ACTIVIDAD 5. Almacenamiento de datos

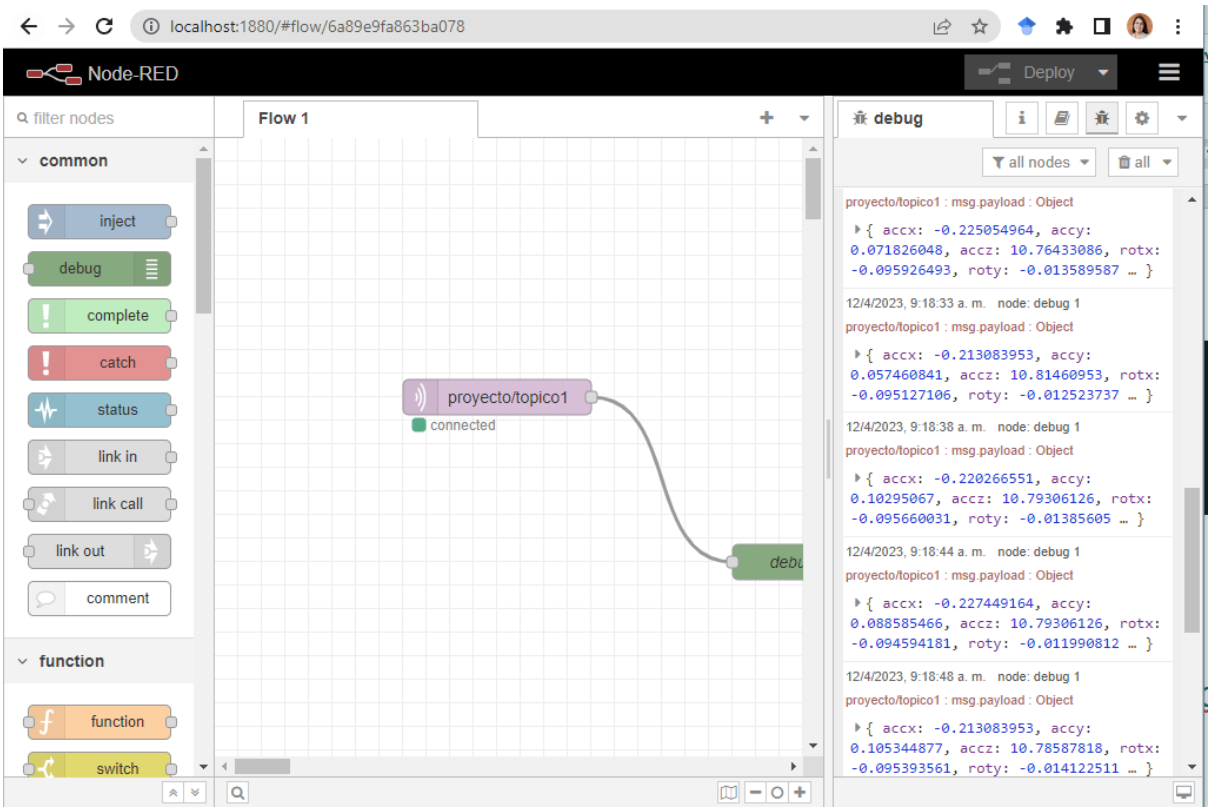
Inicio

El objetivo de esta actividad es configurar la plataforma para que después de recibir los datos, los almacene y los deje disponibles para que desde aplicaciones externas se pueda hacer uso de ellos.

Trabajaremos con Node-RED que ya recibió los datos a través del protocolo MQTT, usaremos MySQL para el almacenamiento de los datos y se crearán unos puntos de acceso a los datos usando el protocolo HTTP.

Recepción de los datos por MQTT en node-RED

Del taller anterior quedamos en el punto en donde ya tenemos los datos en node-Red:



El paso siguiente es almacenarlos.

MySQL

MySQL es un sistema de gestión de bases de datos relacionales (RDBMS) de código abierto y uno de los sistemas de gestión de bases de datos más populares en el mundo. Fue desarrollado por Oracle Corporation y se utiliza para almacenar y administrar datos en aplicaciones web, de negocios y de software.

