My Signals project

Aboubacar BAH
Taoufiq Boussraf

Objectives

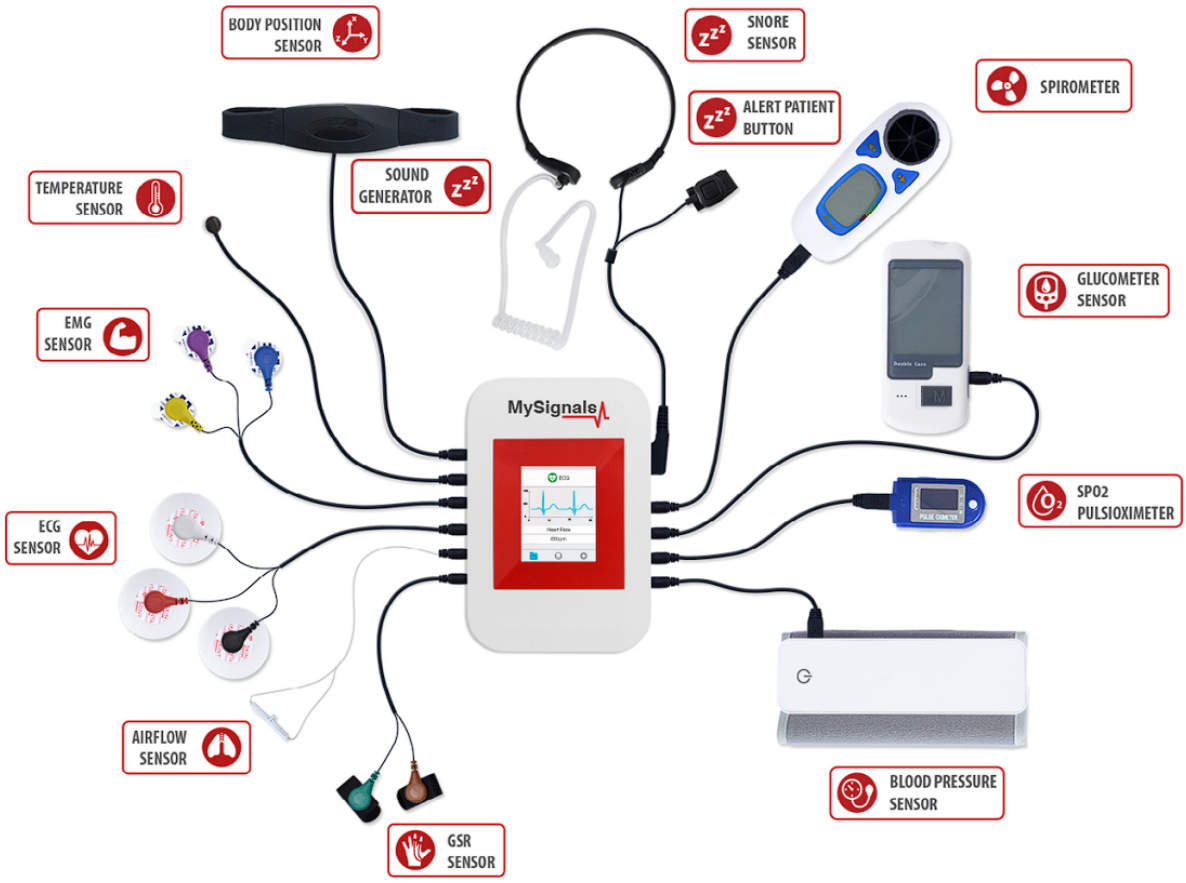
This project involves collecting health data in real time from a "MySignals" IoT device to which several sensors are connected. Once collected, this data needs to be stored in a personal cloud API to be visualised in the form of graphs.

The main objectives are :

1. **Real-Time Data Collection:** Implement a solution to collect real-time data from health sensors connected to the MySignals IoT device.
2. **Integration of Multiple Sensors:** Enable the connection and integration of various types of health sensors such as ECG, EEG, temperature, blood pressure, glucose level, etc.
3. **Storage in Personal Cloud:** Develop a cloud API to securely store and manage collected data in a scalable manner.
4. **Data Visualization:** Create visualization features for data in the form of graphs and tables, allowing users to monitor and analyze collected health data.
5. **User-Friendly Interface:** Design a user-friendly interface that enables users to log in, view their data, and add new sensors data to the system.
6. **Data Security:** Implement security measures to protect sensitive health data stored in the personal cloud, using robust encryption and authentication methods.
7. **Multi-Platform Compatibility:** Ensure compatibility with different platforms, including mobile devices and computers, allowing users to access data from any device.

8. **Notifications and Alerts:** Integrate notification and alert features to inform users of abnormal fluctuations or critical situations in health data.

Specifications



1. **MySignals IoT Configuration:** Configure and set up the MySignals IoT device by connecting appropriate sensors and configuring communication settings.

2. **Cloud API Development:** Design and establish a secure cloud API for storing and managing collected health data.

3. **Android App Development:** Create an attractive and intuitive user interface that allows users to log in, connecting to MySignals Device, view health data, and collect sensors data from the device.

4. **Data Visualization:** Develop interactive graphs and charts for visualizing health data provided by The Cloud API (By using an android App or a Web Application).

5. **Cloud Connectivity Integration:** Implement bidirectional communication between the MySignals device and the cloud API for real-time data transmission.

6. **Security and Authentication:** Implement security measures, including data encryption and authentication mechanisms, to ensure data confidentiality and integrity.

7. **Notifications and Alerts:** Integrate notification and alert mechanisms to inform users of important events related to their health data.

8. **Testing and Validation:** Perform comprehensive testing to ensure that data collection, storage, visualization, and management functions correctly and securely.
9. **Documentation and Training:** Provide a documentation on how to use the application, along with user training on adding sensors, viewing data, and interacting with the user interface.
10. **Deployment and Maintenance:** Deploy the cloud application and ensure ongoing maintenance to ensure optimal performance and updates as needed.

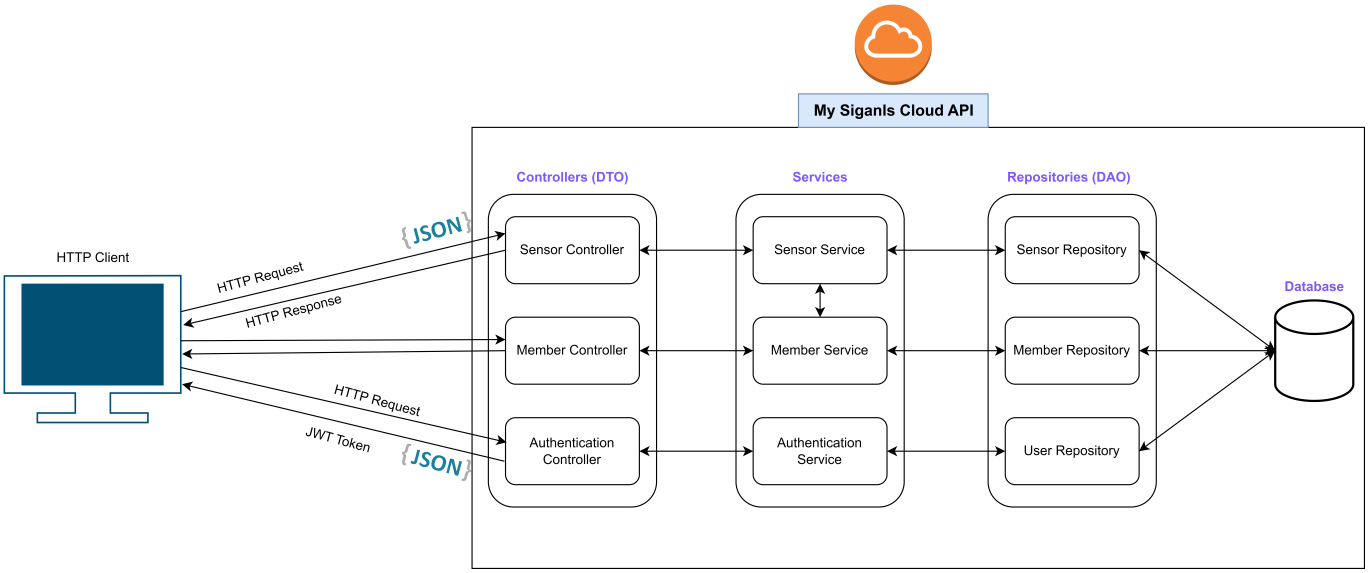
API Development

The API is based SpringBoot API

Tools

- IntelliJ IDEA, Visual Studio Code...
- Java (Version >= 8)
- Maven

Architecture



Services

In the SpringBoot API, we have three services :

Sensors Service

The Sensors Service is responsible for managing the data related to various sensors used in the system. It implements CRUD (Create, Read, Update, Delete) operations to handle sensor information. This service allows you to perform the following actions:

- **Create:** Add new sensor information to the database, including details like sensor type, data units, and any other relevant metadata.
- **Read:** Retrieve sensor information from the database based on different criteria, such as sensor type or sensor ID.

- **Update:** Modify existing sensor information, such as updating sensor metadata or changing sensor properties.
- **Delete:** Remove sensor information from the database when a sensor is no longer needed or is replaced.

The Sensors Service plays a crucial role in storing and managing the characteristics and properties of different sensors within the system.

Member Service

The Member Service handles the management of member data within the system. Similar to the Sensors Service, it provides CRUD operations for member-related information. This service allows you to perform the following actions:

- **Create:** Add new member profiles to the system, including details like name, surname, profile picture, description, height, weight, and other relevant attributes.
- **Read:** Retrieve member profiles from the database based on different criteria, such as member ID or specific attributes.
- **Update:** Modify existing member profiles, including updating personal information or adjusting attributes like height and weight.
- **Delete:** Remove member profiles from the system when necessary.

The Member Service is essential for maintaining records of individuals associated with the health data and providing the necessary details for data analysis and visualization.

Authentication Service (JWT Authentication)

The Authentication Service is responsible for handling user authentication and authorization using JSON Web Tokens (JWT). It provides the necessary functionality to securely manage user access to the API endpoints. This service includes the following features:

- **User Registration:** Allow users to create accounts by providing necessary details, such as username, email, and password.
- **User Login:** Authenticate users by verifying their credentials and generating JWT tokens upon successful login.
- **Token Validation:** Verify the authenticity and integrity of incoming JWT tokens to ensure that users have valid access.
- **Authorization:** Control user access to specific endpoints and resources based on their roles and permissions stored within the JWT.
- **Token Refresh:** Provide the ability to refresh expired tokens, enabling users to extend their session without the need for frequent login.

