UNIVERSIDAD AUTÓNOMA DE NUEVO LEÓN

Facultad de Ciencias Físico Matemáticas

Materia: Programación Básica

Nombre del Docente: Osvaldo Gonzalez

Equipo 3

Producto Integrador de Aprendizaje

Sofia Moreno

Maria Hernandez

Cinthia Melendez

Juan Garcia

Jennifer Casiano

CUADRO COMPARATIVO DE APIS

API	Tipo de datos ofrecidos	Ventajas	Desventajas	Link
INEGI	Datos estadísticos oficiales de México: población, economía, geografía, salud, vivienda, empleo, educación, entre otros.	- Gratuita - Datos actualizados periódicamente - Cobertura nacional y por entidad - Respaldo institucional - Permite acceso vía JSON/REST	- Documentación técnica limitada o poco clara para principiantes - Puede requerir familiaridad con claves y códigos de catálogo - No siempre tiene ejemplos prácticos de uso	https://www.inegi.org.mx/servicios/api_indicadores.html
NASA	imágenes astronómicas del día, imágenes de la tierra, fotos del Mars Rover, información sobre asteroides cercanos a la tierra, etc…	Es de uso gratuito, proporciona datos brutos e imágenes, se centra en datos científicos del espacio y la tierra.	Es compleja, algunos puntos proporcionan cantidades masivas de datos cuyo procesamiento y almacenamiento puede ser abrumador, límites de velocidad, algunos conjuntos de datos no se actualizan en tiempo real.	https://apidog-com.translate.goog/blog/how-to-use-nasa-api/?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es&_x_tr_pto=tc&_x_tr_hist=true
OpenWeatherMap	Clima actual y previsiones, datos meteorológicos y descripción general del tiempo.	Es gratuita, tiene cobertura global, su documentación es clara y tiene una gran variedad de datos.	Tiene limitaciones en el plan gratuito y requiere registro.	https://openweathermap.org/api
Aqicn (Air Quality Index)	Datos sobre la calidad del aire en el mundo y zonas específicas, ofrece mapas con índices señalados por colores distintivos	Es gratuita, tiene cobertura global, acepta 1000 solicitudes por segundo, ofrece documentación, sus datos se actualizan frecuentemente ya que están asociados a diferentes organizaciones (como la NASA)	Algunas estaciones podrían no reportar cierto tipo de contaminantes y se deberá validar si los datos existen antes de usarlos	https://aqicn.org/api/es/

PLANTEAMIENTO DEL PROBLEMA

El fenómeno que deseamos analizar es la calidad del aire en la Zona Metropolitana de Monterrey, debido a que es una región la cual ha experimentado en los últimos años un crecimiento significativo, tanto urbano como industrial. Esta situación ha traído consigo un aumento considerable en los niveles de contaminación atmosférica, esto ha derivado en una deteriorada calidad del aire.

Este tema nos resulta especialmente relevante debido al impacto directo que tiene en la salud de la población, ya que la exposición prolongada a los altos niveles de contaminantes puede causar o agravar enfermedades respiratorias, cardiovasculares y otras afecciones crónicas. Además, en los últimos años se ha observado un incremento en los días con mala calidad del aire, lo cual ha generado preocupación social, ambiental y política en la región.

Para llevar a cabo un análisis, es necesario contar con datos sobre la concentración de partículas contaminantes en la atmósfera, tales como PM10, PM25, ozono (O_3), dióxido de nitrógeno (NO_2), entre otros. Estos datos serán recolectados en diferentes puntos de la zona metropolitana, a lo largo de los días, para identificar patrones y gracias a esto podremos proponer estrategias de mejora.

DESCRIPCIÓN DE LA API ELEGIDA

Para obtener la información sobre la calidad del aire en distintas ciudades del Área Metropolitana de Monterrey (AMM), se utilizó la API del proyecto World Air Quality Index (WAQI), disponible en https://aqicn.org/map/world/. Esta API proporciona datos actualizados en tiempo real sobre los niveles de contaminación atmosférica en miles de estaciones alrededor del mundo.

