Universidad de San Carlos de Guatemala
Facultad de ingeniería
Escuela de Ciencias y Sistemas

Manual Técnico

Laboratorio de Arquitectura
de Computadores y Ensambladores 1
Sección A
Grupo 5

Guatemala, 26 de junio de 2024

Introducción

El cambio climático es uno de los mayores desafíos globales en estos últimos tiempos, caracterizado principalmente por el aumento de la temperatura, alteraciones en los patrones de precipitación y eventos climáticos extremos más frecuentes e intensos. En este contexto, el monitoreo continuo y preciso de las condiciones meteorológicas locales es esencial para comprender y mitigar estos efectos. La implementación de una estación meteorológica proporciona datos cruciales para la investigación climática, la gestión de recursos naturales y la protección de las comunidades frente a eventos adversos.

El objetivo general de este proyecto es aplicar los conocimientos adquiridos en el curso sobre el lenguaje ensamblador para desarrollar una solución que permita monitorear y analizar datos meteorológicos. Específicamente, se busca realizar una combinación de operaciones aritméticas para resolver problemas relacionados con el procesamiento de datos, consolidar los conocimientos de escritura y lectura de archivos, y unificar lo aprendido sobre ensambladores para realizar la solución completa del problema. Este proyecto no solo se busca fortalecer las habilidades técnicas en ensambladores, sino que también poder dar una solución que contribuya a una causa global importante en lo que es el cambio climático.

El proyecto se divide en tres partes importantes que se detallarán en el presente manual.

Controlador ARM

- **Calculator**
  Esta clase es utilizada para realizar varias operaciones estadísticas sobre los datos almacenados en archivos; en esta clase se realiza la importación de “subprocess”, esto permite ejecutar comandos del sistema operativo desde Python

```
1    import subprocess as sp
2
3    class Calculator:
4
```

El método “init”, constructor que inicializa un diccionario llamado “data”.

```
    def __init__(self):
        self.data = {
            'promedio': -1,
            'mediana': -1,
            'desviacion_estandar': -1,
            'maximo': -1,
            'minimo': -1,
            'moda': -1,
            'contador': -1
        }
```

El método “get_air”, devuelve un diccionario con los conteos de aire bueno y malo.

```
def get_air(self):
    air_data = {
        'contador_bueno': self.good_air(),
        'contador_malo': self.bad_air()
    }
    return air_data
```

Métodos para estadísticas, cada uno de estos métodos ejecuta un comando del sistema para calcular un cálculo en específico sobre un archivo de datos y devuelve el resultado:

Promedio

```
def get_average(self, variable):
    result = sp.run(['./ARM/average', f'ARM/DB/{variable}.txt'], text=True, capture_output=True)
    return int(result.stdout)
```

Mediana

```
def get_median(self, variable):
    f = open(f'ARM/DB/{variable}.txt', 'r')
    string = f.read()
    f.close()
    array = string.split()
    array.pop()
    array.sort()
    array.append('$')
    f = open('ARM/DB/sorted.txt', 'w')
    for data in array:
        f.write(str(data) + '\n')
    f.close()
    result = sp.run(['./ARM/median'], text=True, capture_output=True)
    return round(float(str(result.stdout).replace('\x00', '')), 2)
```

Desviación Estándar

```
def get_stnd_dev(self, variable):
    sqaverage = sp.run(['./ARM/sqaverage', f'ARM/DB/{variable}.txt'], text=True, capture_output=True).st
    average = sp.run(['./ARM/average', f'ARM/DB/{variable}.txt'], text=True, capture_output=True).stdout
    sqaverage = int(sqaverage)
    average = int(average)
    variance = sp.run(['./ARM/variance', f'{sqaverage}*{average}$'], text=True, capture_output=True).std
    variance = int(variance)
    stnd_dev = sp.run(['./ARM/SquareRoot'], input=f'{variance}\n', text=True, capture_output=True).stdou
    return round(float(str(stnd_dev).replace('\x00', '')), 2)
```

Máximo

```
def get_max(self, variable):
    result = sp.run(['./ARM/max', f'ARM/DB/{variable}.txt'], text=True, capture_output=True)
    return int(result.stdout)
```

Así mismo se podrán visualizar cada uno de los cálculos Estadísticos restantes, cómo lo son el Mínimo y Moda.

