Proyecto 01

1. Definición del problema (entender el problema, arsenal y requisitos funcionales y no funcionales)

¿Qué es lo que queremos obtener? Proporcionar a los pilotos y sobrecargos el clima de la ciudad de salida de su vuelo y el clima de la ciudad de llegada.

¿Cuales son los datos que tenemos para obtenerlo? La latitud y longitud de las ciudades de destino y origen (3 mil tickets), zona horaria y web services para consultar el clima (OpenWeatherMap y Yahoo Weather).

¿Son suficientes? Sí, al menos que se necesiten más especificaciones particulares.

¿Qué hace que el resultado obtenido resuelva el problema? ¿Cual es la característica que hace de un resultado, una solución? Que el programa cumpla con su funcionalidad, en este caso proporcionar el clima de las ciudades de origen y destino a los sobrecargos y pilotos.

¿Qué operaciones o construcciones se deben obtener para llegar a la solución? Una cuenta en alguna Web services para consultar climas.

2. Análisis del problema.

El objetivo de este programa es crear una aplicación de fácil comprensión que le proporcione a los usuarios, en este caso pilotos y sobrecargos de una aerolínea el clima de la ciudad de salida del vuelo y de llegada.

Para esto contamos con un archivo csv el cual nos proporciona los datos de 3 mil tickets, en este archivo notamos el lugar de origen y el destino, veamos que también nos proporciona las longitudes y latitudes de cada uno de estos lugares.

También contamos con una cuenta en OpenWeatherMap o Yahoo Weathe, en alguno de estos dos web service crearemos una cuenta y utilizaremos la API que nos proporciona insertando la key y url.

Tomemos en cuenta que nuestro programa debe de ser robusto y eficiente, también que debe de llevar un lenguaje un tanto "coloquial" el cual sea de fácil comprensión para el tipo de usuario al que va dirigido.

3. Selección de la mejor alternativa.
Arsenal

- Paradigma de programación: Programación orientada a objetos.
- Lenguaje: Python.
- Herramientas para pre o post procesar los datos: web services OpenWeatherMap y Yahoo Weather, archivo csv con datos de 3 mil tickets.

4. Requisitos funcionales y no funcionales

-Requisitos funcionales: El programa debe de proporcionar el clima de la ciudad de origen y destino que ingrese el usuario.
- Requisitos no funcionales: Eficiencia en el programa, robusto, es decir, que esté preparado para cualquier error que ingrese el usuario, amigabilidad con el usuario, con esto nos referimos a que sean entendibles las instrucciones que se piden, seguridad para los datos personales de los 3 mil tickets, facilidad de lectura, ético y que no tarde tanto en cargar y ejecutar el programa.

5. Pseudocódigo.

// Mostrar el clima del lugar de origen de un aeropuerto y el clima del lugar de llegada.
// Análisis
// Entradas: Lugar de origen y lugar de destino.
// Salidas: Clima de lugar de origen y clima de lugar de destino.

// Variables: origen_latitud, origen_longitud, destino_latitud, destino_longitud, ciudad, ciudad_dest, temperatura, presión, humedad, amanecer, puesta de sol, tiempo.

***NOTA:** Antes de realizar el programa se debe de tener previo una cuenta en OpenWeather para poder adquirir la API y de ahí la Key que nos ayudará a extraer los datos correspondientes.

Proceso archivo_principal

[1] Biblioteca climage
[2] importar clase ciudad_dataset_destino, ciudades_dataset_origen
[3] sub algoritmo imagen_terminal():
[4] **Escribir:** ruta del archivo imagen
[5] Del formato png lo convierte para imprimirlo en la terminal
[6] Manda a llamar al metodo imagen_terminal()
[7] Manda a llamar al metodo clima_ciudades_origen()
[8] Imprime una linea
[9] Manda a llamar al metodo clima_ciudades_destino()
[10] **Fin** del programa archivo_principal

