

Parcial Final IoT

Diego Iván Perea Montealegre (2185751) diego.perea@uao.edu.co

Facultad de Ingeniería, Universidad Autónoma de Occidente

Cali, Valle del Cauca

Siguiendo con el contexto de la problemática identificada que es el mal control de temperatura y humedad relativa en el almacenamiento de los productos de frutas y hortalizas; En donde el objeto de la aplicación son las frutas y hortalizas, y el objeto será una caja inteligente compuesto por sensores de temperatura y humedad relativa que permita recolectar información de la condición de almacenamiento.

La información de las condiciones de la fruta s visualizarán en una aplicación Web que debe entregar los datos recibidos de las frutas u hortalizas, en el que darán avisos de alerta según la personalización del usuario.

a) Temperatura (baja, optima, alta)

b) Humedad (baja, optima, alta)

*Se tomará como ejemplo el limón

Temperatura °C	Mensaje	Humedad relativa %	Mensaje
$T < 10$	Temperatura Baja	$H < 85$	Humedad Baja
$10 \leq T \leq 13$	Temperatura Optima	$85 \leq H \leq 90$	Humedad Optima
$T > 13$	Temperatura Alta	$H > 90$	Humedad Alta

Tabla 1 Visualización de mensaje según datos adquiridos

El mensaje y los datos adquiridos serán visualizados en una pagina web para que el usuario tenga la información clara y detallada de lo que esta pasando en el almacenamiento de este , este almacenamiento es dado por cajas , en cada caja estará un dispositivo en el que le enviará los datos adquiridos de temperatura y humedad al servidor para así visualizarlos como se menciona anteriormente en una página web.

Los datos que visualizará el usuario en la web son : nodo (Representa tipo de alimento) , la temperatura, humedad , fecha y hora.

La solución constará de los siguientes componentes:

-Nodo: Encargado de censer la variable y enviarlas al servidor

-Servidor: Encargado de recibir los datos, procesarlos y almacenarlos.

-Aplicación web: Contendrá toda la interfaz visual que le mostrará todos los datos procesados al usuario.

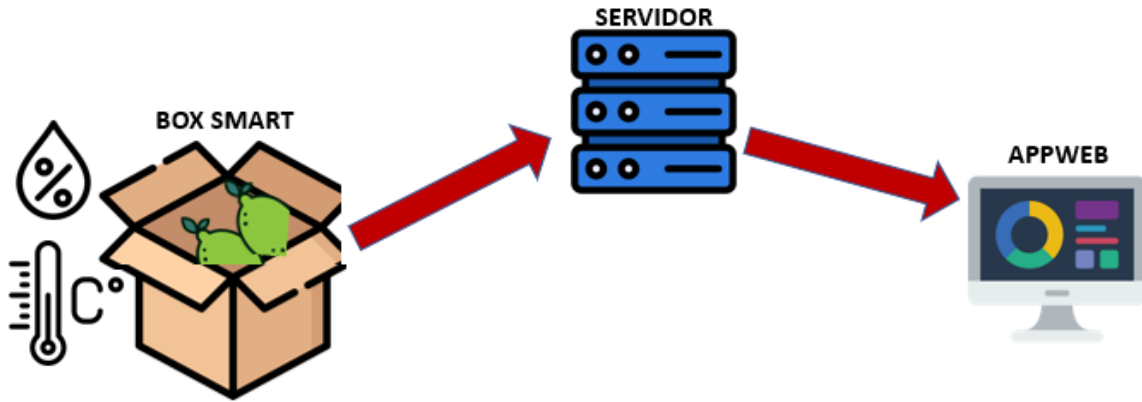


Figura1 Comportamiento solución IoT

Las variables para capturar son temperatura y humedad relativa en donde las medidas son de Temperatura [°C] y Humedad [%].

En este caso se utilizara la fruta (limones) como almacenamiento en la “box smart”, en el cual medirá su temperatura y humedad, este envío de datos se realizará a través del esp32 con el protocolo MQTT debido a que se realizará de forma local y los datos de las fechas y horas no se alteran con este protocolo, en el almacenamiento de los datos esta MYSQL con su respectiva base de datos y tablas.

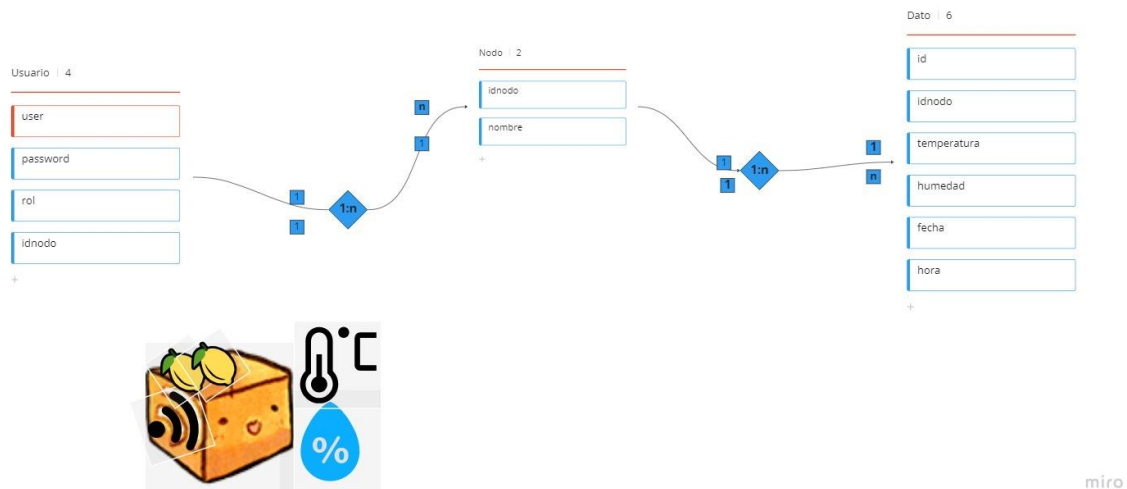


Figura 2 Almacenamiento de tablas

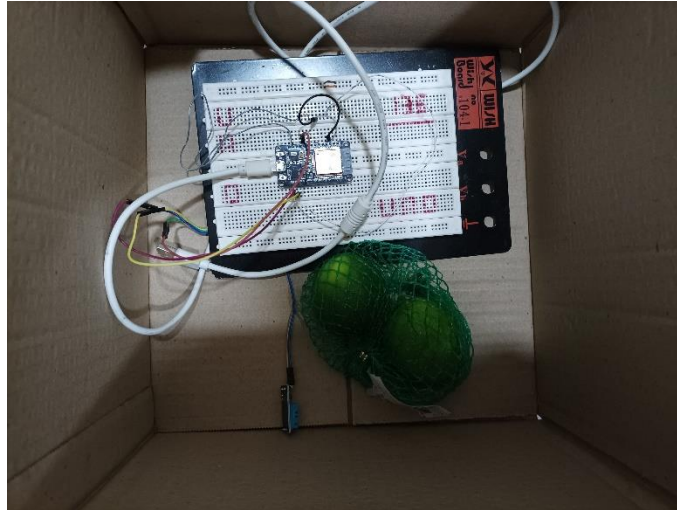


Fig 1. Almacenamiento de limon con “box smart”

El aplicativo web tendrá de usabilidad en la creación de dos diferentes usuario como lo es el cliente y el administrador, en el cual el cliente tendrá la opción de obtener todos lo datos y mensaje dado según el nodo seleccionado y el usuario administrador podrá crear usuario, borrar usuario, modificar usuario, crear nodos, borrar nodos y obtener todos los datos.

