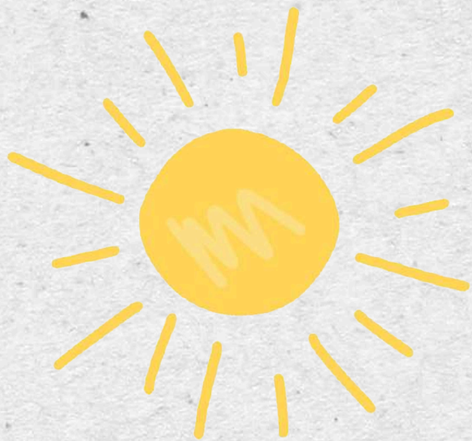
Presentado por Luis Juárez Erick, Barrera Hérnandez Tomás, Rodríguez Jiménez Brenda, Sánchez Gónzalez Oscár Iván y Peña Villegas Diego Eduardo

Presentación de la App del Clima para usuarios y trabajadores

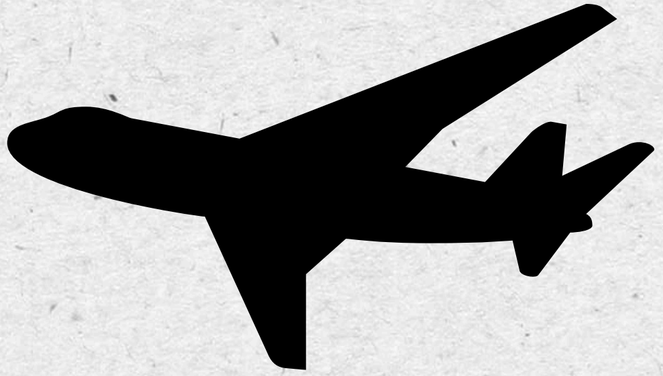
Aeropuerto Internacional de la Ciudad de México

Secciones

- Definición del problema
- Análisis del problema
- Selección a la mejor alternativa
- Diagrama de flujo
- Pensando a futuro (Futuros Mantenimientos)
- Presupuesto
- ¿Por qué nuestra solución es la mejor?



Descripción del problema



Descripción del problema

Contexto: Miles de personas cada día transitan de aeropuertos a aeropuertos, los cuales enfrentan cambios drásticos de zona, horario y clima. Los clientes y el personal necesitan saber la información climática precisa en tiempo real, tanto en el lugar de origen como en el lugar de destino.

Descripción del problema

Requisitos: Crear una aplicación interactiva, intuitiva y amigable para los usuarios, que les permita consultar el clima de cualquier destino en tiempo real.

Descripción del problema

Arsenal

Lenguaje de programación utilizado: Python

- **OpenWeatherMap: es un servicio online que proporciona datos climáticos y meteorológicos en tiempo real a través de una API.**
- **Tkinter: es una API de Python que permite el uso de la librería de código abierto Tk, la cual provee de los elementos básicos para la creación de interfaces gráficas de usuario.**
- **Archivos CSV: Archivos que contienen IATAS de origen y destino con sus respectivas longitudes y latitudes, tickets con sus respectivas IATAS de origen y destino y el nombre de las ciudades de la IATAS válidas.**
- **csv: es un módulo integrado de python que sirve para leer y escribir archivos CSV.**

Descripción del problema

Arsenal

- webbrowser: es un módulo de Python que proporciona una interfaz de alto nivel que permite mostrar a los usuarios documentos basados en la web.
- JSON: es un módulo integrado de Python que es utilizado para convertir datos en formato JSON a objetos Python.
- request: es una biblioteca de Python que ayuda a la realización de solicitudes HTTP y facilita la interacción con API’s y la obtención de datos desde una web.
- unicodedata: es un módulo integrado de Python que nos da acceso a la base da datos de caracteres Unicode, y es utilizado para normalizar textos, trabajar con caracteres especiales y realizar operaciones con Unicode.

