ACTIVIDAD 5. Visualización de datos – Aplicaciones Web

Inicio

El objetivo de esta actividad es entender los conceptos básicos relacionados con desarrollo de aplicaciones web, como mecanismo para la visualización de los datos.

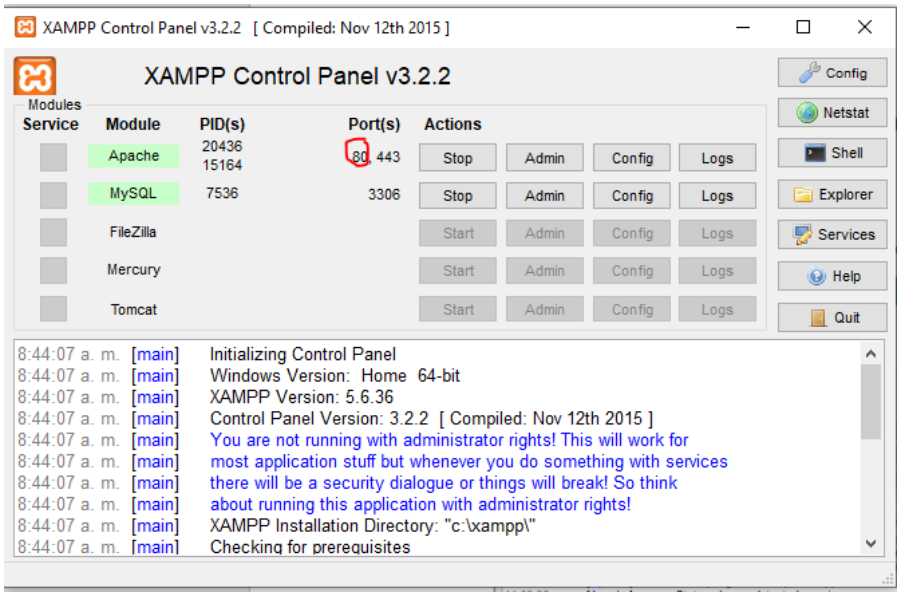
Trabajaremos con Node-RED donde recibimos, almacenamos los datos y se crearon los puntos de acceso usando HTTP. Estos puntos de acceso los usaremos en una aplicación web para acceder a los datos y mostrarlos al usuario.

Usaremos las herramientas de desarrollo web: HTML, CSS, JS y también Bootstrap para facilitar el diseño de la aplicación.

Instalación y configuración del ambiente de desarrollo

1. Instalación de Apache

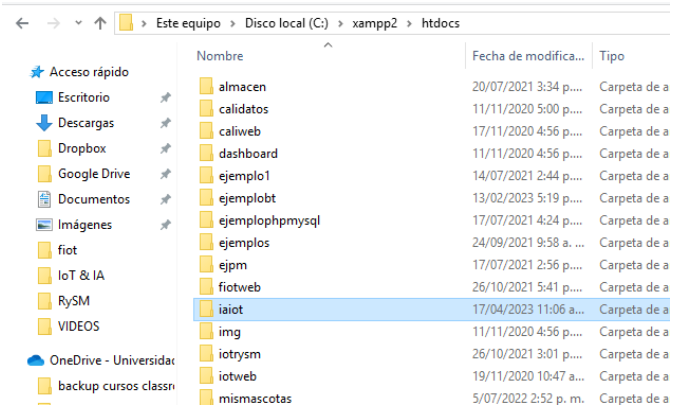
La instalación de Apache se puede hacer a través de XAMPP, el cual ya lo tenemos instalado.



Verificamos el puerto por el cual se va a ejecutar el servidor, por defecto es el 80.

2. Creación de la carpeta que contendrá la aplicación

Dentro de la carpeta **htdocs** que se encuentra en **Xampp**, creamos una carpeta a la cual le daremos un nombre relacionado con el proyecto, en mi caso la llamaré **iaiot**.



3. Creación de las páginas web

Dentro de la carpeta creada creamos los archivos requeridos para mostrar la información al usuario.

La creación y edición de los archivos la haremos usando Visual Code.

4. Página de prueba para entender el funcionamiento de Java Script

Creamos un archivo de prueba llamado prueba.html y escribimos el siguiente código:

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Document</title>
</head>
<body>
    <div id="datos">

    </div>

    <script>
        const url = 'http://localhost:1880/datos';

