



Instituto Tecnológico de San Juan del Río



Proyecto Almacenamiento

P R E S E N T A N:

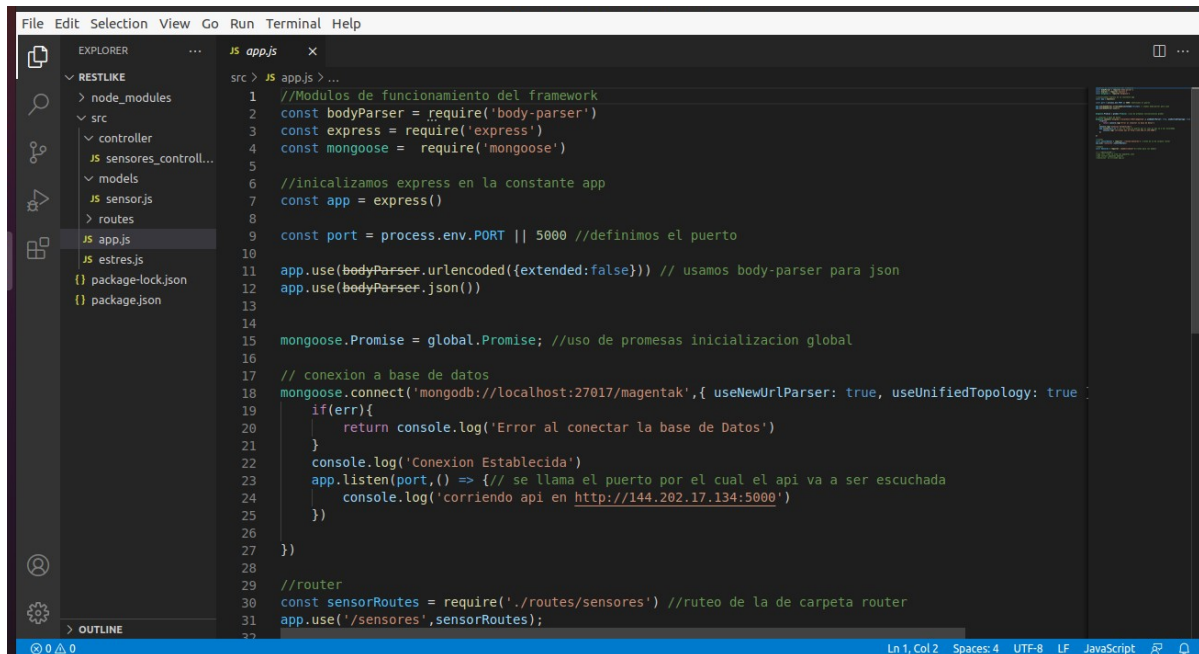
Ana Karla Garcia Gudiño 15590687
David Olaf Menchaca Cruz 15590711

Ingeniería en Sistemas Computacionales

San Juan del Río, Qro., Junio de 2021



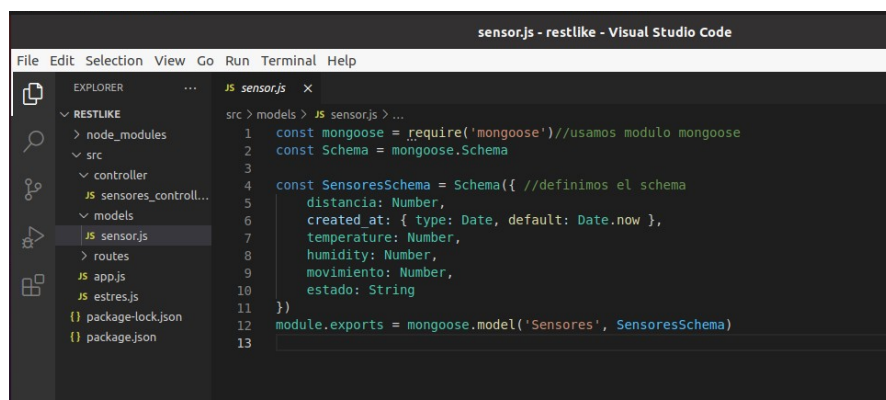
Se utiliza nodejs con expres y como base de datos se utiliza mongo, en el apartado, en la figura 1 se muestra el archivo de app.js, en la cual definimos la conexión con la base de datos



```
File Edit Selection View Go Run Terminal Help
EXPLORER
RESTLIKE
node_modules
src
controller
sensores_controll...
models
sensor.js
routes
app.js
estres.js
package-lock.json
package.json
OUTLINE
src > JS app.js > ...
1 //Modulos 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/magentak',{ useNewUrlParser: true, useUnifiedTopology: true
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 el puerto por el cual el api va a ser escuchada
24   console.log('corriendo api en http://144.202.17.134:5000')
25 })
26
27 })
28
29 //router
30 const sensorRoutes = require('./routes/sensores') //ruteo de la de carpeta router
31 app.use('/sensores',sensorRoutes);
32
```

