MEMORIA DE PRÁCTICA

Inteligencia Artificial (IA)

(Curso Académico 2025-2026)

MetroBuddy: Calculadora de Rutas Óptimas

Metro de Ciudad de México (CDMX)

Grupo de Trabajo: 2

Coordinador:
Gragera Serradilla, Alejandro (22M041)

Miembros del Equipo:
Liu, Yang (23M072)
López Nécula, Antonio Javier (23M041)
Ortega Villanueva, David (23M043)
Yu, Chuanhui (23M064)

Universidad Politécnica de Madrid
Escuela Técnica Superior de Ingenieros Informáticos
(Grado en Matemáticas e Informática)

“En cuanto algo funciona, nadie lo llama inteligencia artificial.”

— *John McCarthy*

Resumen

El proyecto *MetroBuddy CDMX*, tiene como objetivo el desarrollo de una aplicación con interfaz gráfica de usuario para el cálculo del trayecto *óptimo* entre dos estaciones del sistema de transporte colectivo *Metro de la Ciudad de México (CDMX)*. El núcleo de la aplicación reside en la implementación del algoritmo de búsqueda informada $A*$ (A-estrella), que permite encontrar la ruta con menor coste total, ponderando factores como la distancia física, el tiempo estimado de recorrido y la penalización por transbordos. La aplicación se ha desarrollado como una *aplicación web* responsiva, utilizando *JavaScript y Python* y la librería *Leaflet* para la visualización cartográfica. Se incluye una funcionalidad de modo accesible, gestión de favoritos y modo oscuro para mejorar la experiencia de usuario.

Abstract

The *MetroBuddy CDMX* project aims to develop an application with a graphical user interface for calculating the *optimal* route between two stations of the *Mexico City Metro* system. The core of the application lies in the implementation of the informed $A*$ (A-star) search algorithm, which allows finding the route with the lowest total cost by considering factors such as physical distance, estimated travel time, and transfer penalties. The application has been developed as a responsive *web application*, using *JavaScript and Python*, and employs the *Leaflet* library for cartographic visualisation. Additional features include an accessible mode, favourites management, and dark mode to enhance the user experience.

1 Introducción

1.1 Objetivo de la Práctica

El objetivo principal es aplicar los conocimientos de *Inteligencia Artificial* en el dominio de la búsqueda de caminos mediante la construcción de un sistema funcional capaz de determinar la ruta más eficiente a través de una red compleja de transporte. Específicamente, se busca implementar el algoritmo $A*$ para modelar y resolver el problema de la ruta óptima en la red del Metro de CDMX, cubriendo las siguientes líneas y tramos: Línea 1 (Observatorio a Balderas), Línea 3 (Universidad a Juárez), Línea 7 (Barranca del Muerto a Polanco), Línea 9 (Tacubaya a Lázaro Cárdenas) y Línea 12 (Mixcoac a Eje Central).

1.2 Alcance y Limitaciones

El proyecto se centra en la optimización del trayecto considerando las siguientes circunstancias de influencia, tal como se detalla en el enunciado:

- *Distancia/Tiempo:* El costo principal del trayecto.
- *Transbordos:* Se aplica una penalización temporal por transbordo.
- *Accesibilidad:* Se implementa un modo de búsqueda que puede priorizar rutas con estaciones accesibles, modificando los pesos de las aristas o excluyendo nodos no accesibles (elevadores, rampas, etc.).

La aplicación se desarrolla como una solución web (HTML5, CSS3, JavaScript) para garantizar la portabilidad y facilidad de ejecución.

2 Marco Teórico: Algoritmo A*

El algoritmo **A*** es un algoritmo de búsqueda de grafos que encuentra el camino con el costo más bajo desde un nodo de inicio hasta un nodo objetivo. Es una extensión de Dijkstra que utiliza una función heurística para guiar su búsqueda.

2.1 Función de Evaluación $f(n)$

La fortaleza de A^* radica en su función de evaluación, $f(n)$, que estima el costo total del camino a través del nodo n.

$$f(n) = g(n) + h(n)$$

Donde:

- $g(n)$ *(Costo Real):* Es el costo acumulado conocido desde el nodo inicial hasta el nodo actual n. En el modelo del Metro, esto representa la suma de los costos de los tramos recorridos (distancia + tiempo + penalización por transbordo).
- $h(n)$ *(Costo Heurístico):* Es el costo estimado desde el nodo actual n hasta el nodo objetivo. Esta es la función heurística. Para que A^* garantice la optimalidad, $h(n)$ debe ser *admisible* (nunca sobreestimar el costo real hasta el objetivo) y preferiblemente *consistente*.

