Offline Messenger

Ciobanu Ana, Grupa B1

1 Introducere

Proiectul Offline Messenger presupune o aplicație de tip client/server care permite schimbul de mesaje între utilizatorii care sunt conectați și oferă funcționalitatea trimiterii mesajelor și către utilizatorii deconectați. Aplicația oferă de asemenea posibilitatea accesării istoricului conversațiilor și trimiterea unor răspunsuri în mod specific la anumite mesaje primite.

Alegerea proiectului a fost influențată de mai multe criterii. În primul rând este o aplicație utilă, deoarece la momentul actual comunicarea dintre persoane presupune în mare parte un schimb de mesaje scurte, scrise în grabă, fără a acorda importanță corectitudinii acestora. De asemenea am găsit interesantă ideea trimiterii unor răspunsuri în mod specific la anumite mesaje primite, de exemplu la un mesaj "Bună dimineața" este firesc să răspundem cu același text. În al doilea rând posibilitatea utilizării unei baze de date, prin urmare asimilarea unor noi cunoștințe și experienței de lucru cu biblioteci C care implementează un motor de baze de date SQL încapsulat. Toate acestea sumând sub exersarea implimentării unui model TCP concurențial.

2 Tehnologiile utilizate

După cum am precizat în secțiunea anterioară la baza aplicație va fi modelul TCP concurențial, iar pentru administrarea datelor, informațiilor din cadrul aplicației am optat pentru baze de date, în defavoarea fișierelor text.

2.1 TCP

Dat fiind specificul aplicație este necesar ca informația, în cazul dat mesajele să fie transmise corect și în aceeași ordine, în acest sens cea mai bună variantă este modelul TCP. Acesta este un protocol de transport sigur, orientat pe conexiune care asigură transmiterea fără erori a unui flux de octeți de la utilizator la server și invers. Termenul orientat pe conexiune presupune că între cele două aplicații care comunică utilizând TCP trebuie să se stabilească o conexiune înainte ca transferul de date să aibă loc. TCP asigură transferul sigur deoarece implimentează conceptele de confirmare, expirare și retransmitere. Datele sunt împărțite în fragmente, ce poartă numele de segmente. Odată cu trimiterea unui

segment, TCP pornește un timer și, dacă nu se primește confirmarea segmentului respectiv într-un anumit timp, îl retransmite. În momentul primirii unui segment, TCP trimite o confirmare. Pentru a garanta că datele au fost primite în ordinea trimiterii, la destinație, TCP folosește numere de secvență pentru a reordona segmentele înainte de a le livra către nivelul aplicației. Spre deosebire de UDP care nu garantează integritatea și ordinea datelor, TCP asigură că fluxul de date trimis de sursă va fi livrat fără modificări la destinație.

2.2 SQL

SQLite este o bibliotecă C care implementează un motor de baze de date SQL autonom, rapid, fiabil, cu zero-configurare, public. Printre caracteristicile principale se numără: stocarea unei baze întregi de date într-un singur fișier, tranzacțiile sunt atomice, consistente, izolate și durabile chiar și în cazul unor căderi de sistem, poate fi utilizat pe aproape orice sistem de operare. Alături de cele enumerate mai sus, am ales SQLite3 deoarece este sigur și ușor de utilizat, oferă avantajul de a stoca datele într-un mod organizat, accesibil.

3 Arhitectura aplicației

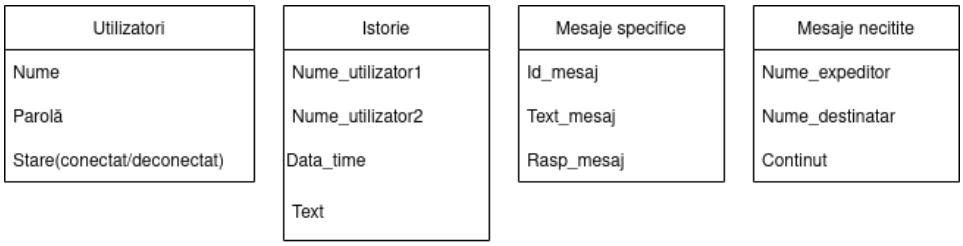

Diagrama1

În Diagrama1 am reprezentat modul de organizare a datelor în baza de date utilizată în cadrul aplicației.