        fetch(url)
          .then(response => response.json())
          .then(result => console.log(result))
          .catch(error => console.error(error));
    </script>
</body>
</html>
```

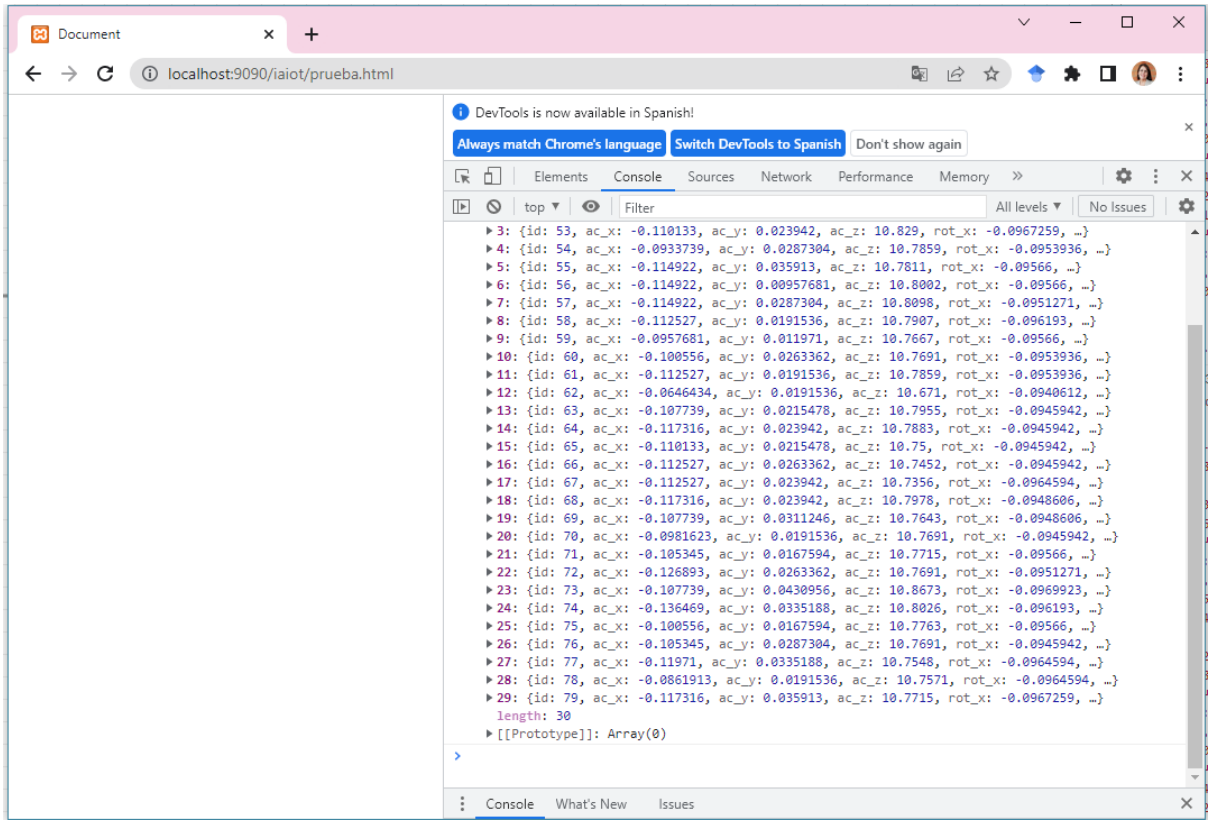
Este código no muestra nada en la página web pero muestra los datos en la consola:

Estamos usando una función fetch que permite hacer una consulta a una URL.

La respuesta de la función fetch es una promesa que se resuelve en un objeto Response que contiene la respuesta HTTP del servidor. Luego se llama al método then que constituye otra

promesa, en la cual se usa el método json del objeto anterior y se convierte en un objeto Java Script que después se muestra en la consola en el segundo then. Aquí ya se podría hacer algo con los datos recibidos. En este ejemplo simplemente lo mostramos en la consola.

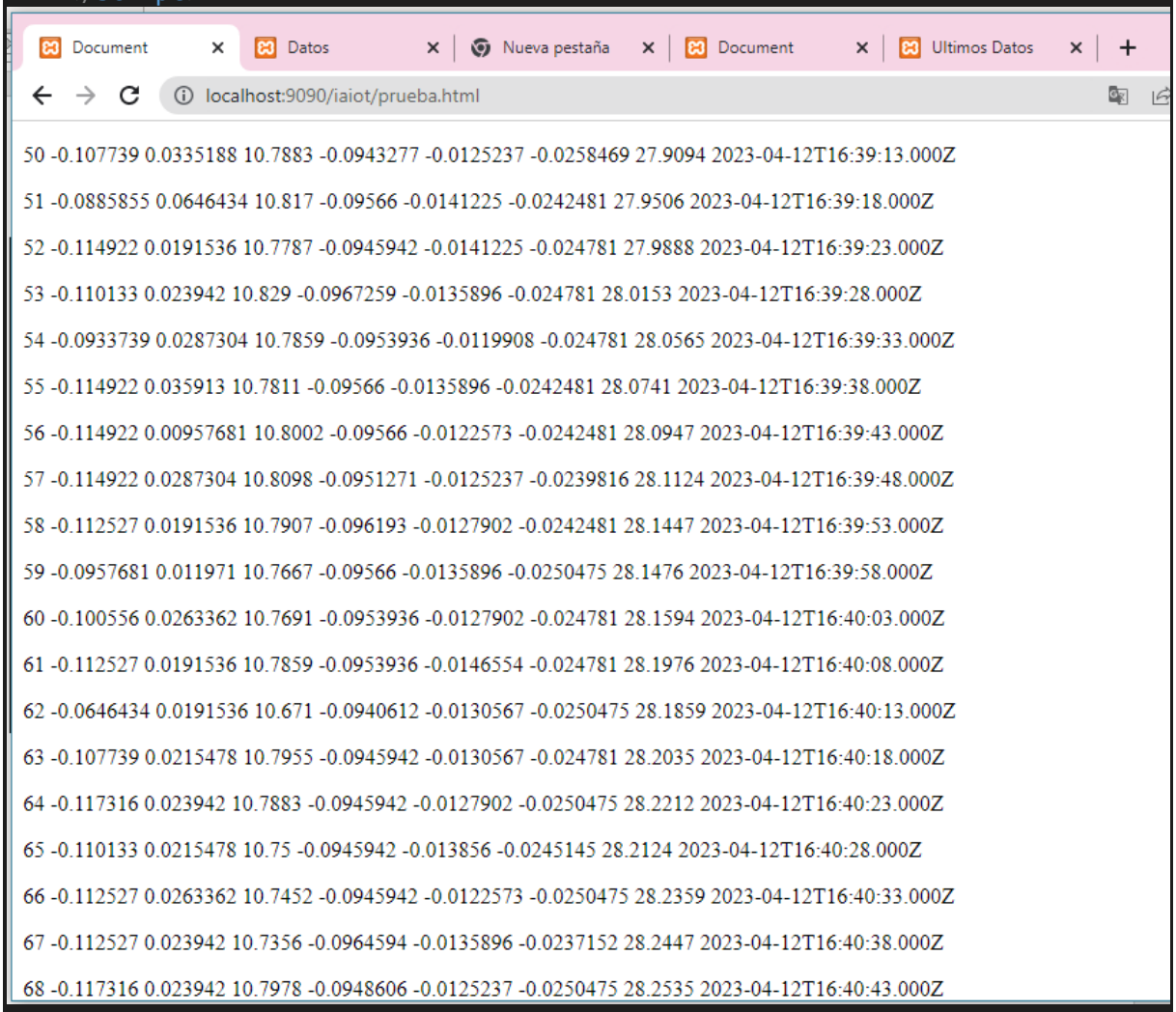
Si llamamos esta página desde el servidor podemos ver en la consola (f12) el resultado de la petición, que es lo que nos devuelve la api de Node-RED.



Podemos hacer lo siguiente para que nos muestre los datos en la página:

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Document</title>
</head>
<body>
    <div id="datos">

    </div>

    <script>
        const url = 'http://localhost:1880/datos';

        fetch(url)
          .then(response => response.json())
          .then(result => {
            const data = document.querySelector('#datos');
```

```
            result.forEach(obj => {
              const parrafo = document.createElement('p');
              parrafo.innerHTML = `${obj.id} ${obj.ac_x} ${obj.ac_y}
${obj.ac_z} ${obj.rot_x} ${obj.rot_y} ${obj.rot_z} ${obj.temperatura}
${obj.fecha}`;
              data.appendChild(parrafo);
            });
          })
          .catch(error => console.error(error));
    </script>
```




```
</body>
</html>
```

