

SMART GREEN CAMPUS

<정보통신공학과>

20181643 서동권

20181745 윤진원

20181676 이동엽

20181680 이진욱

20181692 최민석

20201849 임세나



Index.

01. MQTT Broker

02. Back-End

03. Front-End

04. 결산 및 계획

MQTT Broker

01



이전 발표 내용

NestJS를 이용하여 DB로 데이터 전송

```
const axios = require('axios');

axios
  .post('http://192.168.10.171:3000/sensors', { sensor })
  .then((res: AxiosResponse<any>) => {
    console.log(res);
  })
  .catch((error) => {
    console.log(error);
  });
}
```

DB파트에 데이터를 전송을 위해 axios의 post 기능을 활용했습니다.

	id	sensor_name	location	value	user_id
12	15	TEMPERATURE	N5	25	10
13	16	TEMPERATURE	N5	22	10
14	17	TEMPERATURE	N5	22	10
15	18	TEMPERATURE	N5	29	10
16	19	TEMPERATURE	N5	29	10
17	20	TEMPERATURE	N5	27	10
18	21	TEMPERATURE	N5	18	10
19	22	TEMPERATURE	N5	12	10
20	23	TEMPERATURE	N5	12	10
21	24	TEMPERATURE	N5	12	10

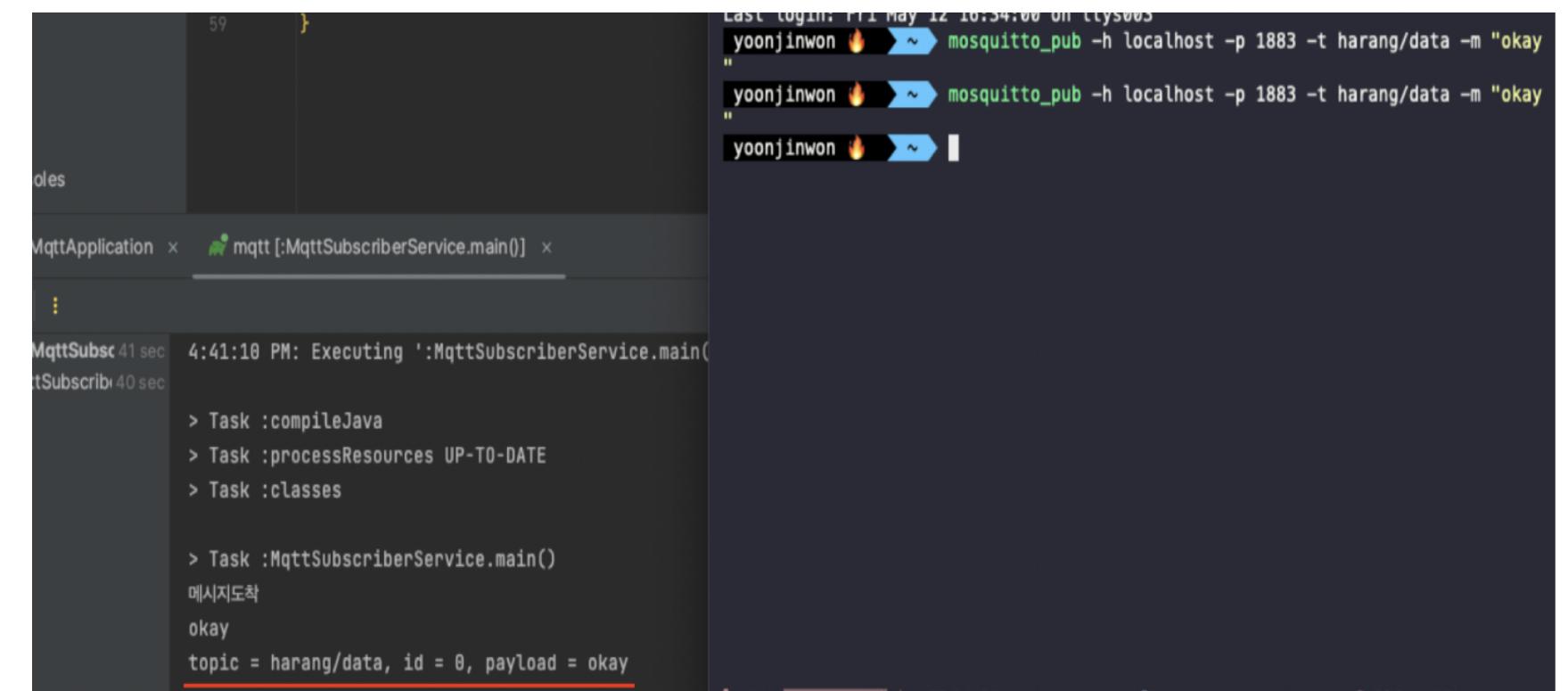
실제로 DB에 저장된 데이터를 확인할 수 있습니다.

