

Actividad 6 IoT

Diego Iván Perea Montealegre (2238513) diego.perea@uao.edu.co
Facultad de Ingeniería, Universidad Autónoma de Occidente
Cali, Valle del Cauca

Creación de tabla de usuarios

```
1 row in set (0.00 sec)

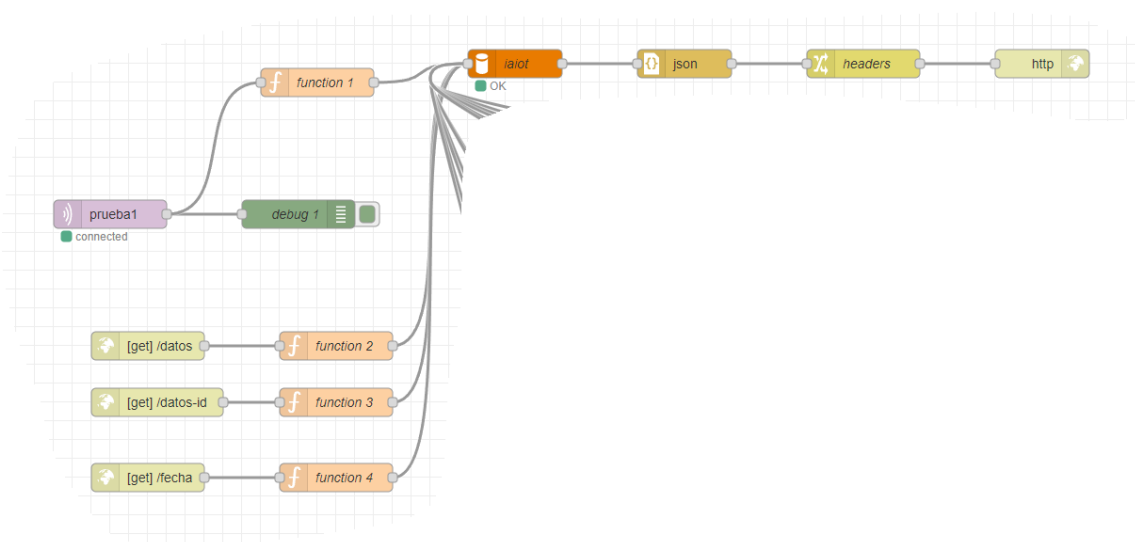
mysql> create table usuarios(
-> user varchar(100),
-> nombre varchar(100),
-> password varchar(100),
-> tipo int ,
-> primary key(user));
Query OK, 0 rows affected (0.06 sec)

mysql>
```

Creación tabla nodos :

```
mysql> create table nodos (
-> idnodo int,
-> nombreNodo varchar(100),
-> ubicacion varchar(100),
-> estado int,
-> user varchar(100),
-> primary key(idnodo));
Query OK, 0 rows affected (0.07 sec)
```

Se usa `C:\Users\User>node-red` 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;
```

POST ▼ http://localhost:1880/crearUsuario Send

Params Authorization Headers (4) **Body** Options Code

☐ none ☐ form-data ☐ x-www-form-urlencoded ☒ raw ☐ binary ☐ GraphQL JSON ▼ Beautify

```
1 {
2   "user": "admin",
3   "nombre": "administrador",
4   "password": "123",
5   "tipo": 1
6 }
```

Body Headers JSON ▼ Status: 200 OK Duration: 42 ms

```
1 {
2   "fieldCount": 0,
3   "affectedRows": 1,
4   "insertId": 0,
5   "info": "",
6   "serverStatus": 2,
```

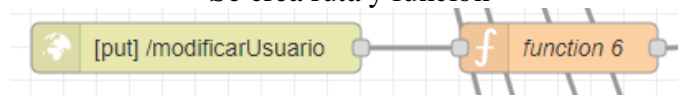
PROBLEMS OUTPUT DEBUG CONSOLE **TERMINAL** PORTS ... docker - prueba1 + ▼ □ 🗑 ... ^ ×

Empty set (0.01 sec)

```
mysql> select * from usuarios;
+-----+-----+-----+-----+
| user | nombre      | password | tipo |
+-----+-----+-----+-----+
| admin | administrador | 123      | 1    |
+-----+-----+-----+-----+
1 row in set (0.00 sec)

mysql>
```

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;
```

PUT `http://localhost:1880/modificarUsuario` **Send**

Params Authorization Headers (4) **Body** Options Code

☐ none ☐ form-data ☐ x-www-form-urlencoded ☒ raw ☐ binary ☐ GraphQL **JSON** [Beautify](#)

```
1 {
2   "user": "admin",
3   "nombre": "administradorChanged",
4   "password": "123",
5   "tipo": 1
6 }
```

Body Headers **JSON** Status: 200 OK Duration: 159 ms

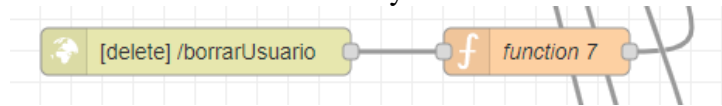
```
1 {
2   "fieldCount": 0,
3   "affectedRows": 1,
4   "insertId": 0,
5   "info": "Rows matched: 1 Changed: 1 Warnings: 0",
6   "serverStatus": 2,
```

PROBLEMS OUTPUT DEBUG CONSOLE **TERMINAL** PORTS ... `docker - prueba1` + - [] [X] ... ^

```
1 row in set (0.00 sec)

mysql> select * from usuarios;
+-----+-----+-----+-----+
| user | nombre           | password | tipo |
+-----+-----+-----+-----+
| admin | administradorChanged | 123      | 1    |
+-----+-----+-----+-----+
1 row in set (0.01 sec)
```

Se crea ruta y función



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

The screenshot shows a REST client interface with a DELETE request to `http://localhost:1880/borrarUsuario?user=admin`. The request body is empty, and the response is a JSON object indicating a successful deletion.

