Logo, company name

Description automatically generated

Faculty of Automation and Computer Science

Distributed Systems

Laboratory – Assignment 3

Remote Procedure Call (RPC)

Student: Itu Anca

Group: 30441

1. Assignment objective

The objective of this assignment was to extend the energy consumption for the online energy utility platform with a chat system created to offer support to clients in case they want to communicate with the administrators of the application.

1. Conceptual Architecture Diagram

This platform contains 4 main modules: a visualization platform – frontend, a data platform – backend, a smart metering device simulator- a standalone application and a database.

A picture containing timeline

Description automatically generated

Graphical user interface

Description automatically generated

The backend is built using RESTful services having the purpose of sending data from the frontend to the database or vice versa.

In the diagram above there can be seen the 4 layers from the backend application. The first one is the business layer, containing the controllers and services of the backend application. This is the layer which is accessed through the REST APIs from the frontend application, then data is transferred between controllers and services. The controllers receive data from or send data to frontend as DTOs. Data is transferred between the controller and the business layer in the same format. The business logic layer contains a package called and it contains the classes which represent the Message Consumer of the application. Data is received here from the Message Broker in JSON format. The presentation layer refers to the frontend. The business logic layer is intermediary between the presentation layer and the persistence layer, and it is responsible for the data processing. An object is received from the frontend as DTO and sent to the persistence layer as entity. The builders are used for this transformation. When data is retrieved from the database in the persistence layer, it is sent to the business logic layer as an entity and here it is converted to DTO in order to be sent to frontend.

The Message Producer is a standalone application which contains 2 main classes: one is responsible for reading data from a file and the other one is responsible for sending the data, as JSON object to the Message Broker. From here, the data is sent to the Message Consumer – the backend.

The Message Consumer contains another package, called webSockets and it contains the configurations for the web sockets used to send notifications to the users who have a device which exceeded its maximum accepted hourly energy consumption

The new classes added for the chat functionality are contained in the proto.chat package. The content of these classes is generated from the proto file. The logic for sending and receiving messages is implemented in a service class, contained in the grpc.service packake.

Diagram

Description automatically generated

The frontend application represents the user interface, and it gives the user the possibility to perform operations on the data existing in the database. First, the start page is displayed to the user. Then, the user has to log in. According to his type (client or admin), he is redirected to the client page or to the admin page. If the user is a client, he can view the smart energy metering devices he owns or he can view the daily energy consumption of each one of his devices in a visual way. If the user is an admin, then he can manage the users, the devices or the associations between these two. He can create, update, view or delete objects from the database. Each operation for receiving data from the database or for sending data to it is done through REST APIs to the backend.

The webSocketConnection package contains the configurations made for the client to receive notifications from backend. If a message is received, then a popup appears to the user whose device exceeded its threshold.

The chat package contains the chat related classes.

Graphical user interface, diagram

Description automatically generated

There is an outside package which contains the proto file, the java compiler and the web compiler. From this folder, the java classes and the react classes for the chat part of the application are generated based on the structure described in the proto buff file.

1. UML Deployment Diagram

Diagram

Description automatically generated

The deployment diagram contains a node which represents the Web Browser, where the user can access the platform, a node for the Tomcat Server which takes care of sending the requests to the server, which contains the Data platform of the application, a node for the RabbitMQ which is he middleware between the Data platform and the Smart Metering Device Simulator, a standalone application, and a node for the database server. I added a proxy between the Energy Visualization Platform and the Energy Data Platform which has the purpose of converting the HTTP/1 requests sent from the client to HTTP/2, in order to be received correctly by the server.