Proceso ciudad_dataset_destino

[1] Biblioteca requests, hora, timezone
[2] importar clase leer_csv
[3] api_key='3db4673f57258dba07e7d28b9f4b8634'
[4] url="http://api.openweathermap.org/data/2.5/weather?"
[5] ciudad$_d$*est* = **Escribe:** *Ingresa ciudad*
[6] completa$_u$*rl*2 = *url* + *api_key* + *ciudad_dest*
[7] res1 = completalaurlanterior
[8] datos_ciudad2 = res1.json()
[9] sub algoritmo clima_ciudades$_d$*estino*() :
[10] si datos_ciudad2[çod"] diferente "404":
[11] kelvin=273.15
[12] guarda_datos = datos$_c$*iudad*2["*main*"]
[13] temperatura1= entero(guarda_datos["temp"]-kelvin)
[14] presion1 = guarda_datos["pressure"]
[15] humedad1 = guarda_datos["humidity"]
[16] amanecer1=datos_ciudad2['sys']['sunrise']
[17] puesta1=datos_ciudad2['sys']['sunset']
[18] timezone1=datos_ciudad2['timezone']
[19] datos_clima = datos_ciudad2["weather"]
[20] descripcion1 = datos_clima[0]["description"]
[21] amanecer_tiempo=tiempo_de_puesta_de_sol_amanecer(amanecer1+timezone1)
[22] puesta_tiempo=tiempo_de_puesta_de_sol_amanecer(puesta1+timezone1)
[23] imprime(Çiudad de destino:")
[24] imprime("Temperatura(Celsius) = - varibale(temperatura1) + "Presiónatmosferica= - variable(presion1) + "Humedad(en porcentaje) = - variable(humedad1) + "Descripción = - variable(descripcion1) + "Latitud = - destino$_l$*atitud* + "*Longitud* = " + *destino_longitud*)
[25] imprime(Amanecer a las amanecer_tiempo y la puesta de sol a las puesta_tiempo)
[26] De lo contrario:
[27] imprime("No se encontró la ciudad ")
[28] **Fin** del programa ciudad_dataset_destino

Proceso ciudades_dataset_origen

[1] Biblioteca requests
[2] importar clase leer_csv
[3] api_key='3db4673f57258dba07e7d28b9f4b8634'
[4] url="http://api.openweathermap.org/data/2.5/weather?"
[5] ciudad = **Escribe:** Ingresa ciudad
[6] completa_url = url + api_key + ciudad
[7] res = completalaurlanterior
[8] datos_ciudad1 = res.json()
[9] sub algoritmo clima_ciudades$_{o}rigen()$:
[10] si datos_ciudad1[çod"] diferente "404":
[11] kelvin=273.15
[12] guarda_datos_ciudad1 = datos$_{c}iudad1$["$main$"]
[13] temperatura= entero(guarda_datos_ciudad1["temp"]-kelvin)
[14] presion= guarda_datos_ciudad1["pressure"]
[15] humedad = guarda_datos_ciudad1["humidity"]
[16] datos_clima1 = datos_ciudad1["weather"]
[17] descripcion=datos_clima1[0]["description"]
[18] imprime(Çiudad de destino:")
[19] imprime("Temperatura(Celsius) = - varibale(temperatura) + "Presiónatmosferica= - variable(presion) + "Humedad(en porcentaje) = - variable(humedad) + "Descripción = - variable(descripcion) + "Latitud = - origen_latitud+ "Longitud = - origen_longitud)
[20] De lo contrario:
[21] imprime("No se encontró la ciudad ")
[22] **Fin** del programa ciudad_dataset_origen

Proceso leer_csv

[1] Biblioteca csv
[2] ruta=**Escribir**
[3] abre (ruta, newline=") as archivo:
[4] csv_leer= lee(archivo)
[5] para row en csv$_{l}eer$:
[6] origen=row['origin']
[7] destino=row['destination']
[8] origen_latitud=row['origin_latitude']
[9] origen_longitud=row['origin_longitude']
[10] destino_latitud= row['destination_latitude']
[11] destino_longitud=row['destination_longitude']
[12] **Fin** del programa leer_csv

Proceso hora

[1] Biblioteca datetime
[2] sub algoritmo tiempo_de_puesta_de_sol_amanecer(da_horas):
[3] tiempo_local=datetime.utcfromtimestamp(da_horas)
[4] regresa tiempo$_{l}ocal.time()$
[5] **Fin** del programa hora