Request:

- Method: DELETE
- URL: `http://localhost:1880/borrarUsuario?user=admin`
- Body: (empty)

Response:

```
{
  "fieldCount": 0,
  "affectedRows": 1,
  "insertId": 0,
  "info": "",
  "serverStatus": 2,
  "warnings": []
}
```

Terminal Output:

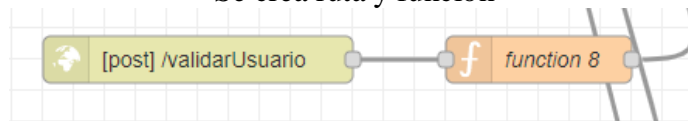
```
mysql> select * from usuarios;
Empty set (0.00 sec)

mysql>
```

user	nombre	password	tipo
admin	administradorChanged	123	1

1 row in set (0.01 sec)

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;
```

POST `http://localhost:1880/validarUsuario` **Send**

Params Authorization Headers (4) **Body** Options Code

☐ none ☐ form-data ☐ x-www-form-urlencoded ☒ raw ☐ binary ☐ GraphQL **JSON** Beautify

```
1 {
2   "user": "user1",
3   "password": "123"
4 }
5
6 }
```

Body Headers **JSON** Status: 200 OK Duration: 27 ms

```
2 {
3   "user": "user1",
4   "nombre": "diego",
5   "password": "123",
6   "tipo": 1
7 }
```

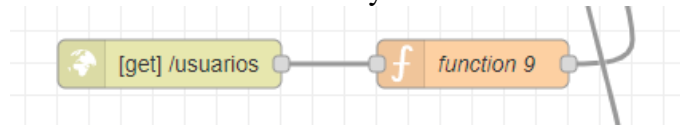
PROBLEMS OUTPUT DEBUG CONSOLE **TERMINAL** PORTS ... `docker - prueba1` + - [] [] ... ^ x

```
1 row in set (0.00 sec)

mysql> select * from usuarios;
+-----+-----+-----+-----+
| user | nombre | password | tipo |
+-----+-----+-----+-----+
| user1 | diego | 123 | 1 |
+-----+-----+-----+-----+
1 row in set (0.00 sec)

mysql>
```

Se crea ruta y función



```
msg.topic = "select * from usuarios";  
return msg;
```

GET ▼ http://localhost:1880/usuarios Send

Params Authorization Headers (4) **Body** Options Code

☐ none ☐ form-data ☐ x-www-form-urlencoded ☒ raw ☐ binary ☐ GraphQL JSON ▼ Beautify

1

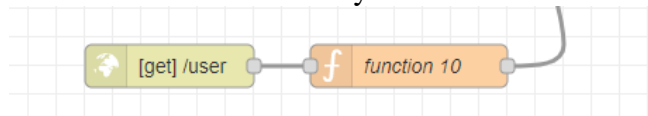
Body **Headers** JSON ▼ Status: 200 OK Duration: 26 ms

```
3  "user": "user1",  
4  "nombre": "diego",  
5  "password": "123",  
6  "tipo": 1  
7  },  
8  {  
9    "user": "user2",  
10   "nombre": "sara",  
    "password": "123",  
    "tipo": 2  
  }  
]
```

PROBLEMS OUTPUT DEBUG CONSOLE **TERMINAL** PORTS ... docker - prueba1 + ▼ □ 🗑 ... ^

```
+-----+-----+-----+-----+  
| user1 | diego | 123   | 1 |  
| user2 | sara  | 123   | 2 |  
+-----+-----+-----+-----+  
2 rows in set (0.00 sec)  
  
mysql>
```

Se crea ruta y función



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

GET `http://localhost:1880/user?user=user1` **Send**

Params Authorization Headers (4) **Body** Options Code

☐ none ☐ form-data ☐ x-www-form-urlencoded ☒ raw ☐ binary ☐ GraphQL **JSON** Beautify

1

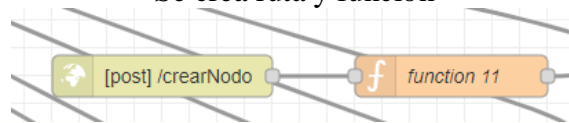
Body Headers **JSON** Status: 200 OK Duration: 34 ms

```
{  
  "user": "user1",  
  "nombre": "diego",  
  "password": "123",  
  "tipo": 1  
}
```

PROBLEMS OUTPUT DEBUG CONSOLE **TERMINAL** PORTS ...

```
+-----+-----+-----+-----+  
1 row in set (0.00 sec)  
  
mysql> select * from usuarios;  
+-----+-----+-----+-----+  
| user | nombre | password | tipo |  
+-----+-----+-----+-----+  
| user1 | diego | 123 | 1 |  
| user2 | sara | 123 | 2 |  
+-----+-----+-----+-----+  
2 rows in set (0.00 sec)
```

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;
```

POST http://localhost:1880/crearNodo Send

Params Authorization Headers (4) Body Options Code

```
1 {
2   "idnodo":1,
3   "nombre": "Nodo Uno",
4   "ubicacion":"Cali Sur",
5   "estado":1,
6   "user":"user1"
7 }
8 }
```

Body Headers JSON Status: 200 OK Duration: 26 ms

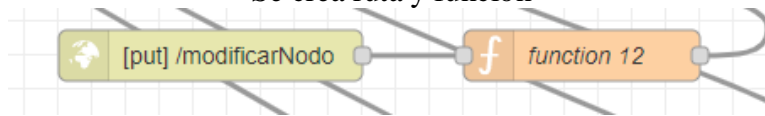
```
1 {
2   "fieldCount": 0,
3   "affectedRows": 1,
4   "insertId": 0,
5   "info": "",
6   "serverStatus": 2,
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS ... docker - prueba1

