A-

Funcionamiento del MQTT una vez configurado todo:

```
manuel@ManuPi:~/Shared/MQTT $ mosquitto_sub -h localhost -t sitio1/temperatura
Sitio1 Temp. = 22 C
Sitio1 Temp. = 23 C
```

```
manuel@ManuPi:~/Shared/MQTT $ mosquitto_pub -h localhost -t sitio1/temperatura -m "Sitio1 Temp. = 22 C"
manuel@ManuPi:~/Shared/MQTT $ mosquitto_pub -h localhost -t sitio1/temperatura -m "Sitio1 Temp. = 23 C"
manuel@ManuPi:~/Shared/MQTT $
```

B-

Para hacer este punto, reutilizamos la mayor parte del código del TP3, lo que se hizo fue crear lo que sería la aplicación de Flask para el publicador, que es la misma que la de nuestro TP3, solo que ahora, se corre un publicador de MQTT en paralelo que cada 20 segundos publica el contenido de la base de datos datos_sensores.db. Y por el otro lado, el suscriptor usará también, la misma app de Flask del TP3, solo que él usa su propia base de datos (datos_sensores_sub.db) que es actualizada por un proceso corriendo en paralelo que añade a la tabla de lectura_sensores lo que recibe del publicador con MQTT. Con esto, ambos el publicador y el usuario pueden trabajar en sus bases de datos a través de la aplicación de Flask y sus rutas, añadiendo/eliminando/buscando información que deseen en exactamente la misma manera que en el TP3, solo que ahora el publicador comparte su data constantemente por MQTT y los subscriptores mantienen su base actualizada con lo publicado por MQTT.

Observamos los cambios en el código (además de obviamente haber cambiado las rutas a las diferentes bases de datos):

sensores.py:
Se incorporó la librería de SQLalchemy configurando la conexión a la base de datos junto el modelo de lectura de sensores. Además, se creó la ruta para poder devolver los datos en formato JSON para la integración de MQTT.

mqtt_pub_r1.py:
Se añadió el flask para utilizar SQLAlchemy y conectarlo a la base de datos. Creando una nueva función cuya función es utilizar el cliente MQTT para publicar los datos de los sensores almacenados en la base de datos,incluyendo la excepciones para errores en la publicación.

mqtt_sub_r1.py:
Configuración del pago.mqtt.client para la conexión y suscripción al tópico de sensores/datos. Se utilizó on connect() para manejar la conexión con el broker y suscribirse al tópico y la función on message para recibir los mensajes recibidos del tópico. Esto se lo metió en un bucle para mantener el cliente MQTT en continua ejecución, esperando y procesando mensajes entrantes.

sensores.py (aplicación del publicador):

```python
from flask import Flask, render_template, request, abort, jsonify, redirect
import os
import json
import logging
import time
import random
import sqlite3
import threading
import paho.mqtt.client as mqtt
from datetime import datetime
from funciones import geo_latlon

# Configuración del logging para depuración
logging.basicConfig(level=logging.DEBUG, format='%(asctime)s %(levelname)s: %(message)s')

app = Flask(__name__)

# Configuración de MQTT
MQTT_BROKER = "localhost"
MQTT_PORT = 1883
MQTT_TOPIC = "sensores/datos"

client = mqtt.Client()
client.connect(MQTT_BROKER, MQTT_PORT, 60)

@app.route('/')
def index():...

@app.route('/api/test')
def test():...

@app.route('/api/datos')
def datos():...

@app.route('/api/primer-registro')
def primerRegistro():...

@app.route('/api/anadir', methods=["POST"])
def añadir():...

@app.route('/api/eliminar', methods=['DELETE'])
```

```python
@app.route('/api/eliminar', methods=['DELETE'])
def eliminar():···

@app.route('/api/data', methods=['PUT'])
def update():···

def publish_data():
    while True:
        try:
            conn = sqlite3.connect('datos_sensores.db')
            cursor = conn.cursor()
            cursor.execute('SELECT * FROM lectura_sensores')
            records = cursor.fetchall()
            conn.close()
            data = []
            for record in records:
                data.append(
                {
                'id': record[0],
                'co2': record[1],
                'temp': record[2],
                'hum': record[3],
                'fecha': record[4],
                'lugar': record[5],
                'altura': record[6],
                'presion': record[7],
                'presion_nm': record[8],
                'temp_ext': record[9]
                })
            client.publish(MQTT_TOPIC, json.dumps(data))
            logging.debug(f"Datos publicados: {data}")
            time.sleep(20)
        except Exception as e:
            logging.error(f"Error al publicar datos: {e}")
            time.sleep(20)

if __name__ == '__main__':
    logging.info("Iniciando publicador de MQTT")

    #Corro publish_data concurrentemente para que no bloquee
    publish_thread = threading.Thread(target=publish_data)
    publish_thread.start()

    app.run(debug=True)
```