Proceso pruebas

```
[1] Biblioteca unittest
[2] sub algoritmo resta_grados(kelvin,celsius):
[3]     para linea en (kelvin, celsius):
[4]      si no es una instancia(linea, entero) y si no es una instancia(linea, de un numero float):
[5]        raise TypeError
[6]     regresa kelvin-celsius
[7] sub algoritmo verificar_archivo(clase):
[8]     ruta=Escribir
[9]     intenta:
[9]       archivo = abre(ruta)
[10]      imprime(archivo)
[11]      archivo.close()
[12]    excepcion FileNotFoundError:
[13]      imprime('No existe el archivo')
[14]      exit()
[15]   return clase
[16] clase TestArchivos(unittest.TestCase):
[17]    sub algoritmo test_resta(self):
[18]      self.assertEqual(resta_grados(273,15, 223), 50,14999999999998)
[19]      con self.assertRaises(TypeError):
[20]        resta_grados(273.15, "Python")
[21] sub algoritmo test_archivo(self):
[22]    si 'latitud_origen' igual a 'longitud_origen':
[23]     self.assertFalse(False)
[24] sub test_archivo(self):
[25]    ruta2=Escribir:
[26]    ruta_bien = ruta2 = Escribir:
[27]    self.assertEqual(verificar_archivo(ruta2),ruta_bien)
[28]    self.assertFalse(False)
[29] si __name__ igual "__main__":
[30]    unittest.main()
[31] Fin del programa pruebas
```

6. Mantenimiento y costo del programa.

Explica que mantenimiento crees que podría requerir en un futuro y explica cuanto crees que cobrarías por este pequeño reto y por futuros mantenimientos.

En un futuro puede requerir la actualización de datos, la robustez del programa para actuar de manera eficiente a problemas del usuario, la rapidez de consulta, la precisión del clima.

El cobro estimado para esta aplicación sería de $\$30,000$ pesos mexicanos. Para sacar este precio tomamos en cuenta gastos de luz, tiempo, pago estimado a programadores en México entre otras cosas.

7. ¿Cómo ejecutar el programa?

Antes de empezar:
Antes de correr el programa se debe de instalara varias cosas como lo son python y distintas librerías como lo son request, json y climage.

Instalación de python
Windows 10: `https://linuxfacil.mx/blog/python/como-instalar-python-en-windows-10/`

Mac:Por defecto Mac ya tiene instalado python, escribe **python –version** para revisar la versión de python que tienes, en caso de mostrarse una versión distinta a la 3 actualizala siguiendo el siguiente tutorial.

https://youtu.be/xxZnunBwC4U

Linux: https://www.datasource.ai/es/data-science-articles/como-instalar-python-3-en-linux-guia-para-inst

Instalación de las librerías
Las dos librerías que usaremos serán **request** y **pandas**

NOTA* Antes de instalar cualquier librería de python se debe instalar pip o anaconda.

Instalación de pip o anaconda: https://www.bing.com/videos/search?q=como+instalar+las+librerias+de+python+en+linux&docid=607991787026778153&mid=D931C2DE7210B0DF4908D931C2DE7210B0DF4908&view=detail&FORM=VIRE

Instalación de **request** desde windows, mac y linux: https://www.geeksforgeeks.org/how-to-install-requests-in-pyt

Instalación de pandas en linux y windows: https://es.acervolima.com/como-instalar-python-pandas-en-windows-y-l#:~:text=Para%20instalar%20Pandas%20en%20Linux%2C%20simplemente%20escriba%20el,ejecutar%20Pandas%20Environment%20en%20Python%3A%20pip3%20instalar%20pandas

Instalación de pandas en mac: https://www.geeksforgeeks.org/how-to-install-python-pandas-on-macos/

¿Cómo corremos el programa?

Nos dirigimos a la carpeta donde tenemos nuestro archivo .py, csv y pruebas unitarias, seleccionamos la opción **abrir en terminal**, nos aparecerá ya la dirección donde se encuentran nuestros archivos, ahora para ejecutar escribiremos:

Mac y Linux:**python3 nombreDeArchivo.py**

Windows: **python nombreDelArchivo.py**

Al ejecutar el archivo **ciudad** nos pedirá ingresar el nombre de la ciudad de origen y destino, el nombre se debe de ingresar completo y no abreviado. A continuación al darle enter nos proporcionará los datos pedidos y se terminará el programa.