```
mysql> select * from nodos;
Empty set (0.06 sec)

mysql> select * from nodos;
+-----+-----+-----+-----+-----+
| idnodo | nombreNodo | ubicacion | estado | user |
+-----+-----+-----+-----+-----+
| 1 | Nodo Uno | Cali Sur | 1 | user1 |
+-----+-----+-----+-----+-----+
1 row in set (0.00 sec)
```


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;
```

The screenshot shows a REST client interface with a PUT request to `http://localhost:1880/modificarNodo`. The request body is a JSON object with the following fields:

```
{
  "idnodo": 1,
  "nombre": "chnaged",
  "ubicacion": "changed",
  "estado": 1,
  "user": "user1"
}
```

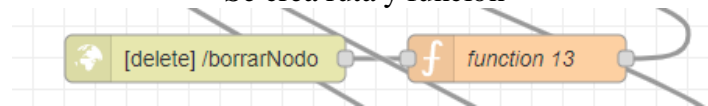
The response status is `200 OK` with a duration of `26 ms`. The response body is a JSON object with the following fields:

```
{
  "fieldCount": 0,
  "affectedRows": 1,
  "insertId": 0,
  "info": "Rows matched: 1 Changed: 1 Warnings: 0",
  "serverStatus": 2
}
```

Below the REST client, a terminal window shows the output of a MySQL query:

```
mysql> select * from nodos;
+-----+-----+-----+-----+-----+
| idnodo | nombreNodo | ubicacion | estado | user |
+-----+-----+-----+-----+-----+
| 1 | chnaged | changed | 1 | user1 |
+-----+-----+-----+-----+-----+
1 row in set (0.00 sec)
```

Se crea ruta y función



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

DELETE ▼ http://localhost:1880/borrarNodo?idnodo=2 Send

Params Authorization Headers (4) Body Options Code

☐ none ☐ form-data ☐ x-www-form-urlencoded ☒ raw ☐ binary ☐ GraphQL JSON ▼ Beautify

```
1 {  
2   "idnodo":2,  
3   "nombre": "Nodo Dos",  
4   "ubicacion":"Cali Sur",  
5   "estado":1,
```

Body Headers JSON ▼ Status: 200 OK Duration: 31 ms

```
1 {  
2   "fieldCount": 0,  
3   "affectedRows": 1,  
4   "insertId": 0,  
5   "info": "",
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS ... docker - prueba1 + ▼ □ 🗑 ... ^

```
| idnodo | nombreNodo | ubicacion | estado | user |  
+-----+-----+-----+-----+-----+  
| 1 | chnaged | changed | 1 | user1 |  
| 2 | Nodo Dos | Cali Sur | 1 | user2 |  
+-----+-----+-----+-----+  
2 rows in set (0.01 sec)  
  
mysql> select * from nodos;  
+-----+-----+-----+-----+-----+  
| idnodo | nombreNodo | ubicacion | estado | user |  
+-----+-----+-----+-----+-----+  
| 1 | chnaged | changed | 1 | user1 |  
+-----+-----+-----+-----+-----+  
1 row in set (0.00 sec)
```

Se crea ruta y función



```
msg.topic = "select * from nodos";  
return msg;
```

GET ▼ http://localhost:1880/nodos Send

Params Authorization Headers (4) **Body** Options Code

- data - urlencoded

```
1 {  
2   "idnodo":2,  
3   "nombre": "Nodo Dos",  
4   "ubicacion":"Cali Sur",  
5   "estado":1,  
6   "user":"user2"  
7 }  
8
```

Body Headers **JSON** Status: 200 OK Duration: 13 ms

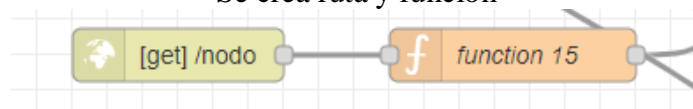
```
3   "idnodo": 1,  
4   "nombreNodo": "chnaged",  
5   "ubicacion": "changed",  
6   "estado": 1,  
7   "user": "user1"  
8 },  
9 {  
10  "idnodo": 2,
```

PROBLEMS OUTPUT DEBUG CONSOLE **TERMINAL** PORTS ... docker - prueba1 + ▼ □ 🗑 ... ^ ×

idnodo	nombreNodo	ubicacion	estado	user
1	chnaged	changed	1	user1
2	Nodo Dos	Cali Sur	1	user2

2 rows in set (0.01 sec)

Se crea ruta y función



```
var idnodo = msg.payload.idnodo;
msg.topic = "select * from nodos where idnodo=" + idnodo;
return msg;
```

GET ▼ http://localhost:1880/nodo?idnodo=1 Send

Params Authorization Headers (4) **Body** Options Code

- data - urlencoded

```
1 {
2   "idnodo":2,
3   "nombre": "Nodo Dos",
4   "ubicacion":"Cali Sur",
5   "estado":1,
6   "user":"user2"
7 }
8
```

Body Headers **JSON** ▼ Status: 200 OK Duration: 14 ms

