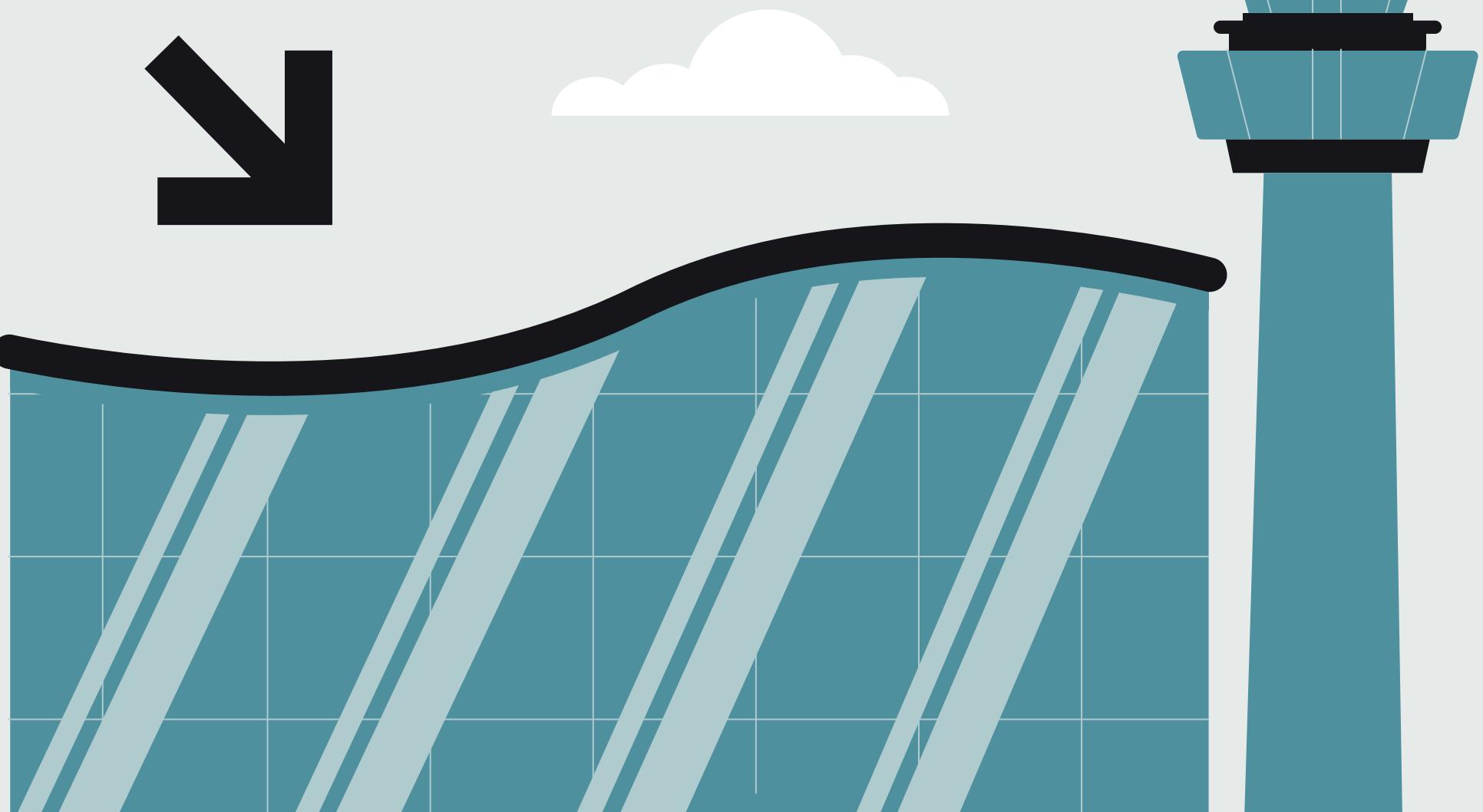
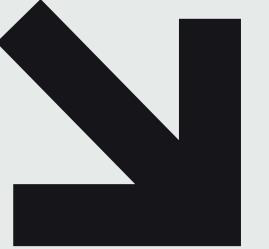




