

# Solution for Gateway project

## Content

Introduction .....	2
Project Structure .....	2
Resources .....	3
Steps to Run the project .....	3
Steps to Test the project.....	4
Internal Test .....	6

# Introduction

This is the solution for the project “Gateway” and has been made using this environment.

OS: Windows 10.

Spring Boot Version: 2.4.0

Java: 1.8

In Memory DB: H2, Very fast, open source, Small footprint: around 2 MB jar file size.

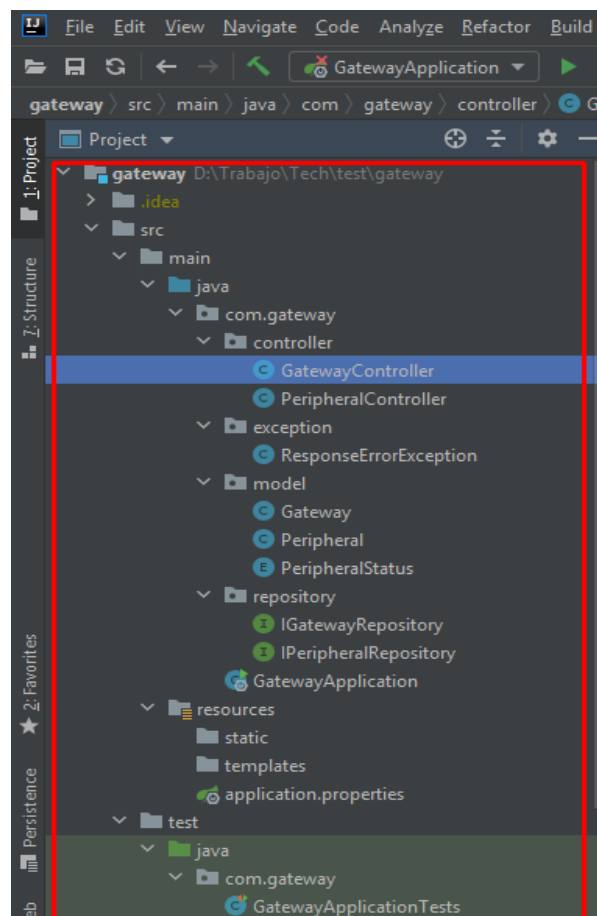
IDE: IntelliJIDEA: 2020.2

## Project Structure

**-Model:** include Gateway and Peripheral classes and also PeripheralStatus enum.

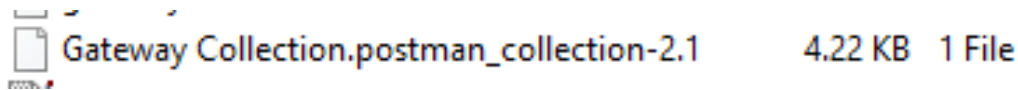
**-Repository:** include IGatewayRepository and IPeripheralRepository.

**-Controller:** include GatewayController and PeripheralController



# Resources

1. Using H2 database there is not additional script to create database.
2. To test Gateway API please import the Postman collection with all endpoint and data.



## Steps to Run the project

1. Go to the project folder and extract the zip file.
2. Open a console and go to the project folder
3. Run the command **mvn package** this step also run 2 internal tests

```
D:\Trabajo\Tech\test\gateway>mvn package
[INFO] Scanning for projects...
[INFO]
[INFO] -----< com.java:gateway >-----
[INFO] Building gateway 0.0.1-SNAPSHOT
[INFO] -----[ jar ]-----
[INFO]
```

```
2020-11-15 16:18:13.118 INFO 12416 --- [extShutdownHook] com.zaxxer.hikari.HikariDataSource : HikariPool-1
[INFO]
[INFO] Results:
[INFO]
[INFO] Tests run: 2, Failures: 0, Errors: 0, Skipped: 0
[INFO]
[INFO]
[INFO] --- maven-jar-plugin:3.2.0:jar (default-jar) @ gateway ---
[INFO]
[INFO] --- spring-boot-maven-plugin:2.4.0:repackage (repackage) @ gateway ---
[INFO] Replacing main artifact with repackaged archive
[INFO]
[INFO] BUILD SUCCESS
[INFO]
[INFO] Total time: 18.144 s
[INFO] Finished at: 2020-11-15T16:18:14-05:00
[INFO]
```

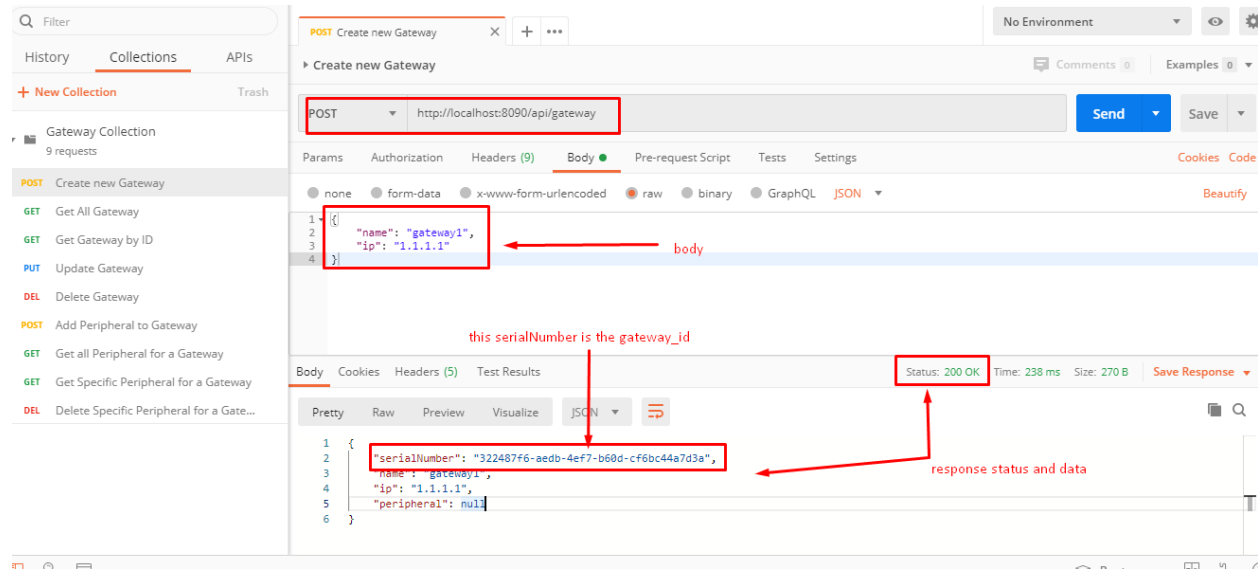
4. Run the command **mvn spring-boot:run**

```
D:\Trabajo\Tech\test\gateway>mvn spring-boot:run
[INFO] Scanning for projects...
[INFO]
[INFO] -----< com.java:gateway >-----
[INFO] Building gateway 0.0.1-SNAPSHOT
[INFO]
```

