

TP 3-A

Consignas:

Instalar Heidi SQL en Windows

Para Linux y Windows se puede usar DBeaver Community Edition.

Tambien 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 de Python importar 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 puede encontrar las versiones r2 con algunas modificaciones

sensores_r1 genera una db llamada datos_sensores.db

sensores_editar_tabla_r2 agrega nuevas rutas y mensajes

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

@app.route('/')

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

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

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

Verificar y mejorar

sensores.py:

```
1 import time
2 import random
3 import sqlite3
4 from flask import Flask, render_template, jsonify
5 from datetime import datetime
6 from funciones import geo_latlon
7
8 app = Flask(__name__)
9
10 def create_table():
11     conn = sqlite3.connect('sensores.db')
12     cursor = conn.cursor()
13     cursor.execute('''CREATE TABLE IF NOT EXISTS lectura_sensores (
14         id INTEGER PRIMARY KEY,
15         co2 REAL,
16         temp REAL,
17         hum REAL,
18         fecha TEXT,
19         lugar TEXT,
20         altura REAL,
21         presion REAL,
22         presion_nm REAL,
23         temp_ext REAL
24     )''')
25     conn.commit()
26     conn.close()
27
28 if __name__ == '__main__':
29     create_table()
30
31     temp_ext, presion, humedad_ext, descripcion_clima = geo_latlon()
32     print("Resultados= ", temp_ext, presion, humedad_ext, descripcion_clima)
33
34     while True:
35         try:
36             lugar = input("Lugar de la captura de los datos: ")
37             tipo_lugar = input("Tipo de lugar [au=abierto urbano] [an=abierto no urbano] [c=cerrado] ")
38             superficie = int(input("Superficie aproximada del lugar [m2]: "))
39             altura = int(input("Altura aproximada del lugar [m]: "))
40             presion_nm = presion
41             cant_capturas = int(input("Cantidad de capturas: "))
42             delta_t_capturas = int(input("Tiempo entre capturas (segs) : "))
43         except ValueError:
44             print("Error al ingresar datos...")
45             continue
46         else:
47             break
48
49     cont = 0
50     while cont < cant_capturas:
51         cont += 1
52         verdadero = 1
53         if verdadero == 1:
54             print("Datos Disponibles!")
55             CO2_medido = random.uniform(250, 1100)
56             temp_sensor = random.uniform(temp_ext, temp_ext + 10)
57             humedad_relativa = random.uniform(40, 80)
58             print("CO2: %d PPM" % CO2_medido)
59             print("Temperatura: %0.2f degrees C" % temp_sensor)
60             print("Humedad: %0.2f %% rH" % humedad_relativa)
61
62             d = datetime.now()
63             print("Fecha", d)
64             timestampStr = d.strftime("%d-%b-%Y (%H:%M:%S.%f)")
65
66             conn = sqlite3.connect('sensores.db')
67             cursor = conn.cursor()
68             cursor.execute('''INSERT INTO lectura_sensores (co2, temp, hum, fecha, lugar, altura, presion, presion_nm, temp_ext)
69                 VALUES (?, ?, ?, ?, ?, ?, ?, ?, ?)''',
70                 (CO2_medido, temp_sensor, humedad_relativa, timestampStr, lugar, altura, presion, presion_nm, temp_ext))
71             conn.commit()
72             conn.close()
73
74             print("Registro insertado..., acumulados:", cont, "\n")
75             time.sleep(delta_t_capturas)
76             print("\nEsperando nuevo registro de datos ...\n")
77
78     print("Cierro conexión ...")
```

Se modificó el código para funcionar como un programa para crear la tabla lectura_sensores, añadiendo valores a la tabla mediante el uso de funciones que ya están hechas en funciones.py