The Authentication Service ensures that only authorized users can access the API endpoints and perform actions, contributing to the security and integrity of the system.

Overall, these three services work together to provide a comprehensive and secure platform for managing sensor data, member profiles, and user authentication within your Spring Boot API.

Database Management System (DBMS)

You can change the DBMS in spingboot configuration file `application.properties`

H2 Database (in developpment)

H2 is the choosen database for developpment mode :

- Database configuration for H2

```
# H2 Database configuration
spring.datasource.url=jdbc:h2:mem:testdb
spring.datasource.username=sa
spring.datasource.password=
spring.datasource.driver-class-name=org.h2.Driver
# Hibernate configuration
spring.jpa.database-platform=org.hibernate.dialect.H2Dialect
spring.jpa.show-sql=true
spring.jpa.hibernate.ddl-auto=create
# H2 console configuration
spring.h2.console.enabled=true
spring.h2.console.path=/h2-console
```

POSTGRES Database (deployment)

In deployment mode, we use Postgres as DBMS

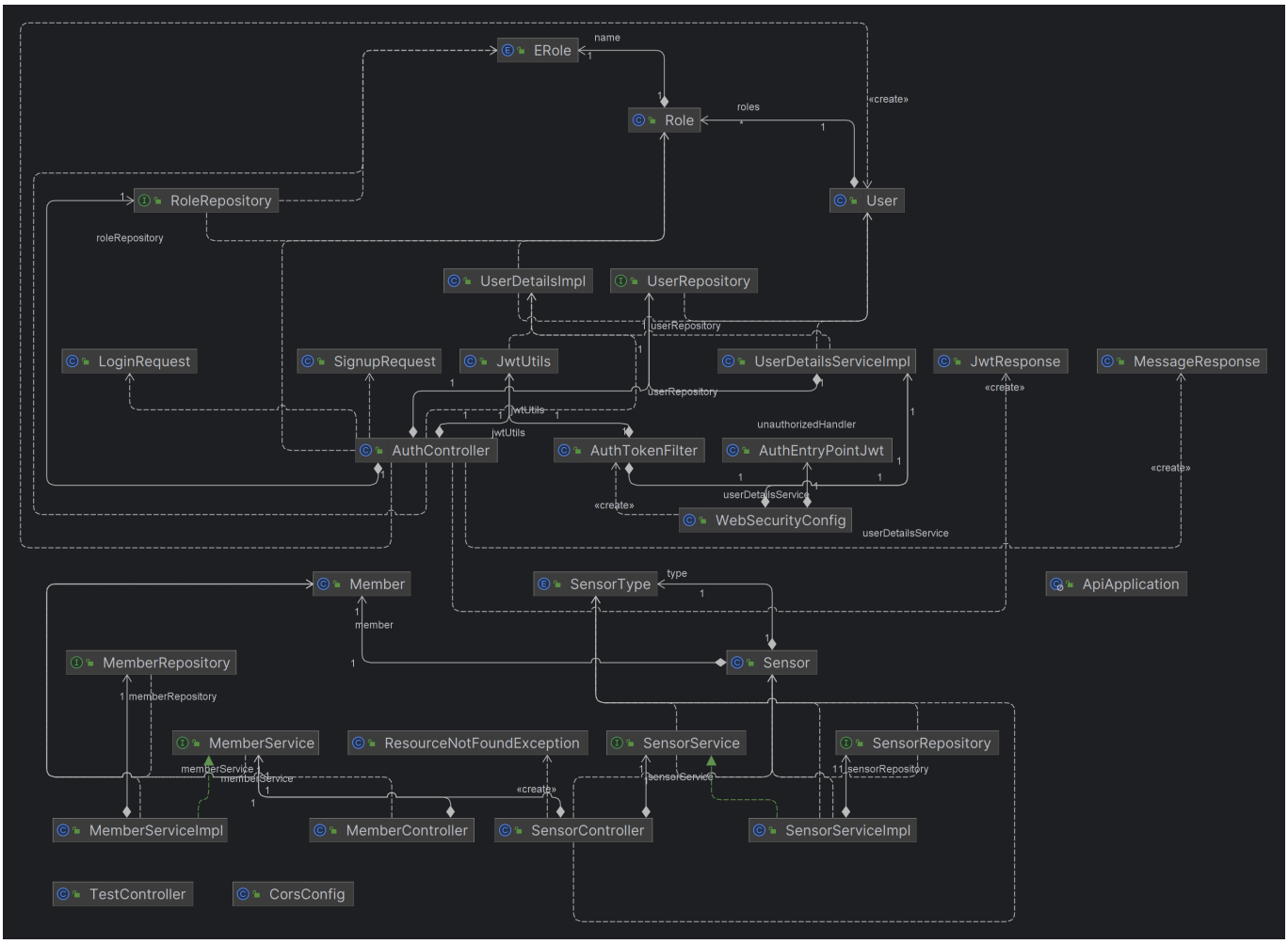
- Database configuration for POSTGRES

```
# Postgres Database
spring.datasource.url= jdbc:postgresql://database_service:5432/sensorsdb
spring.datasource.username= admin
spring.datasource.password= adminadmin
spring.jpa.properties.hibernate.jdbc.lob.non_contextual_creation= true
spring.jpa.properties.hibernate.dialect= org.hibernate.dialect.PostgreSQLDialect
# Hibernate ddl auto (create, create-drop, validate, update)
spring.jpa.show-sql=true
spring.jpa.hibernate.ddl-auto=update
```

> **IMPORTANT** : if H2 database is not used, we should insert data in the `roles` table of our database by excuting these lines (this task is automaticaly done, if H2 Database is used, so we can skip this step) :

```
INSERT INTO roles(name) VALUES('ROLE_USER');
INSERT INTO roles(name) VALUES('ROLE_MODERATOR');
INSERT INTO roles(name) VALUES('ROLE_ADMIN');
```

Classes Diagram



Services deployment via `Docker`

Docker compose file `docker-compose.yml`

- To deploy the services, execute this command in your terminal where your `docker-compose.yml` is located :

```
docker compose down && docker compose build && docker compose up -d
```

- To verify if services are launch, use `docker compose ps` command :

```
docker compose ps
NAME                COMMAND                  SERVICE             STATUS
PORTS
api_services        "/bin/sh -c 'java -j…"   api_services        running
database_service    "docker-entrypoint.s…"   database_service    running
0.0.0.0:5432->5432/tcp
pgadmin4            "/entrypoint.sh"         pgadmin             running
443/tcp, 0.0.0.0:5050->80/tcp
proxy_service       "httpd-foreground"       proxy_service       running
0.0.0.0:8000->80/tcp
```

Github Pipeline (TODO)

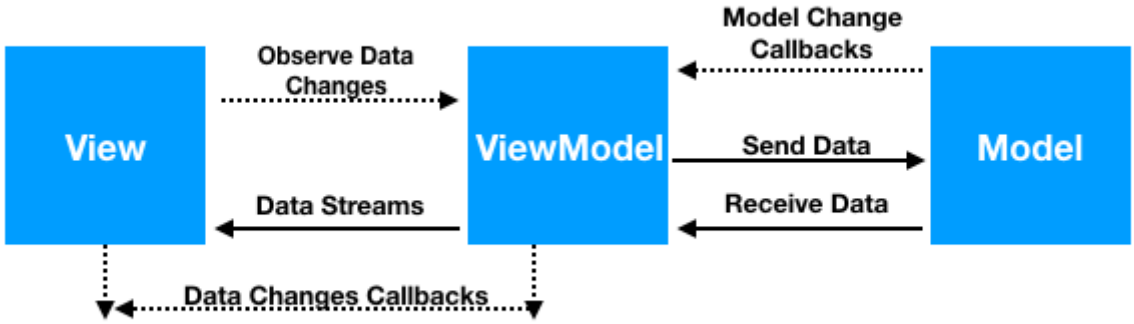
Android Application

Requirements

- IntelliJ IDEA, Android Studio
- Java JDK & JRE (Version >= 8)
- Android Device(android minimum SDK >= 4.3.0)

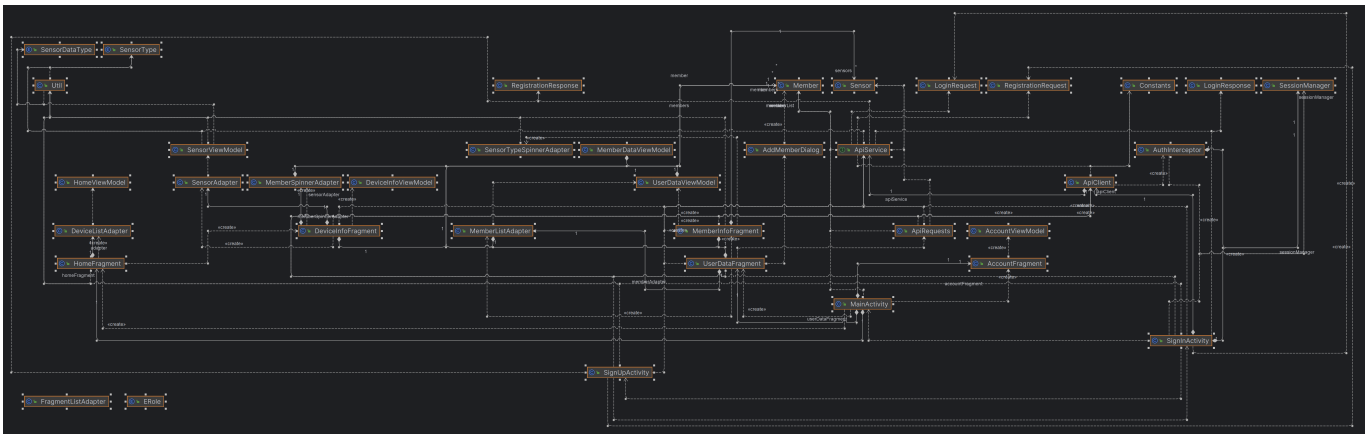
Architecture

The design Pattern used for android developement is MVVM:



- M : Model
- V : View
- VM : ViewModel

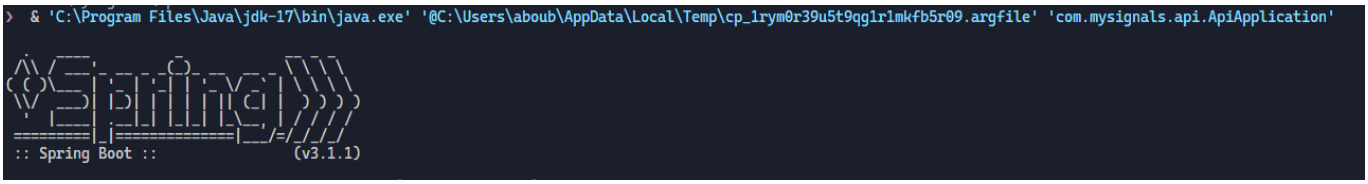
Classes Diagram



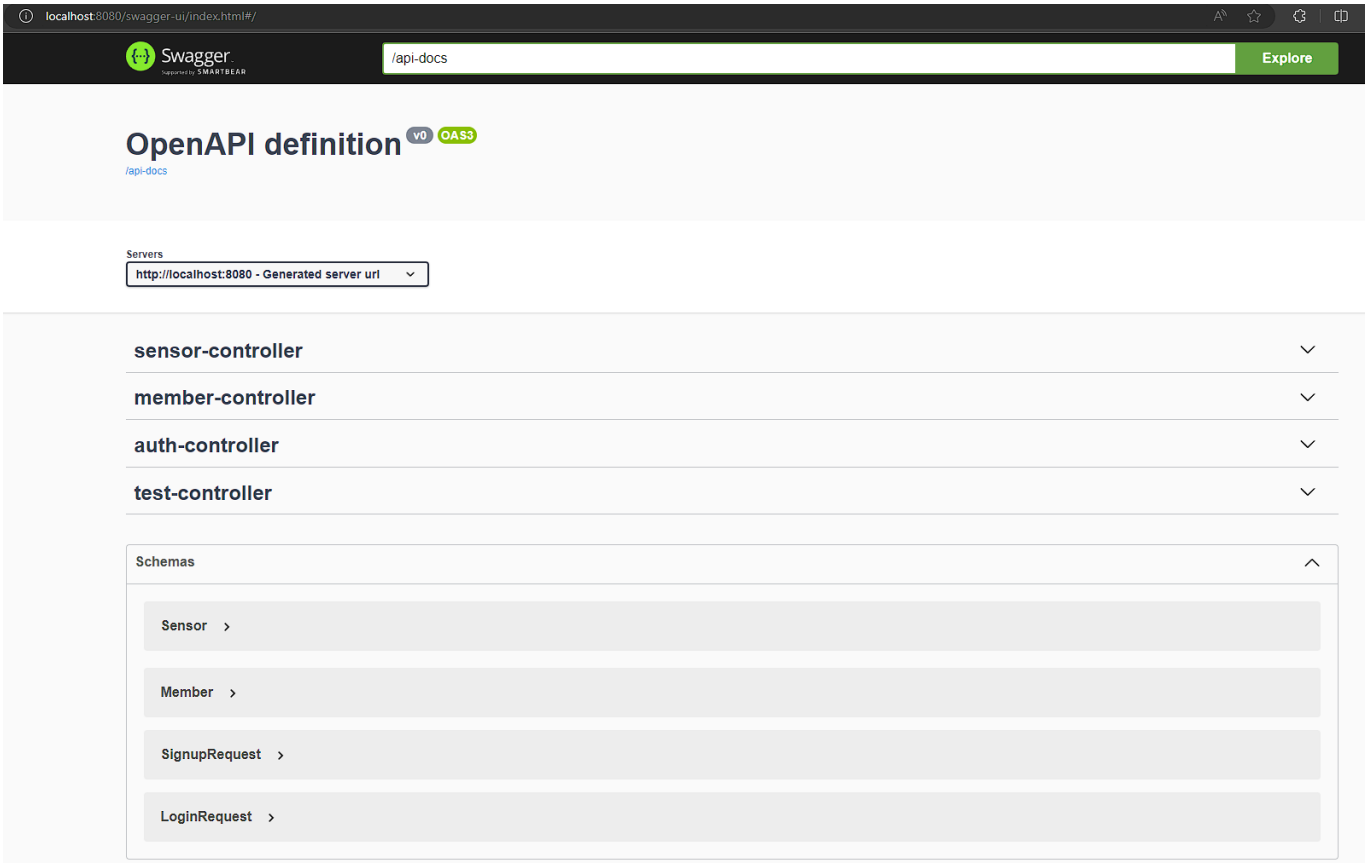
Demo

API

- Launch the API

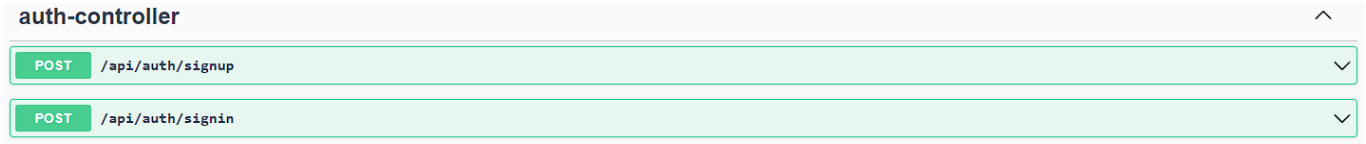


- API Web Inteface : http://localhost:8080/swagger-ui/index.html#/



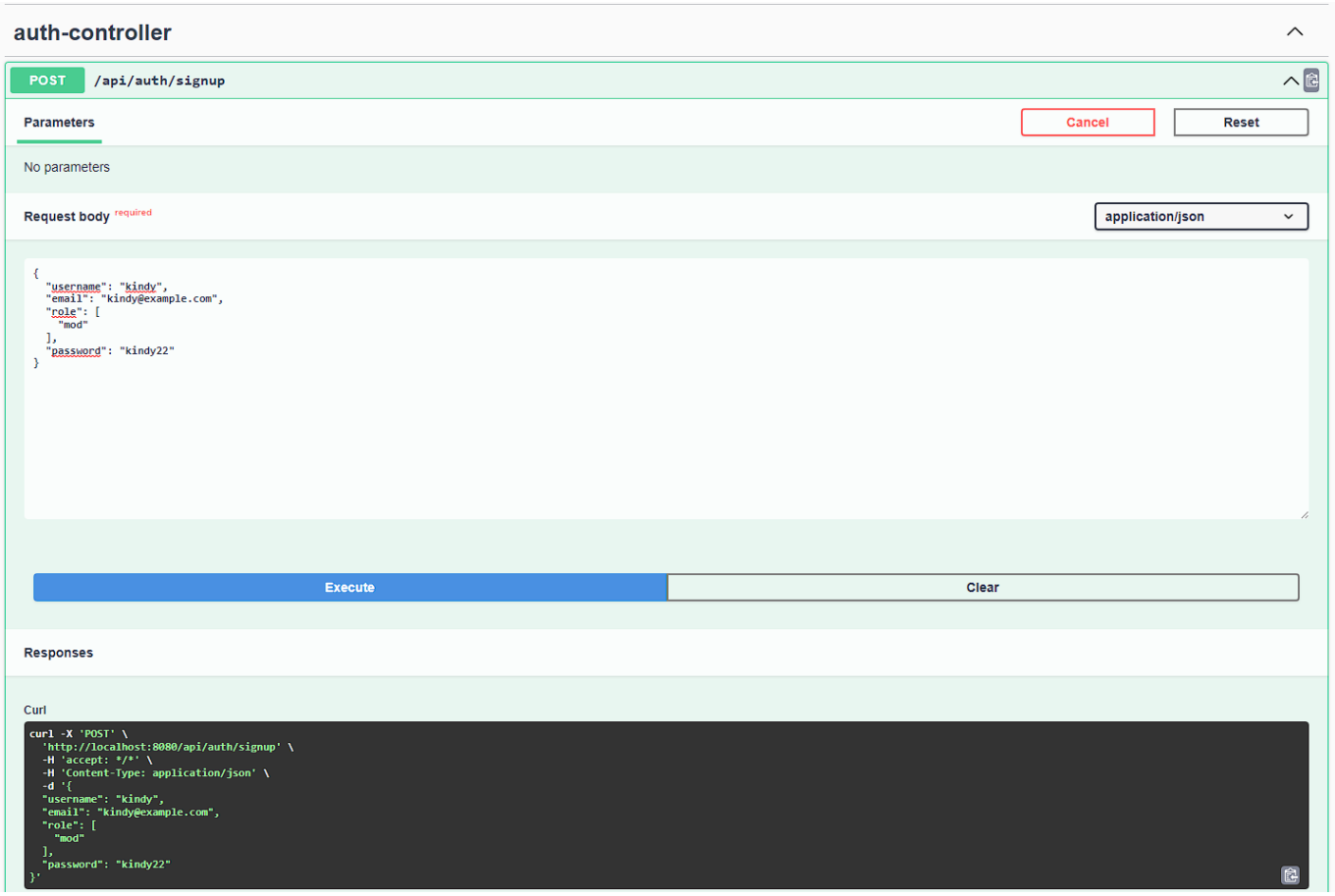
Authentication Service

- Authentication Service `Endpoints` :

