**DISTRIBUTED SYSTEMS**

***Assignment 3***

*Remote Procedure Call (RPC)*

*Chat System for Client Support*

Vilcelean Dalia

30244

2022

***Cuprins:***

1. Introducere

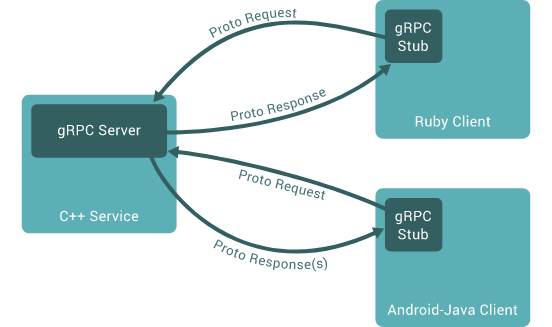
2. Prezentare aspecte generale

3. Functionalitate

4. Concluzii

1. Introducere

In gRPC, o aplicație client poate apela direct o metodă dintr-o aplicatie server pe o altă mașină, ca și cum ar fi un obiect local, facilitând crearea de aplicații și servicii distribuite. Ca și în multe sisteme RPC, gRPC se bazează pe ideea definirii unui serviciu, specificând metodele care pot fi apelate de la distanță cu parametrii și tipurile de returnare ale acestora. Pe partea de server, serverul implementează această interfață și rulează un server gRPC pentru a gestiona apelurile clienților. Pe partea clientului, clientul are un stub (denumit doar un client în unele limbi) care oferă aceleași metode ca și serverul.



Clienții și serverele gRPC pot rula și vorbi între ei într-o varietate de medii - de la servere din Google la propriul desktop - și pot fi scrise în oricare dintre limbile acceptate de gRPC. Deci, de exemplu, puteți crea cu ușurință un server gRPC în Java cu clienți în Go, Python sau Ruby. În plus, cele mai recente API-uri Google vor avea versiuni gRPC ale interfețelor lor, permițându-vă să construiți cu ușurință funcționalitatea Google în aplicațiile dvs.

1. Prezentare Aspecte Generale

**1. Cerințe**

Dezvoltați un sistem de chat pentru a oferi suport clienților platformei energetice dacă au întrebări

legate de consumul lor de energie. Sistemul de chat ar trebui să permită comunicarea între

clienții și administratorul sistemului.

**1.1. Cerințe funcționale:**

➢ Aplicația client afișează o casetă de chat în care clienții pot introduce mesaje.

➢ Mesajul este trimis asincron către administrator, care primește mesajul împreună

cu identificatorul clientului, putând începe un chat cu clientul.

➢ Mesajele pot fi trimise înainte și înapoi între client și administrator în timpul chat-ului

sesiune.

➢ Administratorul poate discuta cu mai mulți clienți simultan.

➢ O notificare este afișată pentru utilizator când celălalt utilizator citește mesajul.

➢ O notificare este afișată pentru utilizator (de exemplu, tastând) în timp ce utilizatorul de la celălalt capăt al

comunicarea tipează mesajul său.

**1.2. Tehnologii de implementare:**

➢ Alegeți una dintre următoarele tehnologii: orice cadru RPC care suportă date streaming (de exemplu: gRPC)

Pentru proiectul nostru:

-utilizatorii vor suna pentru a se alătura chatului de grup

-o metodă care va fi apelată de un utilizator pentru a trimite un mesaj grupului.

-o metodă care va deschide un flux de date pe server, apoi clientul va asculta fluxul de date pentru a le obține. Acest flux de date va fi mesaje trimise de utilizatorii din grup, așa că trebuie să le transmitem în flux la ascultarea tuturor clienților.

-o metodă care va returna toți utilizatorii din grup.

Aceste metode sunt setate folosind cuvântul cheie rpc.

Deci vom avea patru metode. Metoda de alăturare va lua un utilizator utilizator ca parametru, astfel încât să obțineți detaliile utilizatorului de acolo și să adăugați utilizatorul la chatul de grup. Va răspunde cu un tip JoinResponse. SendMsg trimite mesaje la chatul de grup. Ia formatul mesajului ChatMessage ca arg, astfel încât să poată primi mesajul și expeditorul din câmpuri, nu returnează nimic, un obiect gol. receiveMsg setează un flux de chat-uri pe server. Cuvântul cheie stream denotă un flux de date care este creat și transmis către utilizatorul final. Aici, receiveMsg va crea și emite fluxuri de mesaje de chat ChatMessage, clientul va asculta fluxul pentru a primi chat-urile transmise pe canal. Acest lucru face posibilă primirea în timp real a textelor trimise de utilizatori. GetAllUsers returnează utilizatorii în chat.

1. Functionalitate

Pentru acest proiect am avut nevoie de React.js si de o imagine Docker de Envoy, cu care am putut face conexiunea cu srverul.

Graphical user interface, application

Description automatically generated

1. Concluzii

Am aflat despre ce înseamnă gRPC. Am început prin a ne uita la un pic din istoria sa și apoi, cum funcționează.

Am aflat despre streaming gRPC, care ne permite să transmitem și să ascultăm o secvență de date în timp real. Am demonstrat toate acestea prin construirea unei aplicații de chat în React.js, făcând acest lucru, am învățat și despre solicitările proxy de la un client de browser folosind Envoy, am aflat puțin despre configurațiile Envoy și am putut comunica cu un server gRPC de la browserul. Sunt multe lucruri pe care le-am învățat în acest tutorial.

gRPC deschide o lume cu totul nouă de posibilități și, în același timp, valorifică puterea uimitoare a HTTP/2.