

Online Energy Utility Platform

Distributed Systems Laboratory, 2022-23



Student: Keresztes Beáta

Group: 30441

Lab assistant: Cristina Pop

Conceptual Architecture

The responsibilities can be divided between 7 main components:

- **Backend** application, responsible for providing the basic functionality
- **Frontend** application, responsible for providing the user interface
- **Database** server, responsible for persisting the data related to the users, devices and measurements.
- **Desktop application** which sends the collected measurements on a message queue in order to be received by the backend (Spring) application.
- **Message Queue** from **RabbitMQ** acts as the middleware for the indirect communication between the application that collects the sensor data (message producer) and the spring application which reads the measured data (message consumer) and process/stores it.
- **Grpc server** which provides the API to use the chat functionality
- **Envoy proxy** which intercepts the HTTP 1.1 requests from the client (frontend) and converts them to HTTP/2 before forwarding them to the Grpc server. Similarly, the requests from the grpc server in form of HTTP/2 are translated to HTTP 1.1 before transmitting them to the client.

Send device measurements to message queue

The Spring application, after processing the received messages from the queue, it detects if the total energy consumed is higher than a predefined threshold and notifies the client which owns the device using websockets.

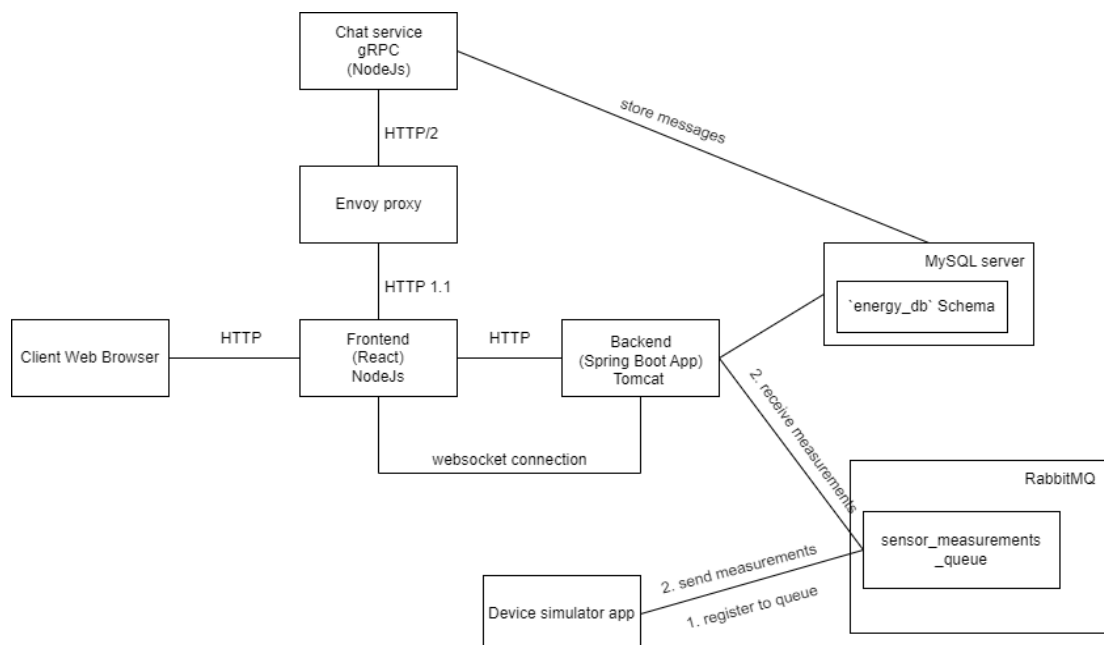
Initially, on start-up the client connects to the server's websocket and subscribes to the notification endpoint. On the server side we register the client's sessionId in order to be able to send private messages to only 1 client, instead of broadcasting it to every connected client. Thus, whenever a message is dispatched notifying the client that a given device consumed a higher energy value, the client which owns the device with the corresponding sessionId, received a notification on the frontend, while other clients will not receive the notification.

Chat functionality using gRPC

The chat functionality is implemented in a NodeJS server using gRPC for generating an API library with the data types and the signatures of the remote methods.

The envoy proxy acts as a middleware and translates the HTTP requests exchanged between the Frontend and Backend application from version 1.1 to version 2 and vice-versa.

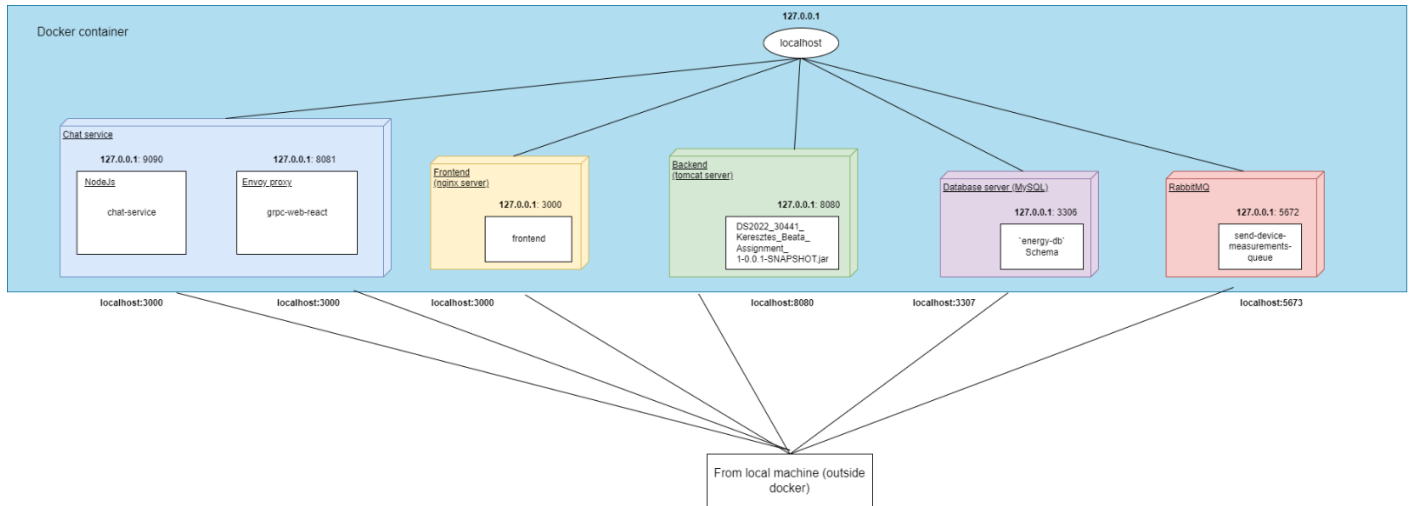
Based on this responsibility division between the components of the system, it is obvious that the preferred architectural model for designing the system should be a layered architecture.



Deployment Diagram

Local Deployment using Docker

A high-level view of the main components (hardware and software) of the system, modeling the physical architecture of the system or how the processing/services are distributed among the nodes:



Deployment on Cloud (Azure) using Docker

The container group with the 4 images for each individual service(backend, frontend, mysql_db, rabbitmq, grpc and envoy) from docker is pushed on the created container registry on Azure: containerregistrybeatakeresztes30441.

Then the application, ds2022_30441_keresztes_beata_assignment_2, is deployed on Azure and it receives a public IP address which exposes the application to be able to access it from a browser from any machine (not only the host's machine).