ARM

- **SquareRoot.s**
  En este apartado ya vemos lo que es código escrito en lenguaje ensamblador ARM, específicamente este apartado realiza las siguientes tareas: leer un número entero de 3 dígitos desde la entrada estándar lo convierte de ASCII a un entero, calcula la raíz cuadrada de ese número, convierte el resultado a un número con punto decimal y lo imprime en la salida estándar. También incluye una función “itoa” para convertir un número entero en una cadena de caracteres.

  Sección de Datos

```
.data
input:  .space 12           // Espacio para el número de entrada (3 dígitos + null terminator)
buffer: .space 32          // Espacio para la salida del número con formato
```

  Sección de Texto

```
.text
_start:
    // Leer número de la entrada estándar (stdin)
    mov x0, 0                  // stdin (descriptor de archivo 0)
    ldr x1, =input             // buffer para almacenar el número leído
    mov x2, 12                  // leer hasta 4 bytes (3 dígitos + null terminator)
    mov x8, 63                 // syscall: read
    svc 0
```

  Conversión de ASCII a entero

```
        ldr x1, =input              // dirección del buffer de entrada
        mov w2, 0                   // inicializar el acumulador
    convert_loop:
        ldrb w3, [x1], #1           // cargar el siguiente byte del buffer y avanzar
        cmp w3, 10                   // ¿es el byte null terminator?
        beq conversion_done         // si es null terminator, salir del bucle
        sub w3, w3, '0'             // convertir ASCII a entero
        mov w4, 10                  // multiplicador para convertir ASCII a entero
        mul w2, w2, w4              // multiplicar el acumulador por 10
        add w2, w2, w3              // sumar el nuevo dígito
        b convert_loop              // repetir el bucle
    conversion_done:
```

Conversión de entero a flotante y cálculo de la raíz cuadrada

```
scvtf s2, w2              // convertir entero a flotante
fsqrt s0, s2              // calcular la raíz cuadrada
fcvt d0, s0               // mover el resultado a un registro de punto flotante de doble p

mov w3, 1000              // mover a una variable entera
scvtf d3, w3              // convertir a flotante
fmul d0, d0, d3           // multiplicar por 1000.0 para convertir a doble precisión
fmul d0, d0, d3           // multiplicar por 1000.0 para convertir a doble precisión
fcvtzu w1, d0            // convertir a entero y redondear
```

Conversión del número entero a texto usando “itoa”

```
//convertir número entero a texto usando itoa
mov w0, w1                  // número entero
sxtw x0, w0
ldr x1, =buffer             // dirección del buffer de salida
bl itoa                    // convertir entero a cadena
```

Función itoa

```
itoa:
    // Guardar registros de retorno
    stp x29, x30, [sp, #-16]!  // Guardar x29 y x30 en la pila
    mov x29, sp                // Actualizar el puntero de marco de pila

    // Inicialización
    mov x2, #10          // base = 10
    mov x3, x1           // Puntero de inicio del buffer
    mov x4, x0           // Número original
    mov x6, 0           // contador de digitos
```

- **Average.s**
  Esta parte de código tiene como funciones principales leer un argumento de línea de una cadena, abrir un archivo, procesar los números en el archivo calculando el promedio de los números presentes y convertir el resultado a una cadena de caracteres e imprimiendo en la salida estándar.

Sección de Definición de Datos

```
.bss
arg1: .space 32          // space for 32 caracters
output: .skip 12         // space for 12 bytes
buffer: .skip 1024       // space for 1024 bytes

.data
newline: .asciz "\n"     // new line
```

Copia del Argumento

```
loop_argv:
    ldrb w3, [x0, x2]    // load byte
    cmp w3, 0            // if null
    beq end_loop_argv    // goto end_loop_argv
    strb w3, [x1, x2]    // store byte
    add x2, x2, 1        // increment counter
    b loop_argv          // goto loop_argv
```

Apertura del Archivo

```
open_file:
    // open file
    mov x0, -100         // open
    ldr x1, =arg1           // filename address
    mov x2, 0            // O_RDONLY
    mov x8, 56           // openat
    svc #0               // syscall
    mov x9, x0           // store file descriptor
```

Lectura del Archivo

```
// read file
mov x0, x9           // file descriptor
ldr x1, =buffer      // buffer address
mov x2, 1024         // size address
mov x8, 63           // read
svc 0                // syscall
```