Cea de a doua diagramă prezintă o sintetizare a modului de funcționare a aplicației la baza căreia se află modelul Client/Server concurențial TCP forked().

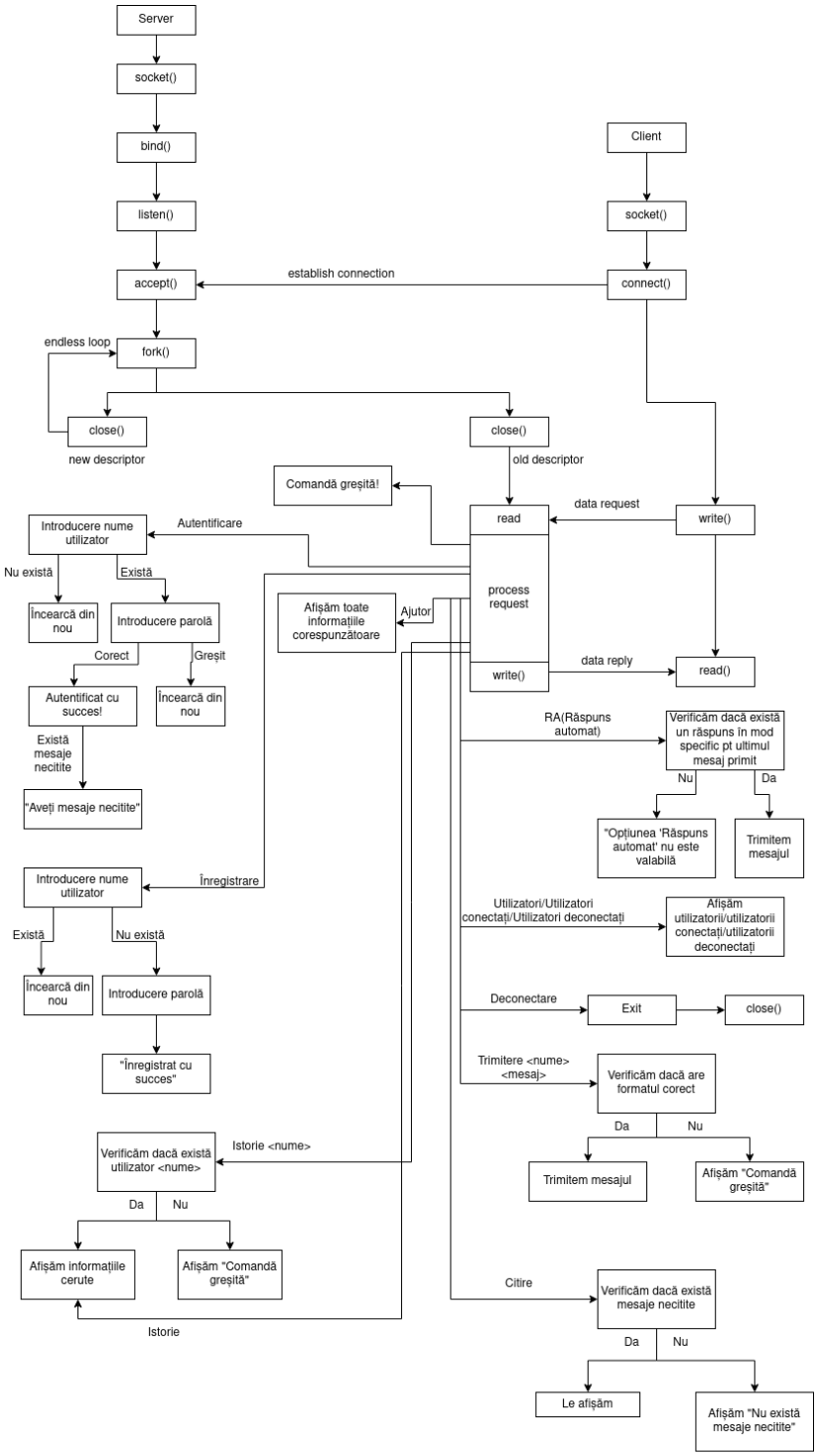
Server
socket()
bind()
listen()
accept()
establish connection
Client
socket()
connect()
endless loop
fork()
close()
new descriptor
close()
old descriptor
Comandă greșită!
read
data request
write()
Introducere nume utilizator
Autentificare
Nu există
Există
Încearcă din nou
Introducere parolă
Corect
Greșit
Autentificat cu succes!
Încearcă din nou
Există mesaje necitite
"Aveți mesaje necitite"
Afișăm toate informațiile corespunzătoare
Ajutor
process request
write()
data reply
read()
RA(Răspuns automat)
Verificăm dacă există un răspuns în mod specific pt ultimul mesaj primit
Nu
Da
"Opțiunea 'Răspuns automat' nu este valabilă
Trimitem mesajul
Introducere nume utilizator
Înregistrare
Există
Nu există
Încearcă din nou
Introducere parolă
"Înregistrat cu succes"
Utilizatori/Utilizatori conectați/Utilizatori deconectați
Afișăm utilizatorii/utilizatorii conectați/utilizatorii deconectați
Deconectare
Exit
close()
Trimitere <nume> <mesaj>
Verificăm dacă are formatul corect
Da
Nu
Trimitem mesajul
Afișăm "Comandă greșită"
Verificăm dacă există utilizator <nume>
Istorie <nume>
Da
Nu
Afișăm informațiile cerute
Afișăm "Comandă greșită"
Istorie
Citire
Verificăm dacă există mesaje necitite
Da
Nu
Le afișăm
Afișăm "Nu există mesaje necitite"