La API permite consultar información detallada de cada ciudad, incluyendo el Índice de Calidad del Aire (AQI) y los valores específicos de varios contaminantes como PM2.5, PM10, monóxido de carbono (CO), dióxido de nitrógeno (NO_2), ozono (O_3) y dióxido de azufre (SO_2), entre otros. La consulta se realiza por medio de peticiones HTTP, enviando como parámetro el nombre de la ciudad y un token de autenticación personal que se obtiene al registrarse en el sitio web oficial del proyecto.

Los datos obtenidos de la API son devueltos en formato JSON, lo que facilita su tratamiento dentro del programa. A partir de esta información, el sistema desarrollado permite generar estadísticas como la media, moda y mediana del AQI, así como visualizar los resultados mediante gráficas de barras que representan los niveles de contaminación por ciudad o por contaminante.

Este enfoque permite tener una visión clara del estado del aire en cada zona del AMM, promoviendo la conciencia sobre la calidad ambiental y facilitando el análisis de tendencias de contaminación en la región.

DESCRIPCIÓN DE LA ESTRUCTURA DE DATOS UTILIZADA

Se implementó una estructura de datos anidada: una lista que contiene diccionarios, donde cada diccionario almacena información correspondiente a un municipio de la zona metropolitana.
Para facilitar el acceso a los datos, se prioriza incluir primero la información más relevante, como el nombre de la ciudad, el identificador (idx) asociado a la estación de monitoreo, y posteriormente, en un tercer par clave-valor, se incorporaron los datos obtenidos del JSON proporcionado por la API.

A continuación, se muestra la estructura de datos utilizada:

```
info = [
    {
        "ciudad": "NOMBRE DE LA CIUDAD",
        "udi": NUMERO DE LA ESTACION,
        "data": {
            "status": "ok",
            "data": {
                "aqi": INDICE DE CALIDAD DEL AIRE,
                "idx": IDX DE LA ESTACIÓN,
                "attributions": {FUENTES DE DATOS},
                "city": {INFORMACIÓN DE LA CIUDAD},
                "dominentpol": "CONTAMINANTE DOMINANTE",
                "iaqi": { VALORES INDIVIDUALES DE CONTAMINANTES},
                "time": {FECHA DE CUANDO SON LOS DATOS},
                "forecast": { # PRONOSTICO
                    "daily": {
                        "pm10": [ PRONOSTICO DE PM10 POR DIA ],
                        "pm25": [ PRONOSTICO DE PM25 POR DIA],
                        "uvi": [ INDICE UV POR DIA]
                    }
                },
                "debug": { INFORMACION DE SINCRONIZACIÓN }
            }
        }
    },
    ...
]
```

Justificación del Tratamiento de los Datos

Para este proyecto se usaron datos de la calidad del aire obtenidos desde la API del World Air Quality Index (WAQI), que proporciona información en tiempo real de diferentes ciudades. Se desarrolló un código en Python que hace una consulta a esa API y guarda los resultados en una lista con diccionarios para que fuera más fácil trabajar con ellos.

Primero, se revisaron los datos para asegurarse de que correspondieran a estaciones válidas. Si alguna ciudad no daba resultados o no estaba disponible, simplemente se omitía para no afectar el análisis.

Después se extrajeron los valores del índice de calidad del aire (AQI) y de contaminantes como PM2.5, PM10, NO2, entre otros. No todos los datos estaban siempre disponibles, así que se programó para que, si no encontraba uno, no marcara error.

También se clasificaron los valores del AQI con base en rangos que indican si el aire es bueno, regular o malo, para que fuera más fácil de entender.

Se hicieron gráficas de barras para comparar la contaminación entre ciudades, y otras líneas para ver el pronóstico diario. Además, se calcularon estadísticas como la media, mediana y moda del AQI para tener una idea general de cómo está el aire en la zona.

En resumen, el tratamiento de los datos fue pensado para organizarlos, analizarlos y presentarlos de forma clara y útil, usando herramientas básicas de programación.

