**CODIGOS FUENTE DEL DISPOSITIVO RECEPTOR**

Contenido

[1. Árbol del proyecto 2](#_Toc105614750)

[2. app.py 3](#_Toc105614751)

[3. grabarEntorno.py 9](#_Toc105614752)

[4. leerTxtVaca.py 12](#_Toc105614753)

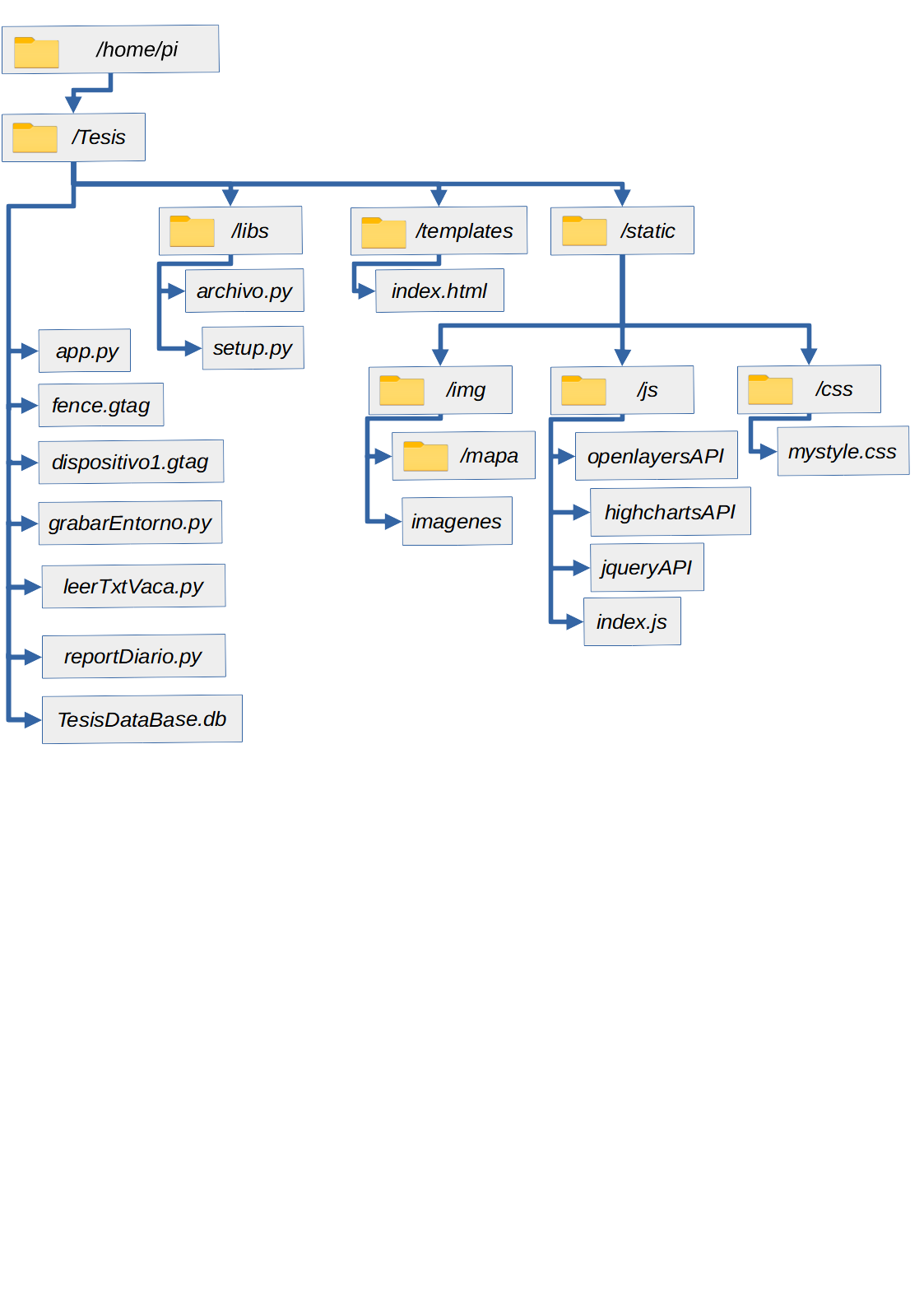
[5. reportDiario.py 14](#_Toc105614754)

[6. index.html 16](#_Toc105614755)

[7. index.js 22](#_Toc105614756)

1. Árbol del proyecto

A continuación, se presenta como están estructurados los directorios que contiene la carpeta ***Programas\_Dispositivo\_Receptor***.

  
Figura 1. Arbol del proyecto

Los scripts desarrollados en PYTHON, que funcionan en el backend se presentan a continuación:

1. app.py

|  |
| --- |
| **app.py** |
| '''      Escuela Politécnica Nacional      PROGRAMA PRINCIPAL DISPOSITIVO RECEPTOR      > Este programa se ejecuta al inicio/ reinicio de la Raspberry Pi, configurado en el archivo rc.local      El programa se encarga de:        >leer el sensor DHT22        >leer los datos recibidos por el RFM95W        >gestionar las peticiones realizadas por la interfaz de usuario        >ingresar los datos a los archivos txt y actualizarlos  '''  '''      1. Inclusion de librerias  '''  from flask import Flask, render\_template, request, Response  import multiprocessing as mp  import json  import Adafruit\_DHT  import adafruit\_rfm9x  import busio  from digitalio import DigitalInOut, Direction, Pull  import board  import numpy  import time  from libs.setup import Setup  from libs.archivo import Archivo  '''      2. Declaracion de pines y configuracion de variables  '''  # Configurar pines para el chip LoRa  CS = DigitalInOut(board.CE1)  RESET = DigitalInOut(board.D25)  spi = busio.SPI(board.SCK, MOSI=board.MOSI, MISO=board.MISO)  prev\_packet = None  sensorAmbiente = Adafruit\_DHT.AM2302  setup=Setup()  archivo = Archivo()  DHT\_PIN = 4  #Inicializar variables  dirPath="/home/pi/tesis"  #temp,bat,lon,lat,id,rssi  datos=mp.Array('d',[0,0,0,0,0,0,0]) #datos de los sensores  datosLon=mp.Array('d',[0,0]) #lon lat  datosLat=mp.Array('d',[0,0]) #lon lat  datosAmbiente=mp.Array('d',[0,0]) #temperatura,humedadrelativa  datosCerca = [0,0,0] #limites de la cerca virtual  app = Flask(\_\_name\_\_)  '''      3. Gestion de peticiones  '''  #Definir la ruta del html que se mostrara al ingresar la ip en el browser  @app.route('/')  def index():      return render\_template('index.html')  #Enviar los datos de hora y temp de la raspberr  @app.route('/Raspberry', methods = ['GET'])  def raspberry():      return json.dumps({'status': True, 'horaRaspberry': setup.get\_time(), 'tempRaspberry': setup.temperature()})  #Enviar los datos del DHT22  @app.route('/Ambiente', methods = ['GET'])  def ambiente():      return json.dumps({'status': True, 'tempAmb': datosAmbiente[0], 'hrAmb': datosAmbiente[1]})  #Enviar info del nodo final (GPS) desde el archivo txt  @app.route('/GPS', methods = ['GET'])  def ubicacion():      datosTXT = archivo.read\_file("dispositivo1.gtag")      return json.dumps({'status': True, 'id': datosTXT[4], 'longitud': datosTXT[2], 'latitud': datosTXT[3]})  #Enviar informacion del nodo final desde el archivo txt que se actualiza constantemente  @app.route('/INFO', methods = ['GET'])  def info():      datosTXT = archivo.read\_file("dispositivo1.gtag")      return json.dumps({'status': True, 'temperatura': datosTXT[0], 'bateria': datosTXT[1], 'id': datosTXT[4], 'rssi': datosTXT[5], 'snr': datosTXT[6],})  #Enviar los datos almacenados en la base de datos a la interfaz  @app.route('/LeerDataBaseEntorno', methods = ['GET'])  def getEntorno():      entorno = setup.leerDataBaseEntorno('/home/pi/tesis/tesisDataBase.db')      entorno = numpy.array(entorno)      dtgVector = entorno[:][0].ravel().tolist()      tempVector = entorno[:][1].ravel().tolist()      humVector = entorno[:][2].ravel().tolist()      return json.dumps({'status': True, 'dtg': dtgVector, 'temp': tempVector, 'hum': humVector})  #Enviar los datos almacenados en la base de datos a la interfaz  @app.route('/LeerDataBaseTempCorporal', methods = ['GET'])  def getTCorporal():      temperaturaCorporal = setup.leerDataBaseReportDiario('/home/pi/tesis/tesisDataBase.db')      temperaturaCorporal = numpy.array(temperaturaCorporal)      dtgVector = temperaturaCorporal[:][1].ravel().tolist()      minVector = temperaturaCorporal[:][2].ravel().tolist()      avgVector = temperaturaCorporal[:][3].ravel().tolist()      maxVector = temperaturaCorporal[:][4].ravel().tolist()      return json.dumps({'status': True, 'dtg': dtgVector, 'tmin': minVector, 'tavg': avgVector, 'tmax': maxVector})  #Enviar los limites de la cerca a la interfaz  @app.route('/LeerLim', methods = ['GET'])  def getfence():      limites = archivo.read\_file("fence.gtag")      return json.dumps({'status': True, 'latCenter': limites[0], 'lonCenter': limites[1], 'radius': limites[2]})  #Configurar los limites de la cerca, con los ingresados en el modal de la interfaz, estos datos se encuentran en el archivo txt  @app.route('/SetFence', methods = ['POST'])  def setfence():      datosCerca[0] = float(request.form['latitudeCenter']) #latitud del punto central de la cerca      datosCerca[1] = float(request.form['longitudeCenter']) #longitud del punto central de la cerca      datosCerca[2] = float(request.form['fenceRadius']) #Radio de la cerca en metros      archivo.write\_file("fence.gtag",datosCerca)      print("[INFO] Actualización del límite de la cerca")      return json.dumps({'status': True})  #Gestionar una solicitud de apagado o reinicio desde la interfaz de usuario  @app.route('/powerRaspberry', methods = ['POST'])  def powerRaspberry():      if(int(request.form['apagado']) == 1):          setup.poweroff()      elif(int(request.form['reinicio']) == 1):          setup.reboot()        print("[INFO] Apagado o reinicio del sistema")      return json.dumps({'status': True})  '''      4. Proceso principal  '''  def proceso(datos, datosLon, datosLat):      #incializar el RFM95W      rfm9x = adafruit\_rfm9x.RFM9x(spi, CS, RESET, 915.0)      rfm9x.set\_Sync\_Word(0x12)      rfm9x.tx\_power = 23        proceso = True      while proceso:            try:              #Incializar con un valor no relevante              data1=["1","100","100","-0.2675","-0.7245","1234"]              #print('RasPi LoRa')              # check for packet rx              data = rfm9x.receive()              rssi = rfm9x.rssi              snr = rfm9x.snr              if data is None:                  #Si no hay datos esperar un segundo                  print('- Waiting for PKT -')                  #data1=["1","100","100","-0.2675","-0.7245","1234"]                  #print("SPREADING FACTOR DEFAULT")                  #print(rfm9x.spreading\_factor)              else:                  try:                      # Obtener el mensaje desde el RFM95W                      prev\_packet = data                      packet\_text = str(prev\_packet, "utf-8")                      data1 = packet\_text.split(',')                      if (len(data1)>1):                          datos[0] = float(data1[0]) #temperatura                          datos[1] = float(data1[1]) #bateria                          datos[2] = float(data1[2])# + (datos[1]/(datos[1]+15))#long                          datos[3] = float(data1[3])# - (datos[1]/(datos[1]+10)) #lat                          datos[4] = int(data1[4]) #id                          datos[5] = int(rssi) #rssi                          datos[6] = int(snr) #snr                          print("temp: " + str(datos[0]) + " bat: " + str(datos[1]) + " lon: " + str(datos[2]) + " lat: " + str(datos[3]) + " id: " + str(datos[4]) + " rssi: " + str(datos[5]) + " snr: " + str(datos[6]))                            #Ingresar lo datos al archivo de texto dispositivoX.gtag                          #Fue realizado asi en caso de incrementar el numero de nodos finales                          if(datos[4]<3 and datos[2] != 0 and datos[3] != 0):                              datosLon[int(datos[4]-1)] = datos[2]                              datosLat[int(datos[4]-1)] = datos[3]                              archivoTexto = "dispositivo" + str(int(datos[4]-1)) + ".gtag"                              archivo.write\_file(archivoTexto,datos)                        else:                          print("algo pasa") #Fallos en la lectura de los datos, una longitud no esperada del mensaje                          #print(data)                  except Exception as errAct: #No hay un msg valido recibido desde el nodo final                      print(errAct)                      print('No decode')                time.sleep(1.0) #Esperar un segundo            except Exception as err:                  print("fallo en el prog principal")                  print(err)  #https://github.com/loraflow-net/loraflow/blob/master/loraflow.py  '''      5. Proceso de lectura de datos del DHT22, funciona en paralelo  '''  def leerAmbiente(datosAmbiente):      proceso = True      while proceso:          humAmbienteLeida, tempAmbienteLeida = Adafruit\_DHT.read\_retry(sensorAmbiente,4)          #Si existe un valor valido de temp y hum rel, almacenarlos en la variable, caso contrario almacenar 0          if humAmbienteLeida is not None and tempAmbienteLeida is not None:                      datosAmbiente[0] = tempAmbienteLeida + 3.0                      datosAmbiente[1] = humAmbienteLeida                      #print(time.localtime().tm\_min)                      #setup.insertDataBaseEntorno(datosAmbiente)                      #print("Temp={0:0.1f}\*CHum={1:0.1f}%".format(tempAmbienteLeida, humAmbienteLeida))          else:                      datosAmbiente[0] = 0                      datosAmbiente[1] = 0                      #print("Fallo")          time.sleep(1.0) # Esperar 1 segundo para una nueva lectura  '''      8. Inicio de los procesos  '''  if \_\_name\_\_ == '\_\_main\_\_':      procesar = mp.Process(target=proceso, args=(datos, datosLon, datosLat,)) #Asociar al proceso las variables      leer = mp.Process(target=leerAmbiente, args=(datosAmbiente,)) #Asociar al proceso las variables      procesar.start() # Iniciar el proceso principal      leer.start()   # Iniciar el proceso de lectura del DHT22      app.run(debug=True, host='0.0.0.0')  #Correr la aplicacion, con el puerto 5000 |

