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

Integrantes:

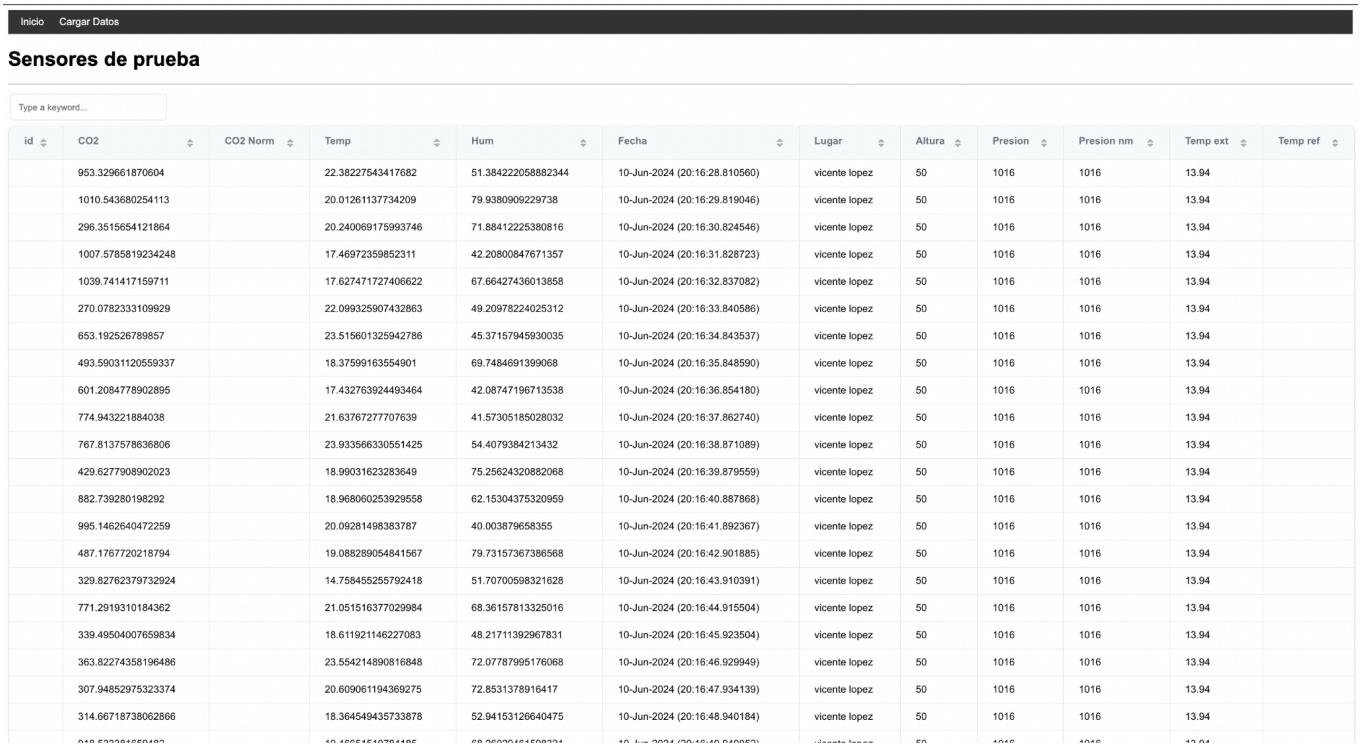
N°	Apellido y Nombre	Carrera	Legajo	E mail
1	Lucas Deberbieri	Informática	151903205	lucasdeb@uca.edu.ar
2	Sebastian Lernoud	Informática	151902738	sebaslernoud@uca.edu.ar
3	Andrés Luza	Informática		andresluza@uca.edu.ar
4	Gonzalo Crucitta	Informática	151903397	gonzalocrucitta@uca.edu.ar

Introducción:

El objetivo del trabajo practico fue mejorar el Flask App creado por el profesor, agregandole funcionalidad e interactuar con la REST API de weather para sacar los datos e informacion del clima. Usamos Flask (Python), HTML y SQLite para la configuracion del trabajo practico. Con todo configurado, logramos mostrar los datos a traves de la pagina en la tabla de los datos solicitados de la API de weather. Nos enfocamos en agregarle la funcionalidad de poder cargar los datos para la solicitud directamente desde la pagina, con un formulario que puede llenar el usuario en base a los datos que quiere mostrar en la table. Los datos se pasan a la base de datos que configuramos a traves de la API para guardarlos y despues mostrarlos en la tabla.