Para el envío de datos del esp32 a localhost , se activo el mqtt , en el cual evidencia el recibimiento de los datos y el almacenamiento de estos en la tabla de dato.

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  JUPYTER
PS C:\Users\User\Desktop\proyectos arduino\Parcial iot final diego> node .\src\indexmqtt.js
{
  idnodo: 1,
  temperatura: 26.20000076,
  humedad: 63,
  fecha: '2022-11-10',
  hora: '18:18:34'
}
Conexion correcta.
datos almacenados
{
  idnodo: 1,
```

Fig subida de datos desde el ESP32

```
MariaDB [proyecto]> select * from dato;
```

id	idnodo	temperatura	humedad	fecha	hora
1	2	40	58	jueves 10 nov 2022	9:46 am
2	1	40	58	jueves 10 nov 2022	9:46 am
3	2	30	58	jueves 10 nov 2022	9:46 am
4	1	26.2	63	2022-11-10	18:18:34
5	1	26.2	63	2022-11-10	18:18:34
6	1	26.2	63	2022-11-10	18:19:28
7	1	26.7	63	2022-11-10	18:50:52
8	1	26.7	63	2022-11-10	18:50:52
9	1	26.7	63	2022-11-10	18:50:52
10	1	26.7	63	2022-11-10	18:50:52
11	1	26.7	63	2022-11-10	18:50:52
12	1	26.7	63	2022-11-10	18:50:52
13	1	26.7	63	2022-11-10	18:50:52
14	1	26.7	63	2022-11-10	18:50:52
15	1	26.7	63	2022-11-10	18:50:52
16	1	26.7	63	2022-11-10	18:50:52
17	2	30	58	viernes 11 nov 2022	6:52 pm

Fig 2. almacenamiento de datos en mysql

Ahora mirems la interfaz de la web y como esta realiza el proceso de datos y su almacenamiento en cada una de las tablas realizadas , cabe recalcar que la descripción en la creación de las tablas es la siguiente:

```
MariaDB [proyecto]> describe dato;
```

Field	Type	Null	Key	Default	Extra
id	int(11)	NO	PRI	NULL	auto_increment
idnodo	int(11)	YES		NULL	
temperatura	float	YES		NULL	
humedad	float	YES		NULL	
fecha	varchar(25)	YES		NULL	
hora	varchar(25)	YES		NULL	

6 rows in set (0.089 sec)

```
MariaDB [proyecto]>
```

Fig 3. Tabla de dato

```
MariaDB [proyecto]> describe nodo;
```

Field	Type	Null	Key	Default	Extra
idnodo	int(11)	NO	PRI	NULL	
nombre	varchar(20)	YES		NULL	

2 rows in set (0.026 sec)

```
MariaDB [proyecto]>
```

Fig 4. Tabla de nodo

```
MariaDB [proyecto]> describe usuario;
```

Field	Type	Null	Key	Default	Extra
user	varchar(30)	NO	PRI	NULL	
password	varchar(10)	YES		NULL	
rol	int(11)	YES		NULL	
idnodo	int(11)	YES		NULL	

4 rows in set (0.026 sec)

Fig 5. Tabla de usuario

Estando en la pagina Web la entrada principal se da en el login en el cual se debe logear con una cuenta y si no es así se debe de registrar para tener una cuenta como de cliente o administrador

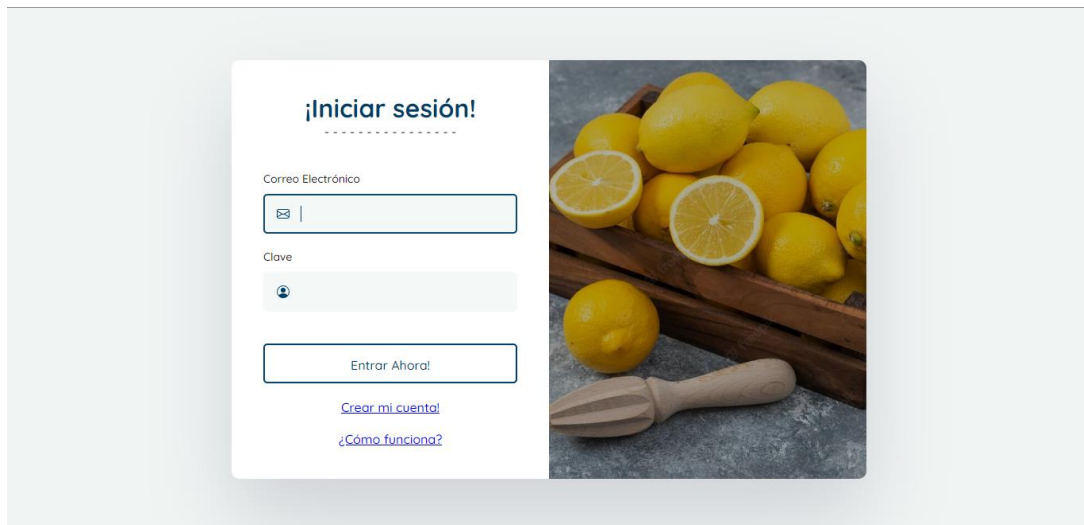


Fig 6. Página principal de la web

En esta vista de página principal hay un link en donde explica el cómo funciona la misma página para los clientes que es el e mercado objetivo , debido a que los administradores serán escogido o seleccionados .En esta misma se visualiza los diferentes pasos a seguir que le cliente debe de acoger , indicándole en texto e imagen de la mejor forma.



Fig 7. Página de cómo funciona la web para los clientes

Se activa el servidor para que se pueda realizar todos los procesos REST necesarios para el funcionamiento de la pagina y el procesamiento de esta.

```
PS C:\Users\User\Desktop\proyectos_arduino\Parcial_iot_final_diego> node .\src\indexrest.js
Servidor funcionando
█
```

Fig 8. Funcionamiento del servidor

Entendido el funcionamiento de la web y seguido los pasos correctamente , es eventualmnte que el cliente realice el procesos de crear cunta por lo que dirreciona a la pagina principal con el link de volver y da click en el link de crear cuenta en donde se visualiza la pagina de crear cuenta , por lo que debe suministrar un correo , una contraseña , el rol en donde debe ser 1 para ser cliente ya especificado en los pasos de la pagina de como funciona la web , y un numero de nodo

Fig 9. Creación de cuenta de cliente y administrador

Ya creado la cuenta se debe redireccionar al login con volver e introducir su correo y contraseña ,por lo que el tipo de usuario cliente le saldrá el siguiente apartado en el servidor y en la web.

Por lo que con REST se dbe de realizar es un get en la tabla de usuario , y esto es gracias a las rutas .js que realizaran los procesos de GET , POST , PUT y DELETE según correspondan en la codificación realizada hecha en estas rutas.

