

SMART GREEN CAMPUS

<정보통신공학과>

20181709 백승준

20181643 서동권

20181745 윤진원

20181676 이동엽

20181680 이진욱

20181692 최민석

20201849 임세나



Index.

01. MQTT Broker

02. Back-End

03. Front-End

MQTT Broker

01



이전 발표 내용

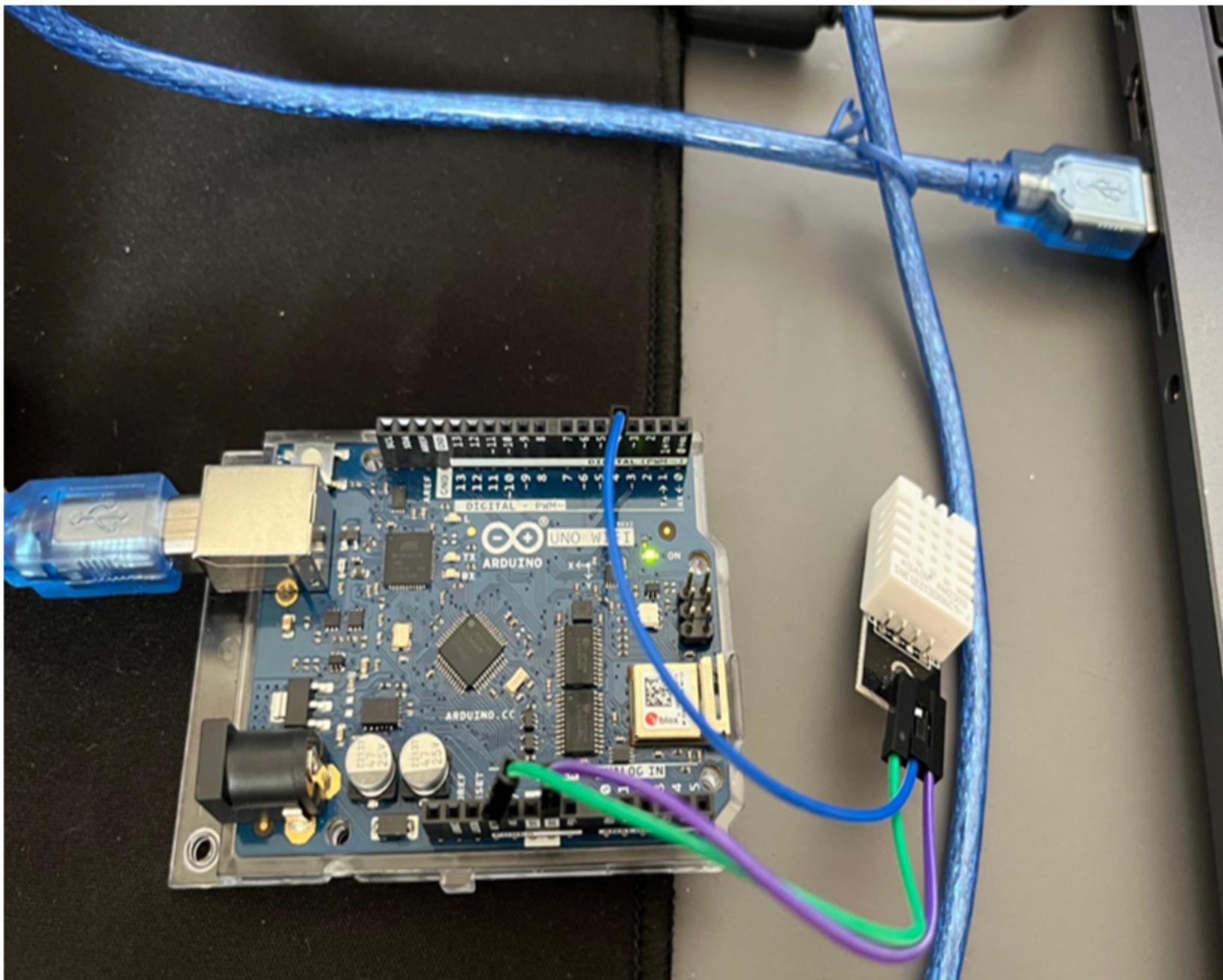
데이터를 송신하여 원하는 포맷으로 수신해보기

```
let message = {  
    "sensor_name": "temperature",  
    "location": "N13",  
    "value": 26  
};
```

```
setInterval(  
() => {  
    message.value=(Math.floor(Math.random()*35));  
  
    client.publish( topic: "green-campus/sensors", JSON.stringify(message));  
},  
2000  
);
```

```
/home/jinuk/.asdf/shims/node /mnt/c/Users/lee99/Developments/smart-green-ca  
{"sensor_name":"temperature","location":"N13","value":27}  
{"sensor_name":"temperature","location":"N13","value":26}  
{"sensor_name":"temperature","location":"N13","value":8}  
{"sensor_name":"temperature","location":"N13","value":33}  
{"sensor_name":"temperature","location":"N13","value":23}  
{"sensor_name":"temperature","location":"N13","value":20}  
{"sensor_name":"temperature","location":"N13","value":17}  
{"sensor_name":"temperature","location":"N13","value":8}  
{"sensor_name":"temperature","location":"N13","value":10}  
{"sensor_name":"temperature","location":"N13","value":25}  
{"sensor_name":"temperature","location":"N13","value":10}  
{"sensor_name":"temperature","location":"N13","value":30}  
^C  
Process finished with exit code 0
```