진행 상황

Spring을 이용하여 MQTT 구현

```
public MqttSubscriberService init(final String server, final String clientId) throws MqttException {
    mqttOptions = new MqttConnectOptions();
    mqttOptions.setCleanSession(true);
    mqttOptions.setKeepAliveInterval(30);
    mqttClient = new MqttClient(server, clientId);
    mqttClient.setCallback(this);
    mqttClient.connect(mqttOptions);

    return this;
}
// 구독 신청
1 usage new*
public boolean subscribe(final String topic) throws MqttException {
    if (topic != null) {
        mqttClient.subscribe(topic, qos: 0);
    }
    return true;
}
// 메시지가 도착하면 호출
new*
@Override
public void messageArrived(final String topic, final MqttMessage message) throws Exception {
    System.out.println("메시지도착");
    System.out.println(message);
    System.out.println("topic = " + topic + ", id = " + message.getId() + ", payload = " + new String(message.getPayload()));
    webClientService.post(message);
}
```



수신 받을 데이터의 Topic을 구독합니다.

임의로 데이터를 전송하였고 구독한 Topic의 데이터를 수신 받습니다.

진행 상황

DB(Back-End파트)로 데이터 전송

```
    public static void post(final MqttMessage message) {
        final WebClientService webClientService = new WebClientService();
        webClientService.requestToApiServer(message);
    }

    1 usage  ↗ Jinwon-Dev
    private Map<String, Object> extractdata(final MqttMessage message) {
        final Map<String, Object> bodyMap = new HashMap<>();

        final String str = new String(message.getPayload());
        final StringTokenizer st = new StringTokenizer(str, " ,");
        st.nextToken();
        final Long id = Long.valueOf(st.nextToken());
        st.nextToken();
        final Double number = Double.valueOf(st.nextToken());
        st.nextToken();
        final String measurement = st.nextToken();

        bodyMap.put("memberId", id);
        bodyMap.put("value", number);
        bodyMap.put("measurement", measurement);

        return bodyMap;
    }
```

