TALLER 7a. Servidor IoT-MySQL

Inicio

El objetivo de este taller es entender los conceptos relacionados con el almacenamiento de los datos en bases de datos relacionales y su integración con los servidores IoT.

Se trabajará sobre el desarrollo realizado en los talleres anteriores

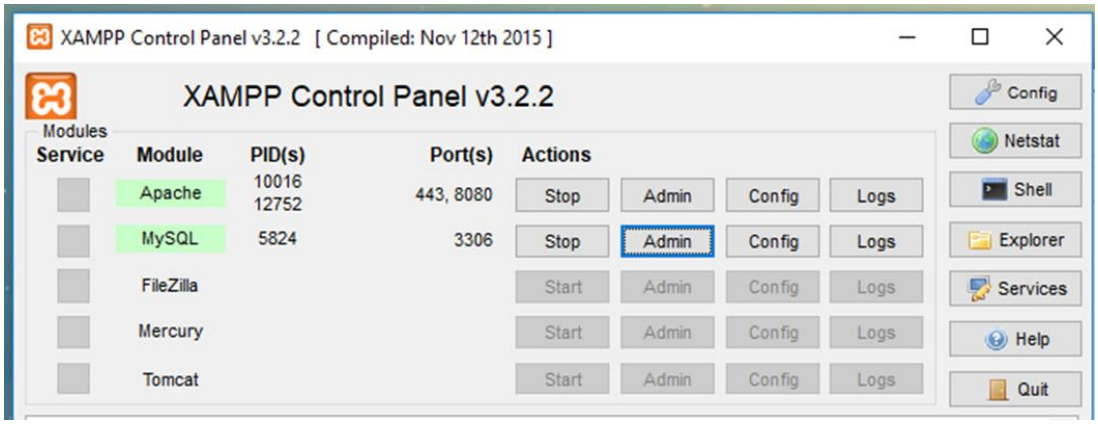
El desarrollo del taller se realizará en 2 fases:

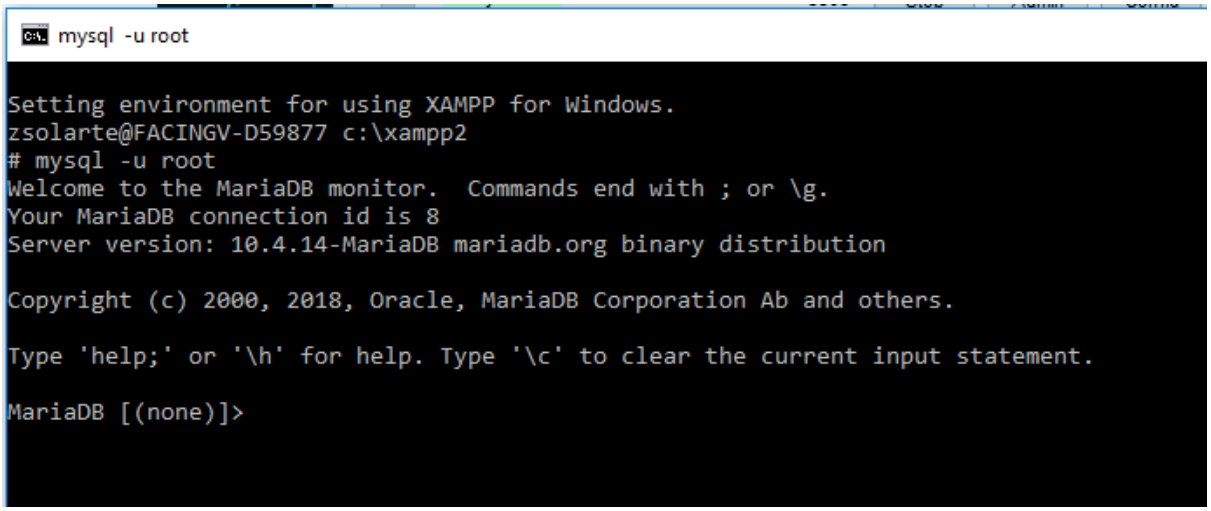
1. Configuración y desarrollo del código para almacenar y recuperar datos almacenados en MySQL a través de un servidor IoT con soporte a MQTT
2. Configuración y desarrollo del código para almacenar y recuperar datos almacenados en MySQL a través de un servidor IoT con soporte a REST