Figura 1: archivo app.js

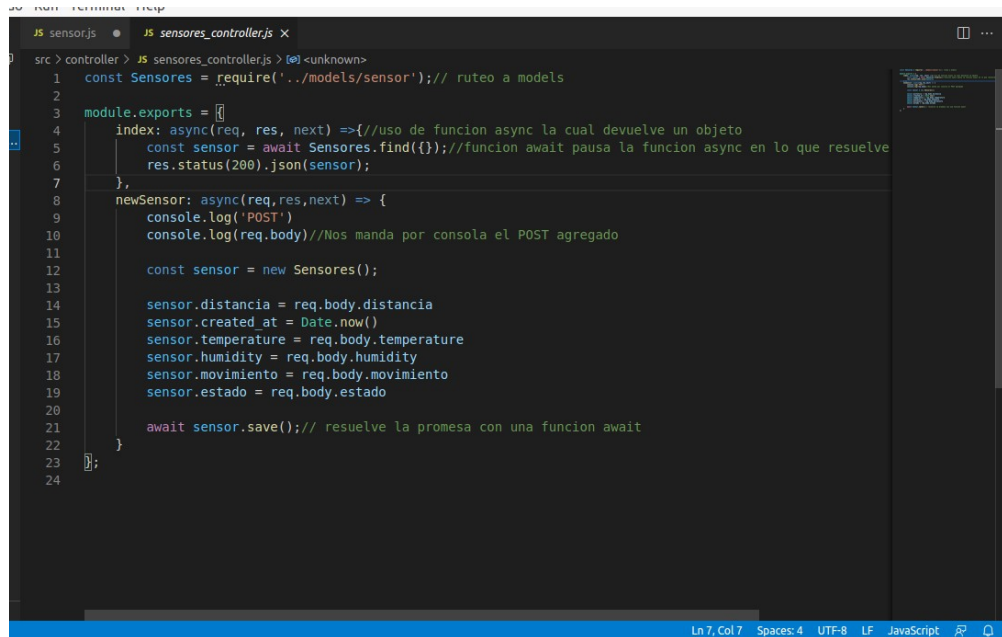
en el archivo de sensor.js definimos los datos que enviamos como lo es temperatura, humedad, distancia, en este archivo tambien mandamos el dia y la fecha que se estan ejecutando los sensores, figura 2



```
sensor.js - restlike - Visual Studio Code
File Edit Selection View Go Run Terminal Help
EXPLORER
RESTLIKE
node_modules
src
controller
sensores_controll...
models
sensor.js
routes
app.js
estres.js
package-lock.json
package.json
OUTLINE
src > models > JS sensor.js > ...
1 const mongoose = require('mongoose')//usamos modulo mongoose
2 const Schema = mongoose.Schema
3
4 const SensoresSchema = Schema({ //definimos el schema
5   distancia: Number,
6   created at: { type: Date, default: Date.now },
7   temperature: Number,
8   humidity: Number,
9   movimiento: Number,
10  estado: String
11 })
12 module.exports = mongoose.model('Sensores', SensoresSchema)
13
```