Análisis Estadístico Realizado

Una vez que se obtuvieron los datos de la calidad del aire, se hizo un análisis básico para entender mejor la información. Lo primero fue calcular la **media** del AQI (Air Quality Index) en cada ciudad, lo que nos dio una idea del nivel promedio de contaminación en esos lugares.

También se calculó la **mediana**, que sirve para saber cuál es el valor que está justo en la mitad cuando se ordenan todos los datos. Esto es útil porque si hay valores muy altos o muy bajos, la mediana no se ve tan afectada como la media.

Además, se calculó la **moda**, que es el valor que más se repite, para ver si había un nivel de AQI que se presentaba con más frecuencia en alguna ciudad.

Estos cálculos se aplicaron principalmente al AQI, pero también se intentó aplicar a otros contaminantes como PM2.5 o PM10, dependiendo de la disponibilidad de los datos.

Con estos resultados se pudo tener una mejor idea del comportamiento de la calidad del aire en cada lugar y compararlos entre sí.

RESULTADOS

CATEGORÍA	RESULTADO	INTERPRETACIÓN
Municipio con AQI más alto	Cadereyta con 117	La calidad del aire es no saludable para grupos vulnerables. Este municipio presenta los niveles más altos de contaminación en la zona.
Municipio con AQI más bajo	Monterrey con 34	La calidad del aire es buena siendo la mejor registrada entre las ciudades analizadas.
Promedio del AQI (AMM)	58.5	La calidad del aire promedio en el área metropolitana es moderada.
Clasificación por número de ciudades	**Buena:** 4 **Moderada:** 5 **No saludable para grupos vulnerables:** 1 **No saludable:** 0 **Muy insalubre:** 0 **peligrosa:** 0	Permite identificar cuántas ciudades están en cada nivel de riesgo según el índice AQI. Útil para priorizar intervenciones.
Mediana del AQI	58	Valor central del conjunto de datos
Moda del AQI	calidad del aire buena	Valor que más se repite
Desviación estándar	595.16	Mide cuánto varía el AQI entre ciudades.
Varianza	24.39	Muestra la dispersión de los datos.

esto con datos del 13 de mayo del 2025

MINUTAS DE TRABAJO

Minutas de trabajo: **Dia 1**

Miembros:

Miembros	**Roles asignados**
Sofia Marisela Moreno Morales	Redactar el planteamiento.
Maria Fernanda Hernandez Grimaldo	Realizar análisis estadísticos.
Cinthia Jalyl Melendez Hernandez	Interpretar resultados y minutas.
Juan Manuel Garcia Navarro	Diseñar estructura de datos.
Jennifer Michelle Casiano Medina	Conectar con el API para descargar datos.

Minutas de trabajo **dia 1:**

Enfoque del código: Por decidirse

Objetivos para el día: Investigar diferentes APIs, para conocer qué tipos de datos ofrecen y conocer tanto sus ventajas, como sus desventajas.

Acuerdos tomados: Investigar cada uno un API pública y llenar un cuadro comparativo con su información, y elegir la que más nos convenga.

Dificultades encontradas: Aún no conocemos del todo lo que es una API y cómo la vamos a manejar.

Próximos pasos: Después de elegir el API, plantear un problema relevante y establecer qué datos necesitamos obtener para nuestro programa.

Minutas de trabajo **día 2:**
Enfoque del código: Analizar la calidad del aire.

Objetivos para el día: Crear nuestras cuentas de github y generar nuestros tokens para usar los datos de la API.

Acuerdos tomados: Crear un código compartido y elegimos qué datos vamos a descargar para la creación nuestra propia estructura.

Dificultades encontradas: No conocemos cómo vamos a acceder a los datos, en el código.

Próximos pasos: Investigar cómo acceder a los datos.

Minutas de trabajo **día 3:**

Objetivos para el día: Realizar el código.

Acuerdos tomados: Delimitamos nuestro análisis al área metropolitana de Monterrey y decidimos la estructura con la que trabajamos nuestro código y los datos en crudo.

Dificultades encontradas: solo pudimos acceder a una estación.

