

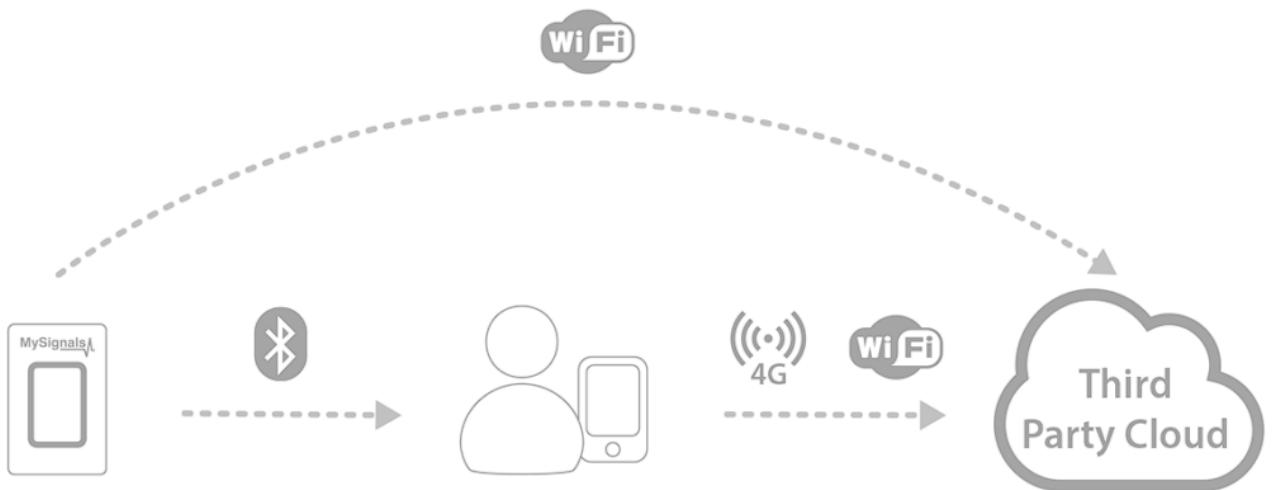
# My Signals project

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.