2.2 Modelado de la Heurística para el Metro

Dada la naturaleza geoespacial del problema, se opta por una heurística basada en la *distancia euclidiana (o de línea recta)* entre la estación actual n y la estación objetivo.

$$\mathbf{h(n)} = \text{Distancia Euclídea}(\mathbf{n}, \text{Objetivo}) \times \mathbf{k}_{\text{avg}}$$

Donde k_{avg} es un factor de conversión que relaciona la distancia física con el costo ponderado (tiempo/coste). La distancia euclídea entre las coordenadas GPS de dos estaciones es un valor que nunca sobreestima el costo real del viaje restante, ya que la ruta real del metro siempre será igual o mayor, garantizando así la *admisibilidad* y la optimalidad del algoritmo.

3 Diseño e Implementación de la Solución

3.1 Modelo de Grafo (Red del Metro)

La red del Metro de CDMX se modela como un *Grafo Dirigido Ponderado* $G = (V, E)$:

- *Vértices (V):* Cada estación del metro es un nodo con atributos (ID, Nombre, Coordenadas LAT/LON, Línea(s), Accesibilidad).
- *Aristas (E):* Los tramos entre estaciones adyacentes o las conexiones de transbordo son las aristas.
- *Ponderación (Costo):* El costo de una arista $C(u, v)$ se define como una función que combina tiempo, distancia y penalizaciones.

$$C(u, v) = \alpha \cdot \text{Distancia}(u, v) + \beta \cdot \text{Tiempo}(u, v) + \text{Penalización}_{\text{transbordo}}$$

La penalización por transbordo (ej. 180 segundos o 3 minutos, como se ve en el código) se añade al costo de la arista que inicia el nuevo segmento de línea, garantizando que el algoritmo penalice activamente el número de cambios de línea.

3.2 Arquitectura de la Aplicación

La aplicación *MetroBuddy* se implementó utilizando una arquitectura web simple (cliente-servidor a nivel de ficheros estáticos).

- *Frontend (`index.html`):* Estructura HTML que define la interfaz de usuario.

- *Estilos (`style.css`):* Definición de la estética, incluyendo los estilos del Modo Oscuro.
- *Motor y Lógica (`script.js`):* Este archivo contiene la *lógica completa del algoritmo* A^*, la definición de la constante `estaciones` (datos del grafo), la gestión de eventos de la UI, el cálculo de la ruta final y la gestión de favoritos (`localStorage`).

La visualización se realiza a través de la librería *Leaflet*, donde la ruta óptima se dibuja como una `L.polyline` y las estaciones se marcan con animaciones secuenciales, proporcionando una UX atractiva.

3.3 Funcionalidad Clave del Frontend

El código JavaScript (`script.js` y `style.css`) revela varias funcionalidades avanzadas de la UI:

1. *Cálculo de Segmento y Transbordo:* El bucle de la función `showRoute` en `script.js` itera sobre los segmentos de línea de la ruta óptima para calcular el tiempo y la distancia total, y para identificar y mostrar el tiempo de transbordo (‘transferTimeMin‘), demostrando la integración de la lógica de costos en la UI.
2. *Modo Oscuro Dinámico:* La sección de `DARK MODE LOGIC` en `script.js` y los estilos asociados en `style.css` (ej. `body.dark-mode .leaflet-tile-pane { filter: invert(...) }`) permiten alternar entre modos, cuidando que los colores de las líneas del metro permanezcan identificables.
3. *Gestión de Favoritos:* Se utiliza el almacenamiento local (`localStorage`) para persistir las rutas frecuentes, con funciones `saveFavorite`, `deleteFavorite` y `loadFavorites`.
4. *Manejo de Errores y Carga:* Se incluye validación de entradas y una pantalla de carga (’loading’) para mejorar la retroalimentación al usuario durante la ejecución del algoritmo A^*.

3.4 Funciones Integradas y Diseño de la Aplicación

La aplicación *MetroBuddy* se ha diseñado siguiendo principios de claridad funcional y usabilidad (UX), garantizando que la potencia del algoritmo A^* se presente a través de una interfaz intuitiva y adaptable.

3.4.1. Interfaz Principal y Cálculo de Ruta Óptima

El diseño de la página se centra en la función principal de la práctica: la búsqueda del camino óptimo.