```
2 {
3   "idnodo": 1,
4   "nombreNodo": "chnaged",
5   "ubicacion": "changed",
6   "estado": 1,
7   "user": "user1"
8 }
9
```

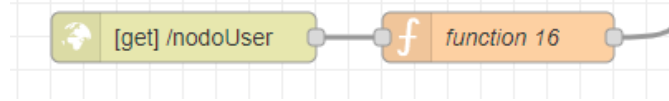
PROBLEMS OUTPUT DEBUG CONSOLE **TERMINAL** PORTS ... docker - prueba1 + ▼ □ 🗑 ... ^ >

idnodo	nombreNodo	ubicacion	estado	user
1	chnaged	changed	1	user1
2	Nodo Dos	Cali Sur	1	user2

2 rows in set (0.01 sec)

mysql> □

Se crea ruta y función



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

GET ▼ http://localhost:1880/nodoUser?user=user1 Send

Params Authorization Headers (4) **Body** Options Code

data urlencoded

1

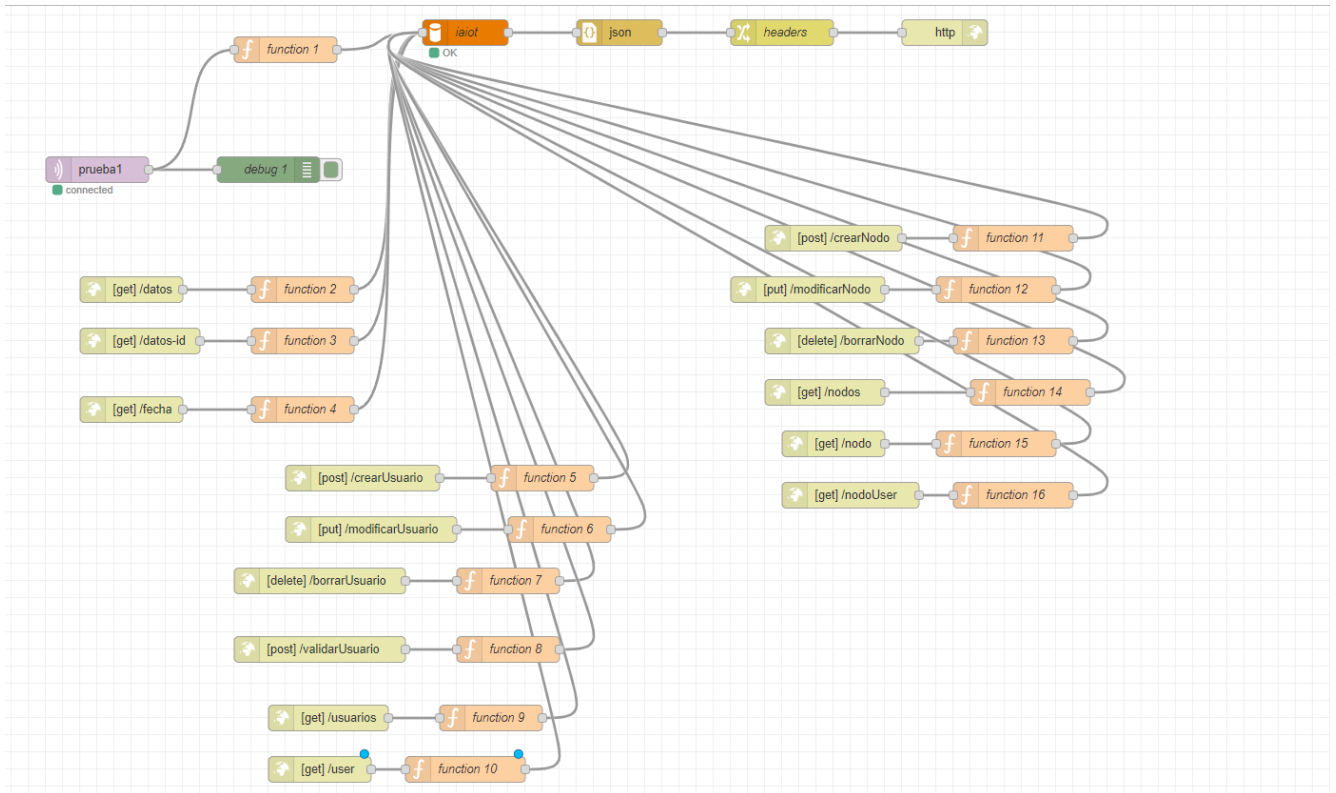
Body Headers **JSON** Status: 200 OK Duration: 14 ms

```
1 [
2   {
3     "idnodo": 1,
4     "nombreNodo": "chnaged",
5     "ubicacion": "changed",
6     "estado": 1,
7     "user": "user1"
8   }
9 ]
```

PROBLEMS OUTPUT DEBUG CONSOLE **TERMINAL** PORTS ... docker - prueba1 + ▼ □ 🗑 ... ^ ×

idnodo	nombreNodo	ubicacion	estado	user
1	chnaged	changed	1	user1
2	Nodo Dos	Cali Sur	1	user2

2 rows in set (0.00 sec)



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 los 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';
```

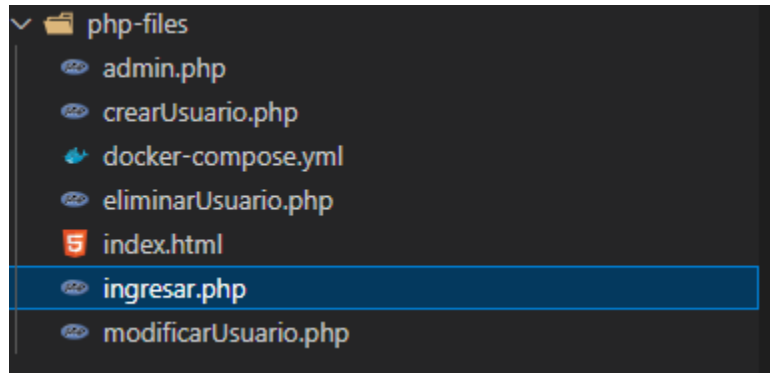
Tocaria cambiarlo por la ip del pc

```
// URL de la solicitud POST
```

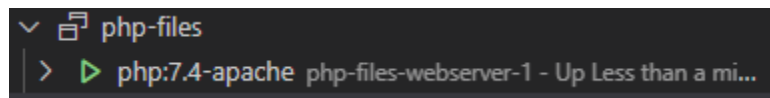
```
$url = 'http://IP_DE_LOCAL:1880/validarUsuario';
```

Y así con los demás archivo que tenga la ruta .