진행 상황



아두이노의 온도센서 연결

```
17_MQTT_TX_04    arduino_secrets.h
#define SECRET_SSID
#define SECRET_PASS
#define SECRET_PublishTopic "smartgreen"

3
subclient.subscribe(topic: "smartgreen");

-----  

PubMsg = "temperature,N5,";
mqttClient.beginMessage(PublishTopic);
mqttClient.print(PubMsg);
mqttClient.print(event.temperature);
```

아두이노 코드와 클라이언트 코드를 수정

진행 상황

```
subclient.on( event: 'message' , cb: function (topic : string , message : Buffer )  
  //let value = message.toString('utf-8');  
  let [sName,loc,tmp]=message.toString().split( separator: ",")  
  .
```

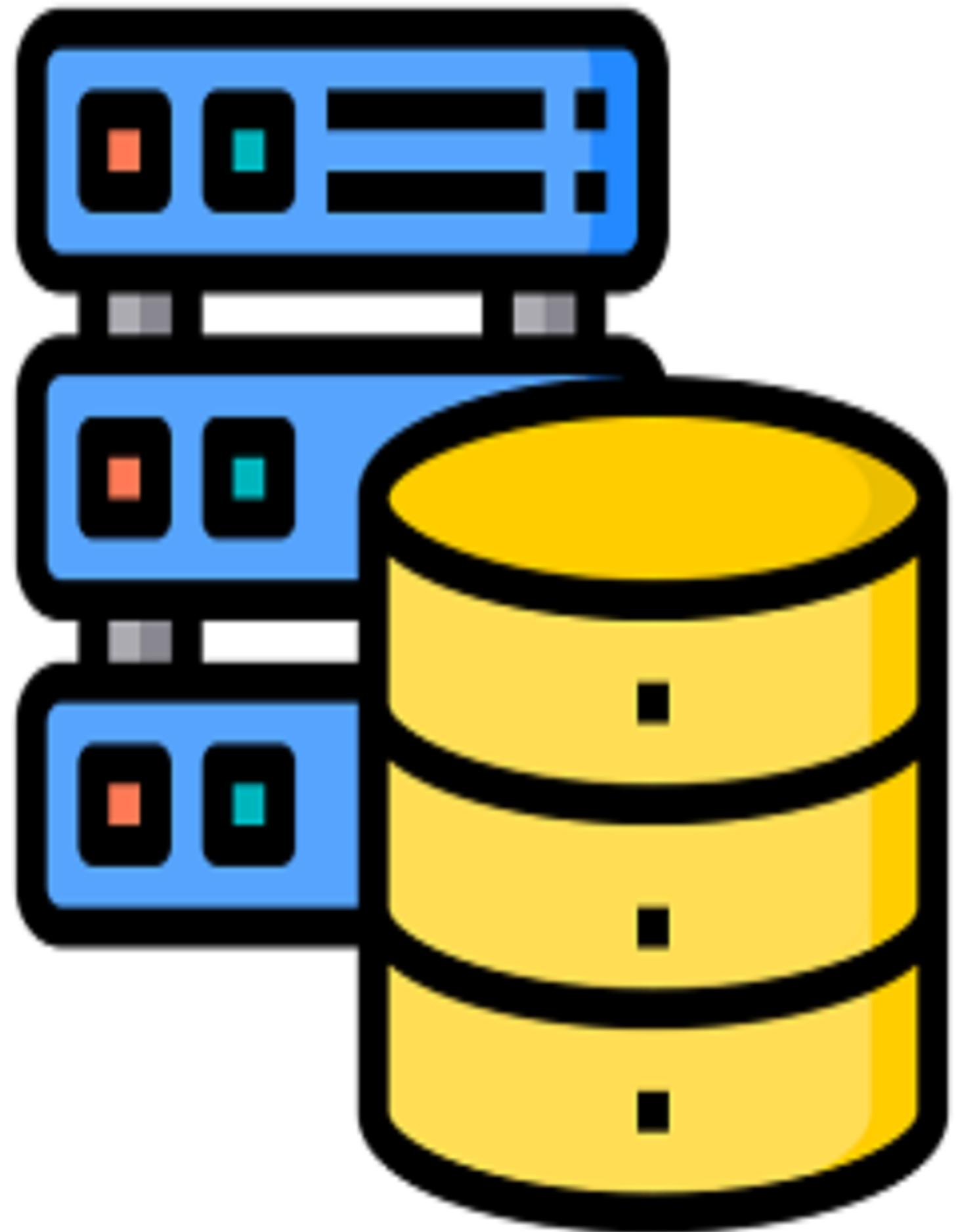
```
{"sensor_name":"temperature","location":"N5","value":"31.30"}  
 {"sensor_name":"temperature","location":"N5","value":"31.30"}  
 {"sensor_name":"temperature","location":"N5","value":"31.50"}  
 {"sensor_name":"temperature","location":"N5","value":"31.60"}  
 {"sensor_name":"temperature","location":"N5","value":"31.40"}  
 {"sensor_name":"temperature","location":"N5","value":"31.40"}  
 {"sensor_name":"temperature","location":"N5","value":"31.30"}  
 {"sensor_name":"temperature","location":"N5","value":"31.30"}  
 {"sensor_name":"temperature","location":"N5","value":"31.10"}
```

수신하는 데이터를 적절한 포맷으로 변경

향후 계획

1. 온도 센서 이외에도 습도나 다른 센서를 이용해서 측정값을 수신해 볼 예정입니다.
2. 차후에 DB파트쪽과 송수신을 통해 데이터 포맷을 수정할 계획입니다.
3. Node 기반의 코드이기 때문에 Nest 기반의 코드로 변경해야 합니다.