1. grabarEntorno.py

|  |
| --- |
| **grabarEntorno.py** |
| import sqlite3  import time  #import adafruit\_dht  import Adafruit\_DHT  import busio  from digitalio import DigitalInOut, Direction, Pull  import board  #sensorAmbiente = adafruit\_dht.DHT22(board.D4)  sensorAmbiente = Adafruit\_DHT.AM2302  '''      Este skecth  se ejecuta cada hora y se encargara de obtener los datos del sensor de temperatura      DTH22; despues hará lo siguiente:        1. Obtener la cantidad de datos que tiene la tabla >>entorno<< (no puede haber mas de 24 datos)      2. Si la cantidad de datos es superior o igual a 24 se borrara la primera de la tabla          La tabla tendra siempre los datos de las ultimas 24 horas, y tendra una estructura de stack FIFO          es decir el primero que entra es lo primero que sale.          Para poder realizar esto, se requiere de asignarle a cada fila un ID          2.1 Actualizar todos los ID, puesto que el id "1" se borraria junto con la fila      3. Asignarle el ID correspondiente al dato que va a ser insertado en la tabla      4. Insertar el dato en la tabla  '''  def insertDataBaseEntorno(datos,base):      try:          dtg=time.strftime('%Y-%m-%d %H:%M:%S',time.localtime())          db = sqlite3.connect(base)          cursor = db.cursor()          cursor.execute('''INSERT INTO entorno (id,dtg,temp,hum) VALUES (?,?,?,?)''',(datos[0],dtg,datos[1],datos[2]))          db.commit()          # Catch any exception      except Exception as e:          # Roll back any change if something goes horribly wrong          db.rollback()          raise e      finally:              # Close the db connection          db.close()  def contarDataBaseEntorno(base):      try:          db = sqlite3.connect(base)          cursor = db.cursor()          numeroDatos = cursor.execute('''SELECT COUNT(\*) FROM entorno''').fetchone()          db.commit()            # Catch any exception      except Exception as e:          # Roll back any change if something goes horribly wrong          db.rollback()          raise e      finally:              # Close the db connection          db.close()          return numeroDatos[0]  #Por editar  def borrarDato(base):      try:          db = sqlite3.connect(base)          cursor = db.cursor()          #Borrar el primer dato y actualizar el "id" de todos los datos en la base          cursor.execute('''DELETE FROM entorno WHERE id = (SELECT min(id) FROM entorno);''')          cursor.execute('''UPDATE entorno SET id = id-1;''')          db.commit()          # Catch any exception      except Exception as e:          # Roll back any change if something goes horribly wrong          db.rollback()          raise e      finally:              # Close the db connection          db.close()  def leerDataBaseEntorno(base):      try:          db = sqlite3.connect(base)          cursor = db.cursor()          #indice = cursor.execute('''SELECT indice FROM vaca01''').fetchall()          id = cursor.execute('''SELECT id FROM entorno''').fetchall()          dtg = cursor.execute('''SELECT dtg FROM entorno''').fetchall()          temp = cursor.execute('''SELECT temp FROM entorno''').fetchall()          hum = cursor.execute('''SELECT hum FROM entorno''').fetchall()          matrix = [id,dtg,temp,hum]      except Exception as e:          db.rollback()          raise e      finally:          db.close()          return matrix  humAmbienteLeida, tempAmbienteLeida = Adafruit\_DHT.read\_retry(sensorAmbiente,4)  #tempAmbienteLeida = 11.3#sensorAmbiente.temperature  #humAmbienteLeida = 9.2 #sensorAmbiente.humidity  '''  print(tempAmbienteLeida)  print(humAmbienteLeida)  #print(leerDataBaseEntorno('/home/pi/tesis/tesisDataBase.db'))  '''  #Obtener la cantidad de datos disponible  numeroDatos = contarDataBaseEntorno('/home/pi/tesis/tesisDataBase.db')  #Si la cantidad de datos es mayor a igual a los que deberia haber en la base de datos  #Borrar el primer dato y actualizar el "id" de todos los datos en la base  if(numeroDatos>=24):      borrarDato('/home/pi/tesis/tesisDataBase.db')  else:      numeroDatos = numeroDatos + 1  #Ingresar los nuevos datos, [id, temperatura, humedad]  datos = [numeroDatos,tempAmbienteLeida,humAmbienteLeida]  insertDataBaseEntorno(datos,'/home/pi/tesis/tesisDataBase.db')  #print(leerDataBaseEntorno('/home/pi/tesis/tesisDataBase.db')) |

1. leerTxtVaca.py

|  |
| --- |
| **leerTxtVaca.py** |
| import time  from math import \*  import sqlite3  from libs.archivo import Archivo  #from libs.archivo import Archivo  '''      Este skecth se ejecutará cada 30 minutos, para obtener el valor de temperatura corporal      que en ese momento se encuentre en el archivo txt dispositivo1.gtag      despues de obtener este valor lo ingresa en la tabla >>reportTemp<<      por el momento graba tambien en >>dispositivo<<  '''  archivo = Archivo()  datosVacas = [0,0,0,0,0,0,0]    def insertDataBaseVacas(datos,indice):      try:          dtg=time.strftime('%Y-%m-%d %H:%M:%S',time.localtime())          db = sqlite3.connect('/home/pi/tesis/tesisDataBase.db')          cursor = db.cursor()          #Leer longitud y latitud -> calcular distancia con el punto central          p1 = [-78.537574,-0.407238] #longitud y latitud del punto [0,0]          lon1 = radians(p1[0])          lat1 = radians(p1[1])          #longitud y latitud del punto dispositivo1.gtag          lon2 = radians(float(datos[2]))          lat2 = radians(float(datos[3]))          R = 6378 #Radio de la tierra Recuatorial 6378km // Rpolar 6357   //Requivol 6371          Dlat = lat2 - lat1          Dlon = lon2 - lon1          #HAVERSINE          a = (sin(Dlat/2)\*\*2)+(cos(lat1)\*cos(lat2)\*sin(Dlon/2)\*\*2)          c = 2\*asin(sqrt(a))          d= int(R\*c\*1000) #Distancia en metros          #cursor.execute('''DELETE FROM dispositivo''' + indice)          instruccion = '''INSERT INTO dispositivo''' + indice + ''' (dtg,id,temp,bat,lon,lat,rssi,snr,dist) VALUES (?,?,?,?,?,?,?,?,?)'''          #'''INSERT INTO dispositivo1 (dtg,id,temp,bat,lon,lat,rssi,snr) VALUES (?,?,?,?,?,?,?,?)'''          cursor.execute(instruccion,(dtg,datos[4],datos[0],datos[1],datos[2],datos[3],datos[5],datos[6],d))          #print(cursor.execute('''SELECT \* FROM dispositivo1''').fetchall())          db.commit()          # Catch any exception      except Exception as e:          # Roll back any change if something goes horribly wrong          db.rollback()          raise e      finally:              # Close the db connection          db.close()  def insertDataBaseTemp(datos):      try:          db = sqlite3.connect('/home/pi/tesis/tesisDataBase.db')          cursor = db.cursor()          #cursor.execute('''DELETE FROM dispositivo''' + indice)          #'''INSERT INTO dispositivo1 (dtg,id,temp,bat,lon,lat,rssi,snr) VALUES (?,?,?,?,?,?,?,?)'''          cursor.execute('''INSERT INTO reportTemp (temp) VALUES (?)''',(datos,))          #print(cursor.execute('''SELECT \* FROM reportTemp''').fetchall())          db.commit()          # Catch any exception      except Exception as e:          # Roll back any change if something goes horribly wrong          db.rollback()          raise e      finally:              # Close the db connection          db.close()  #datosVacas = archivo.read\_file("dispositivo0.gtag")  #insertDataBaseVacas(datosVacas,'0')  datosVacas = archivo.read\_file("dispositivo1.gtag")  insertDataBaseVacas(datosVacas,'1')  insertDataBaseTemp(datosVacas[0])  #print(type('''INSERT INTO dispositivo1 (dtg,id,temp,bat,lon,lat,rssi,snr) VALUES (?,?,?,?,?,?,?,?)''')) |

