Offline Messenger

Bălteanu Sara

Facultatea de Informatică, Universitatea Alexandru Ioan Cuza, Iași
sara.balteanu.03@gmail.com

Abstract. Tema constă în realizarea unei aplicații de tip server-client, ce poate fi folosită simultan de mai mulți utilizatori, care schimbă diverse mesaje între ei.

Keywords: client-server, aplicație de comunicare, threads, TCP

1 Introducere

Acest proiect prezintă dezvoltarea unei aplicații de tip server-client care permite clientului comun posibilitatea de înregistrare, conectare și de a putea comunica în mod privat cu 2 sau mai multe persoane. Aplicația oferă funcționalitatea trimiterii mesajelor și către utilizatorii offline, acestora din urmă apărându-le mesajele atunci când se vor conecta la server. De asemenea, utilizatorii vor avea atât posibilitatea de a trimite un răspuns în mod specific la anumite mesaje primite, cât și vizualizarea istoricului conversațiilor cu fiecare utilizator în parte.

2 Tehnologii aplicate

2.1 TCP

Protocolul de control al transmisiei (TCP) este un protocol de comunicare în rețea conceput pentru a trimite pachete de date pe internet. TCP este un protocol ce este utilizat pentru a crea o conexiune între computerele la distanță, transportând și asigurând livrarea de mesaje prin rețelele de asistență și pe Internet.

Deși protocolul UDP ar fi realizat o rulare mai eficientă din punct de vedere al timpului, am ales protocolul TCP deoarece ne asigură că este realizat cu succes transportul tuturor datelor în cadrul aplicației; acuratețea informațiilor fiind necesară, avem garanția că nu vor apărea diverse erori la nivelul schimuburilor de mesaje dintre utilizatori.

2.2 Threads

Un thread este utilizat pentru a oferi o eficiență mai înaltă la nivelul programelor. În comparație cu procesele, crearea și distrugerea unui thread consumă mai puțin

timp, acestea afișează spațiul de adrese al procesului de care aparțin, și în general folosirea thread-urilor ne expune și la alte numeroase beneficii ale acestora.

La nivelul dezvoltării proiectului, am ales să utilizez thread-uri deoarece varianta ideală a aplicației necesită un server concurent, satisfăcând cererile clienților într-o ordine și într-un timp nedeterminat.

2.3 SQLite3

Pentru realizarea stocării datelor referitoare la useri, conversații și mesaje, serverul utilizează o bază de date relațională prin intermediul bibliotecii SQLite3. Există numeroase beneficii la nivelul acestei biblioteci, printre care se numără:

- API-ul SQLite3 este simplu și ușor de înțeles. Operațiile obișnuite, cum ar fi inserarea, actualizarea și interogarea, sunt realizate printr-un set simplu de funcții ușor de accesat
- biblioteca are o performanță bună pentru scenariile de utilizare comune, mai ales pentru aplicații care implică un volum moderat de date și accesuri concurente
- baza de date este stocată într-un singur fișier, ceea ce face instalarea și administrarea mai ușoară
- biblioteca SQLite3 este scrisă în limbaj C și este disponibilă pe o varietate de platforme, făcând-o ușor portabilă

3 Arhitectura aplicației

Concepte utilizate : În server, pentru fiecare client conectat se realizează un thread, făcându-l mutli-threaded. Thread-ul de la bază este blocat folosind accept(), iar atunci când apare un utilizator se crează un alt thread specific acestuia în care vor fi satisfăcute comenzile date. Am implementat diverse funcții de citire și scriere, pentru ca acestea să ne ofere garanția că, odată ce se deconectează clientul de la server, thread-ul se închide imediat, pentru a preveni situația în care vreun thread ramane deschis.

De asemenea, serverul se ocupă cu accesarea bazei de date, astfel incat pentru fiecare comanda validă primită de la client se apelează functiile specifice ale acesteia, în care au loc diverse interogari SQL ce manipulează datele din baza de date, fie selectând, creând, actualizând sau inseârnd diverse informații necesare.

Comenzile funcționale ale aplicației sunt:

- înregistrare, ce permite clientului să își facă un cont în aplicație
- logare, prin care utilizatorul poate intra în contul deja realizat
- vizualizare conversații, care prezintă o listă de convorbiri cu diverși utilizatori
- trimitere mesaj, unde clientul poate trimite un mesaj adițional unei discuții sau să creeze o discuție nouă
- istoric mesaje, în care se selecteaza o conversație și se vizualizază mesaje anterioare
- reply, ce permite să răspundă, în conversația deschisă, la un mesaj specific
- delogare, prin care se iese din cont
- quit, unde clientul părăsește cu totul aplicația

