Distributed Systems

Assignment 3

Raita Anamaria

Group 30441

Table of Contents

[1. Requirements 2](#_Toc124331117)

[2. Conceptual architecture of the distributed system 2](#_Toc124331118)

[3. UML Deployment diagram 4](#_Toc124331119)

[4. Build and execution 4](#_Toc124331120)

[5.Bibliography 5](#_Toc124331121)

# Requirements

Develop a chat system to offer support for the clients of the energy platform if they have questions related with their energy consumption. The chat system should allow communication between the clients and the administrator of the system.

➢ The client application displays a chat box where clients can type messages.

➢ The message is sent asynchronously to the administrator, that receives the message together with the client identifier, being able to start a chat with the client.

➢ Messages can be sent back and forth between the client and the administrator during chat session.

➢ The administrator can chat with multiple clients at once.

➢ A notification is displayed for the user when the other user reads the message.

➢ A notification is displayed for the us

# Conceptual architecture of the distributed system

For implementing this assignment, I user grpcs. In gRPC, a client application can directly call a method on a server application on a different machine as if it were a local object, making it easier for you to create distributed applications and services. As in many RPC systems, gRPC is based around the idea of defining a service, specifying the methods that can be called remotely with their parameters and return types. On the server side, the server implements this interface and runs a gRPC server to handle client calls. On the client side, the client has a stub (referred to as just a client in some languages) that provides the same methods as the server.

Firstly, we have to define a proto contract, in which we have defined the objects what we want to use, defining for each one the parameters and the unique identifiers. Also, we define the name of the remote procedure calls and what they return.

On the server side, I used a node.js server. There you send the messages and then as a reciever you actually make a request to see if there are any messages for you.We override the methods described in proto contract and provide an implementation which allows us to communicate.

On the front-end side, we have the send and recieve methods. I did some checks, in order to display just the message which are intended for the two people in the chat.

Also, to approach the situation of typing and seen, I have implemented that either on click or on change (depending) that it send a special message which detects this action.

Also, when a user opens the chat, it send a special message and its corresponding button on admin screen is changing its color.

Diagram

Description automatically generated

Figure 1 System Architecture

Diagram

Description automatically generated

Figure 2 - System Architecture in Details

# UML Deployment diagram

The application is like the previous one, with a backend, frontend, database, message producer and message consumer integrated in the backend. Besides that, we also have a node server which communicates with the server through a proxy.

In the picture presented below, you can see the deployment diagram of the application.

Timeline

Description automatically generated

Figure 3- Deployment Diagram

# Build and execution

In order to build the source code, you should have installed on your computer the following:

* InteliJ
* Node js (for react and node server )
* Docker

In order to run the application, having the source code, you should follow these steps:

* Open IntelliJ
* Open the producer desktop application and run it
* Open the spring application (the backend) and run it
* Open a cmd prompt where the node server is and write node server
* Open Visual Studio Code
* Open the frontend application
* Write in browser localhost:3000 here you can log in as a client, register a device, write the device of the id in the program’s arguments of the producer and run that configuration
* Also, you chan log in as a user and initiate the chat with an admin, then the admin can also open its chat and you can start talking with each other.

# 5.Bibliography

* https://daily.dev/blog/build-a-chat-app-using-grpc-and-reactjs