
Modelado y programación

Semestre 2023-1

Proyecto 01: Web service

Alumnos:

Rodríguez Belmonte Lázaro Eduardo 319131099

Pablo César Navarro Santana 31733091

1. Definición del problema

“Día a día miles de personas van y vienen de aeropuertos a aeropuertos, cambiando drásticamente de zona horaria y clima. El aeropuerto de la Ciudad de México te contrata para una tarea, la cual es entregar el informe del clima de la ciudad de salida y la ciudad de llegada para 3 mil tickets que salen el mismo día que se corre el algoritmo.

El programa no busca que sea interactivo, ya que el mercado a quien va dirigido son sobrecargos y pilotos que desconocen de programación, por lo que solo nos interesará el clima.”

Dadas las instrucciones anteriores concluimos que tenemos que presentar una aplicación del clima sencilla, pues no se espera que sea usada para algo distinto de consultas, también debe ser accesible para el público general y ser robusta pues no es deseable que falle la ejecución en medio de un día laboral.

De lo anterior deducimos que el núcleo del programa será consultar con un API el clima, y mostrarlo en pantalla. Sin embargo, la versión gratuita de openweathermap solo permite 1 petición por segundo por lo que será necesario implementar un cache, así concluimos que sería una buena decisión desarrollar una página web pues resulta de conocimiento general para casi todo público hacer uso de ellas.

Entonces podemos comenzar a elegir nuestro arsenal, para el API del clima decidimos usar **openweathermap** pues es gratuito y presenta bastante documentación frente a las demás, además de ofrecer implementaciones sencillas en varios lenguajes de programación. Para desarrollar la página web decidimos usar **html, css y javascript** pues son lenguajes especializados para el desarrollo web. Avanzando un poco más en el problema notamos que se requería el acceso a una base de datos (csv) así que requeríamos una herramienta que permitiera a javascript ejecutar esta tarea por eso es que usamos **ajax** para leer el csv y **jquery-csv** para procesar los datos. Además de todas estas herramientas usamos **git y github** para el trabajo cooperativo y el control de versiones.

Para los **requisitos funcionales** tenemos que dada una base de datos (o la ubicación del lugar deseado) se debe devolver los climas de llegada para los boletos o la ubicación deseada, originalmente en el problema se pide que sea devuelta también el clima del lugar de salida sin embargo resulta un poco redundante pues basta con voltear al cielo para saber este dato.

Para resolver lo anterior tenemos que primero se debe obtener el link de consulta al API para posteriormente realizar la petición (si es que no se encuentra en el cache), también debemos leer los datos para finalmente ser capaces de devolverlos procesados y mostrarlos en pantalla.

Para los **requisitos no funcionales** tenemos que deber ser amigable con el usuario nuevo, debe ser una aplicación robusta que se ejecute en un tiempo eficiente y debe ser segura pues los datos de vuelos pueden contener información sensible.

2. Análisis del problema

Una vez leídas las instrucciones dadas en el pdf proporcionado por la ayudante encontramos que dada una base de datos debemos devolver el clima de la ciudad de salida y la ciudad de llegada, sin embargo la ciudad de salida resulta un poco obvia pues es donde se encuentra la persona en el momento. También notamos que la aplicación sería usada por sobrecargos por lo que tendría que ser lo más accesible posible para personas sin gran conocimiento computacional.

Así mismo necesitamos hacer peticiones de forma efectiva y manejar la información para ser mostrada en pantalla, también de lo anterior encontramos que necesitamos manejar una base de datos y un caché pues la api está limitada a 60 peticiones por segundo.

Finalmente necesitamos resolver el aspecto estético pues si bien una interfaz de usuario no era necesaria para el proyecto sería muy deseable que la tuviese pues resulta muy amigable para gente ajena a la licenciatura.

3. Selección de la mejor alternativa

Backend

Como ya se había visto en las herramientas decidimos usar js para crear una pequeña página web así también decidimos que javascript vanilla haciendo uso de la orientación a objetos sería más que suficiente para este proyecto, pues el uso de otras herramientas requeriría más tiempo de aprendizaje así como más complicaciones para manejar las herramientas.

Para el manejo de la base de datos con js vanilla creamos un objeto que manejaría la información y buscaríamos los objetos desde el mismo, decidimos usar **jquery** para acceder a la información haciendo una petición al archivo y usando **jquery-csv** para hacer la conversión al array de objetos.

Para el caché decidimos crear un objeto que administraría la eliminación cada 5 minutos y permitiría buscar la información deseada, para borrar en 5 minutos decidimos usar una promesa que una vez pasados 5 minutos borra la información que fue añadida.

Para mostrar y manejar el objeto dado por las peticiones se creó un objeto **weather** que contiene el apikey, el caché y la base de datos propia del clima, incluimos también tres funciones, una para realizar la petición, otra para mostrar el clima con **document.querySelector** y finalmente la función que lee lo que se encuentra en la barra de búsqueda y manda a la primera función para hacer la petición.

