

**FACULTY OF AUTOMATION AND COMPUTER SCIENCE COMPUTER SCIENCE DEPARTMENT**

**DISTRIBUTED SYSTEMS**

**Assignment 2**

Asynchronous Communication

Sensor Monitoring System and Real-Time Notification

|  |  |  |
| --- | --- | --- |
|  |  |  |

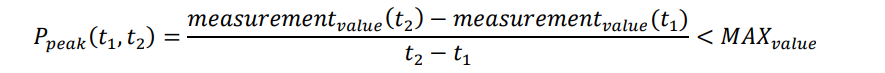
**Student**: Cățănaș-Kelemen Kaj Victor  
**Group**: 30244

2021

# Requirements

The clients of the energy distributor have installed smart meters for each device registered to measure its energy consumption. Each sensor sends data to a server periodically, in the form *(timestamp, sensor id, measurement value)*, where *timestamp* is the time instance when the measurement was made and *measurement value* is the value of the energy counter measuring the total energy consumed by the device in kWh since the sensor was installed.

Implement a system based on a message broker middleware that gathers data from the sensors and pre-processes them before storing them in the database. If the queue consumer application that preprocesses the data detects a measurement power peak that exceeds the sensor maximum threshold (i.e. sensor *maximum value* measure in kW defined in Assignment 1) it notifies asynchronously the client on its web interface. To compute a power peak, the instantaneous power in a measurement interval is computed by averaging the energy consumption and dividing the value to the time interval.



**A Sensor Simulator** will simulate a sensor that reads data from files (sensor.csv), one value at every 10 minutes. The module will contain a timer synchronized with the local clock. The module sends data in the form < *timestamp, sensor id, measurement value* > to the message broker. The timestamp is taken from the local timer, the *measurement value* is read from the file at the corresponding index, representing the energy measured in kWh, and the *sensor id* is unique to each instance of the Sensor Simulator and corresponds to the sensor ID associated to a device of a client from the Energy Database.

The sensor simulator should be developed as a standalone application (i.e. desktop application) to read the sensor monitored activities from the file *sensor.csv,* configured as a message producer and send the monitored sample data to the queue defined. The file *sensor.csv* can be downloaded from <https://dsrl.eu/courses/sd/materials/sensor.csv>. The measurements are sent to the queue.

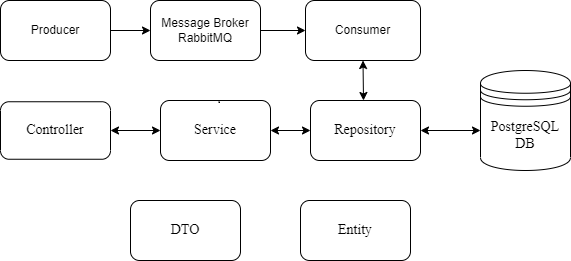
## Functional requirements:

* + - The message-oriented middleware 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 application

## Implementation technologies:

* + - Use the following technologies: RabbitMQ, WebSockets.

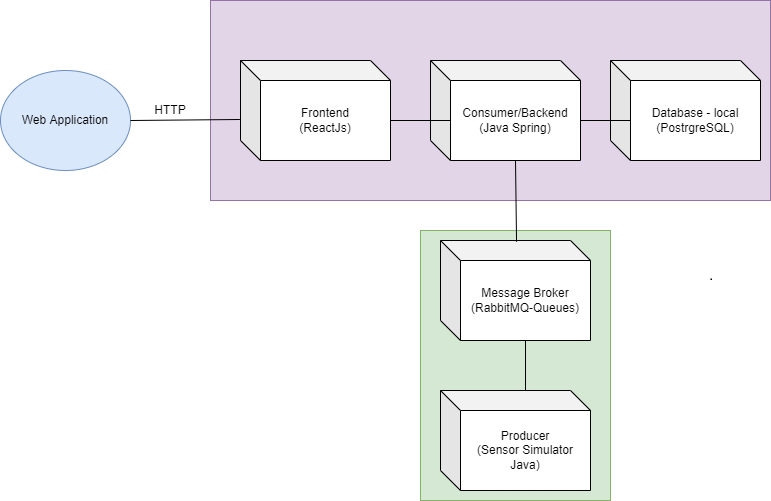
2.) Conceptual architecture of the distributed system



The Controller will receive HTTP requests signaled by the frontend and will send the appropriate reply according to the request. The Service will validate the received data, handle the request (CRUD/login) and access the required data from the Database through the Repository. The data received or sent to the Database by the Repository will be of type Entity and the data sent back or to the Service will be of type DTO (Data Transfer Object).  
  
The Producer will read measurement values from files, send the data through the Message Broker.

The queue Consumer will preprocess the data sent from the broker. If it detects a measurement power peak that exceeds the sensor maximum threshold it notifies asynchronously the client on its web interface. In case the received measurement value does not result in a new power peek, it will be stored in the repository.

3.) UML Deployment Diagram



Client-Server Architecture. The Web application manages through the user-designed frontend to receive and send the appropriate HTTP requests and replies to the backend that will access the corresponding data from the local database.

The message producer will deliver the measurement tuples through the queue in the message broker to the message consumer.

The message-oriented middleware RabbitMQ (open-source message broker software) 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 application.

# Bibliography

* 1. <https://dsrl.eu/courses/sd/>
  2. Lab Book: I. Salomie, T. Cioara, I. Anghel, T.Salomie, *Distributed Computing and Systems: A practical approach*, Albastra, Publish House, 2008, ISBN 978-973-650-234-7
  3. Lab Book: M. Antal, C. Pop, D. Moldovan, T. Petrican, C. Stan, I. Salomie, T. Cioara, I. Anghel, Distributed Systems – Laboratory Guide, Editura UTPRESS Cluj-Napoca, 2018 ISBN 978-606-737-329-5, 2018,

<https://biblioteca.utcluj.ro/files/carti-online-cu-coperta/329-5.pdf>

* 1. <https://spring.io/guides/gs/messaging-stomp-websocket/>
  2. <https://www.rabbitmq.com/documentation.html>