MQTT with MongoDB

This Node.js project connects to an MQTT broker, captures data from specific topics, and saves it into a MongoDB database. The Node.js project also provides API endpoints for specific queries. This project use Docker container.

Prerequisites

- Node.js (v20 or higher)
- MongoDB (v7 or higher)
- MQTT Broker (such as Mosquitto)
- Docker (optional)
- Docker Compose (optional)

If you are going to run the project with Docker, it is not necessary to install Node.js and MongoDB, the containers already use images with the necessary installations.

Clone

1. Clone the repository:

```
git clone https://github.com/gqferreira/mqtt-mongo.git
cd mqtt-mongo
```

1. Install dependencies (only if you intend to run without Docker):

```
npm install
```

1. Configure environment variables by creating a `.env` file in the root of the project with the following content:

```
MQTT_URL=mqtt://test.mosquitto.org
MQTT_PORT=1883
MQTT_TOPIC=mqtt-mongo
MQTT_USERNAME=
MQTT_PASSWORD=
MONGODB_URI=mongodb://db-telemetry:27017
MONGODB_DB=iot
```

1. Run project (only if you intend to run without Docker):

```
npm run dev
```

Project Structure

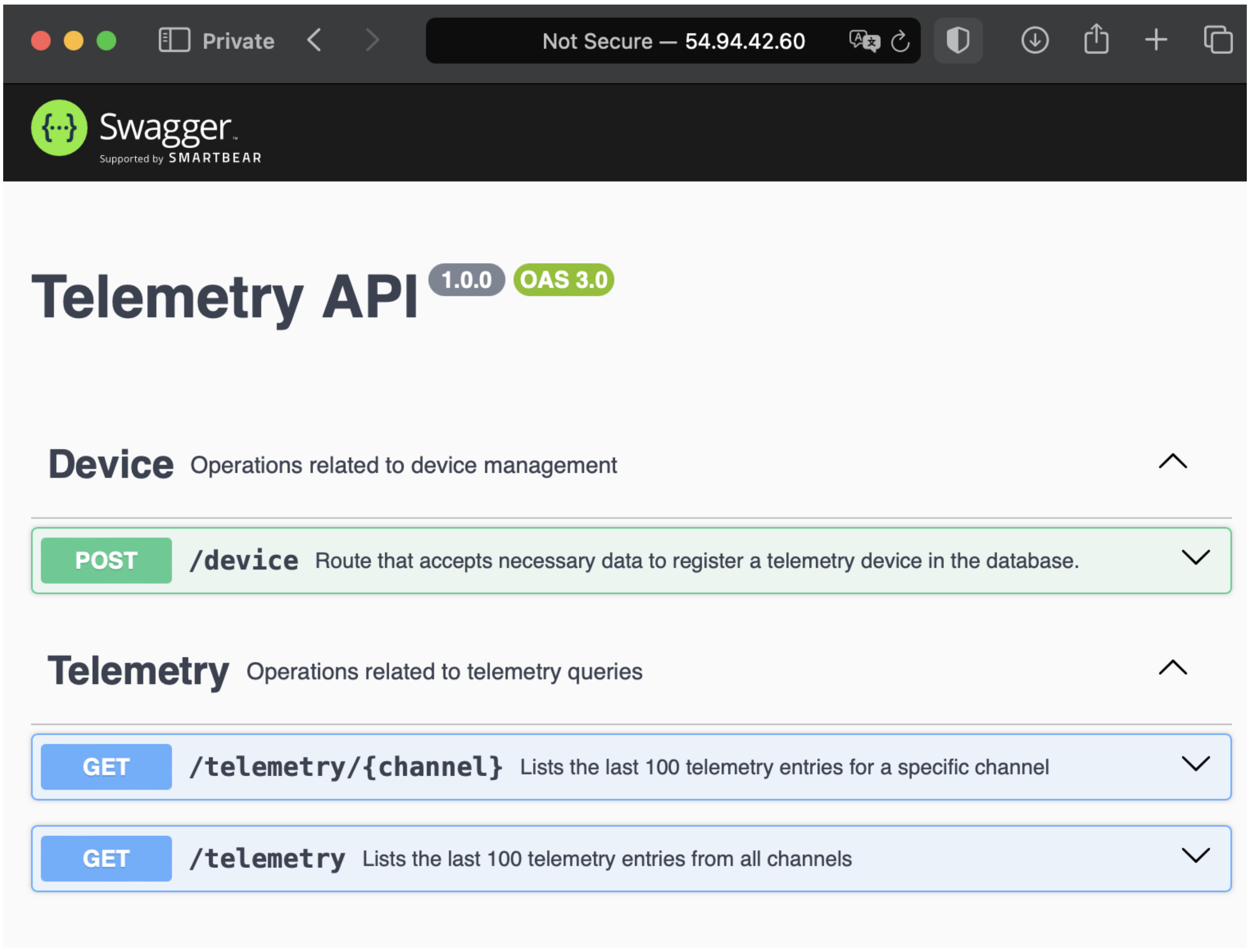
```
- mqtt-mongo/
  |- mqtt/
     |- mqttClient.js
  |- mongodb/
     |- mongoClient.js
  |- routes/
     |- telemetryRoutes.js
     |- deviceRoutes.js
  |- .env
  |- .gitignore
  |- app.js
  |- config.js
  |- docker-compose.yml
  |- Dockerfile
  |- LICENCE
  |- package-lock.json
  |- package.json
  |- swagger.js
```