Clima del Aeropuerto

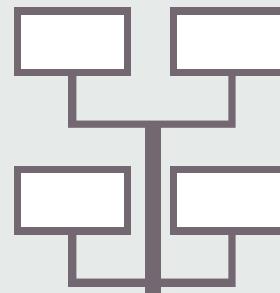
Web Service



Contenido

- 1. Definición del problema
 - a. Entendimiento del problema
 - b. Arsenal
 - c. Requisitos Funcionales
 - d. Requisitos No Funcionales
- 2. Análisis del problema
- 3. Selección de la mejora alternativa
- 4. Pseudocódigo
- 5. Costos y Mantenimiento





Introducción



Este proyecto es una página web que permite a turistas, sobrecargos y pilotos consultar el clima en tiempo real, ofreciendo información específica según el tipo de usuario.

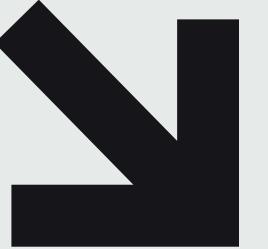


→ Definición Del Problema



Entendimiento del problema

El problema es la falta de acceso a información meteorológica personalizada para usuarios como turistas, sobrecargos y pilotos, quienes requieren datos específicos para planificar eficazmente.

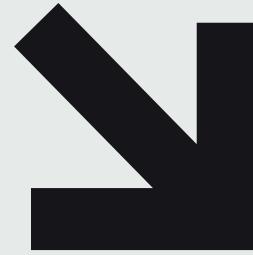


Arsenal (Herramientas y tecnologías utilizadas)



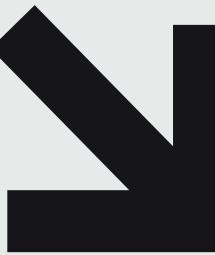
Flask (Python)

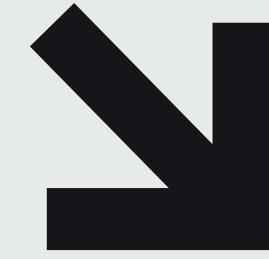
Framework ligero que permite el desarrollo de aplicaciones web rápidas y seguras.



HTML/CSS

Para la construcción y diseño de la interfaz gráfica de la aplicación.





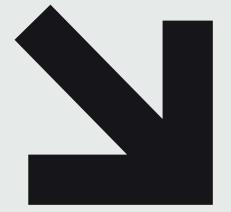
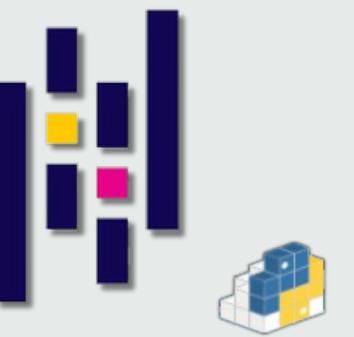
OpenWeatherMap API

Proveerá datos meteorológicos actualizados para las ciudades de interés, incluyendo temperatura, humedad, velocidad del viento, etc.



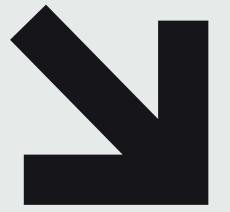
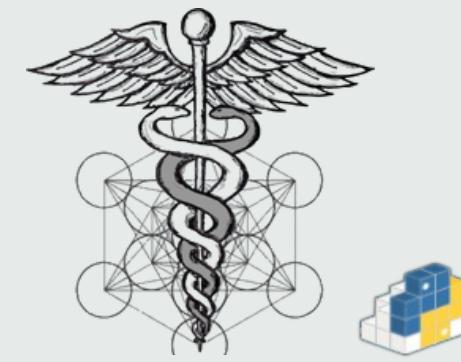
Python-Levenshtein

Para la corrección de errores tipográficos mediante algoritmos de similitud entre cadenas de texto.



Python-Pandas

Para manipulación de datos del csv



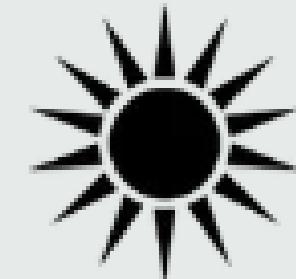
Python-Requests

Para realizar llamadas a las APIs y obtener datos externos.