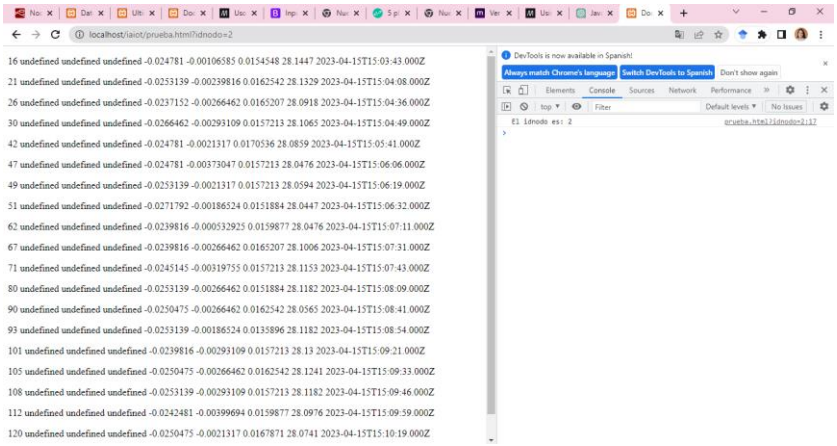
El resultado sería el siguiente:

Hagamos ahora una prueba para pasar un parámetro desde la URL a la página para poder hacer una selección.

```
<!DOCTYPE html>
```

```
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Document</title>
</head>
<body>
    <div id="datos">

    </div>

    <script>
        const urlSearchParams = new URLSearchParams(window.location.search);
        const id= urlSearchParams.get("idnodo");
        console.log("El idnodo es:", id);

        const url = 'http://localhost:1880/datos-id?idnodo='+ id;

        fetch(url)
          .then(response => response.json())
          .then(result => {
            const data = document.querySelector('#datos');
            result.forEach(obj => {
              const parrafo = document.createElement('p');
              parrafo.innerHTML = `${obj.id} ${obj.ac_x} ${obj.ac_y}
${obj.ac_z} ${obj.rot_x} ${obj.rot_y} ${obj.rot_z} ${obj.temperatura}
${obj.fecha}`;
              data.appendChild(parrafo);
            });
          })
          .catch(error => console.error(error));
    </script>
</body>
</html>
```

El resultado sería el siguiente:



Página para mostrar todos los datos en una tabla

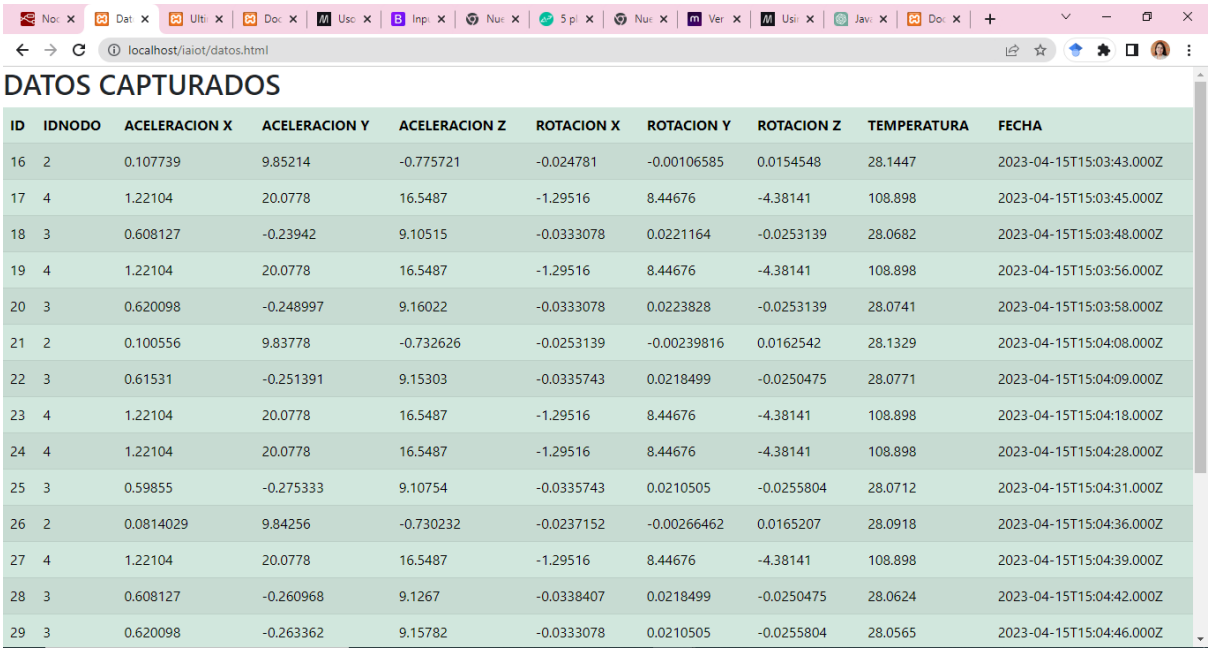
Aquí usaremos Bootstrap para crear la tabla y darle un estilo, y con JavaScript crearemos las filas de la tabla con los datos que nos entrega el servidor.