Diagrama2

Principiul esențial al acestui model constă în existența a câte unui proces dedicat tratării fiecarui client. Pentru a stabili legătura dintre server și clienți folosim socket, respectiv printre funcțiile utilizate în acest sens sunt: socket(), bind(), accept(), connect(). Serverul TCP deschide socketul, atașeaza un port pentru furnizarea serviciului și așteaptă conectarea clienților, bucla de tratare a clienților începe cu acceptarea conexiunilor. După acceptarea unei conexiuni și obținerea socketului asociat respectivului client (returnat de accept()), serverul creează un proces copil prin fork() dedicat clientului/conexiunii respective. În cadrul procesului copil serverul realizează transferul de date către și dinspre client, realizând de asemenea operații asupra bazei de date, după care va finaliza conexiunea. Pentru verificarea existenței unor date, extragerea acestora și modificarea folosim funcții de lucru cu baze de date precum: sqlite3_open(), sqlite3_step, sqlite3_finalize, sqlite3_prepare_v2 etc. Pentru tratarea unui nou client se reia bulca de acceptare. Clientul este responsabil pentru preluarea comenzilor introduse de utilizator în linia de comandă. Avantajul, pe lângă tratarea simultană a mai mulți clienți constă și într-o așteptare minimă a clientului în coada listen până la acceptarea conexiunii. Practic serverul reia bucla de acceptare imediat, copiii ocupându-se în paralel de tratarea cererilor clienților. Pentru a facilita activitatea utilizatorilor în cadrul aplicației, aceasta sugerează printr-un mesaj la început apelarea la comanda Ajutor, pentru a afla mai multe informații referitoare la aplicație și comenzile disponibile.

4 Detalii de implementare

Pentru fiecare client conectat la server, în primul rând apare un text "Aplicația "Offline Messenger", pentru mai multe detalii tastați Ajutor". Comenzile Istorie, Trimitere, RA(răspuns automat) etc pot fi apelate doar dacă un utilizator este autentificat. Pentru aceasta se scrie comanda "Autentificare" la linia de comandă, ca rezultat se cere introducerea numelui de utilizator. Serverul verifică dacă există un utilizator cu acest nume, în caz afirmativ se cere parola, în caz contrar se sugerează printr-un text afișat la linia de comandă să se mai încerce o dată. De asemenea este valabilă comanda de înregistrare, urmată ulterior de autentificare. Verificarea existenței unui utilizator cu numele și parola specificată se realizează prin interogări la tabelele din baza de date corespunzătoare proiectului. Pentru fiecare comandă introdusă la linia de comandă este important să se păstreze sintaxa prezentată în Ajutor. De exemplu istoria conversațiilor dintre utilizatorul curent și toți ceilalți utilizatori poate fi vizualizată prin comanda "Istorie", dacă se dorește doar cu un singur utilizator se folosește "Istorie <nume>", unde <nume> conține numele de utilizator ce ne interesează. Deosebit față de majoritatea aplicațiilor de comunicare este opțiunea de a oferi un răspuns automat pentru o categorie de mesaje. Comanda RA(răspuns automat) se poate aplica pentru ultimul mesaj primit. Se verifică dacă acesta face parte din mesaje speciale, dacă da, se trimite un răspuns, în caz contrar se afișează

"Opțiunea RA nu este valabilă pentru acest mesaj".