Instalación del ambiente de trabajo

1. Instalación de MySQL

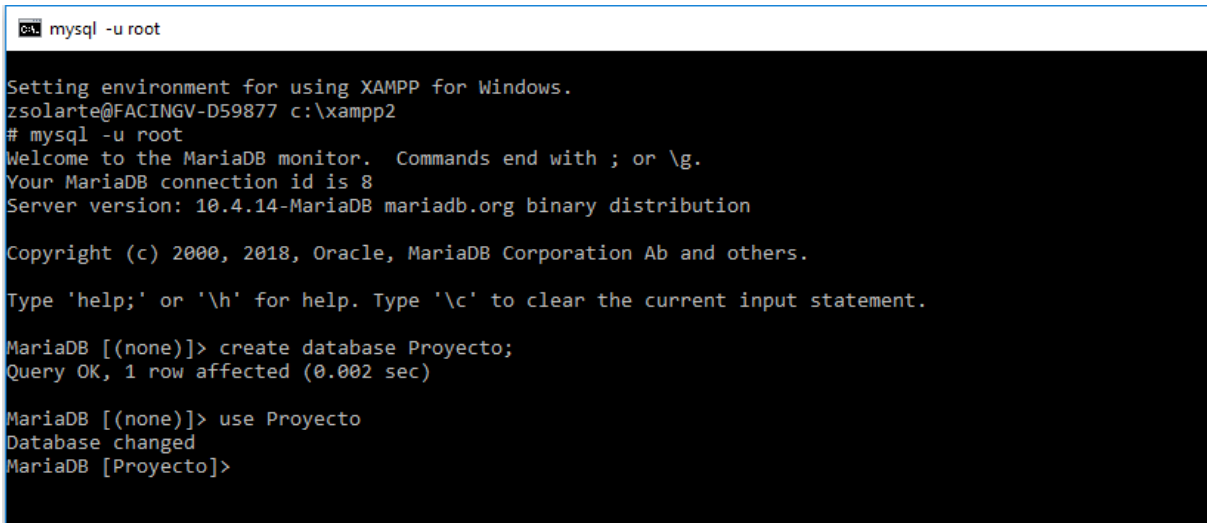
La instalación de MySQL se puede hacer a través de XAMPP y Se puede acceder a él a través de la línea de comandos, después de que se haya iniciado el servicio.





2. Creación de la base de datos y de las tablas.

Ya dentro de MySQL creamos la base de datos.



Debemos crear las tablas, para ello definimos la estructura de la tabla (o tablas) que contendrán los datos, esto incluye definir las columnas, su nombre, el tipo de dato que contendrá y algunas características adicionales.

Cada tabla debería contener una columna cuya información no se repita en los registros (Primary Key). Para el ejemplo de este taller la tabla se llamaría datosNodo y su estructura sería:

- Id – entero – auto incremental (cada vez que se reciba un dato se incrementa automáticamente este id) – primary key
- Idnodo – entero
- Temperatura – double
- Humedad – double
- tiempo – entero grande

```
MariaDB [Proyecto]> create table datosNodo (
    -> id int auto_increment,
    -> idnodo int,
    -> temperatura double,
    -> humedad double,
    -> tiempo bigint,
    -> primary key(id));
Query OK, 0 rows affected (0.015 sec)
```

```
MariaDB [Proyecto]> desc datosNodo;
+-------------+------------+------+-----+---------+----------------+
| Field       | Type       | Null | Key | Default | Extra          |
+-------------+------------+------+-----+---------+----------------+
| id          | int(11)    | NO   | PRI | NULL    | auto_increment |
| idnodo      | int(11)    | YES  |     | NULL    |                |
| temperatura | double     | YES  |     | NULL    |                |
| humedad     | double     | YES  |     | NULL    |                |
| tiempo      | bigint(20) | YES  |     | NULL    |                |
+-------------+------------+------+-----+---------+----------------+
5 rows in set (0.008 sec)
```

Servidor IoT con soporte a MQTT y MySQL

3. Instalación de las librerías necesarias

Para que el servidor IoT pueda acceder a la base de datos, se debe añadir la librería para nodejs de mysql a nuestro proyecto.

```
PS D:\Users\zsolarte\Desktop\ServidorIoT> npm i mysql
npm WARN ServidorIoT@1.0.0 No description
npm WARN ServidorIoT@1.0.0 No repository field.

+ mysql@2.18.1
added 8 packages from 14 contributors and audited 120 packages in 5.827s

11 packages are looking for funding
  run `npm fund` for details

found 0 vulnerabilities
```