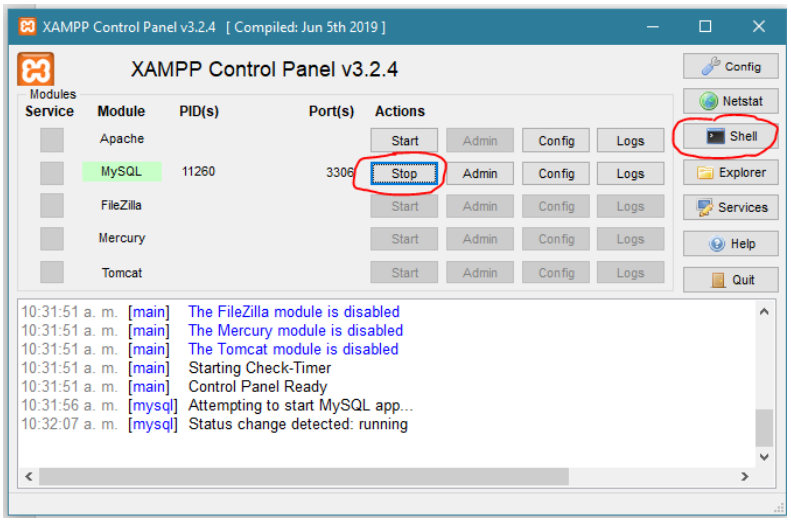
MySQL es una base de datos relacional, lo que significa que se organiza en tablas de datos relacionadas entre sí. Permite la creación, modificación y eliminación de datos de estas tablas, así como la creación de relaciones entre ellas. Además, MySQL ofrece una gran cantidad de funciones y herramientas para administrar y optimizar el rendimiento de la base de datos, incluyendo herramientas de respaldo y recuperación de datos, así como herramientas de análisis y monitoreo del rendimiento.

MySQL es compatible con muchos lenguajes de programación y es utilizado por una amplia gama de aplicaciones y sitios web, desde pequeñas aplicaciones web hasta grandes sistemas de gestión de contenido y comercio electrónico. La comunidad de MySQL también es muy activa y ha creado una gran cantidad de recursos y herramientas para ayudar a los usuarios a aprender y utilizar el sistema.

Para usar MySQL primero debemos instalarlo. La instalación de MySQL se puede hacer a través de XAMPP y Se puede acceder a él a través de la línea de comandos, después de que se haya iniciado el servicio.

XAMPP es un paquete de software libre que incluye una serie de herramientas necesarias para crear y administrar sitios web en un entorno de desarrollo local. XAMPP es una abreviatura de "X (para diferentes sistemas operativos), Apache, MySQL, PHP, Perl". El paquete también incluye otros programas como phpMyAdmin y OpenSSL. La instalación de XAMPP se hace descargándolo del sitio oficial: https://www.apachefriends.org/es/download.html

Ya instalado se desplegará un entorno en el que se pueden iniciar los servicios:



Desde el Shell podemos acceder al entorno de comandos para crear la base de datos y las tablas:

```
mysql -u root

Setting environment for using XAMPP for Windows.
zsolarte@D126474 c:\xampp2
# mysql -u root
Welcome to the MariaDB monitor.  Commands end with ; or \g.
Your MariaDB connection id is 16
Server version: 10.4.14-MariaDB mariadb.org binary distribution

Copyright (c) 2000, 2018, Oracle, MariaDB Corporation Ab and others.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

MariaDB [(none)]>
```

Crearemos una base de datos llamada iaiot

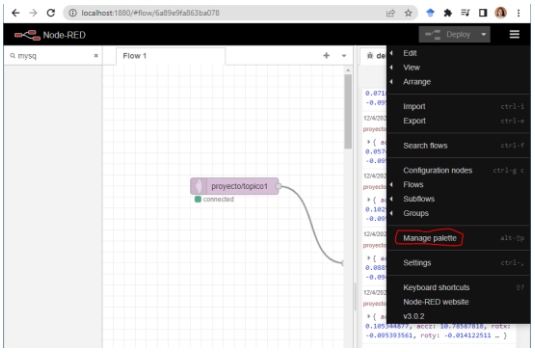
```
MariaDB [(none)]> create database iaiot;
Query OK, 1 row affected (0.001 sec)

MariaDB [(none)]> use iaiot
Database changed
```