데이터를 원하는 형식으로 변환하여 저장합니다.

```
3 usages  ↗ Jinwon-Dev
@Service
public class WebClientService {

    2 usages
    MqttConfig mqttConfig = new MqttConfig();

    1 usage  ↗ Jinwon-Dev
    public void requestToApiServer(final MqttMessage message) {
        Map<String, Object> bodyMap = extractdata(message);

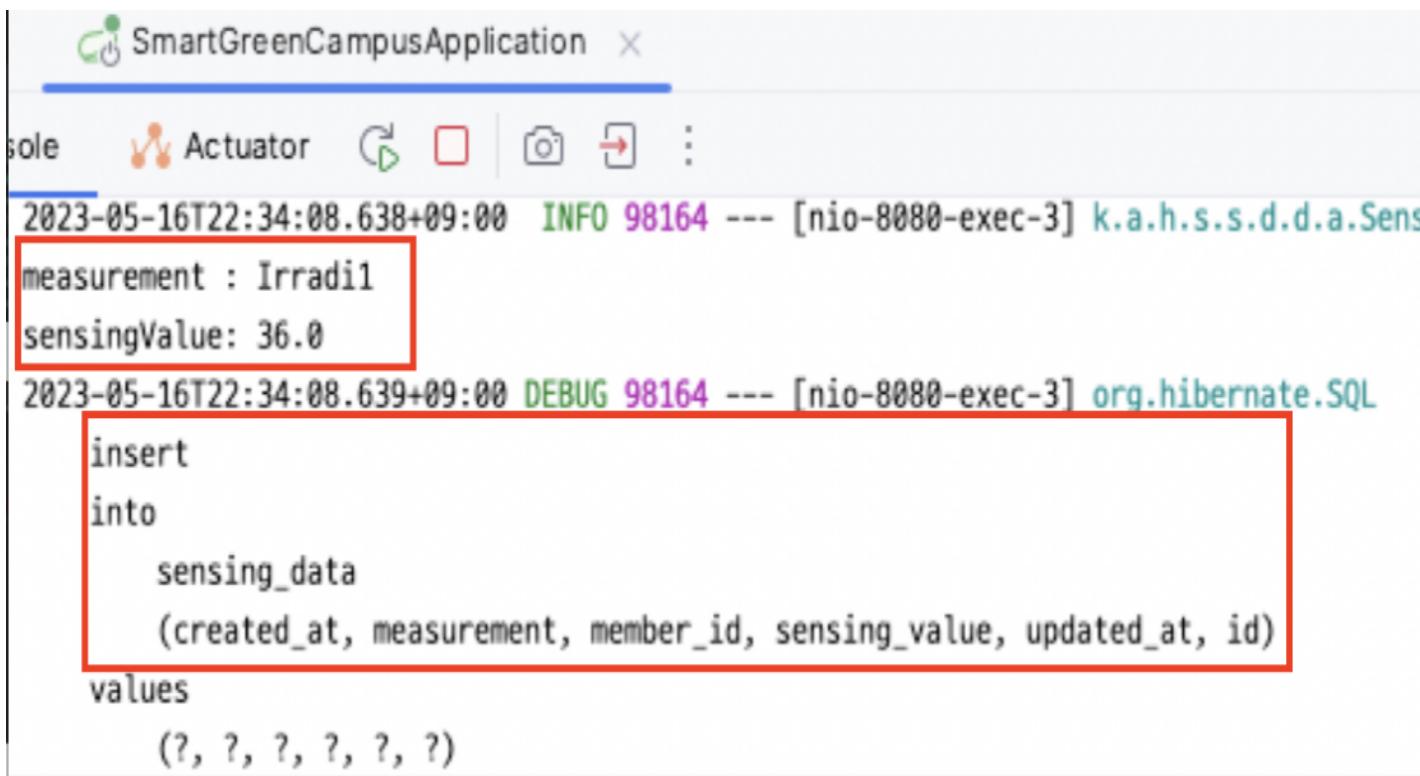
        final WebClient webClient = WebClient
            .builder()
            .baseUrl(mqttConfig.baseUrl)
            .defaultHeader(CONTENT_TYPE, APPLICATION_JSON_VALUE)
            .build();

        final Map<String, Object> response = (Map<String, Object>) webClient
            .post() RequestBodyUriSpec
            .uri(mqttConfig.apiUrl) RequestBodySpec
            .bodyValue(bodyMap) RequestHeadersSpec<capture of ?>
            .retrieve() ResponseSpec
            .bodyToMono(Map.class) Mono<Map>
            .block();
    }
}
```

WebClient 라이브러리를 사용하여 DB파트에 저장한 데이터를 전송합니다.

진행 상황

DB(Back-End파트)로 데이터 전송



The screenshot shows the Actuator dashboard for the SmartGreenCampusApplication. The top navigation bar includes links for Sole, Actuator, and other monitoring tools. The main content area displays log entries and an SQL query.

Log Entries:

```
2023-05-16T22:34:08.638+09:00 INFO 98164 --- [nio-8080-exec-3] k.a.h.s.s.d.d.a.Sens:  
measurement : Irradi1  
sensingValue: 36.0
```