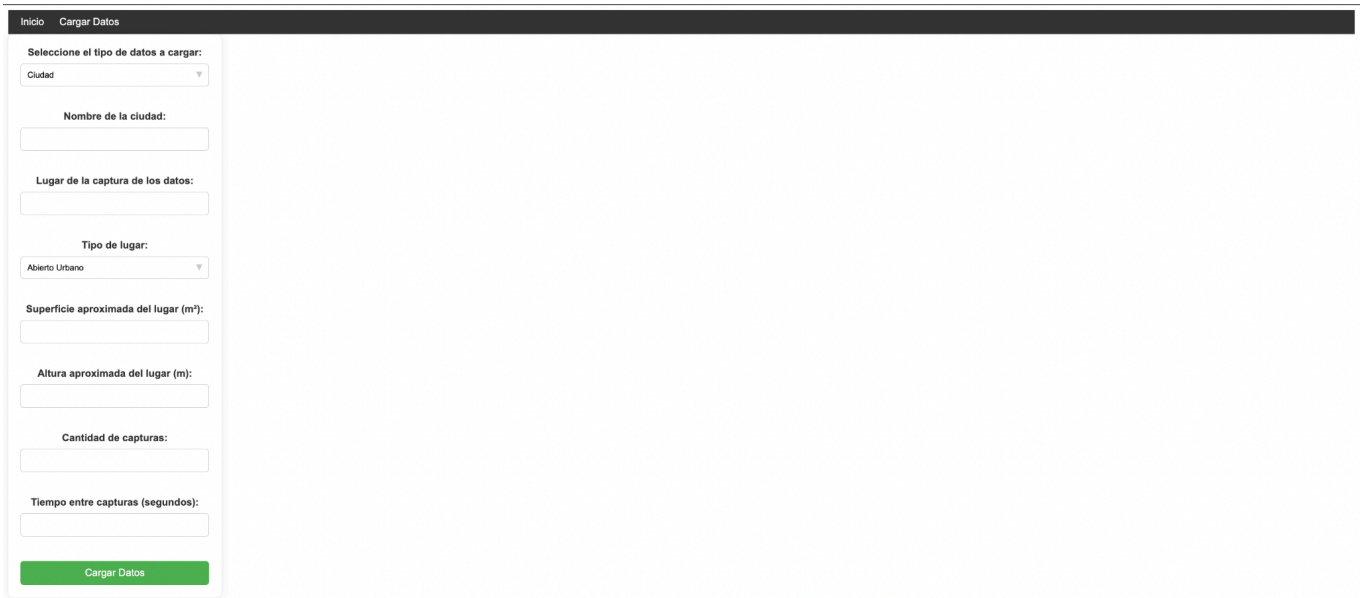
Configuración:

La configuracion de SQLite y Flask se hizo descargando las librerias correspondientes. Corriendo el codigo para la creacion de la tabla vacia, logramos crear el archivo **datos_sensores.db**, que contiene la base de datos para la carga de los datos. A continuacion se muestran las dos paginas corriendo en el localhost. En la primera pagina se ven los datos que se solicitaron, cargados en la base de datos. Despues de cada solicitud, se muestran los datos cargados a la base de datos y mostrados en la tabla principal.