tabla_datos.html:

```
1 <html>
2 <head>
3   <title>Sensores de prueba</title>
4   <link href="https://unpkg.com/gridjs/dist/theme/mermaid.min.css" rel="stylesheet" />
5   <style>
6     body {
7       font-family: Sans-Serif;
8     }
9   </style>
10 </head>
11 <body>
12   <div>
13     <h1>Sensores de prueba</h1>
14     <h2>
15       <form action="/api/añadir" method="POST">
16         <input type="number" name="CO2" placeholder="CO2">
17         <input type="number" name="Temp" placeholder="Temp">
18         <input type="number" name="Hum" placeholder="Hum">
19         <input type="text" name="Lugar" placeholder="Lugar">
20         <input type="number" name="Altura" placeholder="Altura">
21         <input type="number" name="Presion" placeholder="Presion">
22         <input type="number" name="Presion_nm" placeholder="Presion nm">
23         <input type="number" name="Temp_ext" placeholder="Temp ext">
24
25         <input type="submit" value="Añadir">
26       </form>
27     </h2>
28     <hr>
29 <!--Table Container-->
30     <div id="table"></div>
31   </div>
32   <script src="https://unpkg.com/gridjs/dist/gridjs.umd.js"></script>
33   <script>
34     const tableDiv = document.getElementById('table');
35
36     const updateUrl = (prev, query) => {
37       return prev + (prev.indexOf('?') >= 0 ? '&' : '?') + new URLSearchParams(query).toString();
```

```
29 <!--Table Container-->
33   <script>
34     const tableDiv = document.getElementById('table');
35
36     const updateUrl = (prev, query) => {
37       return prev + (prev.indexOf('?') >= 0 ? '&' : '?') + new URLSearchParams(query).toString();
38     };
39
40     const editableCellAttributes = (data, row, col) => {
41       if (row) {
42         return {contentEditable: 'true', 'data-element-id': row.cells[0].data};
43       }
44       else {
45         return {};
46       }
47     };
48
49     new gridjs.Grid({
50       columns: [
51         { id: 'id', name: 'id', 'attributes': editableCellAttributes },
52         { id: 'co2', name: 'CO2', 'attributes': editableCellAttributes },
53         { id: 'temp', name: 'Temp', 'attributes': editableCellAttributes },
54         { id: 'hum', name: 'Hum', 'attributes': editableCellAttributes },
55         { id: 'fecha', name: 'Fecha', 'attributes': editableCellAttributes },
56         { id: 'lugar', name: 'Lugar', 'attributes': editableCellAttributes },
57         { id: 'altura', name: 'Altura', 'attributes': editableCellAttributes },
58         { id: 'presion', name: 'Presion', 'attributes': editableCellAttributes },
59         { id: 'presion_nm', name: 'Presion nm', 'attributes': editableCellAttributes },
60         { id: 'temp_ext', name: 'Temp ext', 'attributes': editableCellAttributes },
61         { id: 'eliminar', name: 'Eliminar', formatter: (_, row) => gridjs.html('<button onclick="eliminar(${row.cells[0].data})">Eliminar</button>') },
62       ],
63       server: {
64         url: '/api/datos',
65         then: results => results.data,
66         total: results => results.total,
67       },
68       search: {
69         enabled: true,
70         server: {
71           url: (prev, search) => {
72             return updateUrl(prev, {search});
```

```

29 <!--Table Container-->
33 <script>
49   new gridjs.Grid({
68     search: {
70       server: {
71         url: (prev, search) => {
72           return updateUrl(prev, {search});
73         },
74       },
75     },
76     sort: {
77       enabled: true,
78       multiColumn: true,
79       server: {
80         url: (prev, columns) => {
81           const columnIds = ['id', 'co2', 'temp', 'hum', 'fecha', 'lugar', 'altura', 'presion', 'presion_nm', 'temp_ext'];
82           const sort = columns.map(col => (col.direction === 1 ? '+' : '-') + columnIds[col.index]);
83           return updateUrl(prev, {sort});
84         },
85       },
86     },
87     pagination: {
88       enabled: true,
89       server: {
90         url: (prev, page, limit) => {
91           return updateUrl(prev, {start: page * limit, length: limit});
92         },
93       },
94     },
95   }).render(tableDiv);
96
97   let savedValue;
98
99   tableDiv.addEventListener('focusin', ev => {
100     if (ev.target.tagName === 'TD') {
101       savedValue = ev.target.textContent;
102     }
103   });

```

```

29 <!--Table Container-->
33 <script>
132   });
133
134   function eliminar(id) {
135     fetch('/api/eliminar', {
136       method: 'DELETE',
137       headers: {'Content-Type': 'application/json'},
138       body: JSON.stringify({
139         id: id.toString(),
140       }),
141     }).then(location.reload());
142   }
143 </script>
144 </body>
145 </html>
146

```

Se añadió un form arriba de la tabla, para que el usuario pueda añadir una nueva entrada a la tabla. Se arregló el código de la grid para que pueda mostrar el id de cada entrada de la tabla. Se agregó una nueva columna a la tabla ('Eliminar'), dónde hay un botón que llama a la función eliminar() que manda un pedido al servidor, mediante el método DELETE, con el id de la entrada que se desea eliminar. Luego de hacer el pedido, la función eliminar refresca la ventana del usuario para poder observar los nuevos cambios a la tabla.