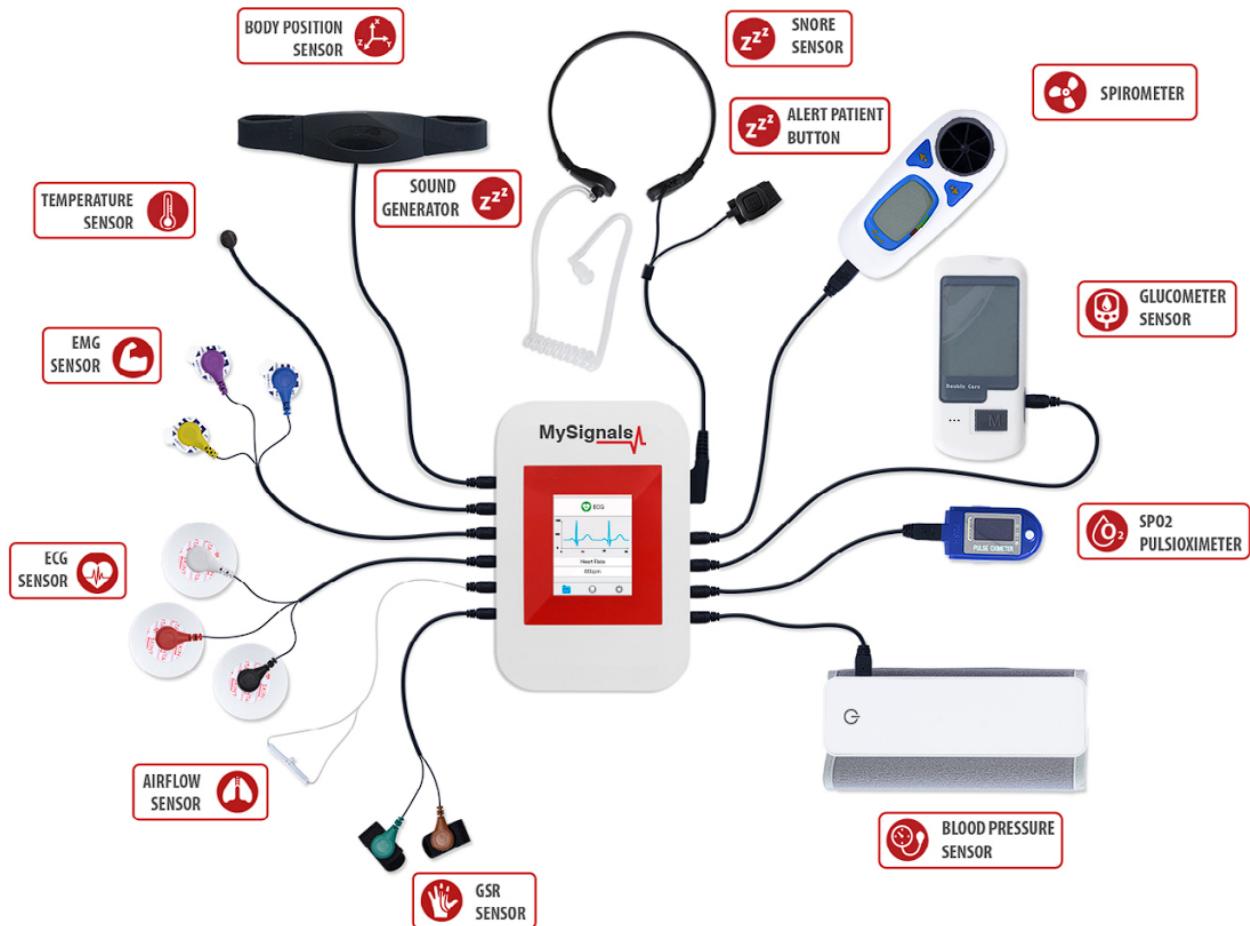


**The main objectives are :**

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

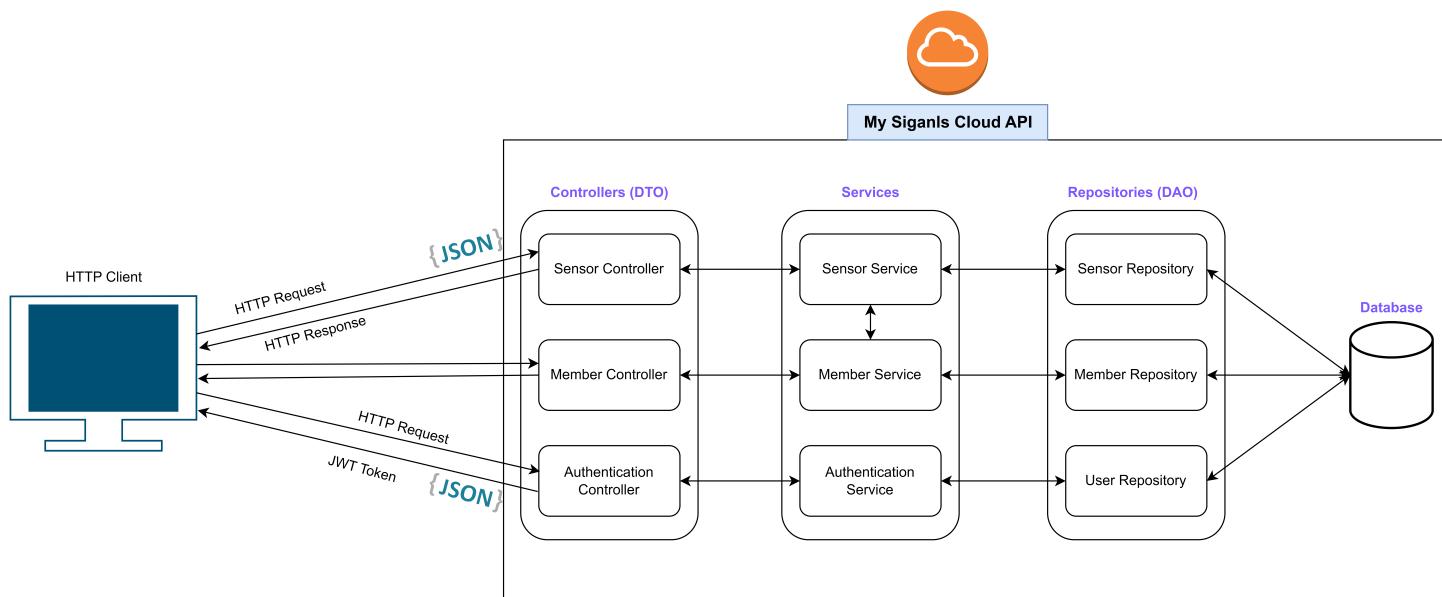
---

The API is based SpringBoot API

## Tools

- IntelliJ IDEA, Visual Studio Code...
- Java JDK & JRE (Version  $\geq 8$ )
- 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 select the DBMS in spingboot configuration file `application.properties`

### H2 Database (in developpment)

- 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)

- Database configuration for POSTGRES

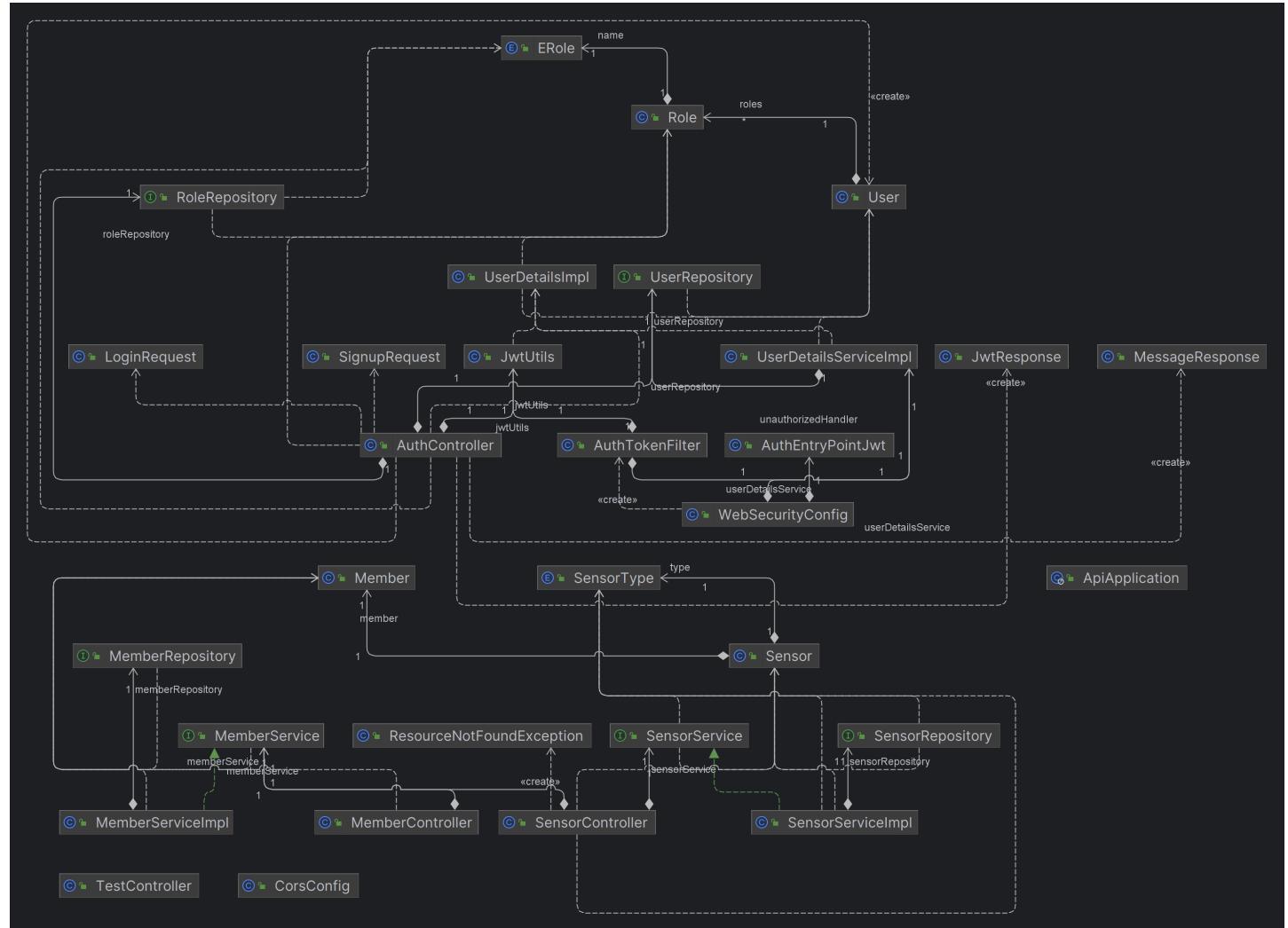
```
# Postgres Database
spring.datasource.url= jdbc:postgresql://database_service:5432/sensorsdb
spring.datasource.username= admin
spring.datasource.password= adminadmin
spring.jpa.properties.hibernate.jdbc.lob.non_contextual_creation= true
spring.jpa.properties.hibernate.dialect= org.hibernate.dialect.PostgreSQLDialect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

```
spring.jpa.show-sql=true
spring.jpa.hibernate.ddl-auto=update
```

if we dont use H2 database, we need to insert data in the roles table of our database by executing these lines :

