MINISTRY OF EDUCATION AND RESEARCH
TECHNICAL UNIVERSITY
OF CLUJ-NAPOCA, ROMANIA

DISTRIBUTED SYSTEMS

ASSIGNMENT 2

Online Energy Utility Platform

Asynchronous Communication

Reckerth Daniel Peter

30444

Contents

I. Assignment Objective

The aim of this project is to create a system for an online energy utility platform designed to manage clients and their associated devices which are equipped with smart sensors monitoring energy consumption.

In the second assignment we have a smart meter which sends data to a server periodically, of the form <timestamp, sensor_id, measurement_value>.

We deal here with a message broker middleware that gathers this data, pre-process them before database storage. We also have to detect a power peak and if it exceeds the sensor maximum threshold we have to asynchronously notify the client on the web interface. The formula is given below:

$$P_{peak}(t_1, t_2) = \frac{measurement_{value}(t_2) - measurement_{value}(t_1)}{t_2 - t_1} < MAX_{value}$$

II. Functional Requirements

- Message middleware broker allows the sensor system to send data tuples in a JSON format
- The message consumer component of the system processes each message and notifies asynchronously using WebSockets the client

III. Design and Implementation

For developing this system we have decided to use the following technologies:

- REST services for backend application – Java Spring
- Frontend – React
- For database – Postgres
- Deployment – Heroku and Docker
- CloudAMQ and RabbitMQ for the broker
- WebSockets for asynchronously communications
- Login page

While working on the application we have to design and implement a Producer and Consumer. We have decided to construct the Producer in Java, although many other options were possible like writing it in Python. We have worked with **Spring for RabbitMQ.** With any messaging-based application we have to create a producer and consumer. Our producer is a standalone Spring application and the consumer/receiver is inside our main backend application.

Concerning messages between browser and server we used WebSocket which is a thin, lightweight layer above TCP. This makes it suitable for using “subprotocols” to embed messages. In this guide, we use STOMP messaging with Spring to create an interactive web application. STOMP is a subprotocol operating on top of the lower-level WebSocket.

We have created a class called WebSocketConfig which extends the WebSocketMessageBrokerConfigurer. Moreover, we have a WebSocketService were we use a SimpMessagingTemplate in order to send our payload to the front, a string, when a peak is detected in the receiver following the formula above.

IV. Conclusions

What I found difficult in this assignment was the development of the WebSockets because it was not unclear to me from first time how this is achieved.

Developing the message-broker communication proved to be a lot easier.

The deployment, following the provided documentation proved to be manageable.

V. Bibliography

- Provided documentation for Docker and CI/CD on Heroku
- https://spring.io/guides/gs/messaging-stomp-websocket/
- https://spring.io/guides/gs/messaging-rabbitmq/