Creamos un archivo de nombre datos.html. El código es el siguiente:

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

<head>
    <!-- Required meta tags -->
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1">

    <!-- Bootstrap CSS -->
    <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0-
alpha1/dist/css/bootstrap.min.css" rel="stylesheet"
        integrity="sha384-
GLhlTQ8iRABdZLl6O3oVMWSktQOp6b7In1Zl3/Jr59b6EGGoI1aFkw7cmDA6j6gD"
crossorigin="anonymous">

    <title>Datos</title>
</head>

<body>
    <h2>DATOS CAPTURADOS</h2>
    <table class="table table-success table-striped" id="tabla-datos">
        <thead>
            <tr>
                <th scope="col">ID</th>
                <th scope="col">IDNODO</th>
                <th scope="col">ACELERACION X</th>
                <th scope="col">ACELERACION Y</th>
                <th scope="col">ACELERACION Z</th>
                <th scope="col">ROTACION X</th>
                <th scope="col">ROTACION Y</th>
                <th scope="col">ROTACION Z</th>
                <th scope="col">TEMPERATURA</th>
                <th scope="col">FECHA</th>
            </tr>
        </thead>
        <tbody>

        </tbody>
    </table>

    <script>
        const url = 'http://localhost:1880/datos';
```

```
        fetch(url)
            .then(response => response.json())
            .then(data => {
                const tbody = document.querySelector('#tabla-datos tbody');

                data.forEach(obj => {
                    const tr = document.createElement('tr');
                    tr.innerHTML =
`<td>${obj.id}</td><td>${obj.idnodo}</td><td>${obj.acc_x}</td><td>${obj.acc_y}
</td><td>${obj.acc_z}</td>
                                <td>${obj.rot_x}</td><td>${obj.rot_y}</td><td>${
obj.rot_z}</td>
                                <td>${obj.temperatura}</td><td>${obj.fecha}</td>
`;
                    tbody.appendChild(tr);
                });
            })
            .catch(error => console.error(error));
    </script>


    <!-- Option 1: Bootstrap Bundle with Popper -->
    <script src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0-
alpha1/dist/js/bootstrap.bundle.min.js"
        integrity="sha384-
w76AqPfDkMBDXo30jS1Sgez6pr3x5MlQ1ZAGC+nuZB+EYdgRZgiwxhTBTkF7CXvN"
        crossorigin="anonymous"></script>

</body>

</html>
```

El resultado es el siguiente:



localhost/iaiot/datos.html

DATOS CAPTURADOS

ID	IDNODO	ACELERACION X	ACELERACION Y	ACELERACION Z	ROTACION X	ROTACION Y	ROTACION Z	TEMPERATURA	FECHA
16	2	0.107739	9.85214	-0.775721	-0.024781	-0.00106585	0.0154548	28.1447	2023-04-15T15:03:43.000Z
17	4	1.22104	20.0778	16.5487	-1.29516	8.44676	-4.38141	108.898	2023-04-15T15:03:45.000Z
18	3	0.608127	-0.23942	9.10515	-0.0333078	0.0221164	-0.0253139	28.0682	2023-04-15T15:03:48.000Z
19	4	1.22104	20.0778	16.5487	-1.29516	8.44676	-4.38141	108.898	2023-04-15T15:03:56.000Z
20	3	0.620098	-0.248997	9.16022	-0.0333078	0.0223828	-0.0253139	28.0741	2023-04-15T15:03:58.000Z
21	2	0.100556	9.83778	-0.732626	-0.0253139	-0.00239816	0.0162542	28.1329	2023-04-15T15:04:08.000Z
22	3	0.61531	-0.251391	9.15303	-0.0335743	0.0218499	-0.0250475	28.0771	2023-04-15T15:04:09.000Z
23	4	1.22104	20.0778	16.5487	-1.29516	8.44676	-4.38141	108.898	2023-04-15T15:04:18.000Z
24	4	1.22104	20.0778	16.5487	-1.29516	8.44676	-4.38141	108.898	2023-04-15T15:04:28.000Z
25	3	0.59855	-0.275333	9.10754	-0.0335743	0.0210505	-0.0255804	28.0712	2023-04-15T15:04:31.000Z
26	2	0.0814029	9.84256	-0.730232	-0.0237152	-0.00266462	0.0165207	28.0918	2023-04-15T15:04:36.000Z
27	4	1.22104	20.0778	16.5487	-1.29516	8.44676	-4.38141	108.898	2023-04-15T15:04:39.000Z
28	3	0.608127	-0.260968	9.1267	-0.0338407	0.0218499	-0.0250475	28.0624	2023-04-15T15:04:42.000Z
29	3	0.620098	-0.263362	9.15782	-0.0333078	0.0210505	-0.0255804	28.0565	2023-04-15T15:04:46.000Z

Página para mostrar los últimos datos en cards

Usamos de nuevo Bootstrap para darle estilo a las cards.

Debemos crear una nueva operación get en Node-RED que me entregue los últimos datos capturados.

Creamos un nuevo archivo llamado datos-ultimos.html. El código es el siguiente:

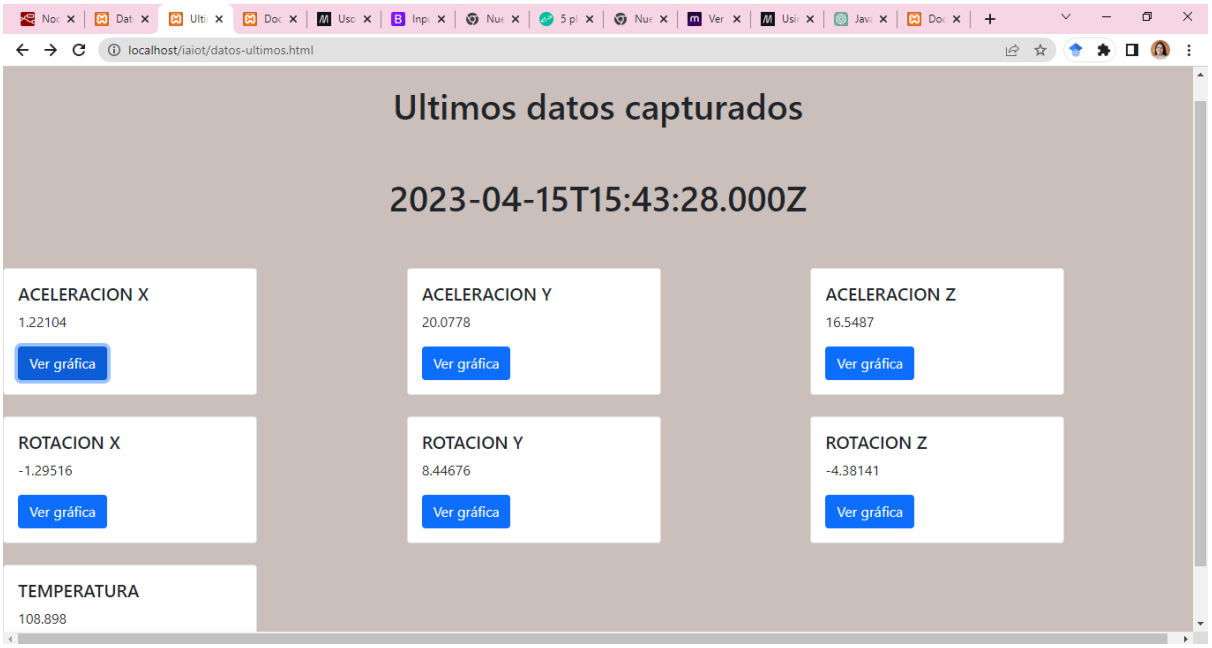
```
<!doctype html>
<html lang="en">