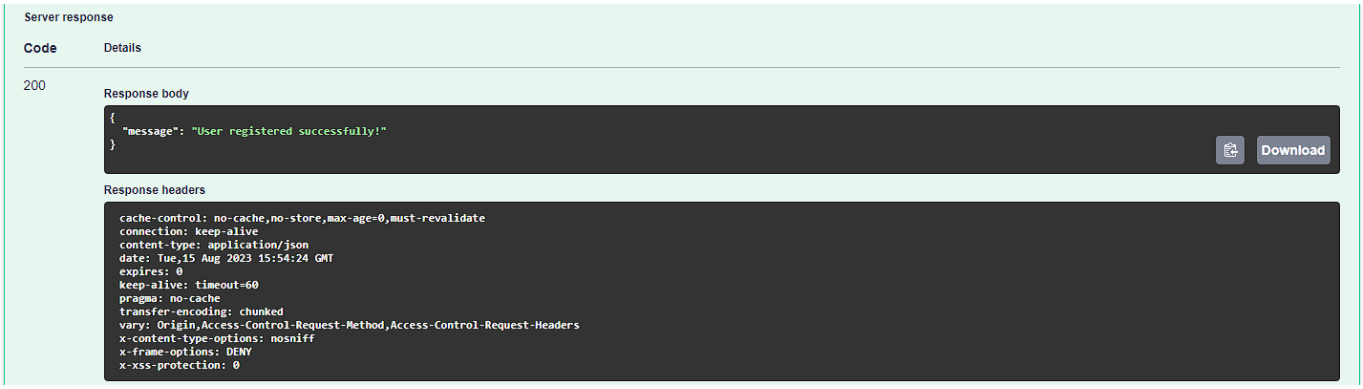


1. Registration

- Client Request

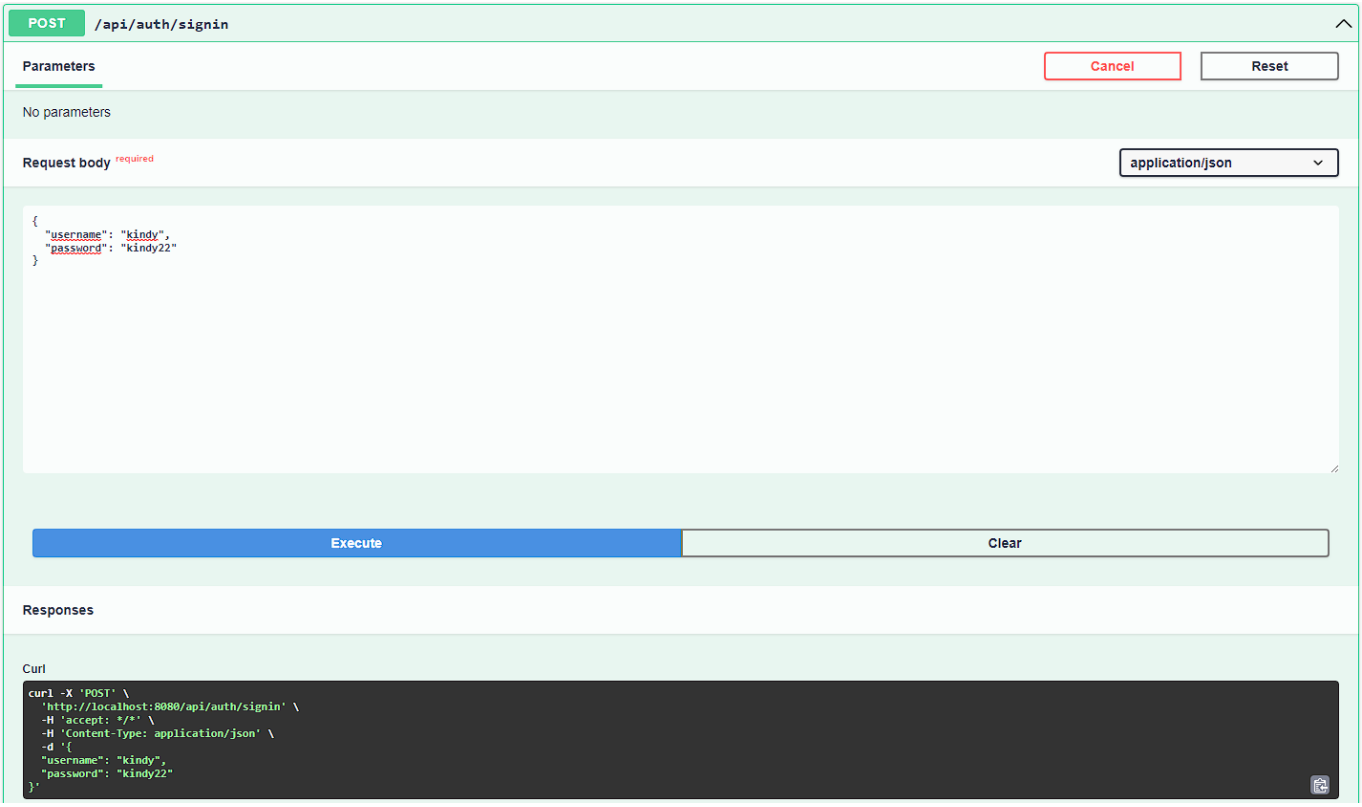


- API Response

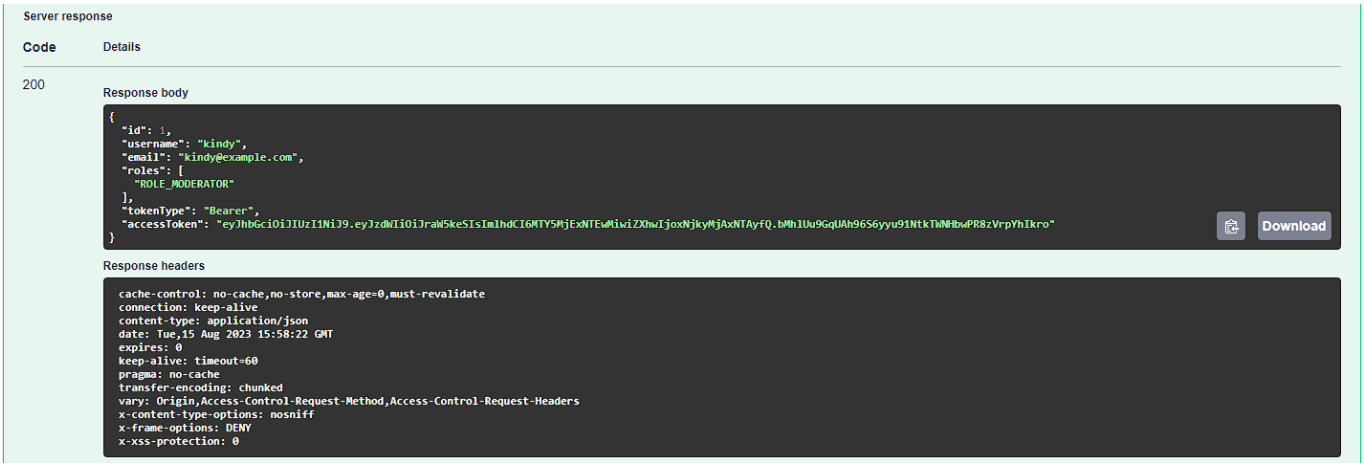


1. Login

- Client Request

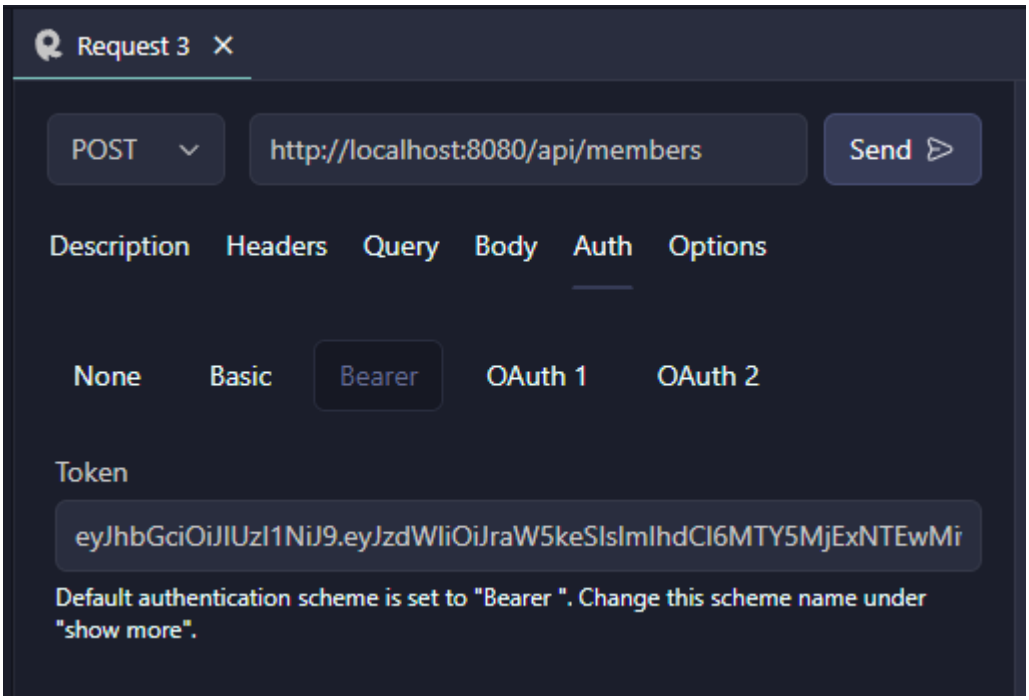


- API Response



How to use the user `AccesToken` to make HTTP Request to the API

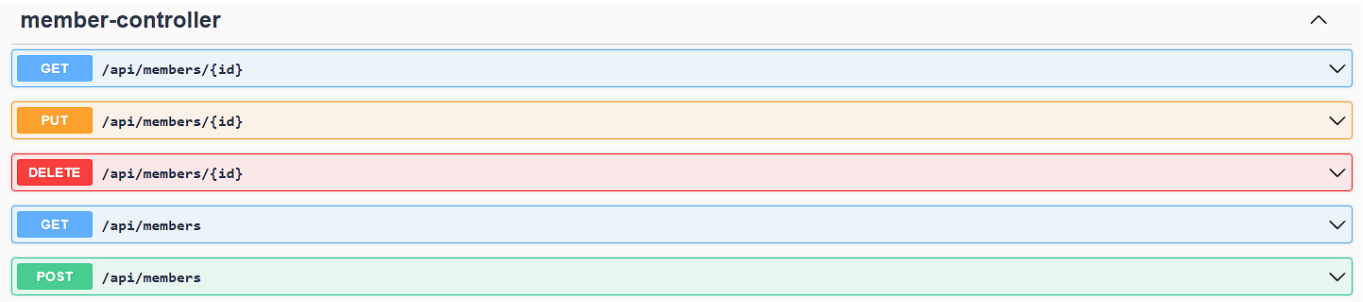
- Add the `AccesToken` to the HTTP `Authorization` header :



Now you can start making request to the API.

