Offline Messenger - proiect

Enea Iustin-Gabriel

B5

1 Introducere

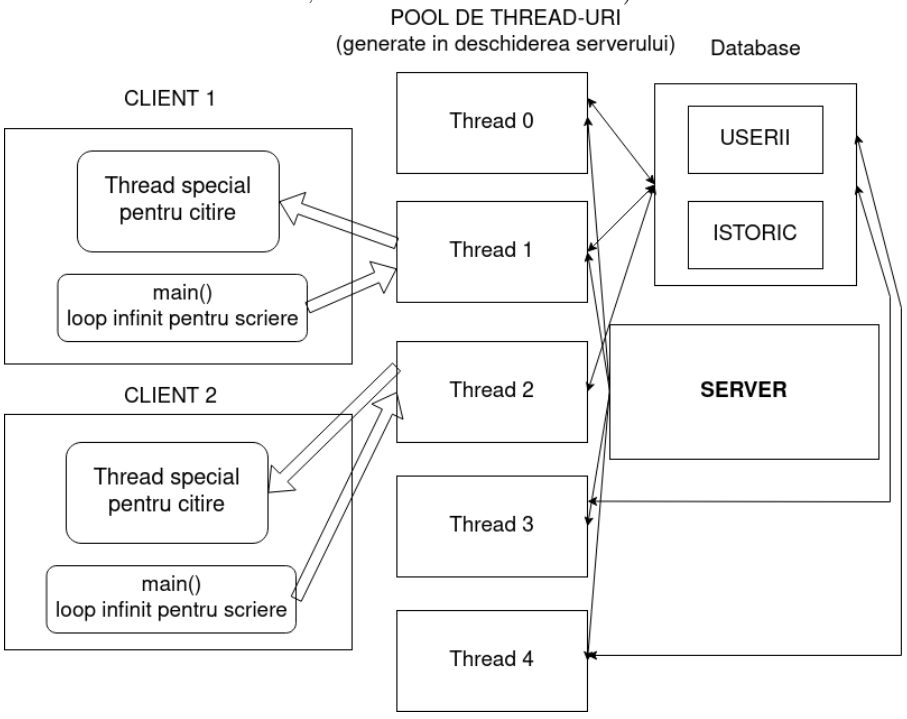
Proiectul Offline Messenger constituie un prototip pentru o aplicatie de chatting, in care utilizatorii se pot conecta si pot trimite mesaje, atat userilor conectati pe aplicatie, cat si celor offline. Totodata, acestia au mai multe facilitati disponibile pe server, precum citirea istoricului conversatiilor si reply-uri cu texte predefinite.