Descripción del problema

Arsenal

- difflib: es un módulo de Python que sirve para comparar cadenas de texto y encontrar las diferencias entre ellas, es útil para búsquedas.
- os: es un módulo integrado de Python que sirve para interactuar con el sistema operativo, es útil para crear, modificar o eliminar archivos, al igual que sirve para crear rutas hacia archivos o carpetas dentro de un proyecto.
- pytest: es un framework de pruebas Python que facilita la escritura de pruebas unitarias.
- cron: herramienta integrada en linux para programar la actualización de los datos en intervalos de tiempo regulares

Descripción del problema

Requisitos Funcionales

El usuario debe interactuar con el programa de la siguiente manera:
Debe proporcionar, a través de su teclado, un código IATA, una ciudad o un ticket de vuelo.

- En caso de haber ingresado una IATA, el programa debe mostrar los datos del clima correspondientes a la ciudad donde se encuentra ubicado el aeropuerto identificado por la IATA del usuario.
- Si fue ingresada una ciudad, el programa debe mostrar los datos del clima correspondientes a esa ciudad.
- Y cuando se ingrese un ticket de vuelo, el programa debe mostrar tanto los datos del clima de la ciudad de origen (de despegue) como la de destino (de aterrizaje).

Descripción del problema

Requisitos Funcionales

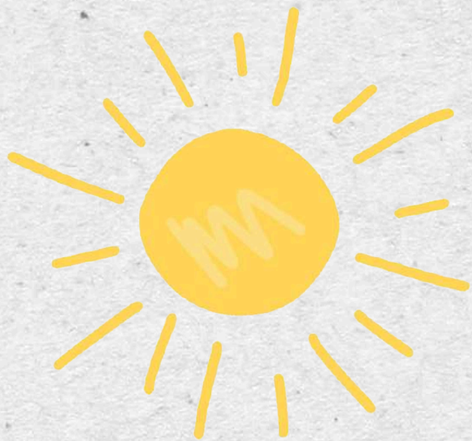
En cualquier caso, estos datos deben ser los siguientes:

- La temperatura
- La humedad
- La probabilidad de lluvia
- La presión atmosférica
- La velocidad del viento

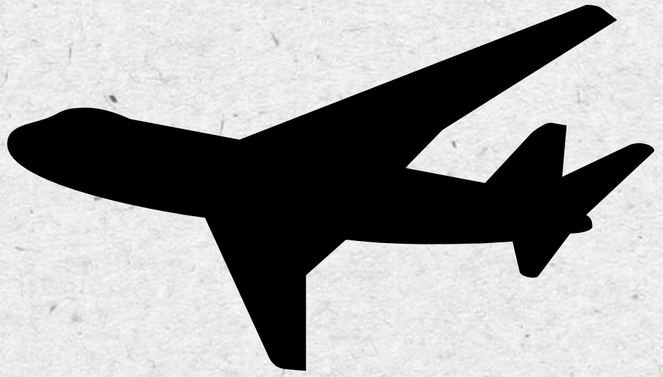
Descripción del problema

Requisitos No Funcionales

El programa tiene un buen rendimiento, ya que entrega los resultados solicitados en cuestión de, a lo más, segundos. Además, es portable, ya que puede ser ejecutado en distintas versiones de Linux y de Windows. Su diseño claro, con alta cohesión y bajo acoplamiento, permite que el código pueda ser modificado, y se le pueda dar mantenimiento, sin dar lugar a errores graves y difíciles de tratar. Por otra parte, el programa es muy intuitivo y amigable con el usuario, dejando toda la información al alcance de un click. Es capaz de manejar errores en la entrada del usuario, logrando identificar correctamente las salidas que debe producir pese a faltas ortográficas que se le pudiesen ingresar.



Análisis del problema



Análisis del problema

Para atacar el problema, podemos proceder como sigue:

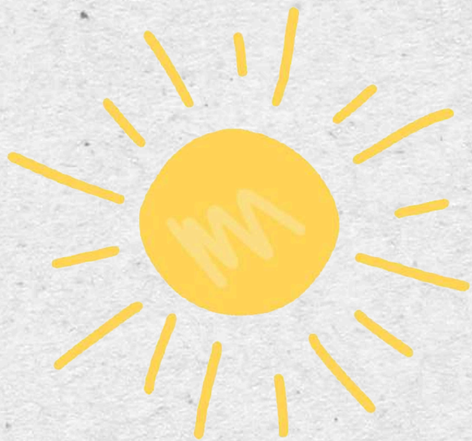
- Construir una URL hacia OpenWeatherMap específica para cada ciudad.
- Devolver en formato JSON los datos que hayan sido recuperados desde esa URL, los cuales serán precisamente los datos del clima de la ciudad especificada.
- De toda esa variedad de datos, solo nos interesan unos cuantos, a saber: temperatura, presión, humedad, probabilidad de lluvia, y velocidad del viento. De modo que hay que extraer del JSON solo esa información, en forma de diccionario.
- También se debe tener un archivo CSV que contenga todas las IATAs pertinentes, así como la latitud y longitud de cada una de ellas. Además, se deberán tener guardados varios tickets de vuelo, para los cuales deberemos ser capaces de obtener la IATA de origen y la de destino en un diccionario.
- Adicionalmente, hay que tener otro archivo CSV que enliste todas las IATAs con las respectivas ciudades donde se ubican sus aeropuertos.

Análisis del problema

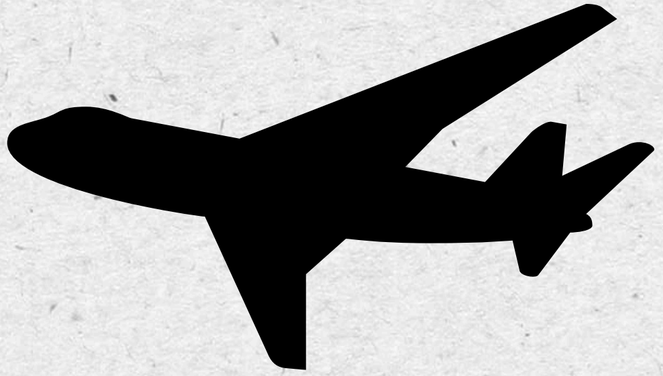
- Construir un buscador que sea capaz de relacionar una ciudad mal escrita con una existente en el archivo CSV, para así poder devolver la IATA asociada según el mismo archivo.
- Una vez se tenga la IATA, obtenemos las coordenadas asociadas, así como la ciudad. Esto se hace accediendo a los archivos CSV correspondientes.
- Con el nombre de la ciudad a la mano, generamos la URL y obtenemos los datos climáticos con el procedimiento descrito anteriormente.
- En el caso de los tickets, obtenemos el diccionario que les asigna los IATAs de origen y de destino, para que con ellos podamos acceder a las ciudades correspondientes. Una vez que las obtengamos, procedemos como en el caso anterior.
- Ya solo bastará con diseñar una interfaz gráfica que cuente con un cuadro de texto que reciba una entrada (IATA, ciudad o ticket), y muestre en pantalla los datos climáticos obtenidos.

Análisis del problema

- Para la actualización continua de los datos en caché, proponemos el uso de cron en sistemas Linux para que tengamos una actualización de los datos periódicamente. (recomendación 5 días).



Selección a la mejor alternativa



Selección a la mejor alternativa

Nuestro programa es la mejor alternativa, pues la interfaz es amigable con el usuario, no está saturada, es clara y concisa, así como la presentación de los datos del clima vienen especificados dependiendo de si el usuario hace una búsqueda por IATA, ciudad o ticket, en donde no importa si el usuario escribe mal la ciudad o IATA, pues hay un sistema de corrección automática de la entrada de datos, facilitando al usuario la consulta.
Asimismo, el programa es eficiente, ya que tiene medidas de seguridad para la única entrada de datos, donde esta se está limitando de mínimo 3 caracteres a máximo 10.
En general el programa es intuitivo, fácil de usar y no está saturado de información, haciendo que el programa sea sencillo, pero que cumpla con los requisitos solicitados.