Próximos pasos: Revisar cómo acceder a más estaciones, para así tener información que nos permita hacer estadísticas.

Minutas de trabajo **día 4:**

Objetivos para el día: Terminar el código e ir agregando nuestros avances al entregable 1.

Acuerdos tomados: Cada integrante va llenando el entregable con la parte que nos tocó.

Dificultades encontradas: tuvimos un problema pequeño a la hora de correr el código, pero se arregló rápidamente.

Próximos pasos: Entregar el entregable 1 en Teams.

GUION

Buen día, nuestro equipo está conformado por Sofía Moreno, María Hernández, Cinthia Meléndez, Juan Manuel y Jennifer Casiano y el día de hoy les vamos a presentar nuestro proyecto eh sobre determinar los índices de la calidad del aire en la zona metropolitana de monterrey por medio de un script.

A continuación procederemos a correr el código y se desprende un menú de opciones. En la primera opción que dice general información del área metropolitana, pues en esta parte se va a llevar a cabo esta obtener la información del área metropolitana por medio de una consulta, directamente en los datos de la API la cual también estaremos utilizando en los siguientes puntos.
La segunda opción, nuestro menú que se llama guardar la información nos permite generar un archivo de texto en el que se guardan información en crudo.
La tercera opción como su nombre lo dice, vamos a mostrar la información sobre la calidad del aire, o sea, nos va a decir en el municipio qué tal está la calidad y de ahí vamos a sacar el promedio, la moda y vamos a hacer los gráficos.

Tenemos otro módulo que te muestra el índice de la cantidad de contaminantes que hay en cada ciudad y puede seleccionar el que te interesa buscar en este caso pues tenemos varios tipos de contaminantes, por ejemplo, el primero que es el monóxido de carbono y si lo seleccionamos, pues ponemos el número uno y nos arroja los datos en la pantalla de del CMD y también nos arroja un un gráfico
En la opción número 5 que dice buscar pronóstico por

ciudad de pm10 y pm25. Lo que se realiza es mostrar el pronóstico de para una de las ciudades que elijamos para las partículas pm10 y pm25 durante los próximos días y vamos a seleccionarlo y se genera un menu de opciones de cada para cada municipio. En este caso vamos a pues ver cómo va a estar cadereyta en los próximos días y me genera este gráfico en donde pues se puede notar cómo va a ir el pronóstico para estas particulas. Se realiza para el pm10 y se realiza para las partículas pm25.
Y por último, en nuestro menú tenemos la opción de terminar que es la cual lo cierra el programa y nos muestra todas las fuentes utilizadas para nuestro script. Muchas gracias por su atención.

ALGORITMO PIA “API DE LA CALIDAD DEL AIRE EN LA ZONA METROPOLITANA DE MONTERREY”

Modulo generar_fuentes(info):

1.Definición de variables
archivo: archivo
texto, patron,fuente, ruta_archivo: cadena
fuentes:conjunto
paginas: lista
//abrir archivo y leer contenido
ruta_archivo= ‘C:\Users\juang\Documents\FCFM\2do Semestre\Programacion Basica\PIA\info_amm.txt'
archivo=abrir(ruta_archivo, ‘lectura’)
texto = leer (archivo)
cerrar (archivo)

//importar expresiones regulares
patron = "https?://[\w(\.|\-)]+/"
//encontrar todas las paginas
paginas = BuscarTodasCoincidencias(patron, texto)
fuentes = ConvertirListaAConjunto(paginas)
Para cada fuente en fuentes
    Hacer
        imprimir(fuente)
    fin_hacer
fin_para
fin del modulo

Modulo ciudad_pronostico(info,ciudades)

1.Definición de variables
city: entero
pronostico: diccionario
2.Repetir con ciudad desde 1 hasta (longitud(ciudades))
    imprimir(f"{ciudad+1}. {ciudades[ciudad]}")
3.fin_repetir
4.Imprimir(“¿de que ciudad desde saber su pronostico?:”)
5.Leer city
6. pronostico = info[city-1]["data"]["data"]["forecast"]["daily"]
7.generar_grafico_pronostico(pronostico)
8.fin_modulo

