



Proyecto 1. Clima.

1.- Definición del problema

Entendimiento del problema

Arsenal

Paradigma: Orientado a Objetos.

Lenguaje de programación utilizado: python

Requisitos funcionales (qué debe hacer el programa)

Entrada: Respuesta del usuario en la interfaz (un nombre de una ciudad).

Salida: Información del clima de la ciudad que proporcionó el usuario.

Por lo que con base en esto, el programa deberá procesar la información que obtiene del usuario desde la interfaz, teniendo en cuenta que para obtener el dato de entrada se requiere de un archivo de tipo csv para tener información de cada ciudad que ayudará a la obtención del clima.

Requisitos no funcionales (cómo debe comportarse el programa)

Eficiencia

Tenemos que la información que se recolecta del archivo csv, se eliminarán las ciudades que se vayan introduciendo a la estructura de datos una vez que ya se haya introducido los datos de la ciudad anteriormente, lo que evita que sea vuelva ineficiente y poco viable para una consulta del clima.

A su vez tenemos que al revisar en caché, se pueden lograr disminuir la cantidad de solicitudes que se puedan realizar en cierto periodo de tiempo.

Tolerancia a fallas.

Para ello se deberán implementar pruebas unitarias, de componentes, de integración y de sistema.

Amigabilidad

Para ello se deberá contar con una interfaz intuitiva y fácil de usar para el usuario.

Escalabilidad

Para este apartado, se espera que nuestras clases estén bien documentadas y que hagan un buen uso de la encapsulación para evitar futuros errores al manipular las clases ya existentes.

Seguridad

Se espera que información sensible como la llave de acceso a OpenWeather no se incluya en el proyecto, si no que se le pida al cliente que obtenga la suya.

Interoperabilidad

Haciendo uso de la orientación a objetos, es fácil dividir y delegar responsabilidades, para posteriormente juntar todos los elementos y crear el proyecto.

2.- Análisis del problema

Tenemos que nos solicitan realizar un programa para la consulta del clima de una ciudad que nos proporcionen. En este caso necesitamos que nos den de la ciudad de origen y también de la de destino. No obstante, el consultar el clima de la ciudad de origen no aporta mucho, o bien, no es algo que en casos como la consulta para ciertos

vuelos en un aeropuerto se requiera del clima de la ciudad de origen, por lo que se tomará en cuenta solo la ciudad de destino.

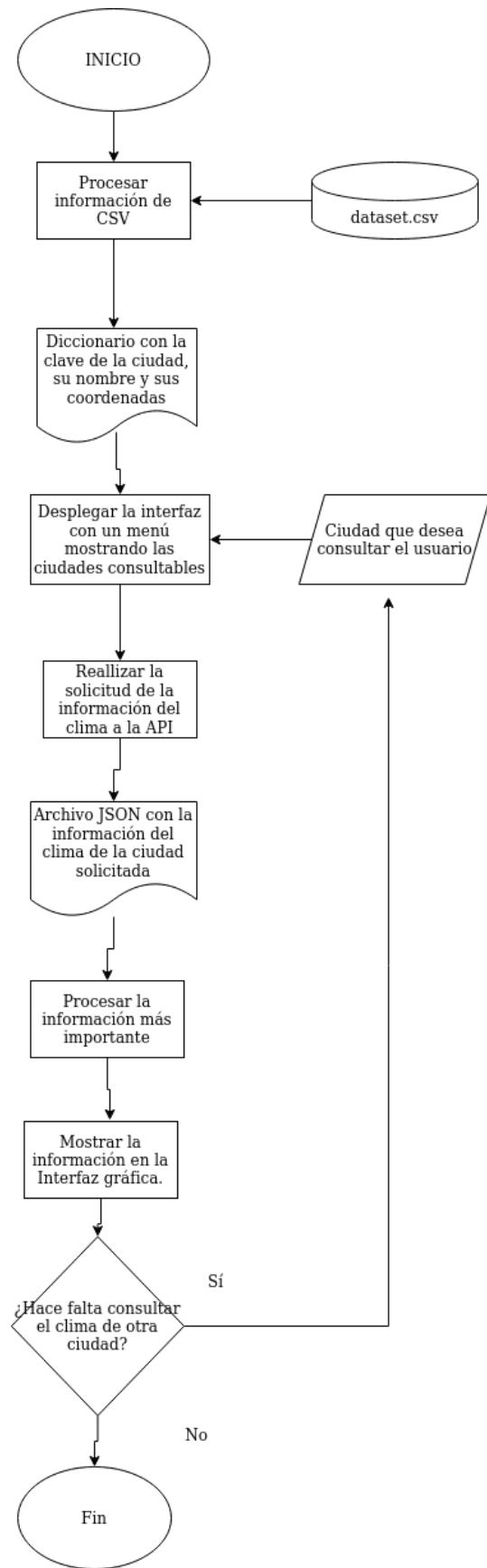
Para obtener información sobre los vuelos, contamos con un archivo CSV de donde haremos la lectura de las ciudades y sus coordenadas, para después procesarlas en una estructura de datos y posteriormente crear una solicitud del clima a OpenWeather, para finalmente mostrar la información más importante al usuario por medio de una interfaz gráfica.