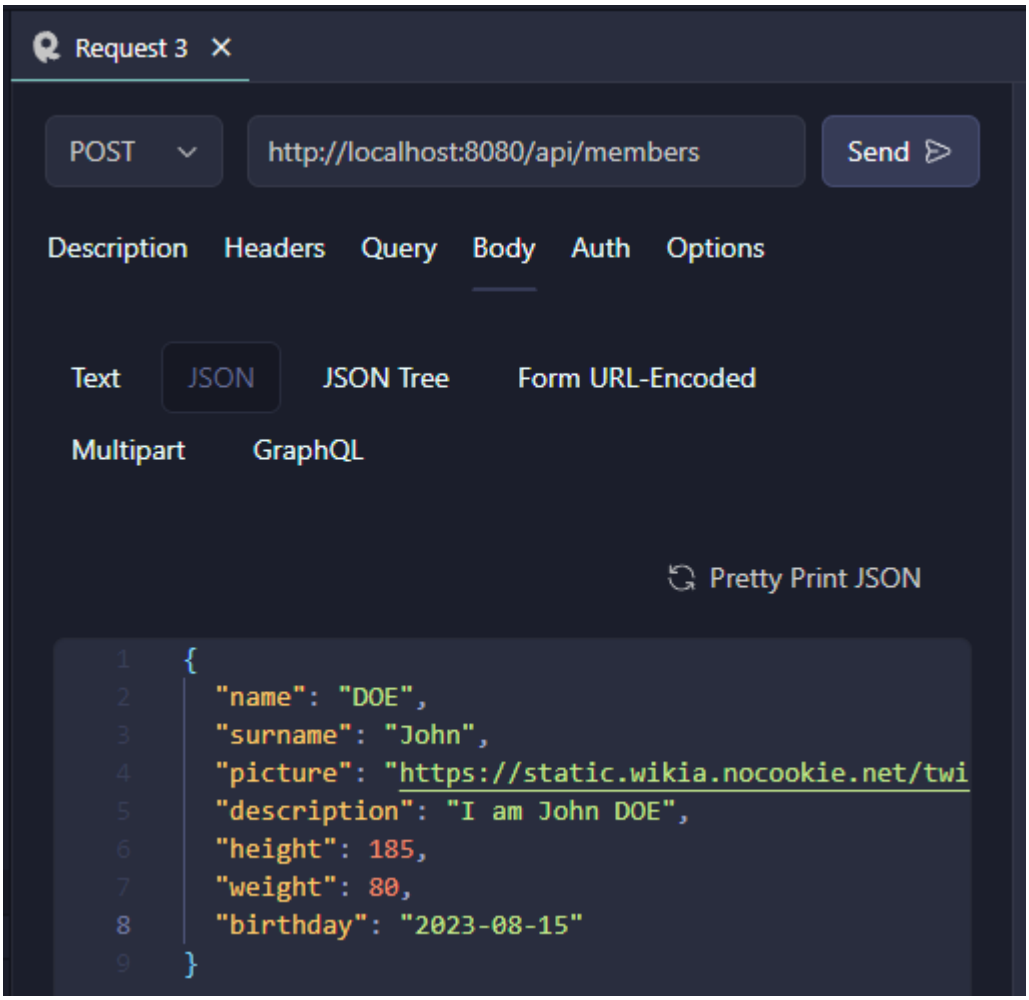
Member Service

- Member Service `Endpoints` :



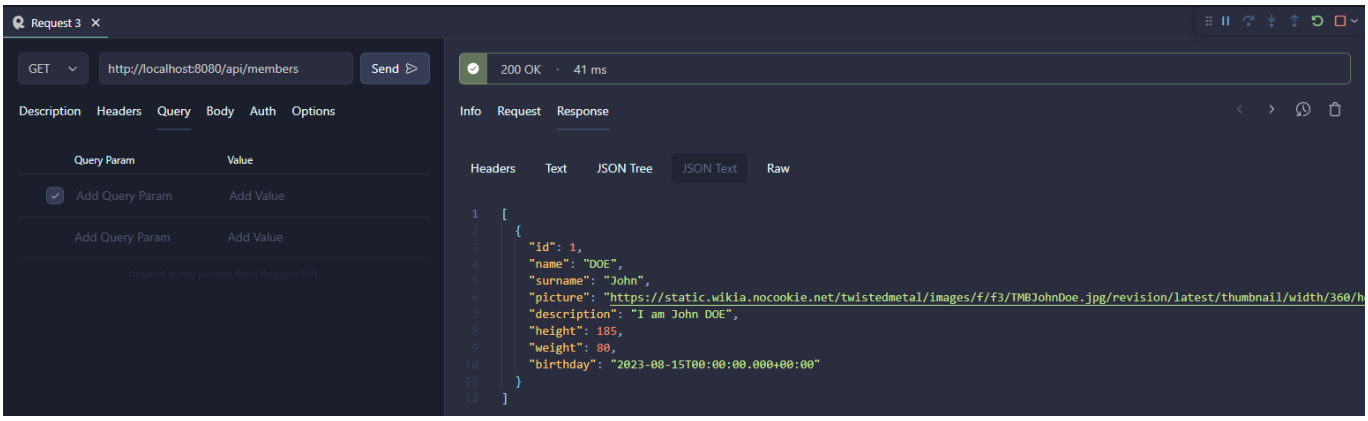
1. Create Operation : add a Member (POST)

- Request

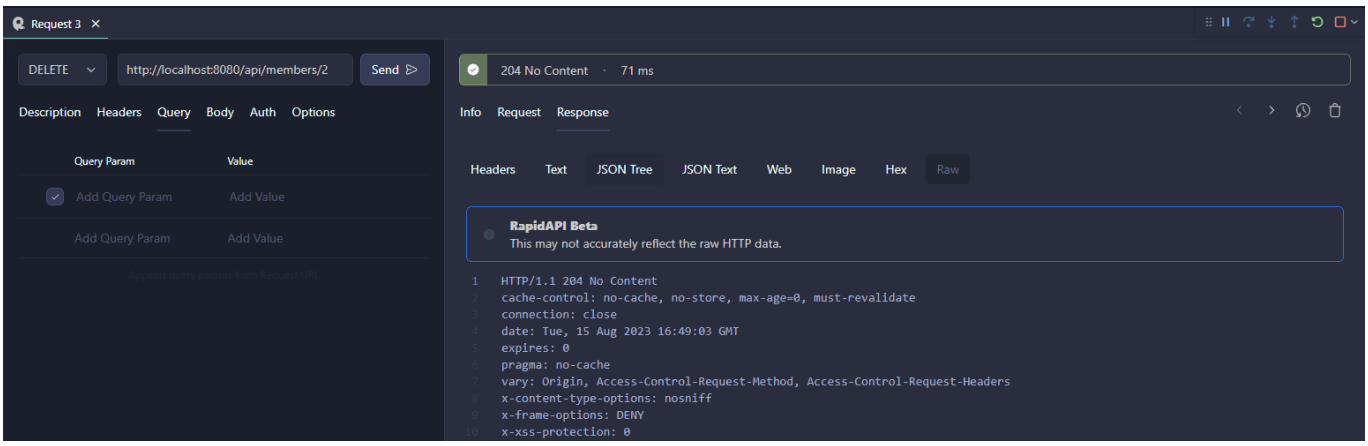


- Response

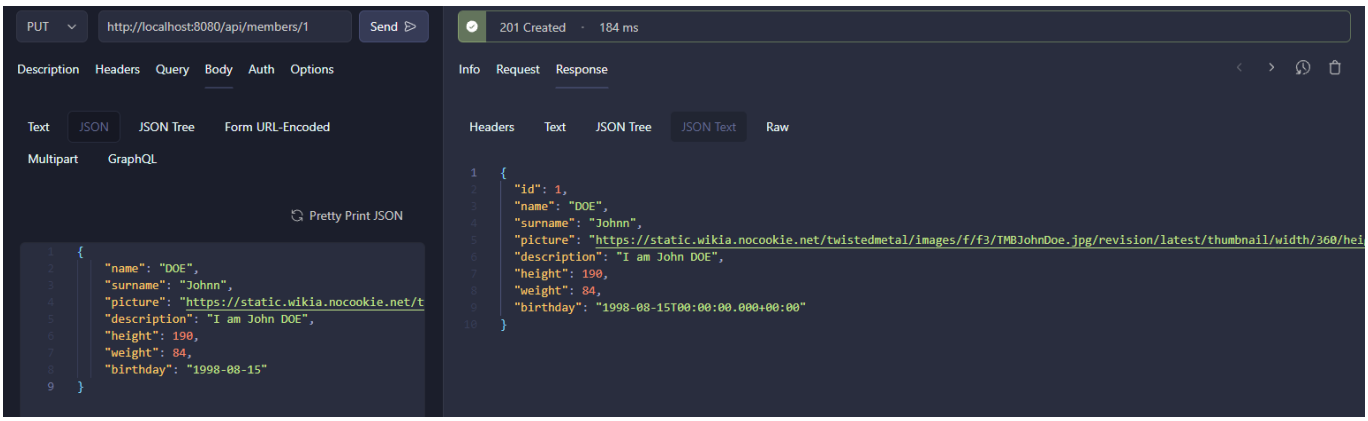
1. Read Operation : get a Member or all member ()



