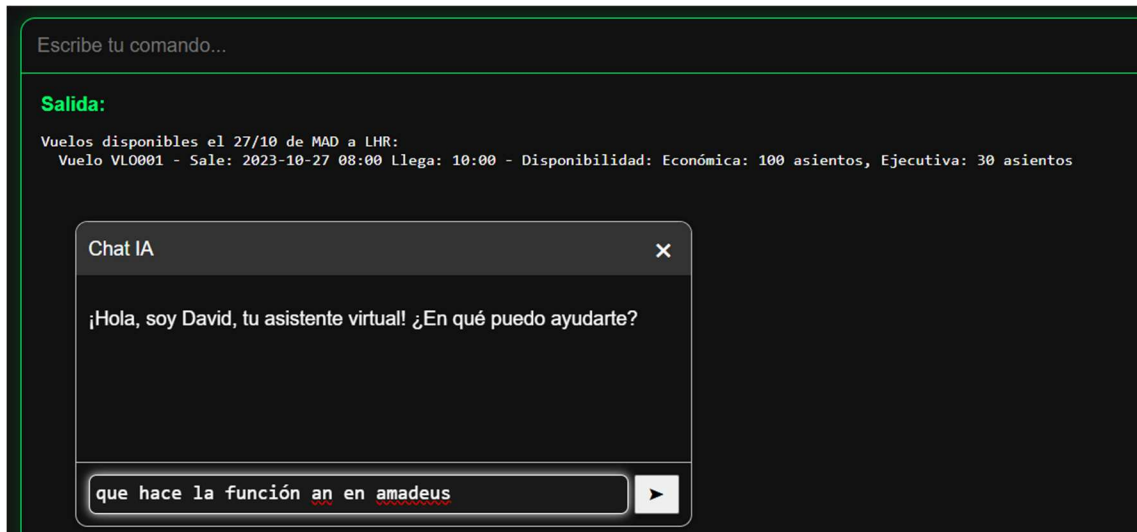
He buscado garantizar que los colores tengan suficiente contraste, incluso en pantallas oscuras, utilizando herramientas como Contrast Finder [17]. Además, las funciones están claramente identificadas: los botones están bien etiquetados, el chatbot es accesible y visible en todo momento, y la navegación superior permite moverse rápidamente por las diferentes secciones de la web.

### 7.2.8 Libertad de interacción

Una característica destacada del diseño de esta aplicación es la libertad de interacción del usuario con el asistente virtual (chatbot). A diferencia de otros sistemas en los que los asistentes se encuentran fijos en una posición, en esta aplicación el usuario tiene la posibilidad de:

- Mover la ventana del chatbot por la pantalla, según su preferencia y comodidad.
- Redimensionar el tamaño de la caja del chat, permitiendo ajustarlo a sus necesidades, ya sea para mantenerlo discreto o para leer largas respuestas con mayor comodidad.

Este diseño fomenta la autonomía del usuario, ya que no impone una posición o tamaño predefinido que pudiera obstaculizar la experiencia en el simulador. Gracias a esta libertad, el asistente se adapta al flujo de trabajo y no al revés, mejorando significativamente la usabilidad, accesibilidad y personalización del entorno.



**Figura 10:** Libertad de interacción

### 7.3 Lógica de la aplicación

La estructura lógica del proyecto sigue el patrón de diseño Modelo–Vista–Controlador (MVC) [16], el cual permite una separación clara entre la lógica del negocio, la interfaz de usuario y el manejo de datos. Este enfoque facilita el mantenimiento del proyecto, la escalabilidad y la organización del código fuente.

#### 7.3.1 Controladores (Views en Django)

En Django, los controladores están representados por las funciones o clases dentro del archivo `views.py` de cada módulo. En este proyecto, se han hecho controladores específicos para tratar tanto las peticiones de la interfaz como las respuestas del sistema IA.

##### 7.3.1.1 AI

- `chat_with_amadeus`: vista que recibe mensajes del usuario y responde mediante el modelo de IA integrado (Gemini), gestionado por la función `ask_amadeus_chatbot()`.
- `chat_interface`: renderiza la interfaz HTML donde se muestra la caja de diálogo del asistente virtual.

##### 7.3.1.2 FlyBot (modelo predictivo de precios)

- `predict_view`: recibe un JSON con datos de un vuelo y realiza una predicción del precio usando un modelo LSTM previamente entrenado. Valida campos numéricos y categóricos según el preprocesador cargado.
- `dashboard_view`: vista general del sistema de predicción, pensada para monitorización (RMSE, MAE, etc.).

