Universidad Nacional Autónoma de México

Facultad de Ciencias

Reto 01:

Modelado y Programación

Johann Ramón Gordillo Guzmán - 418046090

Proyecto presentado como parte del curso de **Modelado y Programación** impartido por el profesor **José de Jesús Galaviz Casas**.

20 de Agosto del 2019

1. Definición del Problema

Dado un archivo .csv (comma separated values) con los nombres de la ciudad de origen, ciudad destino, y la latitud y longitud de ambas ciudades [vésase **Entrada**], determinar las temperaturas en tiempo real de las ciudades origen y destino y mostrarlas al usuario [vésase **Salida**].

1.1. Entrada

Un archivo .csv (comma separated values) con los datos de la ciudad de origen, ciudad destino, y la latitud y longitud de ambas ciudades. El archivo se seleccionará por el administrador del sistema con ayuda de una interfaz gráfica de usuario (GUI).

1.1.1. Formato

.csv (Comma Separated Values).

1.1.2. Tamaño

Cada archivo con, a lo más, $3,000$ vuelos.
$1 \leq vuelos \leq 3,000$

1.1.3. Fuente

Directorio de archivos del usuario.

1.2. Salida

Las temperaturas en tiempo real de las ciudades origen y destino, por la salida estándar.

1.2.1. Formato

Texto plano.

1.2.2. Destino

Salida estándar.

