Universidad Nacional Autónoma de México

Facultad de Ciencias

Modelado y Programación

Web Service

Integrantes de equipo:
Mario Letepichia Romero
Celic Aislinn Liahut Ley
Ivette González Mancera

March 17, 2022

Definición del problema

Recibimos 3 mil tickets como entrada; cada uno contiene los siguiente datos: ciudad de origen y ciudad de llegada. Nuestra encomienda consiste en consultar el clima actual de ambas ciudades para cada ticket.

Análisis del problema

Específicamente en este caso como entrada tenemos un csv file , se debe encontrar la manera de extraer la información para aplicar el algoritmo que generara la salida.Para cada ticket primero se debe checar si ya se proceso esa información que ya se genero previamente para no guardarlo de nuevo, si no se ha generado previamente, entonces se hace el proceso para generar el clima de tal ciudad. Para lo anterior mencionado el programa debe conectarse a un Web service, para este proyecto se implementó con OpenWeather para realizar las peticiones al web service de la API de OpenWeather y de esa manera obtener la información que nos compete, es decir el clima actual de la ciudad del ticket. Posteriormente accedemos al clima y se tendra que ir almacenando la información de cada ticket para al final haber almacenado los 3 mil tickets y entregar la salida que debera contener el clima actual de llegada como también la de salida.

Selección de la mejor alternativa

Después de una ardua investigación llegamos a la conclusión de que la mejor opción es implementar la solución a través de python, esto por la accesibilidad a librerías como *Pandas* que nos permite procesar bases de datos de una manera muy sencilla. Resuelto este problema, el resto consiste en implementar algunos métodos y clases para el correcto procesamiento de los datos.

Mantenimiento del programa

Un problema que puede surgir con esta implementación es el consumo de memoria ya que el uso de Pandas biblioteca utilizada en este proyecto puede generar problemas en el espacio de la memoria.

Cuando se trabaja con datos a pequeña escala (menos de 100 MB) utilizando pandas, el rendimiento generalmente no es un problema. Cuando se enfrentan a datos más grandes (de 100 MB a varios GB), los problemas de rendimiento pueden prolongar el tiempo de ejecución y quedarse sin memoria debido a la memoria insuficiente.

Costo del programa

Una cuestión es para que haya una conexión a OpenWeather hay una versión gratuita, sin embargo tiene una restricción de llamadas por minuto específicamente 60, sin embargo eso es algo bajo para un aeropuerto , por lo que si se aplica este programa a un aeropuerto real se necesitara contratar un plan mas grande que tiene un costo dependiendo la cantidad de llamadas que se ofrecen por mes y por minuto. Los precios varían desde si es un aeropuerto muy pequeño el servicio puede ser gratis y si es una empresa gigante podría llegar a costar 40,729.18 peso mexicanos. Para ahorrarse llamadas al API, y estas se reduzcan se construyo un cache que lo que hace es no hacer llamadas al API repetidas, es decir obtener la información de un ticket con las mismas latitudes , lo cual mejora prolonga el mantenimiento.

Suponiendo que elegimos el plan de 3,000 llamadas por minuto y 100,000,000 por mes cuesta 3,801 pesos mexicanos. Lo que nuestro equipo cobraría por este programa son 200 dolares considerando el trabajo hecho, mas lo que costaría un plan viable de OpenWeather que es el plan de 3,801 pesos.