4. Código para acceder a la base de datos

En el archivo indexmqtt.js agregamos el código que permite acceder a la base de datos para almacenar los datos recibidos mediante la suscripción al tópico manejado. En este caso vamos a suponer que los objetos están enviando datos al tópico1 usando el formato json adecuado, que en este caso debería tener la siguiente estructura:

```
{
        "idnodo": 1,
        "temperatura": 24.5,
        "humedad": 65,
        "timestamp": 1657825613
}
```

El archivo indexmqtt.js quedaría de la siguiente forma:

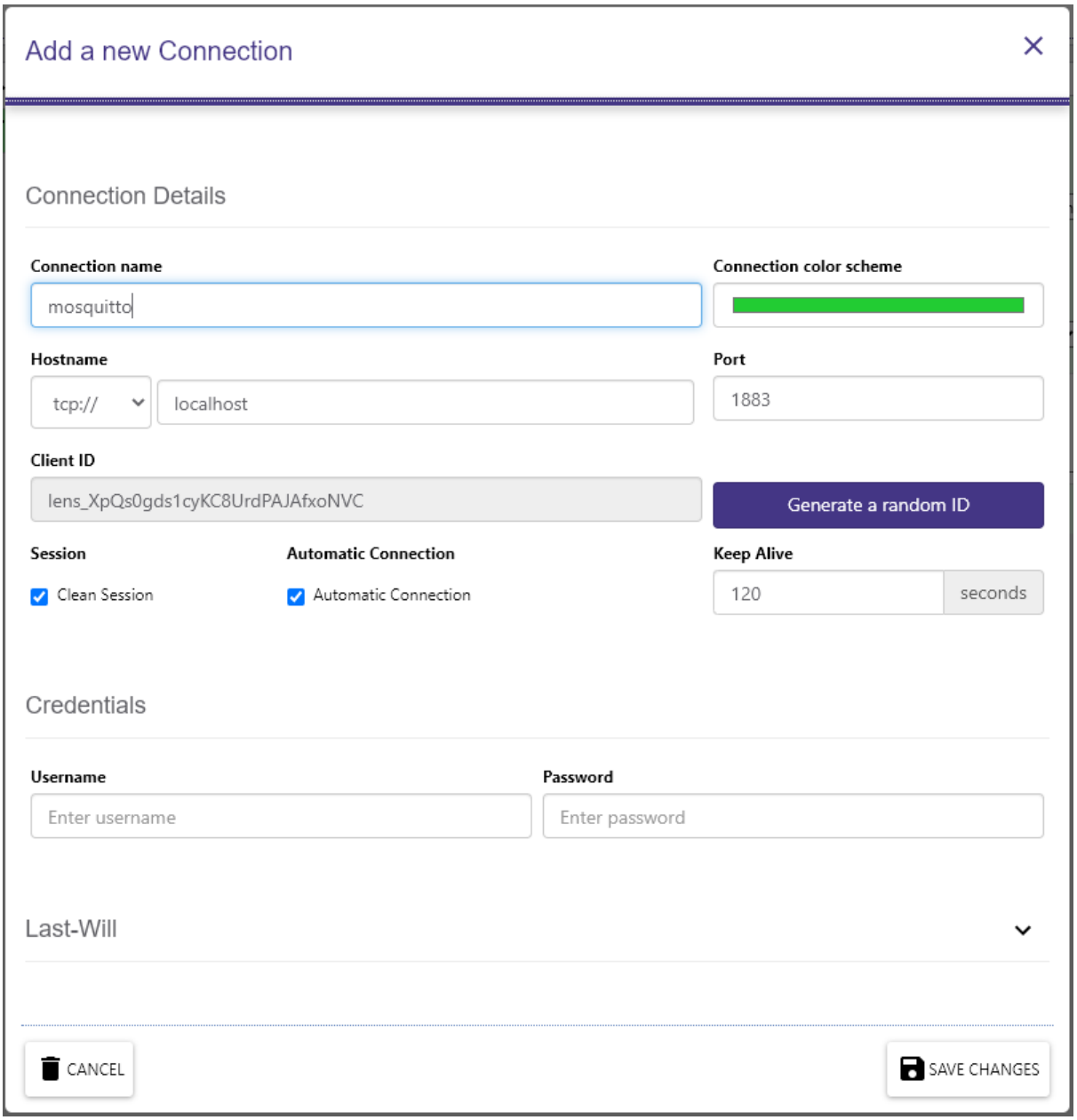
```
var mqtt = require('mqtt')
var client  = mqtt.connect('mqtt://localhost')
const mysql = require('mysql');

// se crea la conexión a mysql
const connection = mysql.createPool({
  connectionLimit:500,
     host: 'localhost',
     user: 'root',
     password: '', //el password de ingreso a mysql
     database: 'Proyecto',
     port: 3306});

client.on('connect', function () {
  client.subscribe('topico1', function (err) {
    if (err) {
      console.log("error en la subscripcion")
    }
  })
})

client.on('message', function (topic, message) {
  // message is Buffer
  json1 = JSON.parse(message.toString());
  console.log(json1);
  //client.publish('topico2', 'mensaje recibido')
  connection.getConnection(function(error, tempConn){ //conexion a mysql
    if(error){
      console.log('Problemas en la conexion'); //en caso de error en la
conexion
    }
    else{
    console.log('Conexion correcta.');
    tempConn.query('INSERT INTO datosNodo VALUES(null, ?, ?,?,?)',
          [json1.idnodo, json1.temperatura, json1.humedad, json1.timestamp],
          function(error, result){ //se ejecuta lainserción
       if(error){
            console.log('error al ejecutar el query');
         }else{
           tempConn.release();
           console.log("datos almacenados"); //mensaje de respuesta en consola
         }

  //client.end()  //si se habilita esta opción el servicio termina
});
}});
})
```