- **Diseño y Flujo Principal:** Al acceder a la página, la jerarquía visual establece el *mapa interactivo* de la red de metro (implementado con la librería Leaflet) como el elemento central. Este se complementa con un panel de control que permite al usuario especificar las estaciones de *Origen* y *Destino* mediante selectores de entrada.

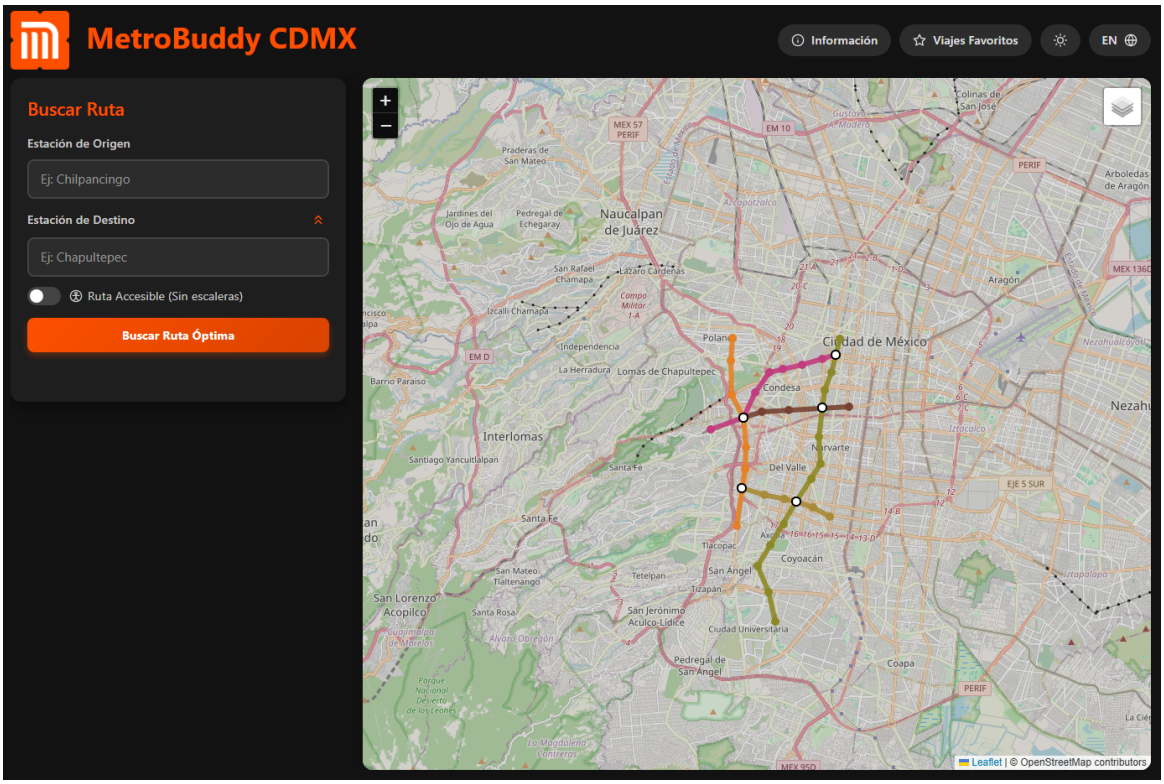

Figura 1: Visualización de la página web.

- **Visualización de Resultados:** Una vez ejecutada la búsqueda, el algoritmo A^* retorna la secuencia óptima de nodos. Esta ruta es visualizada inmediatamente en el mapa mediante una *polilínea destacada* que indica el trayecto completo. Los resultados clave del cálculo (duración total, número de transbordos y distancia total) se presentan de forma destacada (Figura 1).