```
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

### Docker compose file docker-compose.yml

```
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	0.0.0.0:5
pgadmin4	"/entrypoint.sh"	pgadmin	running	443/tcp,
proxy_service	"httpd-foreground"	proxy_service	running	0.0.0.0:8

## Github Pipeline (TODO)

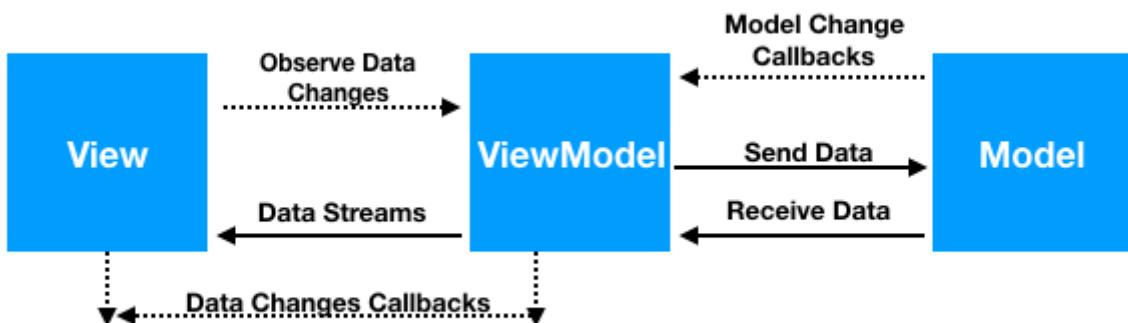
# Android Application

## Requirements

- IntelliJ IDEA, Android Studio
- Java JDK & JRE (Version  $\geq$  8)
- Android Device(android minimum SDK  $\geq$  4.3.0)

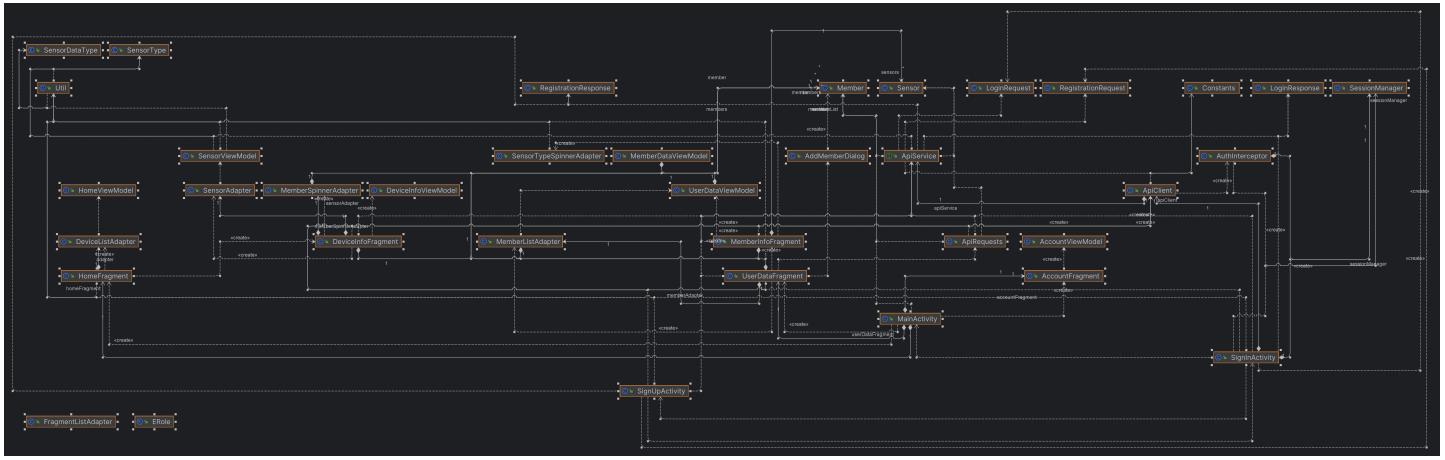
## Architecture

The design Pattern use for android developement is MVVM



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

## Classes Diagram



## Demo

### API

- Launch the API

```
> & 'C:\Program Files\Java\jdk-17\bin\java.exe' '@C:\Users\aboub\AppData\Local\Temp\cp_1rym0r39u5t9qq1r1mkfb5r09.argvfile' 'com.mysignals.api.ApiApplication'
   . . .
:: Spring Boot ::          (v3.1.1)
```

- Interface Web screen : <http://localhost:8080/swagger-ui/index.html#/>

Servers  
http://localhost:8080 - Generated server url

**sensor-controller**

**member-controller**

**auth-controller**

**test-controller**

**Schemas**

- Sensor >
- Member >
- SignupRequest >
- LoginRequest >

# Authentication Service

## auth-controller

POST /api/auth/signup

POST /api/auth/signin

### 1. Registration

- Client Request

## auth-controller

POST /api/auth/signup

### Parameters

No parameters

### Request body required

application/json

```
{  
    "username": "kindy",  
    "email": "kindy@example.com",  
    "role": [  
        "mod"  
    ],  
    "password": "kindy22"  
}
```

Execute

Clear

### Responses

#### Curl

```
curl -X 'POST' \  
  'http://localhost:8080/api/auth/signup' \  
  -H 'accept: */*' \  
  -H 'Content-Type: application/json' \  
  -d '{  
    "username": "kindy",  
    "email": "kindy@example.com",  
    "role": [  
        "mod"  
    ],  
    "password": "kindy22"  
}'
```

- API Response

### Server response

Code Details

200

Response body

```
{  
    "message": "User registered successfully!"  
}
```

Copy Download

Response headers

```
cache-control: no-cache,no-store,max-age=0,must-revalidate  
connection: keep-alive  
content-type: application/json  
date: Tue,15 Aug 2023 15:54:24 GMT  
expires: 0  
keep-alive: timeout=60  
pragma: no-cache  
transfer-encoding: chunked  
vary: Origin,Access-Control-Request-Method,Access-Control-Request-Headers  
x-content-type-options: nosniff  
x-frame-options: DENY  
x-xss-protection: 0
```

### 1. Login

- Client Request

**POST** /api/auth/signin

**Parameters**

No parameters

**Request body** required

application/json

```
{
  "username": "kindy",
  "password": "kindy22"
}
```

**Execute** **Clear**

**Responses**

**Curl**

