



Instituto Tecnológico de San Juan del Río



Miniproyecto. Web Api concurrente con dispositivo de IoT.

P R E S E N T A:

Ana Karla García Gudiño 15590687

David Olaf Menchaca Cruz 15590711

Jonathan Martínez Sánchez 1559070

Ingeniería en Sistemas Computacionales

San Juan del Río, Qro., Marzo de 2021

La conexión del interruptor se conectó al pin D1 y D0 para poder mandar la señal, los cables de interruptor se soldaron para ténelos más fijos y no sea un inconveniente, figura 1

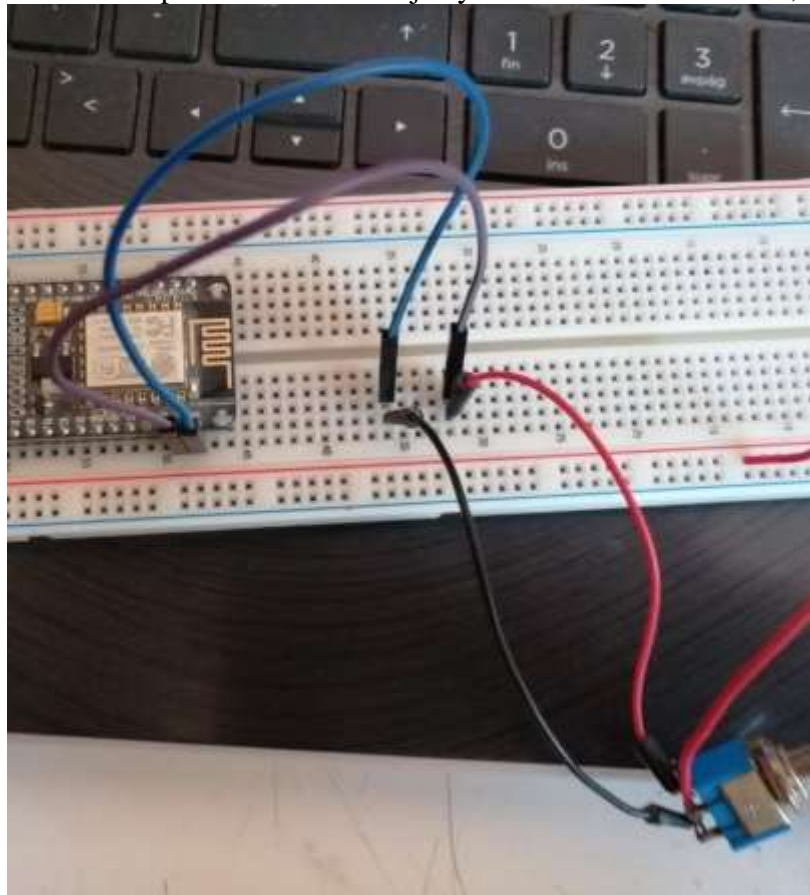


Figura 1: Conexión

Una vez conectados estos conectamos a GND y voltaje utilizando resistencias de 10 kh, dando como circuito final, figura 2