Modulo generar_grafico_pronostico(pronostico)

1.Declaración de variables
etiquetas,valores: lista vacía
2.Para cada clave en pronostico
si clave <> ‘uvi’
entonces
//Importar librería matplotlib.pyplot
Para cada day en pronostico[clave]
Agregar(etiquetas, day['day'])
Agregar(valores, day['avg'])
fin_para
//configuración del gráfico
CrearFigura(ancho=12, alto=6) // Tamaño del gráfico

DibujarLinea(etiquetas, valores, con_marcadores=VERDADERO)
//Personalizar el gráfico

EstablecerTitulo(f"Pronóstico de partículas {clave}")

EstablecerEjeX("Día")

EstablecerEjeY(clave)

RotarEtiquetasEjeX(angulo=45, alineacion="derecha")

ActivarCuadricula(VERDADERO)

AjustarLayout() // Optimizar espacios
fin_si
3. //mostrar el gráfico
4. Mostrar grafico ()
5.fin_para

```
Modulo generar_grafico_de_barras(info, indicador)

//Importar librería matplotlib.pyplot as plt

1.Definición de variables
etiquetas, valores: lista vacía
2.Para cada ciudad en info
        city=ciudad[“ciudad”]
        Agregar cada (city) a etiquetas
3.fin_para
4. //crear lista de valores
5.Si indicador == ‘aqi’
        entonces
                Para cada ciudad en info
                        val = ciudad ["data"]["data"]["aqi"]
                        Agregar cada (val) a valores
                fin_para
        si_no
                Para cada ciudad en info
                        Si indicador en ciudad["data"]["data"]["iaqi"].keys():
                                entonces
                                        val= ciudad["data"]["data"]["iaqi"][indicador]["v"]
                                si_no
                                        val=0
                                        Agregar cada (val) a valores
                        fin_si
                fin_para
        fin_si
//Generar gráfico
6.EstablecerTamaño(12,6)
7.EstablecerGrafico(etiquetas, valores, color=naranja
8.EstablecerTitulo(“Indice de la calidad del aire por ciudad en AMM”)
9.EstablecerEjeX(Rotacion 45)
10.MostrarGrafico
11.fin_modulo
```

Modulo buscar_aqi(city_name)
//Importar librería requests en codigo

1.Declaracion de variables
registro:diccionario
url, ruta_token:cadena
2.ruta_token= "C:\Users\juang\Documents\FCFM\2do Semestre\Programacion Basica\PIA\token_aqicn.txt"
3.url= “https://api.waqi.info/feed/{city_name}/?token={token}”
4.abrir(ruta_token, lectura)
5.respuesta=abrir(url)
6.//data=respuesta.json()
7.Si data[‘data] <> ‘Unknown station”
    entonces
        udi=data[‘data’][‘idx’]
    si_no
        udi= None
        Imprimir(“Estacion desconocida”)
8.fin_si
9.registro= {"ciudad": city_name, "udi": udi, "data": data}
10.Retornar registro
11.fin_modulo

Modulo clasificar _aqi(aqi)