sensor_editar_tabla.py:

```
1  from flask import Flask, render_template, request, abort, jsonify, red
2  import time
3  import random
4  import sqlite3
5  from datetime import datetime
6  from funciones import geo_latlon
7
8  app = Flask(__name__)
9
10 @app.route('/')
11 def index():
12     return render_template('tabla_sensores_para_editar.html')
13
14 @app.route('/api/test')
15 def test():
16     return "Prueba de ruta API..."
17
18 @app.route('/api/datos')
19 def datos():
20     conn = sqlite3.connect('sensores.db')
21     cursor = conn.cursor()
22     cursor.execute('SELECT * FROM lectura_sensores')
23     records = cursor.fetchall()
24     conn.close()
25     data = []
26     for record in records:
27         data.append(
28             {
29                 'id': record[0],
30                 'co2': record[1],
31                 'temp': record[2],
32                 'hum': record[3],
33                 'fecha': record[4],
34                 'lugar': record[5],
35                 'altura': record[6],
36                 'presion': record[7],
37                 'presion_nm': record[8],
38                 'temp_ext': record[9]
39             })
40
41     return jsonify({
42         'data': data,
43         'total': len(data),
44     })
45
46 @app.route('/api/primer-registro')
47 def primerRegistro():
48     try:
49         conn = sqlite3.connect('sensores.db')
50         cursor = conn.cursor()
51         cursor.execute("SELECT * FROM lectura_sensores")
52         primero = cursor.fetchone()
53         conn.close()
54         if not primero:
55             return jsonify({'mensaje': 'No se encontraron datos'}), 404
56         return jsonify(primero)
57     except Exception as e:
58         return jsonify({'error': 'Error al buscar datos', 'detalle': str(e)}), 500
59
60 @app.route('/api/añadir', methods=["POST"])
61 def añadir():
62     try:
63         CO2 = request.form.get('CO2')
64         Temp = request.form.get('Temp')
65         Hum = request.form.get('Hum')
66         Lugar = request.form.get('Lugar')
67         Altura = request.form.get('Altura')
68         Presion = request.form.get('Presion')
```

```
23 def datos():
37     'fecha': record[4],
38     'lugar': record[5],
39     'altura': record[6],
40     'presion': record[7],
41     'presion_nm': record[8],
42     'temp_ext': record[9]
43 )
44
45 return jsonify({
46     'data': data,
47     'total': len(data),
48 })
49
50 @app.route('/api/primer-registro')
51 def primerRegistro():
52     try:
53         conn = sqlite3.connect('sensores.db')
54         cursor = conn.cursor()
55         cursor.execute("SELECT * FROM lectura_sensores")
56         primero = cursor.fetchone()
57         conn.close()
58         if not primero:
59             return jsonify({'mensaje': 'No se encontraron datos'}), 404
60         return jsonify(primero)
61     except Exception as e:
62         return jsonify({'error': 'Error al buscar datos', 'detalle': str(e)}), 500
63
64 @app.route('/api/añadir', methods=["POST"])
65 def añadir():
66     try:
67         CO2 = request.form.get('CO2')
68         Temp = request.form.get('Temp')
69         Hum = request.form.get('Hum')
70         Lugar = request.form.get('Lugar')
71         Altura = request.form.get('Altura')
72         Presion = request.form.get('Presion')
```

```

65 def añadir():
66     Altura = request.form.get('Altura')
67     Presion = request.form.get('Presion')
68     Presion_nm = request.form.get('Presion_nm')
69     Temp_ext = request.form.get('Temp_ext')
70
71     if CO2 or Temp or Hum or Lugar or Altura or Presion or Presion_nm or Temp_ext == None:
72         abort(400)
73
74     current_datetime = datetime.now().strftime("%d-%B-%Y (%H:%M:%S)")
75
76     conn = sqlite3.connect('sensores.db')
77     cursor = conn.cursor()
78     cursor.execute("INSERT INTO lectura_sensores (co2, temp, hum, fecha, lugar, altura, presion, presion_nm, temp_ext) VALUES (?, ?, ?, ?, ?, ?, ?, ?, ?)",
79                    (CO2, Temp, Hum, current_datetime, Lugar, Altura, Presion, Presion_nm, Temp_ext))
80     conn.commit()
81     conn.close()
82
83     return redirect("/")
84 except Exception as e:
85     return jsonify({'error': 'Error al añadir datos', 'detalle': str(e)}), 500
86
87 @app.route('/api/eliminar', methods=['DELETE'])
88 def eliminar():
89     try:
90         data = request.get_json()
91         if 'id' not in data:
92             abort(400)
93
94         conn = sqlite3.connect('sensores.db')
95         cursor = conn.cursor()
96         cursor.execute("SELECT * FROM lectura_sensores WHERE id = ?", (data['id']))
97         record = cursor.fetchone()
98         conn.close()
99
100

