Universidad Católica Argentina
Facultad de Ingeniería y ciencias agrarias
Protocolos DE INTERNET

Trabajo Práctico N° 3

Cursada: 1er Cuatrimestre 2024

Grupo: Monti Facundo, Villanueva Mateo, Lamela Pablo

Profesor/a: Ingeniero Javier A. Ouret.

Integrantes:

N°	Nombre	Mail	Legajo
1	Facundo Monti	facundommontiuca@uca.edu.ar	44-759-207
2	Mateo Villanueva	mateovillanueva@uca.edu.ar	15-225314-9
3	Pablo Lamela	lamelapablo@uca.edu.ar	45-748-517

Corrección:

Entrega 1	**Devolución 1**	**Entrega 2**	**Nota**

TP 3: RestAPI

Índice:

Objetivos:

- Observar el funcionamiento de servidores sin control de estado
- Seguir los indicativos posteados en el GitHub para descargar y configurar el gestor de SNMP
- Gestionar una base de datos

Consigna:

Instalar SQLite.

Instalar Heidi SQL en Windows.

Para Linux y Windows se puede usar DBeaver Community Edition.

También se puede instalar DB Browser for SQLite.

Desde una terminal crear la base de datos, si no existe SQLite la crea automáticamente.

La idea es crear un repositorio de datos para recibir datos de sensores.

Conectarse a la base de datos desde el gestor elegido.

Crear una tabla para recibir los datos de los sensores.

Verificar con el gestor o desde el SQLite desde una terminal que esté todo bien.

Verificar que Python y pip esté instalado.

Acceder a la base de datos de Python para ingresar nuevos valores.

Desde el programa en Python importo las dependencias.

Adaptar el código en github de acuerdo con el criterio de diseño elegido.

Notas:

- Hubo varias consultas referidas a que no se desplegaban correctamente los datos.
- La idea de la versión r1 era que para el TP se adapte el código resolviendo las inconsistencias.
- Ahora podrán encontrar las versiones r2 con algunas modificaciones

    - sensores_r1 genera una db llamada datos_sendores.db

- sensor_editar_tabla_r2 agrega nuevas rutas y mensajes.

@app.route('/api/prueba')

@app.route('/')

@app.route('/api/todos-los-datos')

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

@app.route('/api/directorio-db')

@app.route('/api/insertar-dato')

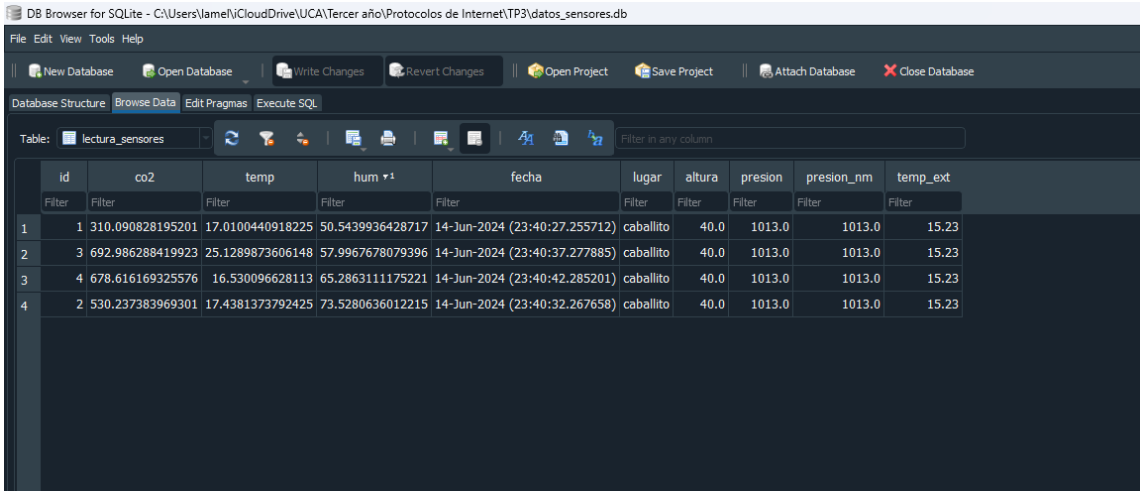
Verificar y mejorar.