5. Prueba del servicio.

Realizamos la prueba del servicio, simulando el envío de datos a través de MqttLens. Primero lanzamos el servidor:

```
PS D:\Users\zsolarte\Desktop\ServidorIoT> node src/indexmqtt.js
```

Creamos la conexión en MqttLens



Publicamos en el topico topico1.

Cuando se hace la publicación aparece la siguiente información en el servidor:

```
PS D:\Users\zsolarte\Desktop\ServidorIoT> node src/indexmqtt.js
{ idnodo: 1, temperatura: 24.5, humedad: 65, timestamp: 1657825613 }
Conexion correcta.
datos almacenados
```

Si verificamos en la base de datos vemos que los datos se almacenaron:

```
MariaDB [Proyecto]> select * from datosNodo;
+----+--------+-------------+---------+------------+
| id | idnodo | temperatura | humedad | tiempo     |
+----+--------+-------------+---------+------------+
|  1 |      1 |        24.5 |      65 | 1657825613 |
+----+--------+-------------+---------+------------+
1 row in set (0.001 sec)
```

Servidor IoT con soporte a REST y MySQL

6. Código para acceder a la base de datos.

En el directorio rutas creamos un nuevo archivo que llamaremos **datos.js** que contendrá el código que permite acceder a la base de datos para almacenar los datos recibidos mediante la operación POST y para recuperar los datos cuando se acceda a la operación GET. En el archivo indexrest.js se hace el llamado a este archivo.

Los datos se envian y se entregan en formato JSON.