<head>
    <!-- Required meta tags -->
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1">

    <!-- Bootstrap CSS -->
    <link
href="https://cdn.jsdelivr.net/npm/bootstrap@5.0.2/dist/css/bootstrap.min.css"
rel="stylesheet"
        integrity="sha384-
EVSTQN3/azprG1Anm3QDgpJLIm9Nao0Yz1ztcQTwFspd3yD65VohhpuuCOmLASjC"
crossorigin="anonymous">

    <title>Ultimos Datos</title>
</head>

<body>
    <div id="datos" style="background-color: rgb(202, 191, 187);">

        <div class="row pt-5">
```

```
            <h3 class="text-center pb-5 h1">Ultimos datos capturados</h3>
        </div>

        <div id="card-container">

        </div>
    </div>

    <script>
        const url = 'http://localhost:1880/datos-ultimo/';

        fetch(url)
            .then(response => response.json())
            .then(data => {
                const cardContainer = document.querySelector('#card-
container');

                data.forEach(obj => {
                    const fecha = document.createElement('div');
                    fecha.setAttribute("class", "text-center pb-5 h1");
                    fecha.innerHTML = `${obj.fecha}`;
                    const card = document.createElement('div');
                    card.setAttribute("class", "row row-cols-1 row-cols-md-3
g-4");
                    card.innerHTML = `<div class="col">
                    <div class="card" style="width: 18rem;">
                        <div class="card-body">
                          <h5 class="card-title">ACELERACION X</h5>
                          <p class="card-text">${obj.acc_x}</p>
                          <a href="grafica.html?tipo=acx" class="btn btn-
primary">Ver gráfica</a>
                        </div>
                      </div>
                </div>
                <div class="col">
                    <div class="card" style="width: 18rem;">
                        <div class="card-body">
                          <h5 class="card-title">ACELERACION Y</h5>
                          <p class="card-text">${obj.acc_y}</p>
                          <a href="grafica.html?tipo=acy" class="btn btn-
primary">Ver gráfica</a>
                        </div>
                      </div>
                </div>
                <div class="col">
                    <div class="card" style="width: 18rem;">
                        <div class="card-body">
                          <h5 class="card-title">ACELERACION Z</h5>
                          <p class="card-text">${obj.acc_z}</p>
```

```
                        <a href="grafica.html?tipo=acz" class="btn btn-
primary">Ver gráfica</a>
                    </div>
                  </div>
            </div>
            <div class="col">
                <div class="card" style="width: 18rem;">
                    <div class="card-body">
                      <h5 class="card-title">ROTACION X</h5>
                      <p class="card-text">${obj.rot_x}</p>
                      <a href="grafica.html?tipo=rotx" class="btn btn-
primary">Ver gráfica</a>
                    </div>
                  </div>
            </div>
            <div class="col">
                <div class="card" style="width: 18rem;">
                    <div class="card-body">
                      <h5 class="card-title">ROTACION Y</h5>
                      <p class="card-text">${obj.rot_y}</p>
                      <a href="grafica.html?tipo=roty" class="btn btn-
primary">Ver gráfica</a>
                    </div>
                  </div>
            </div>
            <div class="col">
                <div class="card" style="width: 18rem;">
                    <div class="card-body">
                      <h5 class="card-title">ROTACION Z</h5>
                      <p class="card-text">${obj.rot_z}</p>
                      <a href="grafica.html?tipo=rotz" class="btn btn-
primary">Ver gráfica</a>
                    </div>
                  </div>
            </div>
            <div class="col">
                <div class="card" style="width: 18rem;">
                    <div class="card-body">
                      <h5 class="card-title">TEMPERATURA</h5>
                      <p class="card-text">${obj.temperatura}</p>
                      <a href="grafica.html?tipo=temp" class="btn btn-
primary">Ver gráfica</a>
                    </div>
                  </div>
            </div>`;
                cardContainer.appendChild(fecha);
                cardContainer.appendChild(card);
            });
        })
```

```
            .catch(error => console.error(error));
    </script>


    <!-- Option 1: Bootstrap Bundle with Popper -->
    <script
src="https://cdn.jsdelivr.net/npm/bootstrap@5.0.2/dist/js/bootstrap.bundle.min
.js"
        integrity="sha384-
MrcW6ZMFYlzcLA8Nl+NtUVF0sA7MsXsP1UyJoMp4YLEuNSfAP+JcXn/tWtIaxVXM"
        crossorigin="anonymous"></script>