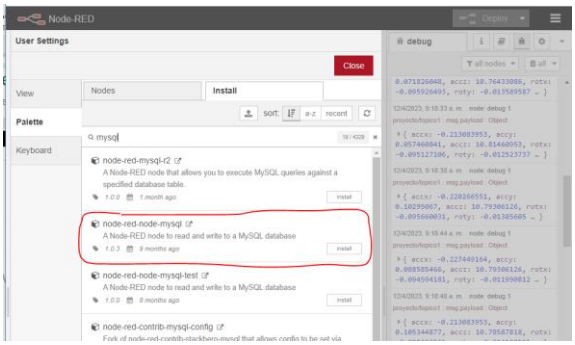
dentro de ella una tabla para almacenar los datos que nos envía el nodo iot, a esta tabla la llamaremos datos.

```
MariaDB [iaiot]> create table datos (
    -> id int auto_increment,
    -> ac_x float,
    -> ac_y float,
    -> ac_z float,
    -> rot_x float,
    -> rot_y float,
    -> rot_z float,
    -> temperatura float,
    -> fecha datetime default now(),
    -> primary key(id));
Query OK, 0 rows affected (0.015 sec)
```

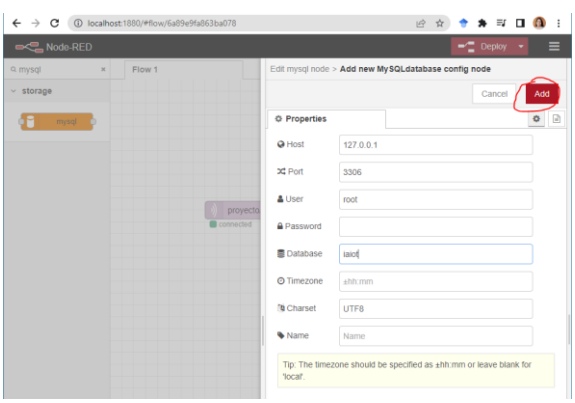
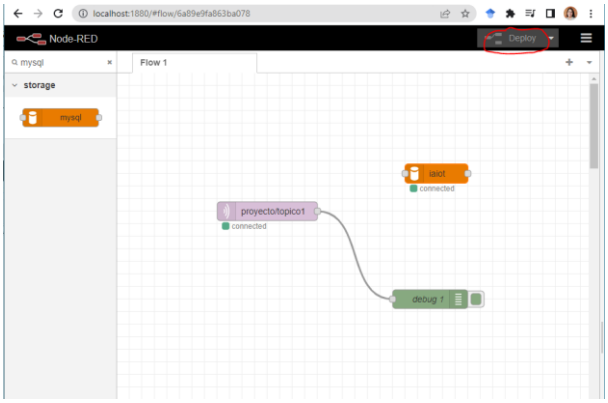
Configuración de MySQL en node-RED

En node-RED incluimos el nodo MySQL, si no se encuentra debemos instalarlo.

