TALLER 7b. Servidor IoT-MongoDB

Inicio

El objetivo de este taller es entender los conceptos relacionados con el almacenamiento de los datos en bases de datos NO 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:

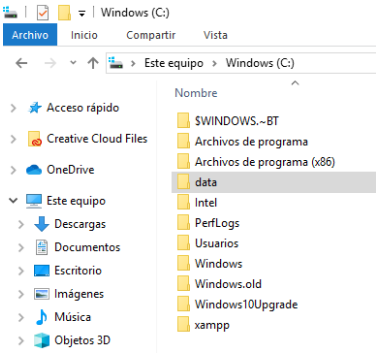
1. Configuración y desarrollo del código para almacenar y recuperar datos almacenados en MongoDB a través de un servidor IoT con soporte a MQTT
2. Configuración y desarrollo del código para almacenar y recuperar datos almacenados en MongoDB a través de un servidor IoT con soporte a REST

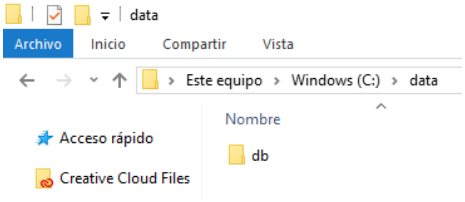
Instalación del ambiente de trabajo

1. Instalación de MongoDB

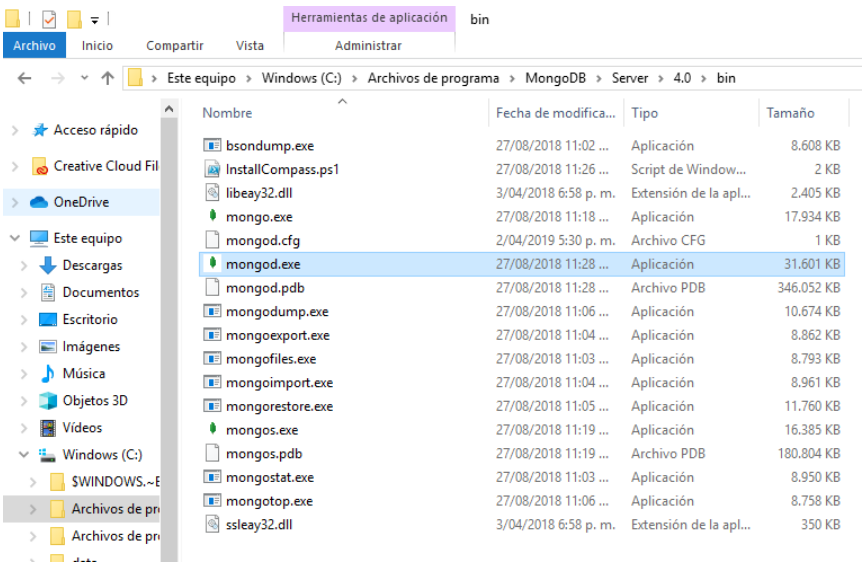
Descargar MongoDB de la siguiente página: https://www.mongodb.com/download-center?jmp=tutorials#community

