Distributed Systems

Online Energy Utility Platform

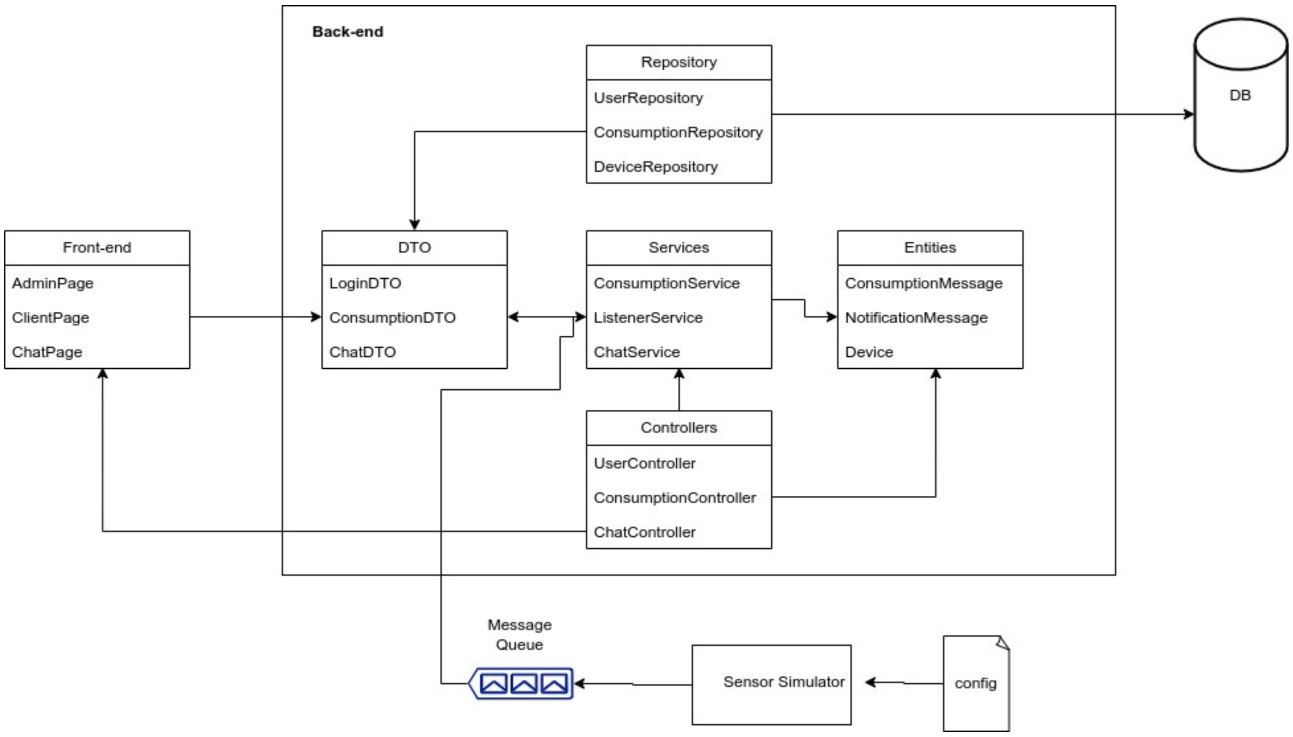
Assignment 3

Bîndilă Adrian-Ioan
30443

This documentation describes the functionality and usage of the Chat System between Administrators and Users of the platform. The chat system is designed to provide real-time support and assistance to the users of the platform. It allows the administrators to communicate with the users and answer any questions they may have related to their energy consumption.

Usage:

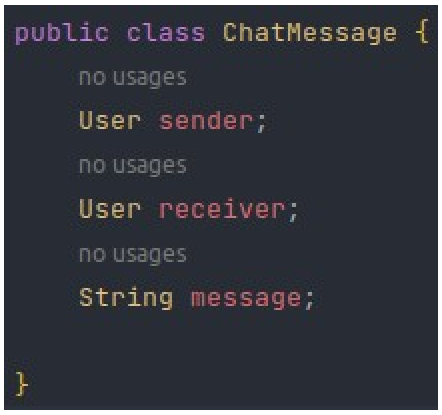
1. The users and administrators need to log in to the platform to access the chat system.
2. Once logged in, the users can ask questions related to their energy consumption and the administrators can answer those questions in real-time.
3. The chat transcripts will be stored on the server for future reference.