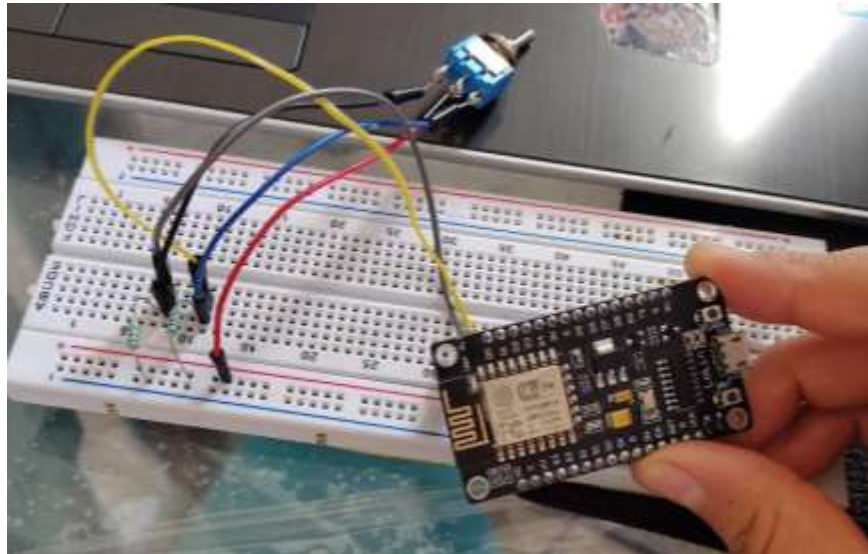


Figura 2: conexión circuito

Se muestra el código Arduino con el que se realizó la primera conexión con el interruptor de 3 estados cola de rata y la placa nodemcu, figura 3



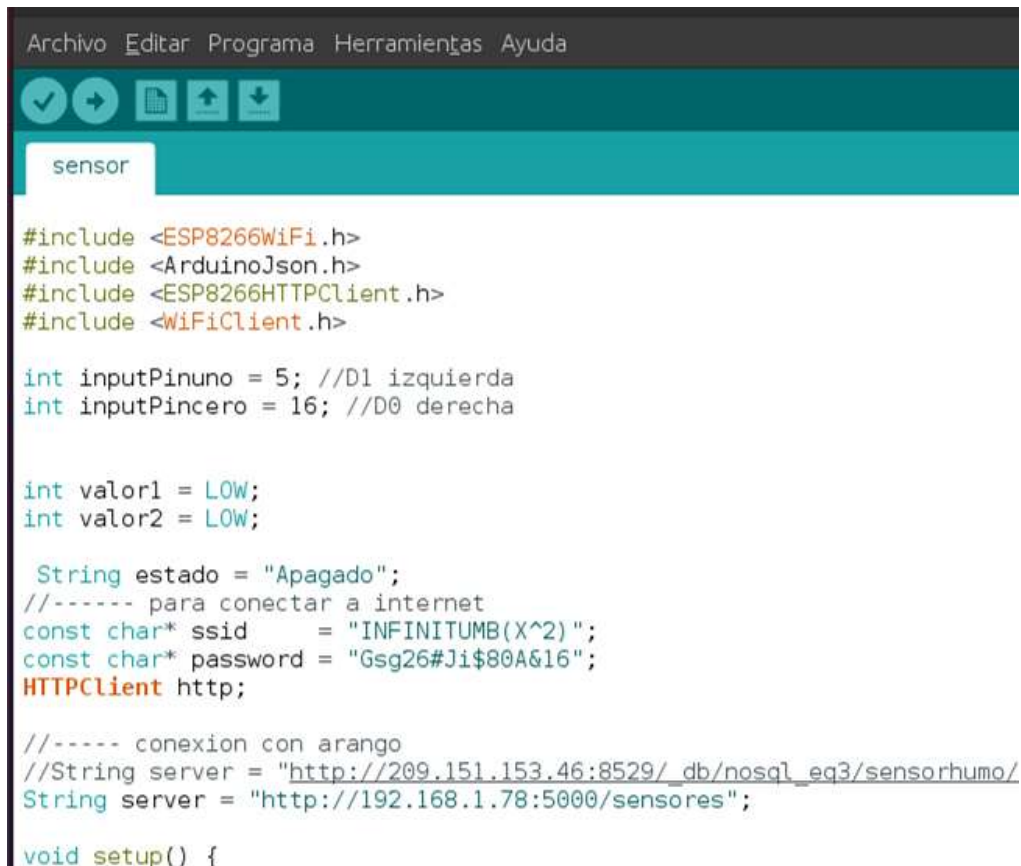
```
int inputPinuno = 5; //D1 izquierda
int inputPincero = 16; //D0 derecha

void setup() {
  Serial.begin(9600);
  pinMode(inputPinuno, INPUT);
  pinMode(inputPincero, INPUT);
}

void loop() {
  int valor1 = digitalRead(inputPinuno); //lectura digital de pin
  int valor2 = digitalRead(inputPincero); //lectura digital de pin
  // Serial.println(valor1);
  //Serial.println(valor2);
  if(valor1 == HIGH){
    Serial.println("IZQUIERDA");
    delay(100);
  }
  else{
    if(valor2 == HIGH){
      Serial.println("DERECHA");
      delay(100);
    }
    else
    {
      Serial.println("APAGADO");
      delay(100);
    }
  }
}
```