3. Delele Operation (HTTP DELETE) : delete a member by id

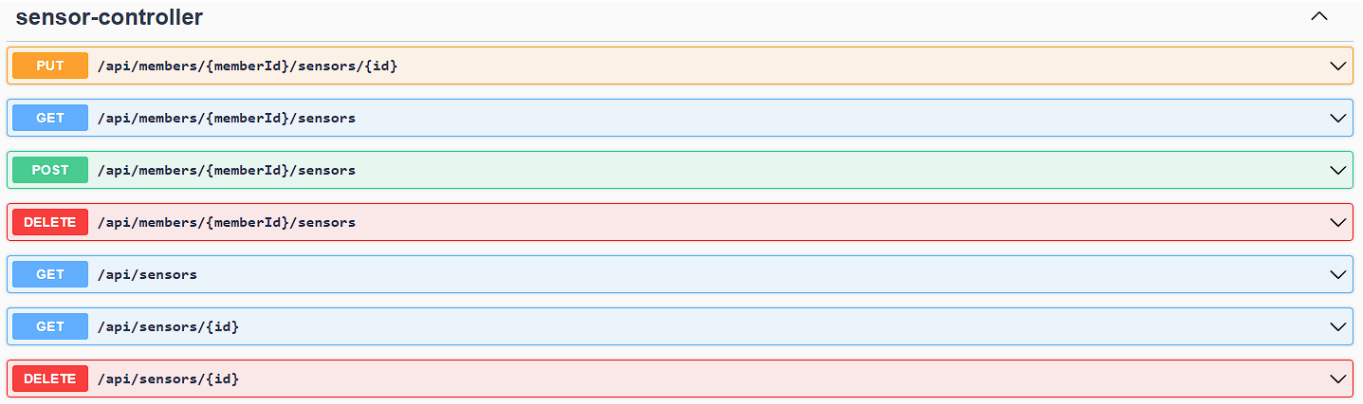


4. Update : edit a member

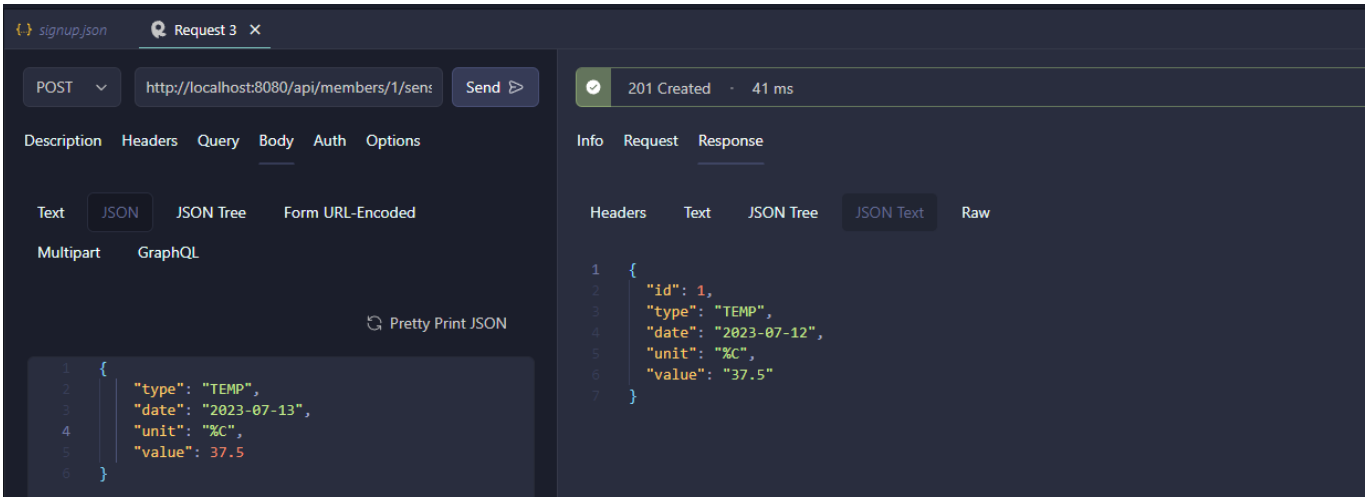


Sensor Service

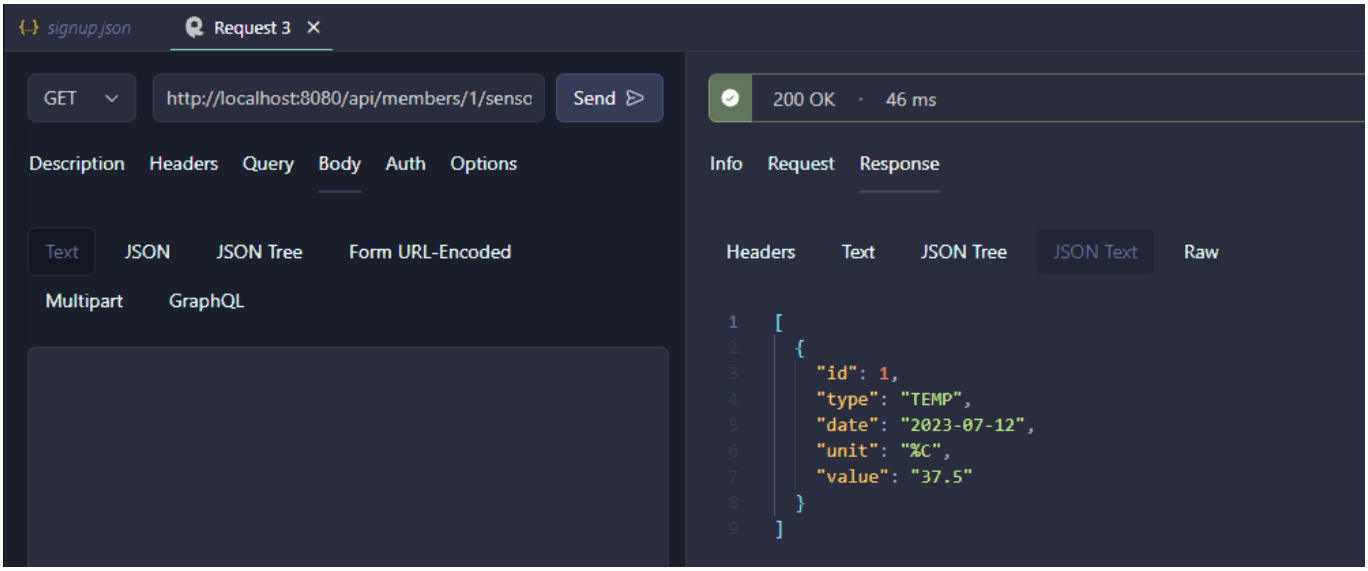
- Sensor Service `Endpoints` :



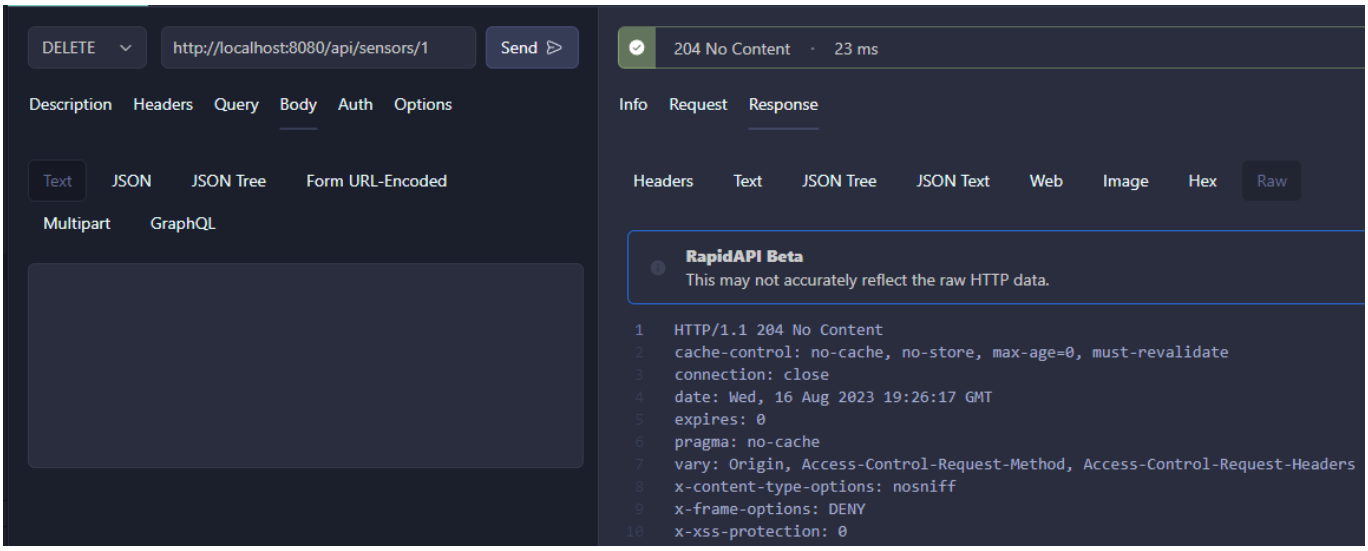
1. Create Operation : add a member's sensor data to the database



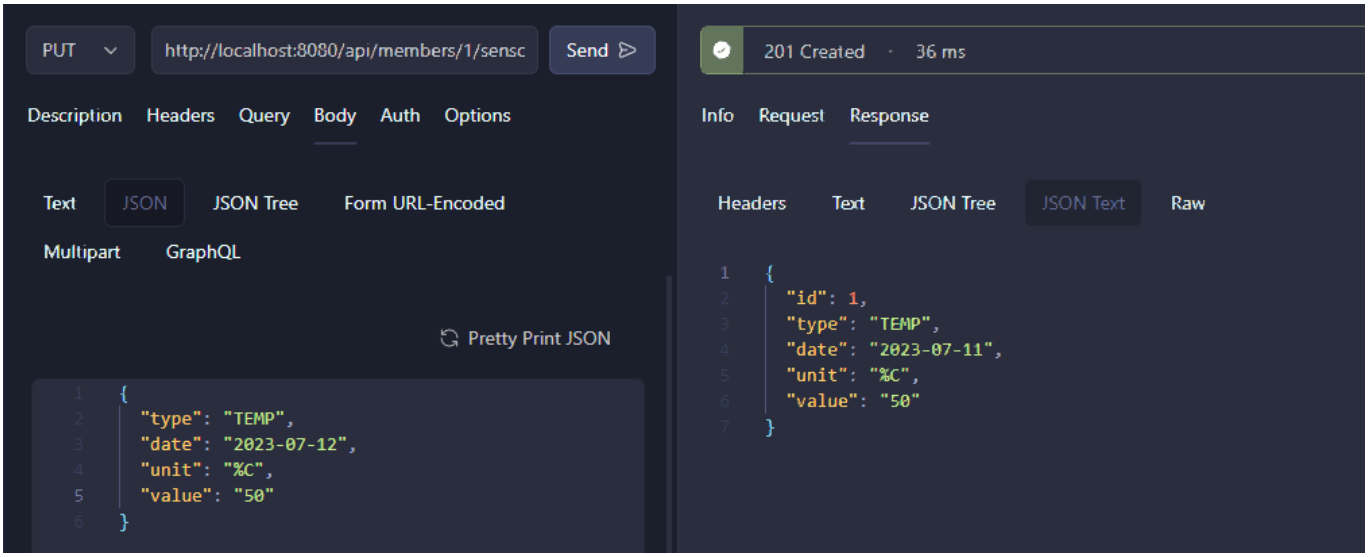
2. Read Operation : get a member's sensors data from the database