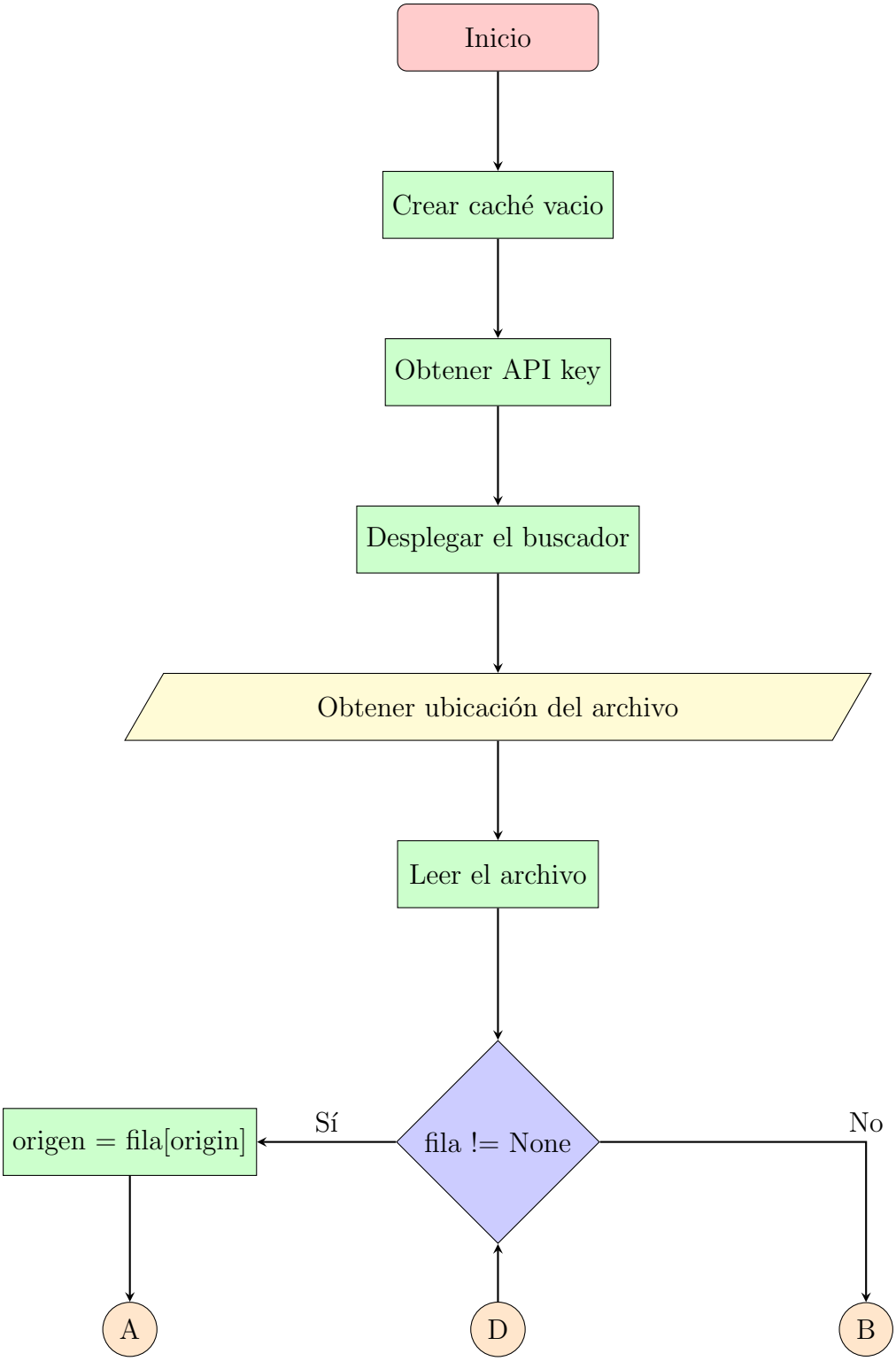
2. Análisis del Problema

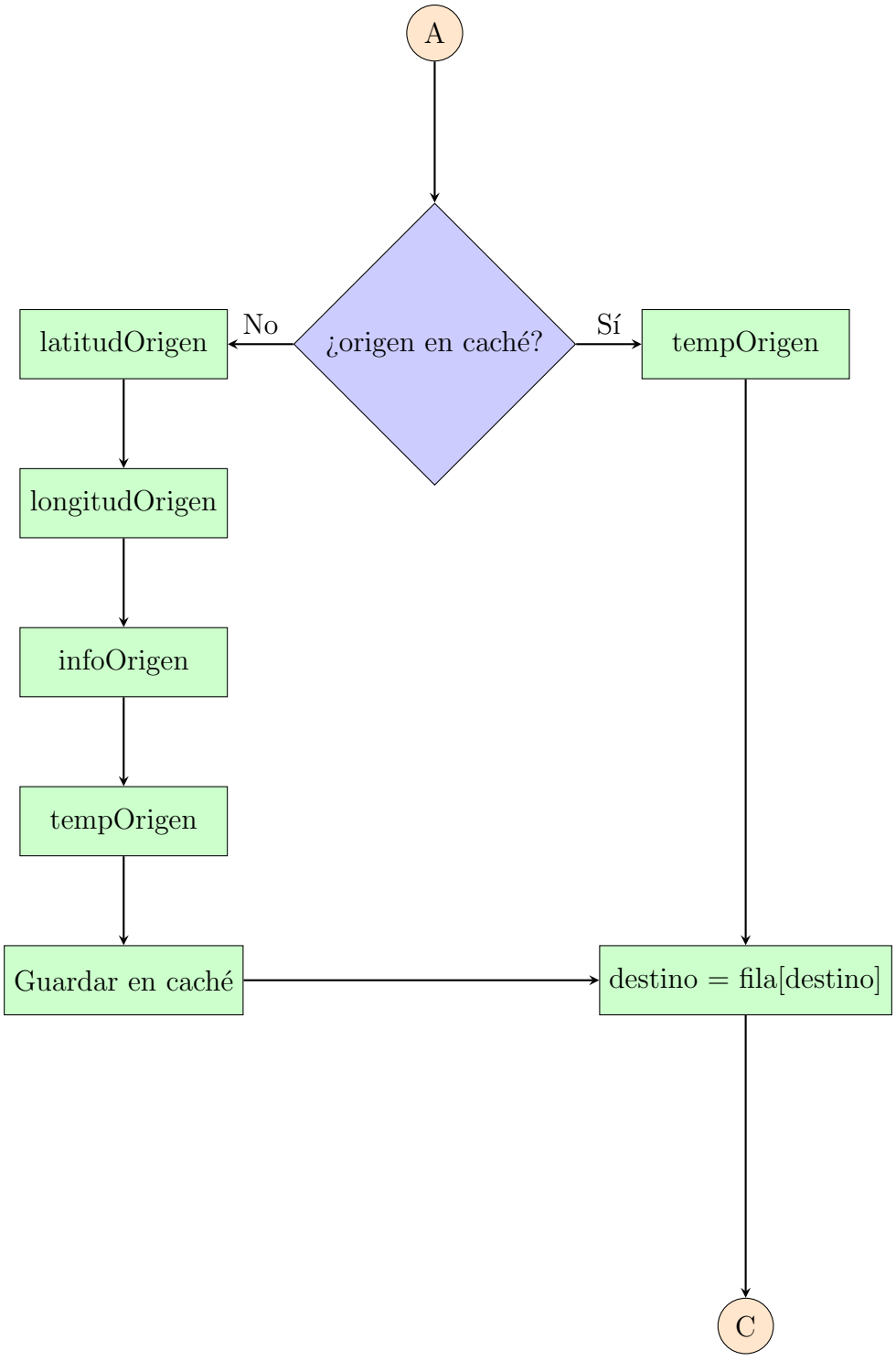
No es interactivo.

