Rețele de calculatoare
RAPORT TEHNIC - Offline Messenger
Tema 2

Moisă Iulia-Elena (2A6)

Universitatea Alexandru Ioan Cuza, Iași
Facultatea de Informatică
iulia.moisa@info.uaic.ro

Offline Messenger

Să se dezvolte o aplicație client/server care să permită schimbul de mesaje între utilizatori care sunt conectați și să ofere 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 posibilitatea de a trimite un răspuns (reply) în mod specific la anumite mesaje primite. Aplicația va oferi și istoricul conversațiilor pentru și cu fiecare utilizator in parte.

1 Introducere

Proiectul *Offline Messenger* permite transmiterea mesajelor între utilizatorii înregistrați, aplicația având ca fundament comunicarea de tip client/server. Utilizatorii conectați vor putea comunica direct unii cu ceilalți, spre deosebire de cei offline care vor primi mesajele abia în momentul autentificarii.

Mesajele sunt stocate într-o bază de date, existând posibilitatea de a accesa istoricul conversațiilor pentru fiecare utilizator. Totodată, utilizatorul poate răspunde la un anumit mesaj specificand un nume de utilizator valid si ID-ul mesajului caruia vrea sa ii dea un raspuns.

Astfel, utilizatorii se pot înregistra pe server cu un username unic și o parolă, apoi se pot autentifica, ulterior putând să acceseze toate funcționalitățile.

2 Tehnologii utilizate

La baza proiectului există **modelul de tip server-client**. Acest model asigură paratajarea de resurse. Clientul inițiază conexiunea către server.Serverul se ocupă de coordonarea clienților, asigurând serviciile necesare pentru fiecare dintre aceștia. Comunicarea dintre server și client se realizează prin socket-uri, pentru a se trimite corespunzător fluxul de mesaje și informații.

Drept protocol de comunicare am ales **TCP (Transmission Control Protocol)**, deoarece scopul principal al acestuia este de a controla transferul de date

în așa fel încât acesta să fie de încredere; fiind vorba de o aplicație de comunicare, mă interesează în primul rând ca mesajele să ajungă la destinație fără să se piardă informații, iar calitatea să fie maximă.

Acest protocol este orientat conexiune si bidirecțional: putem scrie și citi în ambele direcții ale conexiunii concomitent (full-duplex).

Serverul folosit este unul de tip *concurent*, creându-se câte un **thread** pentru fiecare client în parte.Pentru a crea si manipula threadurile, programul utilizeaza standardul Pthreads (Posix Threads). Un thread (fir de executie) reprezinta cea mai mica sectiune de cod ce poate fi administrata de SO. Am ales threadurile datorită avantajelor din punct de vedere al timpului și al resurselor; acestora nu li se aloca memorie, PID, etc., fiind numite si *lightweight processes*.

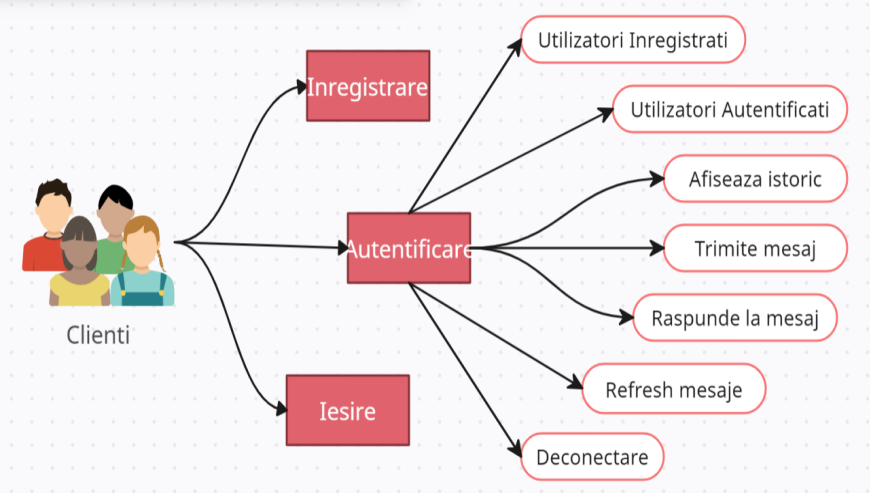
Threadurile si procesul principal impart aceeasi memorie, astfel ca variabilele de la nivelul procesului sunt accesibile oricarui thread.