3. Delele Operation : delete a sensor data from the database

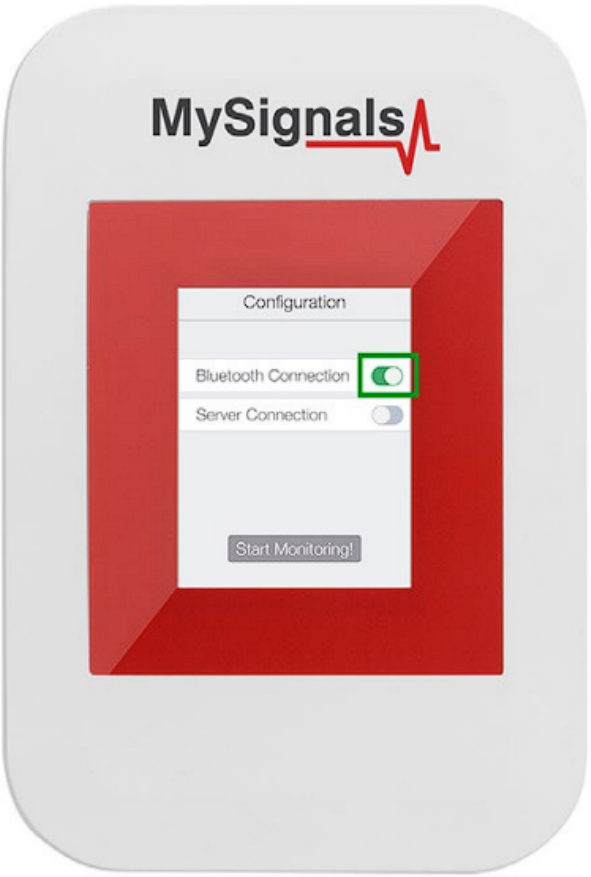


4. Update Operation : edit a member's sensor data from the database

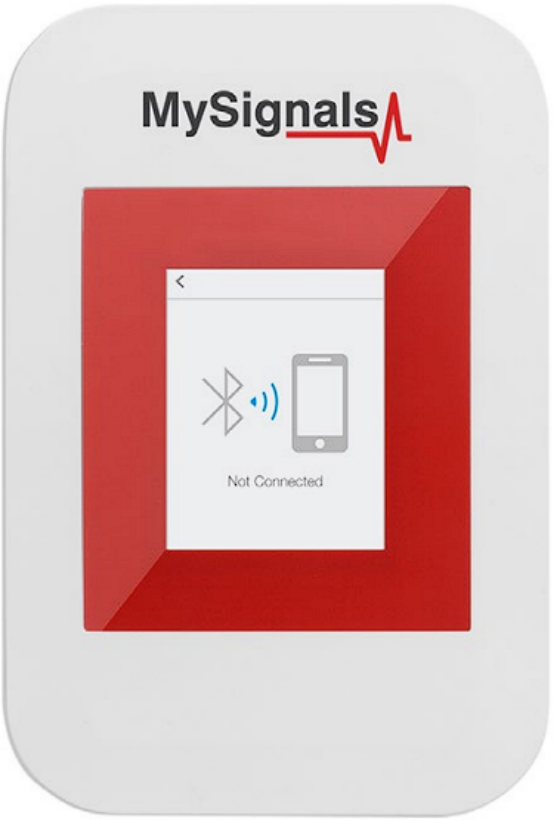


MySignals Bluetooth Device

- Activate Bluetooth



- Start Monitoring



Android App

Installation

> **IMPORTANT**: Before Install the APP, make sure that the API URL in the android source code is correct. If it's not, you should change the `BASE_URL` attribute in the `Constants` Class like this :

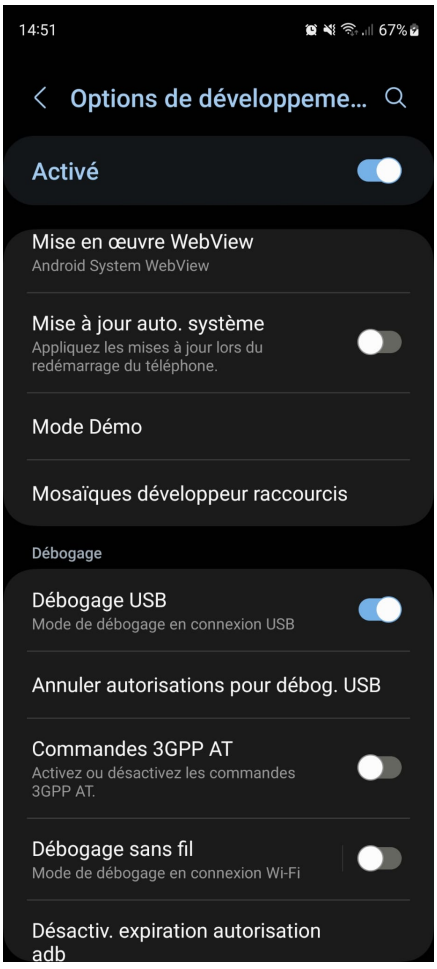
```
package com.example.mysignalsapp.utils;

public class Constants {
    public static final String BASE_URL = "https://172.17.33.64:8080/";
    public static final String LOGIN_URL = "api/auth/signin";
    public static final String REGISTRATION_URL = "api/auth/signup";
    public static final String SENSORS_URL = "api/sensors";
    public static final String MEMBERS_URL = "api/members";
}
```

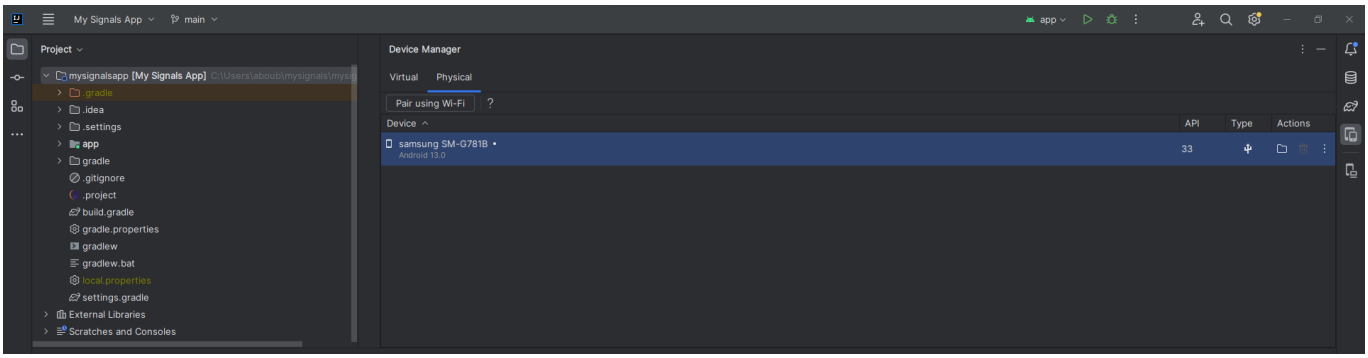
> **IMPORTANT**: to use `HTTP` instead `HTTPS` for the communication to the API, you must add this line to the `AndroidManifest.xml` file (to use only in developpment mode) :

```
android:usesCleartextTraffic="true"
```

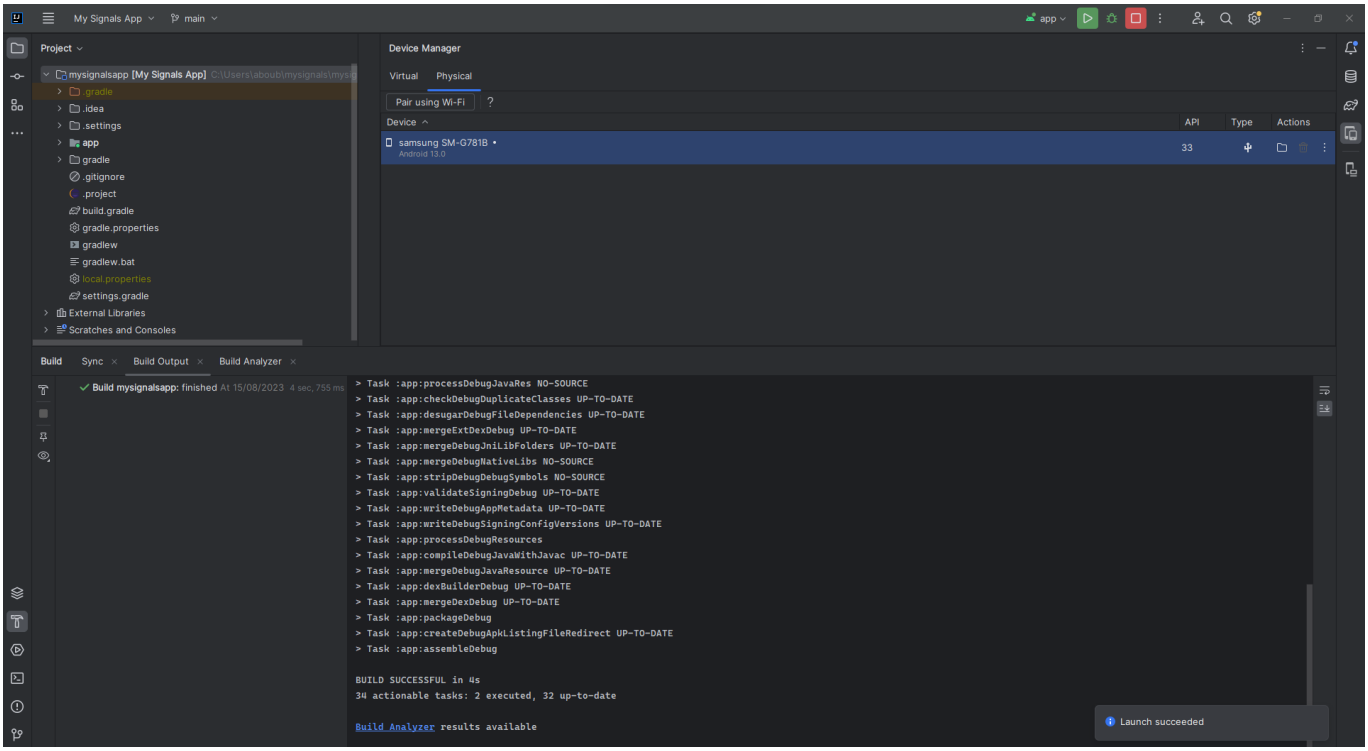
1. Android developpement mode activation



