Documentatie – Offline Messenger (B)

Luchian Vlad-Ilie

1 Introducere

Proiectul Offline Messenger (B) este o aplicatie client-server pentru facilitarea transmiterii de mesaje intre clienti. Functionalitatile aplicatiei sunt: inregistrarea/logarea utilizatorilor urmate de comenzile la care au acces acestia. Acestia pot trimite mesaje unui alt utilizator, pot raspunde la un mesaj specific, pot vizualiza istoricul conversatiilor cu fiecare utilizator in parte.

2 Tehnologii utilizate

2.1 Modelul arhitecturii de retea

Pentru realizarea conexiunii client/server va fi folosit modelul TCP concurent, in detrimentul modelului UDP. Alegerea a fost facuta in acest sens pentru a asigura transmiterea datelor in ordine si pentru a avea acces la mecanismele de control al fluxului si de control al sugestiei. De asemenea, TCP urmareste datele si se asigura ca nu sunt corupte sau pierdute in timpul tranzitului, verificand pachetele pentru erori.

Pentru a asigura posibilitatea servirii mai multor clienti simultan, la fiecare client nou conectat se creaza un thread nou pentru servirea sa.

2.2 Stocarea datelor

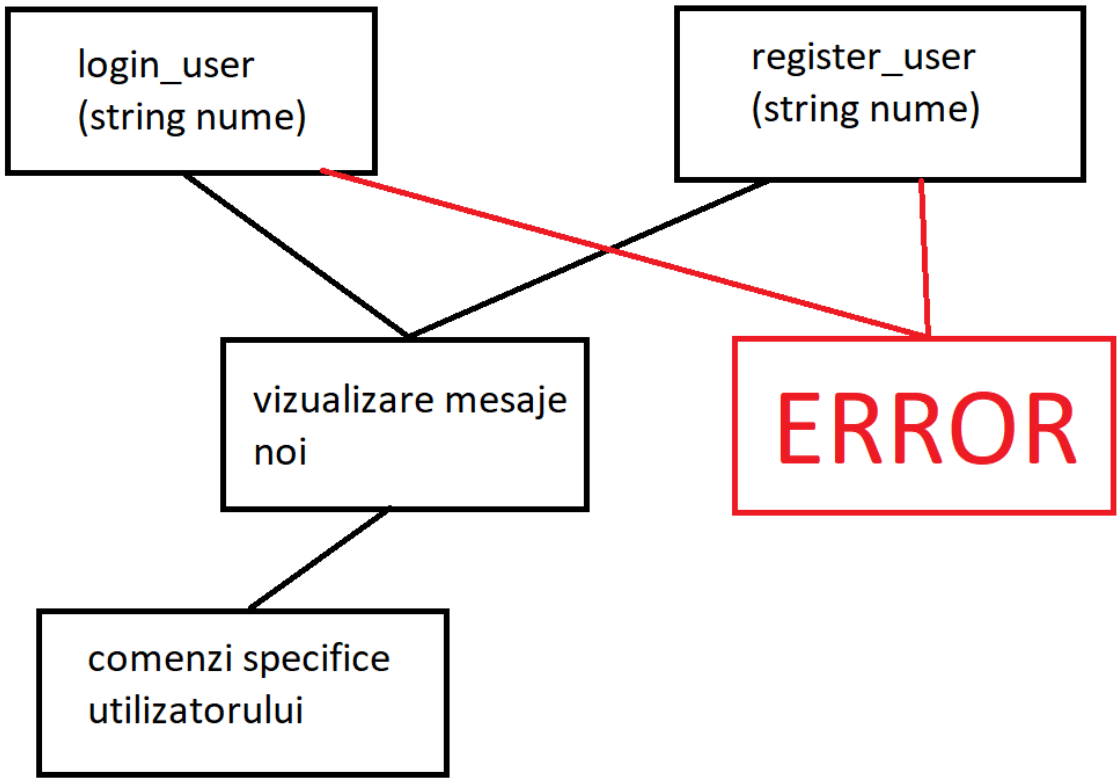
Toate datele vor fi stocate intr- o baza de date, pentru a fi salvate si in cazul unei eventuale inchideri a serverului. Se va folosi o zona de memorie mai mare, dar informatiile nu se vor pierde nici in cazul comutarii offline.

3 Arhitectura aplicatiei

Programul se va folosi de o baza de date sqlite3, soft instalat in prealabil pe versiunea de linux corespunzatoare. Aceasta contine 2 tabele: tabela utilizatorilor si cea

a a mesajelor. Diagramele urmatoare arata cum se va comporta programul din momentul pornirii serverului.

