Assignment 2
Distributed Systems Laboratory

Energy Management System

Voicu Sara-Ioana
Group: 30444

Introduction

The second assignment involved enhancing the energy management system. It introduced a Monitoring and Communication microservice, responsible for receiving information about devices associated with users, including details such as their maximum consumption and hourly consumption. A Simulator application, functioning as a Message Producer, was employed to transmit consumption data to the monitoring service.

RabbitMQ

RabbitMQ serves as a message broker, responsible for accepting and forwarding messages in a communication setup that involves producers and consumers. To facilitate this communication, a producer transmits messages onto a queue, where they await consumption by a consumer.

In my application, there are two producers and one consumer, necessitating the use of two queues:

- *Consumptions Queue*:
    - Producer: The producer is the simulator application, which reads data from a CSV file line by line. It then transmits this data to the monitoring microservice, sending one value every 10 minutes. Each transmission includes: device ID, consumption value, and timestamp.
    - Consumer: The consumer is the monitoring microservice, tasked with receiving data from the producer. Subsequently, it computes the hourly consumption and stores this information in a database.

- *Devices Queue*:
    - Producer: The device microservice functions as the producer, communicating information about devices linked to a user with the consumer. It actively monitors changes such as device deletions, updates, or the creation of new devices.
    - Consumer: Once again, the monitoring microservice serves as the consumer in this scenario. It is responsible for receiving and processing information about devices, ensuring that only those associated with a user ID are stored in the database.

WebSockets

Implementing WebSocket in the monitoring microservice enables real-time communication between the backend (monitoring microservice) and the frontend of the application. WebSocket is a communication protocol that provides full-duplex communication channels over a single TCP connection, allowing for bidirectional communication between the server and the client.

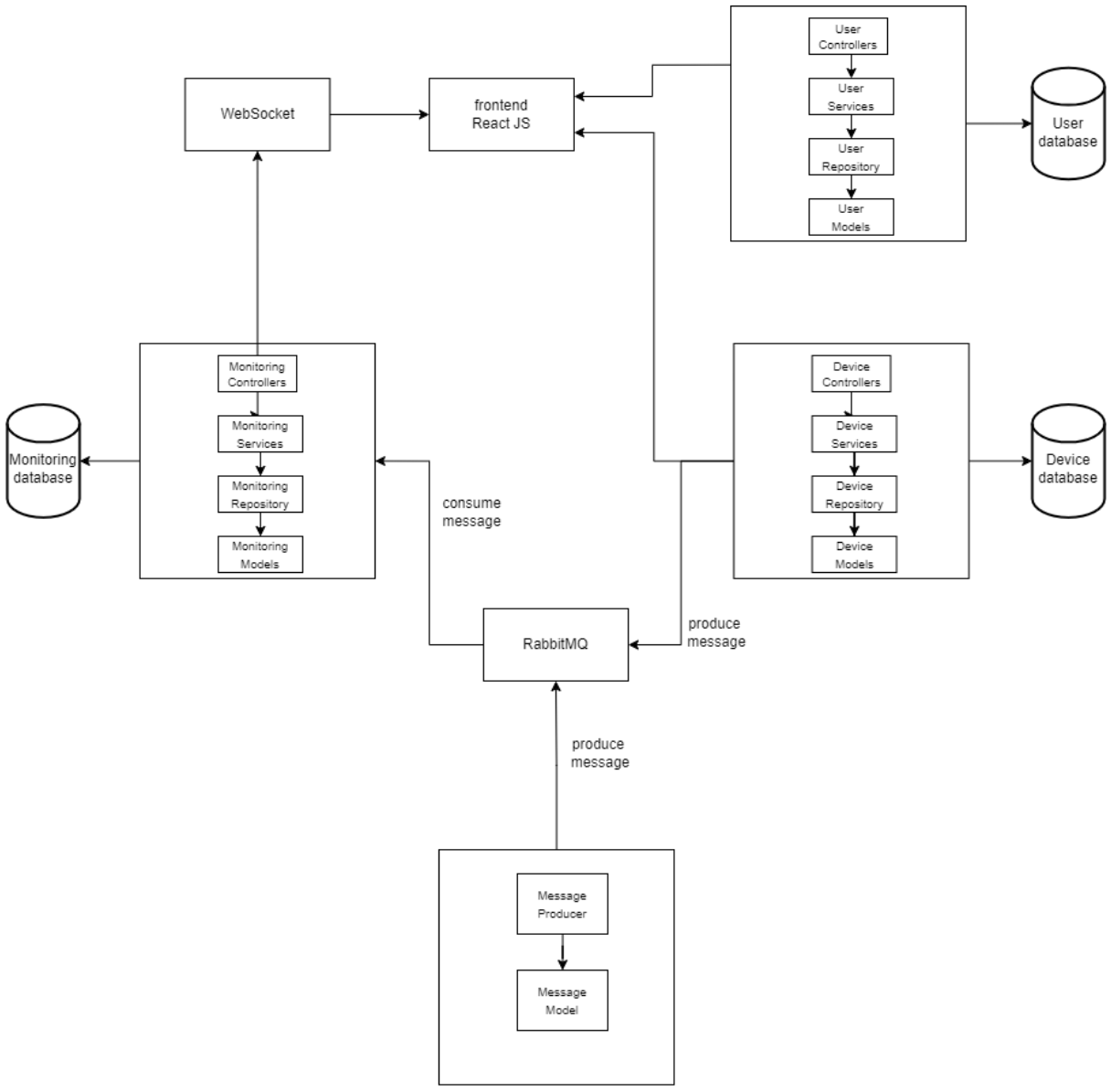
How does WebSocket work in my application?

In the *monitoring microservice*, WebSocket functionality was integrated to facilitate real-time communication with the frontend. When the monitoring microservice calculates details such as hourly consumptions and max consumption for a device, it utilises the WebSocket connection to push these updates to the connected clients.

On the *client side*, the application establishes a WebSocket connection to the monitoring microservice. Through this connection, the frontend is notified in real-time if the hourly consumption exceeds the maximum limit for a specific device. Users receive immediate alerts or notifications on their pages, providing timely awareness of any critical consumption events.

The *frontend* incorporates a charting component. This component dynamically renders the hourly consumption data received via the WebSocket connection. Users can visually interpret consumption trends over time, gaining insights into device usage patterns. To enable users to view hourly consumption on a specific day, the frontend features a calendar component. Users can interact with this component to select a particular day of interest.

Conceptual Architecture



Deployment

In addition to the applications deployed in the initial assignment, I have incorporated two new components: the monitoring microservice and the simulator. The simulator now provides the flexibility to select the device ID for the device being simulated.

The ports mappings are configured to ensure synchronisation according to the following scheme:

User:

- User-microservice: 8080:8080
- User database: 5432:5432

Device:

- Device-microservice: 8081:8081
- Device database: 5433:5432
- RabbitMQ device: 15672:15672

Frontend:

- Frontend: 3000:3000

Monitoring:

- Monitoring microservice: 9001:9001
- Monitoring database: 5431:5432
- RabbitMQ monitoring: 15673:15672

Simulator:

- Simulator application: 9000:9000
- RabbitMQ simulator: 15674:15672