</body>

</html>
```

El resultado es el siguiente:



Página para mostrar las gráficas de los datos.

En este caso usaremos una utilidad de Java Script para hacer gráficas denominada chart. Podemos obtener la información de uso del siguiente link: https://www.chartjs.org/

Creamos un archivo grafica.html al que se le pasa como parámetro la gráfica que se quiere ver en una variable llamada tipo, el parámetro se pasa por la URL.

El código es el siguiente:

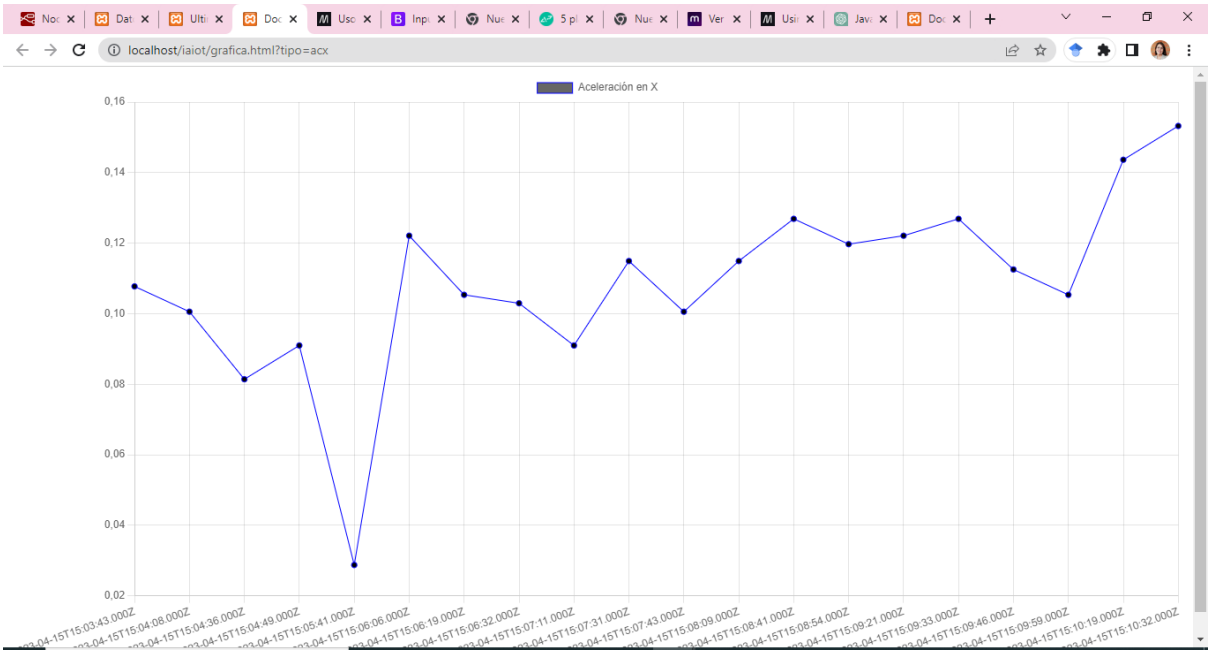
```
<!DOCTYPE html>
<html lang="en">

<head>
```

```
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Document</title>
    <script src="https://cdn.jsdelivr.net/npm/chart.js"></script>
</head>

<body>
    <canvas id="grafica"></canvas>

    <script>
        const urlSearchParams = new URLSearchParams(window.location.search);
        const tipo = urlSearchParams.get("tipo");
        console.log("El tipo es:", tipo);
        switch (tipo) {
            case "acx":
                titulo = "Aceleración en X";
                break;
            case "acy":
                titulo = "Aceleración en Y";
                break;
            case "acz":
                titulo = "Aceleración en Z";
                break;
            case "rotx":
                titulo = "Rotación en X";
                break;
            case "roty":
                titulo = "Rotación en Y";
                break;
            case "rotz":
                titulo = "Rotación en Z";
                break;
            case "temp":
                titulo = "Temperatura";
                break;
        }
        const url = 'http://localhost:1880/datos-id?idnodo=2';
        fetch(url)
            .then(response => response.json())
            .then(data => {
                const $grafica = document.querySelector("#grafica");

                const etiquetas = [];

                const datos = {
                    label: titulo,
                    data: [],
                    backgroundColor: "",
```

```
                borderColor: "blue",
                borderWidth: 1
            };

            for (let i = 0; i < data.length; i++) {
                etiquetas.push(data[i].fecha);
                switch (tipo) {
                    case "acx":
                        datos.data.push(data[i].acc_x);
                        break;
                    case "acy":
                        datos.data.push(data[i].acc_y);
                        break;
                    case "acz":
                        datos.data.push(data[i].acc_z);
                        break;
                    case "rotx":
                        datos.data.push(data[i].rot_x);
                        break;
                    case "roty":
                        datos.data.push(data[i].rot_y);
                        break;
                    case "rotz":
                        datos.data.push(data[i].rot_z);
                        break;
                    case "temp":
                        datos.data.push(data[i].temperatura);
                        break;
                }
            }

            new Chart($grafica, {
                type: 'line',// Tipo de gráfica
                data: {
                    labels: etiquetas,
                    datasets: [
                        datos
                    ]
                },
                options: {
                    scales: {
                        yAxes: [{
                            ticks: {
                                beginAtZero: true
                            }
                        }],
                    },
                }
            });
```

```
            })
            .catch(error => console.error(error));
    </script>
</body>