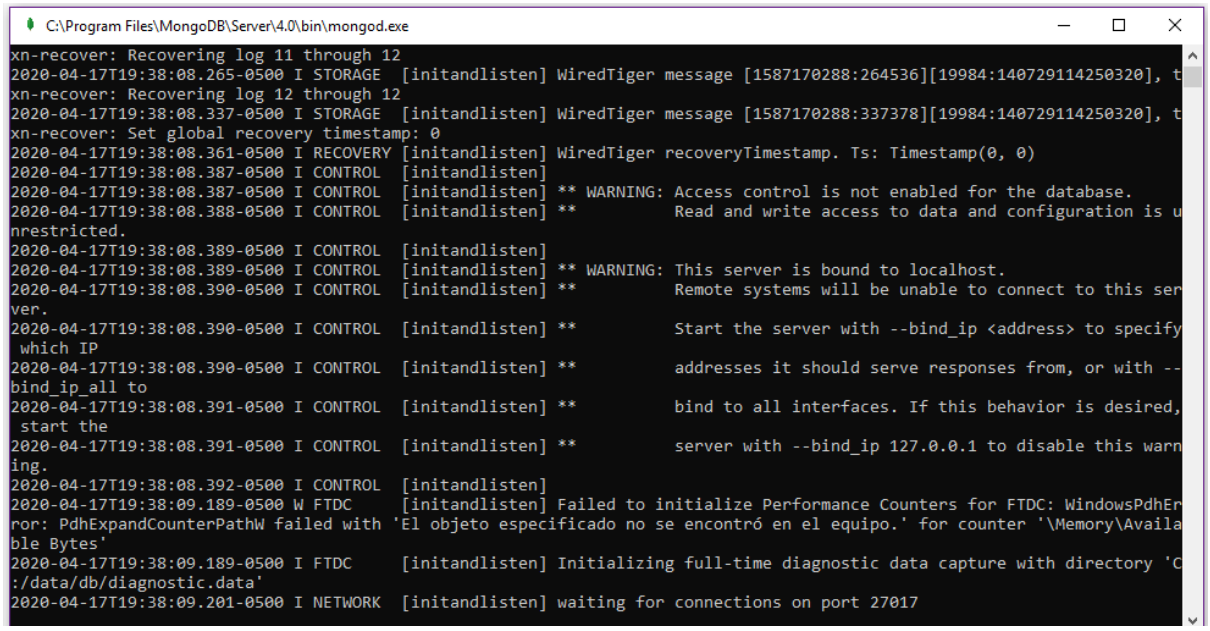
Iniciar el instalador como administrador y crear las carpetas de almacenamiento y configuración de MongoDB: C:\data y C:\data\db



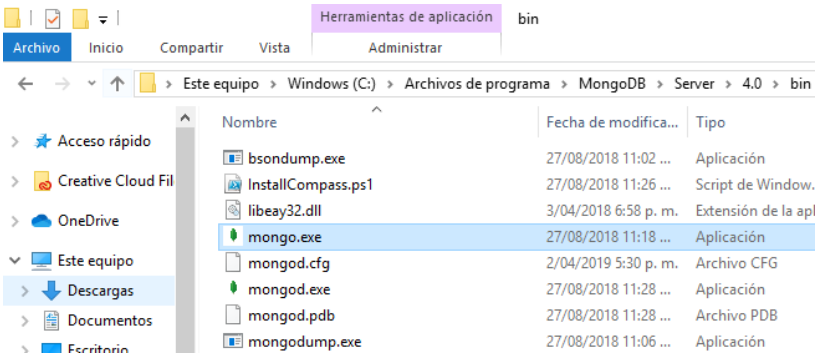


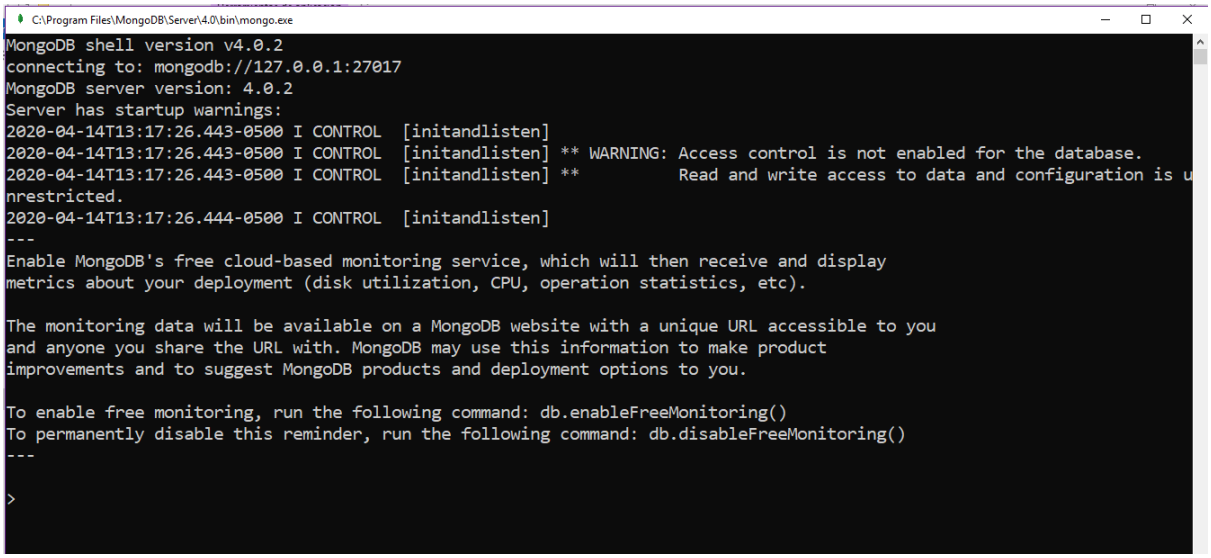
Iniciar el servidor del servicio de MongoDB: mongod. (dejar la ventana del cmd abierta)