Figura 3: código Arduino placa e interruptor

Una vez realizado el código anterior nos guiamos para poder realizar la conexión con wifi y pueda mandar los datos utilizamos librerías como `#include <ESP8266WiFi.h>` , `#include <ArduinoJson.h>` para poder realizar la conexión y poder mandar los datos en formato json, se muestra los pines que fueron utilizados a la red que se conectó y al servidor en el que se registran los valores de la base de datos, figura 4



```
Archivo Editar Programa Herramientas Ayuda

sensor

#include <ESP8266WiFi.h>
#include <ArduinoJson.h>
#include <ESP8266HTTPClient.h>
#include <WiFiClient.h>

int inputPinuno = 5; //D1 izquierda
int inputPincero = 16; //D0 derecha

int valor1 = LOW;
int valor2 = LOW;

String estado = "Apagado";
//----- para conectar a internet
const char* ssid = "INFINITUMB(X^2)";
const char* password = "Gsg26#Ji$80A&l6";
HTTPClient http;

//----- conexion con arango
//String server = "http://209.151.153.46:8529/_db/nosql_eq3/sensorhumo/";
String server = "http://192.168.1.78:5000/sensores";

void setup() {
```

Figura 4: Librerías utilizadas

Se realiza la conexión con los datos establecidos de ssid y password y se realiza la conexión figura 5

```
void setup() {
    Serial.begin(9600); // sets the serial port to 9600
    WiFi.mode(WIFI_STA);
    WiFi.begin(ssid, password);
    while (WiFi.status() != WL_CONNECTED) {
        delay(500);
        Serial.print(".");
    }
    Serial.println("");
    Serial.println("WiFi connected");
    Serial.println("IP address: ");
    Serial.println(WiFi.localIP());
    delay(1000);

    //codigo

    pinMode(inputPinuno, INPUT);|
    pinMode(inputPincero, INPUT);

}
```

Figura 5: conexión wi-fi

En el apartado de Loop condicionamos para que mande el estado según funcione el interruptor y este mande si esta en “izquierda”, “derecha” o “apagado”, figura 6.

```
void loop() {  
  valor1 = digitalRead(inputPinuno); //lectura digital de pin  
  valor2 = digitalRead(inputPincero); //lectura digital de pin  
  // Serial.println(valor1);  
  //Serial.println(valor2);  
  if(valor1 == HIGH && valor2 == LOW){  
    estado = "IZQUIERDA";  
    post(estado);  
    //Serial.println("IZQUIERDA");  
    delay(100);  
  }  
  else  
    if(valor2 == HIGH && valor1 == LOW){  
      estado = "DERECHA";  
      post(estado);  
      //Serial.println("DERECHA");  
      delay(100);  
    }  
  else  
    if(valor1==LOW && valor2 == LOW)  
    {  
      estado = "Apagado";  
    }
```