Usage with Docker

To start the project, run:

```
sudo docker-compose -p telemetry up -d
```

If you are running docker compose on a personal computer locally, you can access the API documentation and interact with it through the following address: `localhost:3001/api-docs`. If you are running on a server, you must use the IP to access and perform the necessary firewall configurations.


Not Secure — 54.94.42.60

Telemetry API 1.0.0 OAS 3.0

Device Operations related to device management

POST /device Route that accepts necessary data to register a telemetry device in the database.

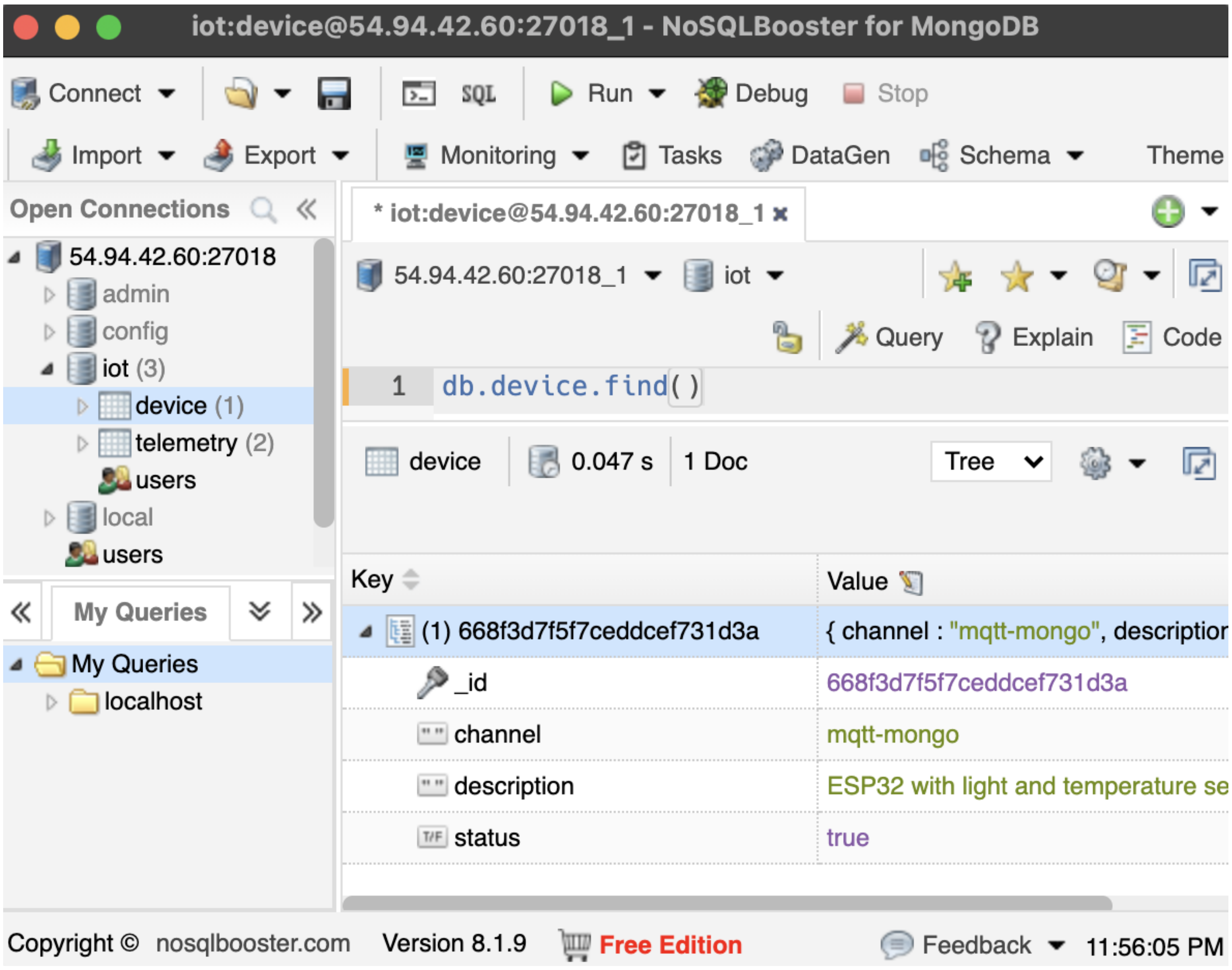
Telemetry Operations related to telemetry queries