```
PS C:\Users\User\Desktop\proyectos_arduino\Parcial_iot_final_diego> node .\src\indexrest.js
Servidor funcionando
Conexion correcta.
{ usuario: 'ejemplo1@gmail.com', password: '123', rol: 1, idnodo: 1 }
GET /usuarios/ejemplo1@gmail.com 200 196.657 ms - 70
█
```

Fig 10. GET para logear al cliente nombrado ejemplo1@gmail.com

Este seguimiento de la pagina con las rutas esta desarrollado en php para el procesamiento de los datos y sus conexiones hacia REST .

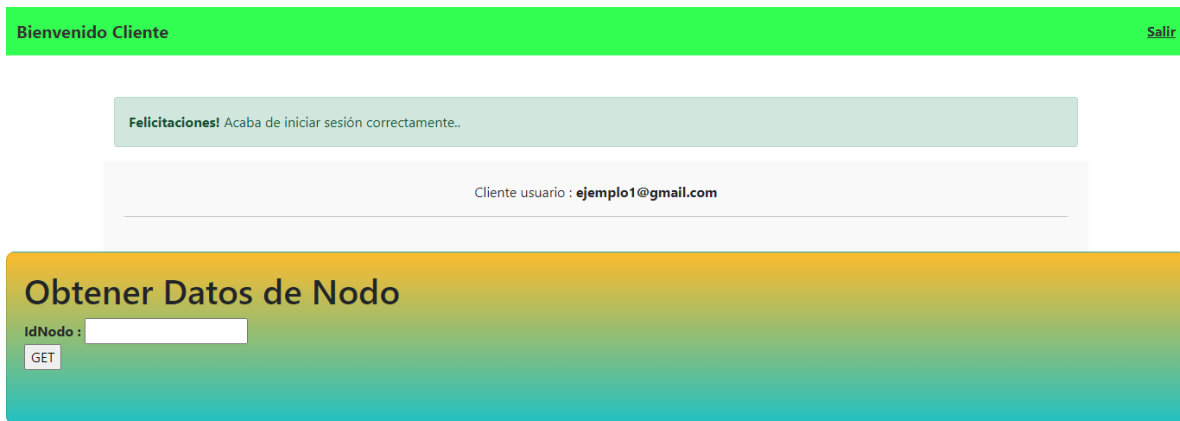


Fig 11. Pagina de cliente al momento de iniciar sesión

En la página de cliente podrá obtener los datos de los nodos seleccionados en la subida de datos realizada por el ESP32 , se realiza dándole click al botón GET en el que lo direccionará a otra pagina para que visualice los datos, en este caso de introducirá el numero '1' en el idNodo.

```
fecha: '2022-11-10',
hora: '18:50:52'
}
{
  idnodo: 1,
  temperatura: 26.7,
  humedad: 63,
  fecha: '2022-11-10',
  hora: '18:50:52'
}
GET /datos/1 200 220.003 ms - 1182
```

Fig 12. Get de los datos con idNodo 1 en el usuario cliente

El usuario cliente visualiza una tabla con todos los datos del idNodo 1 , en donde se observa la temperatura , humedad , fecha , hora y unos mensajes en la tabla advirtiéndole en que tipo de estado está el almacenamiento de la fruta entre los diferentes tipos se puede ver tabla 1; Por lo que en el idNodo 1 la temperatura era alta y una humedad Baja en el almacenamiento de la fruta limón.

idnodo	temperatura	humedad	fecha	hora	Mensaje T	Mensaje H
1	40	58	jueves 10 nov 2022	9:46 am	Temperatura Alta	Humedad Baja
1	26.2	63	2022-11-10	18:18:34	Temperatura Alta	Humedad Baja
1	26.2	63	2022-11-10	18:18:34	Temperatura Alta	Humedad Baja
1	26.2	63	2022-11-10	18:19:28	Temperatura Alta	Humedad Baja
1	26.7	63	2022-11-10	18:50:52	Temperatura Alta	Humedad Baja
1	26.7	63	2022-11-10	18:50:52	Temperatura Alta	Humedad Baja
1	26.7	63	2022-11-10	18:50:52	Temperatura Alta	Humedad Baja
1	26.7	63	2022-11-10	18:50:52	Temperatura Alta	Humedad Baja
1	26.7	63	2022-11-10	18:50:52	Temperatura Alta	Humedad Baja
1	26.7	63	2022-11-10	18:50:52	Temperatura Alta	Humedad Baja
1	26.7	63	2022-11-10	18:50:52	Temperatura Alta	Humedad Baja
1	26.7	63	2022-11-10	18:50:52	Temperatura Alta	Humedad Baja
1	26.7	63	2022-11-10	18:50:52	Temperatura Alta	Humedad Baja
1	26.7	63	2022-11-10	18:50:52	Temperatura Alta	Humedad Baja
1	26.7	63	2022-11-10	18:50:52	Temperatura Alta	Humedad Baja
1	26.7	63	2022-11-10	18:50:52	Temperatura Alta	Humedad Baja

[Volver](#)

Fig 13. Visualización de los datos en el respectivo nodo 1 del cliente

idnodo	temperatura	humedad	fecha	hora	Mensaje T	Mensaje H
2	40	58	jueves 10 nov 2022	9:46 am	Temperatura Alta	Humedad Baja
2	30	58	jueves 10 nov 2022	9:46 am	Temperatura Alta	Humedad Baja
2	30	58	viernes 11 nov 2022	6:52 pm	Temperatura Alta	Humedad Baja

[Volver](#)

Fig 14. Visualización de los datos en el respectivo nodo 2 del cliente

Además dándole click en salir, este sale de la sesión y vuelva a la página principal de login.

Iniciando sesión como administrador la pagina de inicio se visualizará diferentes recuadros para realizar las diferentes acciones REST para ejecutar con los datos obtenidos desde el ESP32 .

```
{ usuario: 'admin1@gmail.com', password: '123', rol: 2, idnodo: 1 }
GET /usuarios/admin1@gmail.com 200 28.015 ms - 68
```

Fig 15 . get de usuario administrador

Bienvenido Admin Salir

Felicitaciones! Acaba de iniciar sesión correctamente.

Admin usuario : admin1@gmail.com

Crear usuario
Usuario :
Password :
Rol :
Nodo :

Borrar usuario
Usuario :

Modificar en Usuario
Usuario :
Password nueva:
Rol nuevo:
Nodo nuevo:

Crear Nodos
IdNodo :
Nombre :

Borrar Nodos
IdNodo :

Obtener Datos TODOS

Fig 16. Página de usuario administrador

En el panel de crear usuario el administrador podrá crear un usuario el que se esté dando administrador como cliente.

Fig 17 .Panel de Crear usuario en administrador

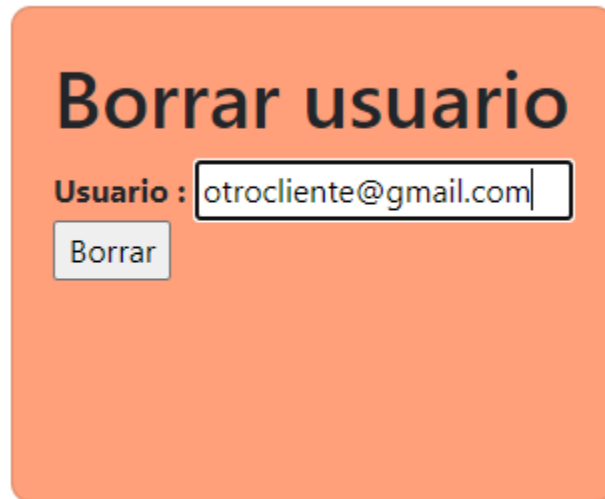
```
{
  user: 'otrocliente@gmail.com',
  password: '123',
  rol: '1',
  idnodo: '1'
}
Conexion correcta.
POST /usuarios 200 285.736 ms - 0
```

Fig 18 creación POST de nuevo usuario en administrador

user	password	rol	idnodo
admin1@gmail.com	123	2	1
babo@gmail.com	123	2	1
borra@gmail.com	123	1	2
borrsa@gmail.com	123	1	2
cinco@gmail.com	12	1	1
cuatro@gmail.com	12	1	2
daaaa@gmail.com	123	2	1
dados@gmail.com	1234	2	1
datosww@gmail.com	1234	2	1
dav@gmail.com	123	1	2
demom@gmail.com	123	2	1
dev@gmail.com	123	1	2
diego@gmail.com	123	2	3
dos@gmail.com	123	1	2
ejemplo1@gmail.com	123	1	1
ha@gmail.com	1234	1	1
haewon@gmail.com	123	1	2
mano@gmail.com	123	1	2
otro@gm.com	123	1	2
otrocliente@gmail.com	123	1	1
sana@gmail.com	123	1	2
tres@gmail.com	123	2	1
uno@gmail.com	123	1	1

Fig 19 . tabla de usuarios en mysql

El otro panel de Borrar un usuario , le permite borrar un usuario introduciendo el usuario con el cual se registró creó

A screenshot of a web interface for deleting a user. It has an orange background. At the top, the title "Borrar usuario" is displayed in large black font. Below it, the label "Usuario :" is followed by a text input field containing "otrocliente@gmail.com". Below the input field is a button labeled "Borrar".

Borrar usuario

Usuario : otrocliente@gmail.com

Borrar

Fig 20. Panel de Borrar usuario en administrador