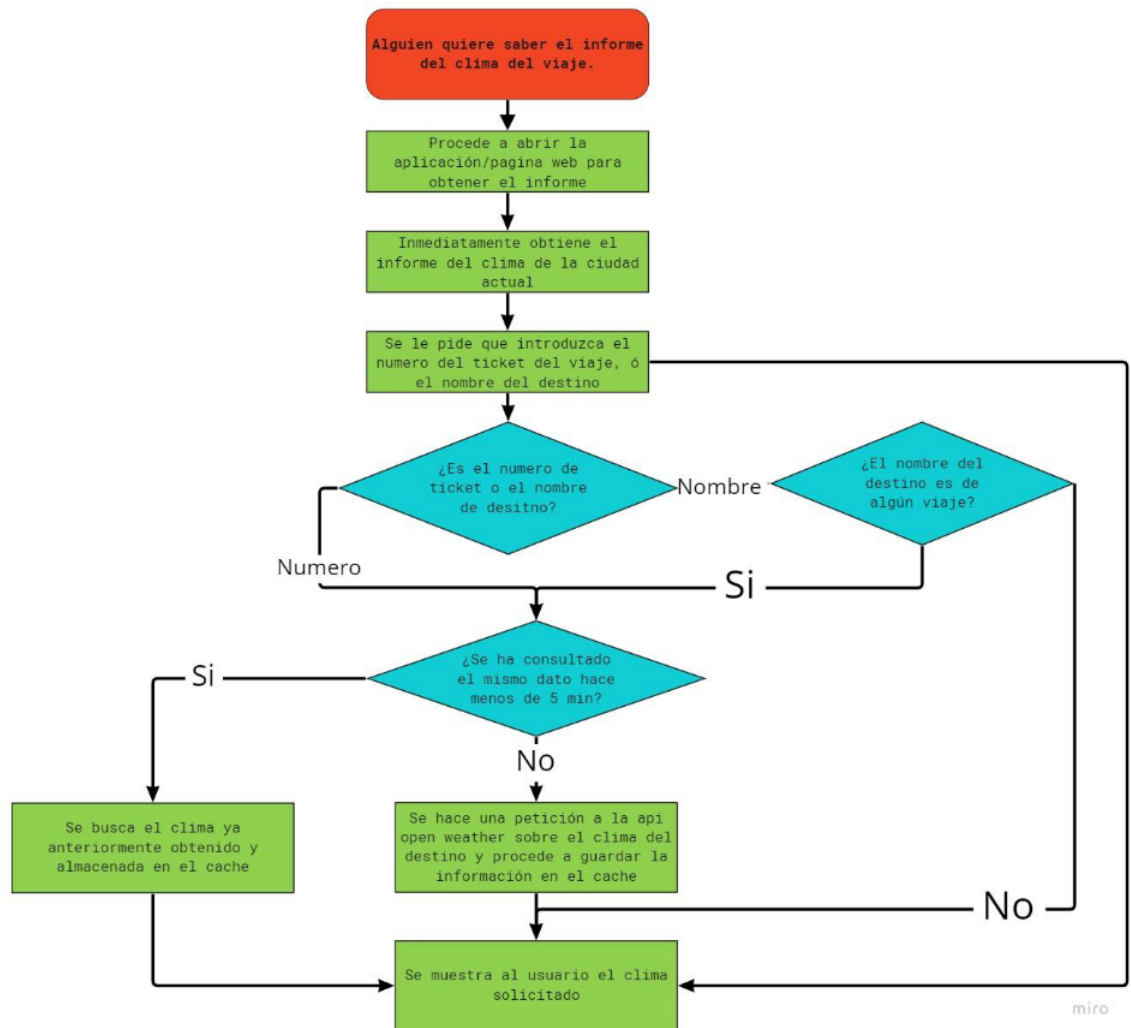
Finalmente no deseábamos deshacernos del clima de salida así que decidimos que una vez se carga la página se accedería a la ubicación del dispositivo para poder mostrar el clima de salida como el primer clima mostrado por la aplicación.

Frontend

Se pensó en mostrar una tabla con todos los climas de la base de datos, sin embargo resultaría por momentos difícil buscar el clima o sería estéticamente feo además que podría tardar más en cargar información por eso se optó por una barra de búsqueda con un recuadro debajo donde se mostrará el clima que se solicitó.

También se optó por cambiar de forma dinámica el fondo pues de esta forma se da un aspecto más estético y dinámico.

4. Diagrama de flujo



De forma más resumida:

El objetivo de este algoritmo es mostrar al usuario el informe del clima de origen (Lo cual es un poco obvio, por ello solamente se le presenta una vez) y luego el clima del destino, ya sea que inserte el número del ticket o con el mismo nombre del destino.

Para optimizar este proceso y no abusar de la API de Open Weather, el algoritmo almacena los datos del clima en un cache y si se le vuelve a preguntar por el clima de este lugar en un lapso de 5 min, devuelve de nuevo el clima pero ahora desde la información del cache.

5. A futuro

A futuro podriamos implementar una seccion de noticias del lado derecho de la pantalla o implementar la tabla mencionada antes pues puede ser util para algùn caso especifico o si es que se desea consultar multiples climas a la vez, tambien podrian implementarse mejoras esteticas y futuras mejoras de seguridad. Ajeno al codigo en js se pediria que en la base de datos venga el id pues en nuestro caso la base de datos fue tratada con python para a  adir la columna de id.

En cuanto al cobro de la aplicaci  n la mayor parte del tiempo fue aprender herramientass y lamentablemente esas no se pagan, fuera de eso pasamos alrededor de 15 dias desarrollando las soluciones y programando la implementaci  n asi que diremos que dos sueldos de una quincena de programador por lo que cobraríamos alrededor de 15000 pesos.

Para futuras mejor a nuestro parecer el codigo no es tan dif  cil de mejorar pues dentro de todo es bastante modular por lo tanto no cobrar  amos mucho aunque depende del grado de mejora.

6. Extras:

Se sigui   parte de este tutorial pues ambos eramos nuevos en el manjo de todas las herramientas mencionadas.

<https://www.youtube.com/watch?v=WZNG8UomjSI>

Usamos como gu  a los ejemplos en el siguiente repo para el manejo de la base de datos.

<https://github.com/evanplaice/jquery-csv>