```

```

94 def eliminar():
95
96     if not record:
97         abort(404)
98
99     conn = sqlite3.connect('sensores.db')
100     cursor = conn.cursor()
101     cursor.execute("DELETE FROM lectura_sensores WHERE id = ?", (data['id']))
102     conn.commit()
103     conn.close()
104     return '', 204
105 except Exception as e:
106     return jsonify({'error': 'Error al buscar datos', 'detalle': str(e)}), 500
107
108 @app.route('/api/data', methods=['PUT'])
109 def update():
110     try:
111         data = request.get_json()
112         if 'id' not in data:
113             abort(400)
114
115         conn = sqlite3.connect('sensores.db')
116         cursor = conn.cursor()
117         cursor.execute("SELECT * FROM lectura_sensores WHERE id = ?", (data['id']))
118         record = cursor.fetchone()
119         conn.close()
120
121         if not record:
122             abort(404)
123
124         if 'lugar' in data:
125             new_lugar = data['lugar']
126             conn = sqlite3.connect('sensores.db')
127             cursor = conn.cursor()
128             cursor.execute("UPDATE lectura_sensores SET lugar = ? WHERE id = ?", (new_lugar, data['id']))
129             conn.commit()
130             conn.close()

```

```

119 def update():
120     conn.close()
141     elif 'co2' in data:
142         new_co2 = data['co2']
143         conn = sqlite3.connect('sensores.db')
144         cursor = conn.cursor()
145         cursor.execute("UPDATE lectura_sensores SET co2 = ? WHERE id = ?", (new_co2, data['id']))
146         conn.commit()
147         conn.close()
148     elif 'temp' in data:
149         new_temp = data['temp']
150         conn = sqlite3.connect('sensores.db')
151         cursor = conn.cursor()
152         cursor.execute("UPDATE lectura_sensores SET temp = ? WHERE id = ?", (new_temp, data['id']))
153         conn.commit()
154         conn.close()
155     elif 'hum' in data:
156         new_hum = data['hum']
157         conn = sqlite3.connect('sensores.db')
158         cursor = conn.cursor()
159         cursor.execute("UPDATE lectura_sensores SET hum = ? WHERE id = ?", (new_hum, data['id']))
160         conn.commit()
161         conn.close()
162     elif 'altura' in data:
163         altura = data['altura']
164         conn = sqlite3.connect('sensores.db')
165         cursor = conn.cursor()
166         cursor.execute("UPDATE lectura_sensores SET altura = ? WHERE id = ?", (altura, data['id']))
167         conn.commit()
168         conn.close()
169     elif 'presion' in data:
170         presion = data['presion']
171         conn = sqlite3.connect('sensores.db')
172         cursor = conn.cursor()
173         cursor.execute("UPDATE lectura_sensores SET presion = ? WHERE id = ?", (presion, data['id']))
174         conn.commit()
175         conn.close()
176     elif 'presion_nm' in data:
177         presion_nm = data['presion_nm']
178         conn = sqlite3.connect('sensores.db')
179         cursor = conn.cursor()
180         cursor.execute("UPDATE lectura_sensores SET presion_nm = ? WHERE id = ?", (presion_nm, data['id']))
181         conn.commit()
182         conn.close()
183     elif 'temp_ext' in data:
184         temp_ext = data['temp_ext']
185         conn = sqlite3.connect('sensores.db')
186         cursor = conn.cursor()
187         cursor.execute("UPDATE lectura_sensores SET temp_ext = ? WHERE id = ?", (temp_ext, data['id']))
188         conn.commit()
189         conn.close()
190
191     return '', 204
192 except Exception as e:
193     return jsonify({'error': 'Error al actualizar datos', 'detalle': str(e)}), 500
194
195 if __name__ == '__main__':
196     app.run(debug=True)
197

```

Añadiendo la libreria sqlite3, se cambiaron y se añadieron algunas rutas.

"/", muestra el html al usuario.