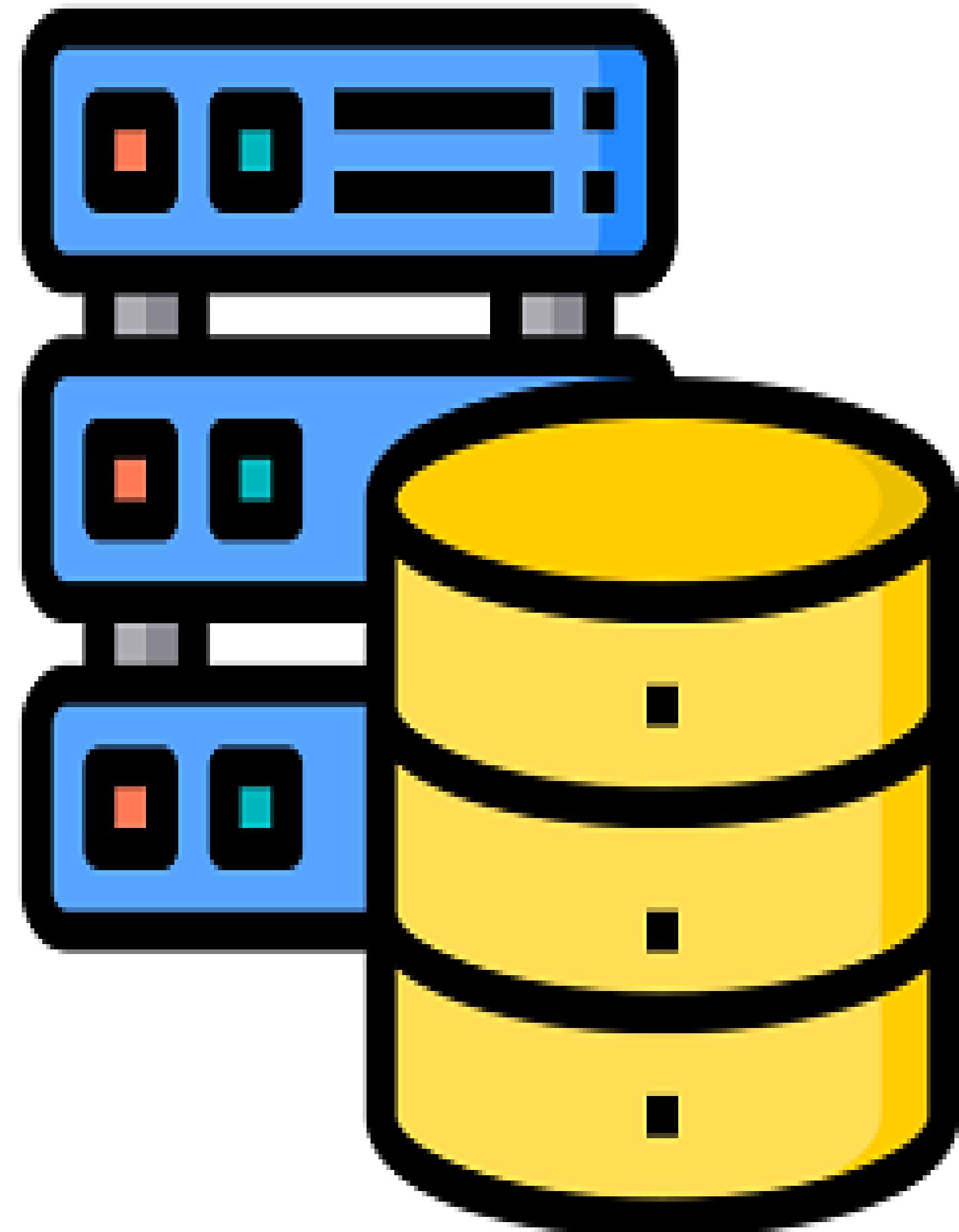
SQL Query (highlighted with a red box):

```
2023-05-16T22:34:08.639+09:00 DEBUG 98164 --- [nio-8080-exec-3] org.hibernate.SQL  
insert  
into  
    sensing_data  
(created_at, measurement, member_id, sensing_value, updated_at, id)  
values  
(?, ?, ?, ?, ?, ?)
```

DB파트에서 데이터를 수신받고 저장되는 과정을
확인할 수 있습니다.

향후 계획

1. 컴퓨터공학과(Publisher)팀과 실제 연동을 해볼 예정입니다.
2. DB(Back-End)파트에서 구현한 사용자 인증에 대한 토큰을 Publisher로부터 전달받아, 데이터와 함께 전송하는 로직을 구현할 계획입니다.