1. reportDiario.py

|  |
| --- |
| **reportDiario.py** |
| import numpy  import sqlite3  import time  '''      Este sketch se ejecutara cada 6 horas para la etapa de pruebas      en el final se ejecutara cada 24 horas, mediante crontab e-      Al ejecutarse, leera los datos acumulados en la tabla >>reportTemp<<      posteriormente sacará un PROMEDIO de todos estos datos y      almacenará los datos maximo, minimo y el promedio en la tabla >>reportDiario<<      finalmente eliminara todos los datos existentes en la tabla >>reportTemp<<  '''    try:      datos = [0,0,0]      db = sqlite3.connect('/home/pi/tesis/tesisDataBase.db')      cursor = db.cursor()      #obtener los datos de temperatura existentes en la base de datos.      temp = cursor.execute('''SELECT temp FROM reportTemp''').fetchall()      #Pasar los datos a un vector de manera que los pueda manipular.      tempArray = numpy.array(temp)      tempVector = tempArray.ravel().tolist()      tempVectorFloat = numpy.float\_(tempVector)      print(tempVectorFloat)      #Obtener el valor minimo "min"      datos[0] = numpy.min(tempVectorFloat)      #Obtener el valor máximo "max"      datos[2] = numpy.max(tempVectorFloat)      #Obtener el valor promedio "avg"      datos[1] = numpy.mean(tempVectorFloat)        #Obtener el numero de datos para actualizar el id      Nid = cursor.execute('''SELECT COUNT(\*) FROM reportDiario''').fetchone()      id = Nid[0] + 1      #print(id)      #Insertar en la base de datos que tiene el informe diario de las vacas      dtg=time.strftime('%Y-%m-%d',time.localtime())      cursor.execute('''INSERT INTO reportDiario (id,dtg,min,avg,max) VALUES (?,?,?,?,?)''',(id,dtg,datos[0],datos[1],datos[2]))        #eliminar los datos en el reportTemp      cursor.execute('''DELETE FROM reportTemp''')      #print (cursor.execute('''SELECT \* FROM reportDiario''').fetchall())      db.commit()  except Exception as e:      db.rollback()      raise e  finally:      db.close() |

Los scripts que funcionan en el front end se presentan a continuación:

1. index.html

|  |
| --- |
| **index.html** |
| <!DOCTYPE html>  <html lang="en">  <head>      <meta charset="UTF-8">      <meta http-equiv="X-UA-Compatible" content="IE=edge">      <meta name="viewport" content="width=device-width, initial-scale=1.0">      <link rel="stylesheet" href="../static/js/ol/css/ol.css">      <link rel="stylesheet" href="../static/css/mystyle.css">      <link rel="shortcut icon" href="{{ url\_for('static', filename='favicon.ico') }}">      <title>Proyecto de Titulacion</title>  </head>  <body scroll="no;">      <!--Presentacion de entrada-->      <div id="cargando" class="panel\_completo">          <div>            <br>            <div>              <img src="../static/img/Membrete.png" class="logo1" alt="" />            </div>          </div>          <br>          <p class="titulo">DISEÑO E IMPLEMENTACIÓN DE UN PROTOTIPO DE <br>MONITOREO DE GEOPOSICIÓN Y TEMPERATURA CORPORAL<br>PARA GANADO VACUNO MEDIANTE DISPOSITIVOS DE<br>COMUNICACIÓN LORA.</p>          <p class = "subtitulo">TRABAJO DE TITULACIÓN PREVIO A LA OBTENCIÓN DEL TÍTULO DE <br>INGENIERO EN ELECTRÓNICA Y CONTROL</p>          <p class = "subtitulo1">EDUARDO ISMAEL CHANGO ANALUISA</p>          <p class = "subtitulo2">DIRECTOR: Dr. GEOVANNY DANILO CHAVEZ GARCIA</p>            <p class="version">Cargando, por favor espere.</p>          <p class="hecho">Hecho en Ecuador</p>          <p class = "subtitulo3">Quito, marzo 2022 </p>        </div>          <!--Presentacion de salida-->        <div id="apagado\_reinicio" class="panel\_completo" style="display: none;">            <div>              <div>                <img src="../static/img/Membrete.png" class="logo1" alt="" />              </div>            </div>            <br>            <p class="titulo">Mensaje.</p>            <p class="version">Prototipo</p>            <br>            <div>              <div>                <img src="../static/img/salida.png" class="logo1" alt="" />              </div>            </div>            <br>            <p class="hecho">Hecho en Ecuador</p>        </div>        <!--Crear una etiqueta de audio, y enlazarla al archivo .wav de alarma-->      <audio id="myAudio">          <source src="../static/audio/Alarm05.wav" type="audio/wav">      </audio>      <!--Crear la division que desplegara la alarma correspondiente-->      <div class= "alarma" id="miAlarma" style="display:none;">          <div class = "column" style = "height: 100%; width: 10%; border-right-width: 2px; border-right-style: solid;">              <img src="../static/img/Alarm1.png" style="padding-top: 17%; width: 75%; height: 66%; z-index: 1000;">          </div>          <div class = "column" style = "width: 79%; padding-top: 3%;">              <div>Alerta: <span id="mensaje1Id"> La vaca se escapó </span> </div>          </div>          <div class = "column" style = "height: 100%; width: 10%; border-left-width: 2px; border-left-style: solid;"> </div>      </div>        <!--Crear la barra superior que contiene opciones-->      <nav class="miBarra">          <a class="active" id="reiniciar" style="width: 2%;"><img src="../static/img/Apagar.png" style="width: 45%; height: 45%; z-index: 1000;"></a>          <a id="limiteCerca">Conf Cerca</a>          <!--<a id="reportar">Reporte</a>-->          <a id="about">Acerca de</a>          <b id = "tempRaspberry" style="position: absolute; right: 155px;">40 °C</b>          <b id = "horaRaspberry" style="position: absolute; right: 5px;">2021-01-03 14:45:10 </b>      </nav>      <!-- Modales que se desplegan desde el menu superior  -->      <div class = "modales" id = "modalReiniciar">          <div class = "modales-contenido">              <p class="dataDisplay">Desea apagar el equipo?</p>              <button onmousedown="apagarFuncion()" style="background-color: #04AA6D;">SI</button>              <button id = "cancelarApagado" style="background-color: rgb(148, 8, 8);">NO</button>              <p class="dataDisplay">Desea reiniciar el equipo?</p>              <button onmousedown="reiniciarFuncion()" style="background-color: #04AA6D;">SI</button>              <button id = "cancelarReinicio" style="background-color: rgb(148, 8, 8);">NO</button>          </div>      </div>      <!--Modal de configuracion de la cerca-->      <div class = "modales" id = "modalConfCerca">          <div class = "modales-contenido">              <span class = "cerrar">&times;</span>              <p class = "titulo\_modal">Límites de la cerca virtual</p>              <div style = "padding-top: 10px;">                  <button onmousedown="conf\_LimCerca()" style="background-color: #04AA6D; width: 20%; border-radius: 5%;">Actualizar</button>              </div>                <div style = "padding-top: 10px;">                  <label >Latitud : </label>                  <input type = "text" value = "-0.4074212" class = "campo" id="latCenterId">              </div>                <div style = "padding-top: 10px;">                  <label >Longitud: </label>                  <input type = "text" value = "-78.5375126" class = "campo" id="lonCenterId">              </div>              <div style = "padding-top: 10px;">                  <label >Radio[m]: </label>                  <input type = "text" value = "200" class = "campo" id="radiusId">              </div>          </div>      </div>      <!--Modal de THI muestra las variaciones de las ultimas 24 hotas -->      <div class = "modales" id = "modalTHI">          <div class = "modales-contenido-THI">              <span class = "cerrar">&times;</span>              <p class = "titulo\_modal">Temperatura Ambiente y Humedad Relativa</p>              <div class="container-fluid--" id="temperaturaId"></div>          </div>      </div>        <!--Modal de Temperatura corporal -->      <div class = "modales" id = "modalTempCorporal">          <div class = "modales-contenido-THI">              <span class = "cerrar">&times;</span>              <p class = "titulo\_modal">Temperatura Ambiente y Humedad Relativa</p>              <div class="container-fluid--" id="desplazamientoId"></div>          </div>      </div>      <div class = "modales" id="modalManualUsuario">          <div class = "modales-contenido-THI">              <span class = "cerrar">&times;</span>              <iframe src="../static/documents/Manual Usuario.pdf" style="width:600px; height:450px;" frameborder="0"></iframe>          </div>      </div>      <!--Organizacion de los elementos en la pantalla principal-->      <div class="row" style="height: 100%;" scroll="yes;">          <!--Columna Izquierda contiene el carnet informativo del dispositivo prototipo-->          <informativos class="column left informativos">              <!--Imagen que contiene los logos de DACI, EPN, LARSI -->              <div>                  <img src="../static/img/Membrete.png" alt="Avatar" class="img\_EPN">              </div>              <!--Carnet informativo -->              <div class="row">                  <div class="carnets">                        <div class="column left" style="width: 30%;">                          <img src="../static/img/Carnet1.png" alt="Avatar" class="img\_carnet">                      </div>                      <div class="column datosCarnet" style="font-size: 10pt">                          <div>ID: <span id="VacaId1"> - </span> </div>                          <div>Temp Act: <span id="tempVacaId1"> 28 °C</span> </div>                          <div>RSSI: <span id="rssiVacaId1"> -100 </span> </div>                          <div>SNR <span id="snrVacaId1"> 0 </span> </div>                          <div>Bat: <span id="batVacaId1"> 100% </span> </div>                      </div>                  </div>              </div>              <br>              <div class="row" style="padding-left: 12%;">                  <button id = "opTempCorporal" class="infoTHI">                      Temperatura corporal                  </button>              </div>              <br>            </informativos>          <!--Columna central, contiene el mapa -->          <div class="column mapa" style="height: 124%; overflow: hidden;">              <div id="mapaId" class="map">                  <div class="ol-popup" id="popupId">                      <a href="#" id="popupCloserId" class="ol-popup-closer"></a>                      <div id="popupContentId"></div>                  </div>              </div>          </div>          <!--Columna derecha que contiene los datos de ambiente -->          <div class="column ambiente" style="height: 124%; overflow: hidden; text-align: center;">              <br>              <button class="infoTHI" id="opTHI">                  <div style="padding-bottom: 10%;">THI</div>                  <div class="column datosCarnet">                      <div id="thiAmbId" style="font-size: 24pt;"> 65.04</div>                  </div>              </button>              <br>              <br>              <div class="row infoAmbiente">                  <div style="padding-bottom: 10%;">Temp. Ambiente</div>                  <div class="column left" style="width: 20%; padding-left: 15%;">                      <img src="../static/img/termometro.png" style="width: 100%; height: 100%;">                  </div>                  <div class="column datosAmbiente">                      <div id="tempAmbId" style="color: #271e57;">18.3 °C</div>                  </div>              </div>              <br>              <div class="row infoAmbiente">                  <div style="padding-bottom: 10%;">Humedad Relativa</div>                  <div class="column left" style="width: 20%; padding-left: 15%;">                      <img src="../static/img/HR.png" style="width: 100%; height: 100%;">                  </div>                    <div class="column datosAmbiente">                      <div id="hrAmbId" style="color: #271e57;">86 %</div>                  </div>              </div>          </div>      </div>      <!--Librerias que se usaron para este proyecto -->      <script src="../static/js/ol/build/ol.js"></script>      <script src="../static/js/highcharts.js"></script>      <script src="../static/js/jquery-3.5.0.min.js"></script>      <!--Incluir el archivo JavaScript que contiene el front end -->      <script src="../static/js/index.js"></script>  </body>  </html> |