"/api/datos", el servidor envia al cliente todas las entradas de la tabla lectura_sensores para que se agreguen a la tabla del html.

"/api/primer-registro", se devuelve al cliente la primera entrada de la tabla.

"/api/añadir", con el metodo POST, el usuario puede añadir una nueva entrada a la BD mediante el form.

"/api/eliminar", con el metodo DELETE, el usuario puede eliminar una entrada de la tabla.

"/api/data", con PUT, el usuario puede editar los campos de la tabla.

Ejemplo de eliminar con id = 5:

Sensores de prueba

CO2

Temp

Hum

Ugler

Altura

Presion

Presion mm

Temp ext

Alisar

Type a keyword...

id	o	CO2	Temp	Hum	Fecha	Lugar	Altura	Presion	Presion mm	Temp ext	Eliminar
1		758.3502469049592	18.93368742431066	55.600635561496205	09-Jun-2024 (22:50:17.970087)	Casa de Juan	50	1009	1009	16.43	Eliminar
2		487.04874911576997	21.260089038140294	40.5991491326726644	09-Jun-2024 (22:50:22.036865)	Casa de Juan	50	1009	1009	16.43	Eliminar
3		1005.5079486255445	26.08965383954648	58.27977207085695	09-Jun-2024 (22:50:26.133501)	Casa de Juan	50	1009	1009	16.43	Eliminar
4		346.6206042868657	16.86415898040444	68.44206781852616	09-Jun-2024 (22:50:36.207001)	Casa de Juan	50	1009	1009	16.43	Eliminar
5		1073.4808003641533	21.19932426996276	62.11732826771275	09-Jun-2024 (22:50:34.287179)	Casa de Juan	50	1009	1009	16.43	Eliminar
6		836.83548137609	22.870518034461146	76.6029168439301	09-Jun-2024 (22:50:38.367856)	Casa de Juan	50	1009	1009	16.43	Eliminar
7		1044.8742759659065	22.876038827914643	68.18416788010104	09-Jun-2024 (22:50:42.440022)	Casa de Juan	50	1009	1009	16.43	Eliminar
8		553.3255344754824	19.63097530816683	68.60379583911019	09-Jun-2024 (22:50:46.516717)	Casa de Juan	50	1009	1009	16.43	Eliminar
9		916.0641634501849	22.9494719328777	61.857078204383924	09-Jun-2024 (22:50:58.600274)	Casa de Juan	50	1009	1009	16.43	Eliminar
10		317.16878114578776	20.486118883616227	49.99579620835955	09-Jun-2024 (22:50:54.665936)	Casa de Juan	50	1009	1009	16.43	Eliminar

Showing 1 to 10 of 10 results

Previous

1

Next

Resultado:

Sensores de prueba

CO2

Temp

Hum

Ugler

Altura

Presion

Presion mm

Temp ext

Alisar

Type a keyword...

id	o	CO2	Temp	Hum	Fecha	Lugar	Altura	Presion	Presion mm	Temp ext	Eliminar
1		758.3502469049592	18.93368742431066	55.600635561496205	09-Jun-2024 (22:50:17.970087)	Casa de Juan	50	1009	1009	16.43	Eliminar
2		487.04874911576997	21.260089038140294	40.5991491326726644	09-Jun-2024 (22:50:22.036865)	Casa de Juan	50	1009	1009	16.43	Eliminar
3		1005.5079486255445	26.08965383954648	58.27977207085695	09-Jun-2024 (22:50:26.133501)	Casa de Juan	50	1009	1009	16.43	Eliminar
4		346.6206042868657	16.86415898040444	68.44206781852616	09-Jun-2024 (22:50:36.207001)	Casa de Juan	50	1009	1009	16.43	Eliminar
6		836.83548137609	22.870518034461146	76.6029168439301	09-Jun-2024 (22:50:38.367856)	Casa de Juan	50	1009	1009	16.43	Eliminar
7		1044.8742759659065	22.876038827914643	68.18416788010104	09-Jun-2024 (22:50:42.440022)	Casa de Juan	50	1009	1009	16.43	Eliminar
8		553.3255344754824	19.63097530816683	68.60379583911019	09-Jun-2024 (22:50:46.516717)	Casa de Juan	50	1009	1009	16.43	Eliminar
9		916.0641634501849	22.9494719328777	61.857078204383924	09-Jun-2024 (22:50:58.600274)	Casa de Juan	50	1009	1009	16.43	Eliminar
10		317.16878114578776	20.486118883616227	49.99579620835955	09-Jun-2024 (22:50:54.665936)	Casa de Juan	50	1009	1009	16.43	Eliminar