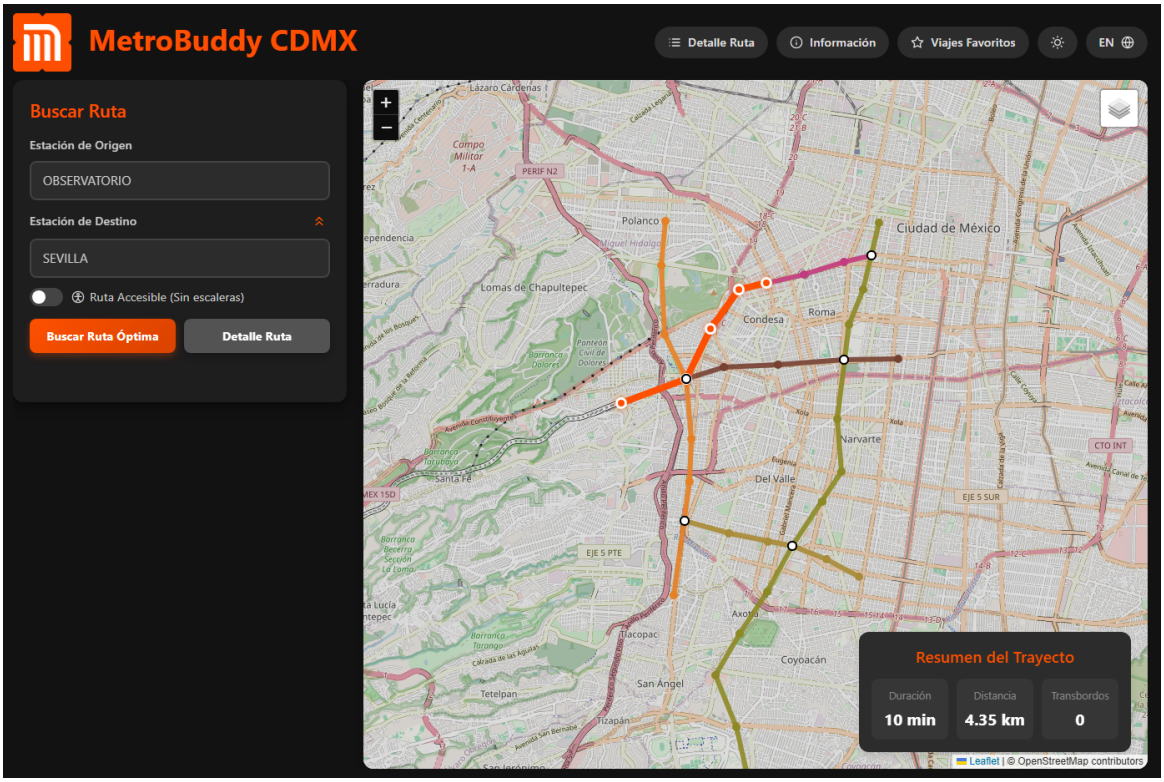

Figura 2: Ejemplo de uso estándar: Ruta óptima calculada por A* entre las estaciones seleccionadas.

3.4.2. Gestión de la Accesibilidad como Costo Variable

La aplicación incorpora una funcionalidad que demuestra la flexibilidad del modelado del grafo al introducir una restricción de coste dinámico:

- **Opción Accesible:** La casilla de *Opción Accesible* permite al usuario modificar el modelo de coste del grafo. Al activarla, el algoritmo A^* penaliza o excluye las aristas correspondientes a estaciones no accesibles, generando una ruta que minimiza la dificultad física del trayecto.

