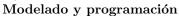


### Proyecto 01 Semestre 2021-1

Prof. José Galaviz Casas Ayud. María Ximena Lezama





Kevin Ariel Merino Peña $^1$  Armando Abraham Aquino Chapa $^2$ 9 de octubre de 2020

# Pseudocódigo

# CSVReader.py

Función 1: read_no_repeated_coordinates
Entrada: Nombre de un archivo (ruta) Salida : Lista de coordenadas no repetidas
<ul> <li>while Archivo(nombre dado) esté abierto do</li> <li>foreach renglones ← documento do</li> <li>if (latitud, longitud) no están en la lista then</li> <li>Agreagrlas</li> </ul>

### Función 2: read\_csv\_file

Entrada: Nombre de un archivo (ruta)
Salida: Lista de diccionarios con vuelos

1 try:

2 Abrir ruta

for  $linea \leftarrow archivo$  do

 $\downarrow$  linea  $\leftarrow$  lista

5 catch FileNotFoundError:

6 muesta Error, escribe una ruta válida

7 exit

8 catch FileExistsError:

muesta Error, archivo válido

10 exit

# Función 3: read\_headers

Entrada: Nombre de un archivo (ruta)

Salida : Lista de cabeceras

1 with:

 $\mathbf{2} \mid \text{lector} \leftarrow \mathbf{Leer} \ \mathbf{primera} \ \mathbf{linea} ( \ \text{ruta} )$ 

 $<sup>^1\</sup>mathrm{Número}$  de cuenta 317031326

 $<sup>^2</sup>$ Número de cuenta n

## main.py

#### Función 4: validate\_file Entrada: Nombre de un archivo (ruta) pasados como argumento al programa 1 if longiutd del argumento no es 2 then muestra: Error Debe indicar la ruta a un archivo csv 3 salir 4 5 if no coincide la extensión .csv then muestra: Error, sólo admito archivos csv salir $\mathbf{8}$ cabezera $\leftarrow$ nombres de listas admitidas entrada\_cabecera $\leftarrow$ read\_headers(argumento/1) if longitud(entrada\_cabecera) no es iqual a longitud(cabezera) then muestra: ERROR El archivo csv debe tener los siguientes encabezados: origin, destination, origin\_latitude, 11 origin\_longitude, destination\_latitude, destination\_longitude salir **12** 13 foreach cabeza in entrada\_cabezera do if cabeza no está en cabezera then 14 muestra: ERROR El archivo csv debe tener los siguientes encabezados: origin, destination, **15** origin\_latitude, origin\_longitude, destination\_latitude, destination\_longitude salir 16

### Función 5: run

 $\bf Entrada:$  Nombre de un archivo (ruta)

- 1 validate\_file(argumentos al correr el programa);
- 2 entradas ← read\_csv\_file(argumento al correr el programa)
- ${\tt 3}$  solicitudes\_no\_repetidas  $\leftarrow$  read\_no\_repeated\_coordinates(argumentos al iniciar)
- 4 foreach  $solicitud\ en\ solicitudes\_no\_repetidas\ do$
- 6 peticiones ← setdefault(solicitud, peticion)
- 7 foreach  $entrada\ en\ entradas\ \mathbf{do}$
- muesta: Datos del clima;) con formato bonito

# Weather.py

### Función 6: make\_api\_request\_by \_coordinates

Entrada: latitud, lontigud

Salida: llamada a función parse\_weather\_info

1 if contador > 59 then

 $\mathbf{2} \mid \text{contador} \leftarrow 0$ 

**3** esperar 1 minuto para continuar

4 get(url + latitud y longitud dadas)

 $\mathbf{5}$  contador  $\leftarrow$  contador + 1

#### Función 7: formato\_de\_horas

Entrada: Número de fecha y hora (unix)

Salida : cadena de texto con hora en formato 12 hrs

1 convierte\_fotante: numero dado

2 local\_timezone ← get\_localzone()()

3 local\_time ← fromtimestamp(flotante, local\_timezone)

4 regresar: local\_time con formato de 12 horas, (CODIGO DEL TIEMPO)

### Función 8: parse\_weather\_info

Entrada: respuesta en formato json

Salida : llamada a función parse\_weather\_info

1 try:

extraer información del archivo json con las llaves proporcionadas por la documentación de la API

з catch KeyError:

4 | regresar: Error, no se pudo consultar la información

5 regresar: El pronóstico del clima es: X , humedad: x

6 Temperatura actual: X°C, mínima: X°C, máxima: X°C Amanecer: X Puesta del sol: X

#### Análisis del problema y selección de la mejor alternativa.

Al ir analizando los diversos rubros que puede abarcar el problema, se determinó que lo más adecuado para resolverlo era obtener las siguientes clases:

#### ■ Clase Weather:.

- Esta clase sería la encargada de realizar las distintas peticiones al servidor para obtener todos los datos correspondientes al clima.
- Manejar los posibles errores, como el exceso de numero peticiones en un lapso de tiempo, para no obtener ningún problema con el servidor, y manejar los errores donde no es posible consultar la información
- Otorgar de un formato correcto y legible a la salida del programa

### Clase CSVReader

- Su función principal sería leer los archivos csv otorgados por el aeropuerto de la Ciudad de México.
- Preprocesar cierto tipo de información que si es esencial de la que es redundante o no tiene utilidad para nuestros objetivos.

•