Datele vor fi stocate într-o **bază de date SQLite** care va cuprinde mai multe tabele, printre care *UtilizatoriÎnregistrați*, *UtilizatoriAutentificați*, *Mesaje*, *MesajeNoi*. Am ales această bază de date deoarece interacționează facil cu limbajul ales, fiind totodată potrivită pentru aplicații de acest gen.

Serverul este cel care are legătură cu baza de date, el comunicând cu aceasta în urma procesării comenzilor primite de la clienți. Prin intermediul interogărilor (studiate la materia *Baze de date*) corespunzătoare comenzilor introduse, putem manipula datele în modul dorit.

Limbaj utilizat: C.

3 Arhitectura aplicației



Utilizatorii vor putea folosi următoarele comenzi:

– **Înregistrare**

Pentru ca un utilizator să acceseze aplicația, e necesar să își creeze un cont (prin introducerea unui username și a unei parole). Se verifică dacă username-ul introdus există; dacă nu, clientul se poate înregistra (îl adăugam în tabela *UtilizatoriÎnregistrați*).

- **Autentificare**
  Utilizatorii care au cont (verificăm existența lor în tabela *UtilizatoriÎnregistrați* cu ajutorul interogărilor) trebuie să se autentifice pentru a accesa aplicația. Autentificarea se efectuează pe baza introducerii unui username și a unei parole. O autentificare reușită presupune înregistrarea în tabela *UtilizatoriAutentificați* a utilizatorului în cauză. Odata autentificat, utilizatorul are acces la toate functionalitatile aplicatiei.

- **Trimite mesaj**
  Un $user_1$ poate trimite mesaj unui $user_2$, dacă $user_2$ există în baza de date. Mesajul va fi stocat intitial atat in tabela *Mesaje*, cat si in tabela *MesajeNoi*.

- **??????????????Citește mesaj???????????**
  Fiecare utilizator poate verifica dacă a primit mesaje noi. Daca exista mesaje noi, se afiseaza si utilizatorul/utilizatorii de la care le-a primit.

- **Răspunde la mesaj**
  Un utilizator va avea posibilitatea de a raspunde la un anumit mesaj (identificat prin ID), primit de la un alt utilizator cu care are o conversatie.