GET /telemetry/{channel} Lists the last 100 telemetry entries for a specific channel

GET /telemetry Lists the last 100 telemetry entries from all channels

> ⚠ **Warning:** Before you can send messages to the brocker intended for a certain channel, you first need to use the API endpoint to create a device with that channel.

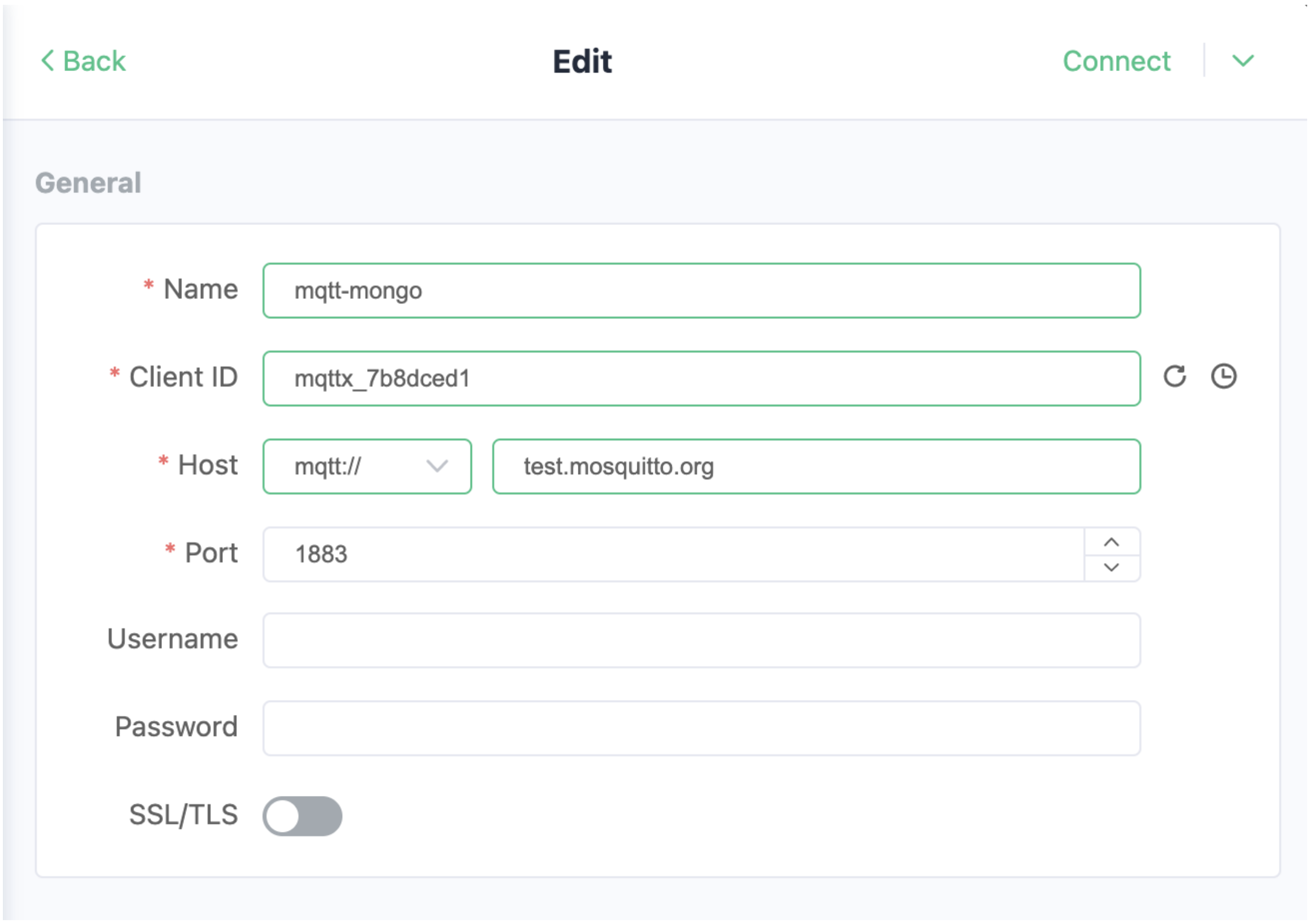
If you are running docker compose on a personal computer locally, you can connect to the database (e.g. with NoSQL Booster) at the following address: `localhost:27018`. If you are running on a server, you must use the IP to access and perform the necessary firewall configurations.



```
db.device.find()
```

Key	Value
(1) 668f3d7f5f7ceddcef731d3a	{ channel : "mqtt-mongo", descriptior
_id	668f3d7f5f7ceddcef731d3a
channel	mqtt-mongo
description	ESP32 with light and temperature se
status	true