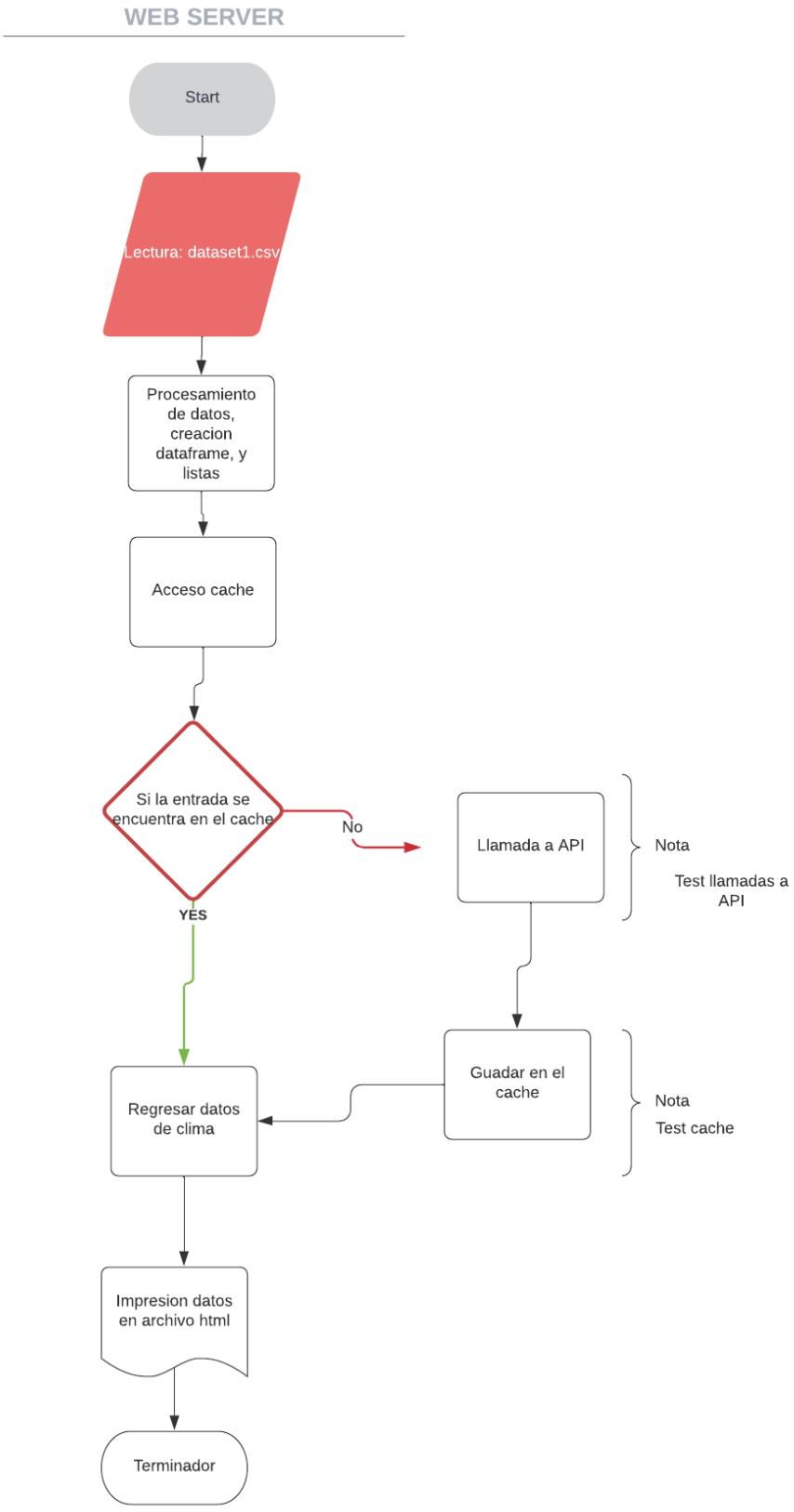

Figure 1: Diagrama de flujo

Justificaciones

La paquetería pandas se escogió porque es una herramienta que ayuda a trabajar con datos tabulares, como datos almacenados en hojas de cálculo o bases de datos, sus métodos son muy eficientes. Ademas permite la manipulación de archivos JSON como dataframe para un procesamiento limpio de datos.

Se utilizo OpenEwather como Web Server porque se considero que tiene una gran información y es utlizada por muchas empresas y esta muy bien documentado. Se utilizo también el modulo request para poder hacer las llamadas al API. Se hizo uso de la paqueteria **unittest** para realizar las respectivas pruebas unitarias de cada modulo.

Fuentes consultadas:

- API
    1. Pandas read-csv method
    2. API's en Python
    3. OpenWeather API documentation
    4. Documentation for dotenv
- Videos consutados:
    1. Python CSV files with Pandas
    2. Pandas valuecounts
    3. Pandas dataframes: Saving/Writing as CSV
    4. CSV column to Python List
    5. Hide API keys using python-dotenv
- Paqueterias usadas:
    1. Pandas
    2. Miniconda
- Configuracion VS
    1. Configurar VSCode para trabajar con Pandas
- Páginas consultadas:
    1. Exceptions throw by Pandas
    2. Read CSV files with python function
    3. Pandas docstring guide
    4. Documenting Python Code
- Pruebas Unitarias
    1. Python testing
    2. Unittest documentation