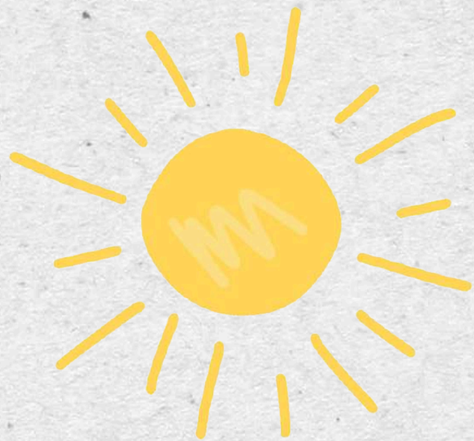


Diagrama de flujo

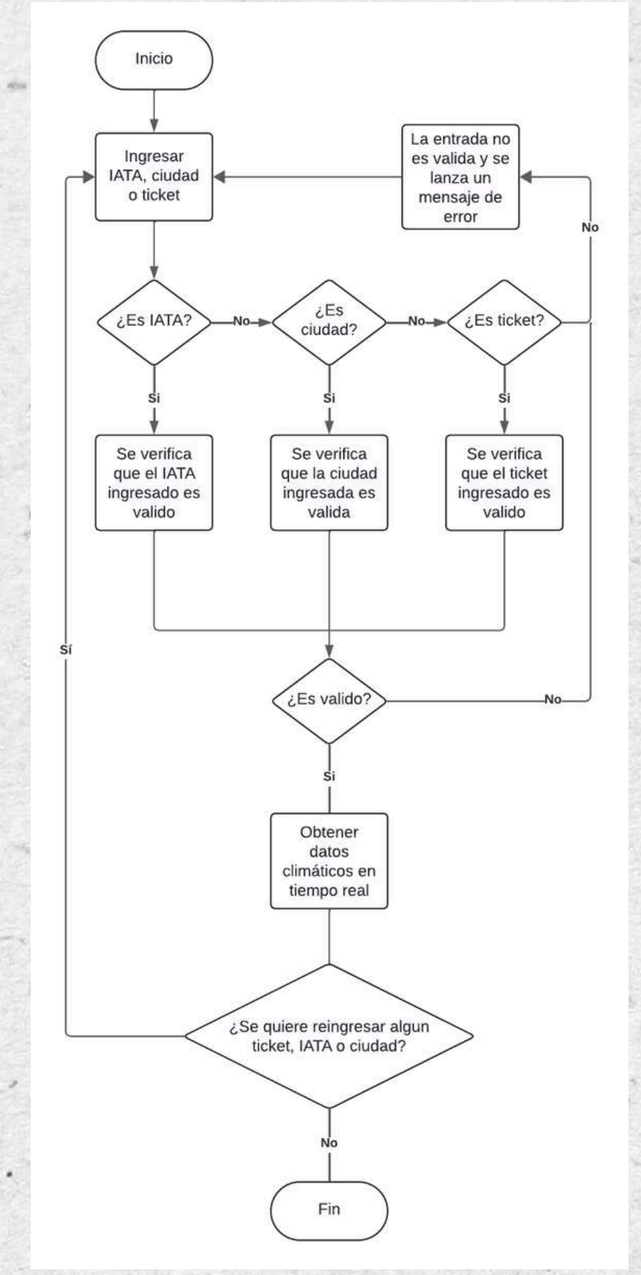
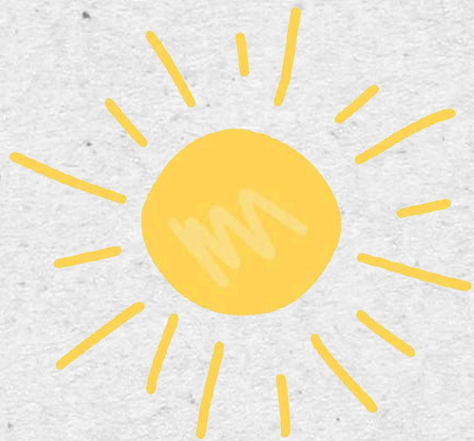

Inicio
Ingresar IATA, ciudad o ticket
La entrada no es valida y se lanza un mensaje de error
No
¿Es IATA?
No
¿Es ciudad?
No
¿Es ticket?
Si
Si
Si
Se verifica que el IATA ingresado es valido
Se verifica que la ciudad ingresada es valida
Se verifica que el ticket ingresado es valido
Sí
¿Es valido?
No
Si
Obtener datos climáticos en tiempo real
¿Se quiere reingresar algun ticket, IATA o ciudad?
No
Fin

