**Ministerul Educației și Cercetării**

**Universitatea Tehnică a Moldovei**

**Facultatea Calculatoare, Informatică și Microelectronică**

**Departamentul Ingineria Software și Automatică**

**Raport**

Lucrarea de laborator nr.1

Disciplina: Programarea Aplicațiilor Distribuite

Tema: Agent de mesagerie(Broker)

**Efectuat**: st.gr. TI-202 Bunescu Gabriel

**Verificat**: asist. univ. Cristofor Fistic

Chișinău 2023

**Scopul lucrării:**

Elaborarea Broker-ului pentru un Agent de mesagerie.

**Mersul lucrării:**

1. Socket - crearea si implementarea Broker.
2. Grpc - crearea si implementarea Broker.

# **Socket - crearea si implementarea Broker.**

Socket sunt folosite pentru două lucruri: pentru a transfera date printr-o rețea și pentru comunicarea între aplicații.

Acesta aplicatie reprezintă o implementare simplă a unui broker de mesaje Un broker de mesaje este un intermediar care primește mesaje de la producători și le distribuie către consumatori în funcție de anumite criterii sau reguli.

Modului de funcționare al acestui cod:

Se importă modulele necesare: socket pentru comunicarea pe rețea și asyncio pentru programarea asincronă.

Sunt definite câteva constante:

* CONNECTIONS\_NR stabilește numărul maxim de conexiuni pe care serverul le va asculta simultan (aici, este setat la 1);
* multicastIP specifică adresa IP multicast la care vor fi redirecționate mesajele primite.

Se definește clasa BrokerSocket pentru gestionarea conexiunilor și manipularea mesajelor:

* în constructorul clasei se creează un socket TCP (self.broker\_socket) și se stabilește adresa IP multicast;
* metoda bind leagă socket-ul la o adresă IP și port specificat;
* metoda listen permite socket-ului să asculte pentru conexiuni de la clienți;
* metoda asincronă accept așteaptă conexiuni de la clienți și le gestionează apelând handle\_client pentru fiecare conexiune acceptată;
* metoda asincronă handle\_client primește date de la client, le decodează și apoi le redirecționează cu ajutorul metodei redirect\_data;
* metoda redirect\_data primește un mesaj și un port și redirecționează mesajul către adresa IP multicast specificată, folosind un socket UDP.

Blocul if \_\_name\_\_ == "\_\_main\_\_" este punctul de intrare în program:

* se creează o buclă de evenimente asyncio;
* se creează o instanță a clasei BrokerSocket și se leagă la portul 9000;
* apoi, serverul începe să asculte conexiuni și să ruleze în mod continuu utilizând loop.run\_until\_complete(broker\_socket.accept()) și loop.run\_forever().

Acesta implementare a unui broker simplu care primește mesaje de la clienți TCP și le redirecționează către o adresă IP multicast utilizând UDP.

# **Grpc - crearea si implementarea Broker.**

Acesta este o reprezentare a implementare a unui serviciu de broker de mesaje folosind. Un broker de mesaje este un intermediar care permite producătorilor să trimită mesaje către consumatori, iar acest serviciu permite abonarea la anumite subiecte (topic-uri) și recepționarea mesajelor asociate acestor subiecte. Iată o prezentare a funcționalităților și a modului de funcționare al acestui cod:

Definirea serviciilor gRPC. Codul începe prin definirea a trei servicii gRPC în fișierele .proto. Acestea sunt:

