Universidad Nacional Autónoma de México

Facultad de Ciencias

Modelado y Programación

Proyecto 01

Alumnos:

Main Cerezo Asahel Said

Reyes López Eduardo Alfonso

Explicación del proyecto

1. Definición del problema

Se nos solicita entregar el informe del clima de la ciudad de origen y de llegada para tres mil tickets que salen el mismo día en el que corremos nuestro algoritmo

a) **Entender el problema:**

- ¿Qué queremos obtener?

  La información del clima para las ciudades de origen y llegada de los tres mil tickets que se nos proporcionan

- ¿Cuáles son los datos que tenemos para obtenerlo?

  Contamos con un archivo csv en el cual se encuentran:

  - Código IATA de cada ciudad de origen y de destino
  - Latitud y longitud de cada ciudad de origen y de destino

- ¿Qué hace que el resultado obtenido resuelva el problema?, ¿cuál es la característica que hace de un resultado, una solución?

  El resultado esperado por el programa es la información del clima de cada ciudad de origen y destino en el momento en el que se ejecuta el algoritmo, que es justamente lo que se nos solicita.

- ¿Qué operaciones o construcciones se deben obtener para llegar a la solución?

  a) Comunicarnos con la base de datos (csv)

b) Extraer los datos

c) Consultar el caché. Si existe, mostrarlo, si no:

d) Crear la petición

e) Hacer el request

f) Procesar la salida

g) Guardar el caché. Mostrar o recuperar la información

b) **Requisitos funcionales:**

La información de cada ticket debe ser almacenada en una estructura de datos. Para cada ciudad se debe hacer un request para consultar la información actual del clima. Se tiene que hacer un caché para evitar hacer requests innecesarias. La información del clima se guardará en un diccionario para posteriormente poder mostrarla al usuario.

Entradas:

- Tipo: matriz
- Formato: csv
- Tamaño: 3000
- Cantidad: 1

Salidas:

- Tipo: diccionario
- Tamaño: cantidad de ciudades distintas

c) **Requisitos no funcionales**

- Eficiencia. Al leer la base de datos y administrar la información de cada ticket. También se debe ser eficientes al momento de hacer las requests, para no hacer innecesarias ni saturar el servidor.
- Tolerancia a fallas. Pueden ocurrir fallas en varios puntos. Primero en la entrada al leer el CSV, al hacer el request a la API o con la entrada proporcionada por el usuario.
- Amigable con el usuario. Como el programa no busca ser interactivo por el usuario objetivo, se debe tener la cantidad justa d elementos para hacer la búsqueda y mostrar el informe.
- Seguridad. Que no se exponga la base de datos, y que no se use la llave particular del programador ligada a su cuenta.
- Escalabilidad. Capacidad para leer bases de datos más grandes o de otras caracterísiticas. Cambios en la ventana principal para manejo de más información.

d) **Arsenal**

- Paradigma de programación: Orientado a objetos
- Herramientas: WebService - OpenWeather. Para consultar el clima
- Lenguaje: Utilizaremos Python debido a las diversas bibliotecas con las que cuenta para poder procesar los datos
- Bibliotecas: request, csv, tkinter

2. Análisis del problema

Queremos obtener la información del clima de las ciudades que se nos solicitan en cada uno de los tickets, para ello tenemos que ocupar algún web service que proporcione información climática. Ejemplos de lo anterior son OpenWeather y Yahoo Weather.

Primero tenemos que hacer una lectura del archivo csv proporcionado, el cual contiene todos los tickets de los cuales queremos conocer información climática de su ciudad de origen y destino. La información del csv la guardamos en una estructura de datos para que sea más fácil de manipular posteriormente

Una vez leído el archivo csv surge un problema con el web service elegido, y es que la versión gratuita de estos servicios tiene un número limitado de llamadas a la API. Este número es menor a la cantidad de tickets a los que queremos obtener el clima de la ciudad de origen y destino. Para solucionar esto podemos usar un caché que guarde la información de las ciudades de las cuales ya se conoce el clima.

Finalmente tenemos que mostrar de alguna manera la información del clima de todas las ciudades solicitadas, ya sea en la terminal o usando una interfaz gráfica.

3. Selección de la mejor alternativa

Utilizamos Python dado que queremos tomar un paradigma orientado a objetos y además porque el lenguaje cuenta con diversas bibliotecas que hacen que sea más fácil leer archivos y hacer las llamadas a la API.

Para leer el archivo csv que guarda la información de todos los tickets optamos por usar la biblioteca csv, que cuenta con una función reader(), que regresa un objeto que permite iterar por las líneas del archivo. Posteriormente separamos cada línea en palabras y con ellas creamos objetos de tipo City, los cuales tienen como atributos el nombre, la latitud y la longitud de una ciudad. Después

de leer el archivo, nos quedamos con un diccionario en el que las llaves son las claves IATA y los valores son los objetos de tipo City

Para el web service optamos por usar OpenWeather por la completa documentación que tiene.

