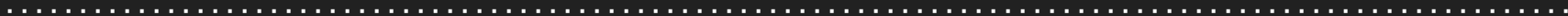


VuelaVuela

Informe de Clima para Aeropuertos



El problema

En el mundo de la aviación, miles de personas se desplazan de un aeropuerto a otro a diario, enfrentando cambios drásticos en zonas horarias y condiciones climáticas. El AICM ha identificado una necesidad crítica: proporcionar información actualizada sobre el clima en los aeropuertos de origen y destino para los vuelos que parten el mismo día en que se realiza la consulta.

.....



Requisitos

El desafío radica en desarrollar una aplicación que sea capaz de mostrar de manera interactiva, intuitiva y amigable la información climática relevante para sobrecargos, pilotos y clientes promedio, sin requerir conocimientos de programación por parte del usuario.

Nuestra solución debe basarse en tecnología web y aprovechar servicios web abiertos, para obtener datos climáticos precisos. Además, debemos asegurarnos de que la aplicación sea capaz de manejar errores, realizar pruebas exhaustivas y considerar políticas de uso de API para evitar bloqueos.

.....

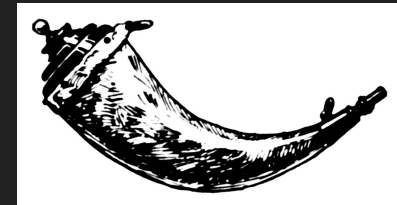
Decidimos basar nuestra solución en **tecnología web** para que sea accesible desde cualquier dispositivo con acceso a Internet. Esto requería el uso de un **framework web** y **servicios web** para obtener datos climáticos. Optamos por utilizar el servicio **Open-Meteo** debido a su disponibilidad, calidad de datos y acceso gratis e ilimitado. Esto nos permitió acceder a información precisa y actualizada sobre el clima en todo el mundo.

.....

Reconocimos la necesidad de una base de datos para almacenar información sobre aeropuertos y vuelos. Esto nos llevó a utilizar SQLite y a diseñar una estructura de base de datos eficiente. Dado que la aplicación se dirige a un público no técnico, nos esforzamos en diseñar una interfaz de usuario intuitiva que permita a los usuarios buscar información climática de manera sencilla y rápida.

Un aspecto clave fue la elección del framework web adecuado. Después de una evaluación exhaustiva, optamos por Flask debido a:

- Flask es un “micro-framework” conocido por su simplicidad y ligereza.
- Flask cuenta con una comunidad activa y una amplia base de usuarios. Esto significa que podemos acceder a una gran cantidad de recursos, bibliotecas y soluciones existentes.
- Flask es usado y probado por grandes compañías para sus productos como Netflix, Uber, Pinterest y Airbnb.



.....

Estructura

En este archivo nos comunicamos con el API de Open-meteo para obtener la información meteorológica de acuerdo a los parametros longitude y latitude.

```
# caché de tamaño maxsize con expiración de una hora  
@lru_cache(maxsize=maxsize)
```

```
def get_forecast_cached(latitude, longitude, time_hash):
```

```
    # información para el API
```

```
    url = "https://api.open-meteo.com/v1/forecast"
```

```
    data = {"longitude": longitude, "latitude": latitude,  
           "Info": "temperature, humidity, precipitation"}
```

```
    # nos comunicamos con el API
```

```
    with urllib.request.urlopen(url + "?" + data) as response:
```

```
        forecast = request.read()
```

```
    # retornamos la info en como un hashmap a partir del JSON
```

```
    return json.loads(forecast)
```

En este archivo obtenemos la información respecto a los aeropuertos e inicializamos la base de datos.

```
# base de datos sqlite3 con columnas:
```

```
# iata_code, municipality, latitude, longitude  
def populate_airports():
```

```
    url = "https://ourairports-data.com/airports.csv"
```

```
    # obtenemos una conexión a la base de datos  
    db = get_db()
```

```
    with urllib.request.urlopen(url) as response:
```

```
        reader = csv.DictReader(response)
```

```
        for row in reader:
```

```
            # insertamos los valores row["iata"], row["city"],
```

```
            # row["latitude"] y row["longitude"] como una
```

```
            # fila a la base de datos por cada fila del csv
```

El punto de
entrada de la
aplicación en
__init__.py

```
# usando las convenciones de Flask
```

```
def create_app():
```

```
    app = Flask(__name__)      # objeto principal
```

```
    # configuramos la app con la dirección de la base de datos con archivo o manual
```

```
    db.init_app(app) # registramos la base de datos con la app
```

```
    @app.route("/")
```

```
    def index():
```

```
        return render_template("index.html") # landing page donde tomamos input
```

```
    @app.route("/view")
```

```
    def view():
```

```
        # obtenemos los valores de ambos aeropuertos
```