The Chat System between Administrators and Users of the platform provides real-time support and assistance to the users of the platform. Its architecture is designed for scalability, security, and the provided client library makes it easy to integrate into existing applications. This system can greatly improve the overall user experience, by providing a fast, easy, and efficient way of communication.

For the server side, the chat system is built on top of a WebSocket library, which will handle connections, message routing and broadcasting messages to connected clients based on the configured channels. Namely, each logged in user is subscribed to a topic based on its username, which enables private communication between admins and clients and ensures that only the two parties involved in the conversation have access to the messages.

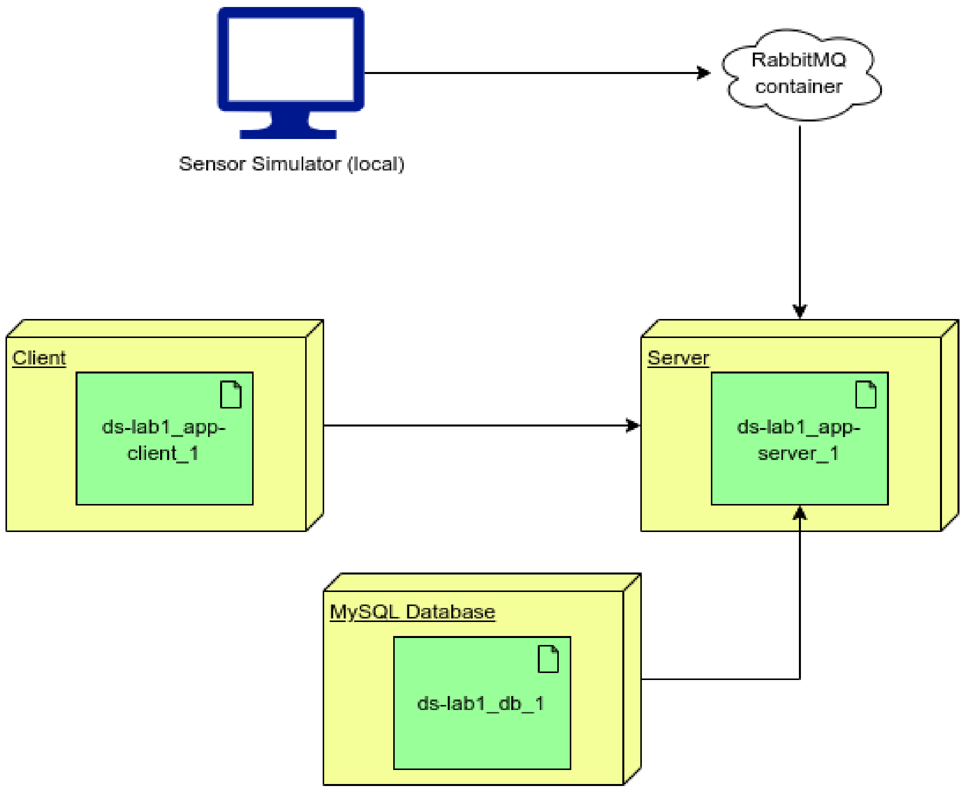
On the client side, a WebSocket library will be used, namely SockJS, together with STOMP (Simple Text Oriented Message Protocol), which allows the client-side implementation of WebSockets to work. The user may initiate a chat with the admin by clicking on the Support button in the Dashboard. Similarly, the Admin, may press on the “Chat” button in order to achieve similar functionality. Any user is able to talk to multiple users at once, due to the way in which the message topics have been constructed.



```
public class ChatMessage {
    no usages
    User sender;
    no usages
    User receiver;
    no usages
    String message;

}
```

When sending messages, an object of type ChatMessage is created, which is then parsed into a JSON format. Thus, from the message, one can determine the sender and the receiver, as well as the contents of the message itself. This message is then sent to the front-end where it is added into an array which is filtered based on the currently selected user with which the current user is communicating with.


Sensor Simulator (local)
RabbitMQ container
Client
ds-lab1_app-client_1
Server
ds-lab1_app-server_1
MySQL Database
ds-lab1_db_1