PDI - TP 3 - A - RestAPI - Servidores sin control de estado - Gestión de la Base de Datos (Entrega 10/06/2024)

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 al 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 podán 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.

Terminales:
app.py

```
yr3b@DESKTOP-QFA0U13:/mnt/c/users/gonza/downloads/facultad/protocolos/tp3$ python3 app.py
 * Serving Flask app 'app'
 * Debug mode: on
WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.
 * Running on http://127.0.0.1:5000
Press CTRL+C to quit
 * Restarting with stat
 * Debugger is active!
 * Debugger PIN: 394-693-553
127.0.0.1 - - [10/Jun/2024 03:16:30] "GET /login HTTP/1.1" 500 -
```

```
127.0.0.1 - - [10/Jun/2024 03:31:11] "GET /login?next=/delete/4 HTTP/1.1" 200 -
127.0.0.1 - - [10/Jun/2024 03:35:08] "GET / HTTP/1.1" 200 -
127.0.0.1 - - [10/Jun/2024 03:36:07] "GET /datos HTTP/1.1" 200 -
127.0.0.1 - - [10/Jun/2024 03:36:09] "GET /login HTTP/1.1" 200 -
127.0.0.1 - - [10/Jun/2024 03:36:25] "POST /login HTTP/1.1" 302 -
127.0.0.1 - - [10/Jun/2024 03:36:25] "GET / HTTP/1.1" 200 -
127.0.0.1 - - [10/Jun/2024 03:36:39] "POST /delete/3 HTTP/1.1" 302 -
127.0.0.1 - - [10/Jun/2024 03:36:39] "GET / HTTP/1.1" 200 -
127.0.0.1 - - [10/Jun/2024 03:37:00] "GET / HTTP/1.1" 200 -
```

Collect_data.py

```
yr3b@DESKTOP-QFA0U13:/mnt/c/users/gonza/downloads/facultad/protocolos/tp3$ python3 collect_data.py
Resultados=  22.3 2 30 cualq clima
Lugar de la captura de los datos: Berlin
Tipo de lugar [au=abierto urbano][an=abierto no urbano] [c=cerrado] au
Superficie aproximada del lugar [m2]: 30000
Altura aproximada del lugar [m]:50
Cantidad de capturas: 1
Tiempo entre capturas(segs) : 0.8
Error al ingresar datos...
Lugar de la captura de los datos: Berlin
Tipo de lugar [au=abierto urbano][an=abierto no urbano] [c=cerrado] au
Superficie aproximada del lugar [m2]: 30000
Altura aproximada del lugar [m]:50
Cantidad de capturas: 1
Tiempo entre capturas(segs) : 1
Datos Disponibles!
CO2: 886 PPM
Temperatura: 31.26 degrees C
Humedad: 42.95 % rH
Fecha 2024-06-10 03:25:53.484397
Registro guardado... acumulados: 1


Esperando nuevo registro de datos ...
```

Add_usuario.py

```
yr3b@DESKTOP-QFA0U13:/mnt/c/users/gonza/downloads/facultad/protocolos/tp3$ python3 add_usuario.py
Nombre para nuevo usuario: MilagrosLuz
Contraseña para nuevo usuario: MLS06
Usuario 'MilagrosLuz' agregado.
```

Tablas de la base de datos cargadas

datos_sensores.users: 3 filas en total

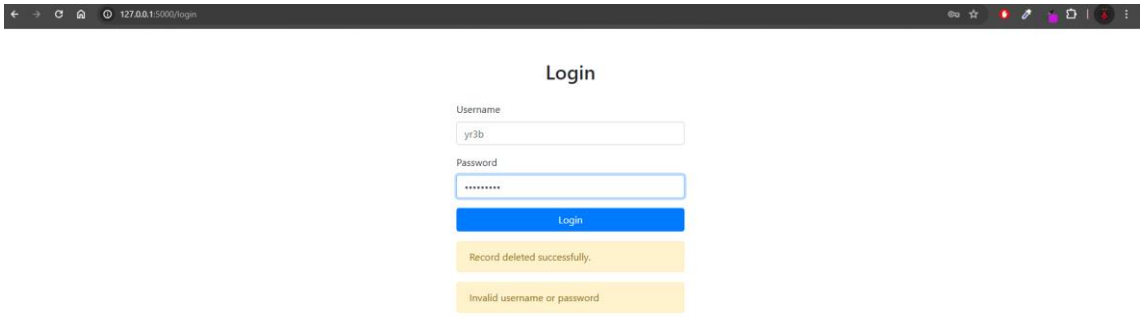
#	id	username	password
1	2	tomas	tobiot
2	3	yr3b	Gonzalo20
3	4	MilagrosLuz	MLS06