```
{
  user: 'otrocliente@gmail.com',
  password: null,
  rol: null,
  idnodo: null
}
Conexion correcta.
DELETE /usuarios/otrocliente@gmail.com 200 43.291 ms - 15
```

Fig 21. DELETE de usuario 'otrocliente@gmail.com'

user	password	rol	idnodo
admin1@gmail.com	123	2	1
babo@gmail.com	123	2	1
borra@gmail.com	123	1	2
borrsa@gmail.com	123	1	2
cinco@gmail.com	12	1	1
cuatro@gmail.com	12	1	2
daaaa@gmail.com	123	2	1
dados@gmail.com	1234	2	1
dadosw@gmail.com	1234	2	1
dav@gmail.com	123	1	2
demom@gmail.com	123	2	1
dev@gmail.com	123	1	2
diego@gmail.com	123	2	3
dos@gmail.com	123	1	2
ejemplo1@gmail.com	123	1	1
ha@gmail.com	1234	1	1
haewon@gmail.com	123	1	2
mano@gmail.com	123	1	2
otro@gmail.com	123	1	2
sana@gmail.com	123	1	2
tres@gmail.com	123	2	1
uno@gmail.com	123	1	1

Fig 22. Evidencia de usuario 'otrocliente@gmail.com' borrado de tabla usuario en mysql

El panel de modificar usuario realiza un PUT en el que modifica la contraseña , rol y nodo , indicándole primero a que usuario se debería realizar esta modificación , en este caso se realizará a “diego@gmail.com” ver anterior figura.



Modificar en Usuario

Usuario :

Password nueva:

Rol nuevo:

Nodo nuevo :

Fig 23. Panel de Modificar usuario en administrador

```
{ user: 'diego@gmail.com', password: '1234', rol: '1', idnodo: '1' }
Conexion correcta.
PUT /usuarios/diego@gmail.com 200 49.740 ms - 19
```

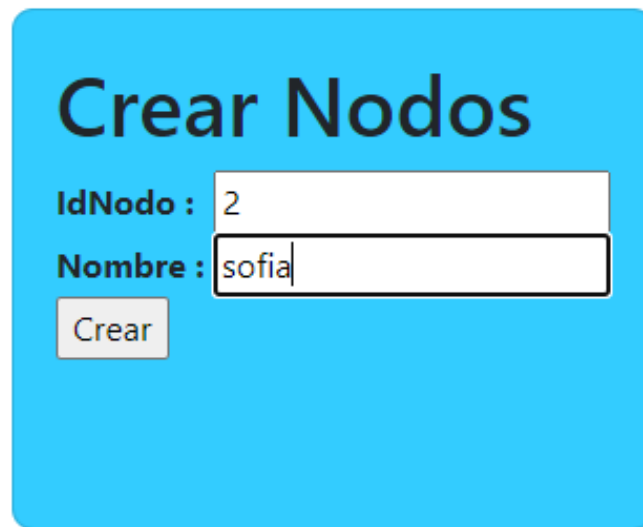
Fig 24. PUT de usuario ‘diego@gmail.com’ en password,rol y idnodo

user	password	rol	idnodo
admin1@gmail.com	123	2	1
babo@gmail.com	123	2	1
borra@gmail.com	123	1	2
borrsa@gmail.com	123	1	2
cinco@gmail.com	12	1	1
cuatro@gmail.com	12	1	2
daaaa@gmail.com	123	2	1
datos@gmail.com	1234	2	1
datosww@gmail.com	1234	2	1
dav@gmail.com	123	1	2
demom@gmail.com	123	2	1
dev@gmail.com	123	1	2
diego@gmail.com	1234	1	1
dos@gmail.com	123	1	2
ejemplo1@gmail.com	123	1	1
ha@gmail.com	1234	1	1
haewon@gmail.com	123	1	2
mano@gmail.com	123	1	2
otro@gm.com	123	1	2
sana@gmail.com	123	1	2
tres@gmail.com	123	2	1
uno@gmail.com	123	1	1

22 rows in set (0.003 sec)

Fig 25. Evidencia de usuario ‘diego@gmail.com’ modificado de tabla usuario en mysql

El panel de crear nodos se crea el nodo introduciendo el número de idnodo y un nombre

A blue rectangular panel with rounded corners. At the top, the text "Crear Nodos" is written in a large, bold, black font. Below this, there are two input fields. The first is labeled "IdNodo :" and contains the number "2". The second is labeled "Nombre :" and contains the text "sofia". Below the input fields is a button labeled "Crear".

Crear Nodos

IdNodo : 2

Nombre : sofia

Crear

Fig 26. Panel de Crear Nodos en administrador

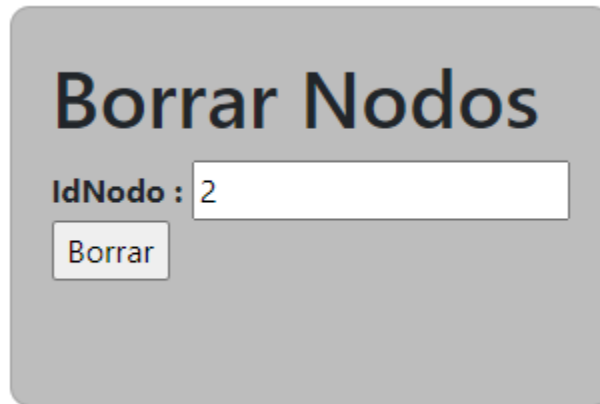
```
{ idnodo: '2', nombre: 'sofia' }  
Conexion correcta.  
POST /nodos/ 200 81.630 ms - 15
```

Fig 27. POST de idnodo y nombre en nodos