Requisitos funcionales

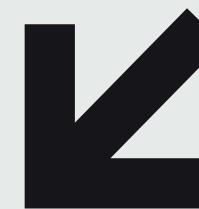
Debe permitir que el usuario ingrese el nombre de una ciudad o IATA de un aeropuerto y su tipo de usuario.



Debe corregir errores tipográficos en el nombre de las ciudades.



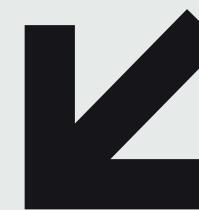
1



2

Debe mostrar el clima en tiempo real de la ciudad ingresada.

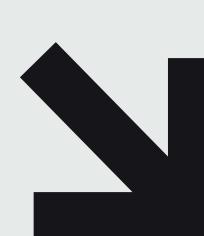
3



4

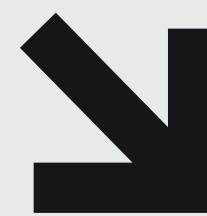
Debe proporcionar información meteorológica según el tipo de usuario:

- Turista: Temperatura y descripción.
- Sobrecargo: Temperatura, humedad y visibilidad.
- Piloto: Coordenadas, velocidad y dirección del viento, presión.

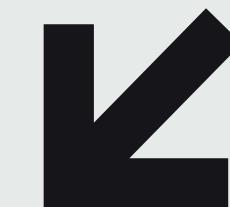


Requisitos No Funcionales

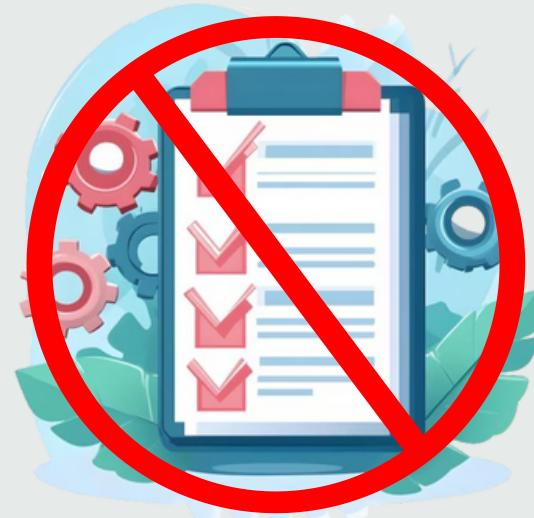
Fácil de usar para personas con pocos conocimientos técnicos.



1



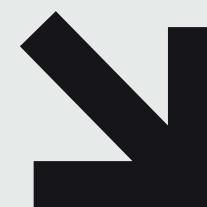
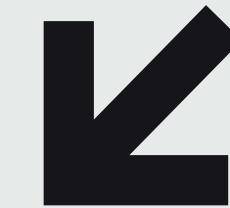
Cumplir con la Ley Federal de Protección de Datos Personales



2

Poder manejar múltiples usuarios y consultas simultáneamente.

3



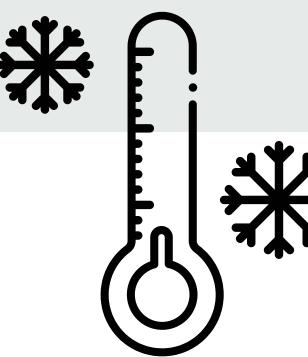
4

Las respuestas deben ser rápidas y en tiempo real, tanto para el clima como para la corrección de ciudades.



Analisis Del Problema

El problema es proporcionar el clima en tiempo real para distintos usuarios, corrigiendo errores en los nombres de ciudades y asegurando que la aplicación escale eficientemente.





Datos Dinámicos

La aplicación OpenWeatherMap debe consultar en tiempo real, requiriendo un manejo eficiente de APIs e integración sólida.





Corrección de errores tipográficos

Para corregir nombres de ciudades ingresados incorrectamente, se usa un algoritmo utilizando Levenshtein que ajusta errores tipográficos comparando con los datos de ciudades e IATA disponibles en archivos JSON.