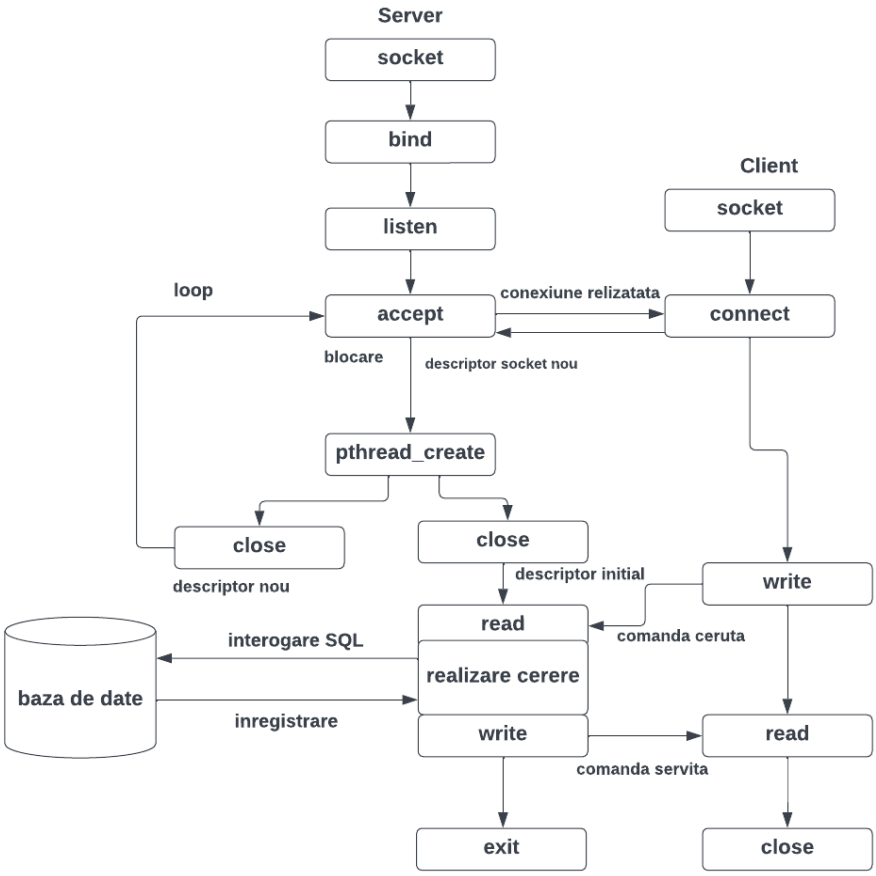

Fig. 1. diagrama specifică aplicației

4 Aspecte de Implementare

4.1 Realizarea bazei de date

Baza de date este relațională, creată cu ajutorul librariei sqlite3. Am folosit cunoștințele dobândite la cursul de baze de date, unde am învățat conceptele de dependențe funcționale, multivaluate, și normalizări etc. Schema bazei de date specifică aplicației este următoarea:

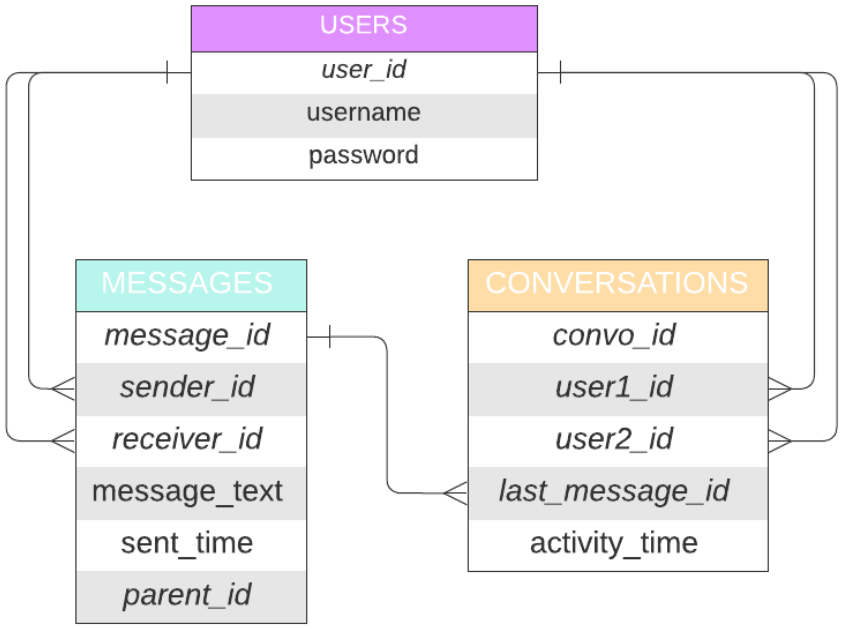

Fig. 2. schema specifică database ului

4.2 Cod relevant

La nivelul codului, clientul prezintă partea abordabilă a aplicației având în vedere că are contact direct cu clienții, introducând diverse comnezi pe care clientul le memorează și le trimite serverului, mai apoi primind un răspuns din partea serverului și pe care, în final, îl livrează înapoi utilizatorului.