Figura 2: archivo sensor.js

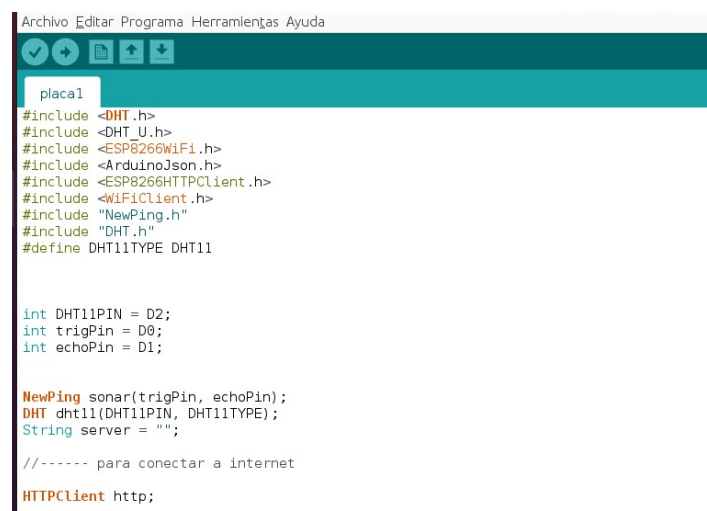
Dentro de los archivos de los controladores utilizando async , y los valores de los sensores que se utilizaron, figura 3



```
1 const Sensores = require('../models/sensor');// ruteo a models
2
3 module.exports = {
4   index: async(req, res, next) =>{//uso de funcion async la cual devuelve un objeto
5     const sensor = await Sensores.find({});//funcion await pausa la funcion async en lo que resuelve
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 Sensores();
13
14    sensor.distancia = req.body.distancia
15    sensor.created_at = Date.now()
16    sensor.temperature = req.body.temperature
17    sensor.humidity = req.body.humidity
18    sensor.movimiento = req.body.movimiento
19    sensor.estado = req.body.estado
20
21    await sensor.save();// resuelve la promesa con una funcion await
22  }
23 };
24
```

Figura 3: Archivo sensores_controller.js

Se crearon dos archivos uno para placa 1 y otro para placa 2, para placa uno utilizamos sensor DTH11, y humedad, utilizando el metodo post para poder mandar los datos figura 4, el codigo completo se muestra en <https://github.com/ana-gGarcia/ProAlmacenamiento.git>,



```
Archivo Editar Programa Herramientas Ayuda
placa1
#include <DHT.h>
#include <DHT_U.h>
#include <ESP8266WiFi.h>
#include <ArduinoJson.h>
#include <ESP8266HTTPClient.h>
#include <WiFiClient.h>
#include "NewPing.h"
#include "DHT.h"
#define DHT11TYPE DHT11

int DHT11PIN = D2;
int trigPin = D0;
int echoPin = D1;

NewPing sonar(trigPin, echoPin);
DHT dht11(DHT11PIN, DHT11TYPE);
String server = "";

//----- para conectar a internet
HTTPClient http;
```

Figura 4: código placa 1

para la placa 2 utilizamos un Sensor De Obstáculos Reflectivo Infrarojo, y un switch el cual nos manda valores de 0 para apagado y 1 para izquierda y 2 para derecha manda esos valores, desde la placa mandamos la fecha en la cual es la que se inserta cada uno de los datos con su fecha y hora correspondiente, figura 5, el código completo se muestra en <https://github.com/ana-gGarcia/ProAlmacenamiento.git> placa2



```
placa2

const char* ssid = "INFINITUMB(X^2)";
const char* password = "Gsg26#Ji$80A&16";
int inputPinuno = 5; //D1 izquierda
int inputPincero = 16; //D0 derecha
const int sensorPin = D2;

int valor1 = LOW;
int valor2 = LOW;

String estado= "";

void setup() {
  Serial.begin(115200); // 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);
}
```

Figura 5: código placa 2

Se actualizaron los archivos dentro de el servidor utilizando la misma URL que se utilizo en la api anterior, dejando la api ejecutada en segundo plano con screen, una vez que quedo ejecutada podemos visualizar la URL desde el navegador sin problema <http://144.202.17.134:5000/sensores>, figura 6