La función publish_data publica la información de la base del publicador por MQTT cada 20 segundos, y al correr la app, se corre publish_data en otro hilo para que no bloquee y la app Flask funcione en paralelo. Las demás rutas son las mismas del TP3.

sensores_sub.py (aplicación del suscriptor):

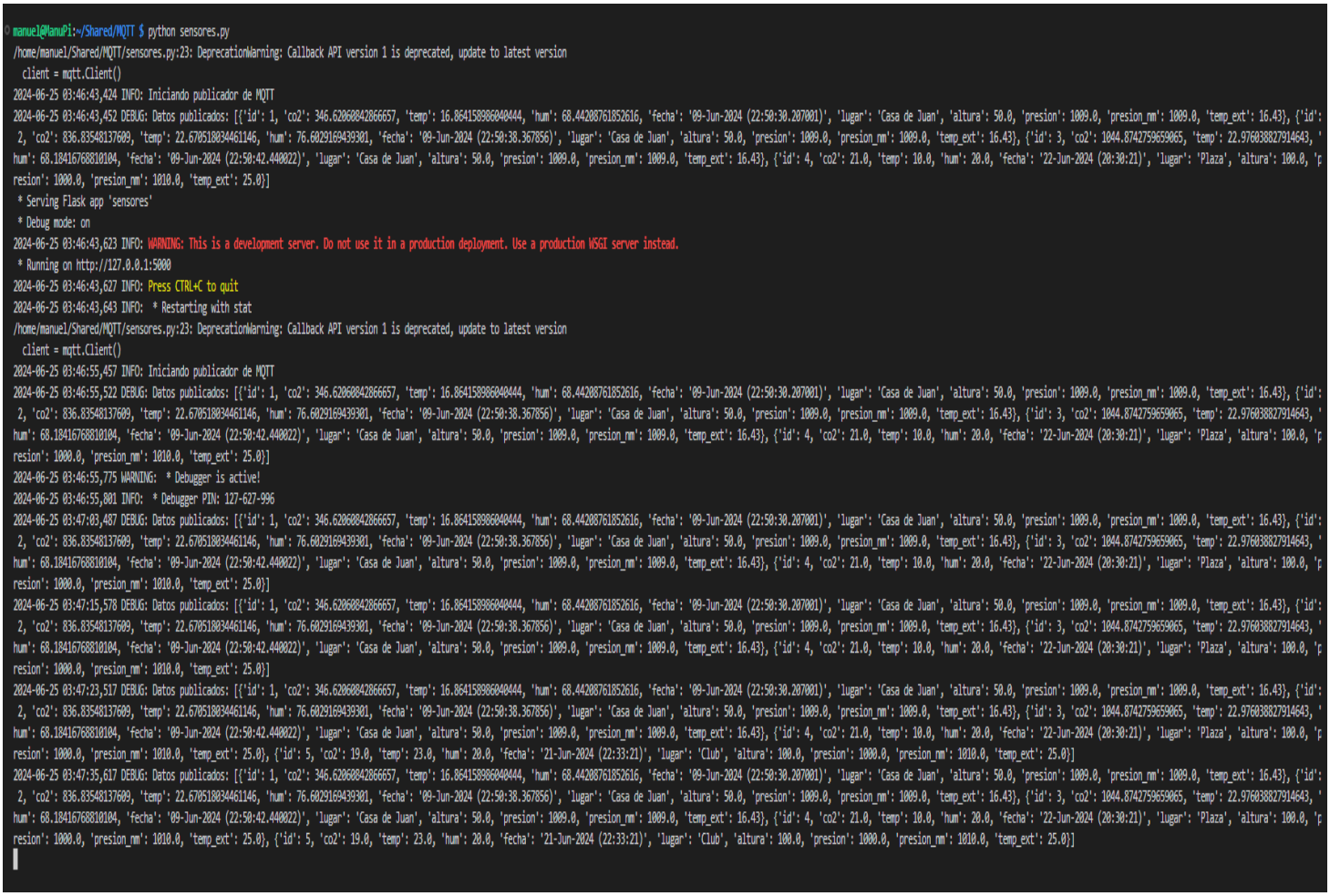
```python
from flask import Flask, render_template, request, abort, jsonify, redirect
import os
import json
import logging
import time
import random
import sqlite3
import threading
import paho.mqtt.client as mqtt
from datetime import datetime
from funciones import geo_latlon

# Configuración del logging para depuración
logging.basicConfig(level=logging.DEBUG, format='%(asctime)s %(levelname)s: %(message)s')

app = Flask(__name__)

# Configuración de MQTT
MQTT_BROKER = "localhost"
MQTT_PORT = 1883
MQTT_TOPIC = "sensores/datos"

@app.route('/')
def index():···

@app.route('/api/test')
def test():···

@app.route('/api/datos')
def datos():···

@app.route('/api/primer-registro')
def primerRegistro():···

@app.route('/api/anadir', methods=["POST"])
def añadir():···

@app.route('/api/eliminar', methods=['DELETE'])
def eliminar():···

@app.route('/api/data', methods=['PUT'])
def update():···

def on_connect(client, userdata, flags, rc):
    if rc == 0:
        logging.info("Conectado al broker MQTT")
```

```python
def on_connect(client, userdata, flags, rc):
    if rc == 0:
        logging.info("Conectado al broker MQTT")
        client.subscribe(MQTT_TOPIC)
    else:
        logging.error(f"Conexión fallida con código de resultado: {rc}")

def on_message(client, userdata, msg):
    try:
        data = json.loads(msg.payload.decode())