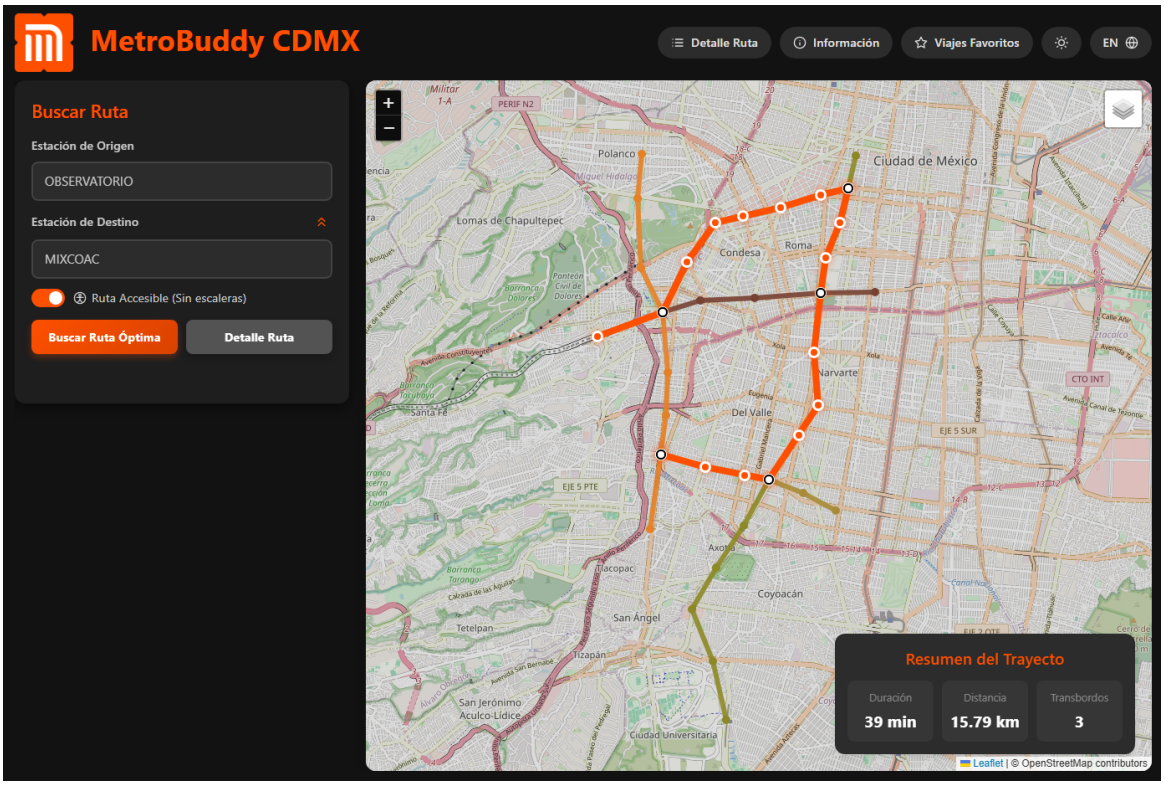

Figura 3: Ejemplo de uso con Opción Accesible activada.

- **Comparativa de Soluciones:** La diferencia entre el cálculo de ruta estándar (Figura 2) y el cálculo con la accesibilidad activada (Figura 3) ilustra cómo el algoritmo adapta su solución óptima a las restricciones impuestas, eligiendo un camino diferente que, aunque podría ser más largo, satisface el criterio de accesibilidad.

3.4.3. Análisis Detallado y Persistencia de Rutas

La aplicación extiende su utilidad con herramientas de análisis y persistencia:

- **Ruta Detallada:** Esta funcionalidad ofrece una descomposición exhaustiva del trayecto, esencial para la verificación. Enseña la distancia y duración de cada línea recorrida, e *indica con precisión el punto y la duración aproximada de cada transbordo.*

Figura 4: Vista detallada de la ruta óptima, mostrando el desglose por segmentos de línea, la duración y los puntos exactos de transbordo.

- **Gestión de Favoritos:** Con el objetivo de mejorar la experiencia de usuario y evitar búsquedas repetitivas, el sistema permite *añadir la ruta calculada a favoritos* (Figura 4), utilizando el almacenamiento local (`localStorage`) para la persistencia.

3.4.4. Opciones de Utilidad y Configuración

Finalmente, se implementaron opciones de configuración para mejorar la adaptabilidad y el alcance de la aplicación:

- **Modo Oscuro:** Implementado con `style.css` y JavaScript, esta opción mejora la visibilidad y reduce la fatiga visual en entornos de baja luz.

- **Internacionalización y Ayuda:** Se incluye una opción para *cambiar el idioma* a inglés, facilitando su uso a turistas. Adicionalmente, el apartado de *información* sirve como guía introductoria y explicación concisa de las funcionalidades de la página a usuarios nuevos.

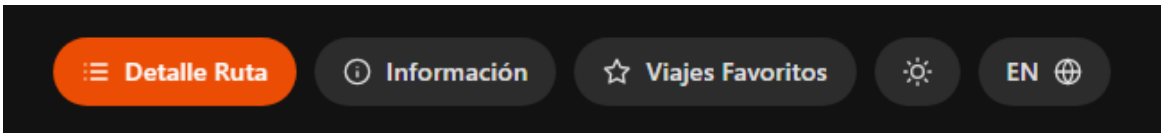

Figura 5: Panel de opciones adicionales.

4 Código Fuente e Instrucciones de Ejecución

4.1 Estructura del Repositorio

Todo el código fuente de la aplicación se encuentra alojado en un repositorio en línea. La estructura de ficheros propia del proyecto es la siguiente:

```
/MetroBuddy/
|-- index.html                                  (Estructura principal de la web)
|-- style.css                                   (Estilos visuales y Modo Oscuro)
|-- script.js                                   (Algoritmo A*, Datos del Grafo y Lógica de UI)
|-- PythonImplementacionAestrella_practica.ipynb (Implementación de referencia - Python)
|-- MapaMetroCDMX_practica.jpg (Mapa de referencia)
|-- Enunciado_practica.pdf
```

4.2 Enlace al Código Fuente

El código fuente de la práctica es accesible para su revisión y calificación a través del siguiente enlace:

Ver Repositorio en GitHub

4.3 Instrucciones de Ejecución

La aplicación ha sido desarrollada como una aplicación web sin dependencia de un servidor backend, lo que facilita su ejecución y acceso al estar desplegada en un dominio público.