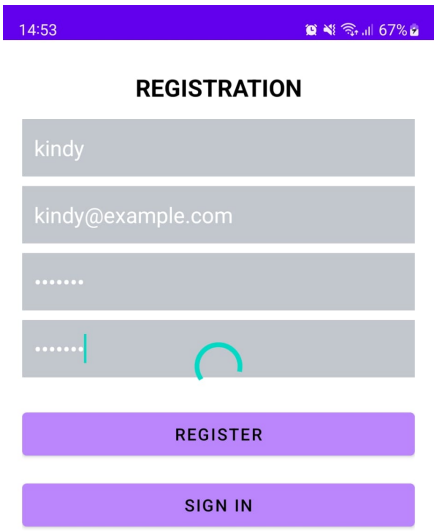
2. Android device detection in intelliJ

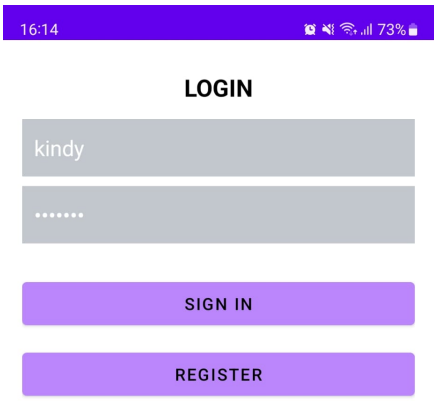


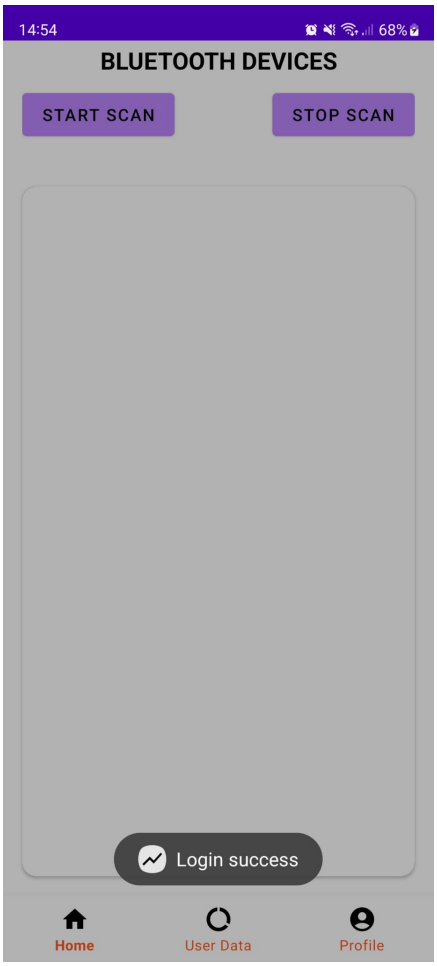
3. `MySignals APK` Installation



Authentication via android App

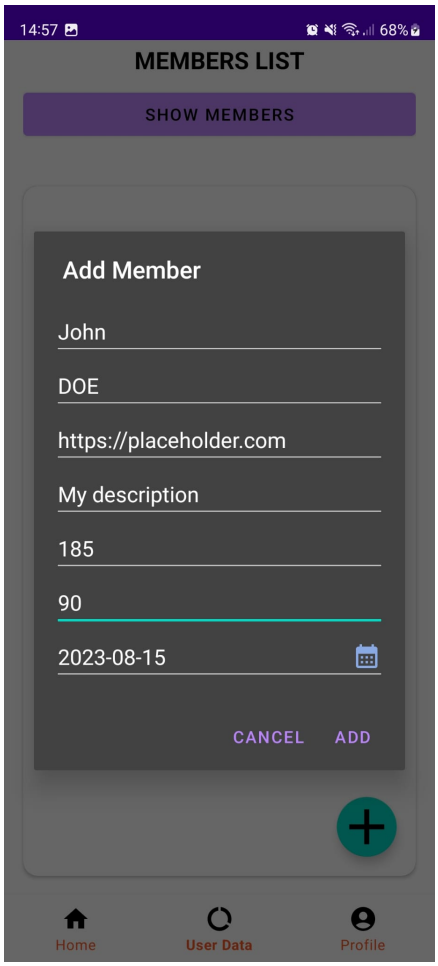




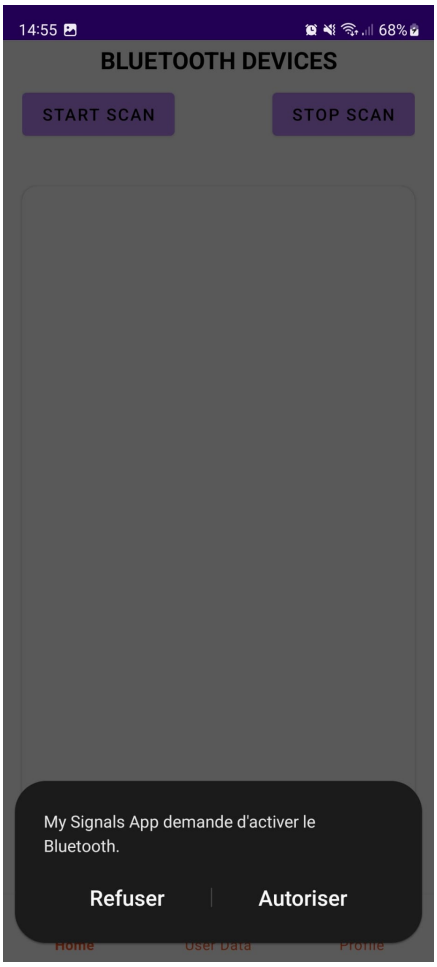


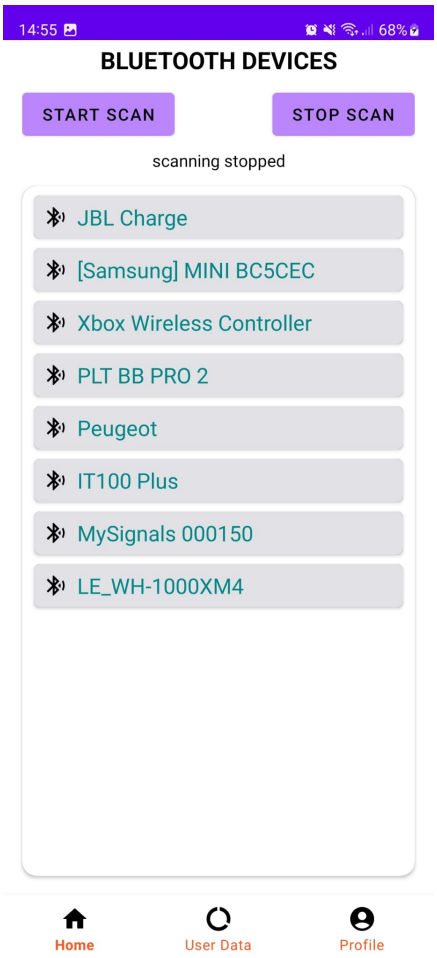
Data Visualization

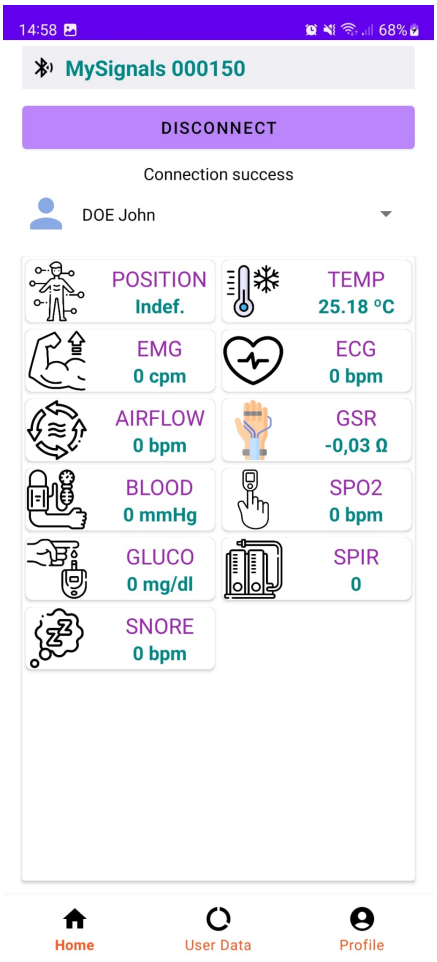
1. Add a member



2. Bluetooth Connexion to MySignal Device, Collect and Save sensors data

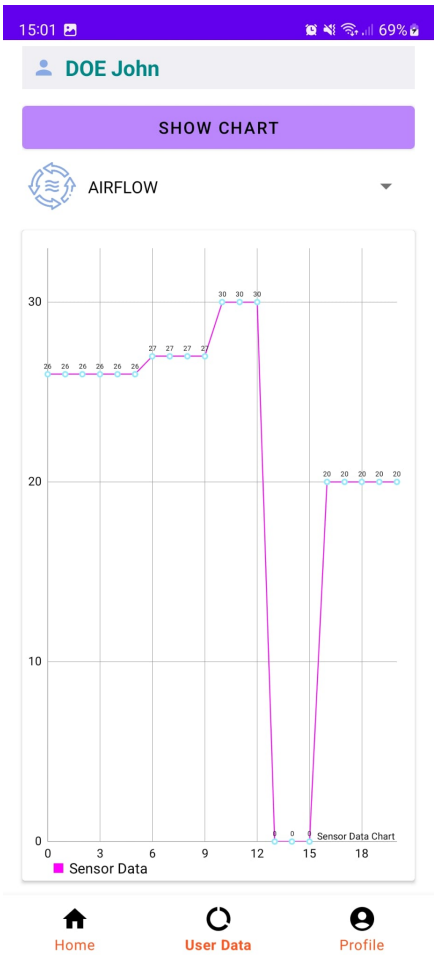






> **IMPORTANT** : The maximum sensors we can only select, in `bluetooth mode` is `11`

1. Realtime Data Visualization



Improvements

- Setup API Base URL dynamycaly for the Android APP
- Make sure the user registrattion provide a valid email (Email verification and validation)
- Actually any user can acces to all members in the API. So the next improvement can be separete each user with his own members
- Determine the maximum an the minimun of each sensor data in data visualization mode
- Add a web interface to the TIGUM website for the data visualization by using the API
- Search and Fixes the bugs in the Android App
- Found a way to show each member's picture in the Android App

Sources

- Documentation : https://www.generationrobots.com/media/mysignals_technical_guide_sw.pdf
- JWT Authentication : https://github.com/bezkoder/spring-boot-spring-security-jwt-authentication.git
- Android/IOS libelium libraries :
    - http://downloads.libelium.com/mysignals/mysignals_android/MySignalsConnectKit.jar.zip
    - http://downloads.libelium.com/mysignals/mysignals_android/MySignalsConnectKitDoc-android.zip
    - http://downloads.libelium.com/mysignals/mysignals_android/MySignalsConnectTest-android.zip

- Code source URL : https://github.com/kindy235/mysignals.git