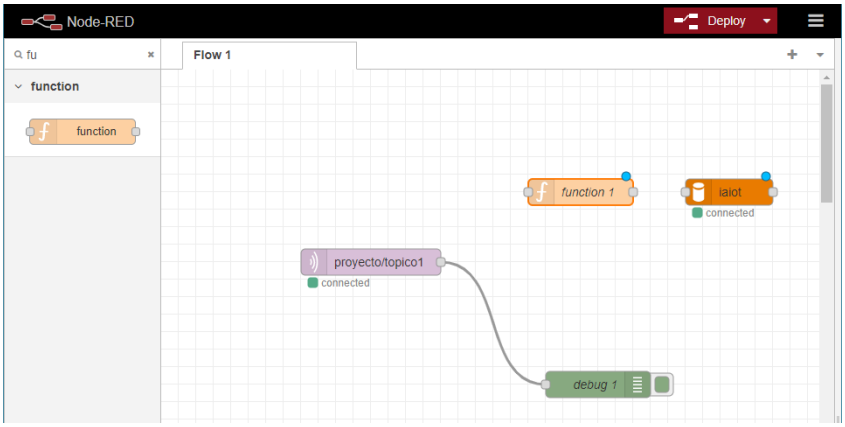




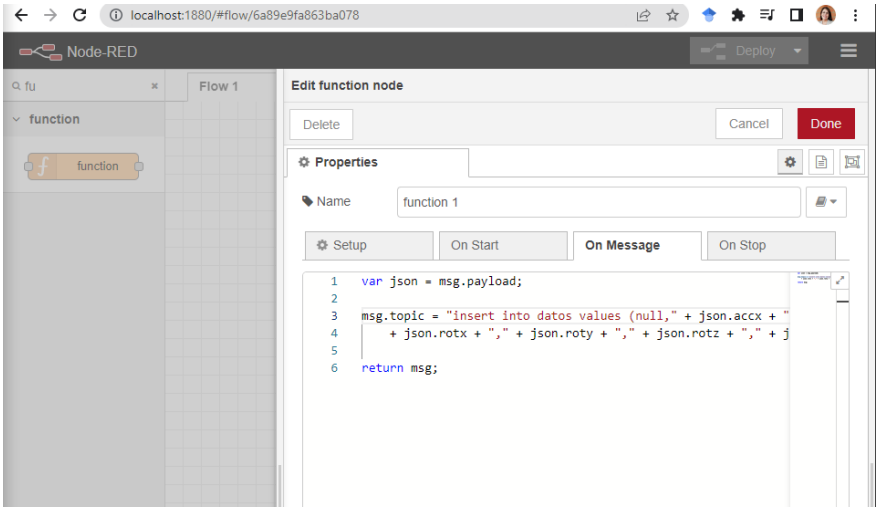
Ya nos aparece el nodo mysql y lo añadimos a nuestra pantalla y le damos doble clic para configurarlo y cuando le damos deploy deberá aparecer conectada.

Para poder insertar los datos que nos están llegando del dispositivo debemos crear una función con el query adecuado.



Una función acepta código java script. La configuramos de la siguiente manera. Le damos dobleclic a la función:

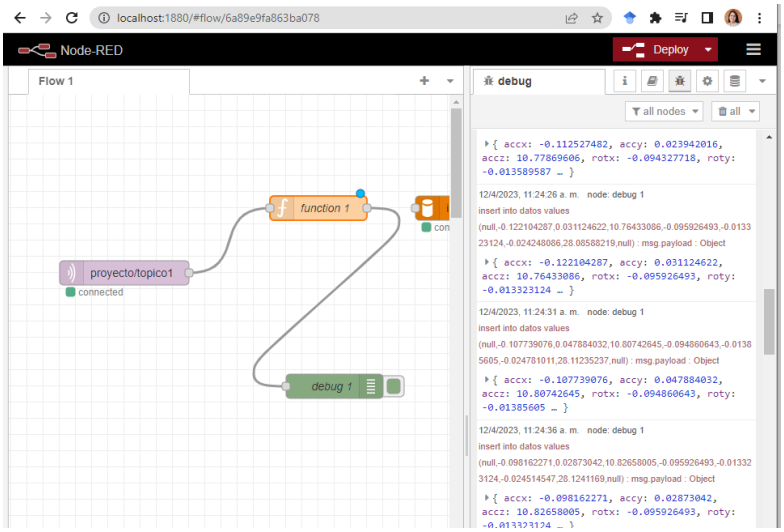


El código es el siguiente:

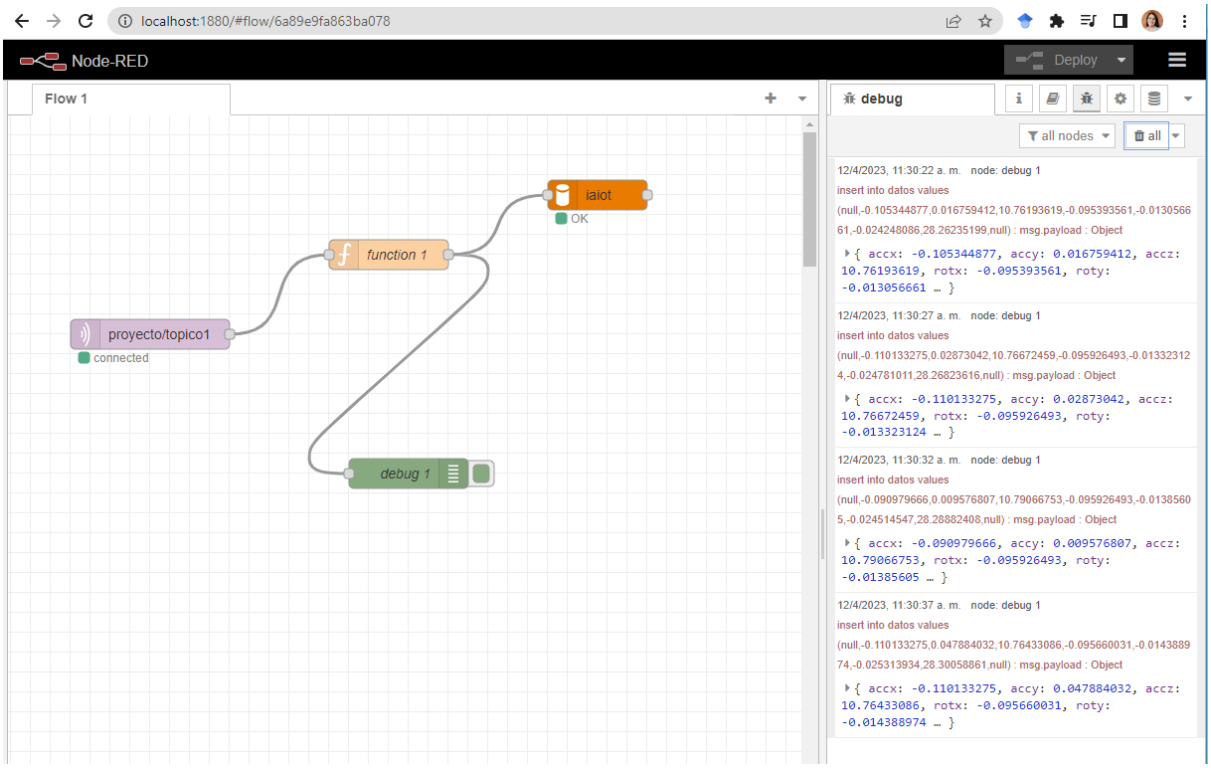
```
var json = msg.payload;

msg.topic = "insert into datos values (null," + json.accx + "," + json.accy +
"," + json.accz + ","
    + json.rotx + "," + json.roty + "," + json.rotz + "," + json.temp +
",now())";
```

```
return msg;
```