```
curl -X 'POST' \
'http://localhost:8080/api/auth/signin' \
-H 'accept: */*' \
-H 'Content-Type: application/json' \
-d '{
  "username": "kindy",
  "password": "kindy22"
}'
```

- API Response

**Server response**

Code	Details
200	<b>Response body</b> <pre>{   "id": 1,   "username": "kindy",   "email": "kindy@example.com",   "roles": [     "ROLE_MODERATOR"   ],   "tokenType": "Bearer",   "accessToken": "eyJhbGciOiJIUzI1NiJ9.eyJzdWIiOiJraW5keSIsImhdCI6MTY5MjExNTEwMiwiZXhwIjoxNjkyMjAxNTAyFQ.bMh1Uu9GqUah96S6yyu91NtkTbNHbwPR8zVrpYhIkro" }</pre> <div style="text-align: right;"> <a href="#">Copy</a> <a href="#">Download</a> </div> <b>Response headers</b> <pre>cache-control: no-cache,no-store,max-age=0,must-revalidate connection: keep-alive content-type: application/json date: Tue,15 Aug 2023 15:58:22 GMT expires: 0 keep-alive: timeout=60 pragma: no-cache transfer-encoding: chunked vary: Origin,Access-Control-Request-Method,Access-Control-Request-Headers x-content-type-options: nosniff x-frame-options: DENY x-xss-protection: 0</pre>

## How to use the user AccessToken to make HTTP Request to the API

Request 3 X

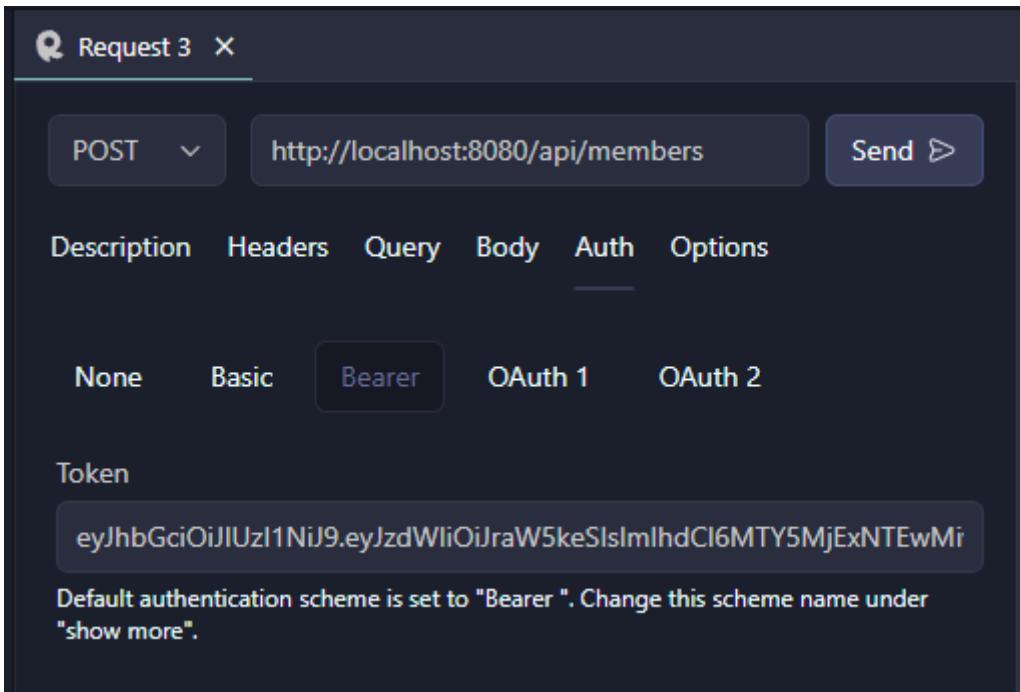
POST ▾ http://localhost:8080/api/members Send ➡

Description Headers Query Body Auth Options

None Basic Bearer OAuth 1 OAuth 2

Token eyJhbGciOiJIUzI1NiJ9.eyJzdWliOiJraW5keSIsImIhdCI6MTY5MjExNTAwMi

Default authentication scheme is set to "Bearer ". Change this scheme name under "show more".



## Member Service

member-controller ^

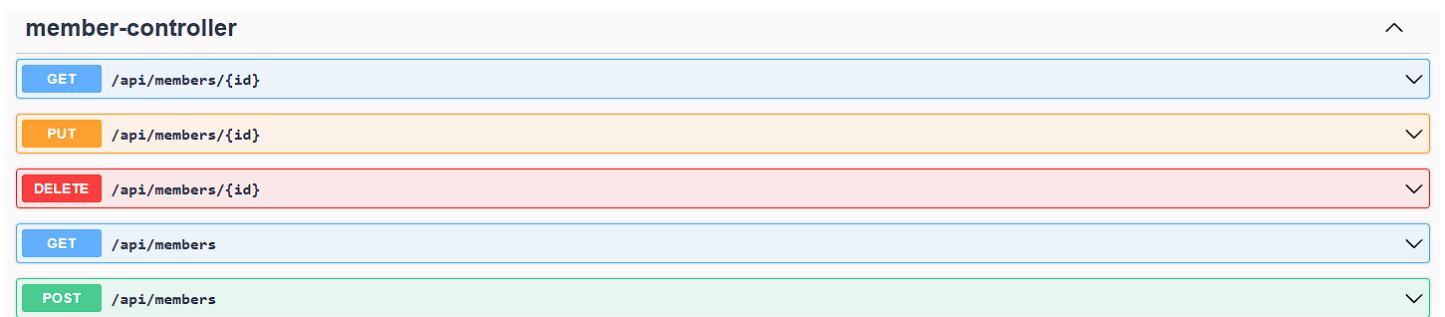
GET /api/members/{id} ▾

PUT /api/members/{id} ▾

DELETE /api/members/{id} ▾

GET /api/members ▾

POST /api/members ▾



1. Create Operation : add a Member (POST)

- Request

### Request 3

POST <http://localhost:8080/api/members> Send ▶

Description Headers Query Body Auth Options

Text JSON JSON Tree Form URL-Encoded

Multipart GraphQL

Pretty Print JSON

```
1 {  
2   "name": "DOE",  
3   "surname": "John",  
4   "picture": "https://static.wikia.nocookie.net/twi  
5   "description": "I am John DOE",  
6   "height": 185,  
7   "weight": 80,  
8   "birthday": "2023-08-15"  
9 }  
10 }
```

- Response

201 Created · 143 ms

Info Request Response

Headers Text JSON Tree JSON Text Raw

```
1 {  
2   "id": 1,  
3   "name": "DOE",  
4   "surname": "John",  
5   "picture": "https://static.wikia.nocookie.net/twistedmetal/images/f/f3/TMBJohnDoe.jpg/revision/latest/thumb/width360/he  
6   "description": "I am John DOE",  
7   "height": 185,  
8   "weight": 80,  
9   "birthday": "2023-08-15T00:00:00.000+00:00"  
10 }  
11 }
```

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

Request 3

GET http://localhost:8080/api/members Send ▶

Description Headers Query Body Auth Options

Query Param Value

Add Query Param Add Value

Add Query Param Add Value

Append query params from Request URL

Info Request Response

Headers Text JSON Tree JSON Text Raw

```

1 [
2   {
3     "id": 1,
4     "name": "DOE",
5     "surname": "John",
6     "picture": "https://static.wikia.nocookie.net/twistedmetal/images/f/f3/TMBJohnDoe.jpg/revision/latest/thumb/width/360/he",
7     "description": "I am John DOE",
8     "height": 185,
9     "weight": 80,
10    "birthday": "2023-08-15T00:00:00.000+00:00"
11  }
12 ]

```

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

Request 3

DELETE http://localhost:8080/api/members/2 Send ▶

Description Headers Query Body Auth Options

Query Param Value

Add Query Param Add Value

Add Query Param Add Value

Append query params from Request URL

Info Request Response

Headers Text JSON Tree JSON Text Web Image Hex Raw

**RapidAPI Beta**  
This may not accurately reflect the raw HTTP data.