Archivo datos.js

```
const { Router } = require('express');
const router = Router();
const mysql = require('mysql');
```

```
// se crea la conexión a mysql
const connection = mysql.createPool({
    connectionLimit: 500,
    host: 'localhost',
    user: 'root',
    password: '', //el password de ingreso a mysql
    database: 'Proyecto',
    port: 3306
});

//function get en la ruta /datos, que trae todos los datos almacenados en la
tabla

router.get('/datos', (req, res) => {
    var json1 = {}; //variable para almacenar cada registro que se lea, en
formato json
    var arreglo = []; //variable para almacenar todos los datos, en formato
arreglo de json

    connection.getConnection(function (error, tempConn) { //conexion a mysql
        if (error) {
            throw error;  //si no se pudo conectar
        }
        else {
            console.log('Conexion correcta.');
            //ejecución de la consulta
            tempConn.query('SELECT * FROM datosNodo', function (error, result)
{
                var resultado = result; //se almacena el resultado de la
consulta en la variable resultado
                if (error) {
                    throw error;
                    res.send("error en la ejecución del query");
                } else {
                    tempConn.release(); //se librea la conexión
                    for (i = 0; i < resultado.length; i++) {        //se lee
el resultado y se arma el json
                        json1 = { "idnodo": resultado[i].idnodo,
"temperatura": resultado[i].temperatura, "humedad": resultado[i].humedad,
"fecha": resultado[i].tiempo };
                        console.log(json1); //se muestra el json en la consola
                        arreglo.push(json1); //se añade el json al arreglo
                    }
                    res.json(arreglo); //se retorna el arreglo
                }
            });
        }
    });
});
```

```
//función post en la ruta /datos que recibe datos
router.post('/datos', (req, res) => {
    console.log(req.body); //mustra en consola el json que llego
    json1 = req.body; //se almacena el json recibido en la variable json1
    connection.getConnection(function (error, tempConn) { //conexion a mysql
        if (error) {
            throw error; //en caso de error en la conexion
        }
        else {
            console.log('Conexion correcta.');
            tempConn.query('INSERT INTO datosNodo VALUES(null, ?, ?,?,?)',
[json1.idnodo, json1.temperatura, json1.humedad, json1.timestamp], function
(error, result) { //se ejecuta la inserción
                if (error) {
                    res.send("error al ejecutar el query");
                } else {
                    tempConn.release();
                    res.send("datos almacenados"); //mensaje de respuesta
                }
            });
        }
    });
});

module.exports = router;
```

Archivo indexrest.js

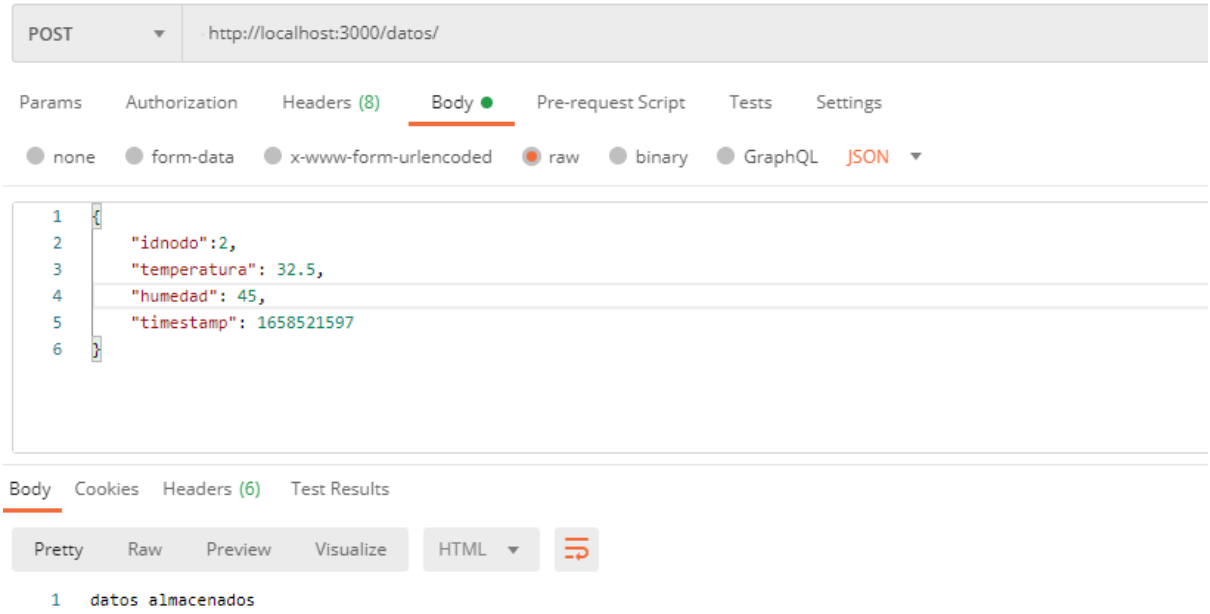
```
const express = require('express'); //se indica que se requiere express
const app = express(); // se inicia express y se instancia en una constante de
nombre app.
const morgan = require('morgan'); //se indica que se requiere morgan

// settings
app.set('port', 3000); //se define el puerto en el cual va a funcionar el
servidro

// Utilities
app.use(morgan('dev')); //se indica que se va a usar morgan en modo dev
app.use(express.json()); //se indica que se va a usar la funcionalidad para
manejo de json                                  de express

//Routes
app.use(require('./rutas/ejemplo.js'));
app.use(require('./rutas/datos.js'));

//Start server
```

```
app.listen(app.get('port'), ()=> {
    console.log("Servidor funcionando");
}); //se inicia el servidor en el puerto definido y se pone un mensaje en la consola.
```