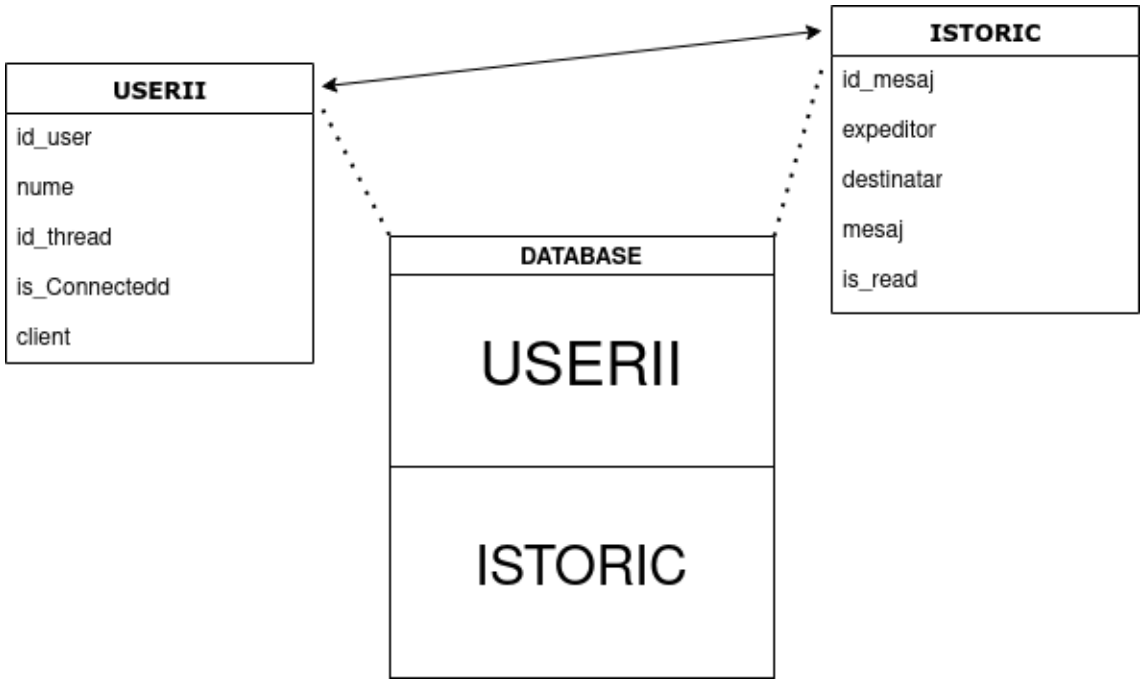
2 Tehnologii

Este folosita tehnologia TCP/IP integral (concurenta), aceasta asigurand o conexiune orientata si sigura, in care datele sunt transferate complet, fiind confirmata validatea tranferului, chiar daca transferul de date va fi mai incet decat in cazul unui sistem UDP/IP. Avand in vedere ca mesajele trebuie sa ajunga complete si in ordine de la un user la altul, sistemul TCP/IP este optim pentru acest proiect, primindu-se totodata confirmarea transferului de date.

3 Arhitectura aplicatiei

Proiectul consta intr-o aplicatie client/server. Pentru fiecare client conectat la server, cream un thread special pentru citire, care va face citire pentru un timp infinit de la server. Clientul se va putea inchide cu comanda quit. Pe partea serverului, vom folosi un thread pool, astfel incat in momentul conectarii unui client, un thread din server este asignat acestuia. Cand clientul se va deconecta, thread-ul va deveni liber si va putea fi folosit de catre alt client ce se va conecta.
Vom utiliza o baza de date cu 2 tabele, prin intermediul SQLite3, care este o mica biblioteca C ce implementeaza un motor de baze de date SQL incapsulat. Cele doua tabele vor fi USERII si ISTORIC. In USERII vom avea numarul userilor, numele userilor, thread-ul asociat acestora, daca sunt conectati sau nu si numarul descriptorului clientului. Tabelul ISTORIC va contine numarul mesajului, mesajul in sine, expeditorul, destinatarul, cat si o casuta in care se va preciza daca mesajul a fost citit sau nu. (daca destinatarul este offline, atunci aceasta va fi bifata)





4 Detalii de implementare

Aplicatia va folosi un protocol, ce va avea urmatoarele comenzi:
login : username - care va loga userul la server propriu-zis, oferindu-i accesul la toate facilitatile acestuia
send : destinatar: mesaj - userul conectat la server va putea trimite oricarui alt user un mesaj de tip text, creandu-se in tabelul istoric o noua inregistrare. Daca destinatarul este activ, vom primi aceasta confirmare, in caz contrar, destinatarul va primi mesajul cand se va loga prima data
history : username - userul conectat va primi intreaga conversatie pe care a avut-o cu userul numit username
logout - va deconecta user-ul de la serverul propriu zis, ne mai avand aceesul la restul facilitatilor, doar la comanda quit
delete me - va sterge user-ul din tabelul USERII, delogandu-l simultan
quit - va deconecta clientul de la server, delogand totodata user-ul conectat(daca este cazul)
Comenzile send, history si delete me pot fi utilizate doar de catre un utilizator este logat. Daca dorim sa ne logam pe un client unde deja este un utilizator logat, va trebui sa ne delogam mai intai.

5 Concluzii

Transferul de mesaje la nivelul aplicatiei nu este securizat, accestea nefiind criptate in niciun fel. Aplicatia ar putea avea mai multe functionalitati, printre care transferul de imagini si alte fisiere, ce vor fi depozitate intr-un fisier unic creat pt fiecare client. Totodata, am putea crea posiblitate de a trimite cereri de prieteni, iar persoanele care sunt prietene sa poata initia chatroom-uri in care toti membrii conectati la room sa poata trimite si primi mesaje. Un alt lucru relevant ar fi dezvoltarea unei interfate grafice, care ar oferi clientilor un mediu mai ”prietenos” de comunicare.

6 Bibliografie

https://profs.info.uaic.ro/ computernetworks/
https://www.tutorialspoint.com/sqlite/index.html
https://www.geeksforgeeks.org/handling-multiple-clients-on-server-with-multithreading-using-socket-programming-in-c-cpp/