```

1 HTTP/1.1 204 No Content
2 cache-control: no-cache, no-store, max-age=0, must-revalidate
3 connection: close
4 date: Tue, 15 Aug 2023 16:49:03 GMT
5 expires: 0
6 pragma: no-cache
7 vary: Origin, Access-Control-Request-Method, Access-Control-Request-Headers
8 x-content-type-options: nosniff
9 x-frame-options: DENY
10 x-xss-protection: 0

```

### 4. Update

PUT http://localhost:8080/api/members/1 Send ▶

Description Headers Query Body Auth Options

Text JSON JSON Tree Form URL-Encoded

Multipart GraphQL

Pretty Print JSON

```

1 {
2   "name": "DOE",
3   "surname": "Johnn",
4   "picture": "https://static.wikia.nocookie.net/twistedmetal/images/f/f3/TMBJohnDoe.jpg/revision/latest/thumb/width/360/he",
5   "description": "I am John DOE",
6   "height": 190,
7   "weight": 84,
8   "birthday": "1998-08-15"
9 }

```

Info Request Response

Headers Text JSON Tree JSON Text Raw

```

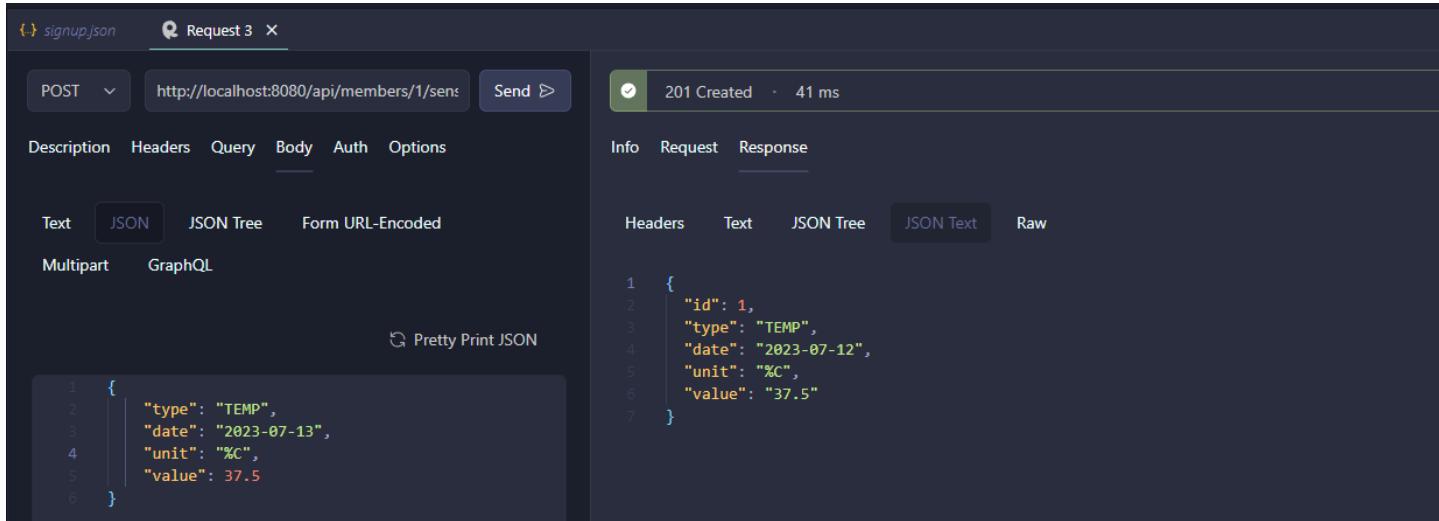
1 {
2   "id": 1,
3   "name": "DOE",
4   "surname": "Johnn",
5   "picture": "https://static.wikia.nocookie.net/twistedmetal/images/f/f3/TMBJohnDoe.jpg/revision/latest/thumb/width/360/he",
6   "description": "I am John DOE",
7   "height": 190,
8   "weight": 84,
9   "birthday": "1998-08-15T00:00:00.000+00:00"
10 }

```

## Sensor Service

sensor-controller	
PUT	/api/members/{memberId}/sensors/{id}
GET	/api/members/{memberId}/sensors
POST	/api/members/{memberId}/sensors
DELETE	/api/members/{memberId}/sensors
GET	/api/sensors
GET	/api/sensors/{id}
DELETE	/api/sensors/{id}

## 1. Create



The screenshot shows a POST request to `http://localhost:8080/api/members/1/sensors`. The JSON body is:

```

1 {
2   "type": "TEMP",
3   "date": "2023-07-13",
4   "unit": "%C",
5   "value": 37.5
6 }

```

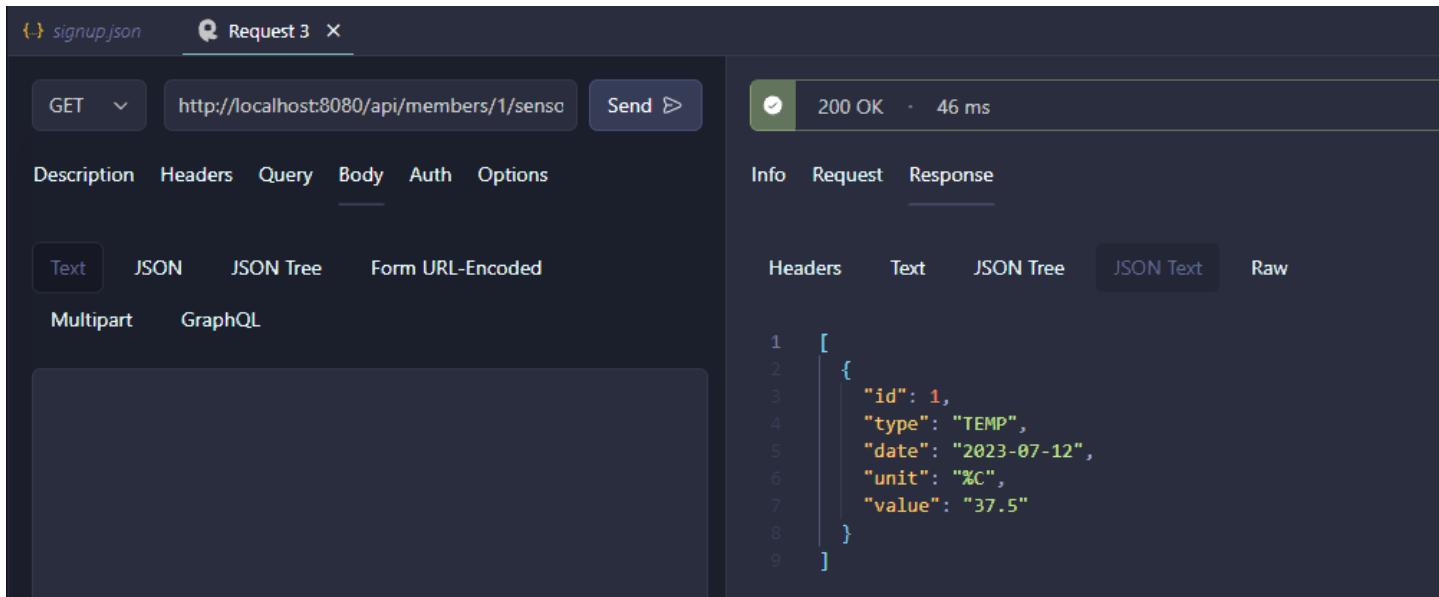
The response status is 201 Created with a response time of 41 ms. The response body is:

```

1 {
2   "id": 1,
3   "type": "TEMP",
4   "date": "2023-07-12",
5   "unit": "%C",
6   "value": "37.5"
7 }

```

## 2. Read



The screenshot shows a GET request to `http://localhost:8080/api/members/1/sensors`. The response status is 200 OK with a response time of 46 ms. The response body is:

```

1 [
2   {
3     "id": 1,
4     "type": "TEMP",
5     "date": "2023-07-12",
6     "unit": "%C",
7     "value": "37.5"
8   }
9 ]

```

## 3. Delete

The screenshot shows the RapidAPI Beta interface. On the left, there's a toolbar with 'DELETE' selected, followed by the URL 'http://localhost:8080/api/sensors/1'. Below the URL are tabs for 'Description', 'Headers', 'Query', 'Body', 'Auth', and 'Options'. Under 'Body', there are tabs for 'Text', 'JSON', 'JSON Tree', 'Form URL-Encoded', 'Multipart', and 'GraphQL'. The main area is a large, dark gray box with rounded corners. On the right, the response details are shown: a green checkmark icon, '204 No Content', and '23 ms'. Below this are tabs for 'Info', 'Request', and 'Response'. Under 'Response', there are tabs for 'Headers', 'Text', 'JSON Tree', 'JSON Text', 'Web', 'Image', 'Hex', and 'Raw'. A note says 'RapidAPI Beta' and 'This may not accurately reflect the raw HTTP data.' Below this note is a code block showing the raw HTTP response headers:

```
1 HTTP/1.1 204 No Content
2 cache-control: no-cache, no-store, max-age=0, must-revalidate
3 connection: close
4 date: Wed, 16 Aug 2023 19:26:17 GMT
5 expires: 0
6 pragma: no-cache
7 vary: Origin, Access-Control-Request-Method, Access-Control-Request-Headers
8 x-content-type-options: nosniff
9 x-frame-options: DENY
10 x-xss-protection: 0
```

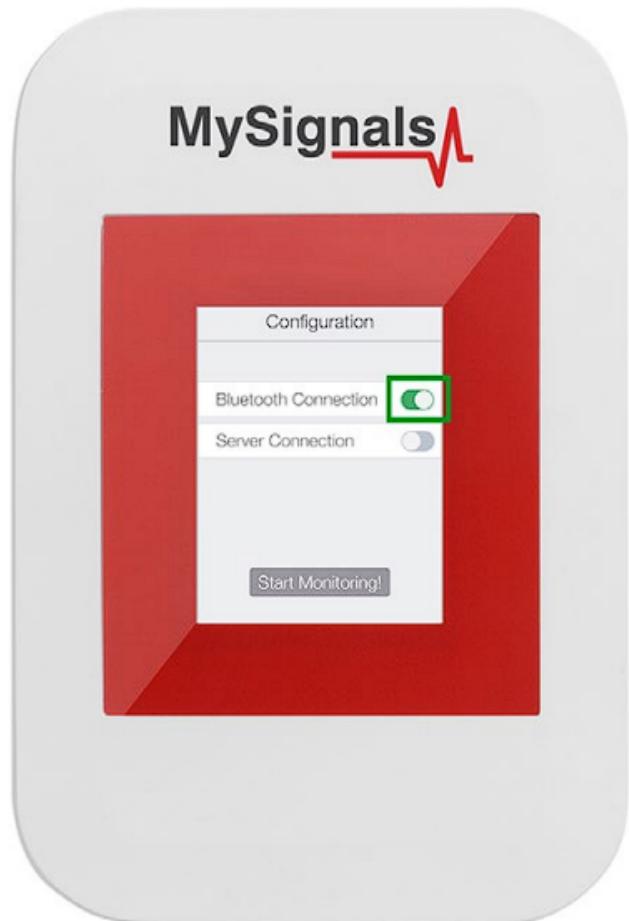
## 4. Update

The screenshot shows the RapidAPI Beta interface. On the left, there's a toolbar with 'PUT' selected, followed by the URL 'http://localhost:8080/api/members/1/sensc'. Below the URL are tabs for 'Description', 'Headers', 'Query', 'Body', 'Auth', and 'Options'. Under 'Body', there are tabs for 'Text', 'JSON', 'JSON Tree', 'Form URL-Encoded', 'Multipart', and 'GraphQL'. The main area is a large, dark gray box with rounded corners. On the right, the response details are shown: a green checkmark icon, '201 Created', and '36 ms'. Below this are tabs for 'Info', 'Request', and 'Response'. Under 'Response', there are tabs for 'Headers', 'Text', 'JSON Tree', 'JSON Text', and 'Raw'. A note says 'Pretty Print JSON'. Below this is a code block showing the JSON response body:

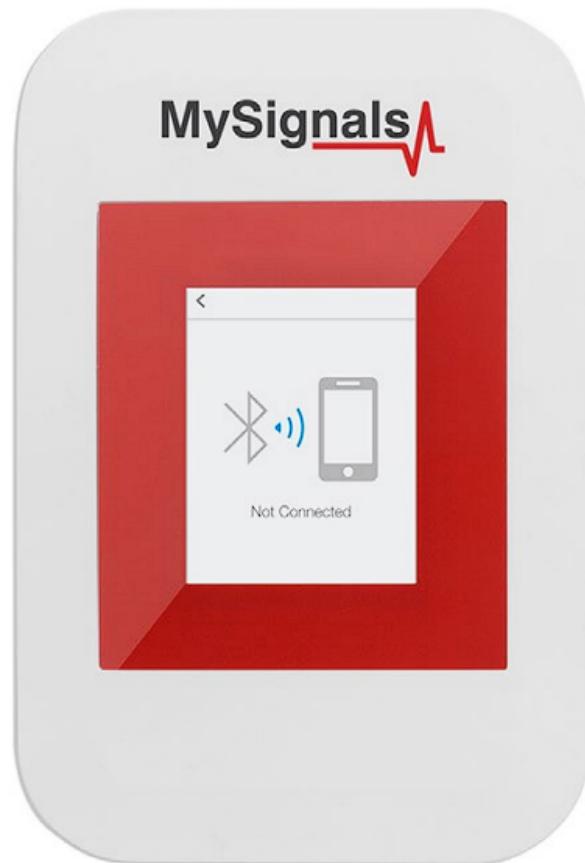
```
1 {
2   "id": 1,
3   "type": "TEMP",
4   "date": "2023-07-11",
5   "unit": "%C",
6   "value": "50"
7 }
```

## MySignals Bluetooth Device

- Activate Bluetooth



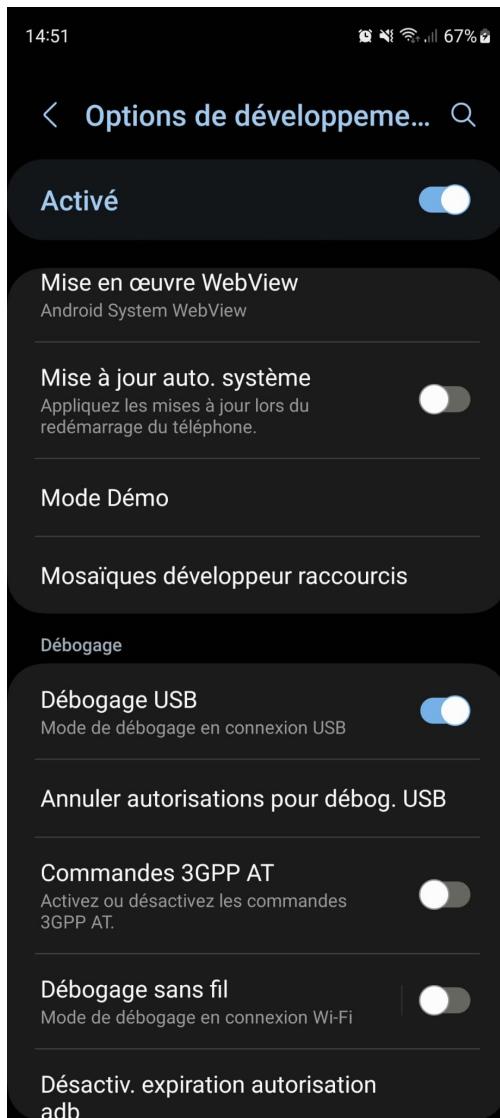
- Start Monitoring



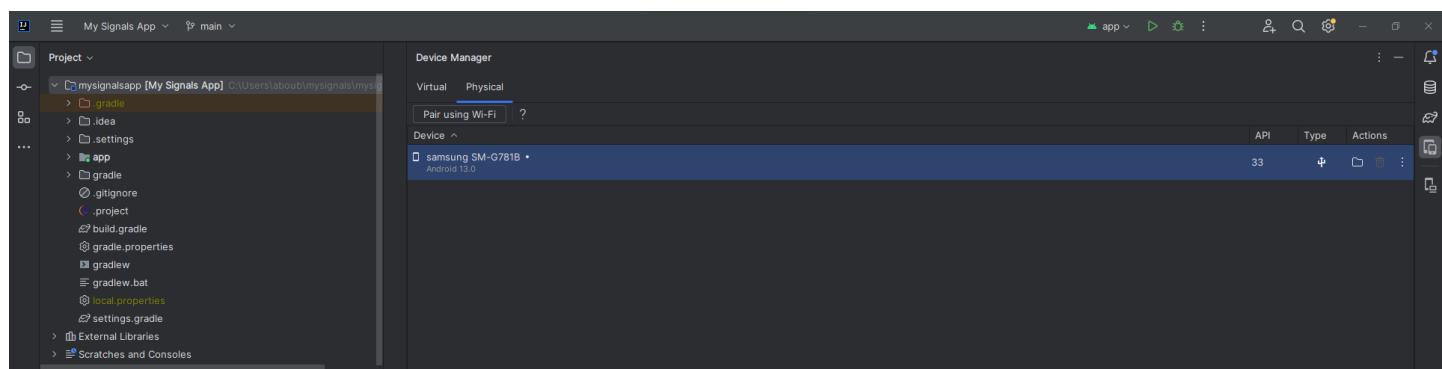