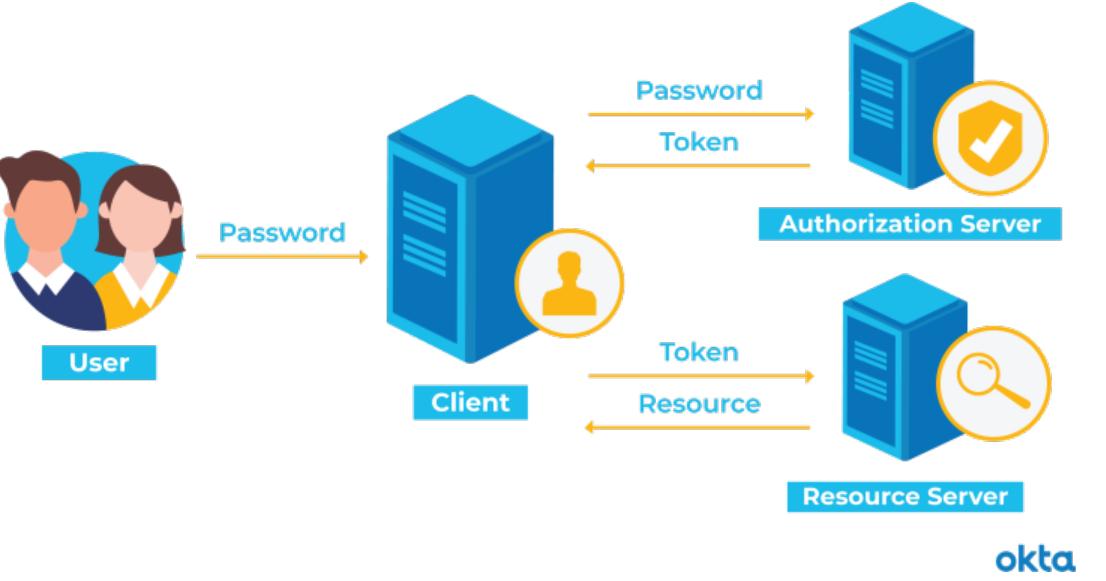


Back-End

02

이전 발표 내용

토큰을 이용한 인증절차

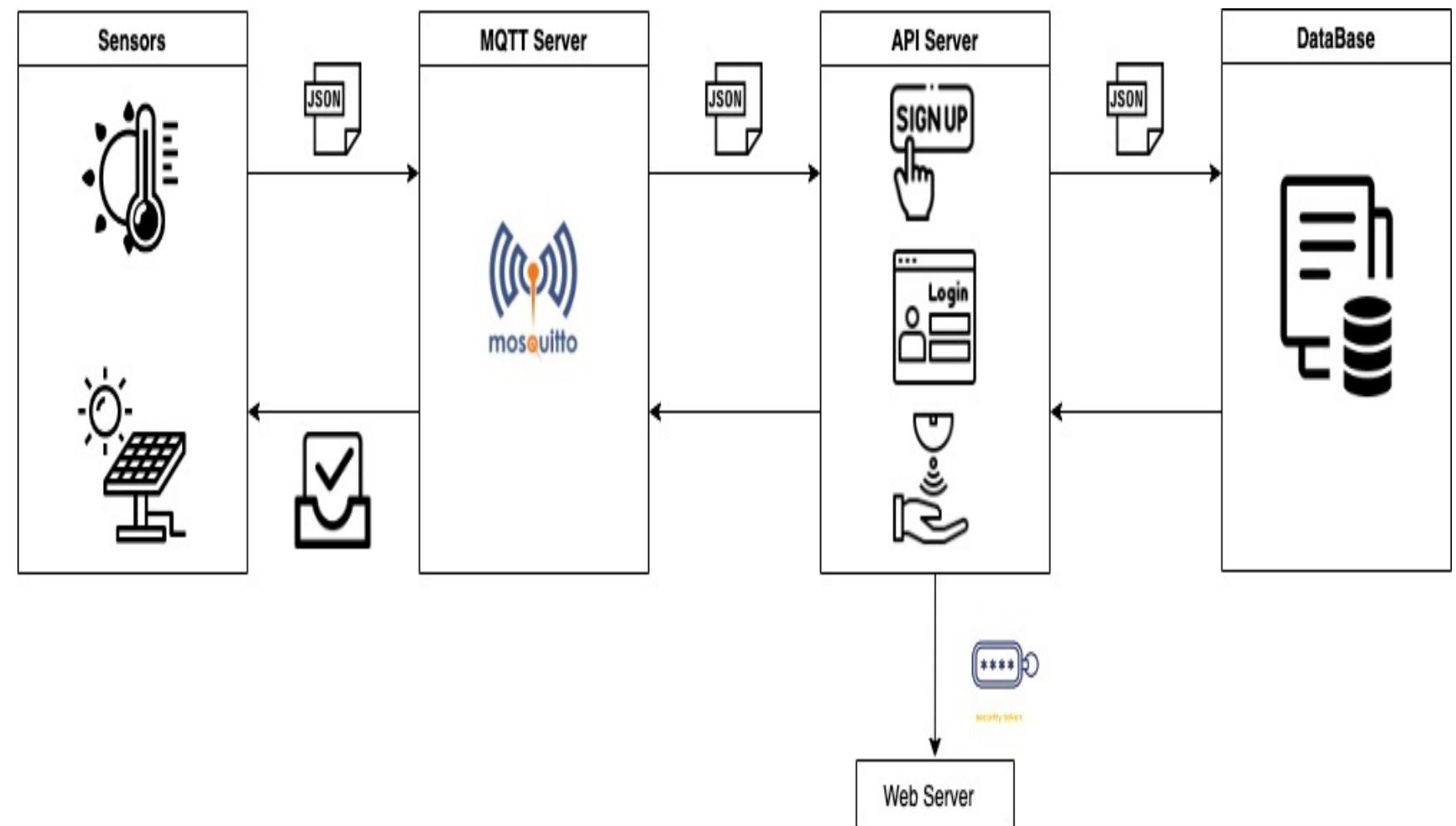


токен은 발급 받으면 사용자의 브라우저에 저장되며, 관리자는 토큰에 대한 사용 제한을 설정하여 로그아웃하거나, 지정된 시간이 지나면 토큰을 자동으로 파기되도록 설정할 수 있습니다.

The screenshot displays three windows of an IDE:

- Java Code Editor:** Shows the `SmartGreenCampus` application's `SensorController` class with methods for creating, reading, updating, and deleting sensors.
- Database Explorer:** Shows the `smart-green-campus` database structure, including tables for `sensor`, `user`, and `location`.
- Test Results:** Shows JUnit test results for the `SensorController` class, indicating successful runs across various test cases.

센서 CRUD



Controller에서 요청을 받으면 Service에서 처리합니다.

해당 코드는 인증 절차를 거쳐서 발급 받은 토큰으로 센서들의 데이터를 처리하는 코드입니다.

진행 상황

협업을 통한 효율적인 개발

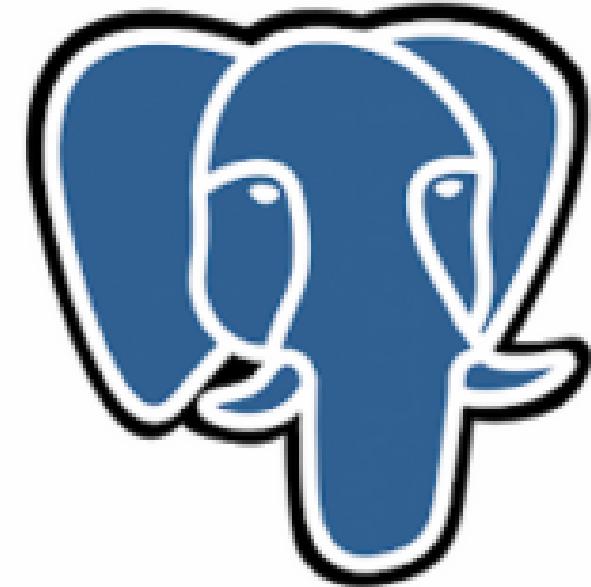