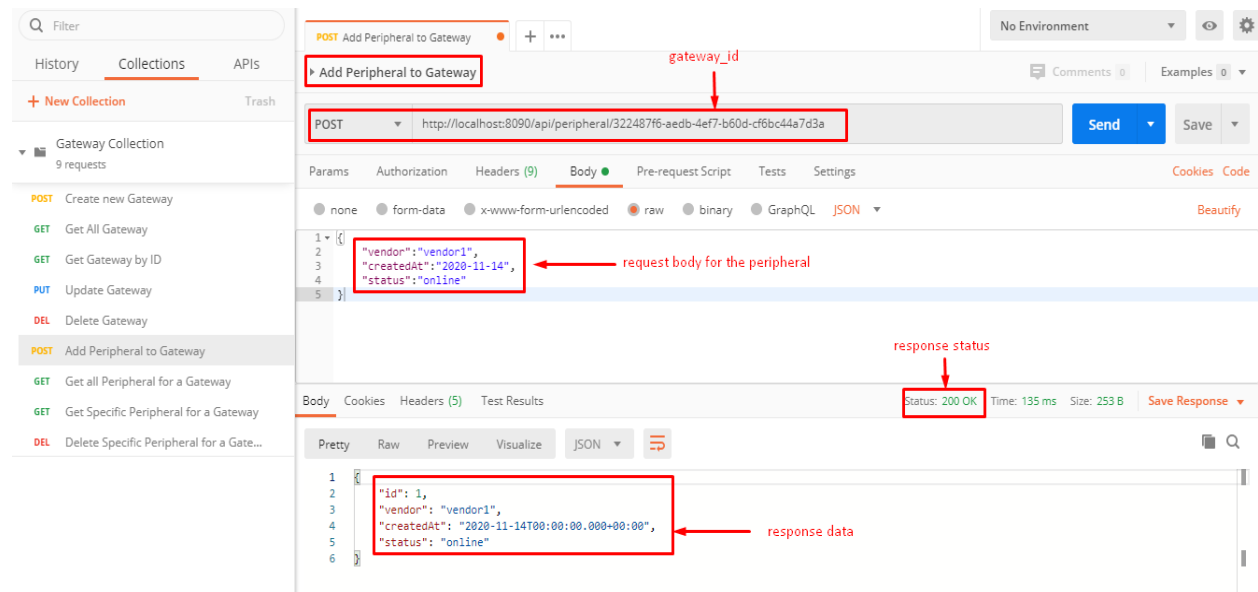
# Steps to Test the project

After the project is running and postman has the Gateway collection imported then proceed to test endpoints, in this doc only is included few tests.

## 1-Create new Gateway



## 2-Add Peripheral to a Gateway



### 3-Get all Gateway

The screenshot shows a Postman interface with a GET request to `http://localhost:8090/api/gateway`. The response status is `200 OK`. The response body is a JSON array containing one gateway object. Red boxes highlight the `gateway` key in the JSON and the `peripheral` array within it. A red arrow points to the `peripheral` array with the label "peripheral for this gateway".

```
{
  "serialNumber": "322487f6-aedb-4ef7-b60d-cf6bc44a7d3a",
  "name": "gateway1",
  "ip": "1.1.1.1",
  "peripheral": [
    {
      "id": 1,
      "vendor": "vendor1",
      "createdAt": "2020-11-14T00:00:00.000+00:00",
      "status": "online"
    }
  ]
}
```

### 4-One gateway cannot have more than 10 peripherals, error is thrown

The screenshot shows a POST request to `http://localhost:8090/api/peripheral/322487f6-aedb-4ef7-b60d-cf6bc44a7d3a` with a status of `406 Not Acceptable`. The request body is a JSON object representing a peripheral. The response body is a JSON object containing an error message. Red boxes highlight the `id` field in the request body and the `error` field in the response body.

Request Body:

```
{
  "vendor": "vendor11",
  "createdAt": "2020-11-14",
  "status": "online"
}
```

Response Body:

```
{
  "timestamp": "2020-11-15T21:49:28.009+00:00",
  "status": 406,
  "error": "Not Acceptable",
  "message": "",
  "path": "/api/peripheral/322487f6-aedb-4ef7-b60d-cf6bc44a7d3a"
}
```

# Internal Test

- There are 2 internal test using Junit, one for Add Gateway and Add Peripheral to a Gateway, you can run this test inside de IDE.