Iniciar el shell de mongo. (en esta ventana se ejecutan los comandos)





Servidor IoT con soporte a MQTT y MongoDB

2. Instalación de las librerías necesarias

Se debe añadir MongoDB al proyecto que venimos manejando:

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

+ mongodb@4.8.0
added 17 packages from 56 contributors and audited 137 packages in 5.952s

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

found 0 vulnerabilities
```

3. Código para acceder a la base de datos

Modificamos el archivo indexmqtt.js para que se acceda a la base de datos MongoDB 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, como en el taller anterior, que en este caso debería tener la siguiente estructura:

```
{
    "idnodo": 1,
    "temperatura": 24.5,
    "humedad": 65,
    "timestamp": 1658522651
}
```

El archivo indexmqtt2.js quedaría de la siguiente forma:

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

var MongoClient = require('mongodb').MongoClient;
var url = "mongodb://localhost:27017/";

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')
    MongoClient.connect(url, function (err, db) {
        if (err) throw err;
        var dbo = db.db("ProyectoMongo");
        dbo.collection("datosNodo").insertOne(json1, function (err, res) {
            if (err) throw err;
            console.log("1 document inserted");
            db.close();
        });
    });

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

4. 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/indexmqtt2.js
```

Publicamos en el tópico topico1

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

```
PS D:\Users\zsolarte\Desktop\ServidorIoT> node src/indexmqtt2.js
{ idnodo: 1, temperatura: 24.5, humedad: 65, timestamp: 1657825613 }
1 document inserted
```

Verificamos en mongoDB que se haya realizado la inserción del dato:

```
> show databases;
BDPrueba        0.000GB
CaliDatos       0.000GB
CaliDatos2      0.000GB
ProyectoMongo   0.000GB
admin           0.000GB
caliclima       0.000GB
caliclima2      0.000GB
calidatos3      0.000GB
clima           0.000GB
config          0.000GB
datos           0.000GB
datoscali       0.000GB
datosdb         0.000GB
local           0.000GB
prueba          0.000GB
> use ProyectoMongo
switched to db ProyectoMongo
> db.datosNodo.find()
{ "_id" : ObjectId("62db0da31f8e1bf3c93ccb42"), "idnodo" : 1, "temperatura" : 24.5, "humedad" : 65, "timestamp" : 1657825613 }
>
```

Servidor IoT con soporte a REST y MongoDB

5. Código para acceder a la base de datos

Creamos en el directorio de rutas un archivo con el nombre datosm.js y copiamos el siguiente código. Aquí se desarrolla el post y el get para traer todos los datos almacenados.

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

var MongoClient = require('mongodb').MongoClient;
var url = "mongodb://localhost:27017/";

router.get('/datosm', (req, res) => {
```

```
    MongoClient.connect(url, function (err, db) {
        if (err) throw err;
        var dbo = db.db("ProyectoMongo");
        dbo.collection("datosNodo").find({}).toArray(function (err, result) {
            if (err) throw err;
            console.log(result);
            res.json(result);
            db.close();
        });
    });
});

router.post('/datosm', (req, res) => {
    console.log(req.body);
    var json2 = req.body;
    MongoClient.connect(url, function (err, db) {
        if (err) throw err;
        var dbo = db.db("ProyectoMongo");
        dbo.collection("datosNodo").insertOne(json2, function (err, res) {
            if (err) throw err;
            console.log("1 document inserted");
            db.close();
        });
    });
    res.send("dato insertado");
});

module.exports = router;
```

Se debe añadir la ruta en el archivo index.js que se encuentra en la carpeta src. Quedaría de la siguiente manera:

```
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'));
```

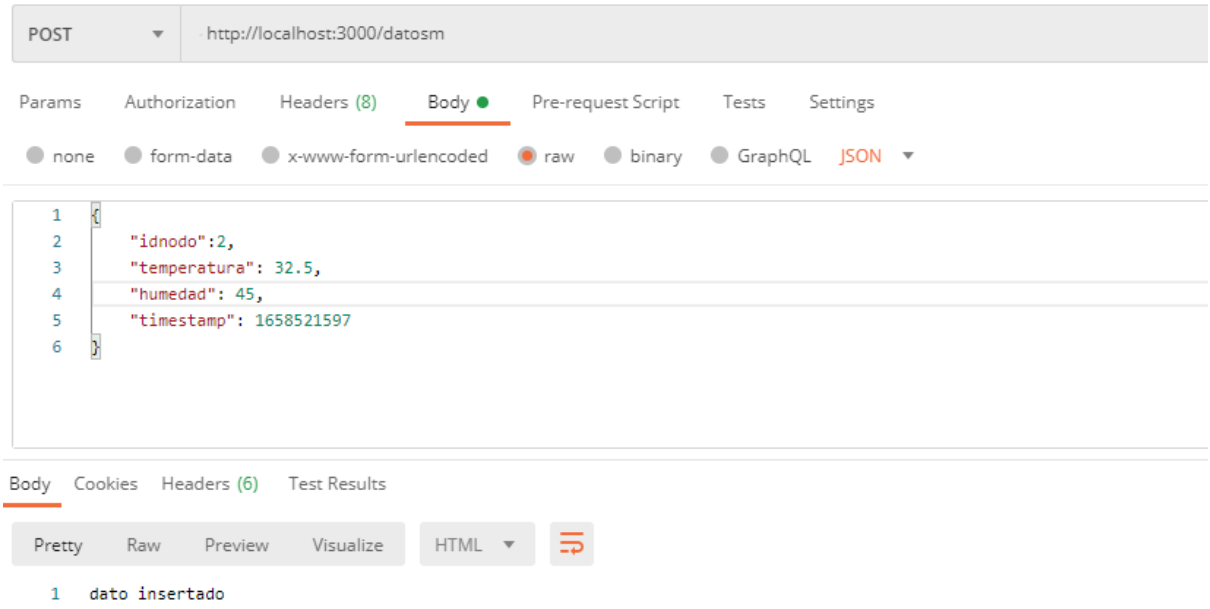
```
app.use(require('./rutas/datosm.js'));