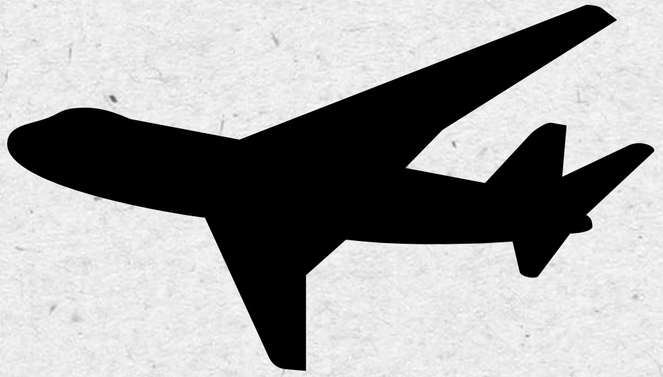
Diagrama de flujo

Explicación del diagrama de flujo

El diagrama de flujo muestra el proceso del programa para obtener datos climáticos en tiempo real al ingresar una IATA, una ciudad o un ticket. El proceso inicia con el usuario ingresando una IATA, una ciudad o un ticket, después de que el usuario introduzca alguna de estas 3 entradas, se verifica qué tipo de entrada es, si es IATA, pasa al siguiente paso, si no lo es, se verifica si es nombre de ciudad. Si es nombre de ciudad, pasa al siguiente paso, si no lo es, se verifica que es ticket. Si es ticket, pasa al siguiente paso. Si no es ninguno de los 3, se muestra un mensaje de error y se vuelve a solicitar al usuario que introduzca alguna de estas 3 entradas. Luego de que se verifique que la entrada es alguno de estos 3, cada tipo de entrada es verificado para saber si es válido. Si la IATA, el nombre de la ciudad o ticket no es válido, se muestra un mensaje de error y se solicita al usuario que ingrese una entrada válida. Si es válida, entonces se devuelven los datos climáticos correspondientes. Finalmente, el usuario tiene la opción de reingresar algún IATA, ciudad o ticket. Si el usuario elige reingresar otra entrada, el proceso vuelve a repetirse, en caso contrario, el proceso finaliza.

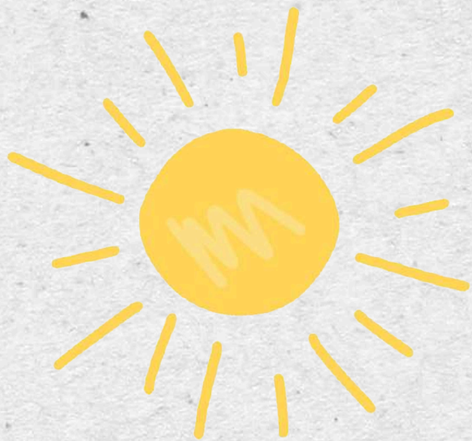


Pensando a Futuro (Futuros mantenimientos)

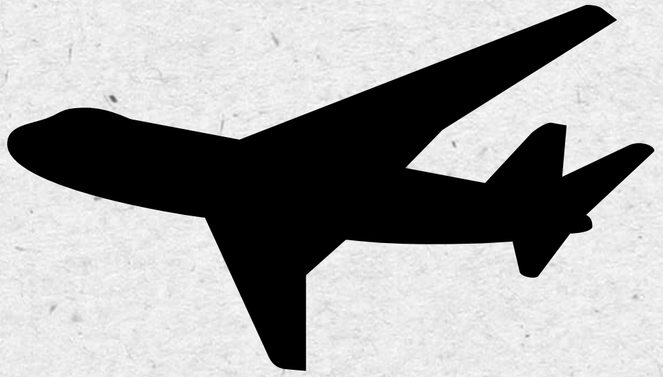


Pensando a Futuro (Futuros mantenimientos)

- Mantenimiento Propuesto: Revisión mensual para actualización de dependencias y optimización de código, con un costo de $50,000 MXN al mes para un equipo de 5 programadores.
- Futuras Mejoras: Implementar nuevas funciones como pronósticos extendidos o alertas climáticas en tiempo real.
- Justificación de Costos: Basado en la cantidad de horas necesarias y el nivel de especialización requerido, esta tarifa asegura un servicio de alta calidad y disponibilidad.



