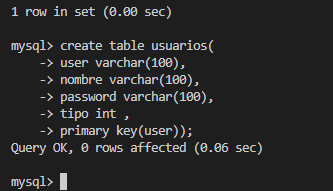
Actividad 6 IoT

Diego Iván Perea Montealegre (2238513) [diego.perea@uao.edu.co](mailto:diego.perea@uao.edu.co)

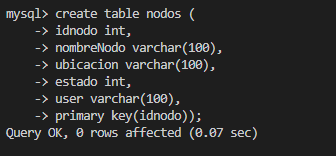
Facultad de Ingeniería, Universidad Autónoma de Occidente

Cali, Valle del Cauca

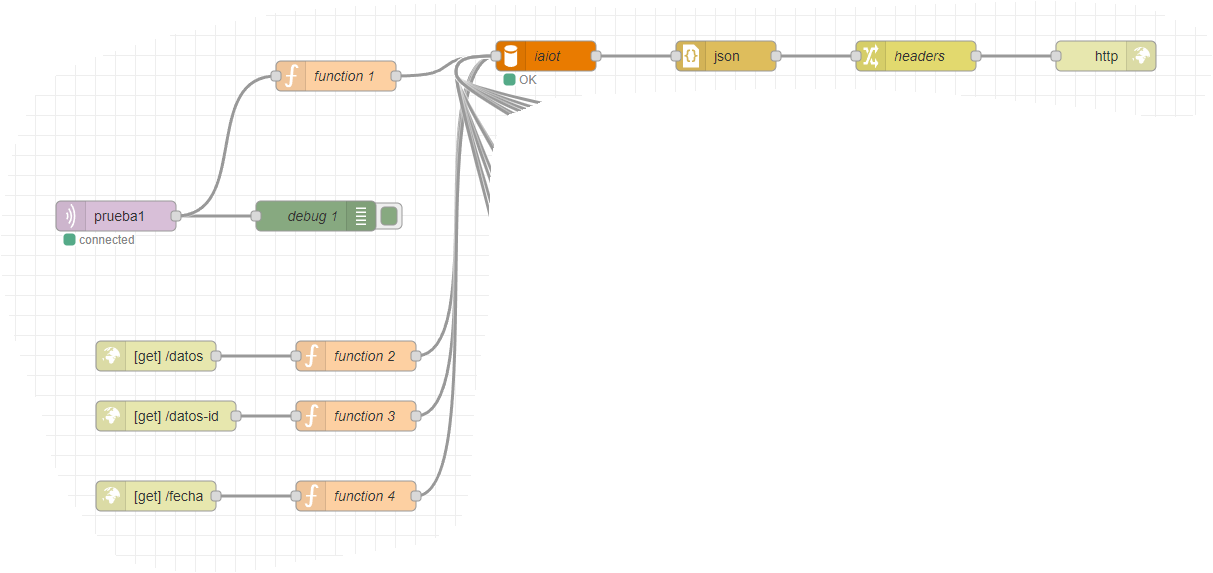
Creación de tabla de usuarios



Creación tabla nodos :



Se usa  para ejecutar node red:



Se cre la primera ruta y función



var user = msg.payload.user;

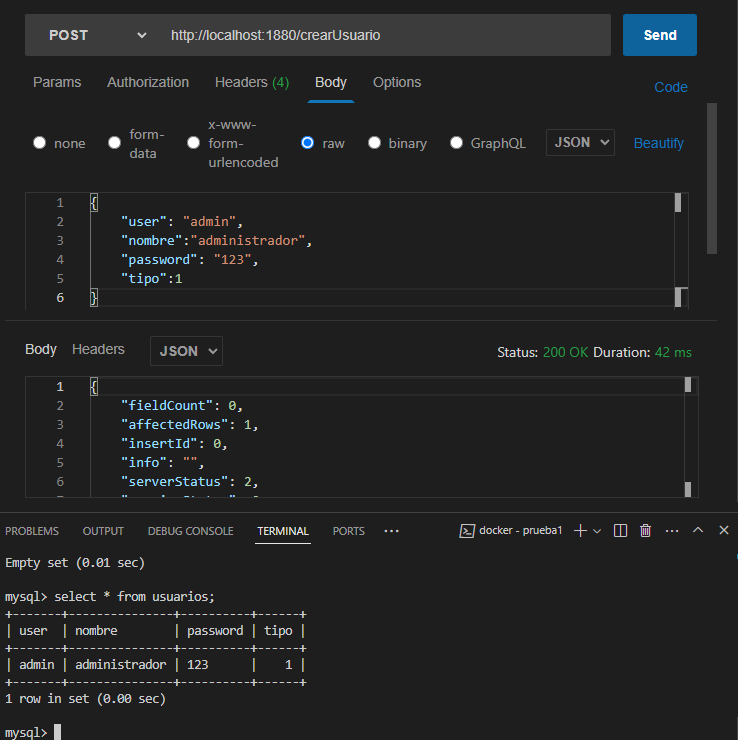
var nombre = msg.payload.nombre;

var pass = msg.payload.password;

var tipo = msg.payload.tipo;

msg.topic = `insert into usuarios values ("${user}","${nombre}","${pass}",${tipo})`;

return msg;



Se crea ruta y función



var user = msg.payload.user;

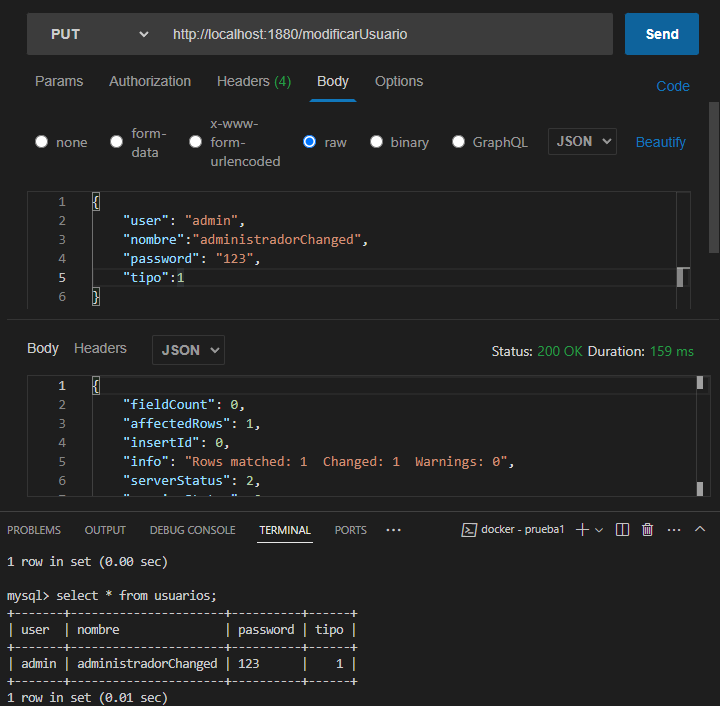
var nombre = msg.payload.nombre;

var pass = msg.payload.password;

var tipo = msg.payload.tipo;

msg.topic = `update usuarios set nombre="${nombre}", password="${pass}", tipo=${tipo} where user="${user}"`;

return msg;



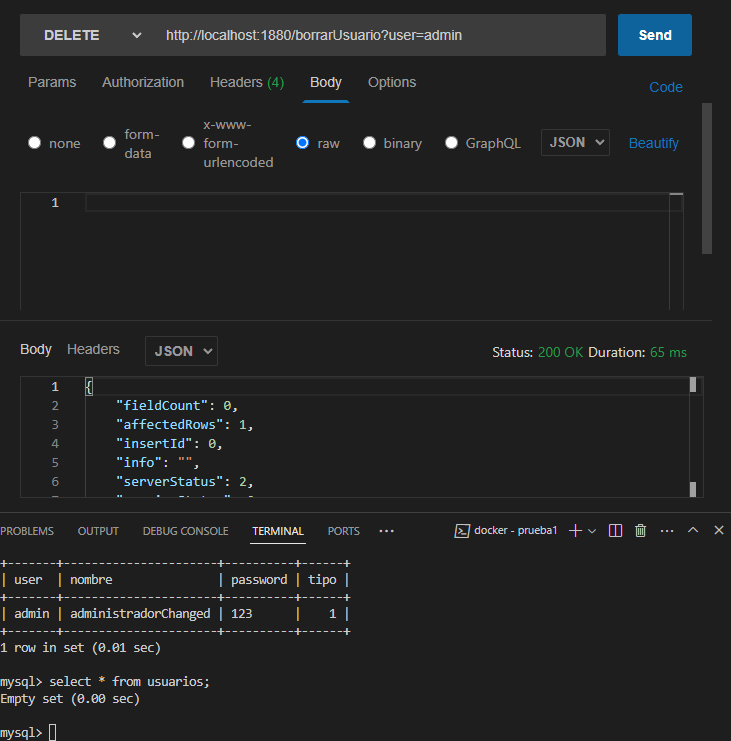
Se crea ruta y función



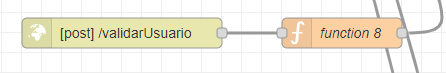
var user = msg.payload.user;

msg.topic = `delete from usuarios where user="${user}"`;

return msg;



Se crea ruta y función

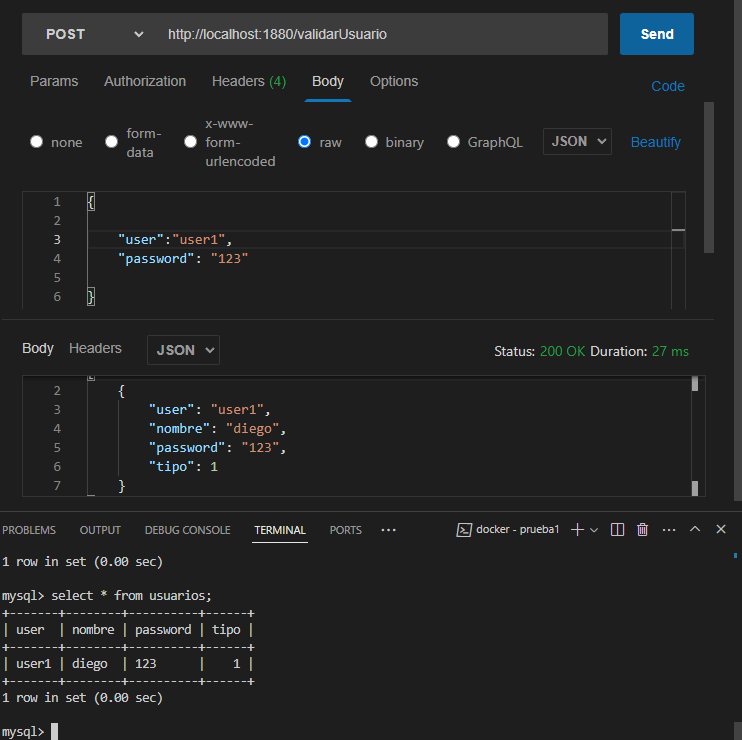


var user = msg.payload.user;

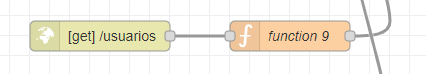
var pass = msg.payload.password;

msg.topic = `select \* from usuarios where user="${user}" and password="${pass}"`;

return msg;