datos_sensores.lectura_sensores: 4 filas en total

Siguientes | Mostrar todo | Ordenación | Columnas (10/10) | Filtro

#	id	co2	temp	hum	fecha	lugar	altura	presion	presion_nm	temp_ext
1	1	678,513111623985	22,8985250376525	60,7476413551997	10-Jun-2024 (00:27:35.357755)	Lanus	2,0	2,0	2,0	22,3
2	2	825,783231339454	27,0826656623381	50,6774437275062	10-Jun-2024 (00:54:25.826032)	Lomas	3,0	2,0	2,0	22,3
3	4	321,050759404784	32,0323609905607	52,0223091698674	10-Jun-2024 (01:52:02.700855)	Berazategui	100,0	2,0	2,0	22,3
4	5	886,239983319836	31,2615831056883	42,9496884640688	10-Jun-2024 (03:25:53.484397)	Berlin	50,0	2,0	2,0	22,3

Login y Muestra de los datos



Sensor Data Table

ID	CO2 (PPM)	Temperature (°C)	Humidity (%)	Date	Location	Altitude (m)	Pressure	Normalized Pressure	External Temperature (°C)	Action
1	678.5131116239854	22.89852503765246	60.747641355199704	10-Jun-2024 (00:27:35.357755)	Lanus	2.0	2.0	2.0	22.3	Delete
2	825.7832313394537	27.08266566233806	50.67744372750619	10-Jun-2024 (00:54:25.826032)	Lomas	3.0	2.0	2.0	22.3	Delete
4	321.0507594047838	32.03236099056075	52.022309169867434	10-Jun-2024 (01:52:02.700855)	Berazategui	100.0	2.0	2.0	22.3	Delete
5	886.2399833198363	31.26158310568833	42.94968846406878	10-Jun-2024 (03:25:53.484397)	Berlin	50.0	2.0	2.0	22.3	Delete

Código:

app.py

```
from flask import Flask, render_template, jsonify, redirect,url_for,
request, flash
from flask_login import LoginManager, UserMixin,
login_user,login_required, logout_user, current_user
from flask_wtf import FlaskForm
from wtforms import StringField, PasswordField, SubmitField
from wtforms.validators import DataRequired
import sqlite3
import os

app = Flask(__name__)
app.config['SECRET_KEY'] = 'mysecretkey'
```

```python
# Flask-Login setup
login_manager = LoginManager()
login_manager.init_app(app)
login_manager.login_view = 'login'

# User class
class User(UserMixin):
    def __init__(self, id, username, password):
        self.id = id
        self.username = username
        self.password = password

# Database setup
def create_table():
    conn = sqlite3.connect('datos_sensores.db')
    cursor = conn.cursor()

# Create lectura_sensores table
    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
    )''')

    # Create users table
    cursor.execute('''CREATE TABLE IF NOT EXISTS users (
    id INTEGER PRIMARY KEY,
    username TEXT UNIQUE,
    password TEXT
    )''')
    conn.commit()
    conn.close()

# Create tables
create_table()

# Login manager user loader
@login_manager.user_loader
def load_user(user_id):
    conn = sqlite3.connect('datos_sensores.db')
```

```
    cursor = conn.cursor()
    cursor.execute('SELECT * FROM users WHERE id = ?', (user_id,))
    user = cursor.fetchone()
    conn.close()
    if user:
        return User(id=user[0], username=user[1], password=user[2])
    return None

# Forms
class LoginForm(FlaskForm):
    username = StringField('Username',validators=[DataRequired()])
    password = PasswordField('Password',validators=[DataRequired()])
    submit = SubmitField('Login')

@app.route('/login', methods=['GET', 'POST'])
def login():
    form = LoginForm()
    if form.validate_on_submit():
        conn = sqlite3.connect('datos_sensores.db')
        cursor = conn.cursor()
        cursor.execute('SELECT * FROM users WHERE username =
?',(form.username.data,))

        user = cursor.fetchone()
        conn.close()
        if user and user[2] == form.password.data:
            user_obj = User(id=user[0], username=user[1],
password=user[2])
        login_user(user_obj)
        return redirect(url_for('index'))
    else:
        flash('Invalid username or password')
    return render_template('login.html', form=form)

@app.route('/logout')
@login_required
def logout():
    logout_user()
    return redirect(url_for('login'))

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

    conn = sqlite3.connect('datos_sensores.db')
    cursor = conn.cursor()
    cursor.execute('SELECT * FROM lectura_sensores')
    records = cursor.fetchall()
    conn.close()
```

```
    return render_template('tabla_sensores_para_editar.html',
records=records)

@app.route('/datos')
@login_required
def datos():
    conn = sqlite3.connect('datos_sensores.db')
    cursor = conn.cursor()
    cursor.execute('SELECT * FROM lectura_sensores')
    records = cursor.fetchall()
    conn.close()
    return jsonify([{
        '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])

# Route to delete a record
@app.route('/delete/<int:id>', methods=['POST'])
@login_required
def delete_record(id):
    conn = sqlite3.connect('datos_sensores.db')
    cursor = conn.cursor()
    cursor.execute('DELETE FROM lectura_sensores WHERE id = ?',
(id,))
    conn.commit()
    conn.close()
    flash('Record deleted successfully.')
    return redirect(url_for('index'))

if __name__ == '__main__':
    app.run(debug=True, host='127.0.0.1', port=5000)
```