Presupuesto



Presupuesto

1. **Desarrollo del Proyecto:**

- **Equipo de desarrollo:** 5 programadores especializados en Python
- **Duración estimada:** 3 meses
- **Costo por programador:** $35,000 MXN x 5 programadores x 3 meses = $525,000 MXN

2. **Mantenimiento:**

- **Equipo de desarrollo:** 5 programadores especializados en Python
- **Precio por hora:** $1,000 MXN
- **Costo mensual de mantenimiento:** $50,000 MXN
- **Costo anual de mantenimiento:** $50,000 MXN x 12 meses = $600,000 MXN

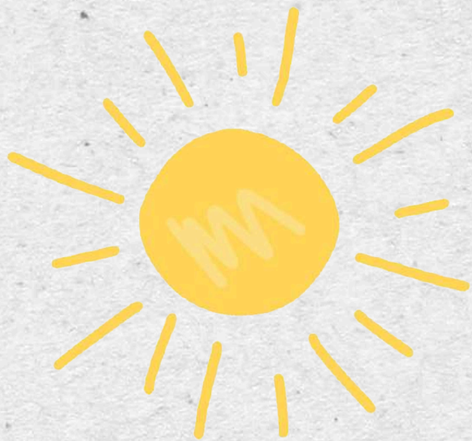
Presupuesto

3. **Costo de electricidad:**

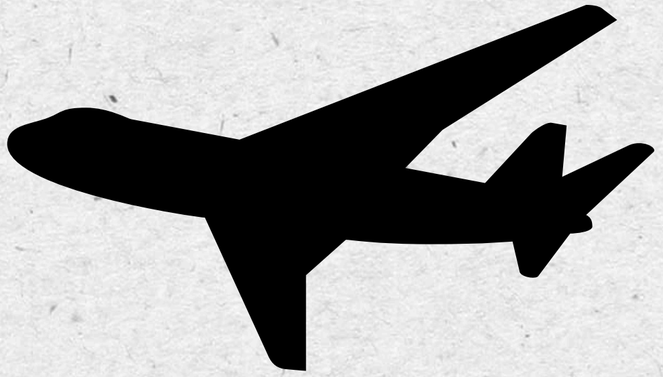
- **Costo mensual por programador:** $500 MXN
- **Número de programadores**: 5
- **Costo mensual total de electricidad**: $500 MXN x 5 programadores = $2,500 MXN
- **Costo total de electricidad durante el desarrollo**: $2,500 MXN x 3 meses = $7,500 MXN

4. **Total del proyecto:**

- **Costo total de desarrollo:** $525,000 MXN
- **Costo total de mantenimiento:** $600,000 MXN
- **Costo total de electricidad:** $7,500 MXN
- **Total general:** $1,132,500 MXN
-



¿Por qué nuestra solución es la mejor?¿Qué necesidades cumple?

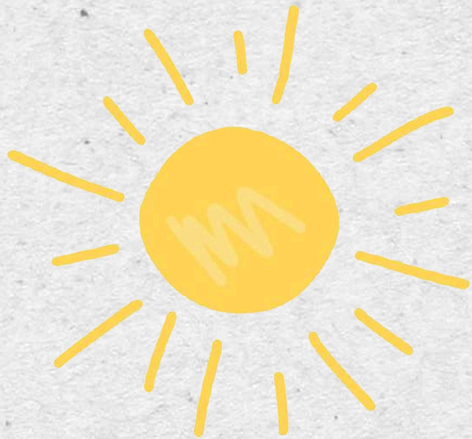


¿Por qué nuestra solución es la mejor?

Nuestro programa cubre con las necesidades de saber cuál es el clima dependiendo los datos que se introduzcan, ya que si buscamos por IATA o ciudad, se nos proporciona los datos de clima de la ciudad. En caso de que hagamos la búsqueda por ticket, entonces se nos proporcionan los datos de clima del lugar de origen y del lugar del destino. También nuestro programa tiene publicidad de sitios web para comprar boletos de avión en la página oficial del Aeroméxico; así como el sitio web para realizar reservas de hospedajes a precios económicos en la página de Trivago. Por lo que estaríamos ayudando a que el usuario se ahorre las búsquedas con estas recomendaciones.

AICM

trivago



¡Muchas gracias!