Back-End



02

이전 발표 내용

데이터베이스와 연동하고 저장될 값과 메소드를 정의

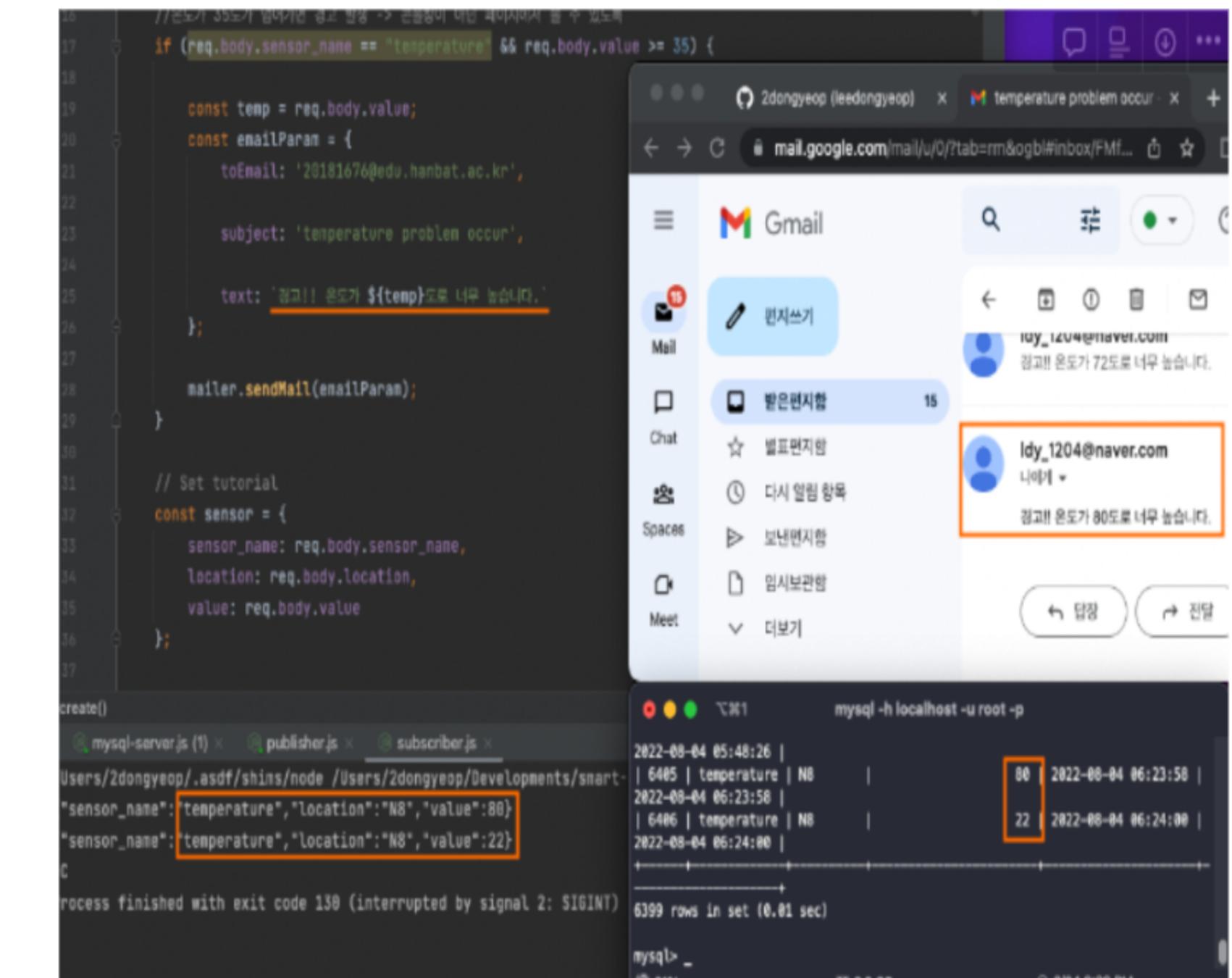
```
1 module.exports = (sequelizeConfig, Sequelize) => {
2   // Set Model
3   const Sensor = sequelizeConfig.define(
4     'sensor',
5     {
6       sensor_name: {
7         type: Sequelize.STRING,
8         allowNull: false,
9       },
10      location : {
11        type: Sequelize.STRING,
12        allowNull: false,
13      },
14      value: {
15        type: Sequelize.DOUBLE,
16        allowNull: false,
17        // unique: true,
18      }
19    }
20  );
21  return Sensor;
22};
```

```
api-server > app > mysql > route > route.js > ...
1  const router = require('express').Router();
2  const sensor = require('../controller/controller.js');
3
4  // Create tutorial
5  router.post('/api/sensors', sensor.create);
6
7  // Retrieve all tutorials
8  router.get('/api/sensors', sensor.findAll);
9
10 // Retrieve tutorial by id
11 router.get('/api/sensors/:id', sensor.findOne);
12
13 // Update tutorial by id
14 router.put('/api/sensors/:id', sensor.update);
15
16 // Delete tutorial by id
17 router.delete('/api/sensors/:id', sensor.delete);
18
19 module.exports = router;
```

