Universidad Católica Argentina

Facultad de Ingeniería y Ciencias Agrarias

Protocolos de Internet
1er Cuatrimestre 2024
Profesor/a: Javier Adolfo Ouret

Trabajo Práctico Nº 3: A-RestAPI- Servidores sin control de estado-Gestión de la Base de Datos

Integrantes:

N°	Apellido y Nombre	Carrera	Legajo	E mail
1	Mateo Denti	Informática	152151542	mate.denti@gmail.com
2	Maximo Luca Soriano Sergi	Informática	152154633	maximo.soriano170103gmail.com
3	Tomas Cortina	Informática	152155669	tomas.cortina@gmail.com
4	Thiago Ezequiel Fantino	Informática	152153181	tf@gmail.com.ar

Corrección:

Entrega 1	Devolución 1	Entrega 2	Nota

Introducción:

Este informe detalla el desarrollo de una aplicación web basada en Flask para la gestión y visualización de datos de sensores ambientales. La aplicación permite registrar y consultar lecturas de sensores de CO2, temperatura, humedad, presión y otros parámetros ambientales en diferentes ubicaciones y condiciones. Además, se ha implementado un sistema de autenticación para asegurar que solo usuarios autorizados puedan acceder y gestionar la información.

La aplicación está respaldada por una base de datos SQLite que almacena tanto los datos de las lecturas de los sensores como la información de los usuarios. Se ha desarrollado un script adicional para facilitar la adición de nuevos usuarios a la base de datos, garantizando un proceso de gestión de usuarios sencillo y seguro.

A lo largo del informe, se detallarán los componentes clave del sistema, incluyendo la estructura de la base de datos, las funcionalidades implementadas en la aplicación Flask, y el proceso para agregar usuarios. Este desarrollo proporciona una solución efectiva y escalable para la gestión de datos ambientales, asegurando la integridad y seguridad de la información.

Funcionalidad:

Creación y Gestión de la Base de Datos:

- Creación de Tablas: El código incluye una función para crear dos tablas en una base de datos SQLite: lectura_sensores para almacenar las lecturas de los sensores y users para almacenar información de usuarios.
- Inserción de Datos de Sensores: El código simula la captura de datos de sensores (CO2, temperatura, humedad, etc.) y los inserta en la tabla lectura_sensores.

Interfaz Web con Flask:

- Visualización de Datos: La aplicación Flask permite visualizar los datos de sensores almacenados en la base de datos a través de una interfaz web.
- Autenticación de Usuarios: Se ha implementado un sistema de autenticación para que solo usuarios registrados puedan acceder a la aplicación. Utiliza Flask-Login para gestionar sesiones de usuario.
- Login y Logout: Se proporcionan rutas para iniciar sesión (/login) y cerrar sesión (/logout).

Script para Agregar Usuarios:

- Añadir Usuarios: Un script independiente (add_user.py) permite agregar nuevos usuarios a la base de datos solicitando un nombre de usuario y una contraseña.

Terminales:

app.py:

```
/bin/python3 /home/mateo/Protocolos/SENSORES/app.py
mateo@MateLenovo:~/Protocolos/SENSORES$ /bin/python3 /home/mateo/Protocolos/SENSORES
/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: 560-336-029
127.0.0.1 - - [09/Jun/2024 17:50:28] "GET / HTTP/1.1" 302 -
127.0.0.1 - - [09/Jun/2024 17:50:28] "GET /login?next=/ HTTP/1.1" 200 -
/bin/python3 /home/mateo/Protocolos/SENSORES/add_user.py
/bin/python3 /home/mateo/Protocolos/SENSORES/add_user.py
/bin/python3 /home/mateo/Protocolos/SENSORES/app.py
127.0.0.1 - - [09/Jun/2024 17:52:34] "GET /login?next=/ HTTP/1.1" 200 -
127.0.0.1 - - [09/Jun/2024 17:52:39] "POST /login HTTP/1.1" 302 -
127.0.0.1 - - [09/Jun/2024 17:52:39] "GET / HTTP/1.1" 200 -
127.0.0.1 - - [09/Jun/2024 17:53:42] "GET / HTTP/1.1" 200 -
```