Conversión de Números

```
loop:
    ldrb w2, [x1]       // load byte
    cmp w2, 36          // if $
    beq convert         // goto convert
    cmp w2, 10          // if \n
    beq skip_loop       // goto skip_loop
    sub w2, w2, 48      // convert to int
    uxtb x2, w2         // convert to 64 bit
    mul x3, x3, x4      // multiply by base
    add x3, x3, x2      // add digit
    add x1, x1, 1       // increment address
    b loop              // goto loop
```

Salto de Línea y Suma

```
skip_loop:
    add x6, x6, 1       // increment counter
    add x1, x1, 1       // increment address
    add x5, x5, x3      // add to sum
    mov x3, 0           // reset number
    b loop              // goto loop
```

Conversión del Resultado

```
convert:
    udiv x5, x5, x6     // calculate average
    ldr x1, =output     // output address
    mov x2, x5          // number to convert
    mov x3, 10          // base number
    mov x4, 0           // number size = 0
```

- **Max.s**
  La función de máximo en el código incluye varias operaciones como la copia de una cadena, la lectura de un archivo, la conversión de números y la impresión de resultados.

  Copia de arg[1] a arg1, incluye un bucle para copiar byte a byte hasta encontrar un byte nulo.

```
loop_argv:
    ldrb w3, [x0, x2]   // load byte
    cmp w3, 0           // if null
    beq end_loop_argv   // goto end_loop_argv
    strb w3, [x1, x2]   // store byte
    add x2, x2, 1       // increment counter
    b loop_argv         // goto loop_argv

    // add eof
end_loop_argv:
    mov w0, 0           // add eof
    strb w0, [x1, x2]   // store byte
```

  Abrir el archivo

```
open_file:
    // open file
    mov x0, -100        // open
    ldr x1, =arg1          // filename address
    mov x2, 0           // O_RDONLY
    mov x8, 56          // openat
    svc #0              // syscall
    mov x9, x0          // store file descriptor
```

  Configurar Variables

```
set_variables:
    ldr x1, =buffer     // buffer address
    mov x3, 0           // number size = 0
    mov x4, 10          // base number
    mov x5, 0           // sum
    mov x6, 0           // counter
```

Conversión de Números, con un bucle principal para procesar cada byte del buffer

```
    //casting number
loop:
    ldrb w2, [x1]       // load byte
    cmp w2, 36          // if $
    beq convert         // goto convert
    cmp w2, 10          // if \n
    beq skip_loop       // goto skip_loop
    sub w2, w2, 48      // convert to int
    uxtb x2, w2         // convert to 64 bit
    mul x3, x3, x4      // multiply by base
    add x3, x3, x2      // add digit
    add x1, x1, 1       // increment address
    b loop              // goto loop
```

Establecer Máximos

```
set_max:
    mov x5, x3          // set max
    mov x3, 0           // reset number
    b loop              // goto loop
```

Obtener el tamaño del número

```
get_size:
    udiv x5, x5, x3     // remove last digit
    add x4, x4, 1       // increment size
    cmp x5, 0           // if number != 0
    bne get_size        // goto get_size
```

Establecer ASCII final

```
setascii:
    ldrb w9, [x1]       // load left digit
    add w9, w9, 48      // set ascii
    strb w9, [x1]       // store ascii
    add x1, x1, 1       // increment addr
    sub x6, x6, 1       // decrement iter
    cmp x6, 0           // if iter != 0
    bne setascii        // goto setascii
```

- **Median.s**
  En este apartado de código se calculará la mediana como función principal.

  Abriendo el Archivo

```
mov x0, -100
ldr x1, =file
mov x2, 0
mov x8, 56
svc 0
mov x9, x0
```

  Contar los Números

```
count_loop:
    // Contar la cantidad de numeros
    ldrb w3, [x1, x2]          // cargar el siguiente byte del buffer y avanzar
    add x2, x2, 1              // avanzar al siguiente byte
    cmp w3, #10                // comparar si es un salto de linea
    csinc x9, x9, x9, ne       // si es un salto de linea es correcto incrementa en 1 al x9, de no ser asi
    cmp w3, #36                // comparar si es el final del buffer
    bne count_loop            // si no es el final del buffer regresa a count_loop
```