        for row in data:
            conn = sqlite3.connect('datos_sensores_sub.db')
            cursor = conn.cursor()
            cursor.execute("SELECT * FROM lectura_sensores WHERE co2 = ? AND temp = ? AND hum = ? AND fecha = ? AND lugar = ? AND altura = ? AND presion = ? AND pr
                           (row["co2"], row["temp"], row["hum"], row["fecha"], row["lugar"], row["altura"], row["presion"], row["presion_nm"], row["temp_ext"]))
            resp = cursor.fetchone()

            if resp is None:
                print("Insertamos datos")
                cursor.execute("INSERT INTO lectura_sensores (co2, temp, hum, fecha, lugar, altura, presion, presion_nm, temp_ext) VALUES (?, ?, ?, ?, ?, ?, ?, ?,
                               (row["co2"], row["temp"], row["hum"], row["fecha"], row["lugar"], row["altura"], row["presion"], row["presion_nm"], row["temp_ext"]))
                conn.commit()
                conn.close()

        logging.debug("Mensaje recibido exitosamente")
    except Exception as e:
        logging.error(f"Error al leer/añadir informacion publicada: {e}")

def sub_MQTT():
    try:
        client.connect(MQTT_BROKER, MQTT_PORT, 60)
        logging.info("Conectando al broker MQTT")
        client.loop_forever()
    except Exception as e:
        logging.error(f"Error al conectar al broker MQTT: {e}")

client = mqtt.Client()
client.on_connect = on_connect
client.on_message = on_message

if __name__ == '__main__':
    logging.info("Iniciando subscriptor de MQTT")
    sub_thread = threading.Thread(target=sub_MQTT)
    sub_thread.start()

    app.run(debug=True, port=5001, use_reloader=False)
```