Creamos un método dentro de la clase City llamado getWeather. En él se hace el request con la respectiva latitud y longitud de la ciudad y se usa el método json() para extraer los datos, dichos datos se usan para crear un objeto de tipo WeatherInfo, que contiene toda la información climática que necesitamos mostrar. Finalmente se regresa dicho objeto.

Para el caché decidimos crear un diccionario en el que las llaves son los códigos IATA de las ciudades y los valores son objetos de tipo WeatherInfo. Antes de hacer un request a OpenWeather, revisamos si nosotros ya habíamos consultado el clima de esa ciudad previamente, si es que sí, entonces regresamos el valor asociado a esa llave en el diccionario, si es que no, entonces hacemos el request a OpenWeather y posteriormente actualizamos el diccionario con la nueva entrada.

Para pedir el ticket del que queremos conocer la información del clima y mostrar la información, decidimos hacer una interfaz gráfica ocupando la biblioteca tkinter. El usuario ingresa la clave IATA en la barra de búsqueda y al darle click a la lupa, se hace la request para esa ciudad y se muestra la información climática obtenida en pantalla

4. Pseudocódigo

Lectura de base de datos

Entrada: CSV

Salida: lista_tickets

1. Abrir el dataset1.csv
2. Guardar dataset1 como archivo de lectura en una variable "base"
3. Declarar una lista de ciudades

4. Por cada renglón en el archivo base:

5. Guardar los nombres de las ciudades de origen y destino. 6. Si la ciudad de origen no está en la lista de ciudades: 7. Crear una nueva ciudad con la información de ese renglón (origen,latitud origen,longitud origen)

8. Añadir la nueva ciudad a la lista_ciudades

9. Si la ciudad de destino no está en la lista de ciudades: 10. Crear una nueva ciudad con la información de ese renglón (origen,latitud destino,longitud destino)

11. Añadir la nueva ciudad a la lista_ciudades

12. Regresar lista_tickets

==========

Clase City

Atributos: origen, latitud, longitud

Algoritmo obtener llave

Requisitos: Archivo que contiene la llave de OpenWeatherMap key.csv

Salida: llave

1. Declaramos la variable "llave"
2. Encontramos la ruta hacia key.csv
3. Abrimos el arhivo key.csv
4. Leemos el contenido
5. Guardamos el contenido en llave
6. Regresamos la llave

Algoritmo obtener clima

Requisitos: llave, URL OpenWeatherMap, latitud y longitud

Salida: reporte_clima

1. Obtenemos la llave

2. Hacemos la petición a la API de OpenWeather con los parámetros de latitud y longitud

3. Recibir el archivo json y guardarlo

4. Acceder a cada dato de nuestro interés y guardarlo en una variable (temperatura, humedad, descripción, etc.)

6. Guardar esos datos en un objeto tipo información_clima

==========

Clase Información_clima

Atributos: descripción, temperatura, temperatura mínima, temperatura máxima, sensación térmica, húmedad.

Algoritmo obtener atributos como lista

Requisitos: Tener un objeto de tipo información_clima instanciado Salida: Lista de atributos 1. Crear la lista atributos 2. Anexar a la lista cada uno de los atributos 3. Devolver la lista

==========

Algoritmo para obtener la información del clima

Entrada: Objeto ciudad

Salida: Lista con la información del clima

1. Obtenemos el nombre de la ciudad

2. Si la ciudad está en el caché:

3. Obtenemos la información en esa entrada del caché

4. Pasamos a una lista esa información

5. Si no está:

6. Buscamos la información climática de la ciudad

7. Guardamos la información en la entrada del caché que tenga como llave el nombre de la

ciudad

8. Pasamos a una lista esa información

9. Regresamos la lista con la información climática

Algoritmo para buscar el informe del clima

Requisitos: Entrada de texto, lista de ciudades

1. Leer la ciudad como cadena de texto

2. Si la ciudad está en la base de datos:

3. Obetner la información del clima de esa ciudad

4. Llenar los respectivos campos de texto con la información

5. Si no está:

6. Indicar que la ciudad no está disponible.

5. Mantenimiento y requerimientos a futuro

- Poder reiniciar el caché que ya tenemos sin necesidad de salir de la aplicación o hacer que se actualice de manera periódica con la ejecución de la app.
- Mejorar la manera en la que se muestran los datos al usuario, en lugar de tener una barra de búsqueda en la que el usuario ingresa la clave IATA sería mejor tener un dropdown con todas las ciudades que estan disponibles, de esa forma sería más amigable para la persona usando la aplicación. De modo que al seleccionar la ciudad en particular, se muestre el informa del clima.
- Mostrar el clima de múltiples ciudades al mismo tiempo
- Mejorar la presentación del informe para tener disponibles más datos que podrían resultarles útiles al usuario.
- Poder acceder a información del clima de otros lugares o lugares cercanos al destino en caso de

que lo necesite el usuario.

Creemos que el proyecto vale ocho mil pesos mexicanos considerando el tiempo de desarrollo