### 7.3.1.3 Simulator

- `simulator_view`: muestra el simulador de comandos Amadeus. Gestiona el conteo de consultas disponibles y muestra el resultado del comando procesado.
- `autocomplete_command_ml`: devuelve sugerencias inteligentes para autocompletar comandos usando un modelo basado en scikit-learn.
- `alternative_routes_view`: ofrece rutas alternativas (como días diferentes) en base al comando introducido por el usuario.

### 7.3.1.4 Pages

- `index`: landing page pública con información y planes disponibles.
- `about`, `terms`, `contact_us`: páginas estáticas del proyecto. En `contact_us`, simula el envío de mensajes al equipo de soporte, en `terms` están los términos y condiciones de la página y el `about` contiene la información sobre la empresa.

### 7.3.1.5 Users

- `login_view`, `register_view`, `logout_view`: controlan el acceso de usuarios, los registros de nuevos usuarios y como salir de tu cuenta después de haber iniciado sesión.
- `profile_view`: muestra los datos del perfil actual, incluyendo el plan y las consultas restantes.
- `payment_view`: gestiona la lógica para cambiar de plan y registrar un pago (en este caso es simulado).

## 7.3.2 Modelos

En mi proyecto, los modelos cumplen una doble función: por un lado, estructuran y almacenan los datos en la base de datos (usando Django ORM) y, por otro lado,

permiten la ejecución de modelos de predicción de inteligencia artificial en el backend, como el predictor de precios de vuelos.

### 7.3.2.1 Users

#### **auth\_user (modelo de Django)**

Modelo de autenticación base. Gestiona el acceso de los usuarios registrados y almacena atributos como username, email, password y last\_login.

#### **PerfilUsuario**

Extiende el modelo de Users con campos personalizados:

- id\_usuario: clave foránea hacia auth\_user.
- consultas\_restantes: número de consultas disponibles.
- id\_plan: referencia al plan actual del usuario.
- datos\_usuario: información adicional del perfil.

#### **Planes**

Contiene los diferentes tipos de suscripciones (Free, Premium, etc.):

- nombre: nombre del plan.
- limite\_consultas: número máximo de consultas si aplica.
- costo: precio del plan.

#### **Pagos**

Registra las transacciones realizadas por los usuarios:

- id\_perfil.
- fecha\_pago.
- monto\_pago.

### 7.3.2.2 Simulador

No contiene modelos propios, pero se conecta con los comandos simulados y modelos IA para ofrecer funciones como:

- Interpretación de comandos (interpretar\_comando).
- Autocompletado basado en búsquedas anteriores (CommandAutoCompleteModel).
- Sugerencias de rutas alternativas (CommandFlightSuggestionModel).

### 7.3.2.3 Flybot

Aunque no está relacionada con un modelo de base de datos, la lógica del predictor se basa en una clase personalizada, clase que:

- Carga un modelo LSTM entrenado (flybot\_lstm\_model\_full.keras) y su preprocesador (.pkl).
- Valida los campos numéricos y categóricos de entrada.
- Realiza la predicción de precios de vuelos a partir de los datos guardados de otros años.

Los datos de entrada deben estar perfectamente alineados con los requerimientos del preprocesador para que el modelo funcione correctamente. Se usan columnas como: origen, destino, fecha, número de pasajeros, clase, aerolínea, etc.

### 7.3.2.4 Otros modelos

- Reservas: Simula una reserva hecha por el usuario sobre un vuelo disponible. Usando ID\_Usuario, Numero\_Vuelo, Fecha\_Hora\_Reserva, Tarifa\_Total.
- Vuelos, Aeropuertos, Aerolíneas, Tarifas: Modelos que forman parte del entorno simulado de Amadeus, que permiten consultar disponibilidad, precios, y condiciones.

## 7.3.3 Vista

En el patrón MVC utilizado, la vista representa la capa encargada de mostrar la información al usuario y gestionar la interacción con la interfaz. En este proyecto, las vistas están desarrolladas con tecnologías estándar del frontend: HTML5, CSS3 y JavaScript, sin utilizar frameworks adicionales como React, Angular o Vue, lo que permite un control más directo sobre el diseño y la lógica visual. A continuación, explicare algunas de las funcionalidades claves implementadas.