진행 상황

nodemailer 모듈을 이용한 알림 구현

```
16 //온도가 35도가 넘어가면 경고 발생 -> 콘솔창이 아닌 페이지에서 볼 수 있도록
17 if (req.body.sensor_name == "temperature" && req.body.value >= 35) {
18
19     const temp = req.body.value;
20
21     const emailParam = {
22
23         toEmail: '20181676@edu.hanbat.ac.kr',
24
25         subject: 'temperature problem occur',
26
27         text: `경고!! 온도가 ${temp}도로 너무 높습니다.`
28
29
30     };
31
32     mailer.sendMail(emailParam);
33 }
```



nodemailer의 sendMail를 사용하여 메일 전송에 주소와 내용을 설정

클라이언트로 부터 전송된 값을 DB에 저장하고, 일정 값 이상일때 메일 전송

향후 계획

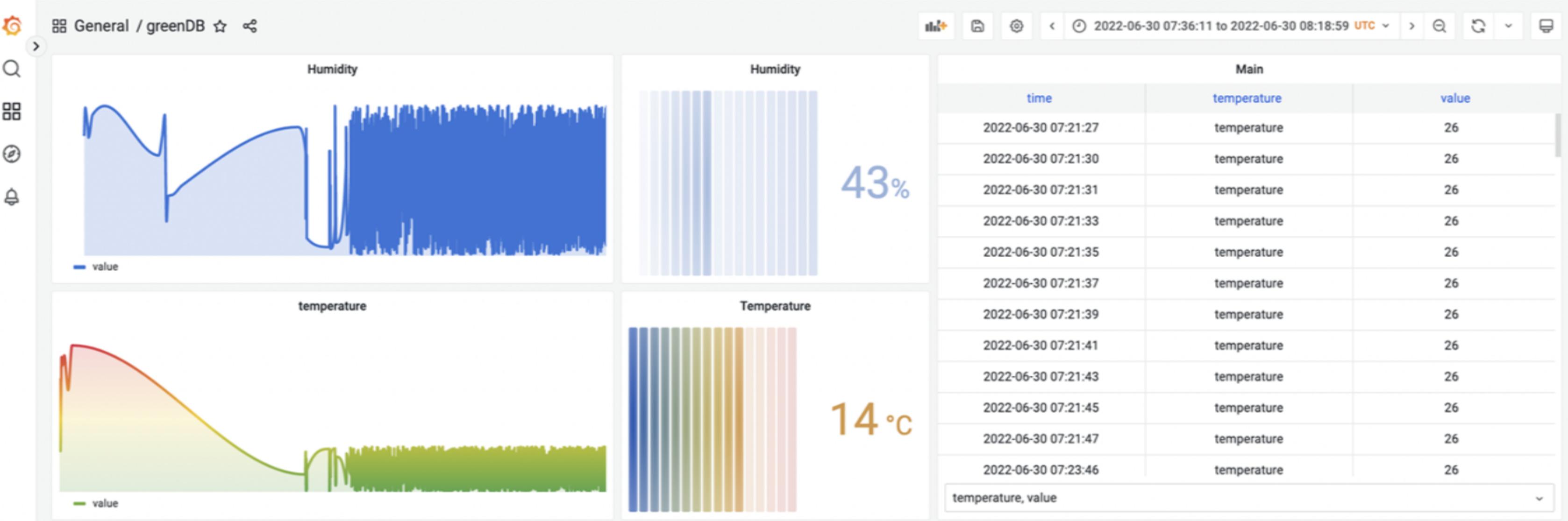
1. 현재 개인 이메일 계정을 통해 송수신을 연습하였기 때문에 추후에 공식 이메일을 생성해야 합니다.
2. 온도 이외에도 습도, 일조량 등에 대해 알림을 보낼 센서 값의 기준을 정해야 합니다.
3. Node 기반의 코드이기 때문에 Nest 기반의 코드로 변경해야 합니다.
4. 메일 전송 대신 Kakaotalk 과 같은 메신저 앱과 연동하기 위해 Firebase와 연동해야 합니다.