1. index.js

|  |
| --- |
| **index.js** |
| /\*      Indice del programa          0. Función principal (inicia en cuanto el documento esta listo)        1. Funciones de los modales        2. CODIGO PARA LOS MAPAS        3. Leer Base        4. Leer Sensores        5. Leer/ Escribir Raspberry        6. Funciones de limite de la cerca        7. Funciones de apagado y reinicio de la Raspberry        8. Código para los gráficos  \*/  //Declaracion de variable para el inicio y el apagado  var raspberry = {    reinicio: 0,    apagado: 0,  }  //Relacionar el elemento de audio declarado en HTML  var x = document.getElementById("myAudio");  //\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* 0. Función principal (inicia en cuanto el documento esta listo) \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/  $(document).ready(function(){      setTimeout(() => {      $("#cargando").fadeOut(350)      $("#reseteo\_valve").css({          'display': 'flex'      })    }, 4000)      leer\_LimCerca(); // Actualizar las dimensiones de la cerca en el mapa    //x.play();    // Función de actualización de los datos internos de la raspberry    setInterval(function(){      Leer\_Raspberry(); //Funcion para obtener los datos de la Raspberry, HORA, Temperatura Nucleo      Leer\_Ambiente(); //Funcion para obtener los datos leido del DHT22      Leer\_Sensores(); //Funcion para obtener datos de temperatura corporal, rssi, snr, nivel de bateria      Leer\_GPS(); //Funcion para obtener datos de GPS      updateCoordinate(item); // Actualizar los datos obtenidos en el mapa    },1000); //Actualización cada segundo  });  //\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* 1. FUNCION PARA LOS MODALES \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*//  //\* 1.1. Modal de reinicio \*/          //Traer el modal          var modalReinicio = document.getElementById("modalReiniciar");          //Definir el boton que despliega el modal          var opcionReinicio = document.getElementById("reiniciar");          //Definir el boton(es) que cierra el modal          var opcionCancelarReinicio = document.getElementById("cancelarReinicio");          var opcionCancelarApagado = document.getElementById("cancelarApagado");          //Abrir el modal          opcionReinicio.onclick = function(){            modalReinicio.style.display="block";          }          //Cerrar el modal          opcionCancelarReinicio.onclick = function(){            modalReinicio.style.display="none";          }          opcionCancelarApagado.onclick = function(){            modalReinicio.style.display="none";          }            function reiniciarFuncion(){            $('#apagado\_reinicio .titulo').html("El equipo se reiniciará en 5 segundos...");            $('#apagado\_reinicio').fadeIn(350);            raspberry.reinicio = 1;            powerRaspberry();          }          function apagarFuncion(){            $('#apagado\_reinicio .titulo').html("El equipo se apagará en 5 segundos...");            $('#apagado\_reinicio').fadeIn(350);            raspberry.apagado = 1;            powerRaspberry();          }    //\* 1.2. Modal de configuracion de la cerca virtual \*/          //Traer el modal          var modalLimCerca = document.getElementById("modalConfCerca");          //Definir el boton que despliega el modal          var opcionLimCerca = document.getElementById("limiteCerca");          //Definir el boton que cierra el modal          var opcionCerrarLimCerca = document.getElementsByClassName("cerrar")[0];          //Abrir el modal          opcionLimCerca.onclick = function(){              modalLimCerca.style.display="block";              leer\_LimCerca();          }          //Cerrar el modal          opcionCerrarLimCerca.onclick = function(){              modalLimCerca.style.display="none";          }  //\* 1.3. Modal de información sobre el THI \*/          //Traer el modal          var modalTHI= document.getElementById("modalTHI");          //Definir el boton que despliega el modal          var opcionTHI = document.getElementById("opTHI");          //Definir el boton que cierra el modal          var opcionCerrarTHI = document.getElementsByClassName("cerrar")[1];          //Abrir el modal          opcionTHI.onclick = function(){              modalTHI.style.display="block";              leer\_DataBase();          }          //Cerrar modal          opcionCerrarTHI.onclick = function(){              modalTHI.style.display="none";          }  //\* 1.4. Modal de información sobre la Temperatura Corporal \*/          //Traer el modal          var modalTempCorporal= document.getElementById("modalTempCorporal");          //Definir el boton que despliega el modal          var opcionTempCorporal = document.getElementById("opTempCorporal");          //Definir el boton que cierra el modal          var opcionCerrarTempCorporal = document.getElementsByClassName("cerrar")[2];          //Abrir el modal          opcionTempCorporal.onclick = function(){              modalTempCorporal.style.display="block";              leer\_DataBaseTemp();          }          //Cerrar modal          opcionCerrarTempCorporal.onclick = function(){              modalTempCorporal.style.display="none";          }  //\* 1.5. Modal de información sobre el sistema, Manual de Usuario \*/          //Traer el modal          var modalManualUsuario= document.getElementById("modalManualUsuario");          //Definir el boton que despliega el modal          var opcionManualUsuario = document.getElementById("about");          //Definir el boton que cierra el modal          var opcionCerrarManualUsuario = document.getElementsByClassName("cerrar")[3];          //Abrir el modal          opcionManualUsuario.onclick = function(){              modalManualUsuario.style.display="block";              //leer\_DataBaseTemp();          }          //Cerrar modal          opcionCerrarManualUsuario.onclick = function(){              modalManualUsuario.style.display="none";          }  //\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* 2. CODIGO PARA LOS MAPAS \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*//      //Latitud si el numero aumenta se va hacia el este, si disminuye hacia el oeste      //Longitud si el numero aumenta se va hacia el sur, si disminuye hacia el norte      //Tambillo      //var longitud= -78.537574;      //var latitud= -0.407238;      //Casa      //var longitud = -78.512724      //var latitud = -0.274338      //Oficina      //var longitud= -78.537574;      //var latitud= -0.407238;      //EPN      var longitud= -78.489526;      var latitud= -0.209524;      //Obtener elementos desde el archivo "index.html", para el cuadro de dialogo que muestra latitud y longitud      var container=document.getElementById("popupId");      var content=document.getElementById("popupContentId");      var closer=document.getElementById("popupCloserId");      //Se crea el mapa base, descargado desde OpenLayers, para trabajar OFFLINE, si se dispone de conexión a internet      //Para pruebas se puede usar el mapa de ARCGIS        //'https://server.arcgisonline.com/ArcGIS/rest/services/World\_Imagery/MapServer/tile/{z}/{y}/{x}'        //Para pruebas en windows se direcciona a los directorios que contiene el tile de esta manera        //'file:///C%3A/Users/eduar/Documents/Tesis/Mapas/CasaQuito/{z}/{x}/{y}.png'        //'file:///C%3A/Users/eduar/Documents/Tesis/Mapas/HaciendaTambillo/{z}/{x}/{y}.png'        //Para pruebas en Raspbian se direcciona a los directorios que contiene el tile de esta manera      //'../static/img/CasaQuito/{z}/{x}/{y}.png'      //'../static/img/HaciendaTambilloArc/{z}/{x}/{y}.png'      //'../static/img/EPN/{z}/{x}/{y}.png'  //\* 2.1. Creacion del mapa \*/      //Se crea el objeto "map", que se dibujara en la division 'mapaId'      var map= new ol.Map({        target: 'mapaId',//aqui se va a mostrar el mapa        renderer: 'canvas', // El tipo de renderizado        layers: [          new ol.layer.Tile({            source: new ol.source.XYZ({              url: '../static/img/mapa/{z}/{x}/{y}.png',            })          })        ],//La unica capa que tendrá sera la baseMapLayer creada arriba        view: new ol.View({          center: ol.proj.fromLonLat([longitud,latitud]),//Donde se va a centrar el mapa, por conveniencia dependera de la geoubicacion de la raspberry          zoom: 18,//El unico zoom para este proyecto por motivos de almacenamiento sera de 16, para trabajos online se puede tener un mayor rango de zoom          maxZoom: 20,          minZoom: 16        }),      });  //\* 2.2. Creacion del marcador \*/      //Agregar el mapa en el punto con multiples marcadores      //for(var i = 0; i < places.length; i++){        //Definir el marcador        var marker= new ol.Feature({          