Calcular la Mediana números impares

```
odd_number:
    // sacamos la mediana para cantidad par de numeros
    mov x3, 2                  // dividir entre 2 (posicion del numero buscado)
    udiv x0, x2, x3             // dividir entre 2 (posicion del numero buscado)
    //------------------variables de entrada------------------
    //x0 contiene la posicion del numero buscado
    add x0, x0, 1              // sumar 1 para obtener la posicion del numero buscado
    //x1 contiene la direccion del buffer
    mov x2, 0                  // posicion de caracter
    mov x3, 0                  // lector de caracter
    mov x9, 1                  // contador de numeros
    bl search_position          // buscar la posicion del numero
```

Calcular la Mediana para Números pares

```
even_number:
    // sacamos la mediana para cantidad par de numeros
    mov x3, 2                  // dividir entre 2 (posicion del numero buscado)
    udiv x0, x2, x3             // dividir entre 2 (posicion del numero buscado)
    //------------------variables de entrada------------------
    //x0 contiene la posicion del numero buscado
    //x1 contiene la direccion del buffer
    mov x2, 0                  // posicion de caracter
    mov x3, 0                  // lector de caracter
    mov x9, 1                  // contador de numeros
    bl search_position          // buscar la posicion del numero
```

- **Mini.s**
  En este apartado de código se tiene como función principal obtener el mínimo de los datos ingresados.

Cargar la dirección

```
_start:
    //argv[1] address
    ldr x0, [sp, 16]    // load address of argv[1]
    ldr x1, =arg1       // load address of arg1
    mov x2, 0           //
```

Abrir el Archivo

```
mov x0, -100        // Abrir archivo
ldr x1, =arg1       // Dirección del nombre del archivo
mov x2, 0           // O_RDONLY
mov x8, 56          // openat
svc #0              // syscall
mov x9, x0          // Almacenar descriptor de archivo
```

Convertir el Número

```
loop:
    ldrb w2, [x1]       // Cargar byte
    cmp w2, 36          // Si es $
    beq convert         // Saltar a convert
    cmp w2, 10          // Si es \n
    beq skip_loop       // Saltar a skip_loop
    sub w2, w2, 48      // Convertir a entero
    uxtb x2, w2         // Convertir a 64 bits
    mul x3, x3, x4      // Multiplicar por base
    add x3, x3, x2      // Añadir dígito
    add x1, x1, 1       // Incrementar dirección
    b loop              // Repetir loop
```

Obtener el Mínimo

```
set_min:
    mov x5, x3          // Establecer mínimo
    mov x3, 0           // Restablecer número
    b loop              // Repetir loop
```

Obtener el tamaño

```
get_size:
    udiv x5, x5, x3     // Eliminar último dígito
    add x4, x4, 1       // Incrementar tamaño
    cmp x5, 0           // Si número != 0
    bne get_size        // Repetir get_size
```

Imprimir el Resultado

```
print:
    mov x0, 1           // stdout
    ldr x1, =output     // Cargar cadena
    mov x2, x4          // Tamaño de cadena
    mov x8, 64          // Número de syscall write
    svc 0               // syscall
```

- **Mode.s**
  Este apartado de código tiene como función principal leer el contenido de un archivo especificado en la línea de comandos, extrae números enteros de ese archivo, encuentra la moda entre ellos y lo imprime en la salida estándar.

  Cargar el nombre del Archivo

```
ldr x0, [sp, 16]    // Cargar dirección de argv[1]
ldr x1, =arg1       // Cargar dirección de arg1
mov x2, 0           // Inicializar contador
```

Leer el Archivo

```
mov x0, x9          // Descriptor de archivo
ldr x1, =buffer     // Dirección del buffer
mov x2, 1024        // Tamaño del buffer
mov x8, 63          // Número de syscall read
svc 0               // syscall
```

Extraer y Almacenar Números

```
loop:
    ldrb w2, [x1]       // Cargar byte
    cmp w2, 36          // Si es $
    beq find_mode       // Ir a find_mode
    cmp w2, 10          // Si es \n
    beq store_number    // Ir a store_number
    cmp w2, 0           // Si es null (fin del buffer)
    beq find_mode       // Ir a find_mode
    sub w2, w2, 48      // Convertir a entero
    uxtb x2, w2         // Convertir a 64 bits
    mul x3, x3, x4      // Multiplicar por base
    add x3, x3, x2      // Añadir dígito
    add x1, x1, 1       // Incrementar dirección
    b loop              // Repetir loop
```

Encontrar la Moda

```
find_mode:
    mov x12, 0          // Frecuencia del modo
    mov x13, 0          // Valor del modo
    mov x14, 0          // Índice del bucle exterior
```

App.py

En este apartado se crea un servidor web usando el framework Flask, este expone varias rutas que proporcionan datos climáticos como temperatura, humedad, velocidad del viento, presión barométrica, calidad del aire, y si es de día o de noche.