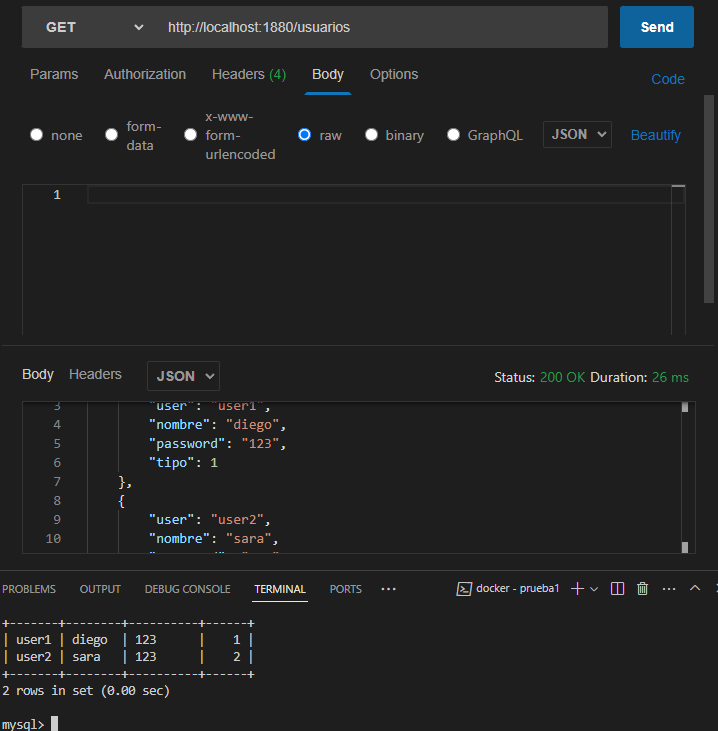


Se crea ruta y función

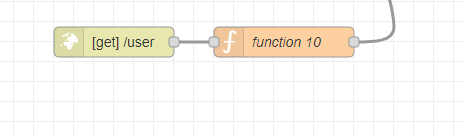


msg.topic = "select \* from usuarios";

return msg;



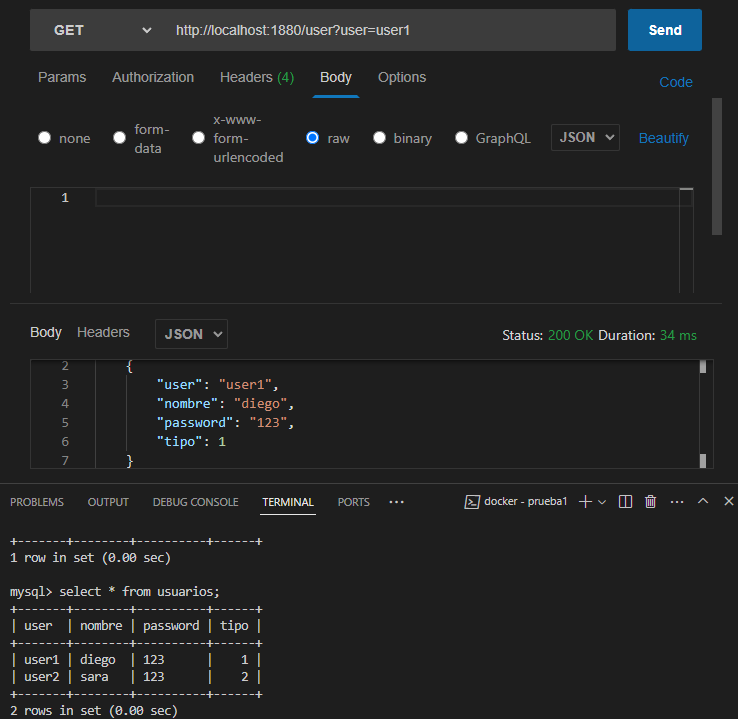
Se crea ruta y función



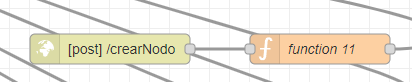
var user = msg.payload.user;

msg.topic = `select \* from usuarios where user="${user}"`;

return msg;



Se crea ruta y función



var idnodo = msg.payload.idnodo;

var nombre = msg.payload.nombre;

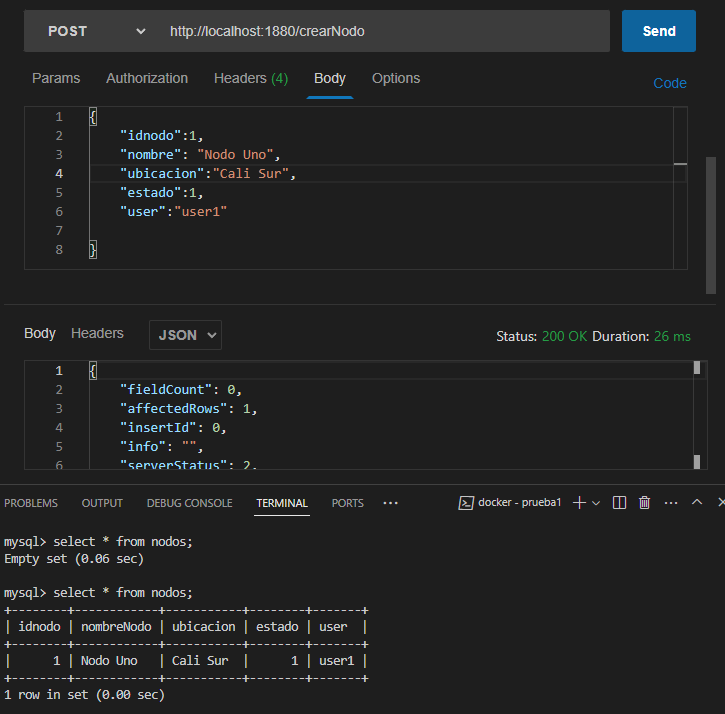
var ubic = msg.payload.ubicacion;

var estado = msg.payload.estado;

var user = msg.payload.user;

msg.topic = `insert into nodos values (${idnodo},"${nombre}","${ubic}",${estado},"${user}")`;

return msg;



Se crea ruta y función



var idnodo = msg.payload.idnodo;

var nombre = msg.payload.nombre;

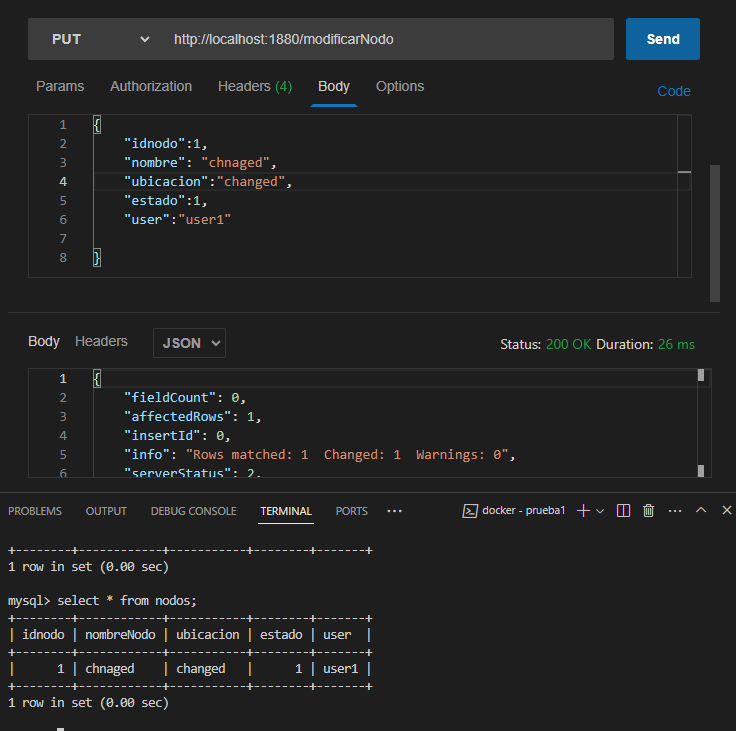
var ubic = msg.payload.ubicacion;

var estado = msg.payload.estado;

var user = msg.payload.user;

msg.topic = `update  nodos set nombreNodo="${nombre}",ubicacion="${ubic}",estado="${estado}",user="${user}" where idnodo=${idnodo}`;

return msg;



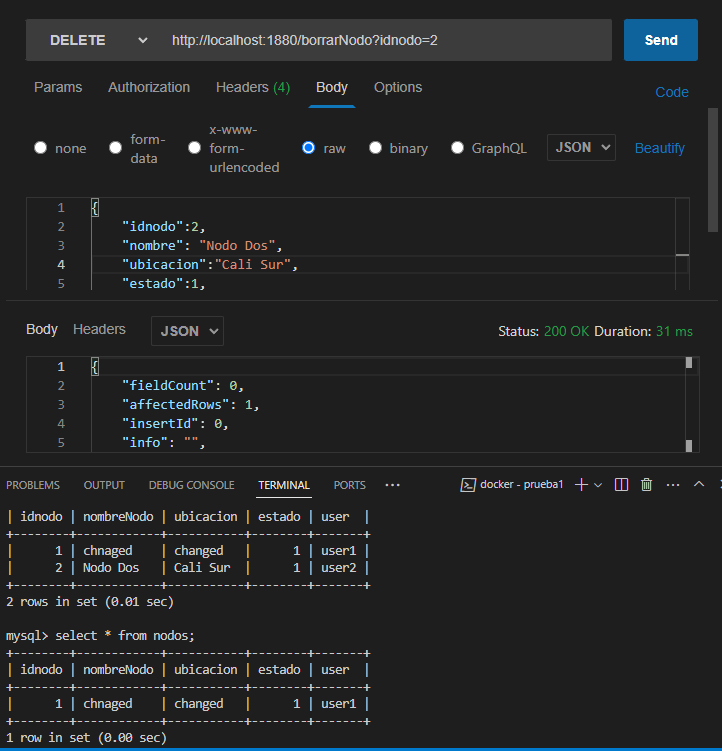
Se crea ruta y función



var idnodo = msg.payload.idnodo;

msg.topic = `delete from nodos where idnodo="${idnodo}"`;

return msg;

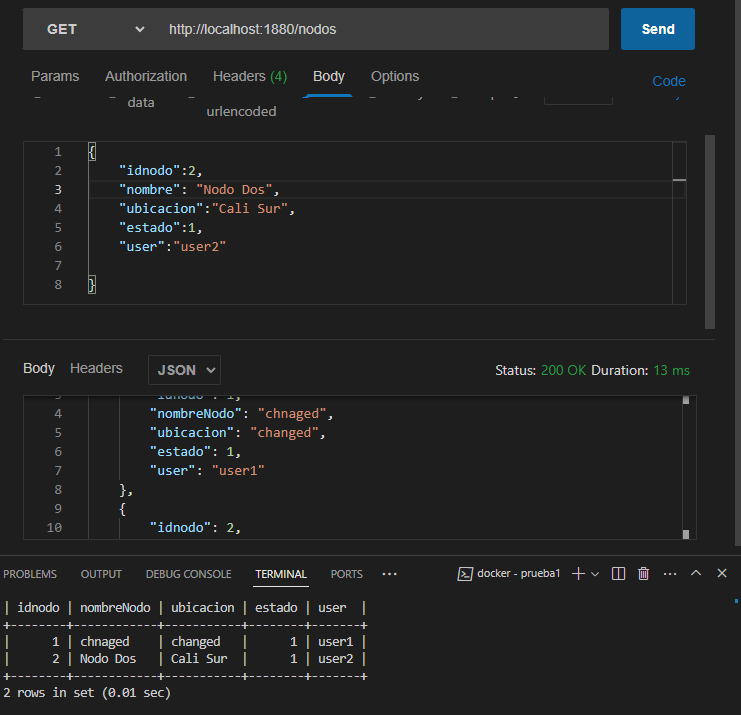


Se crea ruta y función



msg.topic = "select \* from nodos";

return msg;