id	CO2	CO2 Norm	Temp	Hum	Fecha	Lugar	Altura	Presion	Presion nm	Temp ext	Temp ref
	953.329661870604		22.38227543417682	51.384222058882344	10-Jun-2024 (20:16:28.810560)	vicente lopez	50	1016	1016	13.94	
	1010.543680254113		20.01261137734209	79.9380909229738	10-Jun-2024 (20:16:29.819046)	vicente lopez	50	1016	1016	13.94	
	296.3515654121864		20.240069175993746	71.88412225380816	10-Jun-2024 (20:16:30.824546)	vicente lopez	50	1016	1016	13.94	
	1007.5785819234248		17.469723598523 11	42.20800847671357	10-Jun-2024 (20:16:31.828723)	vicente lopez	50	1016	1016	13.94	
	1039.741417159711		17.627471727406622	67.66427436013858	10-Jun-2024 (20:16:32.837082)	vicente lopez	50	1016	1016	13.94	
	270.0782333109929		22.099325907432863	49.20978224025312	10-Jun-2024 (20:16:33.840586)	vicente lopez	50	1016	1016	13.94	
	653.192526789857		23.5156013259427 86	45.37157945930035	10-Jun-2024 (20:16:34.843537)	vicente lopez	50	1016	1016	13.94	
	493.59031120559337		18.37599163554901	69.7484691399068	10-Jun-2024 (20:16:35.848590)	vicente lopez	50	1016	1016	13.94	
	601.2084778902895		17.432763924493464	42.08747196713538	10-Jun-2024 (20:16:36.854180)	vicente lopez	50	1016	1016	13.94	
	774.943221884038		21.63767277707639	41.57305185028032	10-Jun-2024 (20:16:37.862740)	vicente lopez	50	1016	1016	13.94	
	767.8137578636806		23.933566330551425	54.4079384213432	10-Jun-2024 (20:16:38.871089)	vicente lopez	50	1016	1016	13.94	
	429.6277908902023		18.99031623283649	75.25624320882068	10-Jun-2024 (20:16:39.879559)	vicente lopez	50	1016	1016	13.94	
	882.739280198292		18.968060253929558	62.15304375320959	10-Jun-2024 (20:16:40.887868)	vicente lopez	50	1016	1016	13.94	
	995.1462640472259		20.09281498383787	40.003879658355	10-Jun-2024 (20:16:41.892367)	vicente lopez	50	1016	1016	13.94	
	487.1767720218794		19.088289054841567	79.73157367386568	10-Jun-2024 (20:16:42.901885)	vicente lopez	50	1016	1016	13.94	
	329.82762379732924		14.758455255792418	51.70700598321628	10-Jun-2024 (20:16:43.910391)	vicente lopez	50	1016	1016	13.94	
	771.2919310184362		21.051516377029984	68.36157813325016	10-Jun-2024 (20:16:44.915504)	vicente lopez	50	1016	1016	13.94	
	339.49504007659834		18.611921146227083	48.21711392967831	10-Jun-2024 (20:16:45.923504)	vicente lopez	50	1016	1016	13.94	
	363.82274358196486		23.554214890816848	72.07787995176068	10-Jun-2024 (20:16:46.929949)	vicente lopez	50	1016	1016	13.94	
	307.94852975323374		20.609061194369275	72.8531378916417	10-Jun-2024 (20:16:47.934139)	vicente lopez	50	1016	1016	13.94	
	314.66718738062866		18.364549433733878	52.94153126640475	10-Jun-2024 (20:16:48.940184)	vicente lopez	50	1016	1016	13.94	

A continuacion, mostramos el formulario creado para habilitar que el usuario pueda hacer el request por POST directamente desde la pagina. Se introducen todos los datos en el campo correspondiente, que despues se pasan para el request de la API de weather.