Librerías y Módulos necesarios:

Flask: Framework web para Python.

Flask-CORS: Extensión para Flask que permite CORS.

Climate: Módulo personalizado que probablemente maneja la lectura de datos climáticos.

```
from flask import Flask, request, jsonify
from flask_cors import CORS
from Climate import Climate
from ARM.Calculator import Calculator
```

Inicialización de Objetos

```
# Climate reader
new_climate = Climate()
new_climate.read_climate()

# Data calculator
calc = Calculator()
```

Configuración de la Aplicación de Flask

```
app = Flask(__name__)
CORS(app)
```

Definición de Rutas

Se definen varias rutas para obtener diferentes tipos de datos climáticos. Cada ruta utiliza el método calc.get_data() para obtener los datos requeridos y devolverlos en formato JSON.

Devuelve la Temperatura

```
@app.route('/Temperatura')
def get_temp():
    return jsonify(calc.get_data('temp'))
```

Devuelve la Humedad

```
@app.route('/Humedad')
def get_hum():
    return jsonify(calc.get_data('humidity'))
```

Devuelve la Velocidad del Viento

```
@app.route('/VelocidadViento')
def get_vel():
    return jsonify(calc.get_data('wind'))
```

Devuelve la Presión Barométrica

```
@app.route('/PresionBarometrica')
def get_press():
    return jsonify(calc.get_data('pressure'))
```

Devuelve la Calidad del Aire

```
@app.route('/CalidadAire')
def get_air():
    return jsonify(calc.get_air()))
```

Devuelve si es de noche o de día

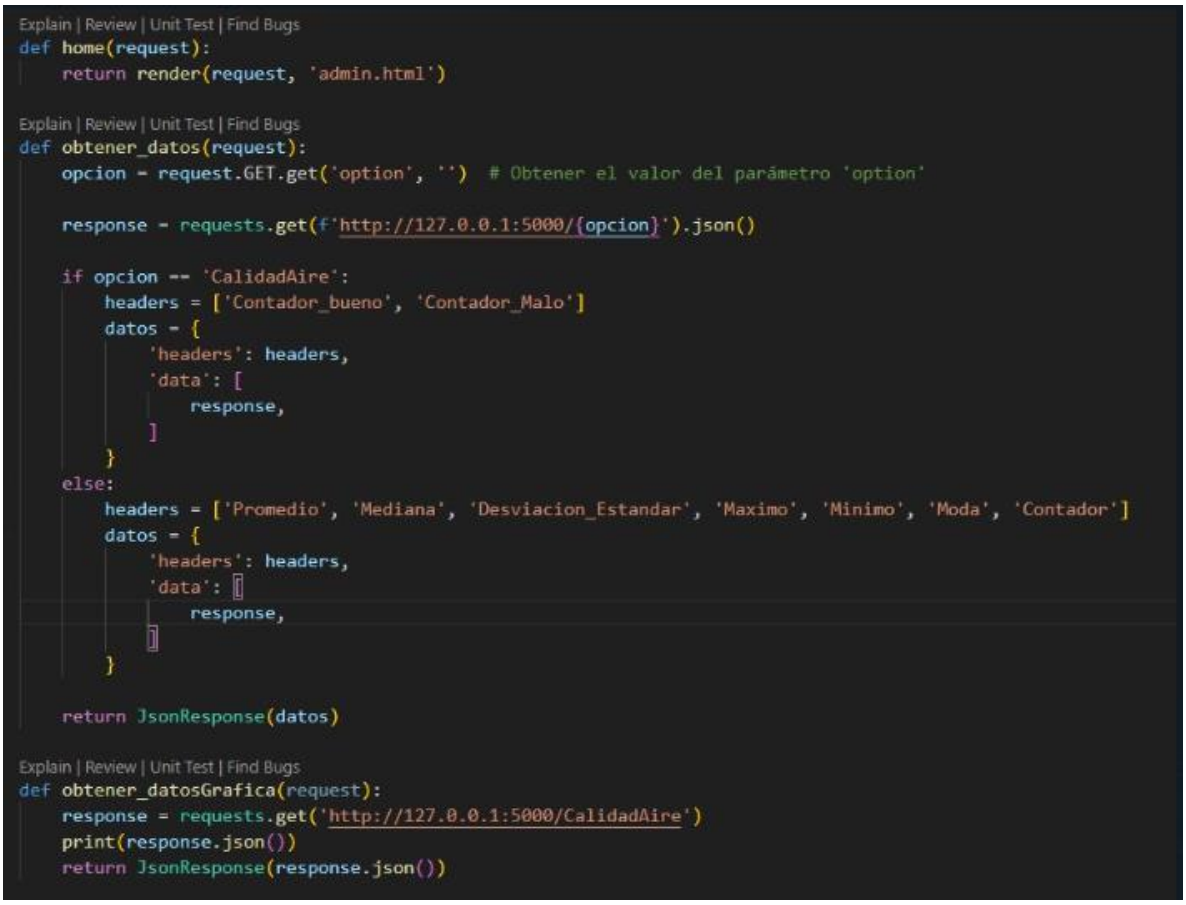
```
@app.route('/NightOrSun')
def daylight():
    response = {
        'nightOrSun': new_climate.daylight
    }
    return jsonify(response)
```