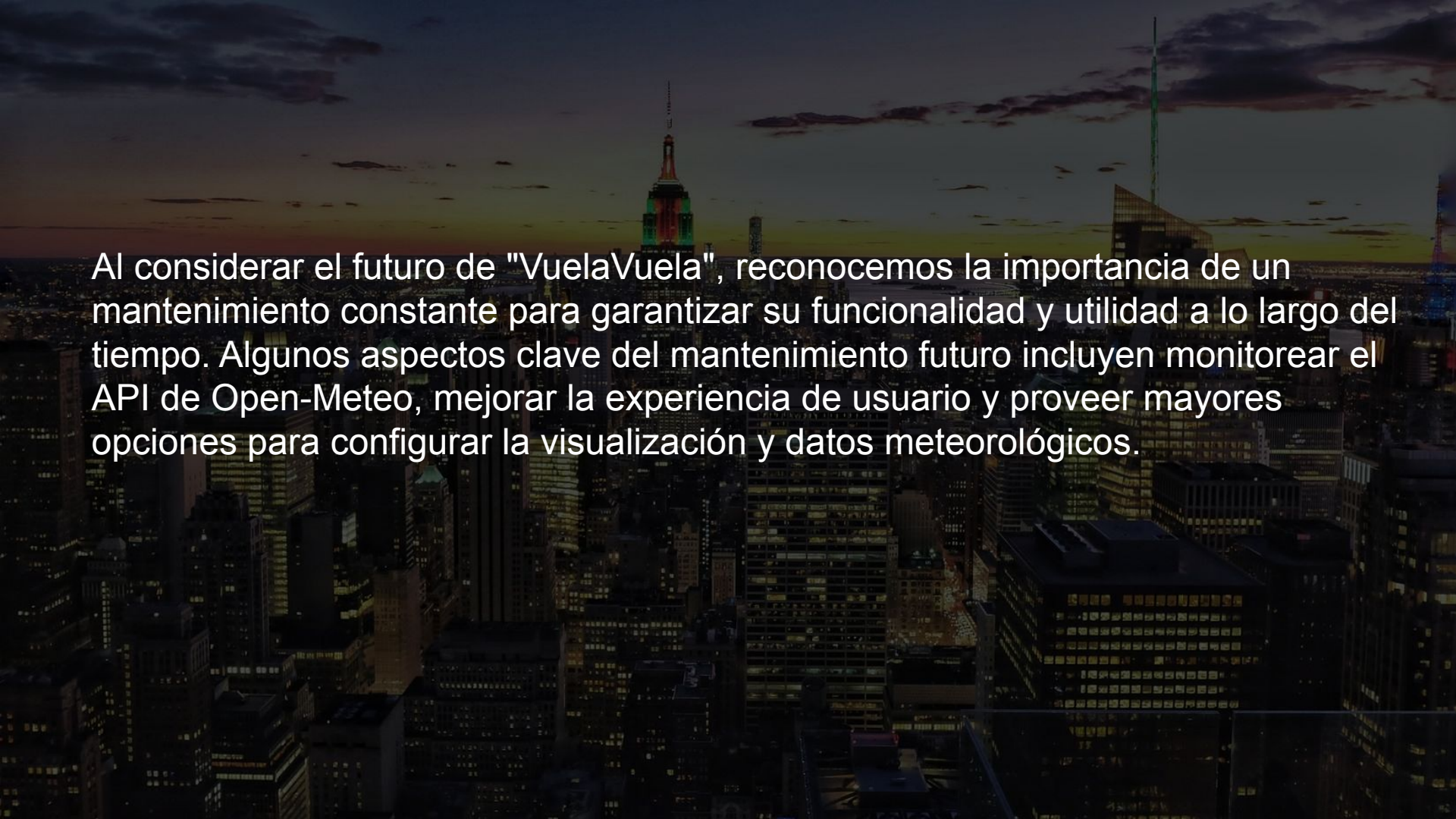
```
        origin, dest = request.form["origin"], request.form["dest"]
```

```
        # obtenemos las coordenadas
```

```
        origin_loc, dest_loc = db.get_info(origin), db.get_info(dest)
```

```
        forigin, fdest = api.get_forecast(origin_loc), api.get_forecast(dest_loc)
```

```
        return render_template("view.html", origin=forigin, dest=fdest)
```


An aerial photograph of a city skyline at dusk or dawn. The sky is a mix of dark purple, blue, and orange. The city is densely packed with skyscrapers, many of which are illuminated with lights. The Empire State Building is prominent in the center, with its spire reaching towards the sky. Other tall buildings are visible on the right side of the frame. The overall scene is a high-angle view of a major metropolitan area.

Al considerar el futuro de "VuelaVuela", reconocemos la importancia de un mantenimiento constante para garantizar su funcionalidad y utilidad a lo largo del tiempo. Algunos aspectos clave del mantenimiento futuro incluyen monitorear el API de Open-Meteo, mejorar la experiencia de usuario y proveer mayores opciones para configurar la visualización y datos meteorológicos.

Será necesario seguir de cerca las actualizaciones y cambios del API de Open-Meteo para asegurarnos que el producto siempre esté al día además de implementar un sistema de alerta para monitorear los cambios y bajas y poder responder de forma rápida y eficiente.

Una mejora continua basada en retroalimentación del usuario es imperativa para nuestra empresa. Nuestra estrategia a futuro involucra seguir recolectando puntos a mejorar por parte de los usuarios y sacar actualizaciones constantes de UX que sólo mejoren la usabilidad sin romper las expectativas de los usuarios.



Finalmente, tenemos muchas extensiones planeadas que nos permitan cumplir de mejor forma las necesidades individuales del usuario como es poder obtener información más o menos detalla del clima, poder pedir información climática de más aeropuertos (para viajes con escalas por ejemplo) y mostrar visualizaciones de los aeropuertos incluyendo información útil para el usuario.

Estas estrategias de mantenimiento futuro aseguran que "VuelaVuela" siga siendo una inversión sólida y que sigamos ofreciendo una experiencia de usuario de alta calidad.