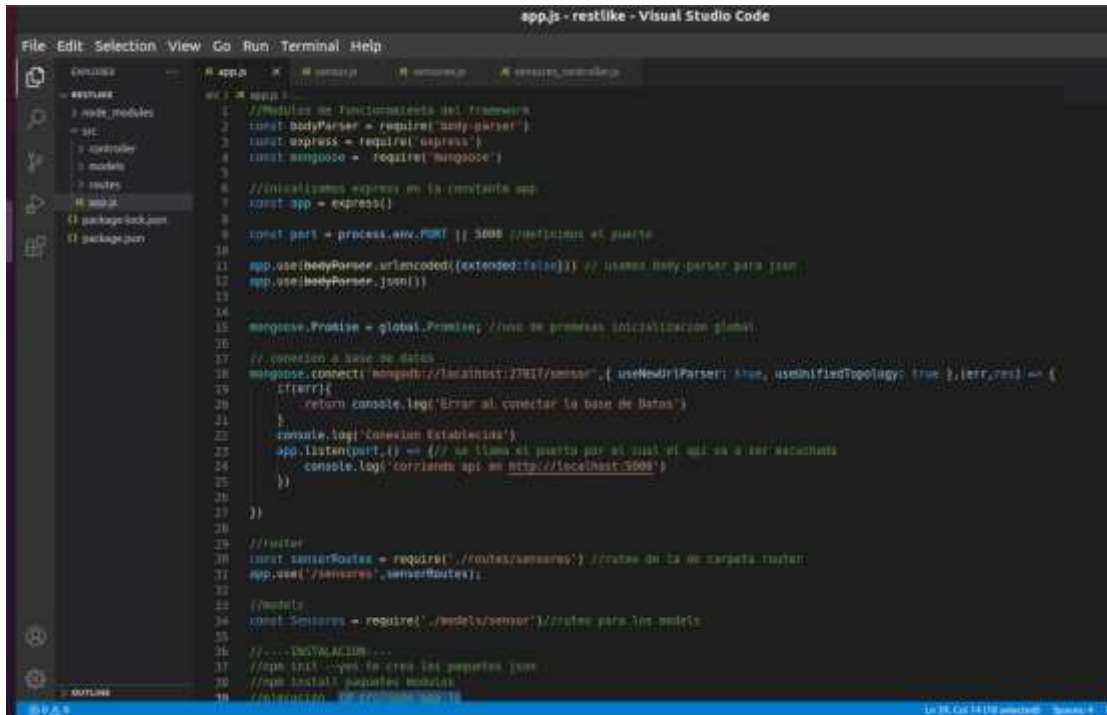
Figura 6: estado de interruptor

Declaramos un método el cual permite leer el estado que se este mandando el interruptor y mande un json mediante un método post , figura 7.

```
    }  
    else  
    if(valor1==LOW && valor2 == LOW)  
    {  
        estado = "Apagado";  
        post(estado);  
        //Serial.println("APAGADO");  
        delay(100);  
    }  
}  
void post(String estado){  
    HTTPClient http;  
    String json;  
    StaticJsonBuffer<200> jsonBuffer;  
    JsonObject& root = jsonBuffer.createObject();  
    root["estado"] = estado;  
    root.printTo(json);  
    Serial.println(""); // salto de linea para http.writeToStream(&Serial);  
    http.begin(server);  
    http.addHeader("Content-Type", "application/json");  
    http.POST(json);  
    http.writeToStream(&Serial);  
    http.end();  
}
```

Figura 7: mandar estado

Para poder realizar la api utilizamos Node.js express utilizando base de datos de mongo, utilizando el módulo de mongoose, creamos la estructura para poder trabajar en app.js insertamos los módulos del funcionamiento del framework, inicializamos constante de express, definimos el puerto, definimos conexión, figura 8.