//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.
```

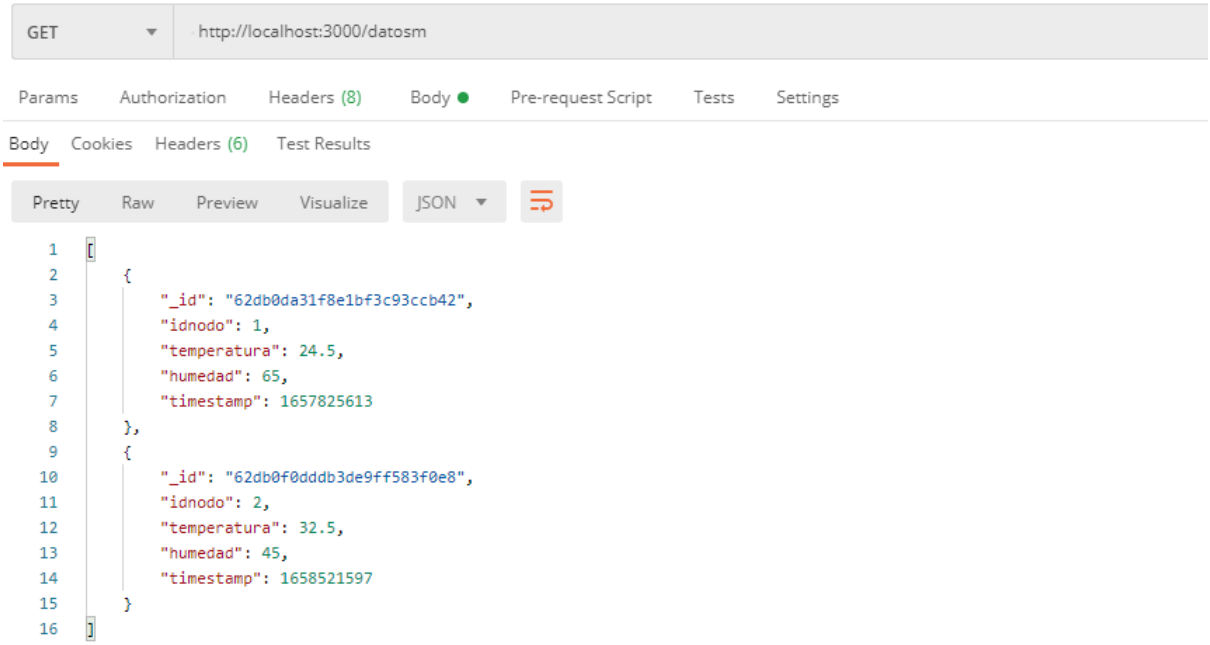
Lanzamos el servicio y lo probamos desde postman.

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

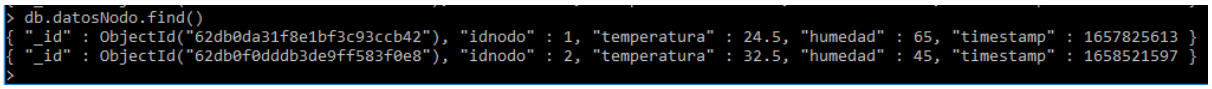
Probamos primero el post para almacenar datos en la base de datos. Recuerde añadir la ruta en la URL.



Probamos el get.



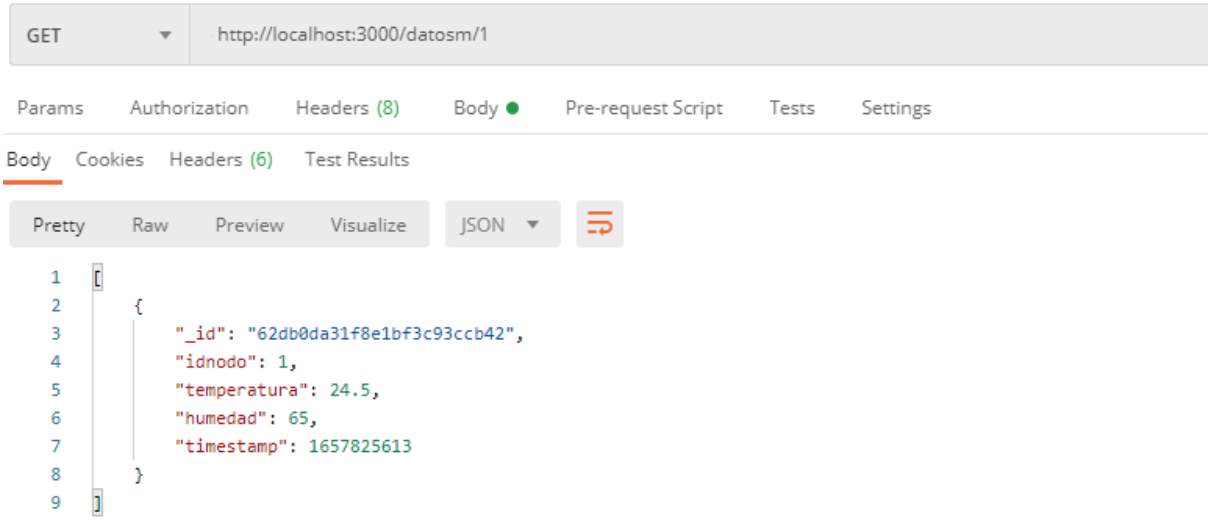
Ahora verificamos los datos en la base de datos:



Ahora creamos un get que nos permita traer los datos de un nodo especifico usando el idnodo. Para eso añadimos el siguiente código a datosm.js

```
router.get('/datosm/:idnodo', (req, res) => {
    var id = req.params.idnodo; //recogemos el parámetro enviado en la url
    MongoClient.connect(url, function (err, db) {
        if (err) throw err;
        var dbo = db.db("ProyectoMongo");
        var query = {};
        query.idnodo = parseInt(id);
        //console.log(query);
        dbo.collection("datosNodo").find(query).toArray(function (err, result)
{
            if (err) throw err;
            console.log(result);
            res.json(result);
            db.close();
        });
    });
});
```

Reiniciamos el servicio y probamos el nuevo get desde postman.



Integración con el objeto IoT

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 MongoDB. Usando tanto MQTT (publicación) como REST (método POST y GET)

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