Ejecución de la Aplicación

```
if __name__ == '__main__':
    app.run()
```

WebApp

En el archivo views.py podemos encontrar los métodos esenciales, en el primero lo que se realiza es cargar la página de inicio, el segundo obtiene los datos dependiendo cuáles sean necesarios, al finalizar regresa al HTML.



```
Explain | Review | Unit Test | Find Bugs
def home(request):
    return render(request, 'admin.html')

Explain | Review | Unit Test | Find Bugs
def obtener_datos(request):
    opcion = request.GET.get('option', '')  # Obtener el valor del parámetro 'option'

    response = requests.get(f'http://127.0.0.1:5000/{opcion}').json()

    if opcion == 'CalidadAire':
        headers = ['Contador_bueno', 'Contador_Malo']
        datos = {
            'headers': headers,
            'data': [
                response,
            ]
        }
    else:
        headers = ['Promedio', 'Mediana', 'Desviacion_Estandar', 'Maximo', 'Minimo', 'Moda', 'Contador']
        datos = {
            'headers': headers,
            'data': [
                response,
            ]
        }

    return JsonResponse(datos)

Explain | Review | Unit Test | Find Bugs
def obtener_datosGrafica(request):
    response = requests.get('http://127.0.0.1:5000/CalidadAire')
    print(response.json())
    return JsonResponse(response.json())
```