MTY



Monterrey

Motery

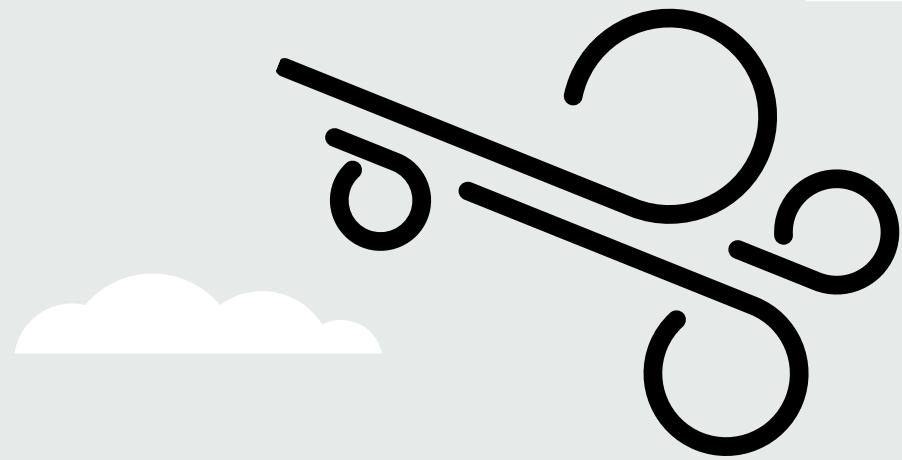


Monterrey



Información específica para cada usuario

Los usuarios requieren distinta información así como de una contraseña para acceder a la información, por ejemplo: los turistas solo necesitan temperatura y descripción, mientras que los pilotos requieren detalles técnicos como velocidad del viento y presión.