Conectamos el debug a la función y verificamos que el mensaje contenga los datos que requerimos en el formato adecuado:



Ahora conectamos la función al nodo de mysql.



Verificamos en mysql que los datos se estén almacenando:

31	-0.112527	0.0454898	10.7787	-0.0953936	-0.0130567	-0.0242481	28.3594	NULL	
32	-0.117316	0.023942	10.7667	-0.0953936	-0.0135896	-0.0245145	28.3682	NULL	
33	-0.105345	0.0191536	10.7787	-0.0948606	-0.0130567	-0.0245145	28.3535	NULL	
34	-0.112527	0.0454898	10.7859	-0.09566	-0.0135896	-0.0253139	28.3506	NULL	
35	-0.11971	0.0383072	10.7404	-0.0953936	-0.0141225	-0.0253139	28.3712	NULL	
36	-0.117316	0.0215478	10.738	-0.0953936	-0.0125237	-0.0242481	28.3712	NULL	
37	-0.107739	0.0191536	10.7524	-0.0951271	-0.013856	-0.0255804	28.3624	NULL	
38	-0.112527	0.0407014	10.7907	-0.0951271	-0.013856	-0.0250475	28.38	NULL	
39	-0.107739	0.0215478	10.7715	-0.09566	-0.0133231	-0.0250475	28.3771	NULL	
40	-0.112527	0.0263362	10.7524	-0.0948606	-0.0127902	-0.0250475	28.3653	NULL	
41	-0.0957681	0.023942	10.7811	-0.0953936	-0.0135896	-0.0242481	28.3653	NULL	
42	-0.100556	0.035913	10.7595	-0.0967259	-0.0130567	-0.0242481	28.38	NULL	
43	-0.11971	0.0263362	10.7452	-0.096193	-0.0135896	-0.0250475	28.3771	NULL	
44	-0.129287	0.0574608	10.7811	-0.0964594	-0.0125237	-0.0242481	28.3535	NULL	
45	-0.100556	0.023942	10.8194	-0.096193	-0.0122573	-0.024781	27.33	2023-04-12 11:38:48	
46	-0.110133	0.0263362	10.8122	-0.09566	-0.0119908	-0.0258469	27.5976	2023-04-12 11:38:53	
47	-0.107739	0.0167594	10.817	-0.0953936	-0.013856	-0.0253139	27.7182	2023-04-12 11:38:58	
48	-0.112527	0.0263362	10.8242	-0.096193	-0.0135896	-0.024781	27.8124	2023-04-12 11:39:03	
49	-0.100556	0.0047884	10.7907	-0.0953936	-0.0133231	-0.0245145	27.8624	2023-04-12 11:39:08	
50	-0.107739	0.0335188	10.7883	-0.0943277	-0.0125237	-0.0258469	27.9094	2023-04-12 11:39:13	
51	-0.0885855	0.0646434	10.817	-0.09566	-0.0141225	-0.0242481	27.9506	2023-04-12 11:39:18	

API REST

Una API REST (Representational State Transfer) es un conjunto de reglas y protocolos que permite la interacción y comunicación entre sistemas informáticos a través de una arquitectura cliente-servidor. Se basa en el uso de HTTP (Hypertext Transfer Protocol) para la transferencia de los datos entre las aplicaciones.