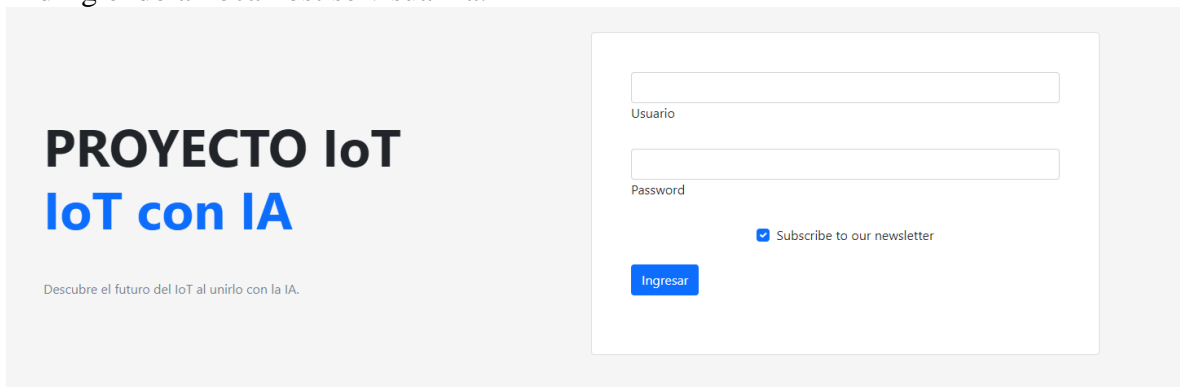
Teniendo ya todos los archivos ordenados en en el mimo directorio del Docker compose



Se realiza “docker-compose up”



Y dirigiendo al localhost se visualiza:



Recorda que en el mysql es :

```
mysql> select * from usuarios;
+-----+-----+-----+-----+
| user  | nombre | password | tipo |
+-----+-----+-----+-----+
| user1 | diego  | 123      | 1    |
| user2 | sara   | 123      | 2    |
+-----+-----+-----+-----+
2 rows in set (0.00 sec)
```

Por eso se intenta ingresar con el user1

PROYECTO IoT

IoT con IA

Descubre el futuro del IoT al unirlo con la IA.

Usuario

Password

☒ Subscribe to our newsletter

Ingresar

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.

Proyecto IoT Usuarios Nodos Datos Logout diego

Nombre	Usuario	Password	tipo	Editar	Eliminar
<input type="text" value="diego"/>	user1	<input type="text" value="123"/>	<input type="text" value="1"/>	MODIFICAR	ELIMINAR
<input type="text" value="sara"/>	user2	<input type="text" value="123"/>	<input type="text" value="3"/>	MODIFICAR	ELIMINAR

CREAR USUARIO

Dando crear usuario:

Proyecto IoT Usuarios Nodos Datos Logout diego

Nombre

Usuario

Password

Tipo

Editar Eliminar

MODIFICAR ELIMINAR

MODIFICAR ELIMINAR

CREAR USUARIO

CREAR USUARIO

Nombre

Usuario

Contraseña

Tipo

Close Crear Usuario

Proyecto lot
Usuarios
Nodos
Datos
Logout diego

Nombre	Usuario	Pass		Editar	Eliminar
diego	user1	123		MODIFICAR	ELIMINAR
sara	user2	123		MODIFICAR	ELIMINAR

CREAR USUARIO

CREAR USUARIO

×

CREAR USUARIO

Nombre

maria

Usuario

user3

Contraseña

...

Tipo

2

Close

Crear Usuario

Proyecto lot
Usuarios
Nodos
Datos
Logout diego

Nombre	Usuario	Password	tipo	Editar	Eliminar
diego	user1	123	1	MODIFICAR	ELIMINAR
sara	user2	123	3	MODIFICAR	ELIMINAR
maria	user3	123	2	MODIFICAR	ELIMINAR

CREAR USUARIO

Se evidencia la creación del usuario en la base de datos

user	nombre	password	tipo
user1	diego	123	1
user2	sara	123	3
user3	maria	123	2

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

Proyecto lot
Usuarios
Nodos
Datos
Logout diego

Nombre	Usuario	Password	tipo	Editar	Eliminar
diego	user1	123	1	MODIFICAR	ELIMINAR
sara	user2	123	3	MODIFICAR	ELIMINAR
maria	user3	123	4	MODIFICAR	ELIMINAR

CREAR USUARIO

user	nombre	password	tipo
user1	diego	123	1
user2	sara	123	3
user3	maria	123	4

Y eliminándolo

Proyecto lot Usuarios **Nodos** Datos Logout diego

Nombre	Usuario	Password	tipo	Editar	Eliminar
<input type="text" value="diego"/>	user1	<input type="text" value="123"/>	<input type="text" value="1"/>	<input type="button" value="MODIFICAR"/>	<input type="button" value="ELIMINAR"/>
<input type="text" value="sara"/>	user2	<input type="text" value="123"/>	<input type="text" value="3"/>	<input type="button" value="MODIFICAR"/>	<input type="button" value="ELIMINAR"/>
<input type="text" value="maria"/>	user3	<input type="text" value="123"/>	<input type="text" value="4"/>	<input type="button" value="MODIFICAR"/>	<input type="button" value="ELIMINAR"/>

```
+-----+-----+-----+-----+
| user | nombre | password | tipo |
+-----+-----+-----+-----+
| user1 | diego | 123 | 1 |
| user2 | sara | 123 | 3 |
+-----+-----+-----+-----+
```