```
MariaDB [proyecto]> select * from nodo;  
+-----+-----+  
| idnodo | nombre |  
+-----+-----+  
|      1 | diego  |  
|      2 | sofia  |  
+-----+-----+  
2 rows in set (0.002 sec)
```

Fig 28. Evidencia de creación de nodos en nodo en mysql

El panel de borrar nodos , se realiza un DELETE dándole a introducir el idNodo correspondiente en este caso ejemplificado será el creado anteriormente el '2' .



Borrar Nodos

IdNodo :

Fig 29. Panel de Borrار Nodos en administrador

```
{ idnodo: '2', nombre: null }
Conexion correcta.
DELETE /nodos/2 200 44.482 ms - 12
```

Fig 30. DELETE de idnodo y nombre en nodos

```
MariaDB [proyecto]> select * from nodo;
+-----+-----+
| idnodo | nombre |
+-----+-----+
|      1 | diego  |
|      2 | sofia  |
+-----+-----+
2 rows in set (0.002 sec)

MariaDB [proyecto]> select * from nodo;
+-----+-----+
| idnodo | nombre |
+-----+-----+
|      1 | diego  |
+-----+-----+
1 row in set (0.001 sec)
```

Fig 31. Evidencia de borrar nodos en nodo en mysql

El panel de obtener datos todos , solo tienen un botón por el cual acciona y est dirrecionara otra pagina que se visualizara todos los datos y sus respectivas variables de temperatura y humedad .



Fig 32. Panel Obtener datos todos en administrador