3.- Selección de la mejor alternativa

Para este proyecto se optó por trabajar con el lenguaje de python debido a que cuenta con Orientación a Objetos, lo cual ayuda a separar el problema en partes más simples y así poder dividir el trabajo. Otra razón para escoger python es que este cuenta con muchas bibliotecas a disposición para trabajar con archivos JSON y CSV, así como también para realizar fácilmente requests e Interfaces gráficas, de modo que no tengamos que programar algo que ya existe y podamos centrarnos totalmente en la resolución del problema.

4.- Diagrama de flujo o pseudocódigo

Diagrama de flujo general:



Ejemplo de pseudocódigo para procesar el archivo CSV:

```
objeto Ciudad:  
    Ciudad(cadena_nombre, flotante_latitud, flotante_altitud)  
  
#Apertura del archivo de tipo csv.  
archivo = archivo.csv  
  
diccionario = {  
    #Arreglo donde almacena los nombres de las ciudades.  
    Arreglo_nombres: [],  
  
    #Arreglo donde almacena los objetos de tipo Ciudad.  
    Arreglo_ciudades: []  
}  
  
#Almacena los datos de cada ciudad en el diccionario verificando que no se repita  
#alguna ciudad.  
inicio for  
    for linea in archivo:  
        inicio if  
            if linea in diccionario:  
                diccionario <- linea  
        fin if  
    fin for  
  
#Devuelve el arreglo de los nombres de las ciudades.  
inicio function nombres()  
    return Arreglo_nombres  
fin function  
  
#Devuelve el arreglo de los objetos de tipo Ciudad.  
inicio function ciudades()  
    return Arreglo_ciudades  
fin function  
  
#Función que se encarga de realizar la búsqueda en el arreglo de objetos de tipo  
#Ciudad para obtener las coordenadas de la ciudad con base en su nombre pasado  
#como parámetro.  
inicio method ciudad(ciudad)  
    inicio for  
        for i in rango del diccionario  
        inicio if  
            if ciudad in nombres()[i]:  
                return ciudades()[i] (arreglo de los objetos de tipo Ciudad en la posición i)  
        fin if  
    fin for  
fin method
```

5.- Pensamientos a futuro

Cabe resaltar que el programa puede estar hecho para recibir cualquier ciudad, sea de origen o de destino. Al final quedaría esto como decisión final que el usuario tomará.

Además, la clase que contiene la solicitud puede ser fácilmente reutilizable por su encapsulación, así como la lectura de la “base de datos” que puede funcionar para cualquier otro CSV.

Posibles mantenimientos

Es posible que con base en la estructura de datos, se requiera de algo más eficiente, debido a la gran cantidad de información que esta pueda tener, además de tener la certeza de que en algún futuro se requiera buscar algo más eficiente para la consulta de datos.

También es posible mejorar la aplicación creando un archivo ejecutable, de tal forma que el usuario no tenga que abrir ninguna terminal.

Cantidad a cobrar por el reto

Pensamos que sería factible cobrar 50,000 pesos, sin incluir los costos de una posible suscripción a un plan más amplio a OpenWeather.