El codigo principal de **sensores_r2.py** modificado que se encarga de manejar las rutas, conexiones funcionales a la base de datos y la logica de las solicitudes del formulario se encuentran detalladas abajo.

```
import time
import random
import sqlite3
from flask import Flask, render_template, jsonify, request, url_for, redirect
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()


@app.route('/')
def index():
    conn = sqlite3.connect('datos_sensores.db')
    cursor = conn.cursor()
    cursor.execute('SELECT * FROM lectura_sensores ORDER BY fecha')
    records = cursor.fetchall()
    conn.close()
    return render_template('tabla_sensores_para_editar.html', records=records)


@app.route('/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([{
        '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])


@app.route('/cargar_datos', methods=['GET', 'POST'])
def cargar_datos():
    create_table()

    if request.method == 'POST':
        temp_ext, presion, humedad_ext, descripcion_clima = geo_latlon(
            request.form['opcion_elegida'], request.form['city_name'])
        print("Resultados= ", temp_ext, presion,
              humedad_ext, descripcion_clima)
        lugar = request.form['lugar']
        tipo_lugar = request.form['tipo_lugar']
        superficie = request.form['superficie']
        altura = float(request.form['altura'])
        presion_nm = presion
        cant_capturas = int(request.form['cant_capturas'])
        delta_t_capturas = int(request.form['delta_t_capturas'])

        cont = 0
        while cont < cant_capturas:
            cont += 1
            verdadero = 1
            if verdadero == 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")
        return redirect(url_for('index'))

    print("Cierro conexión ...")
    return render_template('cargar_datos.html')


if __name__ == '__main__':
    app.run(debug=True)
```

La aplicacion Flask de **sensores_r2.py** se encarga principalmente conectarse con la base de datos que creamos, cargar los datos, configurar las rutas correspondientes con la logica requerida para mostrarlo a traves del webserver local. La funcion create_table(), se encarga de crear la tabla dentro de **datos_sensores.db**, con las columnas necesarias para poder cargar los datos solicitados de manera correcta. En la ruta principal “index”, representado por ”/” como la ruta principal, se encarga de crear la conexion con la base de datos, y hacer el query para poder mostrar los datos que se encuentran ya cargados. De ahi muestra el archivo HTML principal **tabla_sensores_para_editar.html** que tiene el formato necesario para mostrar los datos en la pagina. La logica de la solicitud a la API de weather se encuentra detallada dentro del archivo **funciones.py**.

```
def geo_latlon(op1, op2):
    import geocoder
    import requests
    import json

    g = geocoder.ip('me')
    print(g.latlng)
    lat, lon = g.latlng
    print("Lat =", lat, "Lon = ", lon)

    api_key = "2f66bd561ebc7e4bde0d2a8951df0098"

    # base_url variable to store url
    base_url = "http://api.openweathermap.org/data/2.5/weather?"

    # Give city name
```

```
# city_name = input("Ciudad: ")
print("Ciudad o Geo ? = ")
opcion = ['ciudad', 'geo']
opcion_elegida = ''
mensaje = "Elegir opción:\n"
for index, item in enumerate(opcion):
    mensaje += f'{index + 1}) {item}\n'
mensaje += 'Opción:'

while opcion_elegida.lower() not in opcion:
    opcion_elegida = op1

print('Seleccionado: ' + opcion_elegida)

# complete_url variable to store
# complete url address
if opcion_elegida == 'ciudad':
    city_name = op2
    complete_url = base_url + "appid=" + api_key + \
        "&q=" + city_name + "&units=metric" + "&lang=es"
else:
    complete_url = base_url + "lat=" + str(lat) + "&lon=" + str(
        lon) + "&appid=" + api_key + "&units=metric" + "&lang=es"
print(complete_url)
# get method of requests module
# return response object
response = requests.get(complete_url)

# json method of response object
# convert json format data into
# python format data
x = response.json()
print(x)
print(x["cod"])

# Now x contains list of nested dictionaries
# Check the value of "cod" key is equal to
# "404", means city is found otherwise,
# city is not found
if x["cod"] != "404" and x["cod"] != "400":

    # store the value of "main"
    # key in variable y
    y = x["main"]

    # store the value corresponding
    # to the "temp" key of yRio
    temp_ext = y["temp"]
```

```
        # store the value corresponding
        # to the "pressure" key of y
        presion = y["pressure"]

        # store the value corresponding
        # to the "humidity" key of y
        humedad_ext = y["humidity"]

        # store the value of "weather"
        # key in variable z
        z = x["weather"]

        # store the value corresponding
        # to the "description" key at
        # the 0th index of z
        descripcion_clima = z[0]["description"]

        # print following values
        print(" Temperatura = " +
              str(temp_ext)[0:5] + " C" +
              "\n Presión Atmosférica = " +
              str(presion) + " hPa" +
              "\n Humedad  = " +
              str(humedad_ext) + " %" +
              "\n Cielo = " +
              str(descripcion_clima))

    else:
        print(" Ciudad no encontrada ")
    return temp_ext, presion, humedad_ext, descripcion_clima
```

