Para realizar este trabajo, instalamos SQLite y Heidi SQL para poder manipular y observar el repositorio de datos. Por otro lado, se verifica que tanto Python como pip estén instalados, y se los instala en caso de que no lo estén.

Se utiliza el siguiente código para crear una tabla:

```python
import time
import random
import sqlite3
from flask import Flask, render_template, jsonify
from datetime import datetime
from funciones import geo_latlon

app = Flask(__name__)

def create_table():
    conn = sqlite3.connect('datos_sensores.db')
    cursor = conn.cursor()
    cursor.execute('''CREATE TABLE IF NOT EXISTS lectura_sensores (
                        id INTEGER PRIMARY KEY,
                        co2 REAL,
                        temp REAL,
                        hum REAL,
                        fecha TEXT,
                        lugar TEXT,
                        altura REAL,
                        presion REAL,
                        presion_nm REAL,
                        temp_ext REAL
                    )''')
    conn.commit()
    conn.close()
```

Breve explicación de la función:

- Se conecta a la base de datos “datos_sensores.db”.
- Se indica que la tabla “lectura_sensores” debe crearse en caso que no exista.
- Se indica que la tabla tendrá 10 columnas, entre las que se encuentran **id**, **temp**, **fecha** y **lugar**.
- Se realiza un commit y se cierra la conexión.

Se ingresa un nuevo dato ejecutando la siguiente ruta:

```python
@app.route('/api/insertar-dato')
def insertar_dato_prueba():
    try:
        datos_prueba = LecturaSensores(
            co2=400.00,
            temp=22.50,
            hum=45.00,
            fecha="2023-06-08",
            lugar="Oficina",
            altura=100.00,
            presion=1013.25,
            presion_nm=1010.00,
            temp_ext=18.00
        )
        db.session.add(datos_prueba)
        db.session.commit()
        return jsonify({'mensaje': 'Datos insertados correctamente'})
    except Exception as e:
        db.session.rollback()
        return jsonify({'error': 'Error al insertar datos', 'detalle': str(e)}), 500
```

Breve explicación:

- Se realiza un try, para que se intente insertar los datos de prueba que se encuentran en LecturaSensores.
- Se agrega a la tabla los datos y se realiza un commit.
- Una vez realizado el commit, se muestra el mensaje “Datos insertados correctamente”.
- En caso de haber un error, se muestra el mensaje “Error al insertar datos”.

Similarmente, en la siguiente captura se muestran diferentes rutas que realizan diferentes acciones:

```python
@app.route('/')
def index():
    return render_template('tabla_sensores_para_editar.html')

@app.route('/api/todos-los-datos')
def mostrar_todos_los_datos():
    try:
        records = LecturaSensores.query.all()
        if not records:
            return jsonify({'mensaje': 'No se encontraron datos'}), 404
        return jsonify([record.to_dict() for record in records])
    except Exception as e:
        return jsonify({'error': 'Error al buscar datos', 'detalle': str(e)}), 500

@app.route('/api/primer-registro')
def mostrar_primer_registro():
    try:
        primer_registro = LecturaSensores.query.first()
        if not primer_registro:
            return jsonify({'mensaje': 'No se encontraron datos'}), 404
        return jsonify(primer_registro.to_dict())
    except Exception as e:
        return jsonify({'error': 'Error al buscar datos', 'detalle': str(e)}), 500

@app.route('/api/directorio-db')
def directorio_db():
    return jsonify({'directorio': os.getcwd(), 'path': db_path})
```

- **@app.route(‘/‘):** muestra la tabla en un archivo .html:

Type a keyword...

id	CO2	CO2 Norm	Temp	Hum	Fecha	Lugar	Altura	Presion	Presion nm	Temp ext	Temp ref
1	561.96		21.78	51.70	10-Jun-2024 (20:36:15.963711)	Buenos Aires	2.00	1014.00	1014.00	15.65	
2	856.89		19.80	60.08	10-Jun-2024 (20:36:16.973166)	Buenos Aires	2.00	1014.00	1014.00	15.65	
3	816.60		18.99	73.53	10-Jun-2024 (20:36:17.990382)	Buenos Aires	2.00	1014.00	1014.00	15.65	
4	1066.85		18.53	75.47	10-Jun-2024 (20:36:19.006085)	Buenos Aires	2.00	1014.00	1014.00	15.65	
5	819.84		19.56	51.32	10-Jun-2024 (20:36:20.017996)	Buenos Aires	2.00	1014.00	1014.00	15.65	
6	400.00		22.50	45.00	2023-06-08	Oficina	100.00	1013.25	1010.00	18.00	