- **Modelul unui client TCP:**
  - Creare *socket* pentru conectarea la server: socket()
  - Pregatirea structurilor de date (sockaddr_in)
  - Atasarea socket-ului: bind() – optional
  - Conectarea la server (deschidere activa): connect()
  - Solicitarea de servicii si receptionarea rezultatelor trimise de server: succesiune de write()/read()
  - Inchiderea (directionata) a conexiunii cu serverul: close(), shutdown()



4 Detalii de implementare

Programul va functiona conform ultimei diagrame, adica orice client va trebui sa se logheze sau inregistreze inainte de a avea acces la functionalitatile programului (in afara comenzilor quit si help).

Logarea userilor si administratorilor va returna o eroare daca userul respectiv nu exista sau parola nu e corecta, urmand ca acestia sa mai incerce sau sa paraseasca programul. Inregistrarea userilor va returna eroare in cazul in care username-ul dorit este deja asociat unui cont.

Comenzile la care au acces userii includ: schimbarea numelui, vizualizarea conversatiei cu un anumit utilizator, trimiterea unui mesaj catre un utilizator, vizualizarea mesajelor noi, raspunsul la un mesaj specific.

```
else if (strncmp(fromClient, "register ", 9) == 0)
{
  int maxid = 0;
  char id[100];
  id_mesager == 0;
  strcpy(argument, fromClient + 9);
  sprintf(query, "SELECT id FROM users where name='%s';", argument);
  memset(rezultat, 0, sizeof(rezultat));
  tdL.rc = sqlite3_exec(tdL.db, query, callback, rezultat, &zErrMsg);
  id_mesager = atoi(rezultat);
  if (id_mesager == 0)
  {
    tdL.rc = sqlite3_exec(tdL.db, "select max(id) from users;", callback, id, &zErrMsg);
    maxid = atoi(id) + 1;
    sprintf(query, "INSERT INTO users values(%d,'%s');", maxid, argument);
    memset(rezultat, 0, sizeof(rezultat));
    tdL.rc = sqlite3_exec(tdL.db, query, callback, rezultat, &zErrMsg);
    sprintf(toClient, "Utilizator %s creat cu succes!", argument);
  }
  else
    sprintf(toClient, "Userul %s exista deja!", argument);
}
```

Fig. 1 Inregistrarea unui utilizator nou

De asemenea, clientul va trimite automat, periodic request-uri catre server, prin care se verifica daca utilizatorul a primit un mesaj nou intre timp. Aceasta functie a programului este si cea care afiseaza unui utilizator nou conectat mesajele primite cat timp era offline.

```
else if (strncmp(fromClient, "checkmessages", 13) == 0)
{
  if (id_mesager != 0)
  {
    int check = 0;
    sprintf(query, "select count(*) from messages where id_destinatar=%d and isSeen=0;", id_mesager);
    bzero(rezultat, sizeof(rezultat));
    sqlite3_exec(tdL.db, query, callback, rezultat, &zErrMsg);
    check = atoi(rezultat);
    if (check == 0)
    {
      sprintf(toClient, "nimic");
    }
    else
    {
      sprintf(query, "select id_mesaj,name,text_mesaj,isReplyTo from messages m join users u on m.id_expeditor=u.id where m.isSeen=0 and m.id_destinatar=%d;", id_mesager);
      bzero(rezultat, sizeof(rezultat));
      sqlite3_exec(tdL.db, query, callback_afisare, rezultat, &zErrMsg);
      printf("rezultatul daca ajunge pana aici: %s", rezultat);
      sprintf(toClient, "Mesaje noi:\n %s \n[client]Introduceti comanda: ", rezultat);
      sprintf(query, "UPDATE messages set isSeen=1 where id_destinatar=%d;", id_mesager);
      sqlite3_exec(tdL.db, query, callback_afisare, rezultat, &zErrMsg);
    }
  }
  else
  {
    sprintf(toClient, "nimic");
  }
}
```

Fig. 2 Partea care trateaza requesturile periodice ale clientilor

5 Concluzii

Solutia propusa ar putea fi imbunatatita prin crearea unui sistem de login/register cu parola transmisa catre server intr-un mod criptat.

6 Bibliografie

Site-uri folosite:

https://profs.info.uaic.ro/computernetworks/

https://linux.die.net/man/

https://www.sqlite.org/cintro.html

https://www.youtube.com/