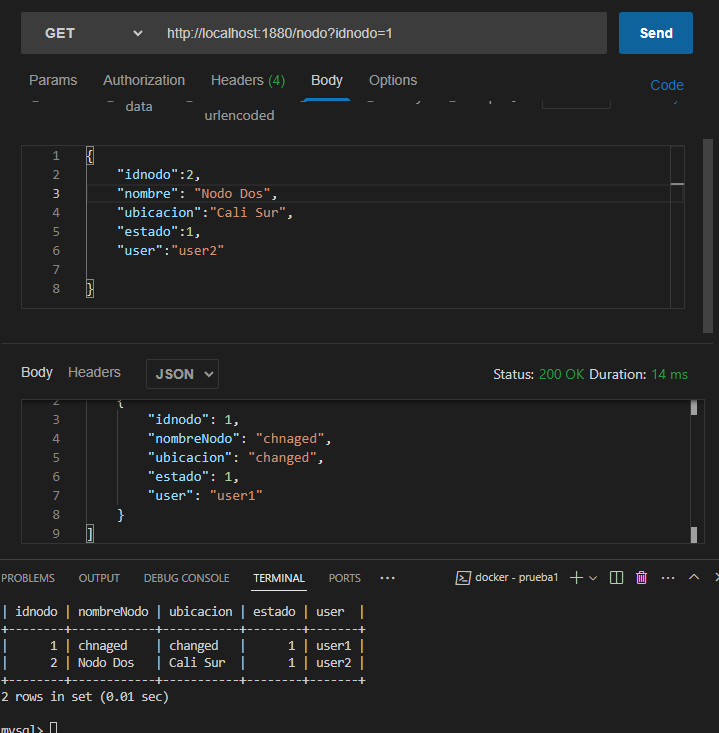
Se crea ruta y función



var idnodo = msg.payload.idnodo;

msg.topic = "select \* from nodos where idnodo=" + idnodo;

return msg;



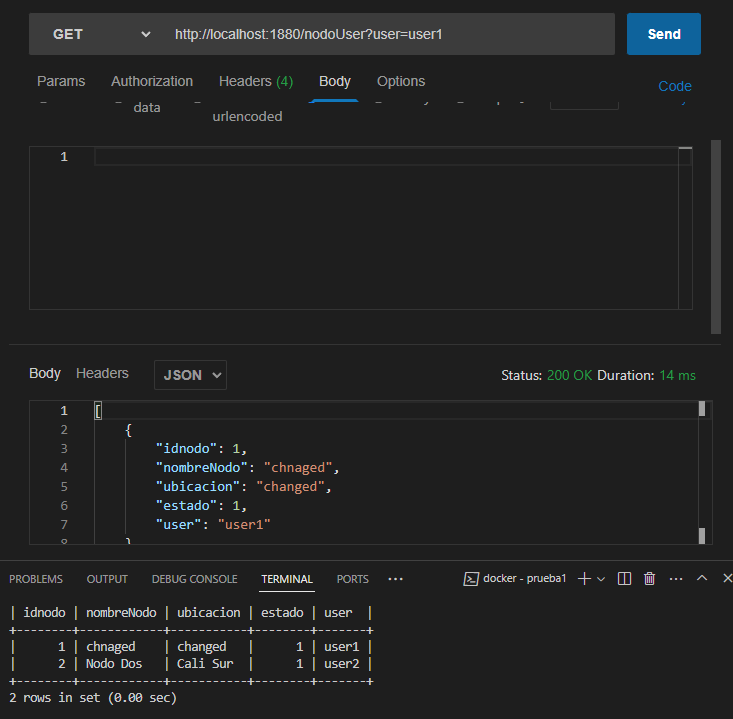
Se crea ruta y función

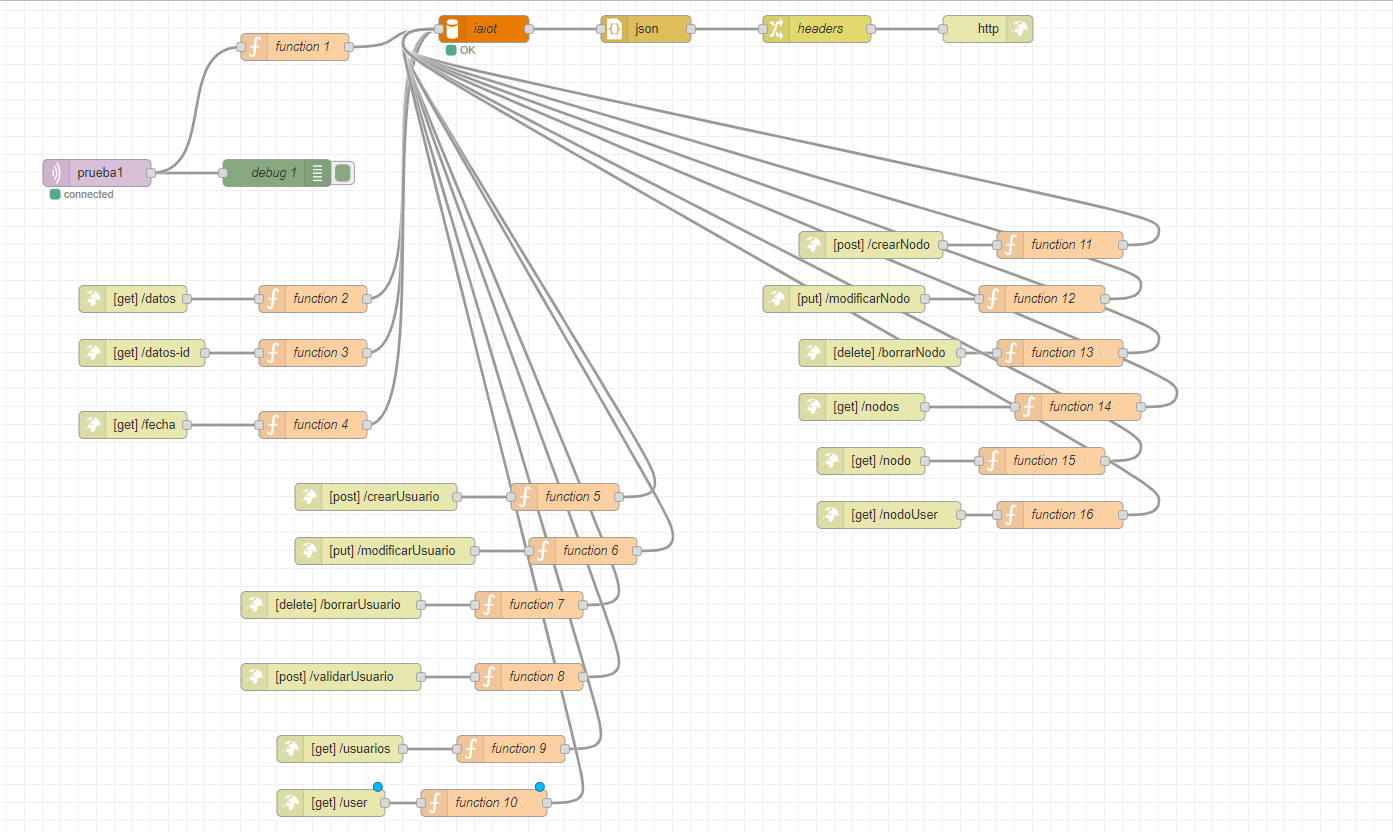


var user = msg.payload.user;

msg.topic = `select \* from nodos where user ="${user}"`;

return msg;





Este proyecto de node-red también se puede importar en el cual esta en un nodered.json

Para visualizar la información de una forma didáctica se crean archivos html y php para la parte backend , en este caso ya que se tenia Docker , para evitar instalar php y apache se creo un docker-compose.yml , en el cual hay una imagen que tiene php y apache juntos.

version: '3'

services:

  webserver:

    image: php:7.4-apache  # Utilizamos una imagen que incluye PHP y Apache

    ports:

      - "80:80"  # Mapeamos el puerto 80 del contenedor al puerto 80 del host

    volumes:

      - C:\path-to\php-files:/var/www/html

Pero como lso archivos php daban a la ruta de localhost se debe cambiar a la ip del pc , por ejemplo si en el archivo ingresar.php esta :

    // URL de la solicitud POST

    $url = 'http://localhost:1880/validarUsuario';

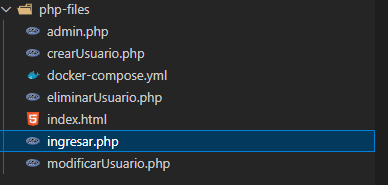
Tocaria cambiarlo por la ip del pc

    // URL de la solicitud POST

    $url = 'http://IP\_DE\_LOCAL:1880/validarUsuario';

Y así con los demás archivo que tenga la ruta .

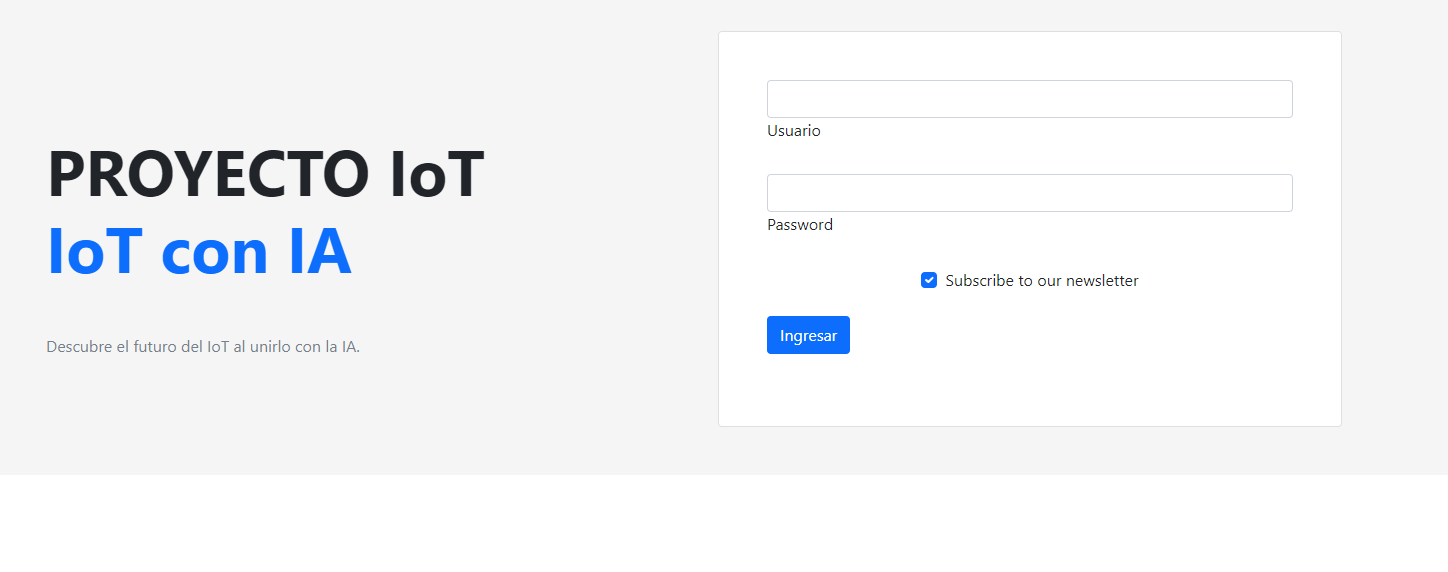
Teniendo ya todos los archivos ordenados en en el mimo directorio del Docker compose



