MINISTRY OF EDUCATION
TECHNICAL UNIVERSITY
OF CLUJ-NAPOCA, ROMANIA
FACULTY OF AUTOMATION AND COMPUTER SCIENCE
COMPUTER SCIENCE DEPARTMENT

DISTRIBUTED SYSTEMS

Assignment 2

Asynchronous Communication and Real-Time Notification

Prof. Tudor Cioara
As. Liana Toderean
As. Antonesi Gabriel

S.l. Marcel Antal
As. Alexandru Rancea

Conf. Cristina Pop
As. Dan Mitrea

2024-2025

1. Requirements

Implement a Monitoring and Communication Microservice for the Energy Management System. The microservice is based on a message broker middleware that gathers data from the smart metering devices, processes the data to compute the hourly energy consumption and stores it in the database of the Monitoring and Communication Microservice.
The synchronization between the databases of Device Management Microservice and the new Monitoring and Communication Microservice is made through an event-based system that uses a topic for device changes (sends device information through a queue for the Monitoring and Communication Microservice).
A Smart Metering Device Simulator application will be implemented as the Message Producer. It will simulate a smart meter by reading energy data from a sensor.csv file (i.e., one value at every 10 minutes) and sends data in the form < timestamp, device_id, measurement_value > to the Message Broker (i.e., a queue). The timestamp is taken from the local clock, and the device_id is unique to each instance of the Smart Metering Device Simulator and corresponds to the device_id of a user from the database (as defined in Assignment 1). The device simulator should be developed as a standalone application (i.e., desktop application). The file sensor.csv can be downloaded from https://dsrl.eu/courses/sd/materials/sensor.csv.
The measurements are sent to the queue using the following JSON format:

{

“timestamp": 1570654800000,
“device_id”: “5c2494a3-1140-4c7a-991a-a1a2561c6bc2”
“measurement_value”: 0.1,
}

The Monitoring and Communication Microservice will have a Message Consumer component that will process the measurements to compute the total hourly energy consumption and store it in the database. If the computed total hourly energy consumption exceeds the device defined maximum value (as defined in Assignment 1) it notifies asynchronously the user on his/her web interface.

1.1. Requirements:

- The message-oriented middleware allows the Smart Metering Device Simulator to send data tuples in a JSON format.
- The message consumer component of the microservice processes each message and notifies asynchronously the client application using WebSocket.
- The hourly energy values will be saved by the consumer component in the Monitoring database.

1.2. Implementation technologies:

- Use the following technologies: RabbitMQ, WebSockets.

1.3. Non-functional requirements:

- The Consumer component will be integrated into the Monitoring and Communication Microservice

2. Deliverables

- A solution description document (about 4 pages, Times New Roman, 10pt, Single Spacing) containing:
  a) Conceptual architecture of the distributed system.
  b) UML Deployment diagram.
  c) Readme file containing build and execution considerations.
- Source files. The source files and the database dump will be uploaded on the personal *gitlab* account created at the *Lab resources* laboratory work, following the steps:
  - Create a repository on *gitlab* with the exact name: *DS2024_Group_LastName_FirstName_Assignment_Number*
  - Push the source code and the documentation (push the code not an archive with the code or war files)
  - Share the repository with the user *utcn_dsrl*

3. Evaluation

3.1. Assignment Related Basic Questions:

During assignment evaluation and grading you will be asked details about the following topics:

- Message Oriented Middleware types
- Queue vs Topic
- Point-to-Point vs Publish Subscribe communication.
- Server pushing data to clients: Sockets, WebSocket, Long Polling

3.2. Grading

The assignment will be graded as follows:

Points	Requirements
5 p	**Minimum to pass** • Implement the Monitoring and Communication Microservice with message consumer component. • Implement the Message producer and Message broker. • Display messages extracted from the queue (at least in Console) • Documentation • Correct answers to 3.1 questions
1 p	Store the hourly energy consumption data in database and add the topic for device changes
1 p	Check if the hourly energy data of the device is above the maximum limit defined in Assignment 1. Create WebSocket and push notifications to clients. Display the notification in the Client page developed in Assignment 1.
2 p	Integration with assignment 1: register a client with a device. Set the device ID for the device simulator using a configuration file. Start the device simulator.

	A client can view: his/her historical energy consumption as line charts or bar charts for one day (OX- hours; OY- energy value [kWh] for that hour). Select day from a calendar.
1 p	Run at least two device simulators simultaneously and view the measurements on two client pages by opening the application in two browsers.

***NOTE:** The sensor application should have a configuration file where you can set the device ID for the client for which you want test the application.

4. 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
4. https://spring.io/guides/gs/messaging-stomp-websocket/
5. https://www.rabbitmq.com/documentation.html