```

fecha: '2022-11-10',
hora: '18:50:52'
}
{
  idnodo: 2,
  temperatura: 30,
  humedad: 58,
  fecha: 'viernes 11 nov 2022',
  hora: '6:52 pm '
}
GET /datos/ 200 148.591 ms - 1451

```

Fig 33. GET de todos los datos enviados

idnodo	temperatura	humedad	fecha	hora
2	40	58	jueves 10 nov 2022	9:46 am
1	40	58	jueves 10 nov 2022	9:46 am
2	30	58	jueves 10 nov 2022	9:46 am
1	26.2	63	2022-11-10	18:18:34
1	26.2	63	2022-11-10	18:18:34
1	26.2	63	2022-11-10	18:19:28
1	26.7	63	2022-11-10	18:50:52
1	26.7	63	2022-11-10	18:50:52
1	26.7	63	2022-11-10	18:50:52
1	26.7	63	2022-11-10	18:50:52
1	26.7	63	2022-11-10	18:50:52
1	26.7	63	2022-11-10	18:50:52
1	26.7	63	2022-11-10	18:50:52
1	26.7	63	2022-11-10	18:50:52
1	26.7	63	2022-11-10	18:50:52
2	30	58	viernes 11 nov 2022	6:52 pm

[Volver](#)

Fig 34. Página de observación de todos los datos en Administrador

Todos estas operaciones REST se comprueban con POSTMAN , ahora empezando el POST de datos en la tabla dato.

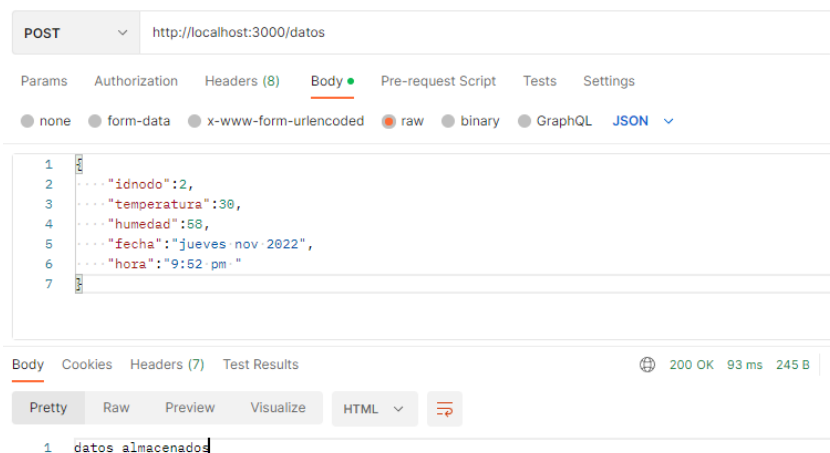


Fig 35. POST de datos en dato con POSTMAN

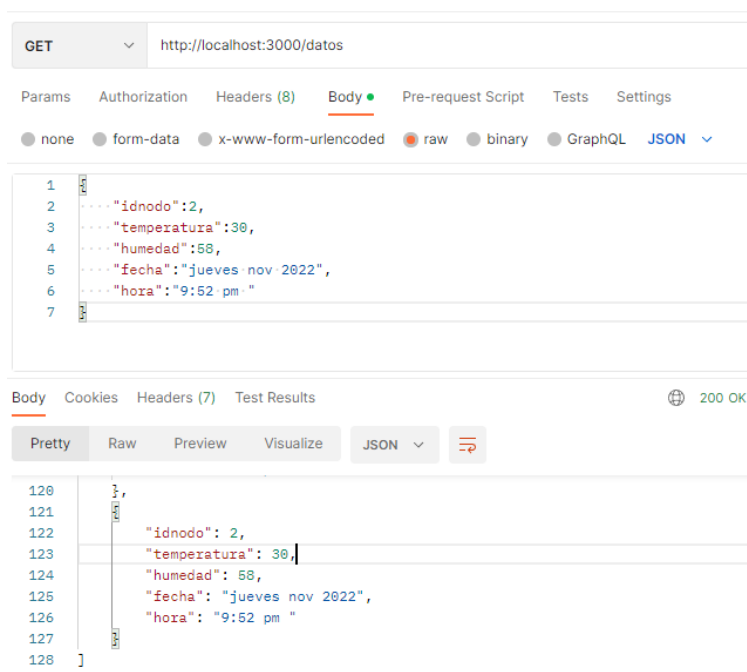


Fig 36. GET de datos en dato con POSTMAN

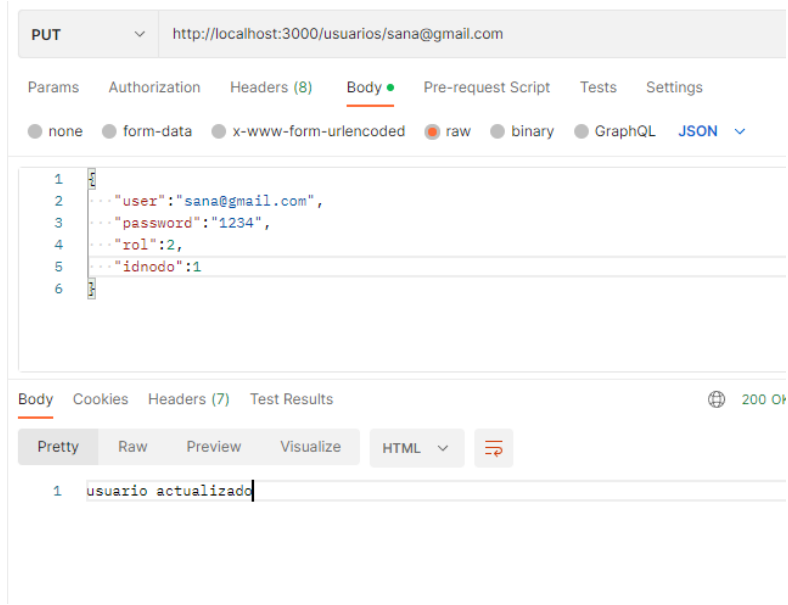


Fig 37. PUT de usuarios en usuario con POSTMAN

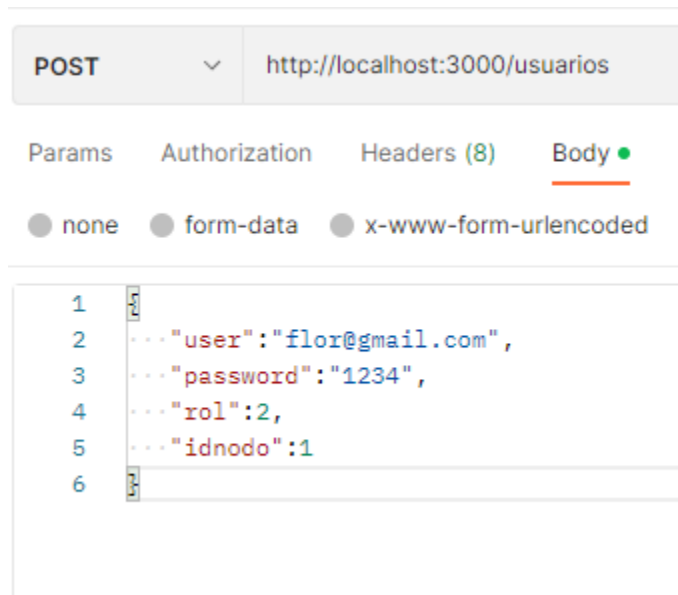


Fig 38. POST de usuarios en usuario con POSTMAN

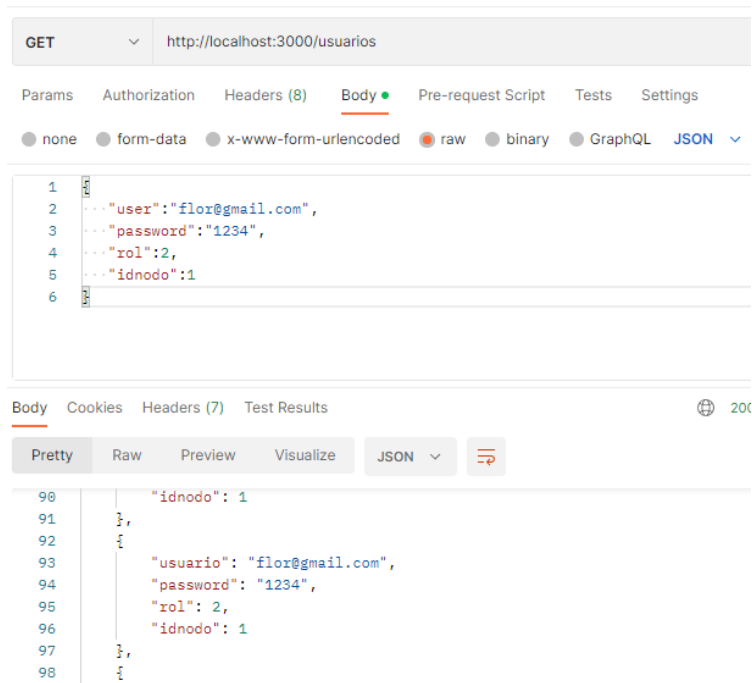


Fig 39. GET de usuarios en usuario con POSTMAN

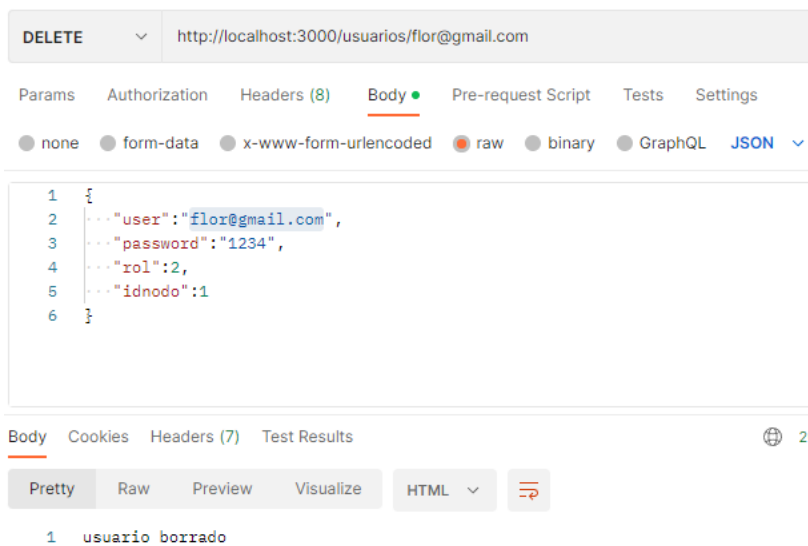


Fig 40. DELETE de usuarios en usuario con POSTMAN

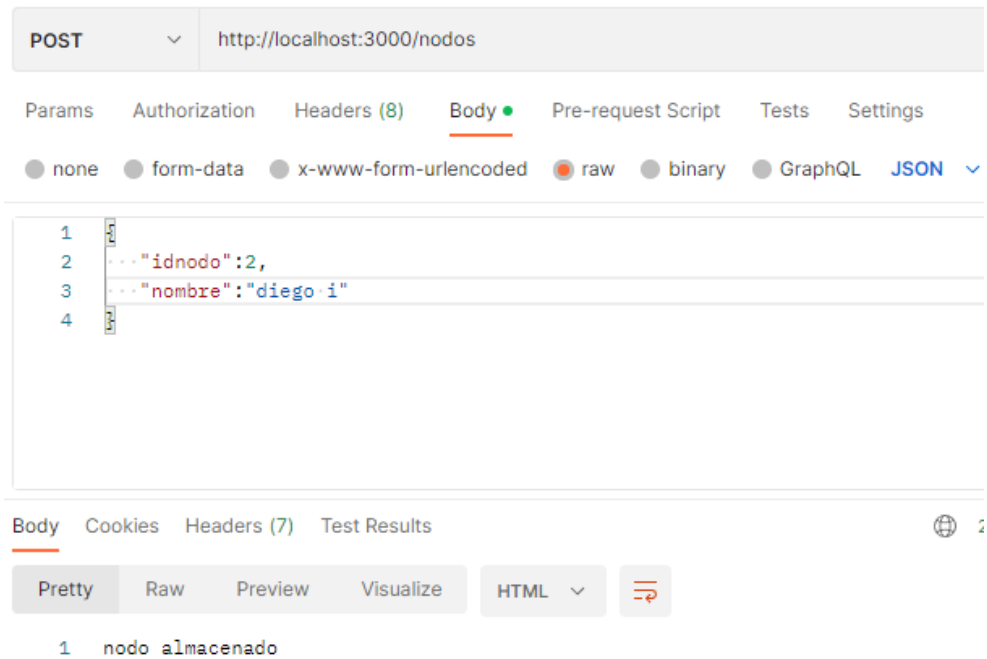


Fig 41. POST de nodos en nodo con POSTMAN

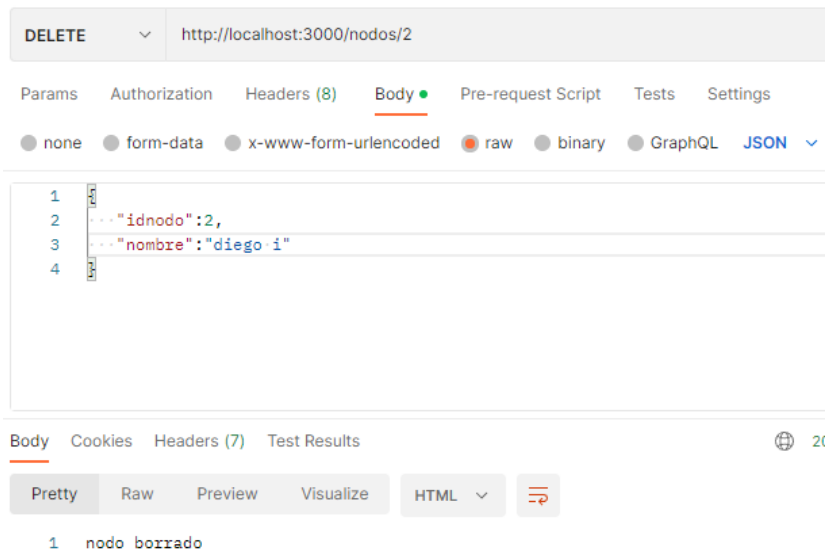


Fig 42. DELETE de nodos en nodo con POSTMAN

Anexos

Código de ESP32 (envío de datos):