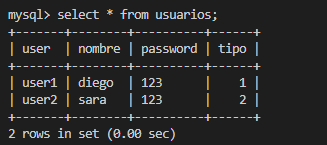
Se realiza “docker-compose up”



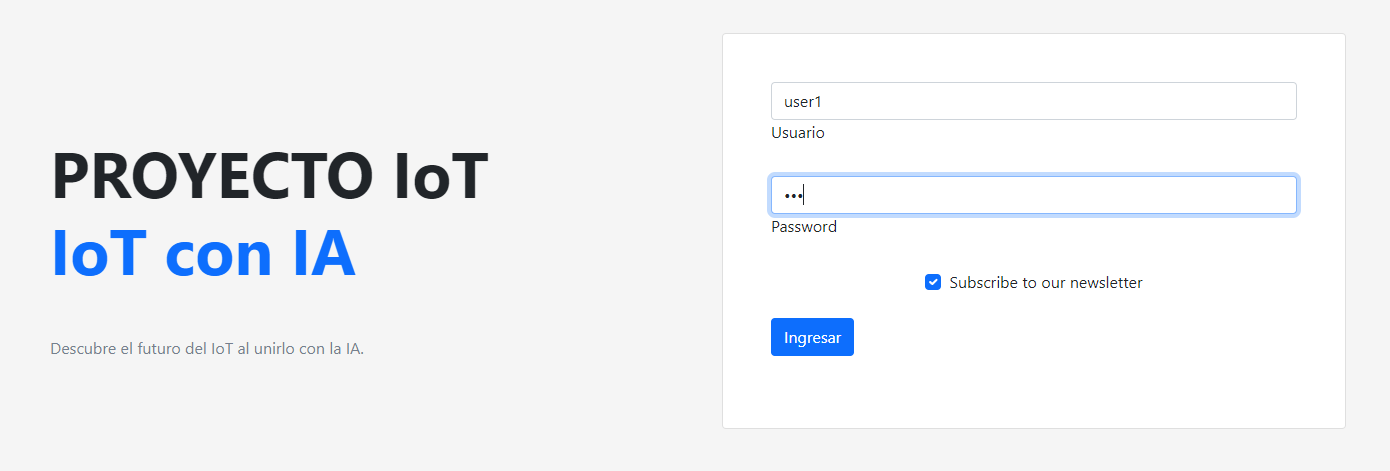
Y dirigiendo al localhost se visualiza:



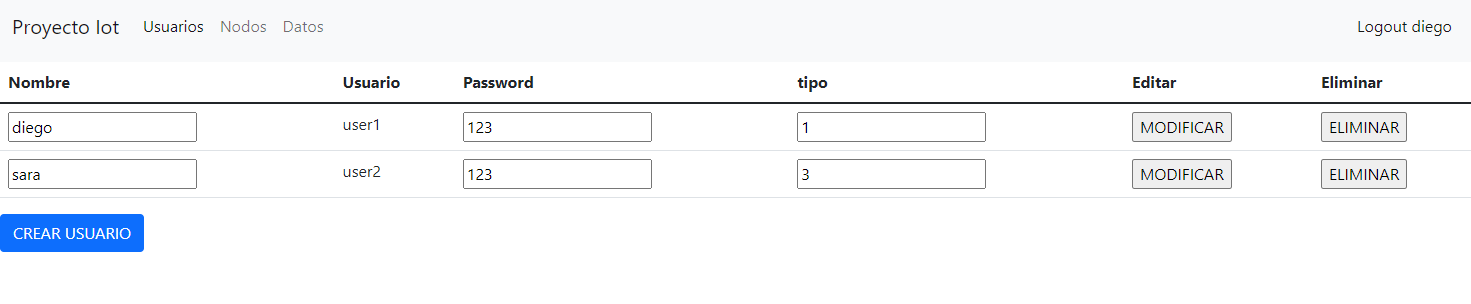
Recorda que en el mysql es :



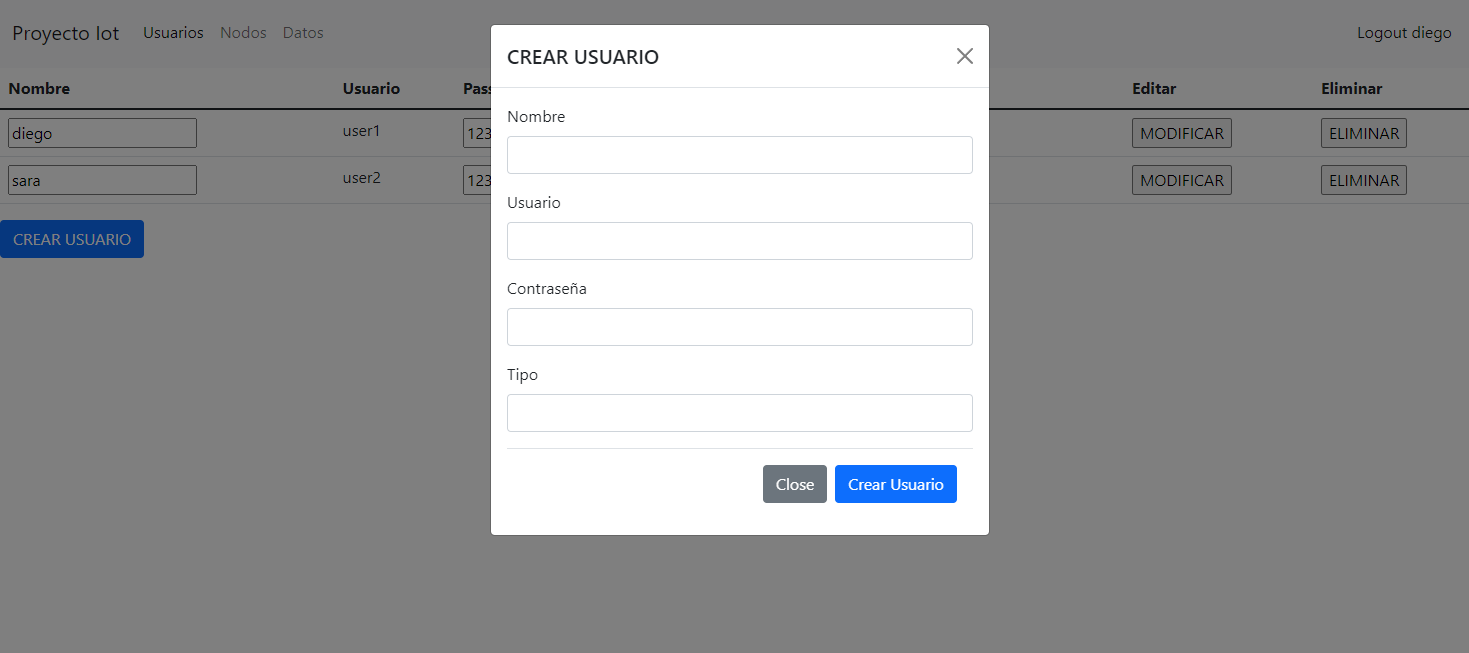
Por eso se intenta ingresar con el user1

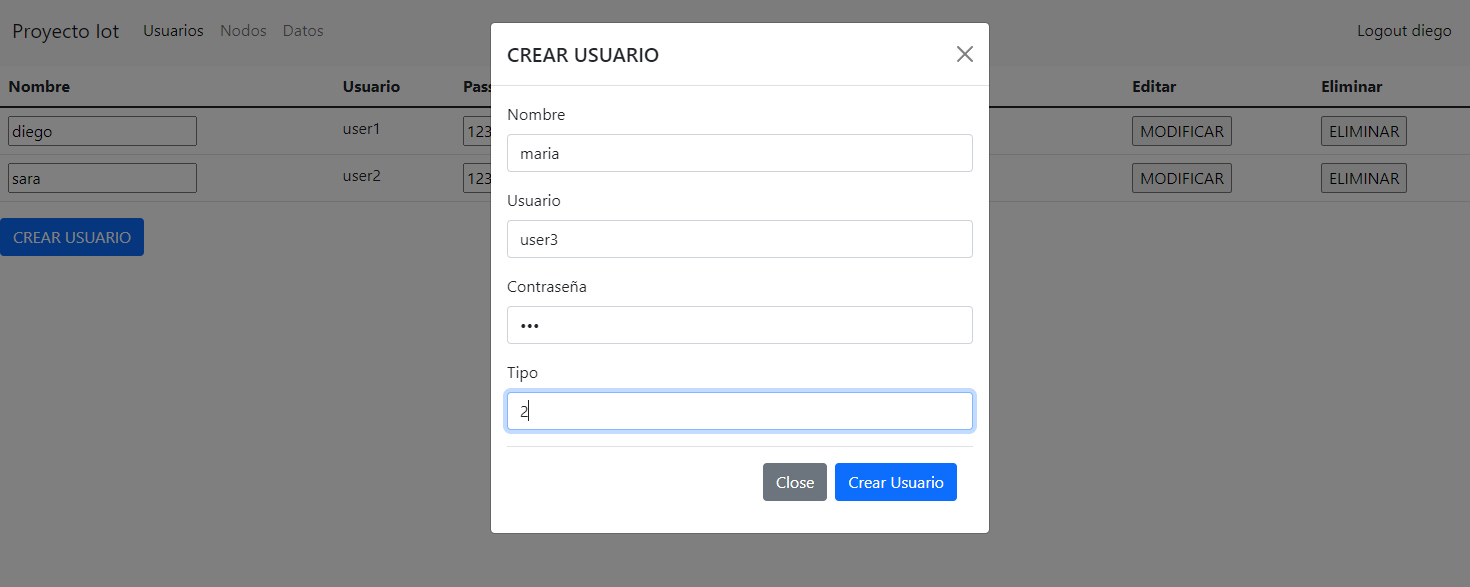


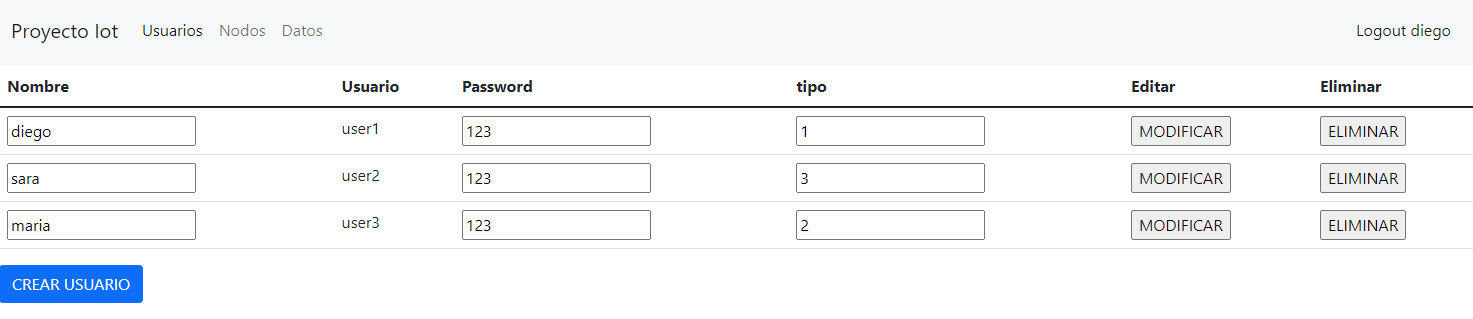
Como admin tendrá esta interfaz en el cual puede crear , modificar s y eliminar usuarios y nodos, y puede dar una visualización de todos los datos.