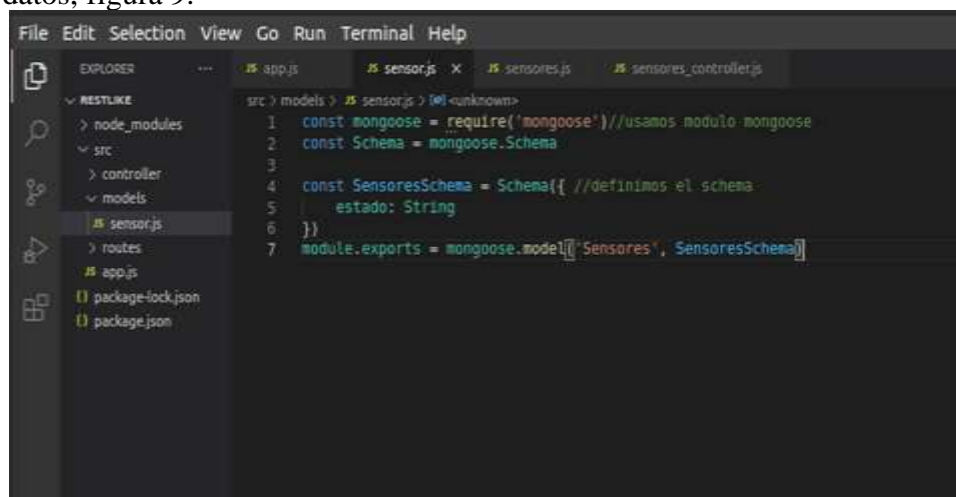
```

1 //Módulos de funcionamiento del framework
2 const bodyParser = require('body-parser')
3 const express = require('express')
4 const mongoose = require('mongoose')
5
6 //inicializamos express en la constante app
7 const app = express()
8
9 const port = process.env.PORT || 5000 //definimos el puerto
10
11 app.use(bodyParser.urlencoded({extended:false})) // usamos body parser para json
12 app.use(bodyParser.json())
13
14
15 mongoose.Promise = global.Promise; //uso de promesas inicializacion global
16
17 //conexion a base de datos
18 mongoose.connect('mongodb://localhost:27017/sensor', { useNewUrlParser: true, useUnifiedTopology: true }, (err, res) => {
19   if(err){
20     return console.log('Error al conectar la base de datos')
21   }
22   console.log('Conexion Establecida')
23   app.listen(port, () => { // se llama al puerto por el cual el api va a ser escuchada
24     console.log('corriendo api en http://localhost:5000')
25   })
26 })
27
28 //rutas
29 const sensorRoutes = require('./routes/sensores') //rutas de la carpeta routes
30 app.use('/sensores', sensorRoutes);
31
32 //models
33 const Sensores = require('./models/sensor') //rutas para los models
34
35 //---INSTALACION---
36 //npm init --yes te crea los paquetes json
37 //npm install paquetes modules
38 //instalacion de express mongoose

```

Figura 8: archivo app

En el archivo de sensor.js utilizamos mongoose para que pueda realizarse la conexión con la base de datos, figura 9.



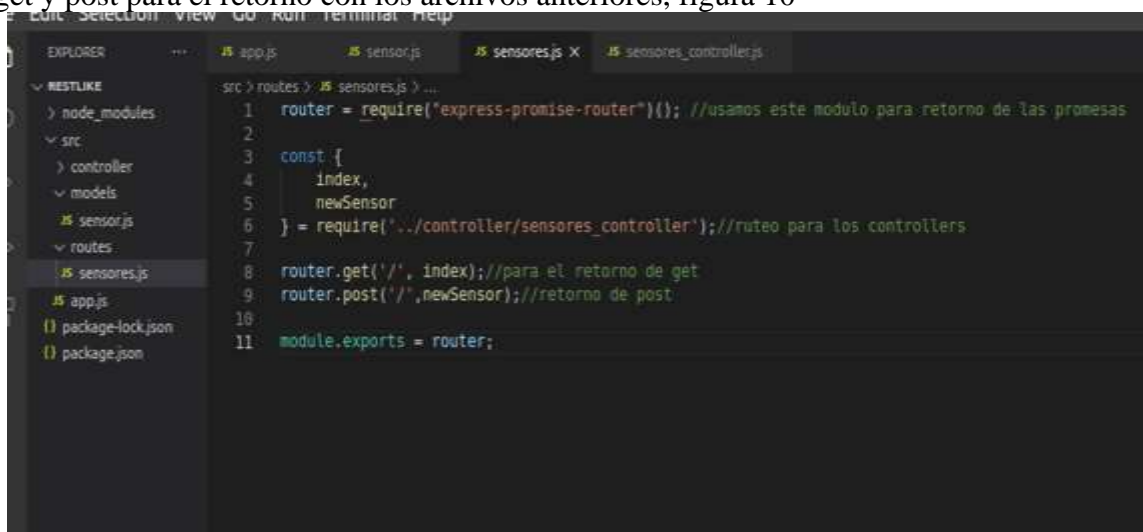
```

1 const mongoose = require('mongoose') //usamos modulo mongoose
2 const Schema = mongoose.Schema
3
4 const SensoresSchema = Schema({ //definimos el schema
5   estado: String
6 })
7 module.exports = mongoose.model('Sensores', SensoresSchema)