Front-End



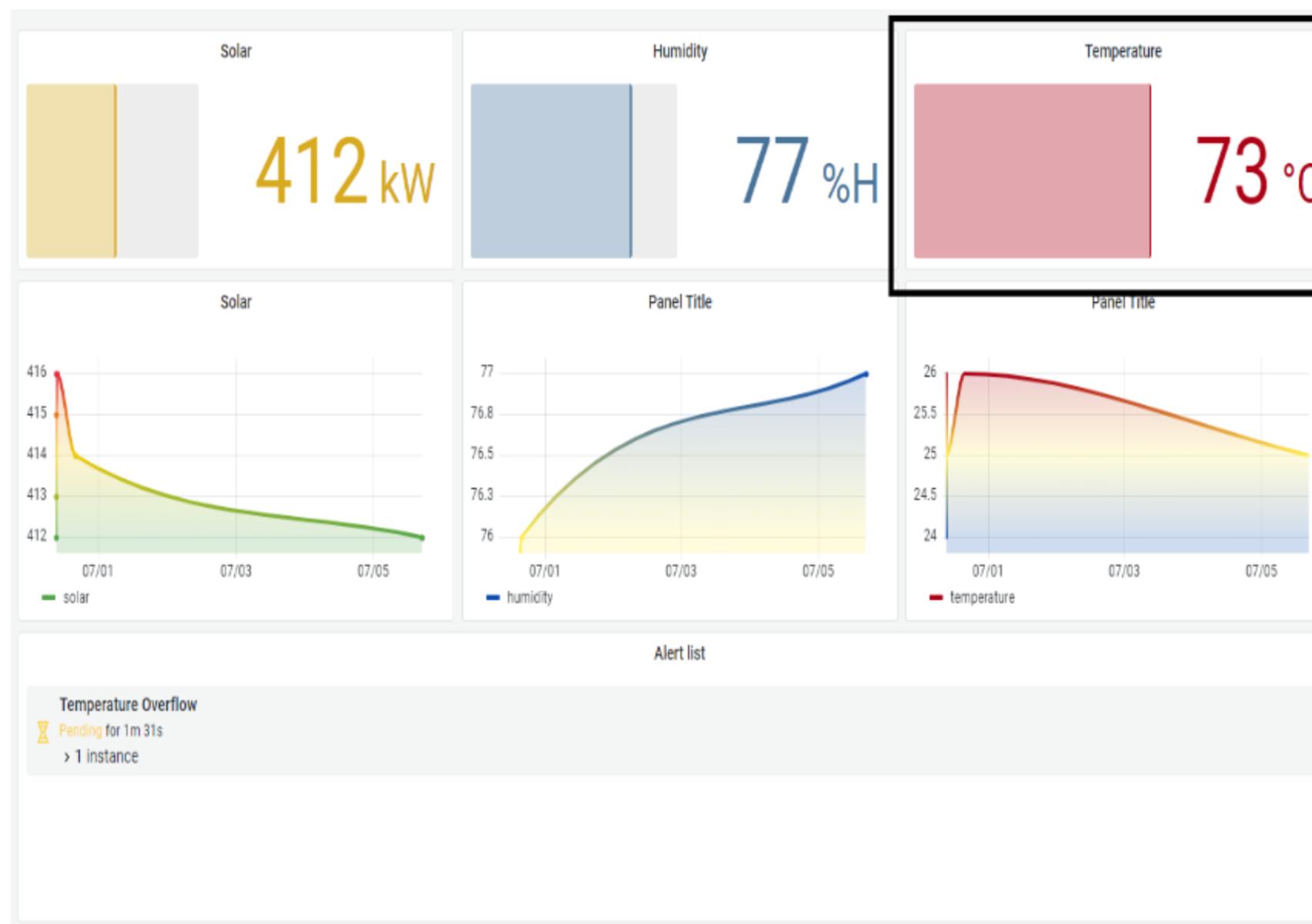
0
3

이전 발표 내용



Grafana를 활용해 DataBase와 연결하여 Graph, Chart, Table 등 여러가지 Panel을 구성함으로써 데이터를 시각화하였다.

진행 상황 - 디자인 수정 및 Alert 기능 추가



grafana-alert ▾

+ 책갈피 추가

Grafana v9.0.1 오늘 오후 3:03

Grafana-alert 오전 3:17

[FIRING:1] (Temperature Overflow)

Firing

Value: [var='B0' metric='temperature' labels={} value=73]

Labels:

- alertname = Temperature Overflow

Annotations:

- description = 온도가 섭씨 40도가 넘어갈 경우 알림
- summary = 온도 값

Source: <http://localhost:3000/alerting/grafana/TYtUv8gVz/view>

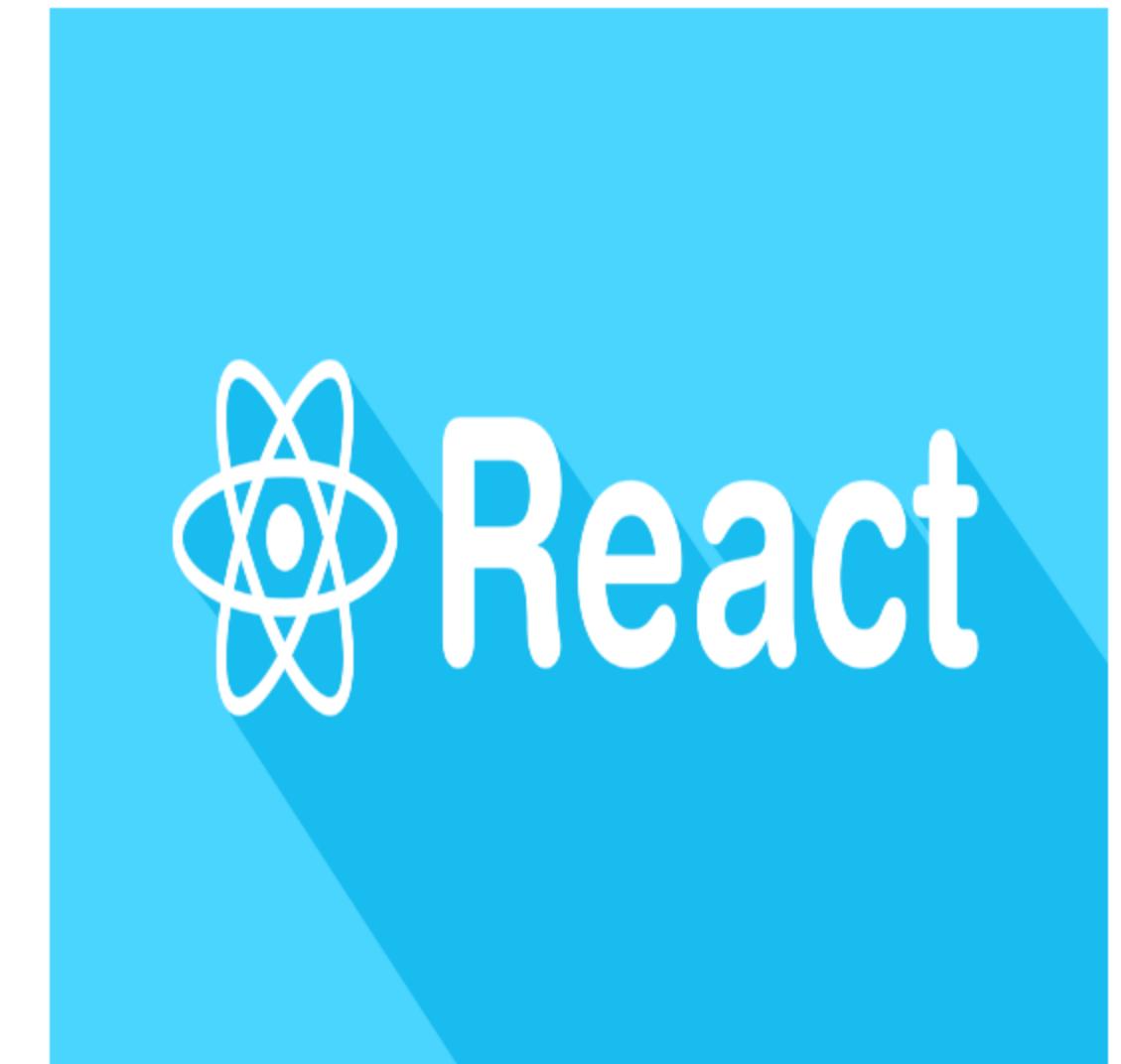
Silence: <http://localhost:3000/alerting/silence/new?alertmanager=grafana&matcher=alertname%3DTemperature+Overflow>

간략히 보기

Grafana v9.0.1 오늘 오후 3:17

Grafana의 Alert 기능을 활용해 설정값 이상의 데이터가 들어올 경우 Panel에 알림이 생성되고 Slack으로 메시지가 전송되는 기능을 추가하였다.

향후 계획



UI 디자인 도구인 **Figma**를 통한 화면 구성 연습과
JavaScript와 React 공부를 병행하여 위와 같은 Panel을 구현할 예정입니다.

**THANK
YOU**