* Publisher este un serviciu care permite producătorilor să publice mesaje pentru anumite subiecte;
* Subscriber este un serviciu care permite consumatorilor să se aboneze la subiecte pentru a primi mesaje;
* Notifier este un serviciu folosit intern de broker pentru a notifica consumatorii cu privire la mesaje noi.
* Clasa Connection reprezintă o conexiune către un client și păstrează informații despre adresa clientului și subiectul la care este abonat;
* Clasa ConnectionStorage gestionează stocarea conexiunilor către clienți. Folosește un mecanism de blocare pentru a asigura accesul concurent la lista de conexiuni. Metodele includ add pentru adăugarea unei conexiuni și remove pentru eliminarea unei conexiuni. De asemenea, furnizează o metodă get\_connections\_by\_topic pentru a obține toate conexiunile asociate cu un anumit subiect;
* Clasa Message reprezintă un mesaj și păstrează informații despre subiectul mesajului și conținutul acestuia;
* Clasa MessageStorage gestionează stocarea mesajelor primite pentru a le trimite ulterior clienților abonați. Utilizează, de asemenea, un mecanism de blocare pentru a asigura accesul concurent la lista de mesaje. Metodele includ add pentru adăugarea unui mesaj, get\_next pentru obținerea următorului mesaj disponibil și is\_empty pentru verificarea dacă lista de mesaje este goală;
* Clasa PublisherService implementează serviciul gRPC Publisher și este responsabilă pentru primirea și stocarea mesajelor publicate de producători. Mesajele sunt adăugate în MessageStorage pentru a fi trimise ulterior către consumatori;
* Clasa SubscriberService implementează serviciul gRPC Subscriber și gestionează abonările clienților la subiecte. Când un client se abonează, o nouă conexiune este adăugată în ConnectionStorage;
* Clasa SenderWorker este un fir de execuție care rulează în fundal și se ocupă de transmiterea mesajelor către consumatori. Verifică periodic MessageStorage pentru a vedea dacă există mesaje de trimis și apoi le trimite la toți clienții abonați la subiectul respectiv. Folosește serviciul Notifier pentru a notifica clienții.

Blocul if \_\_name\_\_ == '\_\_main\_\_'. Aici se inițializează și se configurează toate componentele serviciului:

* se creează instanțe ale claselor ConnectionStorage, MessageStorage, PublisherService, SubscriberService și SenderWorker;
* se configurează un server gRPC cu un executor de fir (ThreadPoolExecutor) pentru a permite manipularea concurentă a cererilor;
* serverul gRPC ascultă pe portul 5001 și este inițializat;
* se pornește un fir de execuție (sender\_thread) pentru a trata trimiterea mesajelor către consumatori în fundal;
* într-un bloc try, serverul și firul de execuție rulează în mod continuu. Serverul poate fi oprit cu Ctrl+C, moment în care serverul gRPC este oprit, iar firul de execuție se așteaptă să se termine.

Acesta reprezentare a unui broker simplu de mesaje care utilizează gRPC pentru a permite publicarea și abonarea la subiecte. Este util în scenarii în care doriți să implementați un sistem de trimitere și recepție de mesaje între diferite componente software, cum ar fi aplicații distribuite.

# **Concluzie:**

Aceste două implementări, una folosind socket-uri și cealaltă folosind gRPC, reprezintă două abordări diferite pentru implementarea unui broker de mesaje. Ambele abordări au scopul de a permite producătorilor să trimită mesaje către consumatori, dar utilizează tehnologii și paradigme diferite pentru a atinge acest obiectiv.

Implementarea cu socket-uri se bazează pe comunicarea directă între client și server folosind TCP pentru primirea mesajelor și UDP pentru redirecționarea acestora către adresa IP multicast. Acestă abordare oferă o implementare simplă și eficientă pentru un broker de mesaje într-un mediu de rețea, dar poate necesita gestionarea manuală a multor aspecte, cum ar fi controlul conexiunilor și gestionarea concurenței.

Pe de altă parte, implementarea cu gRPC folosește un model de comunicare bazat pe RPC (Remote Procedure Call) și oferă un nivel mai înalt de abstractizare pentru gestionarea conexiunilor și transmiterea mesajelor. Este potrivită pentru aplicații distribuite și scalabile, oferind servicii de publicare și abonare într-un mod mai modular și mai ușor de întreținut.

În final se poate de spus că alegerea între aceste două abordări depinde de cerințele specifice ale proiectului. Implementarea cu socket-uri este adecvată pentru scenarii simple și directe, în timp ce implementarea cu gRPC oferă o abordare mai sofisticată și modulară pentru sistemele distribuite și poate fi mai potrivită pentru proiecte mai mari și complexe.