Lanzamos el servicio y lo probamos desde postman.

```
PS D:\Users\zsolarte\Desktop\ServidorIoT> node src/indexrest.js
Servidor funcionando
```

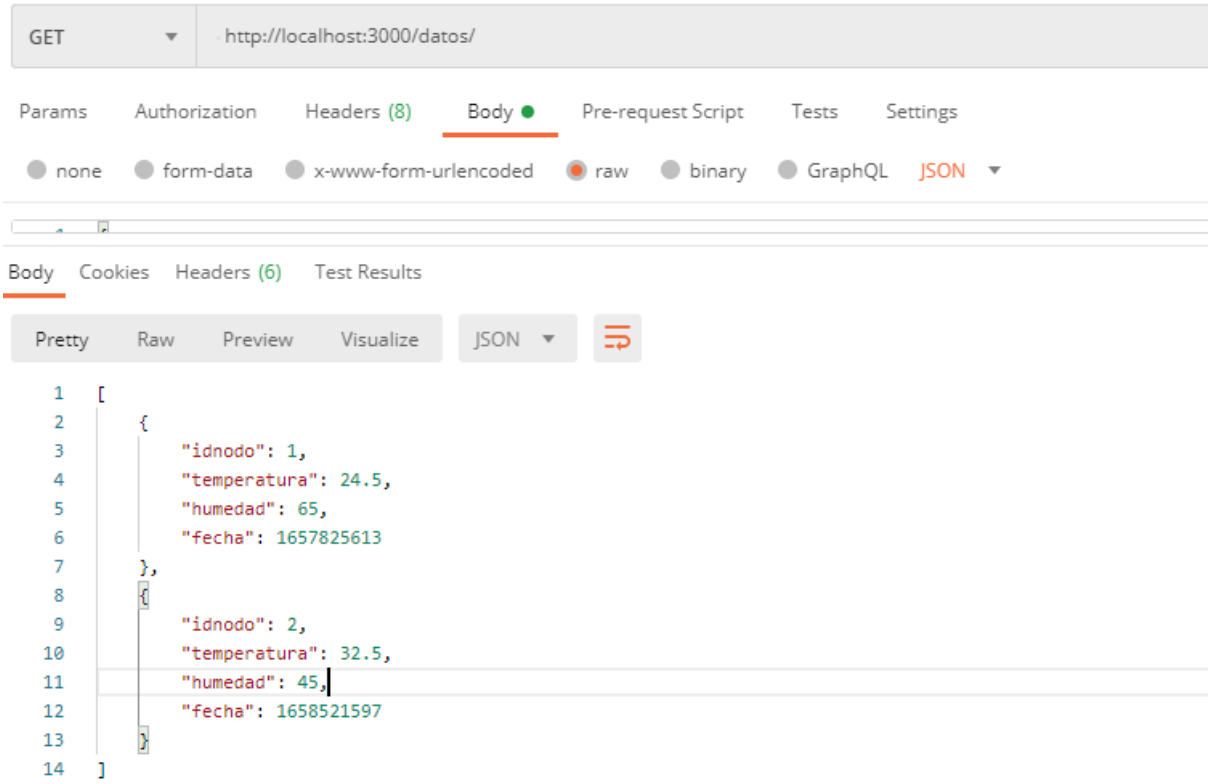
Probamos primero el post para almacenar datos en la base de datos. Recuerde añadir la ruta en la URL.