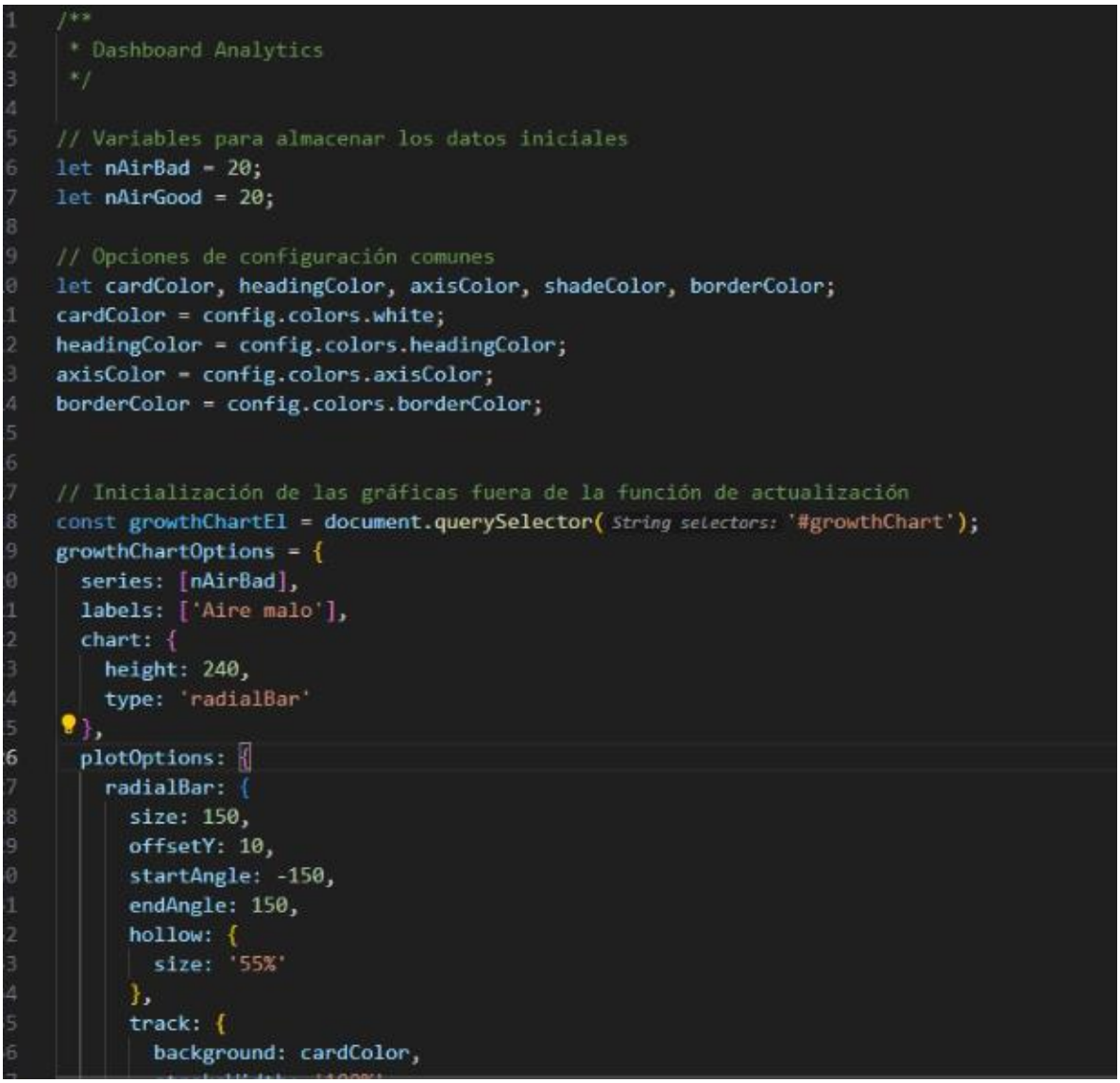
En este apartado del código se obtienen lo que es la calidad del aire, es decir los contadores buenos y malos.

```
Explain | Review | Unit Test | Find Bugs
def obtener_datosGrafica(request):
    response = requests.get('http://127.0.0.1:5000/CalidadAire')
    print(response.json())
    return JsonResponse(response.json())
```

En nightOrSun se obtiene lo que es un boolean que ayuda a validar si está soleado o no.

```
def nightOrSun(request):
    response = requests.get('http://127.0.0.1:5000/NightOrSun')
    print(response.json())
    return JsonResponse(response.json())
```

Dashboards-analytics

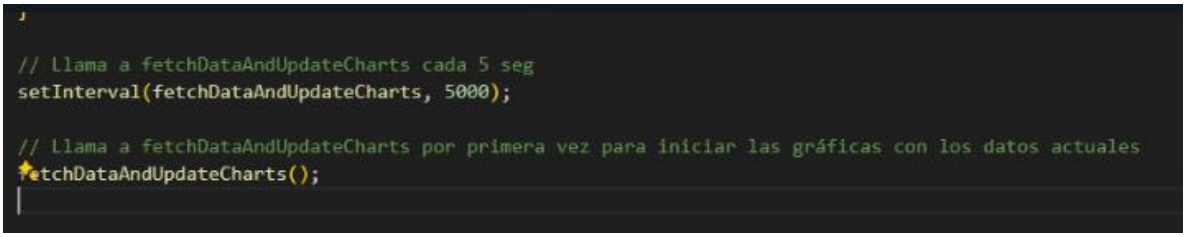


```
/**
 * Dashboard Analytics
 */

// Variables para almacenar los datos iniciales
let nAirBad = 20;
let nAirGood = 20;

// Opciones de configuración comunes
let cardColor, headingColor, axisColor, shadeColor, borderColor;
cardColor = config.colors.white;
headingColor = config.colors.headingColor;
axisColor = config.colors.axisColor;
borderColor = config.colors.borderColor;


// Inicialización de las gráficas fuera de la función de actualización
const growthChartEl = document.querySelector( String selectors: '#growthChart');
growthChartOptions = {
  series: [nAirBad],
  labels: ['Aire malo'],
  chart: {
    height: 240,
    type: 'radialBar'
},
  plotOptions: {
    radialBar: {
      size: 150,
      offsetY: 10,
      startAngle: -150,
      endAngle: 150,
      hollow: {
        size: '55%'
      },
      track: {
        background: cardColor,
```