```

Figura 9: archivo sensor.js

En el archivo de sensores.js declaramos un modulo para el retorno de promesas definimos get y post para el retorno con los archivos anteriores, figura 10



```
src > routes > .js sensores.js > ...
1  router = _require("express-promise-router")(); //usamos este modulo para retorno de las promesas
2
3  const {
4    index,
5    newSensor
6  } = require('../controller/sensores_controller');//ruteo para los controllers
7
8  router.get('/', index);//para el retorno de get
9  router.post('/', newSensor);//retorno de post
10
11 module.exports = router;
```

Figura 10: sensores.js

El archivo de sensores_controller declaramos la constante para el ruteo a models, hacemos el uso de la función de async y await que devuelve la promesa en base a la función, figura 11.



```
src > controller > .js sensores_controller.js > ...
1  const Sensor = require('../models/sensor');// ruteo a modelo
2
3  module.exports = {
4    index: async(req, res, next) =>{//usa de función async la cual devuelve un objeto
5      const sensor = await Sensor.findById();//función await pausa la función async en la que resuelve la promesa pasada
6      res.status(200).json(sensor);
7    },
8    newSensor: async(req, res, next) => {
9      console.log("POST");
10     console.log(req.body);//Nos manda por consola el POST agregado
11
12     const sensor = new Sensor();
13
14     sensor.estado = req.body.estado;
15     await sensor.save();// resuelve la promesa con una función await
16   }
17 };
```

Figura 11: archivo sensores_controllers

Se realizaron pruebas en Insomnia antes de realizar la conexión del Arduino para la base de datos, figura 12.