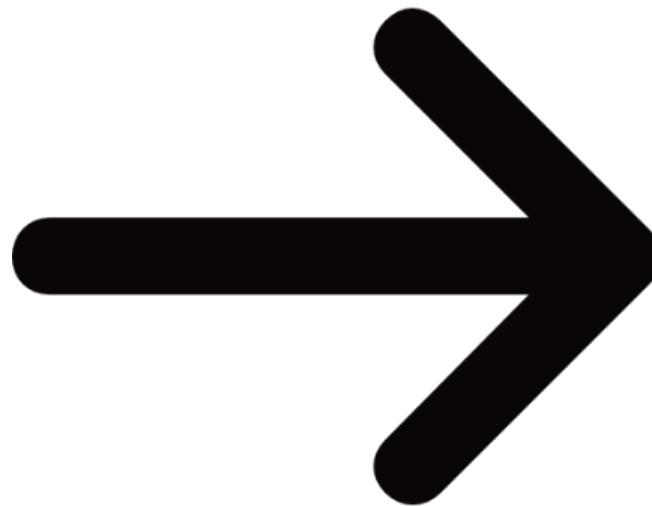
The screenshot shows the GitHub Organization page for 'HBNU-smart-green-campus'. The top navigation bar includes links for Overview, Repositories (3), Projects, Packages, Teams, People (5), and Settings. A search bar at the top allows finding a repository, and dropdown menus for Type, Language, and Sort are available. A green button labeled 'New repository' is also present. Below the header, three repositories are listed:

- mqtt_publisher** (Private)
Java 0 stars 0 forks 0 issues Updated 3 days ago
- smart-green-campus-mqtt-v3** (Public)
Java 2 stars 0 forks 0 issues Updated 3 days ago
- smart-green-campus-api-server-v3** (Public)
smart-green-campus API server v3
Java 1 star 0 forks 0 issues Updated 3 days ago

FE, BE, MQTT 팀들과 효율적인 개발을 위해 Github Organization을 이용하여 코드 및 진행상황을 공유하였습니다.