```
#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
const char* password = "*****"; // clave de wifi
char mqttBroker[] = "192.16*.*.*"; //ip del servidor
char mqttClientId[] = "topico1"; //cualquier nombre
char inTopic[] = "topico1"; //topico 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["idnodo"] = 1;
            doc["temperatura"] = t;
            doc["humedad"] = h;
            doc["fecha"] = dayStamp;
            doc["hora"] = timeStamp;

            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

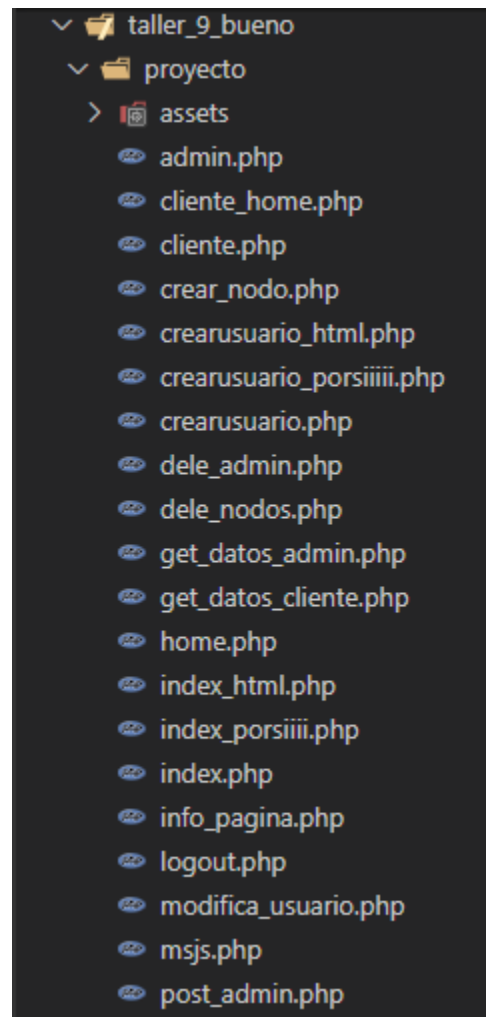
```

```

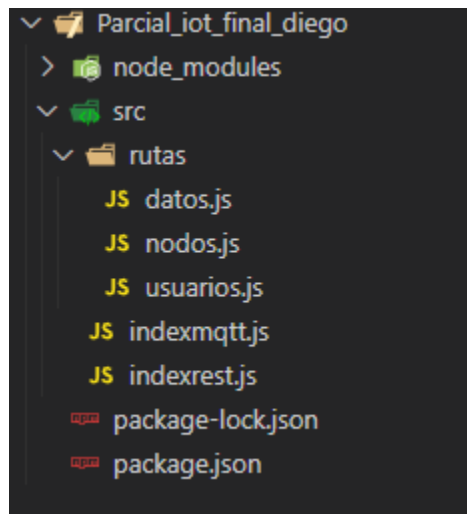
int splitT = formattedDate.indexOf("T");
dayStamp = formattedDate.substring(0, splitT);
//Serial.print("DATE: ");
//Serial.println(dayStamp);
// Extract time
timeStamp = formattedDate.substring(splitT+1, formattedDate.length()-1);
if (!client.connected()) {
  reconnect();
}
client.loop();
}

```

Archivos de la pagina web y procesamiento de REST



Archivos del funcionamiento del servidor y rutas



Código indexmqtt.js

```
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: 3307});

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 dato VALUES(null, ?, ?,?,?,?)',
            [json1.idnodo, json1.temperatura, json1.humedad,
json1.fecha,json1.hora],
            function(error, result){ //se ejecuta la inserción
                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
            });
        });
    })
})

```

Código 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/datos.js'));
app.use(require('./rutas/nodos.js'));
app.use(require('./rutas/usuarios.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.
```

Codigo 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: 3307
});

//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 dato', 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].fecha, "hora": resultado[i].hora };
            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); //muestra 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 dato VALUES(null, ?,?, ?,?,?)",
                [json1.idnodo, json1.temperatura, json1.humedad,
json1.fecha, json1.hora],
                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
                    }
                }
            );
        }
    });
});
});
});

```

```

});

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 dato 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].fecha, "hora": resultado[i].hora };
            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
    }
}
});
});