### 7.3.3.1 Interacción con el simulador Amadeus

- El usuario puede escribir comandos en una caja de texto simulando el entorno real de Amadeus.
- Se utiliza JavaScript para detectar los eventos de escritura (oninput) y enviar peticiones al backend mediante fetch() o XMLHttpRequest.
- El backend procesa el comando y devuelve una respuesta textual simulada, que se muestra al instante en la misma página.

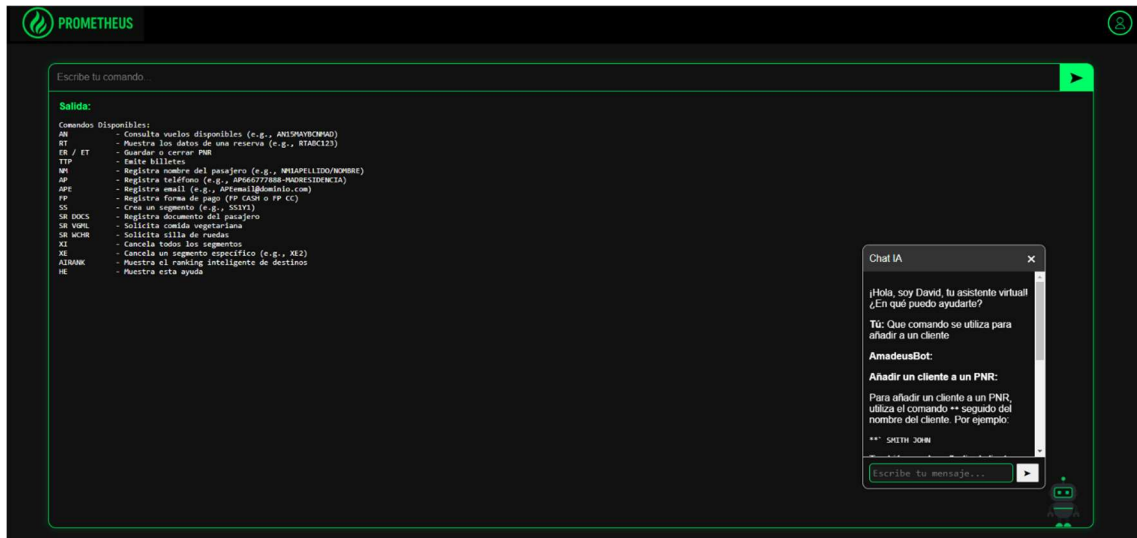


Figura 11: interacción con el simulador Amadeus

### 7.3.3.2 Autocompletado y sugerencias inteligentes

- A medida que el usuario escribe un comando, se lanza una petición a una vista que contiene un modelo de IA (CommandAutoCompleteModel), el cual devuelve una sugerencia en tiempo real.
- Esto se logra utilizando keyup en el input y mostrando la predicción en un dropdown o directamente en el mismo campo.

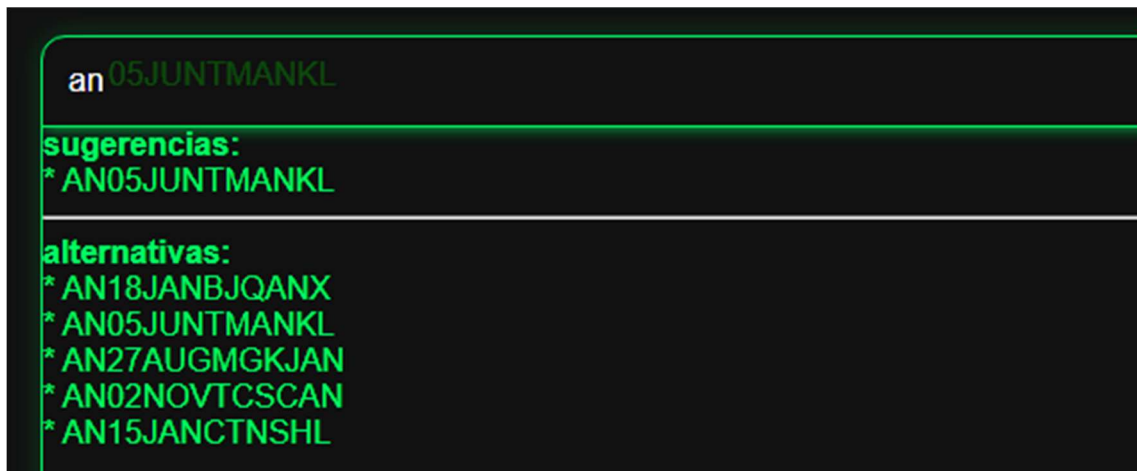


Figura 11: Autocompletado y sugerencias inteligentes

### 7.3.3.3 Chatbot inteligente

- El chatbot se representa como un icono flotante en la esquina inferior derecha de la pantalla.
- Al hacer clic, se despliega una ventana que el usuario puede mover libremente y redimensionar.