진행 상황

Database 이전

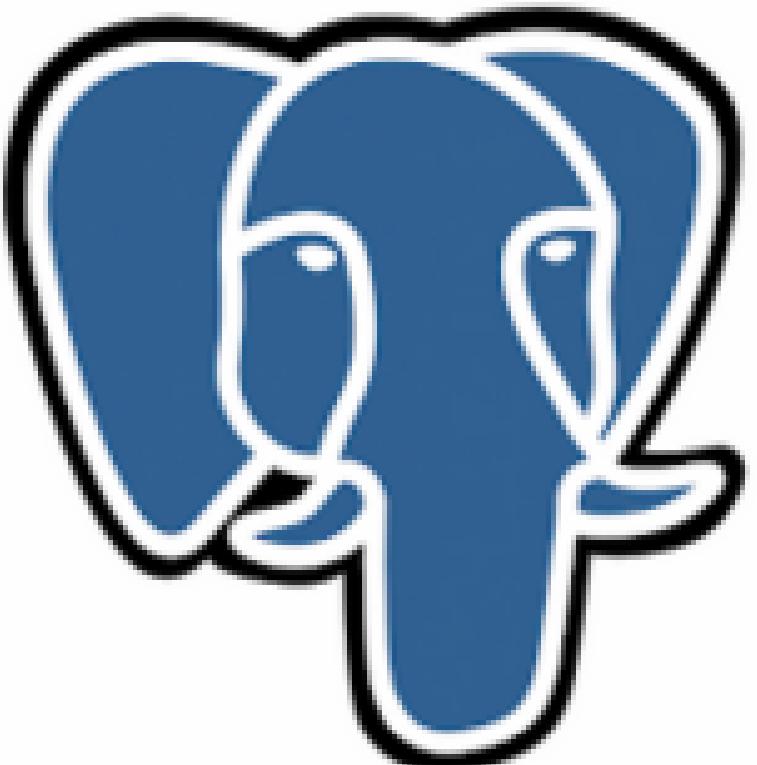


PostgreSQL

H2는 In Memory DB로, 빠르고 가볍다는 장점이 있지만, 안정성과 성능이 부족하고, 백업, 복구 등에 대한 기능 부족하다는 단점이 있습니다.

진행 상황

Why PostgreSQL ?



PostgreSQL

장점

1. 교착상태가 거의 발생하지 않고 트랜잭션 속도가 빠른 MVCC
2. 고가용성을 가지며, 서버 장애 복구에 강점이 있다.
3. 데이터 암호화, SSL 인증서, 고급 인증 방법과 같은 보안기능
4. 활발한 오픈소스 커뮤니티가 지속적으로 솔루션을 개선

단점

1. 초보자에게는 사용이 어려울 수 있음

- MVCC : 하나의 트랜잭션에서 데이터에 접근하는 경우 데이터의 다중버전 상태 중 보장되는 버전에 맞는 값을 반환하여 처리하는 방법을 의미합니다.

진행 상황

API 스펙 조정

```
@Entity  
@Getter  
@NoArgsConstructor(access = AccessLevel.PROTECTED)  
public class SensingData {  
  
    @Id  
    @GeneratedValue  
    private Long id;  
  
    private Double sensingValue;  
  
    @Enumerated(value = EnumType.STRING)  
    private SensingKind kind;  
  
    @Embedded  
    private Location location;  
  
    @CreatedDate  
    @Column(updatable = false)  
    private LocalDateTime createdAt;  
  
    @LastModifiedDate  
    private LocalDateTime updatedAt;  
  
    @ManyToOne(fetch = FetchType.LAZY)  
    @JoinColumn(name = "member_id")  
    private Member member;
```

```
@Entity  
@Getter  
@NoArgsConstructor(access = AccessLevel.PROTECTED)  
public class SensingData {  
  
    @Id  
    @GeneratedValue  
    private Long id;  
  
    private Double sensingValue;  
  
    @Enumerated(value = EnumType.STRING)  
    private Measurement measurement;  
  
    @CreatedDate  
    @Column(updatable = false)  
    private LocalDateTime createdAt;  
  
    @LastModifiedDate  
    private LocalDateTime updatedAt;  
  
    @ManyToOne(fetch = FetchType.LAZY)  
    @JoinColumn(name = "member_id")  
    private Member member;
```

샘플 데이터를 받아 필요없는 값 제거 및 필드명 수정하여 API 스펙을 조정하였습니다.

