Distributed Systems

Assignment 2

Raita Anamaria

Group 30441

Table of Contents

[1. Requirements 2](#_Toc468645026)

[2. Conceptual architecture of the distributed system 2](#_Toc468645027)

[3. UML Deployment diagram 3](#_Toc468645028)

[4. Build and execution 3](#_Toc468645029)

# Requirements

Implement a component for Assignment 1 application based on a message broker middleware that gathers data from the smart metering devices, pre-processes the data to compute the hourly energy consumption and stores it in the database.

The message broker allows Smart Metering Device Simulator to act as messages producer and send data tuples in a JSON format.

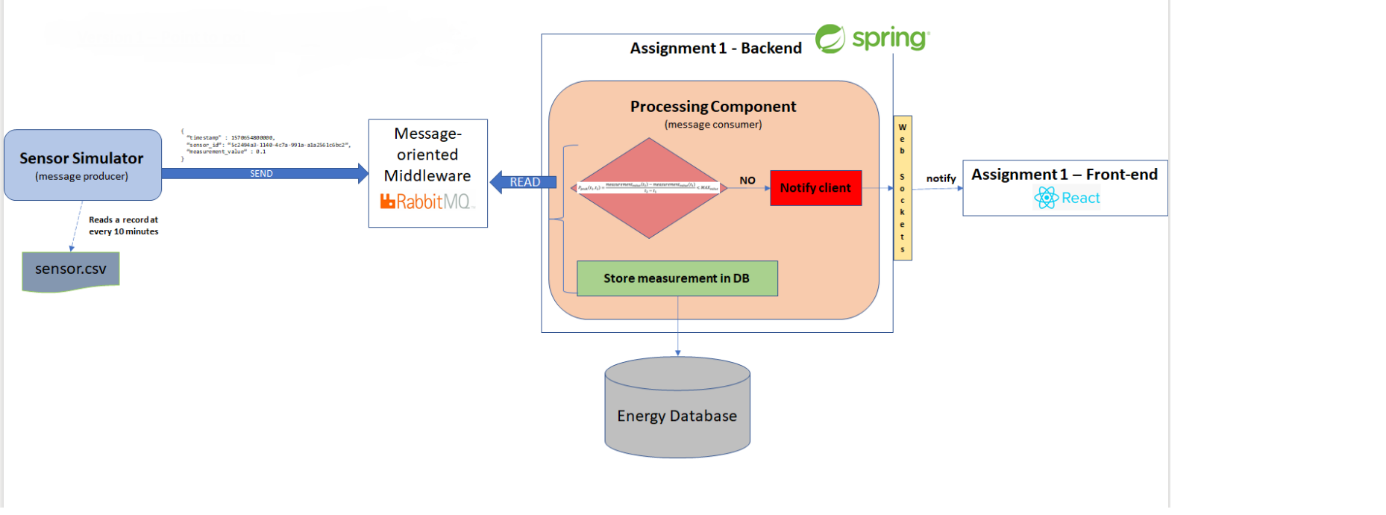
The message consumer component of the system processes each message and notifies asynchronously using WebSockets the client application

Use the following technologies: RabbitMQ, WebSockets

# Conceptual architecture of the distributed system

The architecture on which the system is based on a Message Broker. The application has a Message Producer that produces messages in JSON format and transmits them every 10 minute to the Message Broker, which is a queue called RabbitMQ. From this queue the messages are being consumed by the Message Consumer which in my case is a class with the adnotation of @RabbitListener in the backend application from the first assignment.

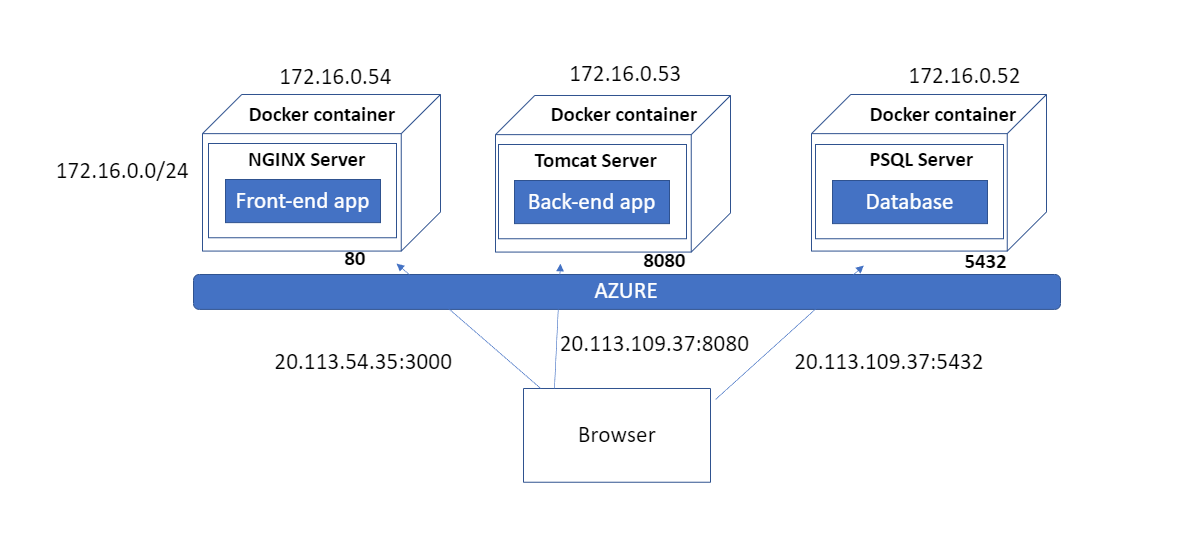
The client, the backend application also processes those messages, it stores them in the database and if the hourly energy consumption for the device exceeds the maximum hourly consumption of the device it will notify the user on the frontend assynchronously with the help of WebSockets.



# UML Deployment diagram

The application consists of a Message Broker, a Message Producer and a Message Consumer and WebSockets integrated in the backend and frontend of the preb

In the picture presented below, you can see the deployment diagram of the application.



# Build and execution

In order to build the source code, you should have installed on your computer the following:

* InteliJ
* Node js (for react)

In order to run the application, having the source code, you should follow these steps:

* Open IntelliJ
* Open the producer desktop application and run it
* Open the spring application (the backend) and run it
* Open Visual Studio Code
* Open the frontend application
* Write in browser localhost:3000 here you can log in as a client, register a device, write the device of the id in the program’s arguments of the producer and run that configuration

If you want to see the application deployed on azure you should just just type in a browser <http://20.113.54.35:3000> .