Collect_data.py:

```
mateo@MateLenovo:~/Protocolos/SENSORES$ python3 collect_data.py
Resultados=  22.3 2 30 cualq clima
Lugar de la captura de los datos: La casa de tf
Tipo de lugar [au=abierto urbano] [an=abierto no urbano] [c=cerrado] c
Superficie aproximada del lugar [m2]: 10
Altura aproximada del lugar [m]: 1
Cantidad de capturas: 1
Tiempo entre capturas (segs) : 1
Datos Disponibles!
CO2: 1082 PPM
Temperatura: 30.98 degrees C
Humedad: 55.08 % rH
Fecha 2024-06-09 17:53:37.804062
Registro insertado..., acumulados: 1


Esperando nuevo registro de datos ...
```

add_user.py:

```
mateo@MateLenovo:~/Protocolos/SENSORES$ python3 add_user.py
Enter username: mateo
Enter password: mateo
User 'mateo' added to the database.
```

Código:

Modificamos el código disponible en el repositorio de github para mejorar la edición del mismo y la modularidad.
Además modificamos las librerías importadas debido a que el firewall inhibe las solicitudes GET y POST emitidas desde el suscriptor.

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'

# 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)
```

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

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

add_user.py:

```
import sqlite3

def add_user(username, password):
    conn = sqlite3.connect('datos_sensores.db')
    cursor = conn.cursor()
    cursor.execute('INSERT 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("Enter username: ")
    password = input("Enter password: ")

    # Agrega el usuario a la base de datos
    add_user(username, password)
    print(f"User '{username}' added to the database.")
```

tabla_sensores_para_editar.py:

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

<head>
    <meta charset="UTF-8">
    <title>Sensor Data Table</title>
    <link rel="stylesheet"
href="https://stackpath.bootstrapcdn.com/bootstrap/4.5.2/css/boots
trap.min.css">
</head>

<body>
    <div class="container">
        <h1 class="mt-5">Sensor Data Table</h1>
        <table class="table table-bordered mt-3">
            <thead>
                <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">Delete</button>
                        </form>
                    </td>
                </tr>
                {% endfor %}
            </tbody>
        </table>
    </div>
</body>

</html>
```

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
href="https://maxcdn.bootstrapcdn.com/bootstrap/4.5.2/css/bootstra
p.min.css" rel="stylesheet">
</head>

<body>
    <div class="container">
        <h2 class="my-4 text-center">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">
                {{ form.submit(class="btn btn-primary") }}
            </div>
        </form>
        {% for message in get_flashed_messages() %}
        <div class="alert alert-warning">{{ 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/po
pper.min.js"></script>
    <script
src="https://maxcdn.bootstrapcdn.com/bootstrap/4.5.2/js/bootstrap.
min.js"></script>
</body>

</html>
```

Display Login:

Login

Username

mateo

Password

•••••

Login

Login

Username

mateo

Password

•••••

Login

Invalid username or password

Finalmente, el display de los datos:

Sensor Data

ID	CO2 (PPM)	Temperature (°C)	Humidity (%)	Date	Location	Height (m)	Pressure (Pa)	Pressure (NM)	External Temperature (°C)
1	1046.5283328793998	26.90222171570493	52.31788670315935	09-Jun-2024 (17:22:08.071041)	Tucuman	400.0	2.0	2.0	22.3
2	773.0510746913161	24.465427792647898	63.69578943201431	09-Jun-2024 (17:22:11.092873)	Tucuman	400.0	2.0	2.0	22.3
3	562.1926990845012	26.89348224980286	68.33183757686582	09-Jun-2024 (17:31:10.674882)	Salta	7000.0	2.0	2.0	22.3
4	940.4283283726229	30.744319008724453	73.61024195505078	09-Jun-2024 (17:31:11.693802)	Salta	7000.0	2.0	2.0	22.3
5	742.5049520592763	28.5159143899539	47.66719146901576	09-Jun-2024 (17:31:12.707979)	Salta	7000.0	2.0	2.0	22.3
6	477.17750430018305	28.921383837468493	58.71982029389649	09-Jun-2024 (17:31:13.728982)	Salta	7000.0	2.0	2.0	22.3
7	968.010868910545	24.665202470368545	68.05230385173148	09-Jun-2024 (17:31:14.749158)	Salta	7000.0	2.0	2.0	22.3
8	1082.6707833235218	30.983848638520485	55.08284675941441	09-Jun-2024 (17:53:37.804062)	La casa de tf	1.0	2.0	2.0	22.3