향후 계획

1. 데이터를 삽입하거나 삭제하는 등의 작업에 대한 인증 / 인가 정책 절차를 구현할 예정입니다.
2. 현재 API server에는 회원가입 및 로그인을 구현해놓았지만, MQTT Client에서 토큰을 요청에 싣어 보내도록 동작 연동을 구현할 예정입니다.

Front-End



03

이전 발표 내용

The image shows a split-screen development environment. On the left is Visual Studio Code with the file `index.js` open. The code uses React and ReactDOM to render an `App` component. It also includes `reportWebVitals` for performance monitoring. A terminal window at the bottom shows ESLint warnings. On the right is a browser window displaying a React application titled "Smart Green Campus". The page has a heading "온도" and a section for "데이터 측정 결과" showing three data points: N4 (13°), N6 (12°), and S8 (9°) with their respective timestamps.

```
import React from 'react';
import ReactDOM from 'react-dom/client';
import './index.css';
import App from './App';
import reportWebVitals from './reportWebVitals';

const root = ReactDOM.createRoot(document.getElementById('root'));
root.render(
  <React.StrictMode>
    <App />
  </React.StrictMode>
);

// If you want to start measuring performance in your app, pass a function
// to log results (for example: reportWebVitals(console.log))
// or send to an analytics endpoint. Learn more: https://bit.ly/CRA-vitals
reportWebVitals();
```

문제 1 출력 디버그 콘솔 터미널

Compiled with warnings.
Compiled with warnings.
Compiled with warnings.
Compiled with warnings.
[eslint]
src\component\page\PostViewPage.jsx
Line 56:20: Expected '===' and instead saw '==' [egeqeq](#)
Search for the [keywords](#) to learn more about each warning.
To ignore, add // eslint-disable-next-line to the line before.

React App

localhost:3000/...

Smart Green Campus

온도

각 건물에서 측정한 온도를 나타냅니다.

데이터 측정 결과

N4 13° / 2022-11-03 PM 6:00

N6 12° / 2022-11-03 PM 7:14

S8 9° / 2022-11-03 PM 8:30

React.js를 활용하여 임의의 데이터를 보여줄 수 있는
간단한 웹 페이지를 구현하였습니다.

진행 상황 - 프로토타입

Smart Green
Campus

건물 ▼

센서 ▼

국립 한밭대학교 <하랑> 동아리의
Smart Green Campus에 오신 것을 환영합니다!

건물 바로가기

센서 바로가기

건물별, 센서별 데이터를 보여주는 웹 애플리케이션을 구현하기에 앞서,
새로운 UI에 대한 프로토타입을 만들었습니다.

진행 상황 - 프로토타입

● 건물별

Smart Green Campus

건물 ▾

센서 ▾

S0 국제교류관

온도 : 23°C
습도 : 31%
전기 사용량 : 5,607kWh

S1 도서관

온도 : 23°C
습도 : 31%
전기 사용량 : 5,607kWh

S2 학생회관

온도 : 23°C
습도 : 31%
전기 사용량 : 5,607kWh

S3 경상학관

온도 : 23°C
습도 : 31%
전기 사용량 : 5,607kWh

S4 인문관

온도 : 23°C
습도 : 31%
전기 사용량 : 5,607kWh

S5 산학협동관

온도 : 23°C
습도 : 31%
전기 사용량 : 5,607kWh

« 1 2 »



● 센서별

Smart Green Campus

건물 ▾

센서 ▾

온도

S0 국제교류관 23°C
S1 도서관 23.5°C
S2 학생회관 23°C
S3 경상학관 24°C
S4 인문관 22.8°C
S5 산학협동관 24.1°C

습도

S0 국제교류관 31%
S1 도서관 30%
S2 학생회관 31.1%
S3 경상학관 31%
S4 인문관 30.5%
S5 산학협동관 31%

전기

S0 국제교류관 5,607kWh
S1 도서관 5,607kWh
S2 학생회관 5,607kWh
S3 경상학관 5,607kWh
S4 인문관 5,607kWh
S5 산학협동관 5,607kWh



Card UI로 구성함으로써 텍스트 등 다양한 요소를 포함할 수 있고,
새로운 데이터를 추가하거나 불필요한 데이터를 삭제하는 등 유지보수가 쉽도록 하였습니다.

향후 계획

1. 새로워진 화면 설계를 React.js로 구현할 예정입니다.

**THANK
YOU**