Showing 1 to 6 of 6 results

Previous 1 Next

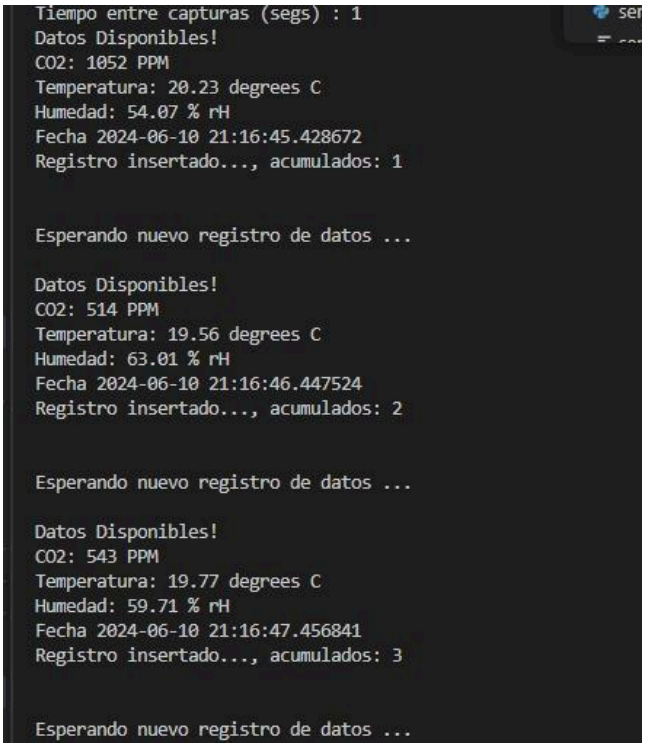
- **@app.route(‘/api/todos-los-datos’):** muestra todos los datos almacenados, mostrando mensajes en el caso de que no haya datos, o que haya ocurrido un error.
- **@app.route(‘/api/primer-registro’):** muestra el primer registro almacenado, mostrando mensajes en el caso de que no haya datos, o que haya ocurrido un error.
- **@app.route(‘/api/directorio-db’):** muestra el directorio de la base de datos.

Dar formato al texto


```
{
  "directorio": "/home/nacho/Python",
  "path": "/home/nacho/Python/datos_sensores.db"
}
```

Cardinale, del Valle, D’imperio, Ratcliffe

A continuación se muestran capturas de pantalla de lo que se muestra en la terminal al ejecutar el archivo “sensores_r2.py”, que se utiliza para insertar registros en la base de datos a mano:



Finalmente, se adjuntan capturas de rutas adicionales que hemos realizado para agregar funcionalidades:

```python
@app.route('/filtro')
def filtro():
    conn = sqlite3.connect('pepe.db')
    cursor = conn.cursor()
    cursor.execute('SELECT DISTINCT lugar FROM lectura_sensores')
    records = cursor.fetchall()
    conn.close()
    return render_template('listado.html', records=records)

@app.route('/resultado', methods=['POST'])
def resultado():
    conn = sqlite3.connect('pepe.db')
    cursor = conn.cursor()
    variable = request.form['lugar']
    query = "SELECT * FROM lectura_sensores WHERE lugar = '" + variable + "'"
    cursor.execute(query)
    records = cursor.fetchall()
    conn.close()
    return render_template('basic_table.html', records=records)
```

```python
@app.route('/eliminacion/<int:idReg>', methods=['DELETE'])
def eliminacion(idReg):
    conn = sqlite3.connect('pepe.db')
    cursor = conn.cursor()
    query = 'SELECT * FROM lectura_sensores WHERE id=' + str(idReg)
    cursor.execute(query)
    registro = cursor.fetchone()
    if registro is None:
        conn.close()
        return jsonify({"error": "Reading not found"}), 404

    query = 'DELETE FROM lectura_sensores WHERE id=' + str(idReg)
    cursor.execute(query)
    conn.commit()
    conn.close()
    return jsonify({"message": "Reading deleted successfully"}), 200
```

```javascript
function eliminarRegistro() {
    const idInput = document.getElementById("idReg");
    const id = idInput.value;
    fetch(`/eliminacion/${id}`, {
        method: 'DELETE',
    })
    .then(response => response.json())
    .then(data => {
        if (data.error) {
            console.log("Error1: " + data.error);
        } else {
            console.log("Success: " + data.message);
        }
    })
    .catch(error => {
        console.error("Error2:", error);
    });
    idInput.value = '';
}
```

- **@app.route(‘/filtro’):** muestra un los valores existentes de la columna lugar en la tabla.
- **@app.route(‘/resultado’):** muestra los registros que coincidan con el lugar ingresado.
- **@app.route(‘/eliminacion’):** elimina un registro por id. Utiliza la función eliminarRegistro, también adjuntada.