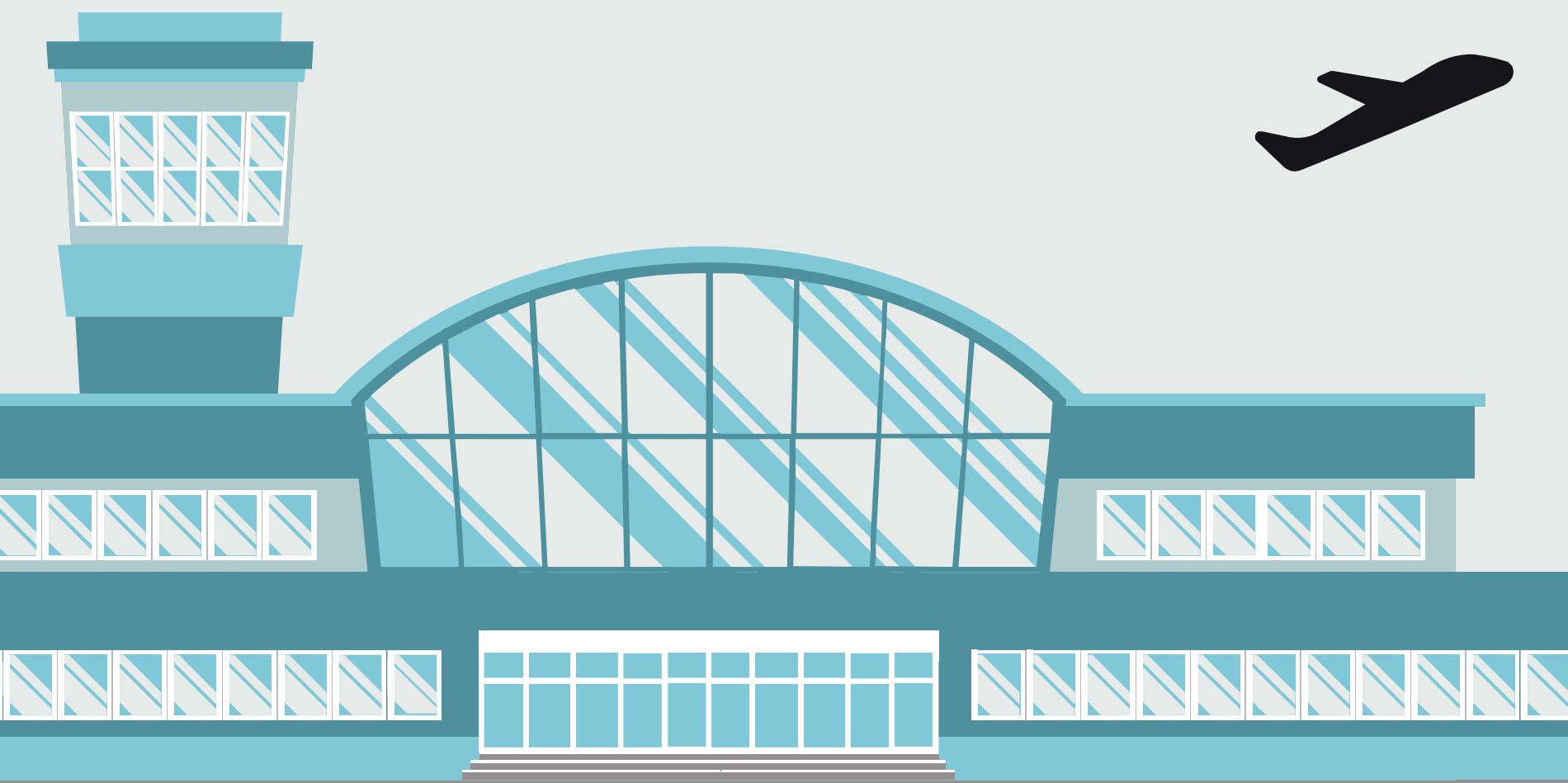
Seguridad y privacidad de los datos



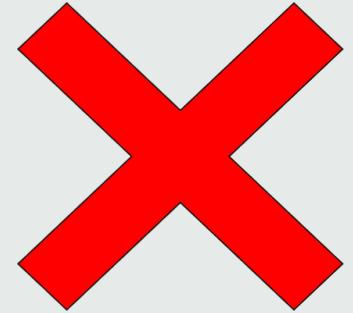
Es crucial cumplir con las normativas de protección de datos, ya que los usuarios ingresan información personal, aunque mínima.



Selección de mejor alternativa



Para abordar el problema, se consideraron varias alternativas antes de seleccionar la solución actual:



- Uso de bases de datos locales para almacenar datos meteorológicos.
- Utilizar una sola interfaz para todos los tipos de usuarios.
- Corrección de los nombres de las ciudades manualmente.



- Se utilizo un algoritmo utilizando Levenshtein para automatizar la corrección de los nombres de las ciudades.
- Personalización de la interfaz según el tipo de usuario
- Se implementaron la creación de Tickets
- Uso de contraseñas para cada tipo de usuario
- OpenWeatherMap fue elegido como API por su robustez y fácil integración.

Pseudocodigo:

El pseudocodigo completo de este programa se puede encontrar en el manual correspondiente de esta aplicacion.

1. Punto de entrada (run.py)

El archivo run.py es el punto de entrada principal de la aplicación. Aquí se define el proceso que inicializa y ejecuta el servidor de Flask.



2. Inicialización de la aplicación (`__init__.py`)

Este archivo inicializa la aplicación Flask y define la estructura general.



Flask

3. Rutas de la aplicación (routes.py)

El archivo run.py es el punto de entrada principal de la aplicación. Aquí se define el proceso que inicializa y ejecuta el servidor de Flask.



4. Servicios de Clima (weather_service.py)

El archivo
`weather_service.py`
contiene la lógica para
interactuar con la API de
clima y formatear los
datos según el tipo de
usuario



5. Validación y corrección de entradas (Robust_Entry.py y city_helpers.py)

Este módulo maneja la validación de las ciudades ingresadas por los usuarios.



6. Obtención de vuelos (Available_Flights.py)

Este módulo recupera vuelos disponibles desde un archivo CSV según el origen o destino.