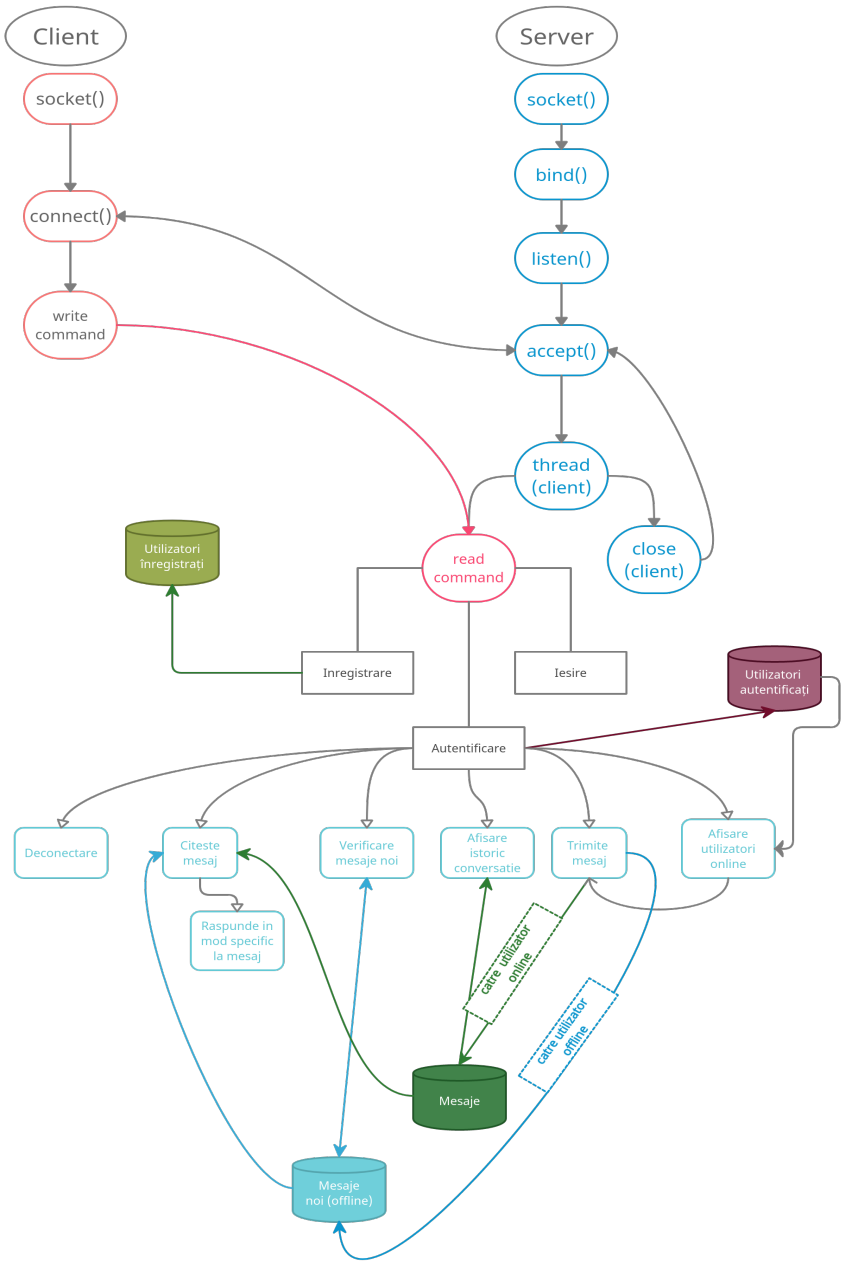
- **Afișare istoric conversație**
  Fiecare utilizator are posibilitatea de a verifica istoricul conversațiilor cu orice alt utilizator inregistrat.

- **Afișare utilizatori online**
  Fiecare utilizator poate verifica ce alti utilizatori sunt conectați, prin accesarea tabelului *UtilizatoriAutentificati*.

- **Afișare utilizatori**
  Fiecare utilizator poate verifica ce alti utilizatori sunt inregistrati in baza de date, in tabelul *UtilizatoriInregistrati*.

- **Deconectare**
  Utilizatorul se poate deconecta prin introducerea acestei comenzi. Astfel, el este eliminat din tabela *UtilizatoriAutentificați*.

- **Ieșire**
  Utilizatorul poate închide aplicația prin introducerea acestei comenzi.



4 Detalii de implementare

Primitiva socket facilitează comunicarea între sever și client. Un socket este un canal bidirecțional, astfel informația poate fi transmisă în ambele părți ale conex-

iunii. Acesta poate fi asociat unuia sau mai multor procese. Clientul conectează descriptorul de socket la adresa serverului (*connect*), care la randul ei este asignată socket-ului, pe partea de server(*bind*). Socket-ul acceptă conexiuni prin ascultare (*listen*). Apelarea *accept* reprezintă conectarea propriu-zisa cu clientii.