geometry: new ol.geom.Point(            //ol.proj.fromLonLat([places[i][0],places[i][1]])            ol.proj.fromLonLat([longitud,latitud])          ),          id: 'marcador1'        });        //Aqui se crea el marcador como un vector de construccion        //el cual tiene un vector fuente con las caracteristicas de este como posicion (latitud y longitud)        var markerVectorLayer = new ol.layer.Vector({          //Crear un vector fuente que contiene las caracteristicas definidas en la seccion de construccion del marcador          source: new ol.source.Vector({            features: [marker]          }),          //Configurar un icono que representara el movimiento del animalito sobre el mapa          //El icono se encuentra almacenado en la tarjeta          style: new ol.style.Style({            image: new ol.style.Icon({              anchor: [41, 62],              anchorXUnits: 'pixels',              anchorYUnits: 'pixels',              src: '../static/img/cow41x62.png',              scale: [0.5,0.5]            })          }),        })        //Agregar el marcador como una capa sobre el mapa        map.addLayer(markerVectorLayer);      //}      //Crear una caja de dialogo con cierta informacion sobre el marcador de la vaca dentro del mapa      var overlay = new ol.Overlay({        element: container, //reference to container var that was declared on html file        autoPan: true,        autoPanAnimation:{          duration: 250        }      });      //Agregar la caja de dialogo, la cual solo aparecer al dar click sobre el marcador      map.addOverlay(overlay);        closer.onclick = function(){        overlay.setPosition(undefined);        closer.blur();        return false;      };      //funcion para mostrar la caja de dialogo      map.on('singleclick',function(event){        if(map.hasFeatureAtPixel(event.pixel)===true){          var coordinate=event.coordinate;          content.innerHTML=('<b>Hola'+'<br/>Longitud: ' + item.Coordinate.Longitude1 + '<br/>Latitud:' +item.Coordinate.Latitude1);          overlay.setPosition(coordinate);        }else{          overlay.setPosition(undefined);          closer.blur;        }      });  //\* 2.3. Creacion de la cerca virtual \*/      //\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*Aqui se dibuja los limites del mapa de aquellos que no debe salir\*/      //Crear un circulo      var circle = new ol.geom.Circle(ol.proj.fromLonLat([longitud,latitud]),80);      //Ingresar las caracteristicas del circulo definidas en la seccion superior      var circleFeature = new ol.Feature(circle);        //Definir el estilo de la cerca, relleno y borde      var circleStyle = new ol.style.Style({        fill: new ol.style.Fill({            color: 'rgba(20, 10, 24, 0.2)'        }),        stroke: new ol.style.Stroke({            width: 3,            color: 'rgba(240, 100, 20, 1)'        }),      });      // Fuente y capa del vector      var circleVectorSource = new ol.source.Vector({        projection: 'EPSG:4326'      });      //Agregar todas las caracteristicas del vector      circleVectorSource.addFeature(circleFeature);      //Constrir el vector, con fuente y estilo      var circleVectorLayer = new ol.layer.Vector({          source: circleVectorSource,          style: circleStyle      });      //Agregar la cerca como una capa sobre el mapa      map.addLayer(circleVectorLayer);  //\* 2.4. Actualizacion del marcador con las coordenadas recibidas\*/      //Making the marker moves around the map      function updateCoordinate(item) {        var coord1 = ol.proj.fromLonLat([item.Coordinate.Longitude2, item.Coordinate.Latitude2]);        //var coord2 = ol.proj.fromLonLat([item1.Coordinate.Longitude2, item1.Coordinate.Latitude2]);        marker.getGeometry().setCoordinates(coord1);          //Disapear the dialog box each refresh time        overlay.setPosition(undefined);        closer.blur;              var distancia = ol.sphere.getDistance([longitud,latitud],[item.Coordinate.Longitude2,item.Coordinate.Latitude2]);        //Evaluar si la vaca salio o no de la cerca        console.log(distancia);        console.log(document.getElementById('radiusId'));        if(distancia > document.getElementById('radiusId').value){          console.log('La vaquita escapo');          document.getElementById("mensaje1Id").innerHTML="Se escapo";          //document.getElementById("alarmaId").style.backgroundColor="rgba(255,255,255,0)";          x.play();          document.getElementById("miAlarma").style.display = "block"        }else{          console.log('Vaquita dentro');          document.getElementById("mensaje1Id").innerHTML="Dentro";          //document.getElementById("alarmaId").style.backgroundColor="rgb(201,23,23)";          x.pause();          document.getElementById("miAlarma").style.display = "none"        }      }        var item = {};      item.id = marker.getId;      item.Coordinate = {};    //\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* 3. Leer Base \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*//  //\* 3.1. Leer la base de datos, que contiene temperatura y humedad relativa \*/      //Obtencion de datos a partir de la base de datos tesisDataBase      function leer\_DataBase(){        arrayData=[]; //vaciar el vector de datos de temperatura ambiente        arrayDataH=[]; //vaciar el vector de datos de humedad relativa        $.ajax({          url: '/LeerDataBaseEntorno', //Enviar la instruccion de lectura al programa principal          type: 'GET',          success: function(response){            const respuesta = JSON.parse(response);            //ingresar los datos a los vectores            respuesta.temp.forEach(element => arrayData.push(parseFloat(element)));            respuesta.hum.forEach(element => arrayDataH.push(parseFloat(element)));            respuesta.dtg.forEach(element => arrayDate.push(element));            //ingresar los vectores al grafico            chartT.series[0].setData(arrayData);            chartT.series[1].setData(arrayDataH);          },          error: function(error){            console.log(error);          }        });      }  //\* 3.2. Leer la base de datos, que contiene temperatura corporal\*/      function leer\_DataBaseTemp(){        //vaciar los vectores        arrayMinTemp = [];        arrayAvgTemp = [];        arrayMaxTemp = [];        $.ajax({          url: '/LeerDataBaseTempCorporal',          type: 'GET',          success: function(response){            const respuesta = JSON.parse(response);            //ingresar cada elemento al vector correspondiente            respuesta.tmin.forEach(element => arrayMinTemp.push(parseFloat(element)));            respuesta.tavg.forEach(element => arrayAvgTemp.push(parseFloat(element)));            respuesta.tmax.forEach(element => arrayMaxTemp.push(parseFloat(element)));            respuesta.dtg.forEach(element => arrayDateTemp.push(element));            //ingresar los vectores al grafico            chartD.series[0].setData(arrayMinTemp);            chartD.series[1].setData(arrayAvgTemp);            chartD.series[2].setData(arrayMaxTemp);          },          error: function(error){            console.log(error);          }        });      }  //\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* 4. Leer Sensores \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*//  //\* 4.1. Funcion para leer los sensores de temperatura, nivel de bateria y el snr y rssi\*/      function Leer\_Sensores(){        $.ajax({          url: '/INFO',          type: 'GET',          success: function(response){            const respuesta = JSON.parse(response);              //Actualizar los elementos al relacionarlos con los definidos en el HTML            document.getElementById('VacaId1').innerHTML = respuesta.id;            document.getElementById('tempVacaId1').innerHTML = parseFloat(respuesta.temperatura).toFixed(2)+" &deg;C";            document.getElementById('rssiVacaId1').innerHTML = respuesta.rssi;            document.getElementById('snrVacaId1').innerHTML = respuesta.snr;            document.getElementById('batVacaId1').innerHTML = respuesta.bateria;            },          error: function(error){            console.log(error);          }        });      }  //\* 4.3. Funcion para leer los datos de GPS\*/      function Leer\_GPS(){        $.ajax({          url: '/GPS',          type: 'GET',          success: function(response){            const respuesta = JSON.parse(response);            console.log(respuesta.longitud)            console.log(respuesta.latitud)            //Actualizar la latitud y la longitud            item.Coordinate.Longitude2=parseFloat(respuesta.longitud).toFixed(7);            item.Coordinate.Latitude2=parseFloat(respuesta.latitud).toFixed(7);          },          error: function(error){            