module.exports = router;

```

Codigo usuarios.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: 3307
});

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

router.get('/usuarios', (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 usuario', 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 = { "usuario": resultado[i].user, "password":
resultado[i].password, "rol": resultado[i].rol, "idnodo":
resultado[i].idnodo};

        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('/usuarios', (req, res) => {
  console.log(req.body); //muestra 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 usuario VALUES(?,?,?,?)',
[json1.user, json1.password, json1.rol, json1.idnodo], function (error,
result) { //se ejecuta la inserción
        if (error) {
          res.send("error al ejecutar el query");
        } else {
          tempConn.release();
          res.send(""); //mensaje de respuesta
        }
      }
    }
  });
});
});

//función get para consultar un nodo
router.get('/usuarios/:idusuario', (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.idusuario; //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 usuario where user=?', [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 = { "usuario": resultado[i].user, "password":
resultado[i].password, "rol": resultado[i].rol, "idnodo":
resultado[i].idnodo};
                        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 put para modificar un usuario
router.put('/usuarios/:idusuario', (req, res) => {
    console.log(req.body); //muestra en consola el json que llego
    json1 = req.body; //se almacena el json recibido en la variable json1
    var id = req.params.idusuario; //recogemos el parámetro enviado en la
url
    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('UPDATE usuario SET password=?, rol=?, idnodo=?
WHERE user=?', [json1.password, json1.rol, json1.idnodo, id], function
(error, result) { //se ejecuta la inserción
 if (error) {
 res.send("error al ejecutar el query");
 } else {
 tempConn.release();
 res.send("usuario actualizado"); //mensaje de respuesta
 }
});
 }
});
});

//función delete para borrar un nodo
router.delete('/usuarios/:idusuario', (req, res) => {
 console.log(req.body); //muestra en consola el json que llego
 json1 = req.body; //se almacena el json recibido en la variable json1
 var id = req.params.idusuario; //recogemos el parámetro enviado en la
url
 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('DELETE FROM usuario WHERE user=?', [id],
function (error, result) { //se ejecuta el borrado
 if (error) {
 res.send("error al ejecutar el query");
 } else {
 tempConn.release();
 res.send("usuario borrado"); //mensaje de respuesta
 }
});
 }
});
});
});

module.exports = router;`

Código nodos.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: 3307
});

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

router.get('/nodos', (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 nodo', 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, "nombre":
resultado[i].nombre, };
      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('/nodos', (req, res) => {
    console.log(req.body); //muestra 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 nodo VALUES(?,?)', [json1.idnodo,
json1.nombre], function (error, result) { //se ejecuta la inserción
                if (error) {
                    res.send("error al ejecutar el query");
                } else {
                    tempConn.release();
                    res.send("nodo almacenado"); //mensaje de respuesta
                }
            });
        }
    });
});

//función get para consultar un nodo
router.get('/nodos/: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 nodo 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, "nombre":
resultado[i].nombre};
            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 put para modificar un nodo
router.put('/nodos/:idnodo', (req, res) => {
    console.log(req.body); //muestra en consola el json que llego
    json1 = req.body; //se almacena el json recibido en la variable json1
    var id = req.params.idnodo; //recogemos el parámetro enviado en la url
    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('UPDATE nodo SET nombre=? WHERE idnodo=?',
[json1.nombre, id], function (error, result) { //se ejecuta la inserción
                if (error) {
                    res.send("error al ejecutar el query");
                } else {
                    tempConn.release();
                    res.send("nodo actualizado"); //mensaje de respuesta

```

```

    }
  });
}
});
});

//función delete para borrar un nodo
router.delete('/nodos/:idnodo', (req, res) => {
  console.log(req.body); //muestra en consola el json que llego
  json1 = req.body; //se almacena el json recibido en la variable json1
  var id = req.params.idnodo; //recogemos el parámetro enviado en la url
  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('DELETE FROM nodo WHERE idnodo=?', [id], function
(error, result) { //se ejecuta el borrado
        if (error) {
          res.send("error al ejecutar el query");
        } else {
          tempConn.release();
          res.send("nodo borrado"); //mensaje de respuesta
        }
      });
    }
  });
});

module.exports = router;

```

Referencias

- [1]"Capítulo 3. Almacenamiento", *Fao.org*, 2022. [Online]. Available: <https://www.fao.org/3/y4893s/y4893s06.htm>. [Accessed: 22- Aug- 2022]
- [2]"Capítulo 35: Mejoramiento de la seguridad alimentaria en el hogar", *Fao.org*, 2022. [Online]. Available: <https://www.fao.org/3/w0073s/w0073s13.htm>. [Accessed: 22- Aug- 2022]

[3] *Procolombia.co*, 2022. [Online]. Available: https://procolombia.co/sites/all/modules/custom/mccann/mccann_ruta_exportadora/files/06-cartilla-cadena-frio.pdf. [Accessed: 22- Aug- 2022]

[4] "Consumo y producción sostenibles - Desarrollo Sostenible", *Desarrollo Sostenible*, 2022. [Online]. Available: <https://www.un.org/sustainabledevelopment/es/sustainable-consumption-production/>. [Accessed: 22- Aug- 2022]

[5] "Hambre y seguridad alimentaria - Desarrollo Sostenible", *Desarrollo Sostenible*, 2022. [Online]. Available: <https://www.un.org/sustainabledevelopment/es/hunger/>. [Accessed: 22- Aug- 2022]

[6] "Sensor de temperatura, escoge el mejor para tus proyectos con Arduino", *Programar fácil con Arduino*, 2022. [Online]. Available: <https://programarfácil.com/podcast/82-escoger-mejor-sensor-temperatura-arduino/>. [Accessed: 22- Aug- 2022]

[7] "HDC1080 Arduino y ESP8266 sensor de temperatura y humedad", *Programar fácil con Arduino*, 2022. [Online]. Available: <https://programarfácil.com/blog/arduino-blog/hdc1080-arduino-esp8266/>. [Accessed: 22- Aug- 2022]

[8] "Sensor de Humedad Relativa HS1101", *VISTRONICA S.A.S*, 2022. [Online]. Available: <https://www.vistronica.com/sensores/sensor-de-humedad-relativa-hs1101-detail.html>. [Accessed: 22- Aug- 2022]

[9] "0.9 € 24% de DESCUENTO|Higrómetro con Sensor de humedad, resistencia sensible, módulo con funda, HR31 HR31D HR202 HR202L HJ3180B HDS10, 5 piezas|Sensores| - AliExpress", *aliexpress.com*, 2022. [Online]. Available: https://es.aliexpress.com/item/32951082181.html?spm=a2g0o.productlist.0.0.4f462a48CJN2YE&algo_pvid=3bcceb09-5679-4253-86c8-80d8cd0ccbec&algo_exp_id=3bcceb09-5679-4253-86c8-80d8cd0ccbec-16&pdp_ext_f=%7B%22sku_id%22%3A%2212000027223692462%22%7D&pdp_npi=2%40dis%21COP%217438.98%215646.45%21%21%2117970.06%21%21%402101d64d16610413989422613e8d0e%2112000027223692462%21sea&curPageLogUid=p6va0gBBweOo. [Accessed: 22- Aug- 2022]

[10] *Youtube.com*, 2022. [Online]. Available: https://www.youtube.com/watch?v=SKg_4ggqz2U. [Accessed: 22- Aug- 2022]

[11] *Youtube.com*, 2022. [Online]. Available: https://www.youtube.com/watch?v=mlJxILi_xds. [Accessed: 22- Aug- 2022]

[12] "ESP32: Date and Time (NTP Client)", *Phatiphat Thounthong*, 2022. [Online]. Available: <https://phatiphatt.wordpress.com/esp32-date-and-time-ntp-client/>. [Accessed: 22- Aug- 2022]

[13] "PlatformIO Registry", *PlatformIO Registry*, 2022. [Online]. Available: <https://registry.platformio.org/libraries/paulstoffregen/Time>. [Accessed: 22- Aug- 2022]

[14]Falconmasters, “Grid-vs-flexbox/estilos.css at codigo_base · falconmasters/grid-vs-flexbox,” *GitHub*, 29-Sep-2021. [Online]. Available: https://github.com/falconmasters/grid-vs-flexbox/blob/codigo_base/estilos.css. [Accessed: 10-Nov-2022]

[15]“Bootstrap testimonial carousel,” *Bootstrap Testimonial Carousel Template*. [Online]. Available: <https://www.tutorialrepublic.com/snippets/preview.php?topic=bootstrap&file=testimonial-carousel>. [Accessed: 10-Nov-2022]

[16]“Como crear un sistema de login de Usuario USANDO Sesiones Con php y mysql totalmente fácil,” *YouTube*, 11-Dec-2021. [Online]. Available: <https://www.youtube.com/watch?v=RdVAmwb96l4>. [Accessed: 10-Nov-2022]