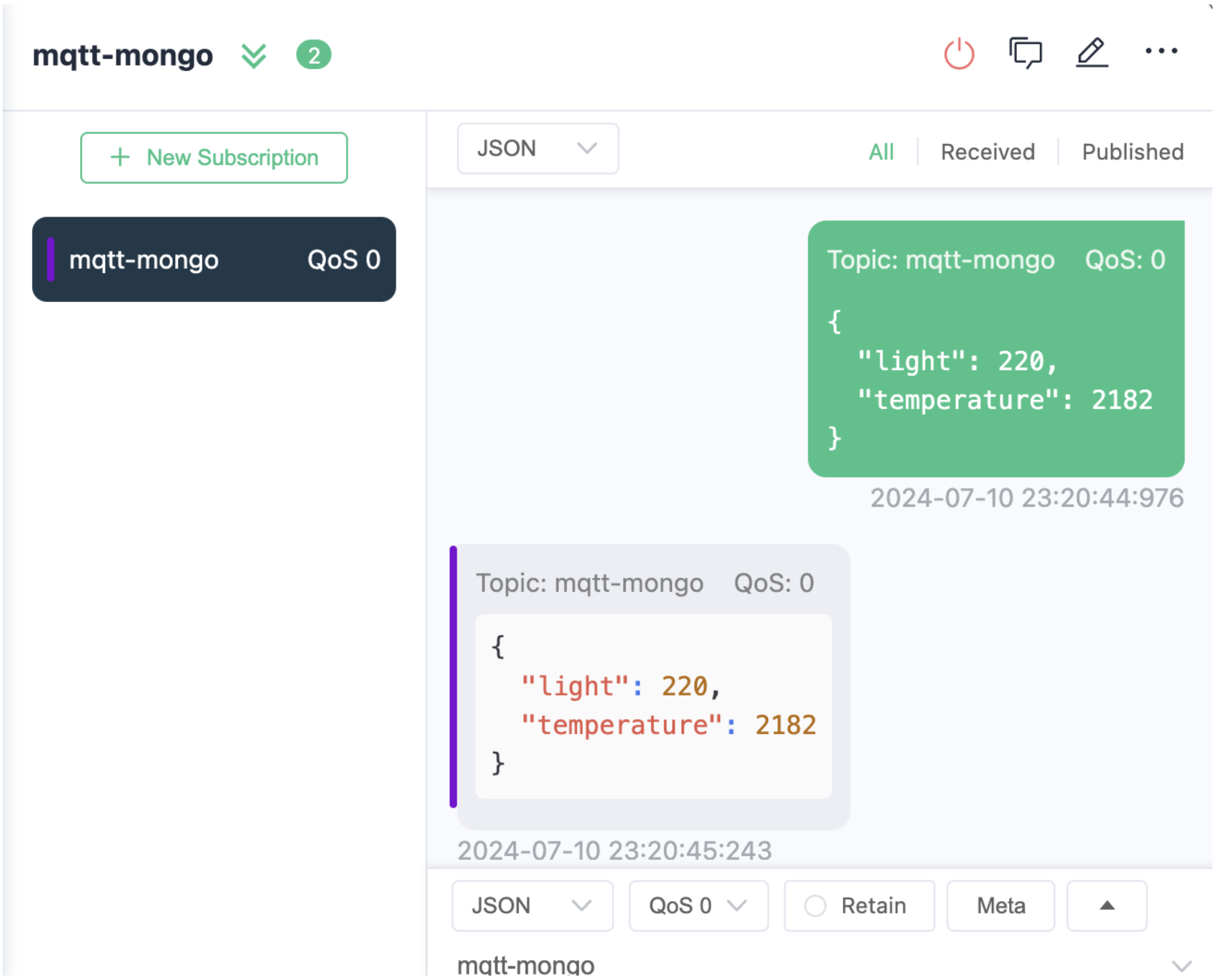
If you want to test and don't have an IoT device, you can use the MQTTX app to create a connection to the Mosquitto test brocker and send messages to the channel registered to your device.



< Back **Edit** Connect

General

* Name	mqtt-mongo
* Client ID	mqttx_7b8dced1
* Host	mqtt:// test.mosquitto.org
* Port	1883
Username	
Password	
SSL/TLS	off



mqtt-mongo 2

+ New Subscription

mqtt-mongo QoS 0

JSON All Received Published

Topic: mqtt-mongo QoS: 0

```
{
  "light": 220,
  "temperature": 2182
}
```

2024-07-10 23:20:44:976

Topic: mqtt-mongo QoS: 0

```
{
  "light": 220,
  "temperature": 2182
}
```

2024-07-10 23:20:45:243

JSON QoS 0 Retain Meta

mqtt-mongo

```
{
  "light": 220,
  "temperature": 2182
}
```

To stop the project, run:

```
sudo docker-compose -p telemetry down
```

To rebuild after changes if you change Dockerfile (need start again after):

```
sudo docker-compose -p telemetry build
```

If you wish, you can access the application container in interactive mode and use PM2 to monitor the application logs:

```
docker exec -it app-telemetry bash
pm2 monit
```

If you wish, you can access the database container in interactive mode and use mongosh to query the collections documents.

```
docker exec -it db-telemetry bash
mongosh
use iot
```

Database:

The telemetry collection in the database should be named `telemetry` and have the following structure:

```
use iot

db.telemetry.insertOne(
    {
        "date": ISODate('2024-06-12T10:09:00Z'),
        "light": 3500,
        "temperature": 1900,
        "device": {
            "$ref": "device",
            "$id": ObjectId("000000000000000000000001"),
            "$db": "iot"
        }
    }
);
```

```
db.device.insertOne(
    {
        "channel": 'mqtt-mongo',
        "description": 'Environmental light and temperature monitoring system',
        "status": true
    }
);
```