În server se găsește o buclă infinită prin intermediul căreia se acceptă clienți. Odată ce a sosit un client, se creează un thread pentru acesta.

```
typedef struct thData{
  int idThread; //id-ul thread-ului tinut in evidenta de acest
     program
  int cl; //descriptorul intors de accept
}thData;
static void *treat(void *); /* functia executata de fiecare
    thread ce realizeaza comunicarea cu clientii */
```

```
 pthread_t th[100];    //Identificatorii thread-urilor
 int i=0;
  while (1)
    {
      int client;
      thData * td; //parametru functia executata de thread
      int length = sizeof (from);

      printf ("[server]Asteptam la portul %d...\n",PORT);
      fflush (stdout);

      // acceptam un client-stare blocanta
      if ( (client = accept (sd, (struct sockaddr *) &from, &
    length)) < 0)
      {
        perror ("[server]Eroare la accept().\n");
        continue;
      }
    /* s-a realizat conexiunea, se astepta mesajul */
  td=(struct thData*)malloc(sizeof(struct thData));
  td->idThread=i++;
  td->cl=client;//cl = descriptorul intors de accept

  pthread_create(&th[i], NULL, &treat, td);//creare fir de
    executie nou

  }//while
//din curs
```

Programul cuprinde tabelele $UtilizatoriInregistrati$, $UtilizatoriAutentificati$, $Mesaje$ si $MesajeNoi$. Crearea acestora este prezentată mai jos.

```
void creareTabele(){
  char *err_msg = 0;
  sqlite3* db;
  char *sql;
```

```
5   int rc = sqlite3_open("OM_BazaDeDate.db", &db); //deschidere baza de date
6   if(rc != SQLITE_OK) {
7     fprintf(stderr, "baza de date nu poate fi deschisa: %s\n", sqlite3_errmsg(db));
8         sqlite3_close(db);
9     }
10    else
11        printf("Baza de date a fost deschisa cu succes\n");
12    //creare tabele
13    sql = "CREATE TABLE IF NOT EXISTS UtilizatoriInregistrati (user_ID INTEGER PRIMARY KEY NOT NULL, nume_user TEXT NOT NULL, parola TEXT NOT NULL);";
14  rc = sqlite3_exec(db, sql, 0, 0, &err_msg );
15  if(rc != SQLITE_OK){
16    fprintf(stderr, "Eroare la crearea tabelei1: %s\n", err_msg );
17    sqlite3_free(err_msg);//The allocated message string must be freed
18    sqlite3_close(db);
19  }
20  else fprintf(stdout, "Tabela UtilizatoriInregistrati s-a creat cu succes!\n");
21
22  sql = "CREATE TABLE IF NOT EXISTS UtilizatoriAutentificati ( user_ID INTEGER PRIMARY KEY, nume_user TEXT NOT NULL);";
23  rc = sqlite3_exec(db, sql, 0, 0, &err_msg );
24    if(rc != SQLITE_OK){
25      fprintf(stderr, "Eroare la crearea tabelei2: %s\n", err_msg );
26      sqlite3_free(err_msg );
27    }
28  else fprintf(stdout, "Tabela UtilizatoriAutentificati s-a creat cu succes!\n");
29        sql = "CREATE TABLE IF NOT EXISTS Mesaje(mesaj_ID INTEGER PRIMARY KEY, expeditor TEXT NOT NULL, destinatar TEXT NOT NULL, continut_mesaj TEXT NOT NULL);";
30  rc=sqlite3_exec(db, sql, 0 ,0, &err_msg);
31  if(rc != SQLITE_OK){
32    fprintf(stderr, "Eroare la crearea tabelei3: %s\n",err_msg );
33    sqlite3_free(err_msg );
34  }
35  else fprintf(stdout, "Tabela Mesaje s-a creat cu succes!\n");
36
37        sql = "CREATE TABLE IF NOT EXISTS MesajeNoi(mesaj_ID INTEGER PRIMARY KEY, expeditor TEXT NOT NULL, destinatar TEXT NOT NULL, continut_mesaj TEXT NOT NULL);";
38  rc = sqlite3_exec(db, sql, 0, 0, &err_msg );
```

```
  if(rc != SQLITE_OK){
    fprintf(stderr, "Eroare la crearea tabelei4: %s\n",err_msg
    );
    sqlite3_free(err_msg );
  }
  else fprintf(stdout, "Tabela MesajeNoi s-a creat cu succes!\
    n");
}
```