1.Declaracion de variables
2.Si (aqi<=50)
    entonces
        Imprimir(“La calidad del aire es buena”)
    si_no
    si (51 <= aqi <=100)
        entonces
            Imprimir (“La calidad del aire es moderada”)
        si_no
        si (100 < aqi <= 150)
            entonces
                Imprimir(“La calidad del aire no es saludable para grupos sensibles”)
            si_no
            si (150 < aqi <= 200)
                entonces
                    Imprimir(“La calidad del aire no es saludable”)
                si_no
                si (200 < aqi <= 300)
                    Entonces
                        Imprimir(“La calidad del aire no es nada saludable”)

```
                                        si_no
                                                Imprimir(“La calidad del aire es muy peligrosa”)
                                    fin_si
                                fin_si
                        fin_si
                3.fin_si
4.fin_modulo


Modulo mostrar_aqi_total(info)
//Importar librería statistics

1.Declaracion de variables
moda, datos: lista vacía
aqi_min=0
2.Para cada ciudad en info
        Imprimir({ciudad['data']['data']['aqi']}\t{ciudad['ciudad']}
        {clasificar_aqi(ciudad['data']['data']['aqi'])}")
3.Agregar (ciudad['data']['data']['aqi']) a datos
4.//Calcular la zona mayor
Si (ciudad ['data']['data']['aqi'] > aqi_max)
        entonces
                city_max = ciudad['ciudad']
                aqi_max = ciudad['data']['data']['aqi']
fin_si
5.//calcular la zona menor
Si aqi_min==0
        entonces
                aqi_min= ciudad['data']['data']['aqi']
        si_no
        si
                entonces
                        Si (ciudad ['data']['data']['aqi'] < aqi_min)
                                entonces
                                        city_min = ciudad['ciudad']
                                        aqi_min = ciudad['data']['data']['aqi']
                        fin_si
        fin_si
6.fin_si
7.Agregar (clasificar_aqi(ciudad['data']['data']['aqi'])) a moda
8.Imprimir("Promedio: {statistics.mean(datos)}")
9.Imprimir("Moda: {statistics.mode(moda)}")
10.Imprimir(“Mediana: {statistics.median(datos)}")
11.Imprimir(“Mediana: {statistics.median(datos)}")
12.Imprimir("Aqi mas alto: {max(datos)} en {city_max}")
13.Imprimir("Aqi mas bajo: {min(datos)} en {city_min}")
14.Imprimir("La varianza de es {statistics.variance(datos)}")
15.Imprimir("La desviacion estandar es de {statistics.stdev(datos)}")
16.indicador=’aqi’
```

17.// generar_grafico_de_barras(info,indicador)
18.fin_modulo

Modulo mostrar_por_contaminante(info)
1.Delcaración de variables
while=true:booleano
cont:entero
2.Imprimir(“Que contaminante desea buscar en el AMM:\n1. co: Monóxido de carbono.\n2. no2: Dióxido de nitrógeno.\n3. o3: Ozono.\n4. pm10\n5. pm25\n6. so2: Dióxido de azufre.\n7. Terminar de buscar\nSeleccionaste: "))
3.Leer(cont)
4.Si (cont==1)
entonces
cont=’co’
si_no
si (cont==2)
entonces
cont=’no2’
si_no
si (cont==3)
entonces
cont=’o3’
si_no
si (cont==4)
entonces
cont=’pm10’
si_no
si (cont==5)
entonces
cont=’pm25’
si_no
si (cont==6)
entonces
cont=’so2’
si_no
Imprimir(“No existe esa opción”)
fin_si
fin_si
fin_si
fin_si
fin_si
5.fin_si
6.Para cada ciudad en info
si (cont en ciudad['data']['data']['iaqi'])
entonces
Imprimir("{ciudad['data']['data']['iaqi'][cont]['v']}\t{ciudad['ciudad']}")
si_no
Imprimir(“"Dato no disponible en {ciudad['ciudad']}")

fin_si
7.fin_para
8.// generar_grafico_de_barras(info,cont)

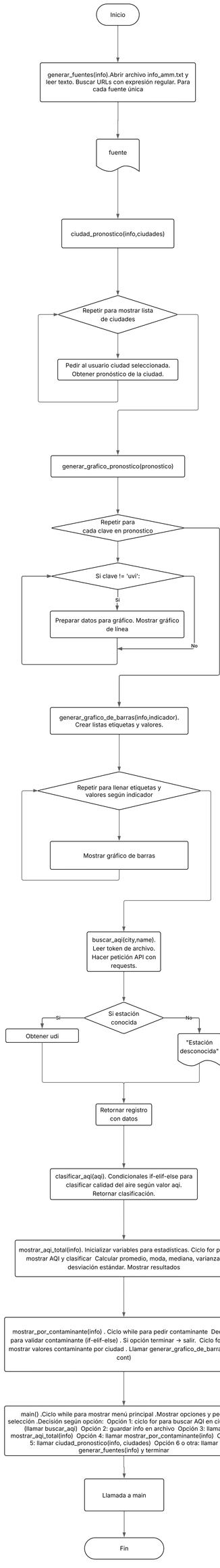
Modulo Principal
1.Declaración de variables
info:diccionario
op:entero
while:booleano