console.log(error);          }        });      }    //\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* 5. Leer/Escribir Raspberry \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*//  //\* 5.1. Funcion para obtener la hora, y temperatura del nucleo de la raspberry\*/      function Leer\_Raspberry(){        $.ajax({          url: '/Raspberry',          type: 'GET',          success: function(response){            const respuesta = JSON.parse(response);            //console.log('Rpi')            document.getElementById('tempRaspberry').innerHTML = parseFloat(respuesta.tempRaspberry).toFixed(2)+" &deg;C";            document.getElementById('horaRaspberry').innerHTML = respuesta.horaRaspberry;          },          error: function(error){            console.log(error);          }        });      }  //\* 5.2. Funcion para leer los datos del DHT22\*/      function Leer\_Ambiente(){        $.ajax({          url: '/Ambiente',          type: 'GET',          success: function(response){            const respuesta = JSON.parse(response);            //console.log('Ambiente')            //Actualizar las variables de temperatura y humedad relativa al relacionarlos            //con los datos definidos en el archivo HTML            T1= parseFloat(respuesta.tempAmb).toFixed(2)            RH1= parseFloat(respuesta.hrAmb).toFixed(2)            document.getElementById('tempAmbId').innerHTML = T1 +" &deg;C";            document.getElementById('hrAmbId').innerHTML = RH1 +" %";            document.getElementById('thiAmbId').innerHTML = (0.8\*T1 + RH1\*0.01\*(T1-14.4) + 46.4).toFixed(2)          },          error: function(error){            console.log(error);          }        });      }  //\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* 6. Funciones de limite de la cerca \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*//  //\* 6.1. Funcion para leer el limite de la cerca desde el archivo "fence.gtag", a través del prg principal\*/      function leer\_LimCerca(){        $.ajax({          url: '/LeerLim',          type: 'GET',          success: function(response){            const respuesta = JSON.parse(response);            console.log(respuesta);            //Actualizar estos datos definidos en uno el modal configuracion cerca definido en el HTML            document.getElementById("latCenterId").value = respuesta.latCenter; //longitud + 1            document.getElementById("lonCenterId").value = respuesta.lonCenter; //longitud - 1            document.getElementById("radiusId").value = respuesta.radius; //latitud + 1              //Actualizar el dato leido, definir el punto centro y el radio            circle.setCenterAndRadius(ol.proj.fromLonLat([parseFloat(respuesta.lonCenter),parseFloat(respuesta.latCenter)]),parseFloat(respuesta.radius));            },          error: function(error){            console.log(error);          }        });      }  //\* 6.2. Funcion de configuracion de la cerca virtual\*/      function conf\_LimCerca (){        //Obtener datos ingresados en el modal de latitud, longitud y radio        var campos = {          latitudeCenter: document.getElementById("latCenterId").value,          longitudeCenter: document.getElementById("lonCenterId").value,          fenceRadius: document.getElementById("radiusId").value,        }        //Solicitar el ingreso de los datos de la cerca        $.post('/SetFence', campos, function(datos){          console.log(datos);        });        //Actualizar el limite de la cerca        leer\_LimCerca();      }  //\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* 7. Funciones de apagado y reinicio de la Raspberry \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*//      function powerRaspberry(){        $.post('/powerRaspberry', raspberry, function(datos){          console.log(datos);        });      }    //\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* 8. Código para los gráficos \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*//  //Obtener del html la div donde quiero dibujar el plot chart de temperatura  var divTemperatura=document.getElementById("temperaturaId");  //Declarar los vectores para temperatura, humedad relativa ambiente, y la hora a la que fueron tomadas  var arrayData=[];  var arrayDataH=[];  var arrayDate=[];  var chartT = new Highcharts.Chart({    chart:{ //Caracteristicas del chart      renderTo : divTemperatura.id, //en que container estoy graficando      backgroundColor: 'rgb(255,255,255)',//color de la grafica //de que color es el fondo del chart      width: 850, //ancho del chart      height: 350, //altura      borderWidth: 1, //ancho del borde      borderRadius: 20,//para hacer border redondeados      borderColor: '#111111', //color del borde      type: 'line' //tipo de    },    title: { //Caracteristicas del Title        text: 'Temperatura y Humedad Relativa Ambiente'    },    xAxis: {      gridLineWidth: 1, //revisar si salen las lineas      type: 'datetime',      categories: arrayDate,      labels:{        style: {          fontSize: '4pt',          }      }    },    yAxis: [{      gridLineWidth: 1,      title: { text: 'Temperatura [°C]',        style: {          fontWeight: 'bold',          color: Highcharts.getOptions().colors[1]}      },      labels: {        format: '{value} °C',        style: {          fontWeight: 'bold',          color: Highcharts.getOptions().colors[1]}      },      min: 0,      max: 35,    },{      gridLineWidth: 1,      title: { text: 'Hum Relativa [%]',        style: {          fontWeight: 'bold',          color: Highcharts.getOptions().colors[3]}      },      labels: {        format: '{value} %',        style: {          fontWeight: 'bold',          color: Highcharts.getOptions().colors[3]}      },      min: 40,      max: 100,      opposite: true,    }],    plotOptions: {      line: {        animation: true,        dataLabels: { enabled: false }      },    },    series: [{      name: 'Grados celsius',      data:[],      yAxis: 0,      color: Highcharts.getOptions().colors[1],    },{      name: 'Porcentaje',      data:[],      yAxis: 1,      color: Highcharts.getOptions().colors[3],    }],    credits: { enabled: true }  });    //\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*GRAFICO DE BARRAS PARA EL DESPLAZAMIENTO\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* \*/  //Relacionar el container declarado en HTML  var divDesplazamiento=document.getElementById("desplazamientoId");  //Declarar los vectores que componen el gráfico, inicialmente esta vacio  var arrayDateTemp = [];  var arrayMinTemp = [];  var arrayAvgTemp = [];  var arrayMaxTemp = [];  var chartD = new Highcharts.Chart({      chart:{ //Caracteristicas del chart        renderTo : divDesplazamiento.id, //en que container estoy graficando        backgroundColor: 'rgb(255,255,255)',//color de la grafica //de que color es el fondo del chart        width: 850, //ancho del chart        height:350, //altura        borderWidth: 1, //ancho del borde        borderRadius: 20,//para hacer border redondeados        borderColor: '#111111', //color del borde        type: 'column' //tipo de      },      title: { //Caracteristicas del Title          text: 'Temperatura Corporal Diaria'      },      series: [{//Series permite:        showInLegend: false,        data: []      }],      plotOptions: {          line: { animation: false,          dataLabels: { enabled: false }          },          series: { color: '#059e8a' }      },      xAxis: {        gridLineWidth: 1, //Habilitar cuadriculas        type: 'datetime', //El eje x es de tiempo        categories: arrayDateTemp,        labels:{          style: {            fontSize: '4pt',            }        }      },      yAxis: {        gridLineWidth: 3, //Habilitar cuadriculas        title: { text: 'Temperatura Corporal [°C]' },        min: 2,        max: 40      },      //Permite que al mover el cursor sobre las columnas aparece en un solo grafico      //maximo, min , y average      // Tomado de: https://jsfiddle.net/gh/get/library/pure/highcharts/highcharts/tree/master/samples/highcharts/demo/column-basic      tooltip: {        headerFormat: '<span style="font-size:10px">{point.key}</span><table>',        pointFormat: '<tr><td style="color:{series.color};padding:0">{series.name}: </td>' +            '<td style="padding:0"><b>{point.y:.1f} °C</b></td></tr>',        footerFormat: '</table>',        shared: true,        useHTML: true      },      plotOptions: {        series: {            label: {                connectorAllowed: false            },            pointStart: 1        }      },      series: [{        name: 'min',        data: []      }, {        name: 'avg',        data: []      }, {        name: 'max',        data: []      }],      credits: { enabled: true }  }); |