Showing 1 to 9 of 9 results

Previous

1

Next

Ejemplo del form para añadir entrada:

Sensores de prueba

CO2

Temp

Hum

Ugler

Altura

Presion

Presion mm

Temp ext

Alisar

Type a keyword...

id	o	CO2	Temp	Hum	Fecha	Lugar	Altura	Presion	Presion mm	Temp ext	Eliminar
1		758.3502469049592	18.93368742431066	55.600635561496205	09-Jun-2024 (22:50:17.970087)	Casa de Juan	50	1009	1009	16.43	Eliminar
2		487.04874911576997	21.260089038140294	40.5991491326726644	09-Jun-2024 (22:50:22.036865)	Casa de Juan	50	1009	1009	16.43	Eliminar
3		1005.5079486255445	26.08965383954648	58.27977207085695	09-Jun-2024 (22:50:26.133501)	Casa de Juan	50	1009	1009	16.43	Eliminar
4		346.6206042868657	16.86415898040444	68.44206781852616	09-Jun-2024 (22:50:36.207001)	Casa de Juan	50	1009	1009	16.43	Eliminar
6		836.83548137609	22.870518034461146	76.6029168439301	09-Jun-2024 (22:50:38.367856)	Casa de Juan	50	1009	1009	16.43	Eliminar
7		1044.8742759659065	22.876038827914643	68.18416788010104	09-Jun-2024 (22:50:42.440022)	Casa de Juan	50	1009	1009	16.43	Eliminar
8		553.3255344754824	19.63097530816683	68.60379583911019	09-Jun-2024 (22:50:46.516717)	Casa de Juan	50	1009	1009	16.43	Eliminar
9		916.0641634501849	22.9494719328777	61.857078204383924	09-Jun-2024 (22:50:58.600274)	Casa de Juan	50	1009	1009	16.43	Eliminar
10		317.16878114578776	20.486118883616227	49.99579620835955	09-Jun-2024 (22:50:54.665936)	Casa de Juan	50	1009	1009	16.43	Eliminar
11	200		22	31	09-June-2024 (22:55:52)	Plaza	10	1005	1000	30	Eliminar

Showing 1 to 10 of 10 result(s)

Previous

1

Next

Resultado:

Sensores de prueba

CO2

Temp

Hum

Ugler

Altura

Presion

Presion mm

Temp ext

Alisar

Type a keyword...

id	o	CO2	Temp	Hum	Fecha	Lugar	Altura	Presion	Presion mm	Temp ext	Eliminar
1		758.3502469049592	18.93368742431066	55.600635561496205	09-Jun-2024 (22:50:17.970087)	Casa de Juan	50	1009	1009	16.43	Eliminar
2		487.04874911576997	21.260089038140294	40.5991491326726644	09-Jun-2024 (22:50:22.036865)	Casa de Juan	50	1009	1009	16.43	Eliminar
3		1005.5079486255445	26.08965383954648	58.27977207085695	09-Jun-2024 (22:50:26.133501)	Casa de Juan	50	1009	1009	16.43	Eliminar
4		346.6206042868657	16.86415898040444	68.44206781852616	09-Jun-2024 (22:50:36.207001)	Casa de Juan	50	1009	1009	16.43	Eliminar
6		836.83548137609	22.870518034461146	76.6029168439301	09-Jun-2024 (22:50:38.367856)	Casa de Juan	50	1009	1009	16.43	Eliminar
7		1044.8742759659065	22.876038827914643	68.18416788010104	09-Jun-2024 (22:50:42.440022)	Casa de Juan	50	1009	1009	16.43	Eliminar
8		553.3255344754824	19.63097530816683	68.60379583911019	09-Jun-2024 (22:50:46.516717)	Casa de Juan	50	1009	1009	16.43	Eliminar
9		916.0641634501849	22.9494719328777	61.857078204383924	09-Jun-2024 (22:50:58.600274)	Casa de Juan	50	1009	1009	16.43	Eliminar
10		317.16878114578776	20.486118883616227	49.99579620835955	09-Jun-2024 (22:50:54.665936)	Casa de Juan	50	1009	1009	16.43	Eliminar
11	200		22	31	09-June-2024 (22:55:52)	Plaza	10	1005	1000	30	Eliminar

Showing 1 to 10 of 10 results

Previous

1

Next