1. *Acceso:* Abrir la URL proporcionada (`https://metrobuddy.kevian.xyz/`) en cualquier navegador web moderno (Chrome, Firefox, Edge, Safari).
2. *Uso:*
   - Introducir el nombre de la estación de inicio (`start`) y de destino (`end`).
   - (Opcional) Marcar la casilla *Ruta Accesible* modifica los pesos del grafo puesto que busca estaciones sin escaleras.
   - Presionar el botón *Buscar Ruta Óptima* para visualizar el camino óptimo en el mapa y el detalle de los pasos y transbordos con un resumen de trayecto indicando duración, distancia y número de transbordos.
   - (Tras ejecutar el programa) Presionar el botón *Detalle Ruta* nos muestra una visión más clara y sencilla de nuestra ruta. Gracias a esta función podemos conocer el tiempo estimado en cada línea, la duración del transbordo, etc.

5 Conclusiones

El desarrollo de la aplicación *MetroBuddy* ha permitido aplicar de forma práctica los fundamentos del algoritmo de búsqueda A^* en un problema real (red de metro), sirviendo como una valiosa experiencia tanto técnica como de colaboración.

5.1 Conclusiones Técnicas y Lecciones Aprendidas

5.1.1. Dificultades Técnicas Encontradas

Las dificultades se concentraron principalmente en el **modelado del grafo** y la **función de costo**:

- **Definición de Costos Híbridos:** El mayor desafío fue integrar la **distancia euclídea** (base de la heurística y parte de $g(n)$) con las penalizaciones discretas. Establecer un peso adecuado para el **transbordo** y para la condición de **accesibilidad** fue un proceso iterativo para asegurar que la ruta óptima resultante fuera coherente con la realidad del usuario.
- **Implementación del Algoritmo A^* en JavaScript:** Adaptar la lógica del algoritmo, desarrollada inicialmente como prototipo en Python (`PythonImplementacionAestrella_practica.ipynb`), al entorno JavaScript implicó asegurar el correcto funcionamiento de la **cola de prioridad** y la gestión eficiente del grafo.

5.1.2. Aspectos Técnicos Positivos

- **Validación del Algoritmo:** La principal lección técnica fue confirmar la **efectividad y eficiencia de A***. El algoritmo resolvió el camino óptimo en la red compleja del metro de forma casi instantánea.
- **Integración *Frontend* y Despliegue:** Se logró una integración completa del algoritmo con una interfaz gráfica (**Leaflet**), demostrando la capacidad de crear una solución completa y funcional usando únicamente tecnologías del lado del cliente, culminando con un **despliegue exitoso** de la aplicación (`https://metrobuddy.kevian.xyz/`).

5.2 Conclusiones de Gestión del Grupo

5.2.1. Gestión, Ambiente y Comunicación

El grupo mantuvo un **ambiente positivo y altamente colaborativo** durante toda la práctica.

- **Qué se ha hecho bien como grupo:** La **comunicación** fue excelente. La **repartición de tareas** fue clara (Modelado/Algoritmo, Adquisición de Datos, Desarrollo Web), y se mantuvo flexible para ayudar en áreas donde surgían bloqueos técnicos.
- **Habilidades Interpersonales:** Se demostró una alta capacidad para alcanzar acuerdos de forma rápida y consensuada, lo que se reflejó en la estabilidad y progresión constante del proyecto, sin entrar en desesperaciones ni discusiones.

5.2.2. Puntos de Mejora

- *Adquisición de Lenguajes Clave:* Se constató que la falta de fluidez previa en lenguajes como Python y JavaScript, que han sido herramientas fundamentales en este tipo de prácticas, elevó la curva de aprendizaje y aumentó el tiempo total dedicado al proyecto. De cara al futuro, se priorizará la dedicación de tiempo para conocer y dominar estos lenguajes por simple curiosidad y aplicación práctica, lo que sin duda reducirá los tiempos de desarrollo en tareas similares.