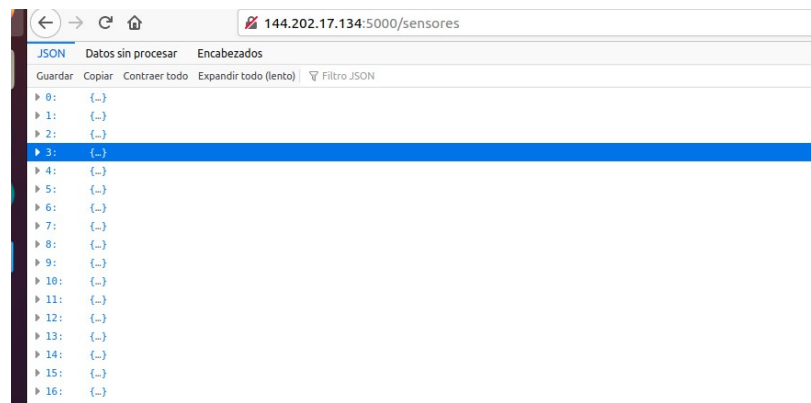


Figura 6: Visualizar api

dentro de cada uno podemos visualizar que tenemos la fecha registrada, figura 7

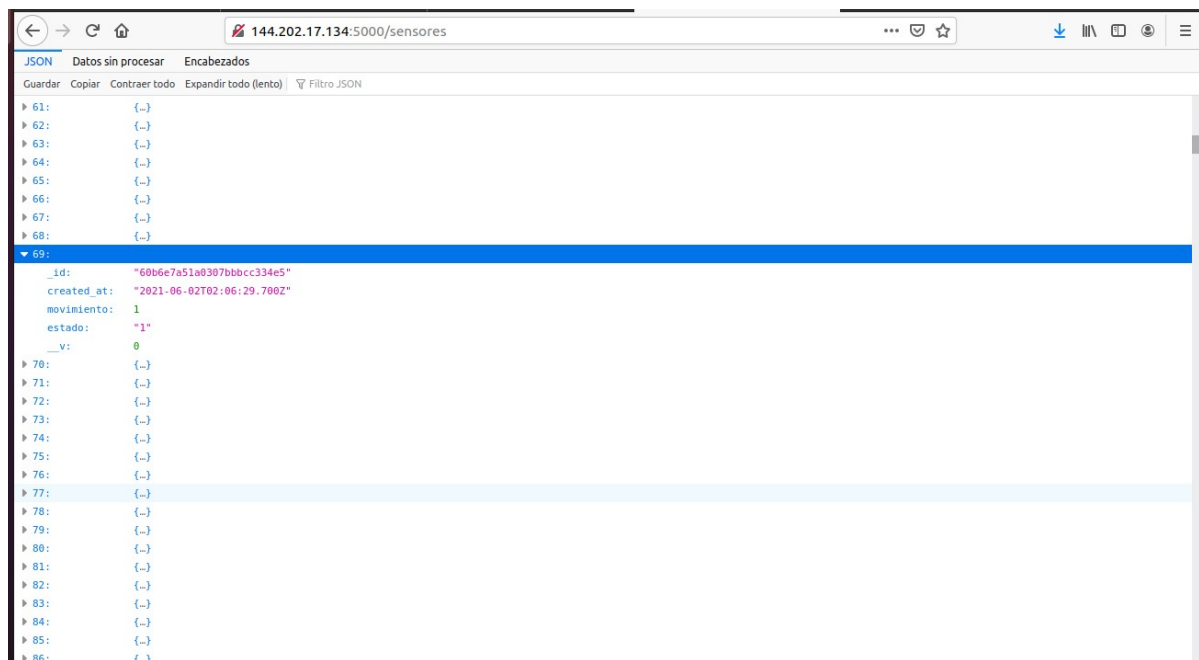


Figura 7: se muestra fecha que se ingresa