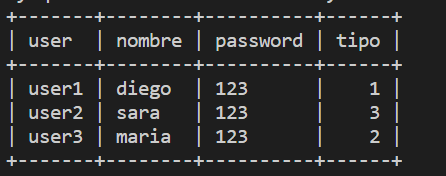
Dando crear usuario:



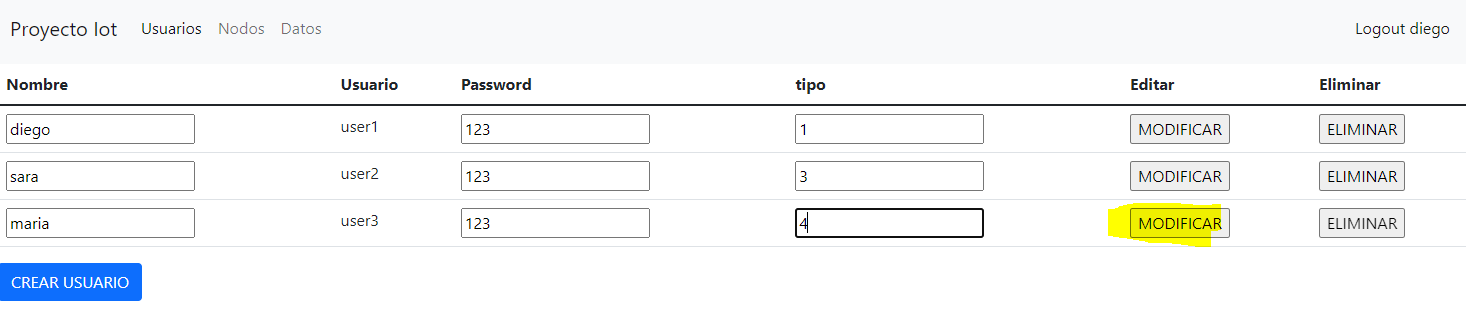


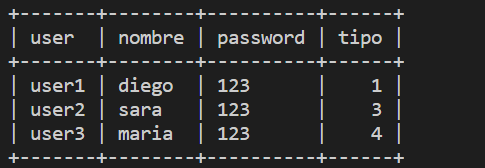


Se evidencia la creación del usuario en la base de datos

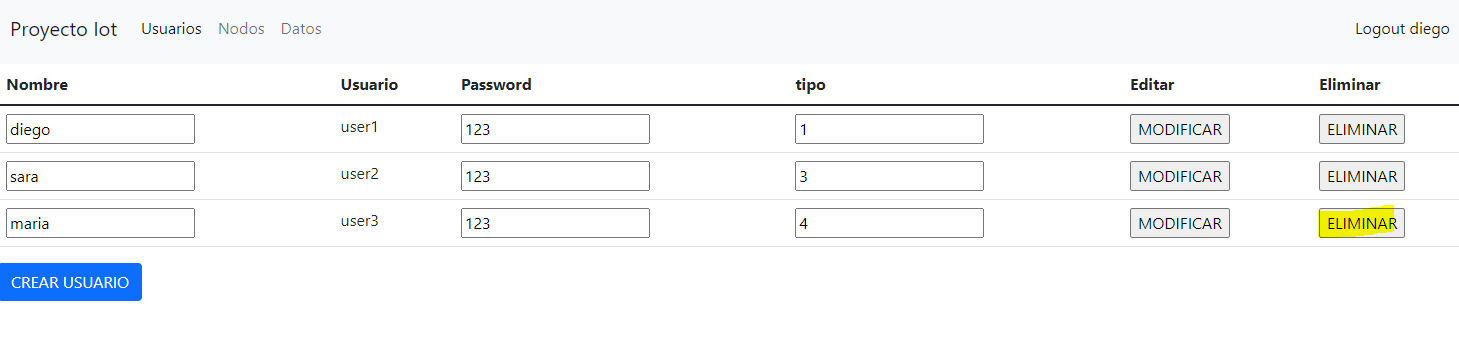


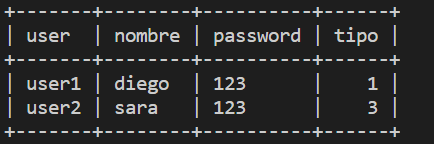
Si se quisiera modificar por ejemplo un usuario en este caso el user 3 maria con el tipo ahora de 4 :



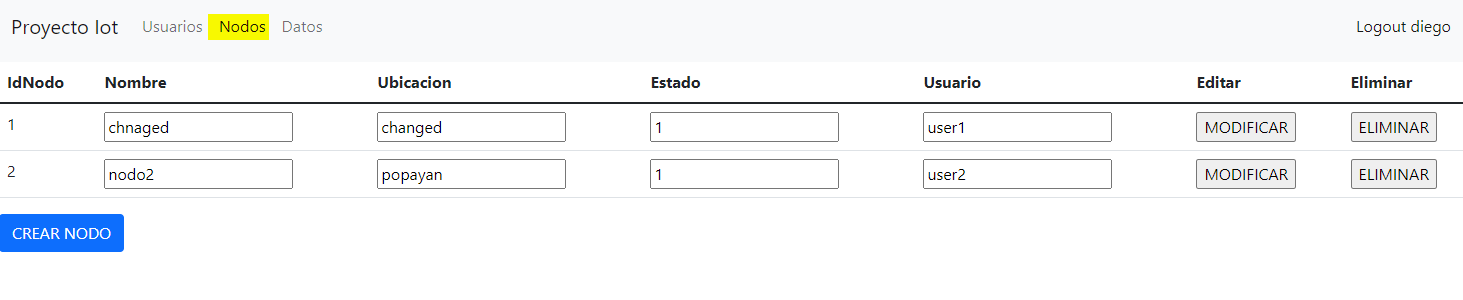


Y eliminándolo

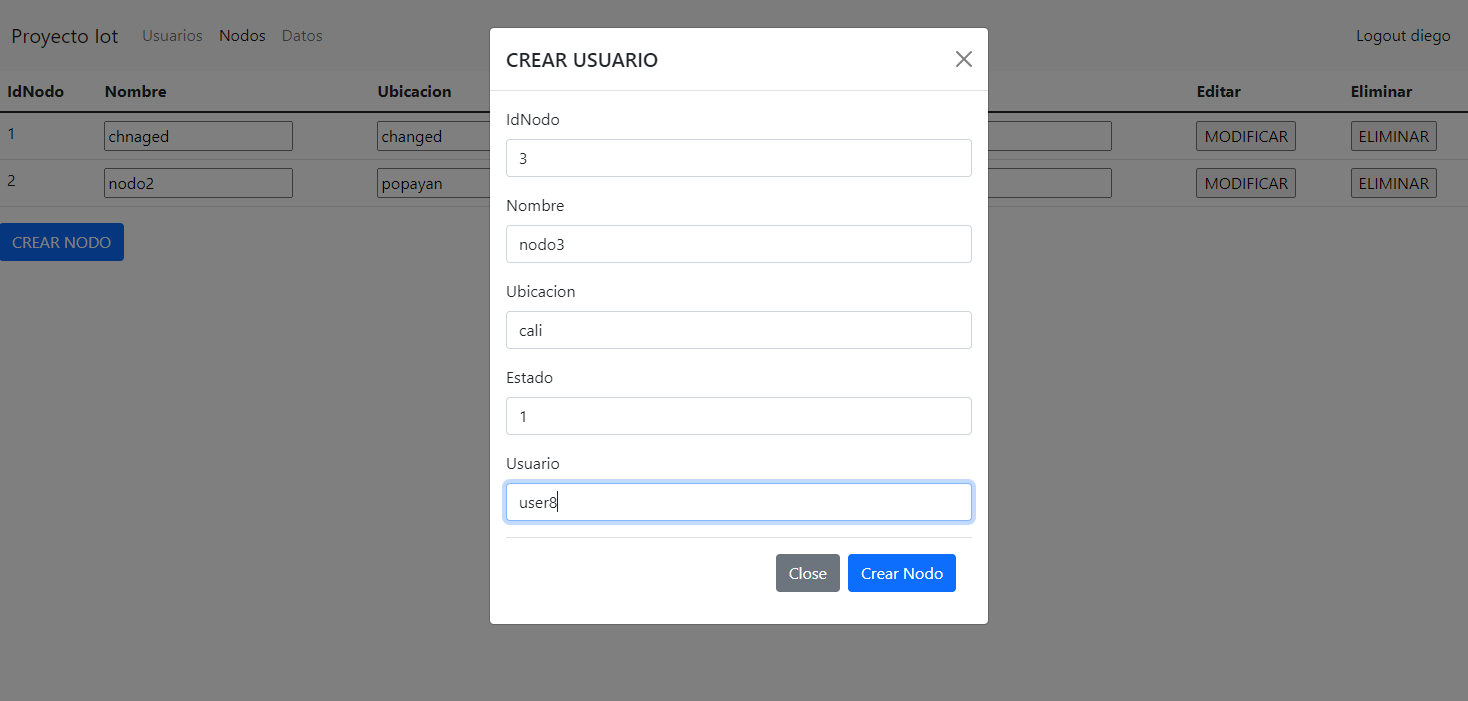


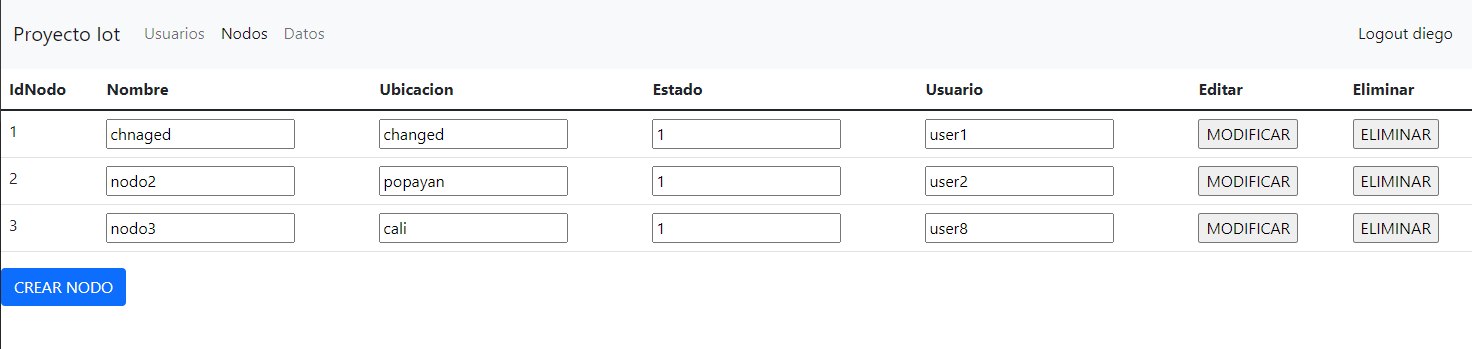


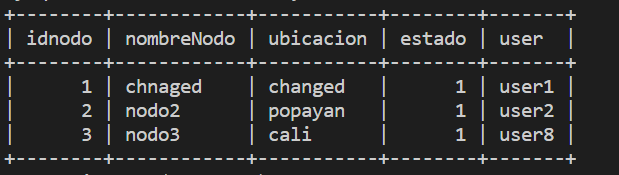
Lo mismo se puede hacer con los nodos:



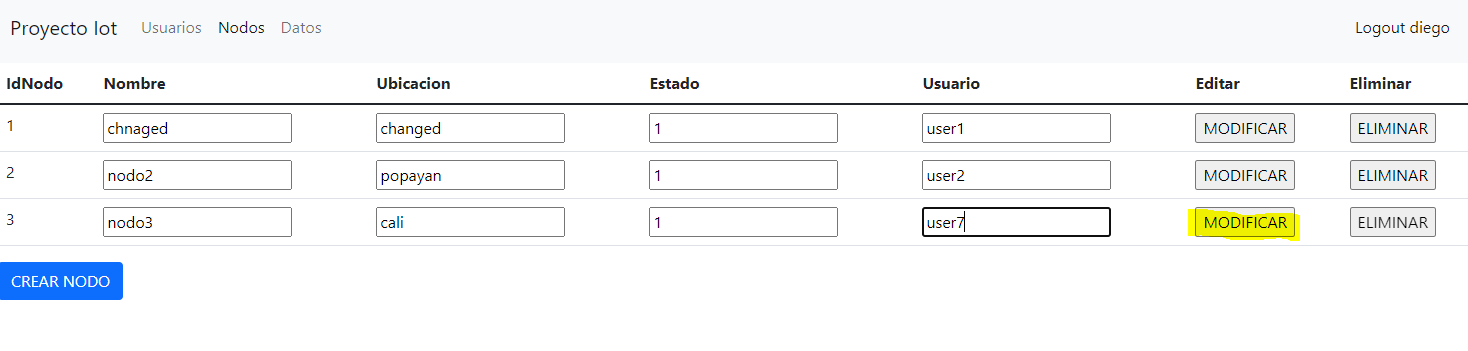
Se crea un nodo

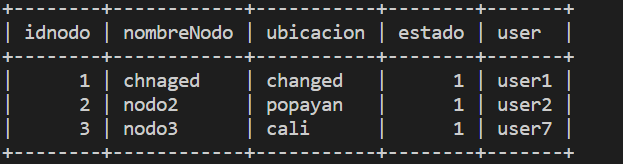




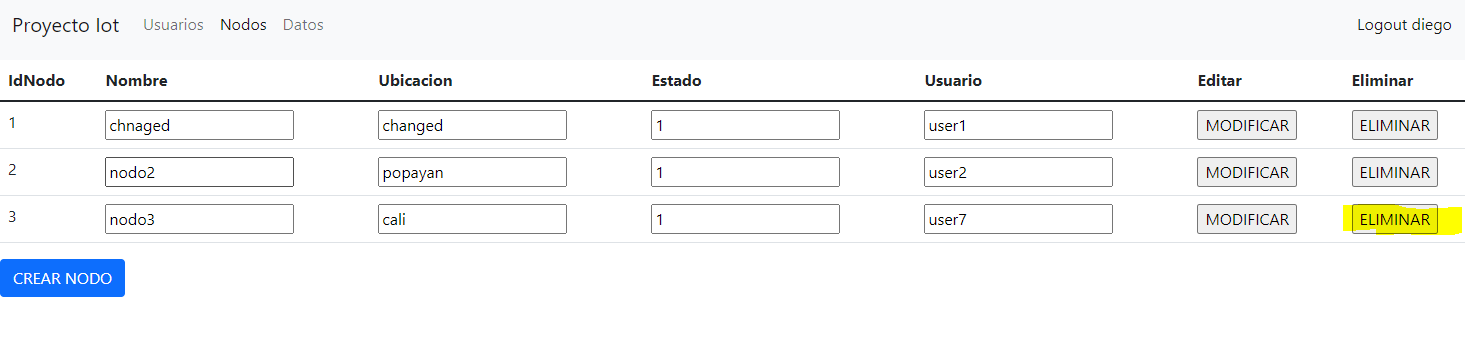
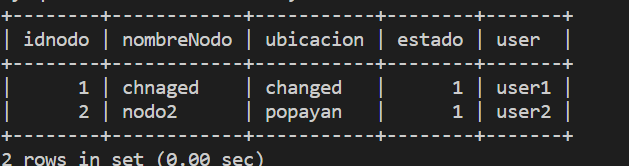


Ahora modificar el usuario del idnodo 3 :





Y eliminar

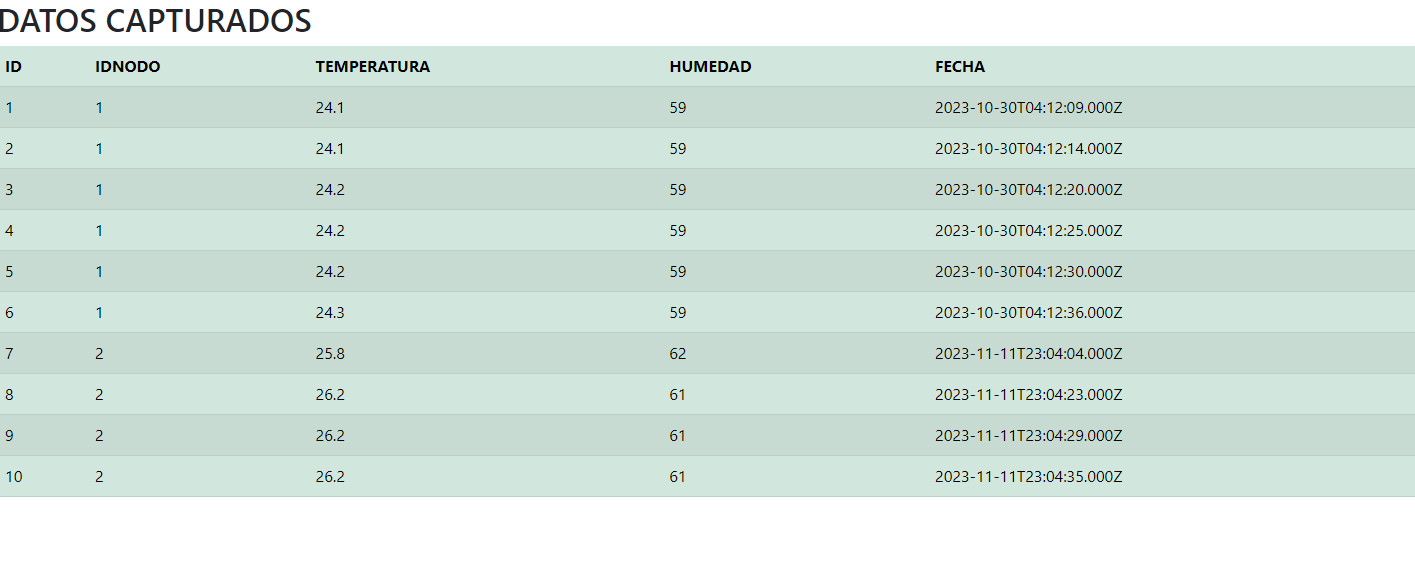
 

En Datos se puede observar la visualizacion de diferentes formas de todos los datos:

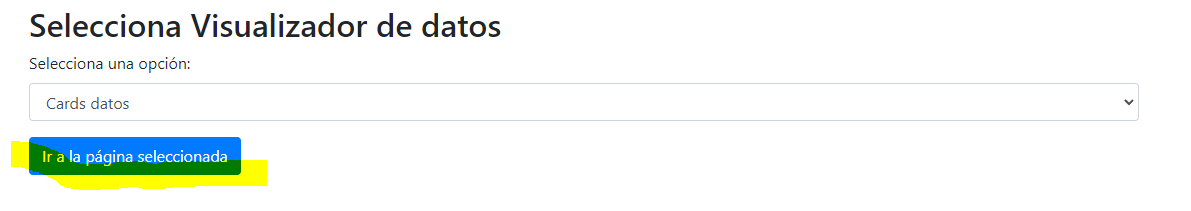


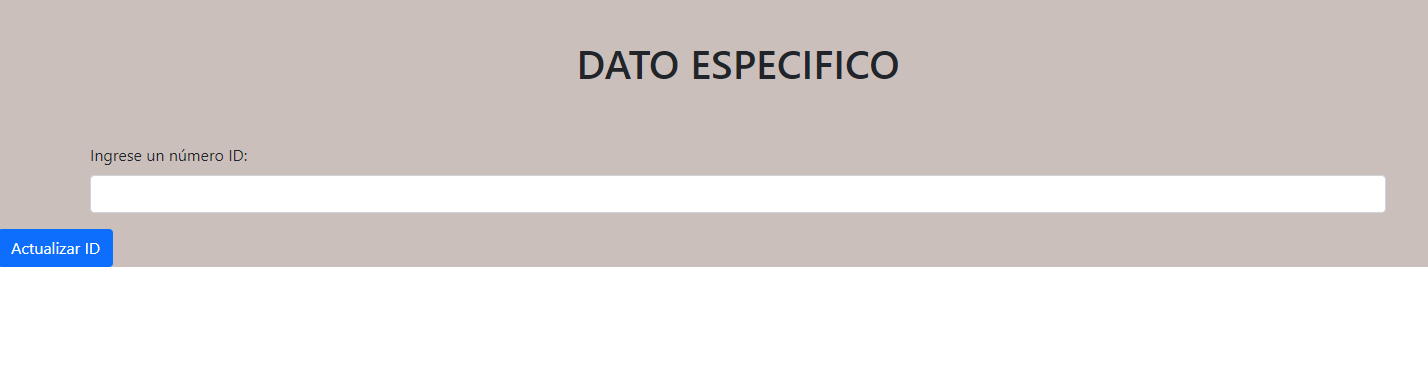


Se selecciona la forma de visualización y se le da click en “ir a la pagina seleccionada” , en este caso Tabla de datos:



En Cards datos se visualiza :