Mediante Ajax se hacen las peticiones respectivas al servidor, el cual mostrará la cantidad de aire bueno.

```
Explain | Review | Unit Test | Find Bugs
function fetchDataAndUpdateCharts() {
  $.ajax({
    url: '/obtener_datosGrafica/',
    method: 'GET',
    Explain | Review | Unit Test | Find Bugs
    success: function(data) {
      console.log('Data fetched', data);

      // Actualiza los datos de las series
      nAirBad = data.contador_malo;
      nAirGood = data.contador_bueno;

      console.log('nWind', nAirBad);
      console.log('nAirGood', nAirGood);

      // Actualiza las opciones de las gráficas existentes
      growthChart.updateOptions({
        series: [nAirBad]
      });

      growthChart2.updateOptions({
        series: [nAirGood]
      });

      // Otros gráficos y actualizaciones si es necesario...

    },
    Explain | Review | Unit Test | Find Bugs
    error: function(error) {
      console.error('Error fetching data', error);
    }
  });
}
```

Mediante estos métodos se realiza una petición que actualiza constantemente todos los datos.



```
// Llama a fetchDataAndUpdateCharts cada 5 seg
setInterval(fetchDataAndUpdateCharts, 5000);

// Llama a fetchDataAndUpdateCharts por primera vez para iniciar las gráficas con los datos actuales
fetchDataAndUpdateCharts();
```

Así mismo desde una petición mediante Ajax, se determina lo principal para poder verificar si está soleado o no.

```
Explain | Review | Unit Test | Find Bugs
function updateNightOrSunImage() {
  $.ajax({
    url: '/nightOrSun/', // Api
    method: 'GET',
    Explain | Review | Unit Test | Find Bugs
    success: function (data) {
      console.log( data[0]: 'Data fetched', data[1]: data);

      let nightOrSun = data.nightOrSun; // Suponiendo que `data.nightOrSun` es verdadero o falso

      if (nightOrSun) {
        $('#nightSunImage').attr('src', "{% static 'assets/img/elements/sun.png' %}"); // URL de la imagen de
      } else {
        $('#nightSunImage').attr('src', "{% static 'assets/img/elements/night.png' %}"); // URL de la imagen d
      }
    },
    Explain | Review | Unit Test | Find Bugs
    error: function (error) {
      console.error('Error fetching data', error);
    }
  });
}
```

En el archivo admin.html mediante Ajax se puede determinar los datos de la tabla y así mismo con la petición que se mostró en views.p

```
<script>
  $(document).ready(function() {
      $('#select2Basic').change(function() {

          // Actualizar el título de la tabla según la selección
          $('#tableTitle').text(selectedOption);

          // Hacer una llamada AJAX para obtener los datos desde Django views.py
          $.ajax({
              url: '/obtener_datos/',  //  URL  en Django
              type: 'GET',
              data: {
                  'option': selectedOption
              },
              success: function(response) {
                  // Limpia el cuerpo y encabezados de la tabla
                  $('#dataHead').empty();
                  $('#dataBody').empty();

                  // Inserta los encabezados según la opción seleccionada
                  var headers = response.headers;
                  var headerRow = '<tr>';
                  $.each(headers, function(index, header) {
                      headerRow += '<th>' + header + '</th>';
                  });
                  headerRow += '</tr>';
                  $('#dataHead').append(headerRow);

                  // Inserta los datos obtenidos en el cuerpo de la tabla
                  $.each(response.data, function(index, item) {
                      var row = '<tr>';
                      $.each(headers, function(index, header) {
                          row += '<td>' + item[header.toLowerCase().replace(/\s+/g, '')] + '</td>';
                      });
```

Presupuesto

Artículo	Precio	Cantidad
Raspberry Pi4	Q1,000.00	1
Cargador Raspberry	Q100.00	1
Módulo Convertidor ADC	Q45.00	1
Alambre para Protoboard	Q37.50	15
Módulo de Fotointerruptor	Q29.00	1
Disco para Encoder	Q7.50	1
Sensor de Calidad del aire	Q39.00	1
Sensor de Presión	Q26.00	1
Maqueta	Q60.00	1
Total	Q1,344.00	-----