Lo mismo se puede hacer con los nodos:

Proyecto lot Usuarios **Nodos** Datos Logout diego

IdNodo	Nombre	Ubicacion	Estado	Usuario	Editar	Eliminar
1	<input type="text" value="chnaged"/>	<input type="text" value="changed"/>	<input type="text" value="1"/>	<input type="text" value="user1"/>	<input type="button" value="MODIFICAR"/>	<input type="button" value="ELIMINAR"/>
2	<input type="text" value="nodo2"/>	<input type="text" value="popayan"/>	<input type="text" value="1"/>	<input type="text" value="user2"/>	<input type="button" value="MODIFICAR"/>	<input type="button" value="ELIMINAR"/>

Se crea un nodo

Proyecto lot Usuarios **Nodos** Datos Logout diego

IdNodo	Nombre	Ubicacion	Estado	Usuario	Editar	Eliminar
1	<input type="text" value="chnaged"/>	<input type="text" value="changed"/>	<input type="text" value="1"/>	<input type="text" value="user1"/>	<input type="button" value="MODIFICAR"/>	<input type="button" value="ELIMINAR"/>
2	<input type="text" value="nodo2"/>	<input type="text" value="popayan"/>	<input type="text" value="1"/>	<input type="text" value="user2"/>	<input type="button" value="MODIFICAR"/>	<input type="button" value="ELIMINAR"/>

CREAR USUARIO

IdNodo

Nombre

Ubicacion

Estado

Usuario

Close

Crear Nodo

Proyecto lot
Usuarios
Nodos
Datos
Logout diego

IdNodo	Nombre	Ubicacion	Estado	Usuario	Editar	Eliminar
1	<input type="text" value="chnaged"/>	<input type="text" value="changed"/>	<input type="text" value="1"/>	<input type="text" value="user1"/>	MODIFICAR	ELIMINAR
2	<input type="text" value="nodo2"/>	<input type="text" value="popayan"/>	<input type="text" value="1"/>	<input type="text" value="user2"/>	MODIFICAR	ELIMINAR
3	<input type="text" value="nodo3"/>	<input type="text" value="cali"/>	<input type="text" value="1"/>	<input type="text" value="user8"/>	MODIFICAR	ELIMINAR

CREAR NODO

idnodo	nombreNodo	ubicacion	estado	user
1	chnaged	changed	1	user1
2	nodo2	popayan	1	user2
3	nodo3	cali	1	user8

Ahora modificar el usuario del idnodo 3 :

Proyecto lot
Usuarios
Nodos
Datos
Logout diego

IdNodo	Nombre	Ubicacion	Estado	Usuario	Editar	Eliminar
1	<input type="text" value="chnaged"/>	<input type="text" value="changed"/>	<input type="text" value="1"/>	<input type="text" value="user1"/>	MODIFICAR	ELIMINAR
2	<input type="text" value="nodo2"/>	<input type="text" value="popayan"/>	<input type="text" value="1"/>	<input type="text" value="user2"/>	MODIFICAR	ELIMINAR
3	<input type="text" value="nodo3"/>	<input type="text" value="cali"/>	<input type="text" value="1"/>	<input type="text" value="user7"/>	MODIFICAR	ELIMINAR

CREAR NODO

idnodo	nombreNodo	ubicacion	estado	user
1	chnaged	changed	1	user1
2	nodo2	popayan	1	user2
3	nodo3	cali	1	user7

Y eliminar

Proyecto lot
Usuarios
Nodos
Datos
Logout diego

IdNodo	Nombre	Ubicacion	Estado	Usuario	Editar	Eliminar
1	<input type="text" value="chnaged"/>	<input type="text" value="changed"/>	<input type="text" value="1"/>	<input type="text" value="user1"/>	MODIFICAR	ELIMINAR
2	<input type="text" value="nodo2"/>	<input type="text" value="popayan"/>	<input type="text" value="1"/>	<input type="text" value="user2"/>	MODIFICAR	ELIMINAR
3	<input type="text" value="nodo3"/>	<input type="text" value="cali"/>	<input type="text" value="1"/>	<input type="text" value="user7"/>	MODIFICAR	ELIMINAR

CREAR NODO

idnodo	nombreNodo	ubicacion	estado	user
1	chnaged	changed	1	user1
2	nodo2	popayan	1	user2

2 rows in set (0.00 sec)

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

Proyecto lot
Usuarios
Nodos
Datos

Selecciona Visualizador de datos

Selecciona una opción:

Tabla de datos

Ir a la página seleccionada

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

DATOS CAPTURADOS

ID	IDNODO	TEMPERATURA	HUMEDAD	FECHA
1	1	24.1	59	2023-10-30T04:12:09.000Z
2	1	24.1	59	2023-10-30T04:12:14.000Z
3	1	24.2	59	2023-10-30T04:12:20.000Z
4	1	24.2	59	2023-10-30T04:12:25.000Z
5	1	24.2	59	2023-10-30T04:12:30.000Z
6	1	24.3	59	2023-10-30T04:12:36.000Z
7	2	25.8	62	2023-11-11T23:04:04.000Z
8	2	26.2	61	2023-11-11T23:04:23.000Z
9	2	26.2	61	2023-11-11T23:04:29.000Z
10	2	26.2	61	2023-11-11T23:04:35.000Z

En Cards datos se visualiza :

Selecciona Visualizador de datos

Selecciona una opción:

Cards datos

Ir a la página seleccionada

DATO ESPECIFICO

Ingresa un número ID:

Actualizar ID

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

DATO ESPECIFICO

Ingrese un número ID:

1

Actualizar ID

2023-10-30T04:12:09.000Z

TEMPERATURA

24.1

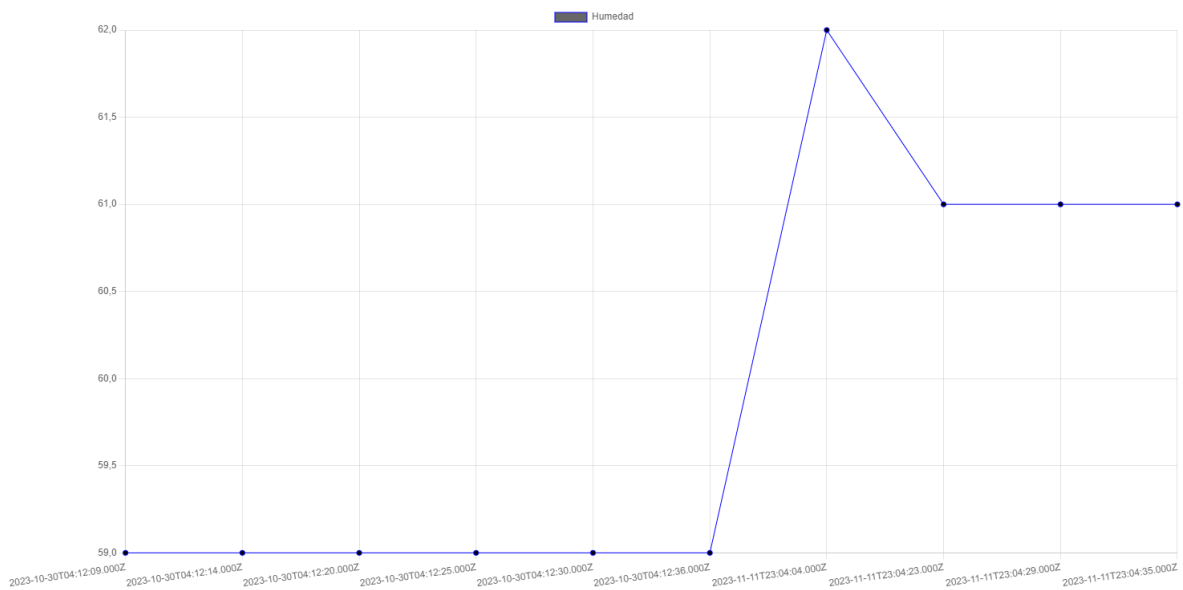
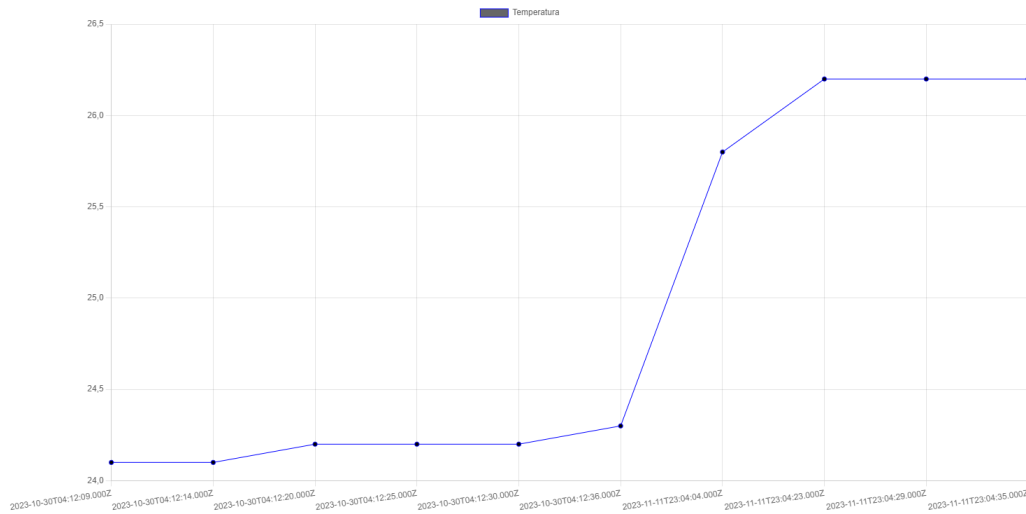
Ver gráfica

HUMEDAD

59

Ver gráfica

Se observa la fecha , temperatura y humedad en ese id específico .
En ver grafica se visualiza todos los datos de temperatura y humedad respectivos



Estas graficas también se pueden observar en el seleccionador :

Selecciona Visualizador de datos

Selecciona una opción:

Grafica de temperatura

Ir a la página seleccionada

Selecciona Visualizador de datos

Selecciona una opción:

Grafica de humedad

Ir a la página seleccionada

Para salir de la sesión con logout :

Proyecto IoT Usuarios Nodos Datos							Logout diego
IdNodo	Nombre	Ubicacion	Estado	Usuario	Editar	Eliminar	
1	chnaged	changed	1	user1	MODIFICAR	ELIMINAR	
2	nodo2	popayan	1	user2	MODIFICAR	ELIMINAR	
CREAR NODO							

Ingresando con un usuario normal , no admin :
En este caso es el user2:

PROYECTO IoT

IoT con IA

Descubre el futuro del IoT al unirlo con la IA.