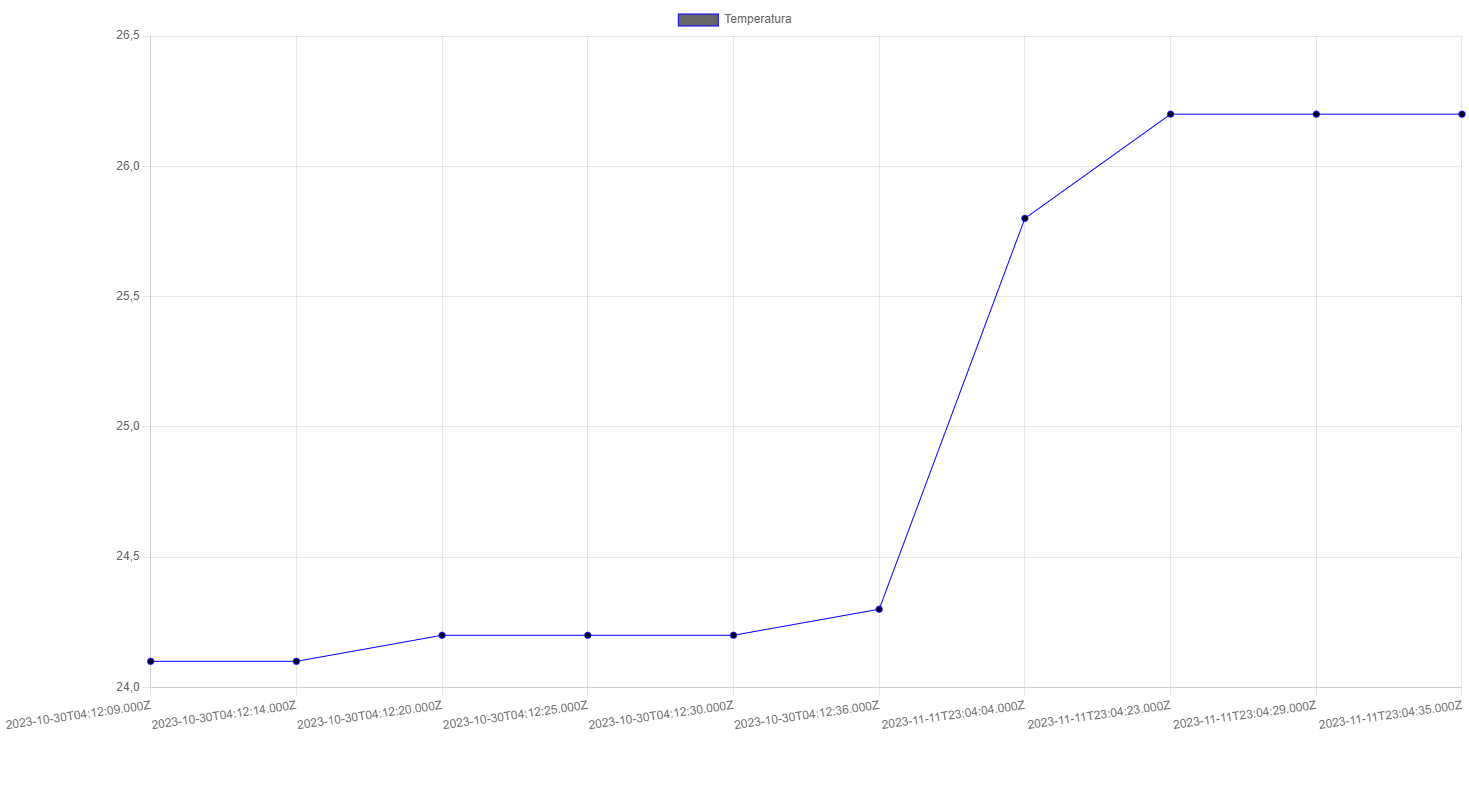


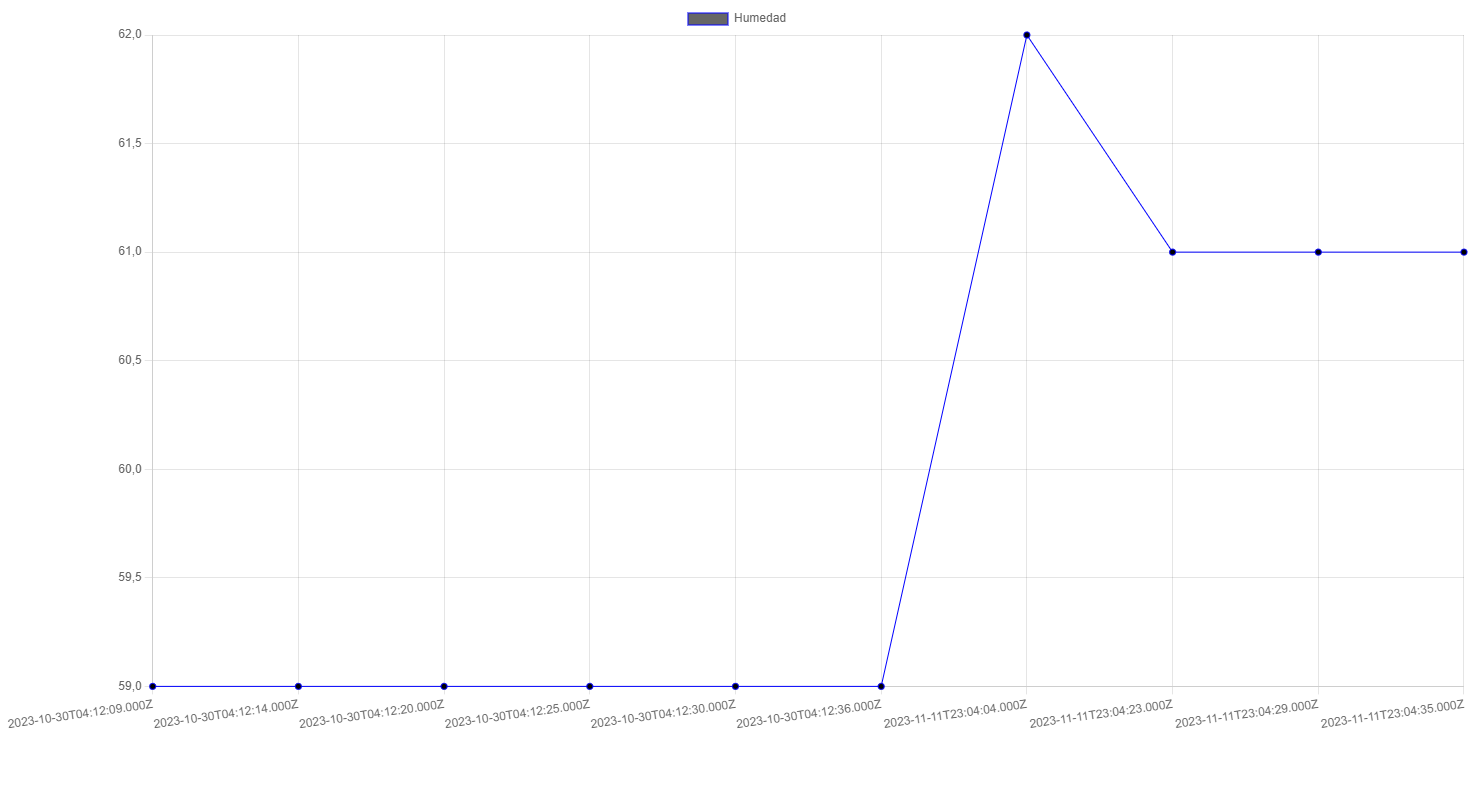
Se indica el id a observar , en este caso el 1



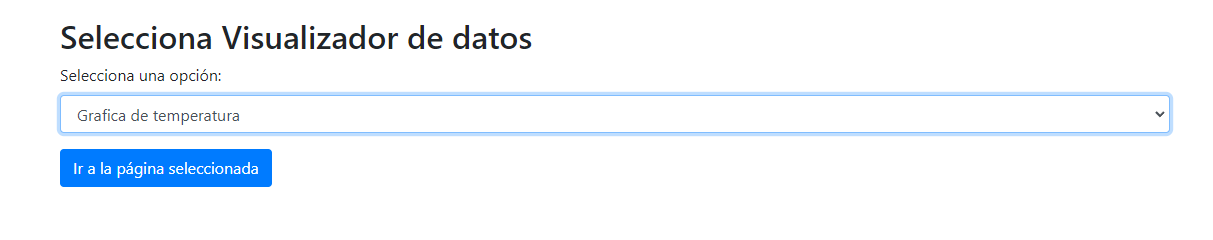
Se observa la fecha , temperatura y humedad en ese id especifico .

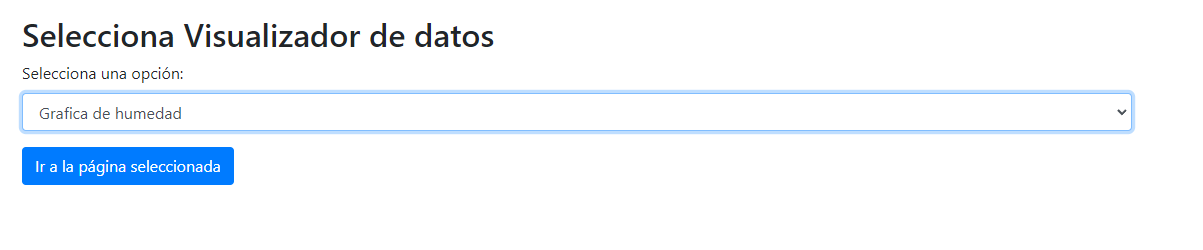
En ver grafica se visualiza todos los datos de temperatura y humedad respectivos



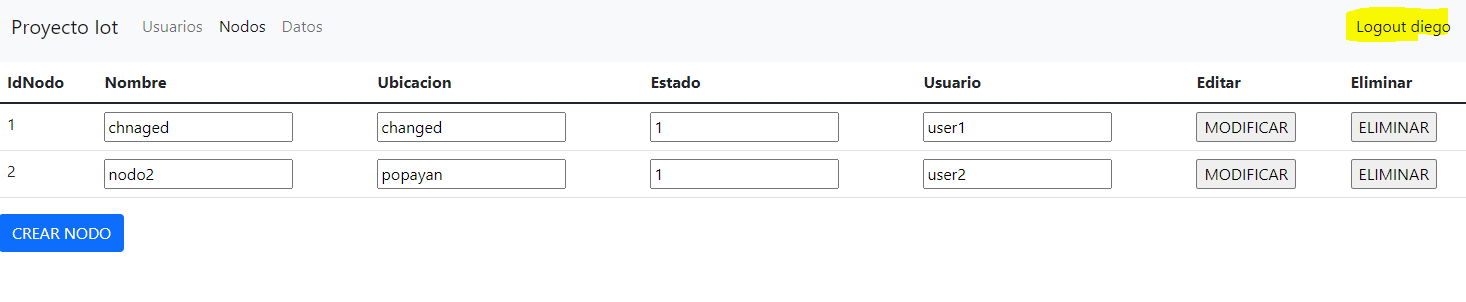


Estas graficas también se pueden observar en el seleccionador :



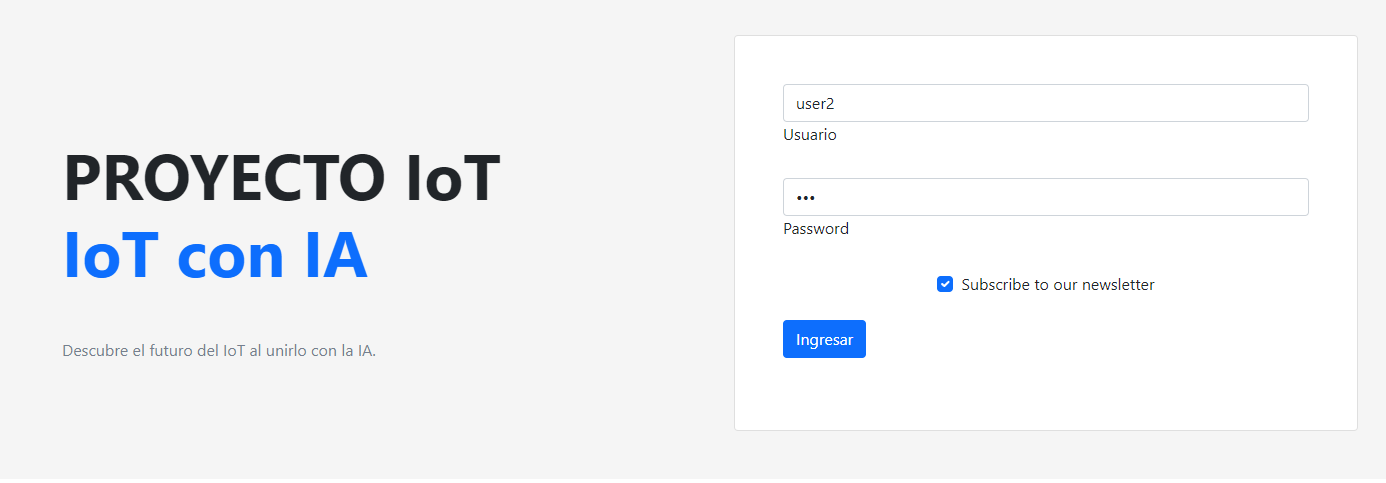


Para salir de la sesión con logout :