Los servicios basados en REST se caracterizan por ser "stateless" o sin estado, lo que significa que no guardan información sobre las solicitudes previas de un cliente. En su lugar, cada solicitud realizada por el cliente incluye toda la información necesaria para llevar a cabo la acción deseada.

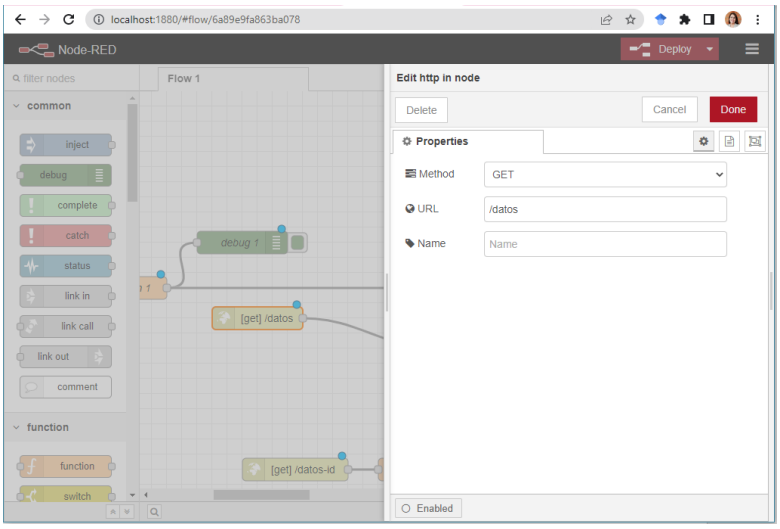
Una API REST se compone de recursos (como por ejemplo, "productos" o "clientes") que se identifican mediante una URL específica. Cada recurso puede ser accedido a través de una serie de verbos HTTP (GET, POST, PUT, DELETE) que permiten realizar distintas operaciones sobre el mismo.

- GET: Esta operación se utiliza para solicitar información de un recurso específico en el servidor. El cliente envía una solicitud GET al servidor con la URL del recurso que desea obtener, y el servidor responde con los datos correspondientes. Por ejemplo, una solicitud GET a la URL "https://api.misitio.com/productos" podría devolver una lista de todos los productos disponibles.
- POST: Esta operación se utiliza para crear un nuevo recurso en el servidor. El cliente envía una solicitud POST con los datos del nuevo recurso que desea crear, y el servidor lo procesa y devuelve una respuesta indicando si la creación fue exitosa o no. Por ejemplo, una solicitud POST a la URL "https://api.misitio.com/productos" podría crear un nuevo producto en la base de datos del servidor.
- PUT: Esta operación se utiliza para actualizar un recurso existente en el servidor. El cliente envía una solicitud PUT con los datos actualizados del recurso, y el servidor lo procesa y devuelve una respuesta indicando si la actualización fue exitosa o no. Por ejemplo, una solicitud PUT a la URL "https://api.misitio.com/productos/123" podría actualizar los datos del producto con ID 123.
- DELETE: Esta operación se utiliza para eliminar un recurso existente en el servidor. El cliente envía una solicitud DELETE con la URL del recurso que desea eliminar, y el servidor lo procesa y devuelve una respuesta indicando si la eliminación fue exitosa o no. Por ejemplo, una solicitud DELETE a la URL "https://api.misitio.com/productos/123" podría eliminar el producto con ID 123 de la base de datos del servidor.

API REST en Node-RED

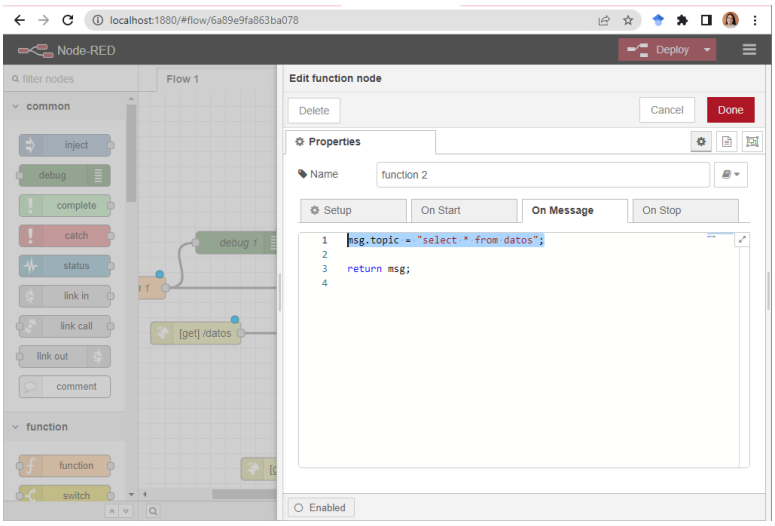
Para crear una API REST en node-RED que aaceda a los datos almacenados y permita acceder a ellos a través de operaciones GET hacemos lo siguiente:

1. Arrastramos un nodo http-in a la pantalla y lo configuramos de la siguiente manera:

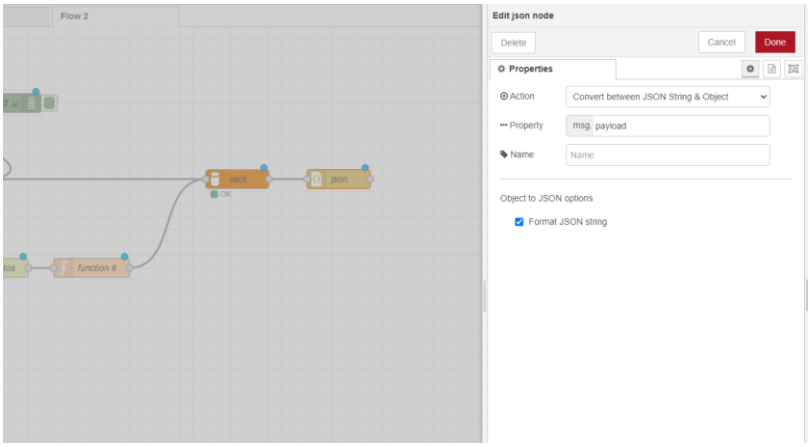


Aquí estamos configurando la operación y la ruta.

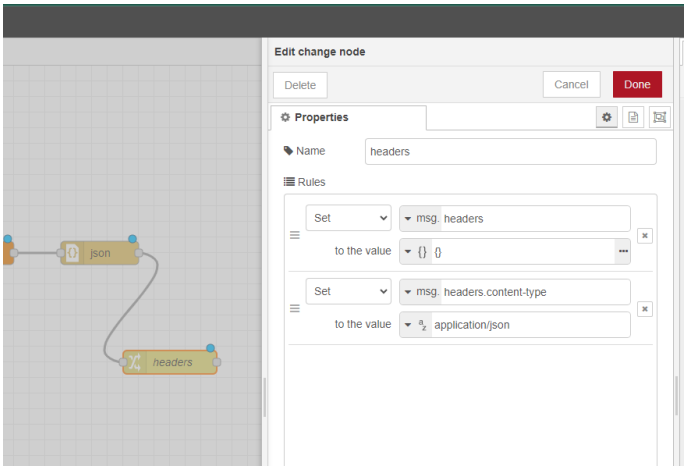
2. Colocamos una función que se encargará de ejecutar el query a la base de datos para consultar los datos almacenados. Esta función se conectará al nodo anterior y también deberá conectarse al nodo de la base de datos. Lo configuramos de la siguiente manera:



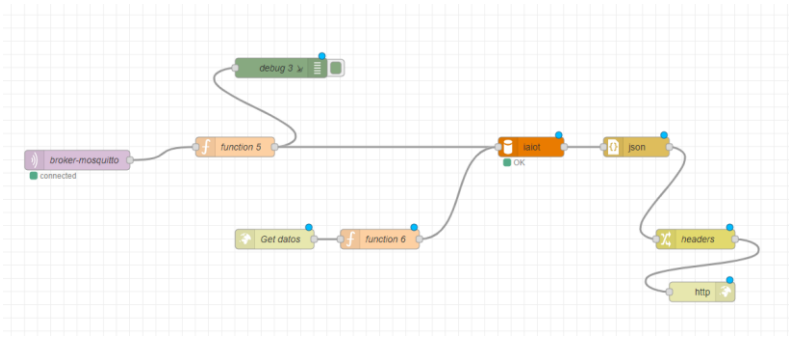
3. A la salida de la base de datos conectamos un nodo json que se encargará de dar el formato adecuado a los datos. Se configura de la siguiente manera:



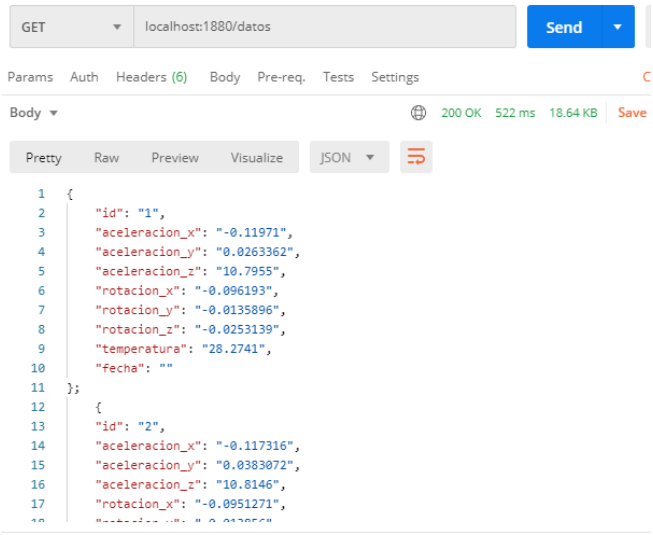
4. Añadimos un nodo change para configurar los headers de la respuesta http. Lo podemos configurar de la siguiente manera:



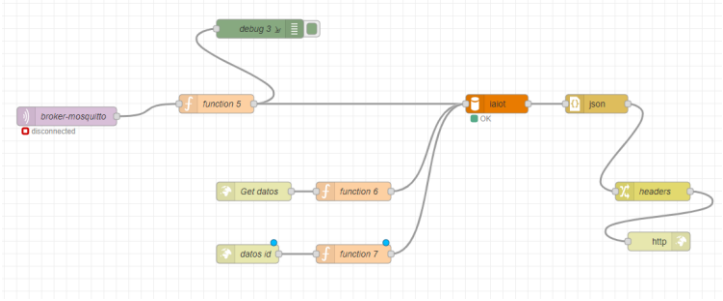
5. Añadimos el nodo de la respuesta http. Queda nuestra pantalla de la siguiente manera:

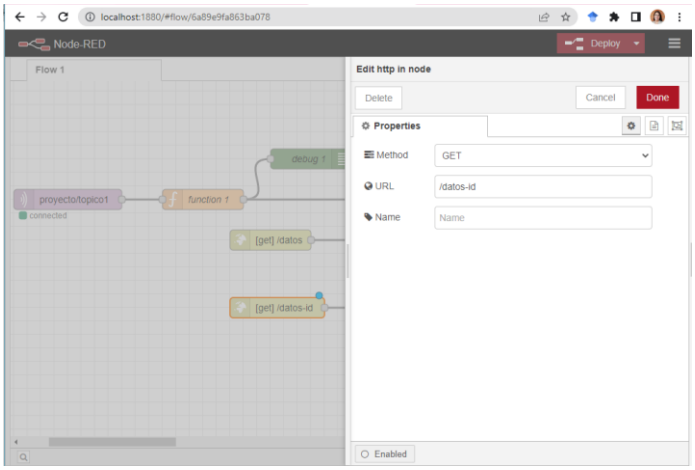


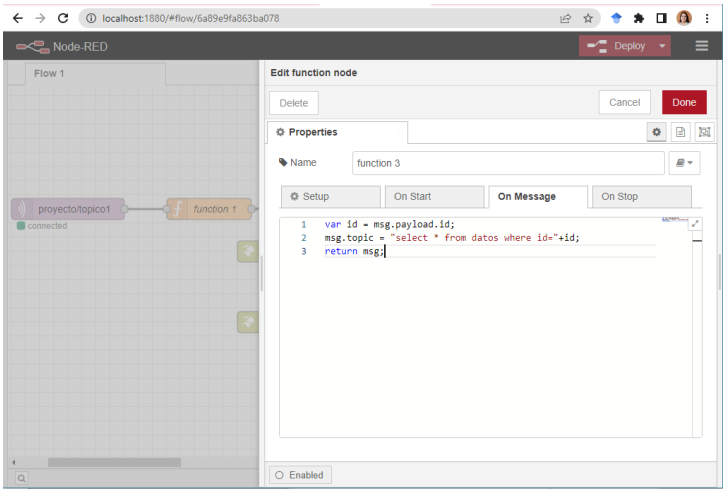
6. Para hacer las pruebas de nuestra API REST usamos Postman, que es un cliente http que permite hacer distintas peticiones a los servidores REST.



7. Podemos añadir otra ruta para hacer otro tipo de consulta. Para ello arrastramos otro nodo http-in y otra función, las conectamos al nodo de la base de datos y las configuramos de la siguiente manera:







8. Probamos con Postman