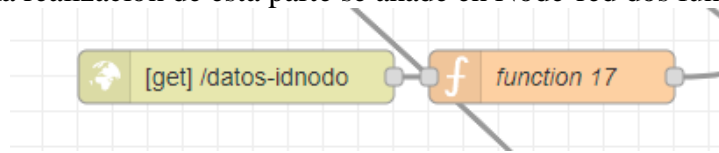
Usuario

Password
☒ Subscribe to our newsletter

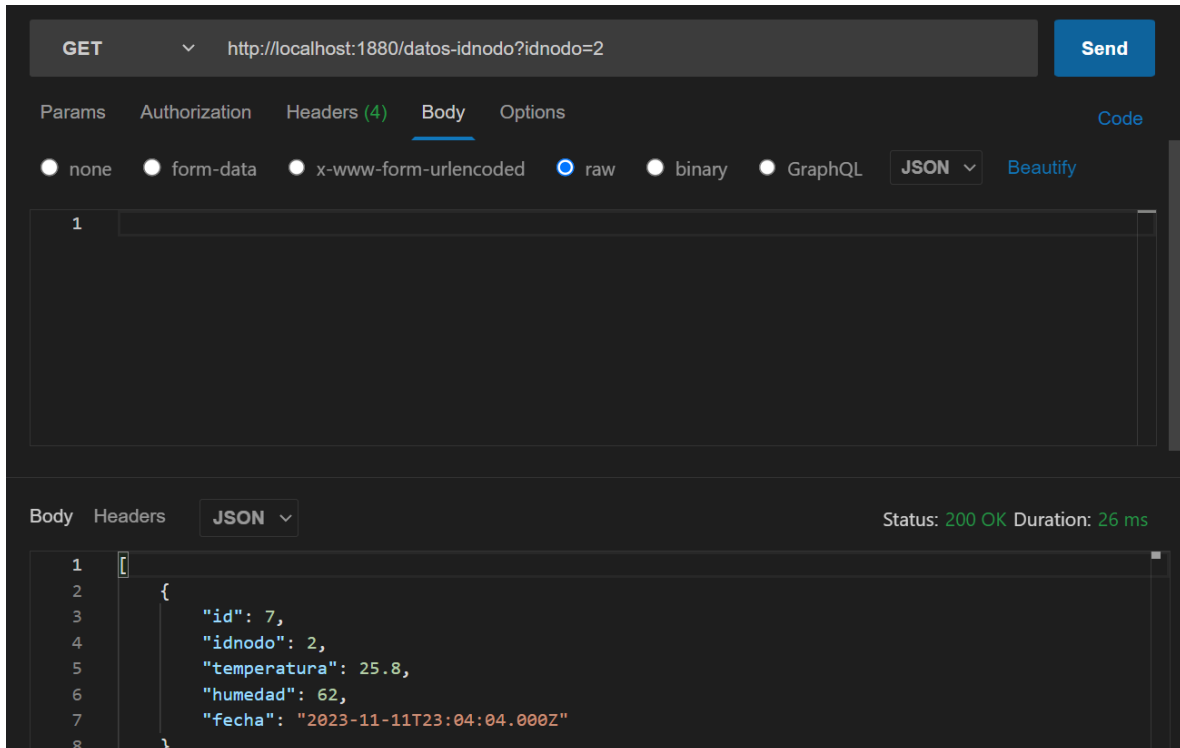
Observará los datos de su ID Nodo correspondiente en este caso su ID Nodo es el 2

ID Nodo	Temperatura	Humedad	Fecha
2	25.8	62	2023-11-11T23:04:04.000Z
2	26.2	61	2023-11-11T23:04:23.000Z
2	26.2	61	2023-11-11T23:04:29.000Z
2	26.2	61	2023-11-11T23:04:35.000Z

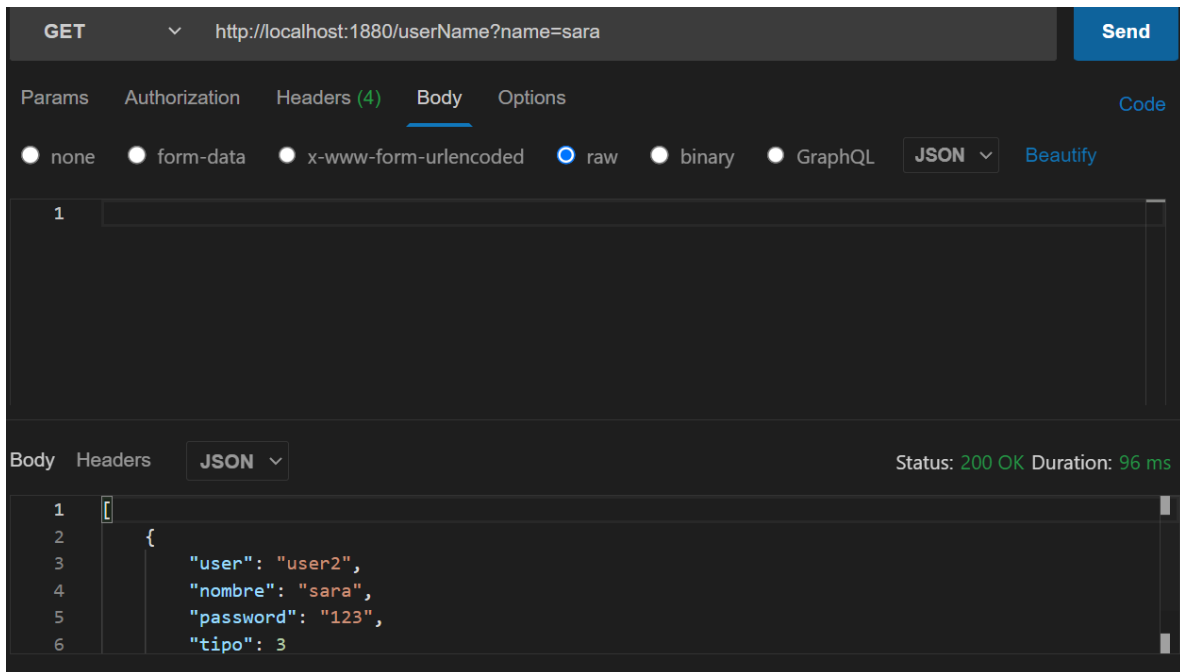
Para la realización de esta parte se añade en Node-red dos funciones:



```
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"; //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["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();
}
```