Add_usuario.py

```
import sqlite3

def add_user(username, password):
    conn = sqlite3.connect('datos_sensores.db')
    cursor = conn.cursor()
    cursor.execute('INSERT OR REPLACE INTO users (username, password)
VALUES(?, ?)', (username, password))
    conn.commit()
    conn.close()

if __name__ == "__main__":
    # Solicita el nombre de usuario y la contraseña para el nuevo usuario
    username = input("Nombre para nuevo usuario: ")
    password = input("Contraseña para nuevo usuario: ")
    # Agrega el usuario a la base de datos
    add_user(username, password)
    print(f"Usuario '{username}' agregado.")
```

collect_data.py

```
import time
import random
import sqlite3
from datetime import datetime


def collect_data():
    geo_latlon = (22.3, 2, 30, "cualq clima")
    temp_ext, presion, humedad_ext, descripcion_clima = geo_latlon
    print("Resultados= ", temp_ext, presion, humedad_ext,
descripcion_clima)

    while True:
        try:
            lugar = input("Lugar de la captura de los datos: ")
            tipo_lugar = input("Tipo de lugar [au=abierto
urbano][an=abierto no urbano] [c=cerrado] ")
            superficie = int(input("Superficie aproximada del lugar [m2]:
"))
            altura = int(input("Altura aproximada del lugar [m]:"))
            presion_nm = presion
            cant_capturas = int(input("Cantidad de capturas: "))
            delta_t_capturas = int(input("Tiempo entre capturas(segs) :
"))
        except ValueError:
            print("Error al ingresar datos...")
            continue
```

```
        else:
            break

    cont = 0
    while cont < cant_capturas:
        cont += 1
        print("Datos Disponibles!")
        CO2_medido = random.uniform(250, 1100)
        temp_sensor = random.uniform(temp_ext, temp_ext + 10)
        humedad_relativa = random.uniform(40, 80)
        print("CO2: %d PPM" % CO2_medido)
        print("Temperatura: %0.2f degrees C" % temp_sensor)
        print("Humedad: %0.2f %% rH" % humedad_relativa)
        d = datetime.now()

        print("Fecha", d)
        timestampStr = d.strftime("%d-%b-%Y (%H:%M:%S.%f)")

        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_medido, temp_sensor, humedad_relativa,
        timestampStr, lugar, altura, presion, presion_nm, temp_ext))
        conn.commit()
        conn.close()

        print("Registro guardado... acumulados:", cont, "\n")
        time.sleep(delta_t_capturas)
        print("\nEsperando nuevo registro de datos ...\n")

if __name__ == '__main__':
    collect_data()
```