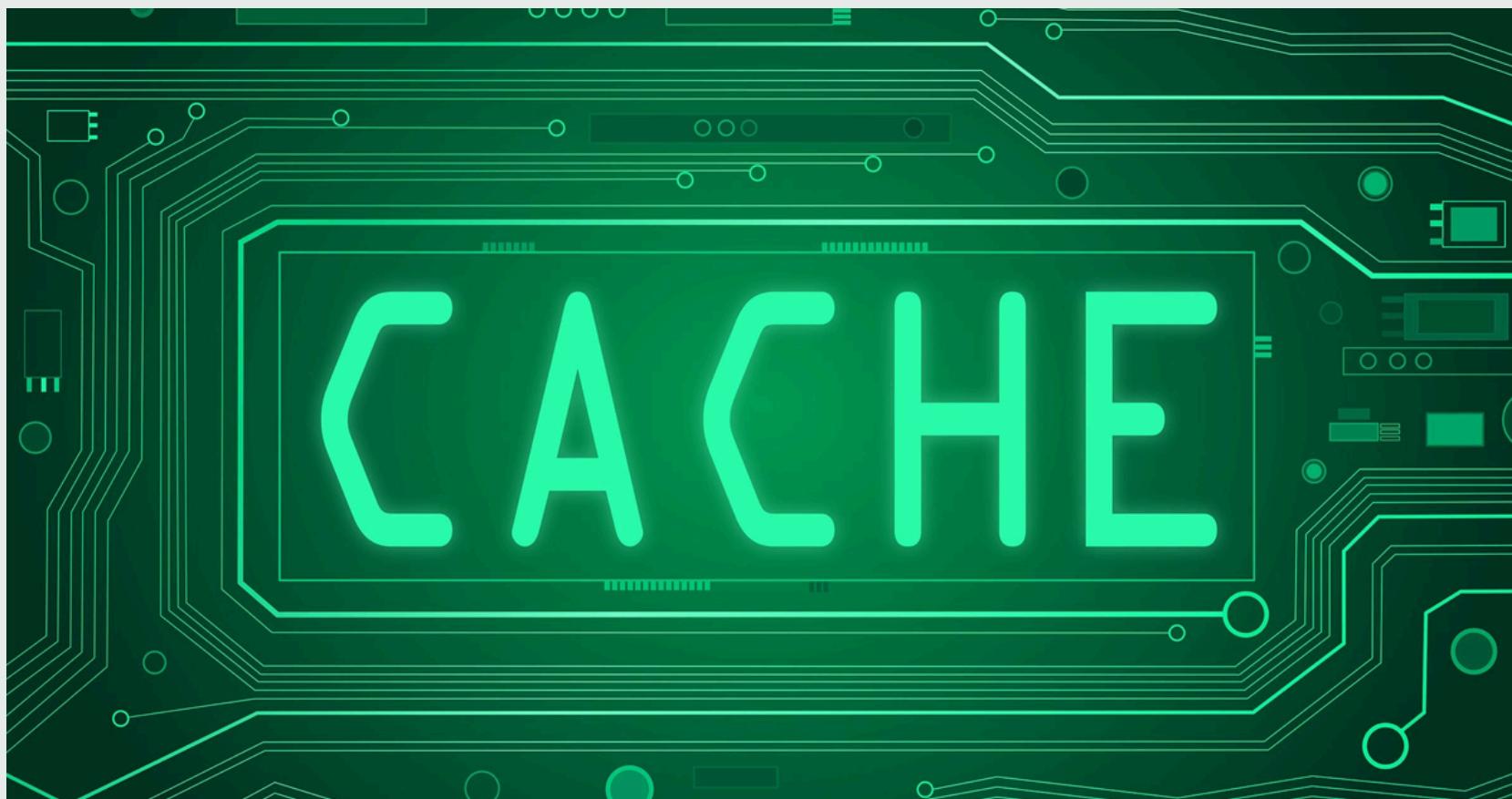
7. Generación y búsqueda de tickets (ticket_helpers.py)

Estas funciones permiten generar y buscar tickets en un archivo CSV.