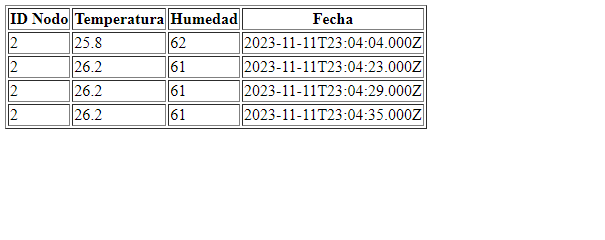


Ingresando con un usuario normal , no admin :

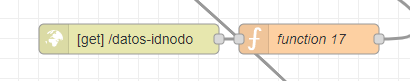
En este caso es el user2:



Observará los datos de su ID Nodo correspondiente en este caso su ID Nodo es el 2



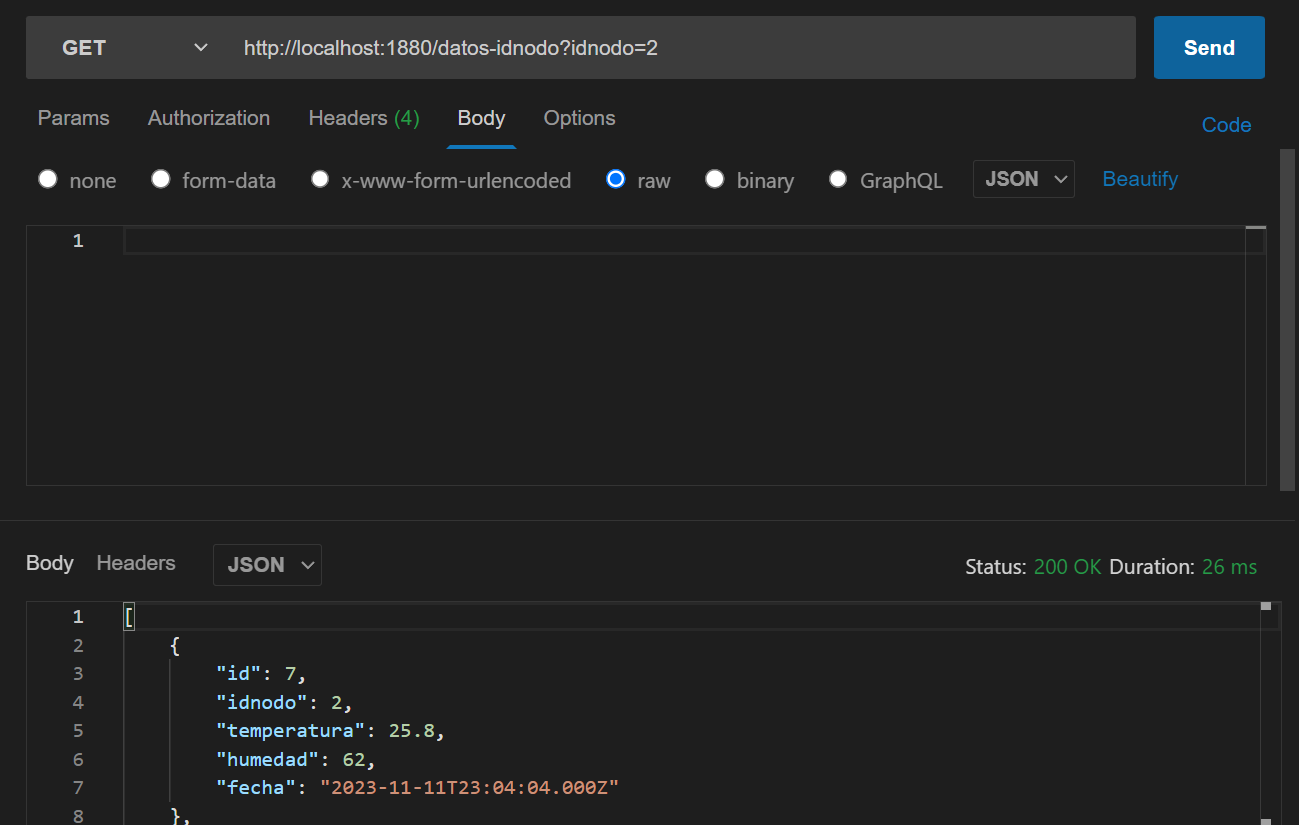
Para la realización de esta parte se añade en Node-red dos funciones:



var idnodo = msg.payload.idnodo;

msg.topic = "select \* from datos where idnodo =" + idnodo;

return msg;

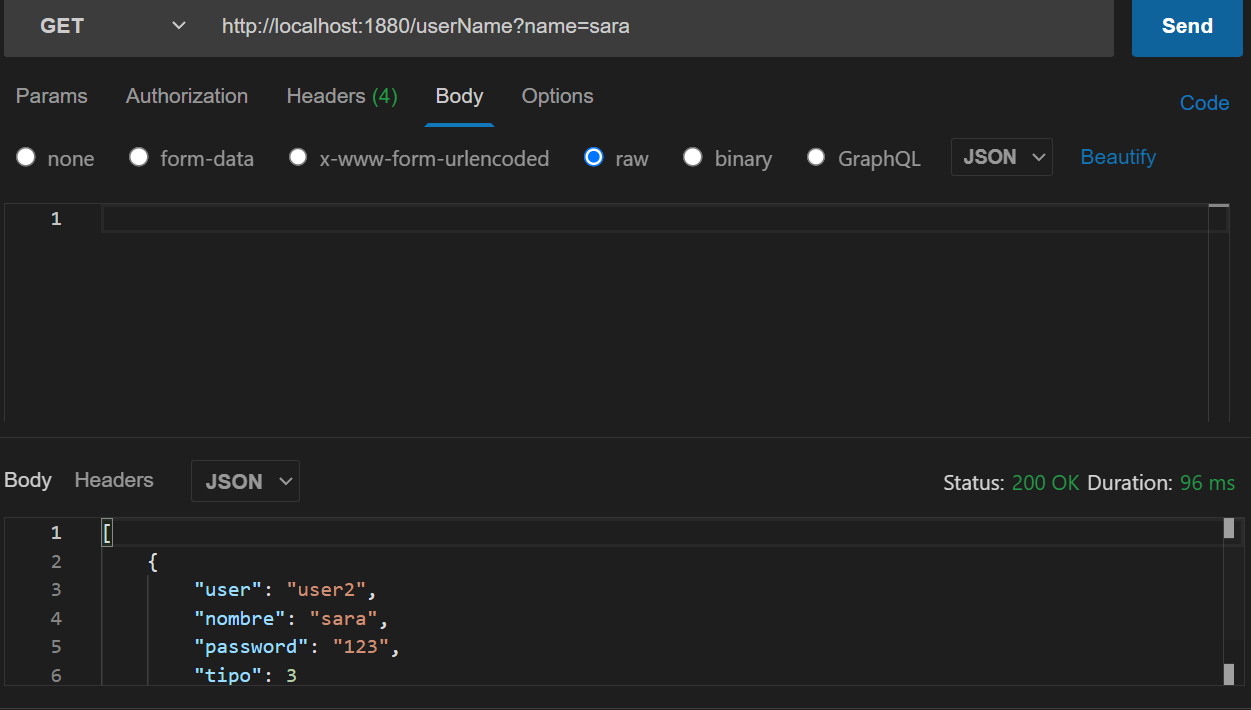




var name = msg.payload.name;

msg.topic = `select \* from usuarios where nombre="${name}"`;

return msg;



Codigo ESP32 Platformio con el idnodo 2 :

#include <Arduino.h>

#include <ArduinoJson.h>

#include <WiFi.h>

#include <PubSubClient.h>

//LIBRERIAS PARA DHT11 (TEMPERATURA Y HUMEDAD)

#include <Adafruit\_Sensor.h>

#include <DHT.h>

//LIBRERIAS PARA FECHA Y HORA

#include <WiFi.h>

#include <NTPClient.h>

#include <WiFiUdp.h>

//DEFINICION DE PINES DHT11

#define DHTPIN 14   // 4 = PIN D4

#define DHTTYPE    DHT11

DHT dht(DHTPIN, DHTTYPE);

// Define NTP Client to get time

WiFiUDP ntpUDP;

NTPClient timeClient(ntpUDP);

// Variables to save date and time

String formattedDate;

String dayStamp;

String timeStamp;

#define mqttUser ""

#define mqttPass ""

#define mqttPort 1883

const char\* ssid = "\*\*NAME\_WIFI\*";//name wifi

const char\* password = "\*PASSWORD\_WIFI\*"; // clave de wifi

char mqttBroker[] = "192.168.\*\*.\*"; //ip del servidor

char mqttClientId[] = "prueba1"; //cualquier nombre

char inTopic[] = "prueba1";//topcico a suscribirse

void callback(char\* topic, byte\* payload, unsigned int length) {

  Serial.print("Message arrived [");

  Serial.print(topic);

  Serial.print("] ");

  for (int i=0;i<length;i++) {

  Serial.print((char)payload[i]);

}

  Serial.println();

}

WiFiClient BClient;

PubSubClient client(BClient);

void reconnect() {

// Loop until we're reconnected

  while (!client.connected()) {

  Serial.print("Attempting MQTT connection...");

  // Attempt to connect

  if (client.connect("", mqttUser, mqttPass)) {

  Serial.println("connected");

  // Once connected, publish an announcement...

 // Once connected, publish an announcement...

  float h= dht.readHumidity();

  float t =dht.readTemperature();

  String variable;

  StaticJsonDocument<256> doc;

 /\*  doc["Fecha"] = dayStamp;

  doc["Hora"] = timeStamp; \*/

  doc["idnodo"] = 2;

  doc["temperatura"] = t;

  doc["humedad"] = h;

  serializeJson(doc, variable);

  int lon = variable.length()+1;

  Serial.println(variable);

  char datojson[lon];

  variable.toCharArray(datojson, lon);

  client.publish(inTopic,datojson);

  client.disconnect();

  delay(5000);

  // ... and resubscribe

  //client.subscribe("topic2");

  } else {

  Serial.print("failed, rc=");

  Serial.print(client.state());

  Serial.println(" try again in 5 seconds");

  // Wait 5 seconds before retrying

  delay(5000);

}

}

}

void setup\_wifi() {

  delay(10);

  // We start by connecting to a WiFi network

  Serial.println();

  Serial.print("Connecting to ");

  Serial.println(ssid);

  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());

  // Initialize a NTPClient to get time

  timeClient.begin();

  // Set offset time in seconds to adjust for your timezone, for example:

  // COLOMBIA -5 , entonces -5\*3600 ->  -18000

  timeClient.setTimeOffset(-18000); //Thailand +7 = 25200

}

void setup()

{

  Serial.begin(9600); //Serial connection

  setup\_wifi(); //WiFi connection

  client.setServer(mqttBroker, mqttPort );

  client.setCallback( callback );

  Serial.println("Setup done");

  delay(1500);

}

void loop(){

    while(!timeClient.update()) {

    timeClient.forceUpdate();

    }

    // The formattedDate comes with the following format:

    // 2018-05-28T16:00:13Z

    // We need to extract date and time

    formattedDate = timeClient.getFormattedDate();

    // Extract date

    int splitT = formattedDate.indexOf("T");

    dayStamp = formattedDate.substring(0, splitT);

    //Serial.print("DATE: ");

    //Serial.println(dayStamp);

    // Extract time

    timeStamp = formattedDate.substring(splitT+1, formattedDate.length()-1);

    if (!client.connected()) {

    reconnect();

    }

    client.loop();

}