Reemplazando los datos necesarios, pasados por parametro por el formulario, como las variables “op1” y “op2” en funciones.py, tenemos los datos necesarios para hacer la solicitud al API de weather y obtener el response. Se retornan los datos de la temperatura, presion, humedad_ext, y descripcion_clima a la funcion cargar_datos() en sensores_r2.py, donde se insertan los valores en la base de datos.

A continuación, mostramos el archivo HTML del formulario para que el usuario llene los datos para el request.

```
<!DOCTYPE html>
<html>

<head>
    <title>Sensores de prueba</title>
    <link href="https://unpkg.com/gridjs/dist/theme/mermaid.min.css" rel="stylesheet" />
    <style>
        nav {
```

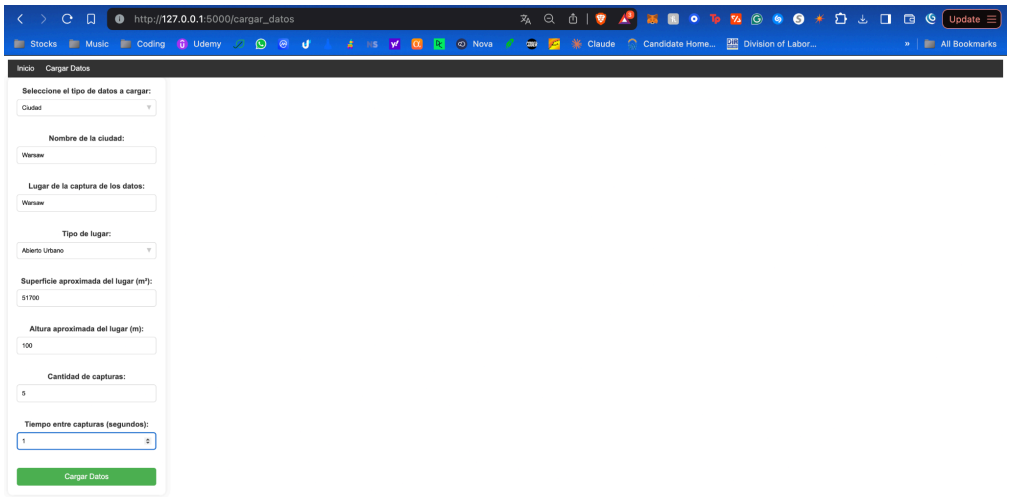
```
        background-color: #333;
        color: white;
        padding: 10px;
    }

    nav a {
        color: rgb(255, 255, 255);
        text-decoration: none;
        margin: 0 10px;
    }

    body {
        font-family: Arial, sans-serif;
        justify-content: center;
    }

    .tabla {
        text-align: center;
    }

    form {
        background: #fff;
        padding: 20px;
        border-radius: 10px;
        box-shadow: 0 0 10px rgba(0, 0, 0, 0.1);
        width: 300px;
    }

    form label {
        display: block;
        margin-bottom: 8px;
        font-weight: bold;
        color: #333;
    }

    form input[type="text"],
    form input[type="number"],
    form select {
        width: 100%;
        padding: 10px;
        margin-bottom: 20px;
        border: 1px solid #ccc;
        border-radius: 5px;
        box-sizing: border-box;
    }

    form input[type="submit"] {
        background-color: #4CAF50;
        color: white;
```

```
            padding: 10px 15px;
            border: none;
            border-radius: 5px;
            cursor: pointer;
            width: 100%;
            font-size: 16px;
        }

        form input[type="submit"]:hover {
            background-color: #45a049;
        }

        form select {
            appearance: none;
            background: #fff;
            background-image: url('data:image/svg+xml,<svg
xmlns="http://www.w3.org/2000/svg" width="10" height="10" viewBox="0 0 10 10"><polygon
points="0,0 10,0 5,10" fill="%23ccc"/></svg>');
            background-repeat: no-repeat;
            background-position: right 10px center;
            background-size: 10px;
        }
    </style>
</head>

<body>
    <div>
        <nav>
            <a href="/">Inicio</a>
            <a href="/cargar_datos">Cargar Datos</a>
        </nav>
    </div>
    <div class="tabla">
        <form method="POST" action="/cargar_datos">
            <label for="opcion_elegida" required>Seleccione el tipo de datos a
cargar:</label>
            <select id="opcion_elegida" name="opcion_elegida">
                <option value="ciudad">Ciudad</option>
                <option value="geo">Geo</option>
            </select><br><br>

            <label for="city_name">Nombre de la ciudad:</label>
            <input type="text" id="city_name" name="city_name"><br><br>

            <label for="lugar">Lugar de la captura de los datos:</label>
            <input type="text" id="lugar" name="lugar" required><br><br>

            <label for="tipo_lugar">Tipo de lugar:</label>
            <select id="tipo_lugar" name="tipo_lugar">
```

```
                <option value="au">Abierto Urbano</option>
                <option value="an">Abierto No Urbano</option>
                <option value="c">Cerrado</option>
            </select><br><br>

            <label for="superficie">Superficie aproximada del lugar (m²):</label>
            <input type="number" id="superficie" name="superficie" required><br><br>

            <label for="altura">Altura aproximada del lugar (m):</label>
            <input type="number" id="altura" name="altura" required><br><br>

            <label for="cant_capturas">Cantidad de capturas:</label>
            <input type="number" id="cant_capturas" name="cant_capturas"
required><br><br>

            <label for="delta_t_capturas">Tiempo entre capturas (segundos):</label>
            <input type="number" id="delta_t_capturas" name="delta_t_capturas"
required><br><br>

            <input type="submit" value="Cargar Datos">

        </form>
    </div>
</body>

</html>
```