În schimb, serverul primește numeroase inputuri din partea clientului și oferă întotdeauna un output corespunzător, ridicându-se la răspunsul așteptat de către utilizator. Procesul realizării serverului fiind unul mai lung, se vor prezenta și explica anumite secvențe de cod:

– crearea socketului, unde se pregatesc si structurile de date necesare, pe care sa le completam in sevrer si, la final, atasam socketul folosind functia bind()

```
if ((sd = socket (AF_INET, SOCK_STREAM, 0)) == -1)
{
    perror ("[server]Eroare la socket().\n");
    return errno;
}

setsockopt(sd,SOL_SOCKET,SO_REUSEADDR,&on,sizeof(on));

bzero (&server, sizeof (server));
bzero (&from, sizeof (from));

server.sin_family = AF_INET;
server.sin_addr.s_addr = htonl (INADDR_ANY);
server.sin_port = htons (PORT);

if (bind (sd, (struct sockaddr *) &server, sizeof (struct sockaddr)) == -1)
{
    perror ("[server]Eroare la bind().\n");
      eturn errno;
}
```

– functia de accept, care ramane in stare blocanta pana la realizarea conexiunii cu un client. Dupa ce conexiunea se stabileste, se creaza un thread pentru acesta.

```
 if ( (client = accept (sd, (struct sockaddr *) &from, &length)) < 0)
{
    perror ("[server]Eroare la accept().\n");
    continue;
}

td=(struct thData*)malloc(sizeof(struct thData));
td->idThread=i++;
td->cl=client;

pthread_create(&th[i], NULL, &treat, td);
```

– functia de deschidere a fisierului asociat bazei de date folosite

```
sqlite3* open_database(const char* dbname)
{
    sqlite3 *db;
    char *zErrMsg = 0;
    int rc;
    char *sql;
```

```
    rc = sqlite3_open(dbname, &db);

   if( rc )
   {
      fprintf(stderr, "Can't open database: %s\n", sqlite3_errmsg(db));
      return(0);
   } else {
      fprintf(stdout, "Opened database successfully\n");
      return db;
   }
}
```

– seventa specifica comenzii "conversatii", in care se realizeaza functia socketului, write(), functia de deschidere si inchidere a bazei de date, si functia showconversations in care se prelucreaza date prin interogari SQL

```
if(strcmp("conversatii", msg)==0)
{
    if(in==0)
    {
        int nr=-1;
        printf("[Thread %d]Trimitem mesajul inapoi...%d\n",tdL.idThread, nr);

        if (write (tdL.cl, &nr, sizeof(int)) <= 0)
        {
            printf("[Thread %d] ",tdL.idThread);
            perror ("[Thread]Eroare la write() catre client.\n");
        }
    }

    if(in==1)
    {
        load=0;
        memset(a_user, 0, sizeof a_user);

        char conversations[256][256];
        int count;
        sqlite3* db = open_database("my_database.db");
        show_conversations(db, current_user,conversations, &count);
        close_database(db);

        int i;
        for(i=0; i<count ; i++)
        {
            printf("%s\n", conversations[i]);
        }
```

```
            in_convo=0;

            printf("[Thread %d]Trimitem mesajul inapoi...%d\n",tdL.idThread, count);

            if (write (tdL.cl, &count, sizeof(int)) <= 0)
            {
                printf("[Thread %d] ",tdL.idThread);
                perror ("[Thread]Eroare la write() catre client.\n");
            }

            ssize_t bytes_written = write_wrapper(tdL.cl, conversations, sizeof(conversation

            if (bytes_written > 0) {
                printf("Wrote %zd bytes to the socket.\n", bytes_written);
            }
            else { printf("Error at bytes_written.\n");}
        }
    }
```

4.3 Utilizari reale

De-a lungul tuturor generațiilor noastre am considerat comunicarea ca fiind o necesitate a omului comun, dezvoltând constant fel și fel de metode revoluționare. În prezent, tehnologia ne este cunoscută în formele ei cele mai avansate, prezentându-ne diverse soluții de comunicare cât mai accesibile, cât mai ușor de înțeles și de folosit.

5 Concluzii

Proiectul Offline Messenger reprezintă atât aplicația ce permite utilizatorilor să aibă acces la diverse funcționalități ce le permite să comunice cu alți utilizatori, fie ei conectați sau deconectați, cât și o modalitate de a ne face o idee cât mai clară a ceea ce știm depre programele client-server, thread-uri, server concurent și lucru cu baze de date.

Idei de îmbunătățiri O serie de idei ce ar contribui la o realizare ideală a acestui proiect:

1. Adaptarea la o comandă ce permite adăugarea username-urilor într-o listă de prieteni, astfel orice utilizator ar putea trimite și primi mesaje numai de la useri ce se află în lista sa de prieteni
2. Adăugarea unei comenzi ce realizează un group chat, în care pot discuta mai mult de 2 useri în mod simultan

3. Actualizarea funcției reply astfel încât să se creeze o structură ierarhică în conversații, formatul conversațiilor fiind mai clar și mai ușor de citit de către utilizator.
4. Posibilitatea utilizatorilor de a își trimite fotografii, fișiere, nu doar mesaje text.

References

1. https://www.overleaf.com/learn/latex/Commands
2. https://profs.info.uaic.ro/ computernetworks/cursullaboratorul.php
3. https://www.lucidchart.com/pages/
4. LNCS Homepage, http://www.springer.com/lncs.
5. https://www.sqlite.org/index.html
6. SQLite Tutorial, https://www.tutorialspoint.com/sqlite/index.htm