Un utilizator nou trebuie să se înregistreze în aplicație cu un username (nemaifolosit înainte) și o parolă (comanda *Înregistrare*), pentru a avea acces la functionalitatile aplicatiei. Daca username-ul este valid, inregistram utilizatorul in tabela.

```
int Inregistrare(char* nume_user, char* parola)
{
  sqlite3 *db;
  sqlite3_stmt * res;//a single sql statement
  char *err_msg = 0;
  int ok = 0;
  if (sqlite3_open("OM_BazaDeDate.db", &db) != SQLITE_OK)
  {
    fprintf(stderr, "Baza de date nu poate fi deschisa: %s\n",
    sqlite3_errmsg(db));
    sqlite3_close(db);
  }
  char sql[256];
  sprintf(sql, "INSERT INTO UtilizatoriInregistrati (nume_user
    , parola) VALUES ('%s', '%s');", nume_user, parola);
  int rc = sqlite3_exec(db, sql, 0, 0, &err_msg);//allows an
    application to run multiple sql stmts without having to
    use a lot of C code
    if (rc != SQLITE_OK) {
        fprintf(stderr, "Eroare la inregistrare: %s\n",
    err_msg);
        sqlite3_free(err_msg);
    }
    else {
        ok = 1;
        printf("Inregistrarea a avut loc cu succes!\n");
  }
  sqlite3_close(db);
  return ok;
}
```

Un utilizator conectat, se poate deconecta de la aplicatie in orice moment prin apelarea comezii *Deconectare*. Astfel, va fi eliminat din tabela *UtilizatoriAutentificati* si nu va mai avea acces la functionalitatile aplicatiei.

```
int Deconectare(char* nume_user){
  int rc, ok = 0;
```

```
  sqlite3 *db;
  sqlite3_stmt * res;
  if (sqlite3_open("OM_BazaDeDate.db", &db) != SQLITE_OK){
    fprintf(stderr, "Baza de date nu poate fi deschisa: %s\n",
     sqlite3_errmsg(db));
    sqlite3_close(db);
  }
  else{
    char* sql = "DELETE FROM UtilizatoriAutentificati WHERE
    nume_user=?2;";
    rc = sqlite3_prepare_v2(db, sql, -1, &res, NULL);
    if (rc == SQLITE_OK){
      ok = 1;
      sqlite3_bind_text(res, 2, nume_user, -1, SQLITE_STATIC);
      sqlite3_step(res);
      sqlite3_finalize(res);
    }
    else printf("Ceva nu e ok la deconectare\n");
    sqlite3_close(db);
  }
  return ok;
}
```

5 Concluzii

Motivul principal pentru care am ales acest proiect este faptul că aplicațiile de chatting reprezintă un instrument utilizat zilnic în zilele noastre. Astfel, mi s-a părut util să înțeleg cum funcționează ele și să încerc și eu să implementez o astfel de aplicație.

Printre îmbunătățirile care ar putea fi aduse aplicației mele se numără:

- ștergerea unui anumit mesaj sau a unei conversații intregi
- comunicarea în groupchats
- o interfață atractivă
- opțiune de a trimite fișiere (poze, pdf-uri, etc...) între utilizatori
- blocare conturi

6 Bibliografie

- https://profs.info.uaic.ro/~computernetworks/index.php
- https://profs.info.uaic.ro/~ioana.bogdan/
- https://man7.org/linux/man-pages/
- https://profs.info.uaic.ro/~computernetworks/files/NetEx/S12/ServerConcThread/servTcpConcTh2.c
- https://zetcode.com/db/sqlitec/

– https://www.sqlite.org/cintro.html
– https://profs.info.uaic.ro/~georgiana.calancea/Laboratorul_12.pdf
– https://www.sqlite.org/c3ref/bind_blob.html