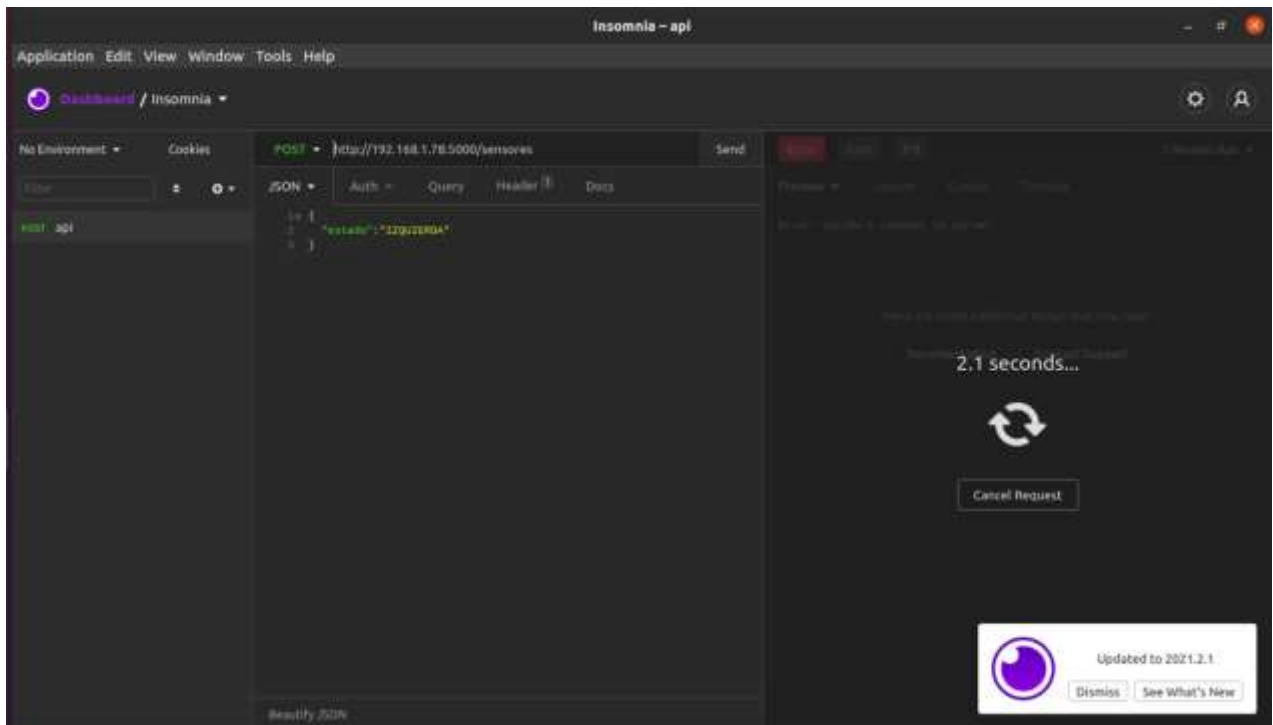


Figura 12: prueba con Insomnia

Tanto en la prueba con Insomnia y con la conexión desde el sensor se registraron correctamente en la base de datos, figura 13

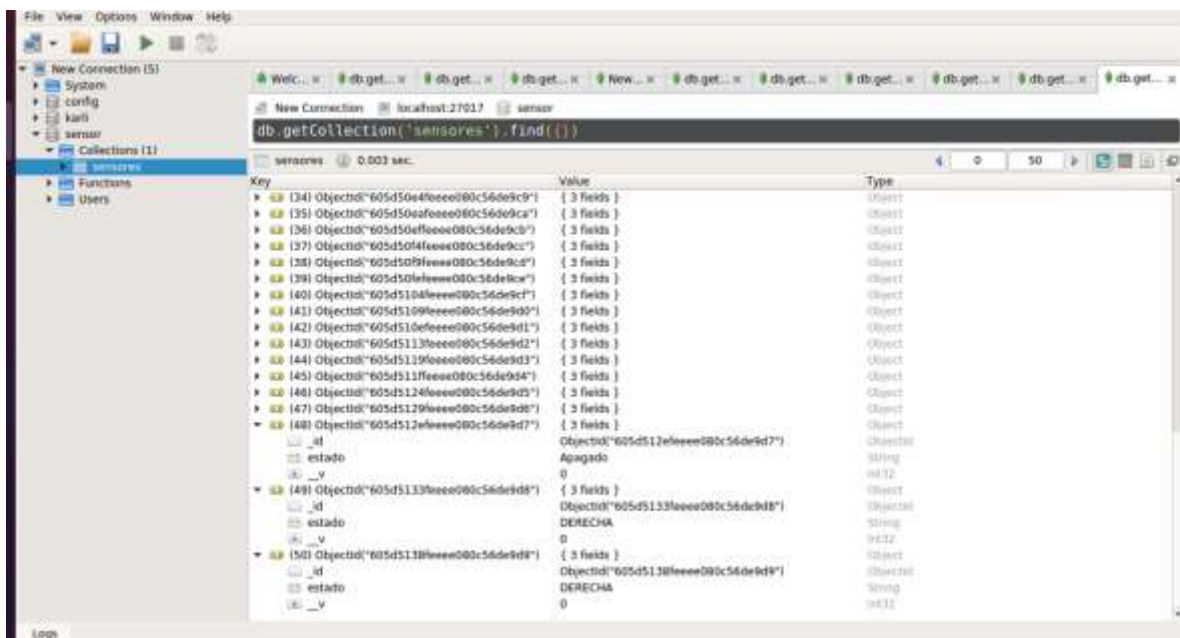


Figura 13: Base de Datos

Diagrama UML