</html>
```

El resultado es el siguiente:

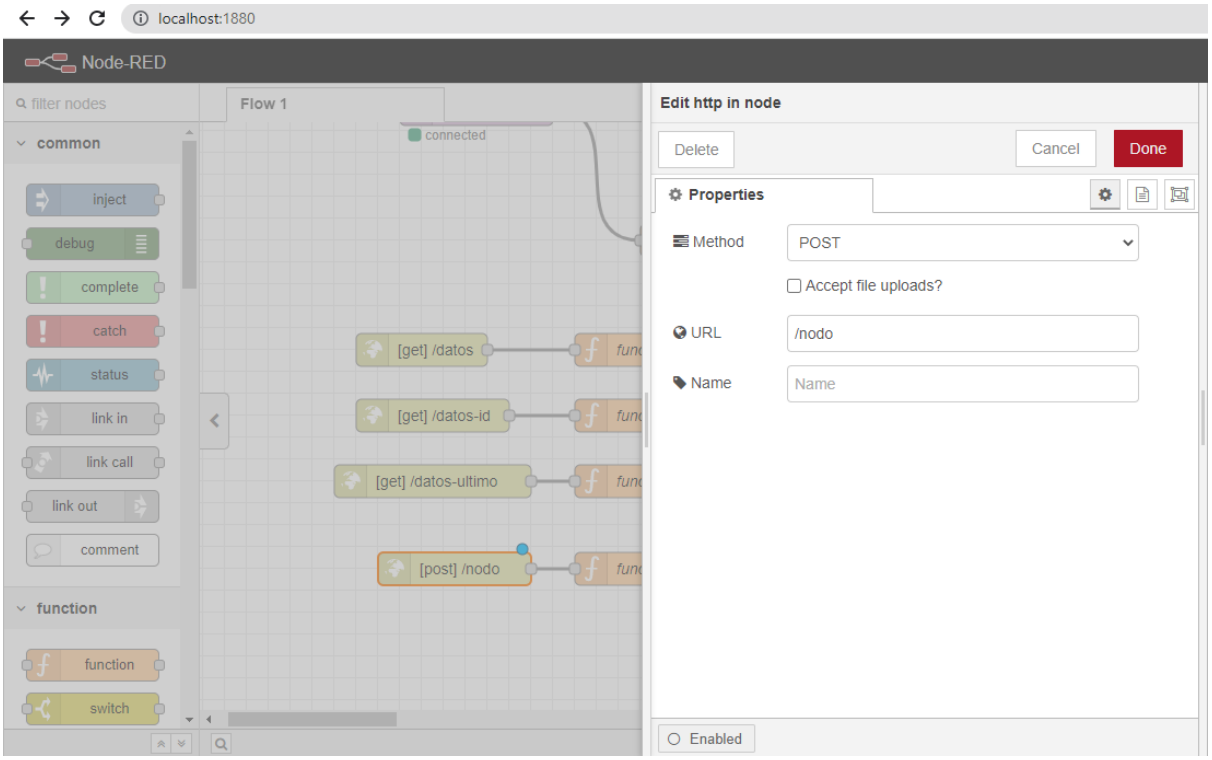


Operaciones POST

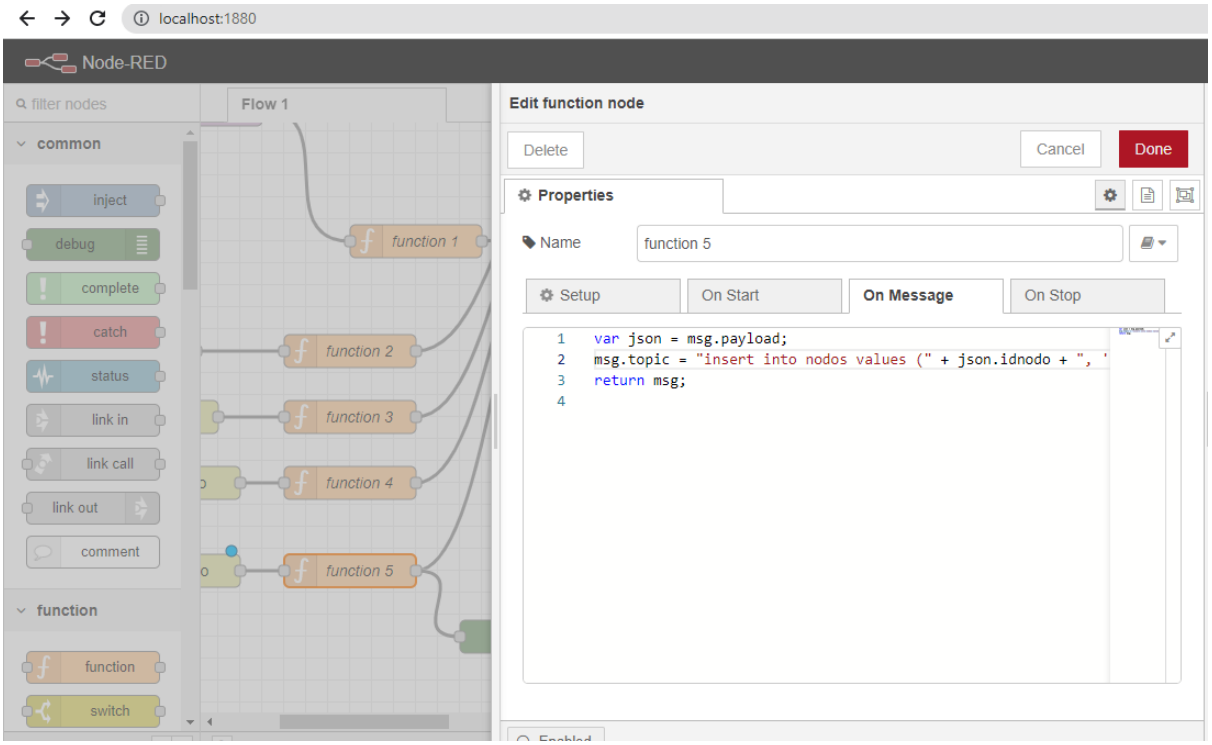
Vamos a crear primero una función en Node-RED para realizar una operación POST. En este ejemplo lo realizaremos para la creación de nodos. Previamente se debe haber creado la tabla en MySQL.

```
MariaDB [iaiot]> desc nodos;
+-------------+--------------+------+-----+---------+-------+
| Field       | Type         | Null | Key | Default | Extra |
+-------------+--------------+------+-----+---------+-------+
| idnodo      | int(11)      | NO   | PRI | NULL    |       |
| descripcion | varchar(100) | YES  |     | NULL    |       |
| ubicacion   | varchar(30)  | YES  |     | NULL    |       |
+-------------+--------------+------+-----+---------+-------+
3 rows in set (0.01 sec)
```

En Node-RED añadimos un nodo http-in y lo configuramos con la operación y la ruta:



Agregamos un nodo función, la conectamos con el nodo de la base de datos, y allí configuramos el query para agregar el nodo a la tabla:



El código de la función es el siguiente:

```
var json = msg.payload;
msg.topic = "insert into nodos values (" + json.idnodo + ", '" +
json.descripcion + "', '" + json.ubicacion + "')";
return msg;
```

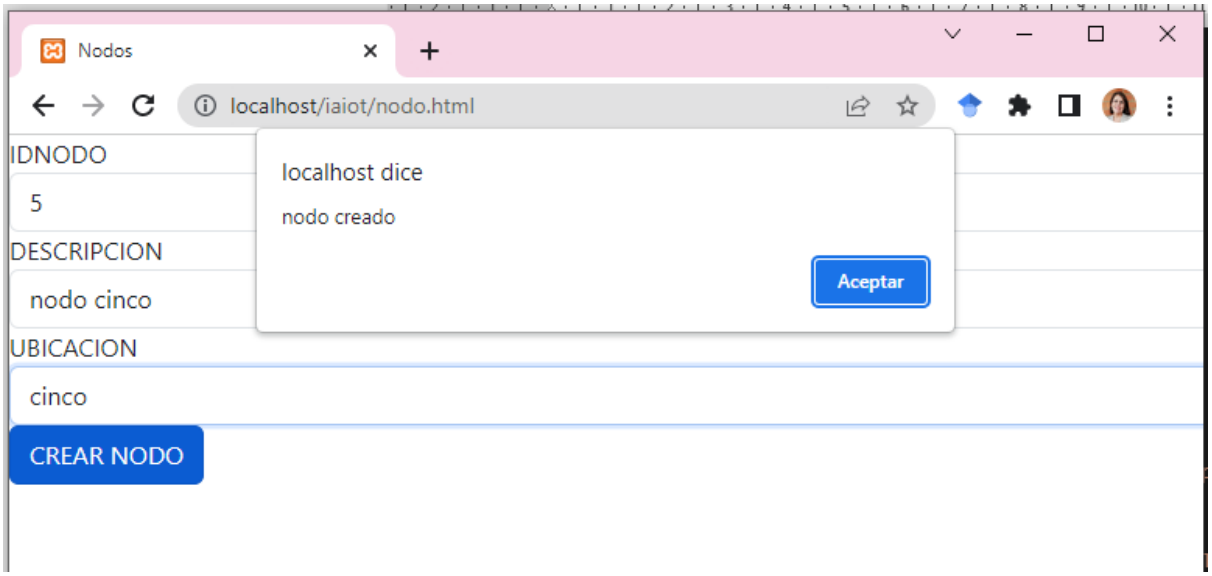
Desarrollamos una página web, nodo.html, en donde ponemos un formulario y a través de un script en donde ejecutamos un fecth llamamos a la función POST de Node-RED.

El código de la página es el siguiente:

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

<head>
    <!-- Required meta tags -->
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1">

    <!-- Bootstrap CSS -->
    <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0-
alpha1/dist/css/bootstrap.min.css" rel="stylesheet"
        integrity="sha384-
GLhlTQ8iRABdZLl6O3oVMWSktQOp6b7In1Zl3/Jr59b6EGGoI1aFkw7cmDA6j6gD"
crossorigin="anonymous">

    <title>Nodos</title>
</head>

<body>
    <form id="formulario">

        <div class="form-group">
            <label for="idnodo" class="control-label">IDNODO</label>
            <input type="text" class="form-control" id="idnodo" name="idnodo"
>
        </div>

        <div class="form-group">
            <label for="descripcion" class="control-label">DESCRIPCION
</label>
            <input type="text" class="form-control" id="descripcion"
name="descripcion">
        </div>

        <div class="form-group">
            <label for="ubicacion" class="control-label">UBICACION</label>
            <input type="text" class="form-control" id="ubicacion"
name="ubicacion" >
        </div>

        <div class="form-group"> <!-- Submit Button -->
            <button type="submit"  class="btn btn-primary">CREAR NODO</button>
        </div>

    </form>
```

```
    <script>
        const formulario = document.querySelector('#formulario');

        formulario.addEventListener('submit', (event) => {
            event.preventDefault();

            const url = 'http://localhost:1880/nodo';
            const data = {
                idnodo: formulario.idnodo.value,
                descripcion: formulario.descripcion.value,
                ubicacion: formulario.ubicacion.value
            };

            const options = {
                method: 'POST',
                headers: {
                    'Content-Type': 'application/json'
                },
                body: JSON.stringify(data)
            };

            fetch(url, options)
                .then(response => response.json())
                .then(data => console.log(data))
                .catch(error => console.error(error));
            alert("nodo creado", "");
            formulario.reset();
        });

    </script>

    <!-- Option 1: Bootstrap Bundle with Popper -->
    <script src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0-
alpha1/dist/js/bootstrap.bundle.min.js"
        integrity="sha384-
w76AqPfDkMBDXo30jS1Sgez6pr3x5MlQ1ZAGC+nuZB+EYdgRZgiwxhTBTkF7CXvN"
        crossorigin="anonymous"></script>

</body>

</html>
```

El resultado es el siguiente:



Cuando se le da aceptar queda en la misma página y se borran los campos del formulario.

Podemos ver que en la base de datos se ha creado el nodo:

```
MariaDB [iaiot]> select * from nodos;
+--------+-------------+-----------+
| idnodo | descripcion | ubicacion |
+--------+-------------+-----------+
|      1 | nodo uno    | uno       |
|      2 | nodo dos    | dos       |
|      3 | nodo tres   | tres      |
|      5 | nodo cinco  | cinco     |
+--------+-------------+-----------+
4 rows in set (0.00 sec)
```

Aplicación web completa.

Desarrollar una aplicación web completa que cumpla con los siguientes requerimientos:

1. Se manejarán dos tipos de usuario: un usuario cliente y un usuario administrador
2. Todos los usuarios deberán ingresar con Loguin y password
3. El usuario cliente deberá tener asociado al menos un nodo y cuando ingrese se le dará la opción de seleccionar uno de sus nodos y se le mostrarán los últimos datos del nodo seleccionado.
4. Cuando ingrese el usuario administrador se le mostrará un menú en donde podrá seleccionar ver todos los datos, ver datos de un nodo, ver los últimos datos de un nodo, ver gráficas de las variables asociadas a un nodo.
5. Cuando seleccione alguna de las opciones del menú se le mostrará la información asociada a la selección.

Realice todos los cambios necesarios en la base de datos y en Node-RED para poder cumplir con las peticiones de manera adecuada.