În continuare este prezentată o secvență de cod, ce cuprinde o funcție cu rolul de a verifica dacă există un utilizator cu numele dat sau nu, apelând la funcții specifice SQLite3. Această funcție este utilă atât pentru comanda Autentificare, cât și pentru Înregistrare, deoarece nu pot exista doi utilizatori cu același nume de utilizator.

```
1  int vf_nume_utilizator(char* nume)
2  {
3
4  sqlite3 *db;
5  int openBD;
6  openBD=sqlite3_open("OfflineMessenger.db", &db);
7  int st;
8  int gasit=0;
9  sqlite3_stmt *stmt;
10 char *query = NULL;
11 asprintf(&query, "SELECT Nume FROM Utilizatori WHERE
12 Nume='%s';",nume);
13 sqlite3_prepare_v2(db, query,-1, &stmt, NULL);
14 while ( (st = sqlite3_step(stmt)) == SQLITE_ROW)
15 { gasit++;}
16 sqlite3_finalize(stmt);
17 free(query);
18 if(gasit==1)
19 return 1;
20 return 0;
21 }
```

De fiecare dată când se trimite un mesaj de la un utilizator la altul, textul mesajului, numele utilizatorilor precum și data și ora trimiterii se adaugă în baza de date, în cadrul tabelei Istorie. Cele expuse sunt executate în cadrul funcției adaugare_Istorie.

```
1  int adaugare_Istorie(char *Nume1,char *Nume2,char * text)
2  {sqlite3 *db;
3  char *query=NULL;
4  char *data_time;
5  time_t t;
6  time(&t);
7  data_time=ctime(&t);
8  int rc;
9  sqlite3_stmt *stmt;
10 int openBD;
```

```
11 openBD=sqlite3_open("Offline_Messenger.db",&db);
12 asprintf(&query,"INSERT INTO Istorie
13 (Nume_utilizator1,Nume_utilizator2,
14 Data_time,Text) Values ('%s','%s','%s','%s');",
15 Nume1,Nume2,data_time,text);
16 sqlite3_prepare_v2(db,query,strlen(query),&stmt,NULL);
17 rc=sqlite3_step(stmt);
18 if(rc!=SQLITE_DONE)
19 {
20 printf("Eroare la inserarea datelor in baza de date:
21 %\n",sqlite3_errmsg(db));
22 return 0;}
23 sqlite3_finalize(stmt);
24 free(query);
25 return 1;
26 }
```

5 Concluzii

Aplicația "Offline Messenger" se include în categoria aplicațiilor dezvoltate după modelul TCP concurențial. Aceasta acoperă cerințele de bază necesare comunicării dintre utilizatori, însă ar putea fi dezvoltată pentru a oferi o gamă mai largă și complexă de opțiuni. O primă idee ar fi oferirea utilizatorului posibilitatea de a alege limba în care dorește să utilizeze aplicația, un minim de 3 limbi de exemplu ar fi binevenit. Ar fi util și o comandă ce prezintă istoria conversațiilor dintr-un interval orar sau de la o anumită dată calendaristică. De asemenea căutarea unui mesaj după un fragment din acesta, ar facilita aflarea informațiilor necesare din istoria conversațiilor.

6 Bibliografie

- "Rețele locale" Răzvan Rughiniș, Răzvan Deaconescu, Andrei Ciorba, Bogdan Doinea
- `https://profs.info.uaic.ro/~eonica/rc/lab06.html`
- `https://profs.info.uaic.ro/~eonica/rc/lab07.html`
- `https://www.sqlite.org/cintro.html`
- `https://zetcode.com/db/sqlitec/`