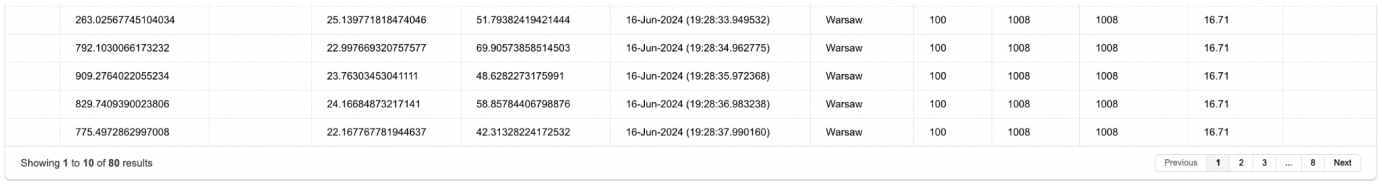
Ejemplo de Uso:

Cargamos los datos en la ruta “/cargar_datos” usando Varsovia como ciudad de ejemplo.



Una vez que hacemos la solicitud por POST, se cargan los datos dentro de la funcion cargar_datos() en sensores_r2.py, armando el request a la API de weather y cargando los datos

recibidos del request a la tabla en nuestra base de datos. Finalmente, se muestran los datos en la pagina index con los resultados del request.



	263.02567745104034		25.139771818474046	51.79382419421444	16-Jun-2024 (19:28:33.949532)	Warsaw	100	1008	1008	16.71	
	792.1030066173232		22.997669320757577	69.90573858514503	16-Jun-2024 (19:28:34.962775)	Warsaw	100	1008	1008	16.71	
	909.2764022055234		23.76303453041111	48.6282273175991	16-Jun-2024 (19:28:35.972368)	Warsaw	100	1008	1008	16.71	
	829.7409390023806		24.16684873217141	58.85784406798876	16-Jun-2024 (19:28:36.983238)	Warsaw	100	1008	1008	16.71	
	775.4972862997008		22.167767781944637	42.31328224172532	16-Jun-2024 (19:28:37.990160)	Warsaw	100	1008	1008	16.71	

Showing **1** to **10** of **80** results

Previous **1** 2 3 ... 8 Next