Verificamos en la base de datos:

```
MariaDB [Proyecto]> select * from datosNodo;
+----+--------+-------------+---------+------------+
| id | idnodo | temperatura | humedad | tiempo     |
+----+--------+-------------+---------+------------+
|  1 |      1 |        24.5 |      65 | 1657825613 |
|  2 |      2 |        32.5 |      45 | 1658521597 |
+----+--------+-------------+---------+------------+
2 rows in set (0.001 sec)
```

Ahora probamos el get

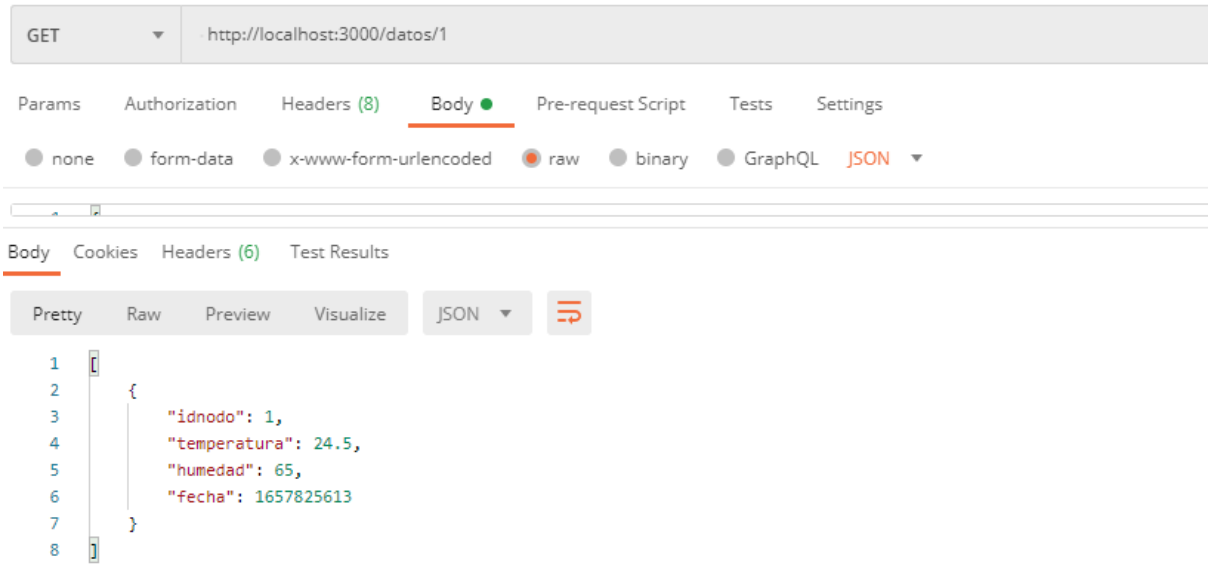


Ahora añadamos el código que permita traer solo los datos de un nodo. Para ello debemos añadir el siguiente código en el archivo datos.js. la idea es añadir otra función get que haga el procedimiento que se requiere.

```
router.get('/datos/:idnodo', (req, res) => {
    var json1 = {}; //variable para almacenar cada registro que se lea, en
formato json
    var arreglo = []; //variable para almacenar todos los datos, en formato
arreglo de json
    var id = req.params.idnodo; //recogemos el parámetro enviado en la url

    connection.getConnection(function (error, tempConn) { //conexion a mysql
        if (error) {
            throw error;  //si no se pudo conectar
        } else {
            console.log('Conexion correcta.');
            //ejecución de la consulta
            tempConn.query('SELECT * FROM datosNodo where idnodo=?', [id],
function (error, result) {
                var resultado = result; //se almacena el resultado de la
consulta en la variable resultado
                if (error) {
                    throw error;
                    //res.send("error en la ejecución del query");
                } else {
                    tempConn.release(); //se libera la conexión
```

```
                    for (i = 0; i < resultado.length; i++) {        //se lee
el resultado y se arma el json
                        json1 = { "idnodo": resultado[i].idnodo,
"temperatura": resultado[i].temperatura, "humedad": resultado[i].humedad,
"fecha": resultado[i].tiempo };
                        console.log(json1); //se muestra el json en la consola
                        arreglo.push(json1); //se añade el json al arreglo
                    }
                    res.json(arreglo); //se retorna el arreglo
                }
            }
            );
        }
    });
});
```

Levantamos de nuevo el servidor y probamos el get, pasando el parámetro del nodo-sensor del que se quieren obtener los datos.



Integración con el objeto IoT adaptado

Realizar la adaptación y configuración de los códigos necesarios para que el objeto manejado en los talleres anteriores envié los datos al servidor desarrollado y estos sean almacenados en la base de datos MySQL. Usando tanto MQTT (publicación) como REST (método POST y GET)

En un documento describa todo el procedimiento realizado y muestre evidencias de su funcionamiento.