- Toda la comunicación se maneja vía JavaScript, que envía los mensajes del usuario y recibe las respuestas generadas por IA usando JSON y endpoints del backend (chat\_with\_amadeus).

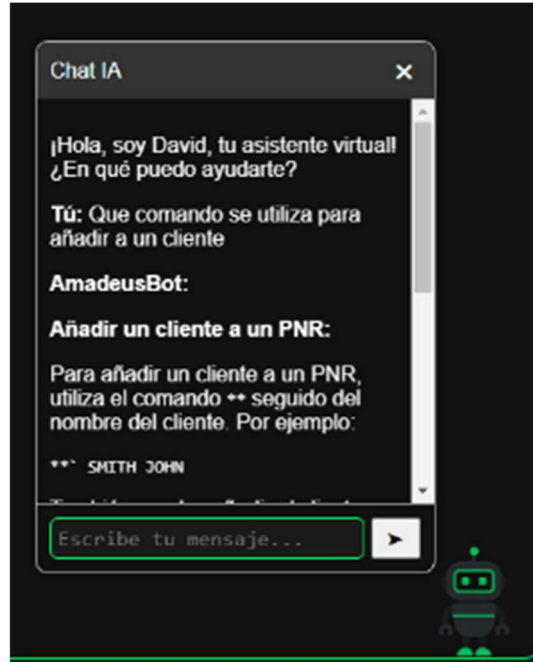


Figura 12: Chatbot inteligente

#### 7.3.3.4 Predicción de precios

- En la sección correspondiente, el usuario introduce datos de vuelos en un formulario.
- Al enviar el formulario, se realiza una petición POST al backend que utiliza el modelo FlyBot para predecir el precio del vuelo.
- La respuesta se muestra en pantalla con el precio estimado y posibles recomendaciones.

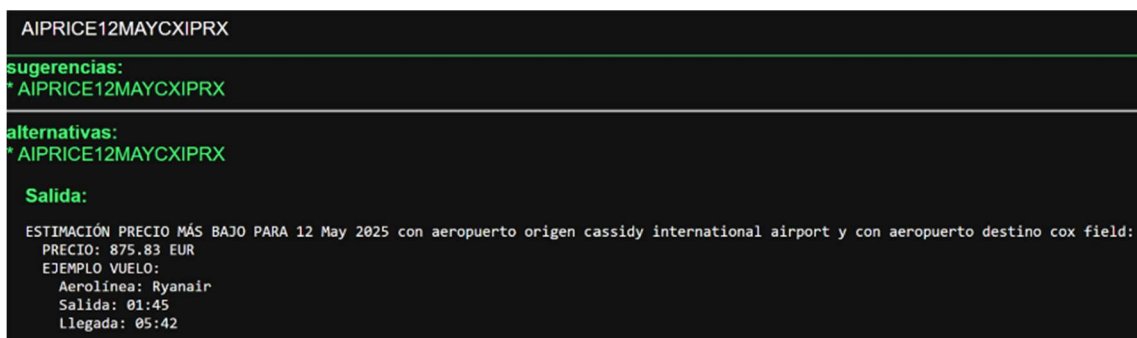


Figura 13: predicción de precios

#### 7.3.3.5 Estilo y diseño visual

- El diseño general es **oscuro (dark mode)**, con fondo negro y elementos en verde y blanco, buscando un estilo moderno y minimalista.
- El diseño es **responsive en parte**, pero prioriza el uso en escritorio, ya que para un móvil podría ser incomodo al tener una pantalla muy pequeña.
- Se aplican transiciones, animaciones sutiles y efectos hover para mejorar la estética sin comprometer el rendimiento.

Gracias a esta arquitectura, la interfaz ofrece una experiencia interactiva, fluida y visualmente clara, sin necesidad de recargar la página, lo cual es fundamental para simular una experiencia real de uso profesional.