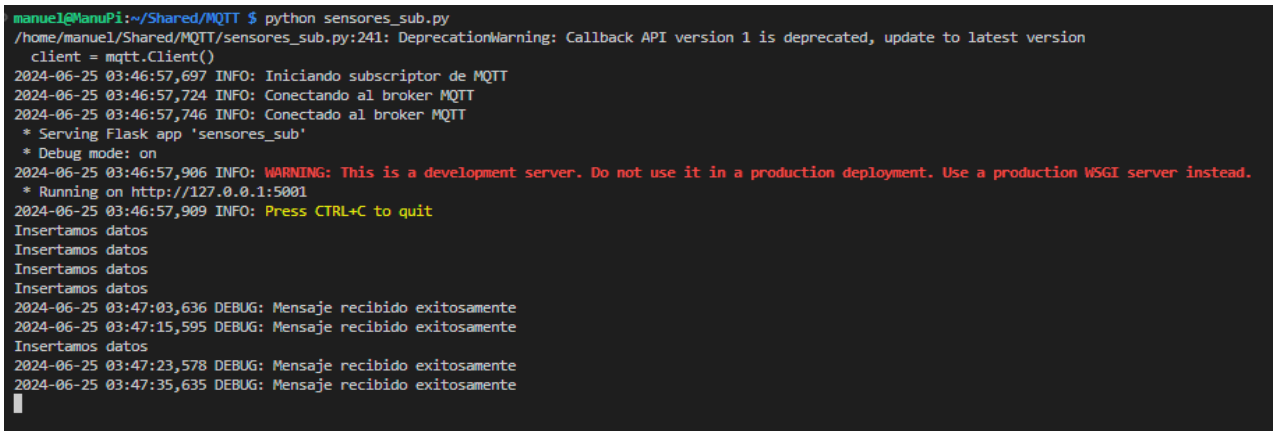
La función on_connect() es la misma del ejemplo dado, se subscribe al topico especificado al conectarse al broker MQTT, on_message() es llamada al recibir un mensaje MQTT, al correr, primero busca en la base de datos local del subscriptor si la data envíada por el publicador ya esta ahí, si no lo está, procede a insertarla en la tabla de lectura_sensores. Luego, sub_MQTT() concecta el subscriptor al broker y comienza un loop infinito para recibir mensajes. Finalmente, dentro del úiltimo if solo se especifica que la función que corre los procesos de MQTT (sub_MQTT) sea corrida en un hilo difertente, para no bloquear al resto de la app de Flask, y se especifica que la app se corra en el puerto 5001, ya que el publicador correrá en el 5000. El resto de esta app son las mismas rutas con el mismo funcionamiento que en sensores.py que a su vez, como ya ha sido aclarado, son las mismas rutas del TP3.

Ejemplo:

Corremos los programas para mostrar su funcionamiento, el único detalle es que no se podrá ver la interfaz gráfica del HTML como si se pudo en el TP3, ya que la computadora donde se esta corriendo este código es una computadora Linux sin salida de display, así que correremos todo desde terminales. De igual manera, la interfaz gráfica en el navegador funcionaría exactamente igual que en el TP3, así que no hay más para mostrar en ese ámbito.



Se corre la app del servidor, en 127.0.0.1:5000, se inicializa el publicador MQTT, y se publica la data ya en la tabla como se ve. A las 3:47:23, se insertó una nueva línea en la tabla, de id 5, y esta se comparte también, como se puede ver.



Se corre la app del subscriptor, en 127.0.0.1:5001, se conecta al bróker, se insertan las 4 filas que se reciben en la base local, luego, cuando el publicador envía una nueva versión con la nueva fila, esta se agrega también.