Html utilizados:

Login.html

```
<!DOCTYPE html>
<html lang="en">

<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-
scale=1.0">
    <title>Login</title>
    <link rel="stylesheet"
href="https://stackpath.bootstrapcdn.com/bootstrap/4.5.2/css/bootstrap.mi
n.css">
    <style>
        body {
            padding-top: 50px;
        }
        .container {
            max-width: 400px;
        }
        h2 {
            text-align: center;
            margin-bottom: 30px;
        }
    </style>
</head>

<body>
    <div class="container">
        <h2>Login</h2>
        <form method="POST" action="{{ url_for('login') }}">
            {{ form.hidden_tag() }}
            <div class="form-group">
                {{ form.username.label }}
                {{ form.username(class="form-control") }}
            </div>
            <div class="form-group">
                {{ form.password.label }}
                {{ form.password(class="form-control") }}
            </div>
            <div class="form-group text-center">
                {{ form.submit(class="btn btn-primary btn-block") }}
            </div>
        </form>
        {% for message in get_flashed_messages() %}
        <div class="alert alert-warning mt-3">{{ message }}</div>
        {% endfor %}
```

```
    </div>
    <script src="https://code.jquery.com/jquery-
3.5.1.slim.min.js"></script>
    <script
src="https://cdn.jsdelivr.net/npm/@popperjs/core@2.5.4/dist/umd/popper.mi
n.js"></script>
    <script
src="https://stackpath.bootstrapcdn.com/bootstrap/4.5.2/js/bootstrap.min.
js"></script>
</body>

</html>
```

Tabla_sensores_para_editar.html

```
<!DOCTYPE html>
<html lang="en">

<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-
scale=1.0">
    <title>Sensor Data Table</title>
    <link rel="stylesheet"
href="https://stackpath.bootstrapcdn.com/bootstrap/4.5.2/css/bootstrap.mi
n.css">
    <style>
        body {
            padding-top: 50px;
        }
        .container {
            max-width: 800px;
        }
        .table-container {
            display: flex;
            justify-content: center;
        }
        .table th, .table td {
            vertical-align: middle;
            text-align: center;
        }
        h1 {
            text-align: center;
            margin-bottom: 30px;
```

```
        }
    </style>
</head>

<body>
    <div class="container">
        <h1>Sensor Data Table</h1>
        <div class="table-container">
            <table class="table table-bordered">
                <thead class="thead-dark">
                    <tr>
                        <th>ID</th>
                        <th>CO2 (PPM)</th>
                        <th>Temperature (°C)</th>
                        <th>Humidity (%)</th>
                        <th>Date</th>
                        <th>Location</th>
                        <th>Altitude (m)</th>
                        <th>Pressure</th>
                        <th>Normalized Pressure</th>
                        <th>External Temperature (°C)</th>
                        <th>Action</th>
                    </tr>
                </thead>
                <tbody>
                    {% for record in records %}
                    <tr>
                        <td>{{ record[0] }}</td>
                        <td>{{ record[1] }}</td>
                        <td>{{ record[2] }}</td>
                        <td>{{ record[3] }}</td>
                        <td>{{ record[4] }}</td>
                        <td>{{ record[5] }}</td>
                        <td>{{ record[6] }}</td>
                        <td>{{ record[7] }}</td>
                        <td>{{ record[8] }}</td>
                        <td>{{ record[9] }}</td>
                        <td>
                            <form action="{{ url_for('delete_record',
id=record[0]) }}" method="post">
                                <button type="submit" class="btn btn-
danger btn-sm">Delete</button>
                            </form>
                        </td>
                    </tr>
                    {% endfor %}
                </tbody>
            </table>
        </div>
```

```
    </div>
    <script src="https://code.jquery.com/jquery-
3.5.1.slim.min.js"></script>
    <script
src="https://cdn.jsdelivr.net/npm/@popperjs/core@2.5.4/dist/umd/popper.mi
n.js"></script>
    <script
src="https://stackpath.bootstrapcdn.com/bootstrap/4.5.2/js/bootstrap.min.
js"></script>
</body>

</html>
```