2.info=[]

3. ciudades = ['santa-catarina','monterrey','guadalupe','apodaca','juarez','cadereyta','garcia','san-nicolas-de-los-garza','general-escobedo','san-pedro-garza-garcia']
4.Hacer
Imprimir("Ingrese alguna opcion\n1. Generar info del AMM\n2. Guardar la informacion\n3. Mostrar informacion sobre la calidad del aire\n4. Buscar informacion por contaminate\n5. Buscar pronostico por ciudad de pm10 y pm25\n6. Terminar\nSeleccionado: "))
Leer(op)
//Busca las ciudades ingresadas en la variables ciudades
Si (op==1)
entonces
Para cada ciudad en ciudades
datos=buscar_aqi(ciudad)
//Agregar (datos) a info
si_no
si (op==2)
entonces
//guardar la info en “crudo”
info_amm=”C:\Users\juang\Documents\FCFM\2do Semestre\Programacion Basica\PIA\info_amm.txt”
abrir(info_amm,escritura)
//info_amm.write(str(info))
cerrar info_amm
si_no
si (op==3)
entonces
//buscar información sobre la calidad del aire
llamar mostrar_aqi_total(info)
si_no
si (op==4)
entonces
//buscar información por contaminante
llamar mostrar_por_contaminante(info)
si_no
si (op==5)
entonces

```
                                        //dar información y graficar el pronostico de pm10 y
                                        pm25
                                            llamar ciudad_pronostico(info,ciudades)
                                        si_no
                                        Imprimir("La informacion presentada fue tomada de:
                                            ")
                                        llamar generar_fuentes(info)
                                    fin_si
                                fin_si
                            fin_si
                        fin_si
                fin_si
5.Mientras while=true
6.fin_modulo


Programa principal
1.Llamar Prinicipal
2.fin
```


Inicio
generar_fuentes(info).Abrir archivo info_amm.txt y leer texto. Buscar URLs con expresión regular. Para cada fuente única
fuente
ciudad_pronostico(info,ciudades)
Repetir para mostrar lista de ciudades
Pedir al usuario ciudad seleccionada. Obtener pronóstico de la ciudad.
generar_grafico_pronostico(pronostico)
Repetir para cada clave en pronostico
Si clave != 'uvi':
si
Preparar datos para gráfico. Mostrar gráfico de línea
No
generar_grafico_de_barras(info,indicador). Crear listas etiquetas y valores.
Repetir para llenar etiquetas y valores según indicador
Mostrar gráfico de barras
buscar_aqi(city,name). Leer token de archivo. Hacer petición API con requests.
Si estación conocida
Si
No
Obtener udi
"Estación desconocida"
Retornar registro con datos
clasificar_aqi(aqi). Condicionales if-elif-else para clasificar calidad del aire según valor aqi. Retornar clasificación.
mostrar_aqi_total(info). Inicializar variables para estadísticas. Ciclo for para mostrar AQI y clasificar Calcular promedio, moda, mediana, varianza, desviación estándar. Mostrar resultados
mostrar_por_contaminante(info) . Ciclo while para pedir contaminante Decisión para validar contaminante (if-elif-else) . Si opción terminar → salir. Ciclo for para mostrar valores contaminante por ciudad . Llamar generar_grafico_de_barras(info, cont)
main() .Ciclo while para mostrar menú principal .Mostrar opciones y pedir selección .Decisión según opción: Opción 1: ciclo for para buscar AQI en ciudades (llamar buscar_aqi) Opción 2: guardar info en archivo Opción 3: llamar mostrar_aqi_total(info) Opción 4: llamar mostrar_por_contaminante(info) Opción 5: llamar ciudad_pronostico(info, ciudades) Opción 6 o otra: llamar generar_fuentes(info) y terminar
Llamada a main
Fin