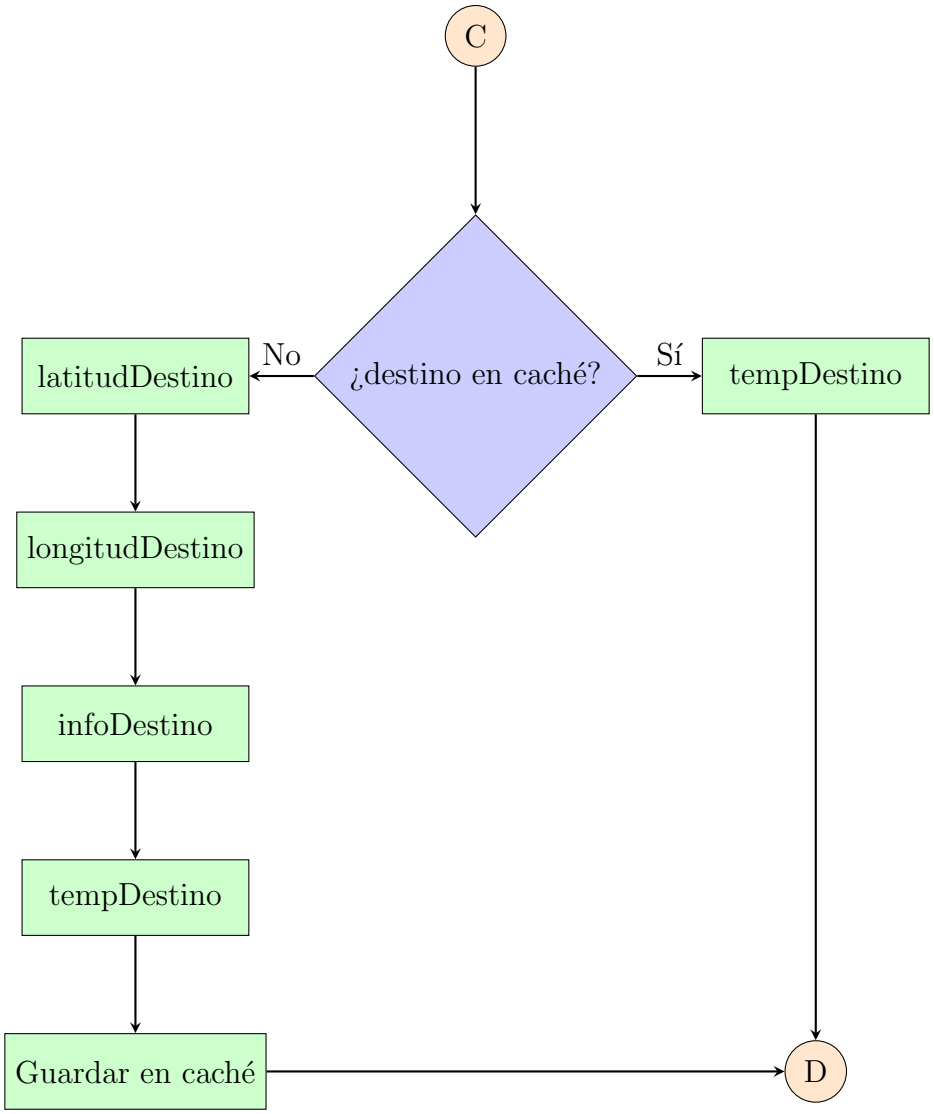
3. Selección de la mejor alternativa

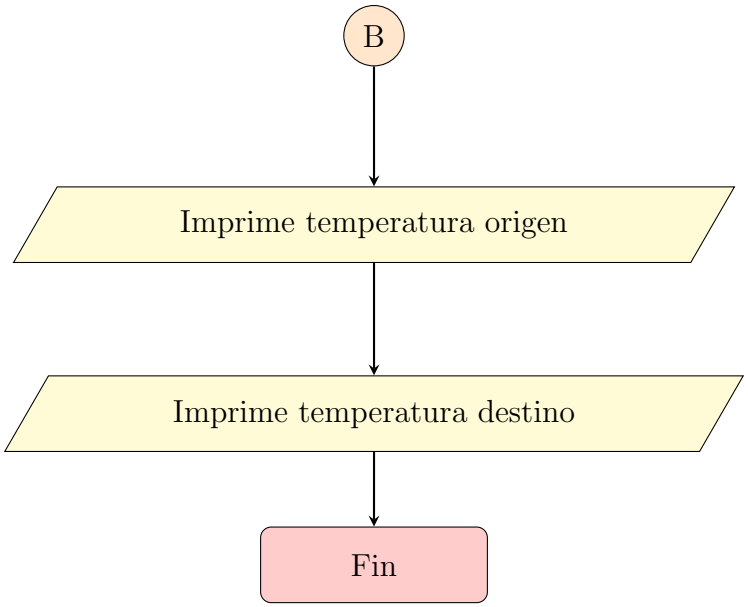
Se ha decidido usar el lenguaje de programación Python para la elaboración del proyecto. Python es un lenguaje interpretado muy fácil de usar, además de muy poderoso, y con muchísimas librerias, como 'requests' y 'pytest', que podrían ser de muucha ayuda en el programa.