Procedimiento:

Primero se instaló DB Browser for SQLite con tal de poder manipular y visualizar los resultados y tablas. También instalamos pip, Python y otros verificando la consigna asignada.

Consiguiente a ello se crea desde una terminal la base de datos, autogenerada por SQLite.

Posteriormente, nos conectamos a la base de datos desde el gestor seleccionado y se verifica la carga correcta de los datos en la tabla “datos_sensores.db”.



	id	co2	temp	hum	fecha	lugar	altura	presion	presion_nm	temp_ext
1	1	310.090828195201	17.0100440918225	50.5439936428717	14-Jun-2024 (23:40:27.255712)	caballito	40.0	1013.0	1013.0	15.23
2	3	692.986288419923	25.1289873606148	57.9967678079396	14-Jun-2024 (23:40:37.277885)	caballito	40.0	1013.0	1013.0	15.23
3	4	678.616169325576	16.530096628113	65.2863111175221	14-Jun-2024 (23:40:42.285201)	caballito	40.0	1013.0	1013.0	15.23
4	2	530.237383969301	17.4381373792425	73.5280636012215	14-Jun-2024 (23:40:32.267658)	caballito	40.0	1013.0	1013.0	15.23

Imagen 1: Tabla “datos sensores.db”

A continuación, se define "***@app.route('/')***", éste nos renderiza un archivo html que contiene la tabla que se rellena luego de hacer una request al endpoint "/api/todos-los-datos".

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

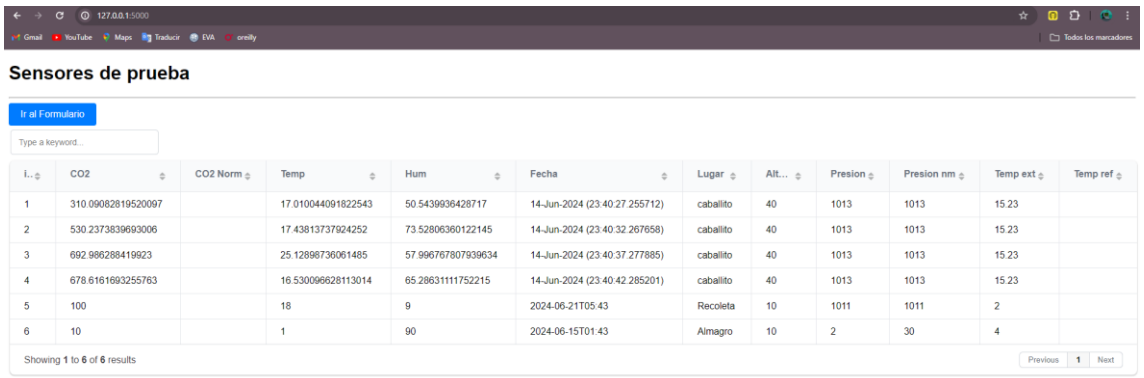


Imagen 2: Tabla en un html.

Ahora definimos "***@app.route('/api/todos-los-datos')***" que nos enseña todos los datos de la tabla o un mensaje de error o advertencia por falta de datos.

```python
@app.route('/api/todos-los-datos')
def datos():
    conn = sqlite3.connect('datos_sensores.db')
    cursor = conn.cursor()
    cursor.execute('SELECT * FROM lectura_sensores')
    records = cursor.fetchall()
    conn.close()
    return jsonify([{
        '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]
    } for record in records])
```

Imagen 3: Recorte de los datos mostrados.