Se utiliza curl para pedir los datos de ambos el subscriptor y el publicador en urls distintas, y podemos observar que las respuestas son iguales, lo que significa que el publicador esta correctamente enviando su base de datos actualizada, y que el subscriptor esta recibiendo correctamente lo publicado tambien.

```
manuel@ManuPi:~/Shared/MQTT $ curl http://127.0.0.1:5000/api/datos
{
  "data": [
    {
      "altura": 50.0,
      "co2": 346.62060842866657,
      "fecha": "09-Jun-2024 (22:50:30.207001)",
      "hum": 68.44208761852616,
      "id": 1,
      "lugar": "Casa de Juan",
      "presion": 1009.0,
      "presion_nm": 1009.0,
      "temp": 16.864158986040444,
      "temp_ext": 16.43
    },
    {
      "altura": 50.0,
      "co2": 836.83548137609,
      "fecha": "09-Jun-2024 (22:50:38.367856)",
      "hum": 76.6029169439301,
      "id": 2,
      "lugar": "Casa de Juan",
      "presion": 1009.0,
      "presion_nm": 1009.0,
      "temp": 22.670518034461146,
      "temp_ext": 16.43
    },
    {
      "altura": 50.0,
      "co2": 1044.8742759659065,
      "fecha": "09-Jun-2024 (22:50:42.440022)",
      "hum": 68.18416768810104,
      "id": 3,
      "lugar": "Casa de Juan",
      "presion": 1009.0,
      "presion_nm": 1009.0,
      "temp": 22.976038827914643,
      "temp_ext": 16.43
    },
    {
      "altura": 100.0,
      "co2": 21.0,
      "fecha": "22-Jun-2024 (20:30:21)",
      "hum": 20.0,
      "id": 4,
      "lugar": "Plaza",
      "presion": 1000.0,
      "presion_nm": 1010.0,
      "temp": 10.0,
      "temp_ext": 25.0
    },
    {
      "altura": 100.0,
      "co2": 19.0,
      "fecha": "21-Jun-2024 (22:33:21)",
      "hum": 20.0,
      "id": 5,
      "lugar": "Club",
      "presion": 1000.0,
      "presion_nm": 1010.0,
      "temp": 23.0,
      "temp_ext": 25.0
    }
  ],
  "total": 5
}
```

```
manuel@ManuPi:~/Shared/MQTT $ curl http://127.0.0.1:5001/api/datos
{
  "data": [
    {
      "altura": 50.0,
      "co2": 346.62060842866657,
      "fecha": "09-Jun-2024 (22:50:30.207001)",
      "hum": 68.44208761852616,
      "id": 1,
      "lugar": "Casa de Juan",
      "presion": 1009.0,
      "presion_nm": 1009.0,
      "temp": 16.864158986040444,
      "temp_ext": 16.43
    },
    {
      "altura": 50.0,
      "co2": 836.83548137609,
      "fecha": "09-Jun-2024 (22:50:38.367856)",
      "hum": 76.6029169439301,
      "id": 2,
      "lugar": "Casa de Juan",
      "presion": 1009.0,
      "presion_nm": 1009.0,
      "temp": 22.670518034461146,
      "temp_ext": 16.43
    },
    {
      "altura": 50.0,
      "co2": 1044.8742759659065,
      "fecha": "09-Jun-2024 (22:50:42.440022)",
      "hum": 68.18416768810104,
      "id": 3,
      "lugar": "Casa de Juan",
      "presion": 1009.0,
      "presion_nm": 1009.0,
      "temp": 22.976038827914643,
      "temp_ext": 16.43
    },
    {
      "altura": 100.0,
      "co2": 21.0,
      "fecha": "22-Jun-2024 (20:30:21)",
      "hum": 20.0,
      "id": 4,
      "lugar": "Plaza",
      "presion": 1000.0,
      "presion_nm": 1010.0,
      "temp": 10.0,
      "temp_ext": 25.0
    },
    {
      "altura": 100.0,
      "co2": 19.0,
      "fecha": "21-Jun-2024 (22:33:21)",
      "hum": 20.0,
      "id": 5,
      "lugar": "Club",
      "presion": 1000.0,
      "presion_nm": 1010.0,
      "temp": 23.0,
      "temp_ext": 25.0
    }
  ],
  "total": 5
}
```