4. Diagrama de Flujo




A
¿origen en caché?
No
Sí
latitudOrigen
longitudOrigen
infoOrigen
tempOrigen
tempOrigen
Guardar en caché
destino = fila[destino]
C


C
¿destino en caché?
No
Sí
latitudDestino
tempDestino
longitudDestino
infoDestino
tempDestino
Guardar en caché
D



4.1. Explicación

1. Iniciamos la ejecución del programa.

2. Primero creamos un caché donde guardaremos los datos solicitados al servidor de Open Weather Map, para evitar hacer consultas repetidas.

3. El siguiente paso es obtener la key de la API, para poder acceder al servicio web.

4. Una vez hecho esto, desplegamos un buscador de archivos (GUI) para que el usuario seleccione el archivo a procesar.

5. Ya con el archivo identificado, obtenemos su ruta para posteriormente leerlo.

6. Checamos que la fila del archivo leido no sea vacia, es decir, que haya algo que leer en el archivo. De ser así, pasa a «**8**». En caso contrario pasa a «**7**».

7. Ya se ha leido todo el archivo, y por ende se ha procesado ya toda la información requerida por el proyecto, imprime el clima de las ciudades origen y destino y pasa a «**16**».

8. Hay que leer los datos para la ciudad de origen, para ello, primero hacemos a la varibale origen igual a la columna ciudad que le corresponde en el archivo.

9. Si esta ciudad ya está dentro del caché, únicamente la extraemos y la guardamos en la variable **temp_origen**.

10. Si no lo está, buscamos su latitud y su longitud en el archivo leido.

11. Ya con la latitud y longitud ubicadas, hacemos la petición al servicio web para que nos regrese la información del clima en estas coordenadas geográficas, y guardamos los datos recibidos en una variable **info_origen.**

12. Extraemos de la variable **info_origen** la temperatura de la ciudad origen y la guadamos en la variable **temp_origen.**

13. Guardamos en el cache la información correspondiente a esta ciudad.

14. Repetimos los pasos «**8**» a «**13**», pero ahora con la ciudad destino.

15. Regresa al paso «**5**».

16. Finalizamos el programa.

5. Sobre el mantenimiento del código

5.1. Nuevas funcionalidades

El programa solicitado por el cliente solicita únicamente la temperatura de las ciudades de origen y destino. Sin embargo, Open Weather Map otorga más datos, como la temperatura máxima, la temperatura mínima y el pronóstico del clima; por lo que si el cliente así lo desea, el desarrollador podría agregar estas funcionalidades en futuras versiones.

Si el cliente en algún momento decide almacenar los datos obtenidos en un archivo, también se puede implementar sin ningún problema.

5.2. Bugs

Ningún programa es libre de bugs, y el desarrollador está conciente de ello, por lo que ante descubrimientos de bugs y errores en el código, el desarrollador se encargará de corregirlos y agregarlos a las pruebas unitarias.

5.3. Eficiencia

Si el programador descubre una manera más eficiente de implementar la solución al problema, notificará al cliente para estimar el costo de nuevas implementaciones. Por ejemplo, si se encuentra un algoritmo que permita reducir el tiempo en el que se hacen peticiones al servidor de Open Weather Map.

6. Costo final del proyecto

El costo del proyecto fue calculado con base en las horas que el desarrollador del mismo, Johann G., tuvo que invertir para verificar que la solución al problema planteado por el usuario, Aeropuerto Internacional de la Ciudad de México, fuera eficiente; y el código de la implementación, robusto.

Se implementaron pruebas para que en un futuro, desarrolladores que trabajen en el proyecto puedan implementar nuevas funcionalidades y aún así verificar si el código sigue funcionando correctamente bajo ciertos casos planteados estrategicamente por el desarrollador.

El costo final es de **$5,000** pesos mexicanos.

Para aclaraciones del proyecto y del costo del mismo contactar con **jgordillo@ciencias.unam.mx**