# Android App

## Installation

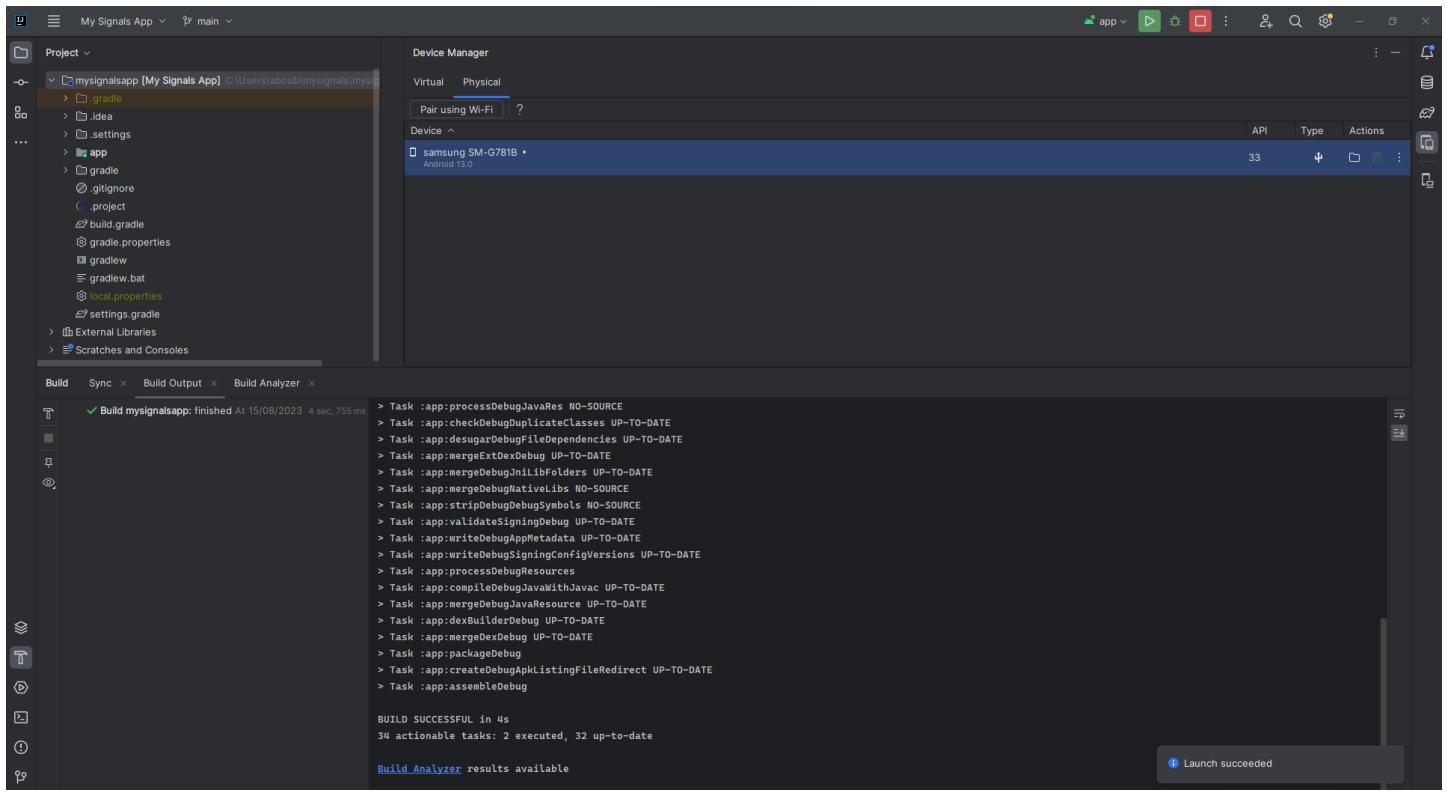
### 1. Android developpement mode



### 1. Android device detection in intelliJ



### 3. My Signals APK Installation



## Authentication via android App

14:53

67%

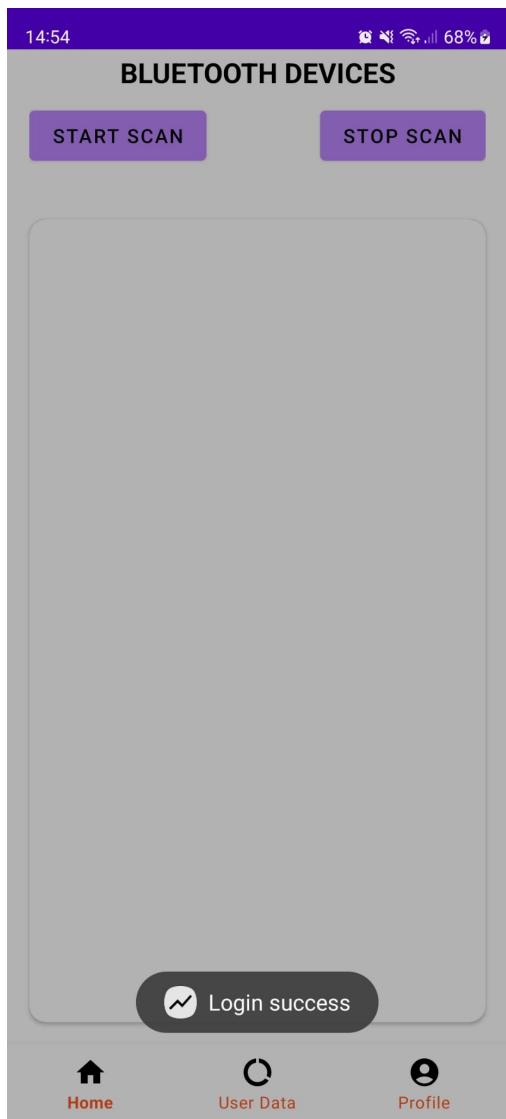
16:14

73%

## REGISTRATION

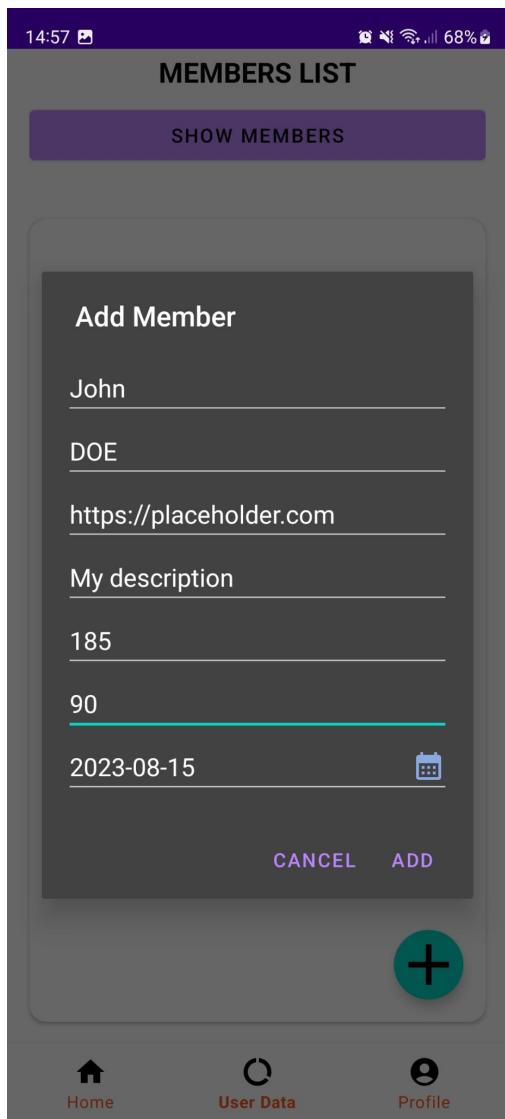


## LOGIN

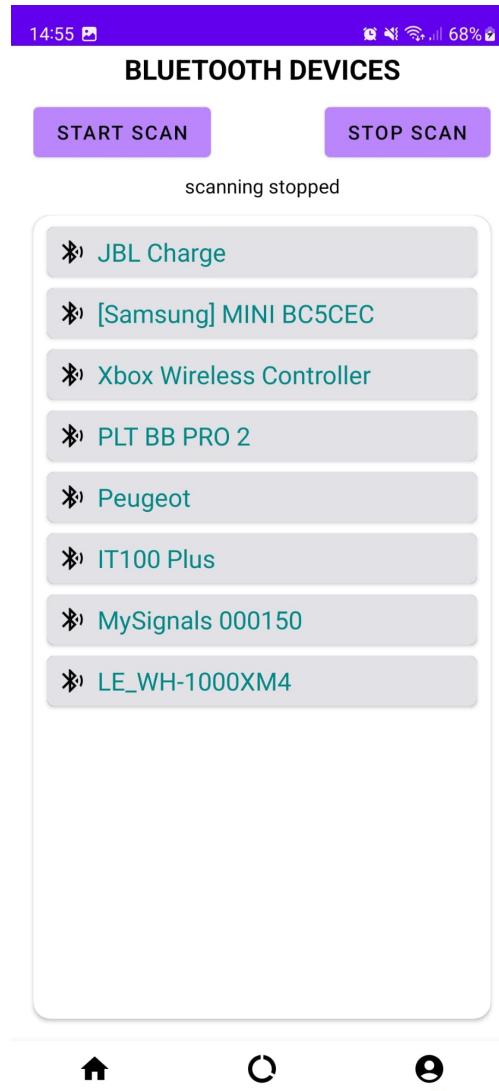


## Data Visualization

1. Add a member



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



14:58 68%

\* MySignals 000150

DISCONNECT

Connection success

 DOE John

	<b>POSITION</b> Indef.		<b>TEMP</b> 25.18 °C
	<b>EMG</b> 0 cpm		<b>ECG</b> 0 bpm
	<b>AIRFLOW</b> 0 bpm		<b>GSR</b> -0,03 Ω
	<b>BLOOD</b> 0 mmHg		<b>SPO2</b> 0 bpm
	<b>GLUCO</b> 0 mg/dl		<b>SPIR</b> 0
	<b>SNORE</b> 0 bpm		

 Home

 User Data

 Profile

### 3. Realtime Data Visualization



Home

User Data

Profile

## Improvements

### Sources

- Documentation : [https://www.generationrobots.com/media/mysignals\\_technical\\_guide\\_sw.pdf](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/MySignalsConnectKit.jar.zip)
  - [http://downloads.libelium.com/mysignals/mysignals\\_android/MySignalsConnectKitDoc-android.zip](http://downloads.libelium.com/mysignals/mysignals_android/MySignalsConnectKitDoc-android.zip)
  - [http://downloads.libelium.com/mysignals/mysignals\\_android/MySignalsConnectTest-android.zip](http://downloads.libelium.com/mysignals/mysignals_android/MySignalsConnectTest-android.zip)