#### 7.4 Análisis y conclusiones

El objetivo de mi proyecto era diseñar, desarrollar e implementar una página web funcional e interactiva, que actuara como simulador del entorno Amadeus, integrando inteligencia artificial para asistir al usuario, mejorar la experiencia de aprendizaje y optimizar los tiempos de consulta, aparte de estructura el proyecto de forma clara y mantenible.

Durante el desarrollo del proyecto se ha conseguido construir una aplicación web desde cero, en la que los usuarios pueden interactuar con un entorno que simula el uso de comandos Amadeus. A través de funcionalidades como el chatbot inteligente, el texto predictivo, la sugerencia de comandos y la predicción de precios de vuelos, se ha logrado acercar la experiencia real de trabajo con Amadeus a estudiantes o profesionales en formación.

El proyecto se ha estructurado utilizando el patrón de Modelo-Vista-Controlador (MVC), una técnica que me enseñaron tanto en el grado como en el curso de IA que finalice.

Además, se ha trabajado con técnicas de inteligencia artificial para entrenar modelos de predicción de precios (basado en LSTM) y autocompletado de comandos, integrándolos exitosamente en la estructura de la aplicación.

Aunque el resultado final ha sido bastante satisfactorio, la verdad es que no tengo ni la mitad de funcionalidades que tiene el software de Amadeus, al intentar estudiar su código, me di cuenta que es un código tan antiguo que se ha ido actualizando poco a poco, poniendo parches según lo fueran intentando, haciendo que su código actual en verdad no siga un orden claro y vaya cambiando según la acción que hace, si quisiera lograrlo me tendría que pasar meses estudiando a fondo su código y luego más meses para poder implementarlo en el back.

En conclusión, este TFG me ha enseñado como se tiene que encarar un proyecto cuando se empieza desde cero y también me permitió obtener una gran cantidad de

conocimiento del sector de la inteligencia artificial y lo dura que es, porque en las clases todo era más fácil ya que se tenía una base de datos limpia para hacer los ejercicios. Todo el esfuerzo que puse en este TFG termino con el resulta de una herramienta funcional, con potencial educativo y profesional.

## Referencias

- [1] Django. *The Web Framework for Perfectionists with Deadlines*. URL: <https://www.djangoproject.com/>
- [2] Python Software Foundation. *The Python Programming Language*. URL: <https://www.python.org/>
- [3] OpenAI. *API Documentation*. URL: <https://platform.openai.com/docs/>
- [4] Gemini API – Google AI. *Gemini API for Language Models*. URL: <https://ai.google.dev/gemini-api/docs>
- [5] Anaconda. *Anaconda Distribution for Scientific Computing*. URL: <https://www.anaconda.com/>
- [6] Pandas. *Python Data Analysis Library*. URL: <https://pandas.pydata.org/>
- [7] NumPy. *The Fundamental Package for Scientific Computing in Python*. URL: <https://numpy.org/>
- [8] Scikit-learn. *Machine Learning in Python*. URL: <https://scikit-learn.org/>
- [9] TensorFlow. *An End-to-End Open Source Machine Learning Platform*. URL: <https://www.tensorflow.org/>
- [10] HTML & CSS – W3C. *Web Standards*. URL: <https://www.w3.org/standards/webdesign/htmlcss>
- [11] JavaScript – MDN Web Docs. *JavaScript Guide*. URL: <https://developer.mozilla.org/en-US/docs/Web/JavaScript>
- [12] JSON. *JavaScript Object Notation*. URL: <https://www.json.org/json-en.html>
- [13] Pytest. *Python Testing Framework*. URL: <https://docs.pytest.org/>
- [14] Visual Studio Code. *Code Editing. Redefined*. URL: <https://code.visualstudio.com/>
- [15] Amadeus. *About Amadeus IT Group*. URL: <https://amadeus.com/en>
- [16] MVC. *Model–View–Controller*. URL: <https://en.wikipedia.org/wiki/Model–view–controller>

[17] Contrast Finder. *Herramienta de accesibilidad de contraste de color*. URL:  
<https://contrast-finder.tanaguru.com/>

[18] Wikipedia. *Artificial intelligence*. URL:  
[https://en.wikipedia.org/wiki/Artificial\\_intelligence](https://en.wikipedia.org/wiki/Artificial_intelligence)

Respositorio Github. *TFG* .URL:

<https://github.com/cesarces/TFG>