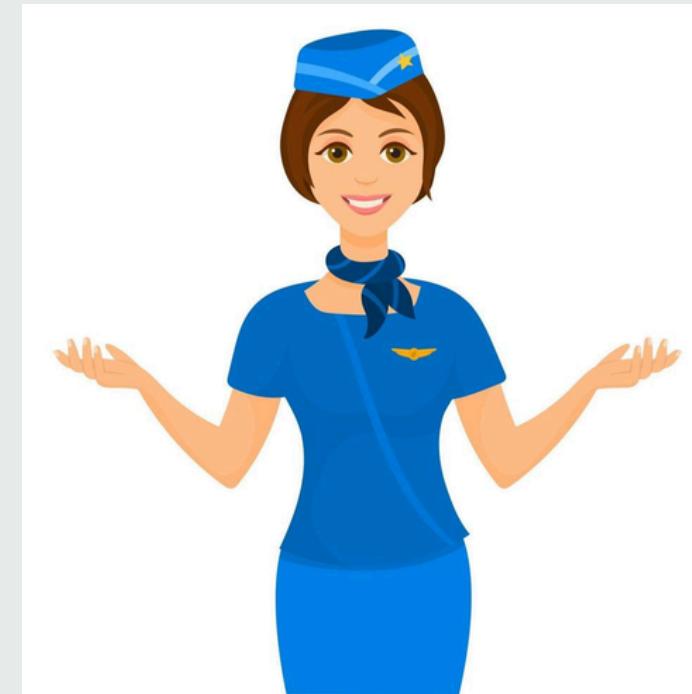
8. Caché (cache_helpers.py)

Este módulo permite almacenar en caché los datos del clima por un periodo de tiempo para mejorar el rendimiento.



9. Formateo de datos meteorológicos (`data_formatter.py`)

Aquí se define cómo se formatean los datos del clima según el tipo de usuario.



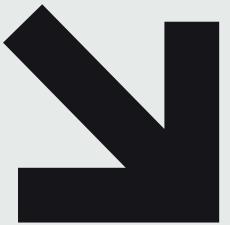
10. Validación de formularios (form_helpers.py)

Este módulo se encarga de extraer y validar los datos enviados por los usuarios a través de los formularios en el frontend.

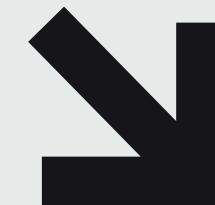


\$30,815.00

Costo del Proyecto



Mantenimiento



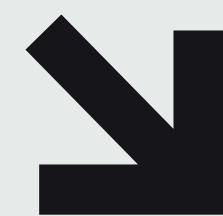
Mantenimiento de la API y datos

Verificar regularmente que la API, OpenWeather esté funcionando correctamente y que no haya cambios en sus endpoints o límites de uso. Si hay actualizaciones en la API, ajustaremos el código para que se mantenga compatible.

Verificar que el límite de solicitudes no se esté excediendo y ajustar la lógica para evitar bloqueos en caso de que se llegue al límite.

Asegurar que las imágenes relacionadas con el clima se estén mostrando correctamente y que su formato no haya cambiado.

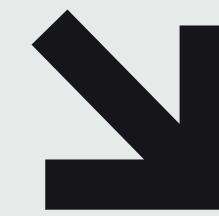
Mantenimiento



Gestión de roles y autenticación

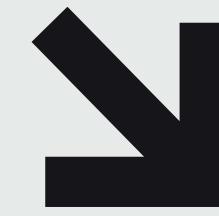
Asegurar que los roles de piloto y sobrecargo puedan acceder a los resultados sin problemas, mediante la verificación correcta de la contraseña.

Revisar el almacenamiento de contraseñas y considerar el uso de hashing para mayor seguridad.



Optimización de la Interfaz y Diseño

Mantener la interfaz actualizada, asegurando que los datos del clima y la imagen relacionada se visualicen correctamente.



Actualización del Contenido

En función de las temperaturas, revisar y actualizar el contenido periódicamente para que sea relevante, esto en tanto a las recomendaciones.

Costo por mantenimiento

- Mantenimiento preventivo o menores ajustes: Tarifa mensual fija, por ejemplo, entre \$1,550 y \$4,300.
- Mantenimiento correctivo o mejoras mayores: Por horas trabajadas, entre \$500 y \$960, dependiendo de la magnitud de las tareas.