Casi por último tenemos "***@app.route('/api/primer-registro')***" quien nos muestra el primer renglón de la tabla o un mensaje de error o advertencia por falta de datos.

```python
@app.route('/api/primer-registro')
def primer_registro():
    conn = sqlite3.connect('datos_sensores.db')
    cursor = conn.cursor()
    cursor.execute('SELECT * FROM lectura_sensores LIMIT 1')
    record = cursor.fetchone()
    conn.close()

    if record is None:
        return jsonify({'error': 'No records found'}), 404

    return jsonify([{
        '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]
    }])
```

Imagen 4: recorte del primer renglón de la base de datos.

También "***@app.route('/api/directorio-db')***" nos provee el directorio de la base de datos.

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

Imagen 5: Directorio provisto.

Finalmente "***@app.route('/api/insertar-dato)***" nos permite ingresar un registro a la tabla tomando los datos enviados a través de un form. Luego, se nos redirecciona a la tabla en la que se muestran todos los datos.

```python
@app.route('/api/insertar-dato', methods=["POST"])
def insertar_dato():
    data = request.form
    co2 = data.get('co2')
    temp = data.get('temp')
    hum = data.get('hum')
    fecha = data.get('fecha')
    lugar = data.get('lugar')
    altura = data.get('altura')
    presion = data.get('presion')
    presion_nm = data.get('presion_nm')
    temp_ext = data.get('temp_ext')

    conn = sqlite3.connect('datos_sensores.db')
    cursor = conn.cursor()
    cursor.execute('''
        INSERT INTO lectura_sensores (co2, temp, hum, fecha, lugar, altura, presion, presion_nm, temp_ext)
        VALUES (?, ?, ?, ?, ?, ?, ?, ?, ?)
    ''', (co2, temp, hum, fecha, lugar, altura, presion, presion_nm, temp_ext))
    conn.commit()
    conn.close()

    return redirect(url_for('index'))
```

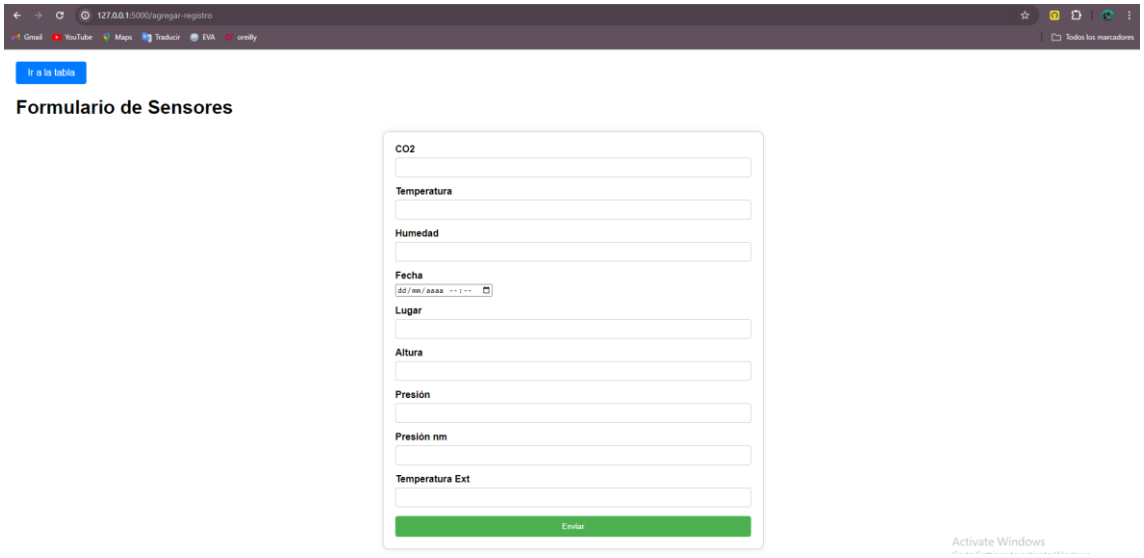


Imagen 6: Uso de la función para ingresar un dato.