REȚELE DE CALCULATOARE
Facultatea de Informatică Iași
TEMĂ 2
Raport tehnic al proiectului
OFFLINE MESSENGER

Ghiorghiu Ana-Maria (2A4)

January 14, 2022

Offline Messenger(B)
Sa se dezvolte o aplicatie client/server care sa permita schimbul de mesaje intre utilizatori care sunt conectati si sa ofere functionalitatea trimiterii mesajelor si catre utilizatorii offline, acestora din urma aparandu-le mesajele atunci cand se vor conecta la server. De asemenea, utilizatorii vor avea posibilitatea de a trimite un raspuns (reply) in mod specific la anumite mesaje primite. Aplicatia va oferi si istoricul conversatiilor pentru si cu fiecare utilizator in parte.

1 Introducere

Proiectul Offline Messenger se bazează pe comunicarea de tip client/server prin intermediul căreia putem trimite mesaje utilizatorilor atât online cât și offline. Utilizatorii care nu sunt conectați vor primi mesajele în momentul conectării iar mesajele sunt stocate într-o bază de date de unde va fi furnizat și istoricul mesajelor. Clienții au posibilitatea de a se loga la server cu username și parolă, iar din momentul în care autentificarea a luat loc, aceștia pot comunica între ei, pot da reply la anumite mesaje, iar apoi se pot deconecta folosindu-se de logout.

2 Tehnologiile utilizate

- Aplicația "Offline Messenger" este implementată în limbajele de programare C/C++.
- Din punct de vedere al protocolului de comunicare utilizat, am folosit TCP(un protocol orientat conexiune în loc de UDP, protocol neorientat conexiune).
  - Protocolul de comunicare TCP asigură conexiune de calitate maximă între client și server deoarece folosește mecanisme care evită pierderea de informații, element vital pentru o aplicație destinată schimbului de mesaje.
  - Un alt motiv pentru care am ales acest protocol este faptul că se pot transmite și capta date simultan din ambele direcții.
- Din punct de vedere al modalității de gestionare a proceselor, am ales threadurile, deoarece sunt mai avantajoase decât procesele ca timp și resurse.
- Din punct de vedere al stocării datelor(mesajelor, numelor de utilizatori, parole) am folosit baza de date MySQL. Biblioteca folosită în cod este <mysql/mysql.h>. Folosindu-ne de interogările studiate ce țin de inserarea datelor, căutarea printre date, sortarea, crearea și ștergerea lor, putem naviga prin bază și să obținem numele utilizatorilor, parola aleasă de aceștia, precum și să verificăm dacă aceștia s-au mai înregistrat anterior, mesajele trimise, istoricul de mesaje trimise de utilizatori.

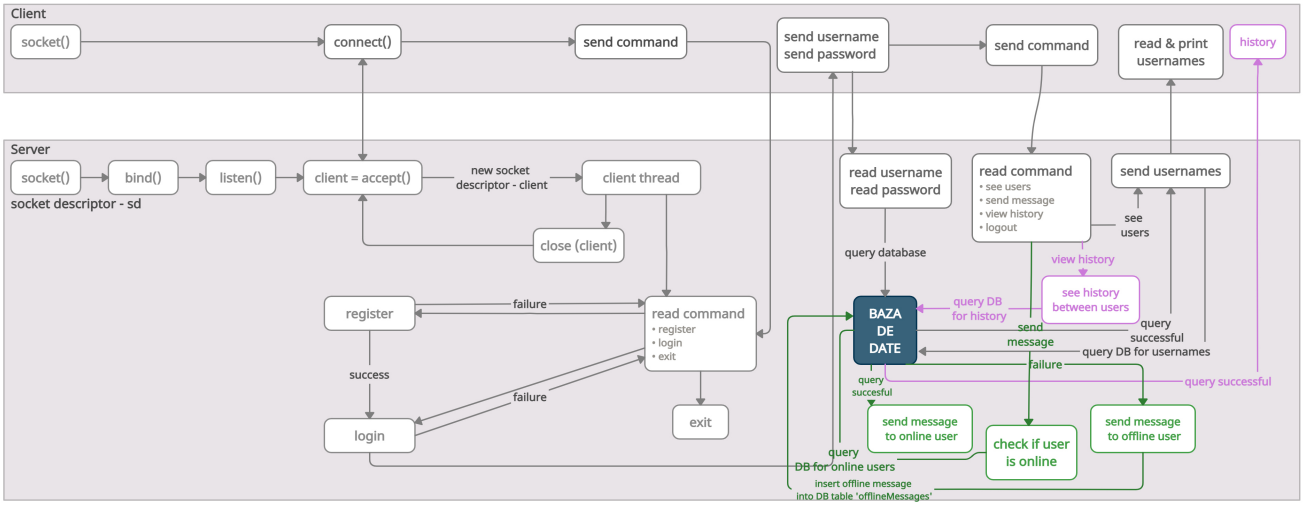
3 Arhitectura aplicației

Aplicația 'Offline Messenger' este structurată în server și client. Pe server se vor conecta mai mulți clienți simultan. Comunicarea dintre server și client se realizează prin socket-uri, pentru a se trimite corespunzător fluxul de mesaje și informații.
În cadrul aplicației, utilizatorii vor putea folosi următoarele comenzi, pentru care am implementat câte o funcție:

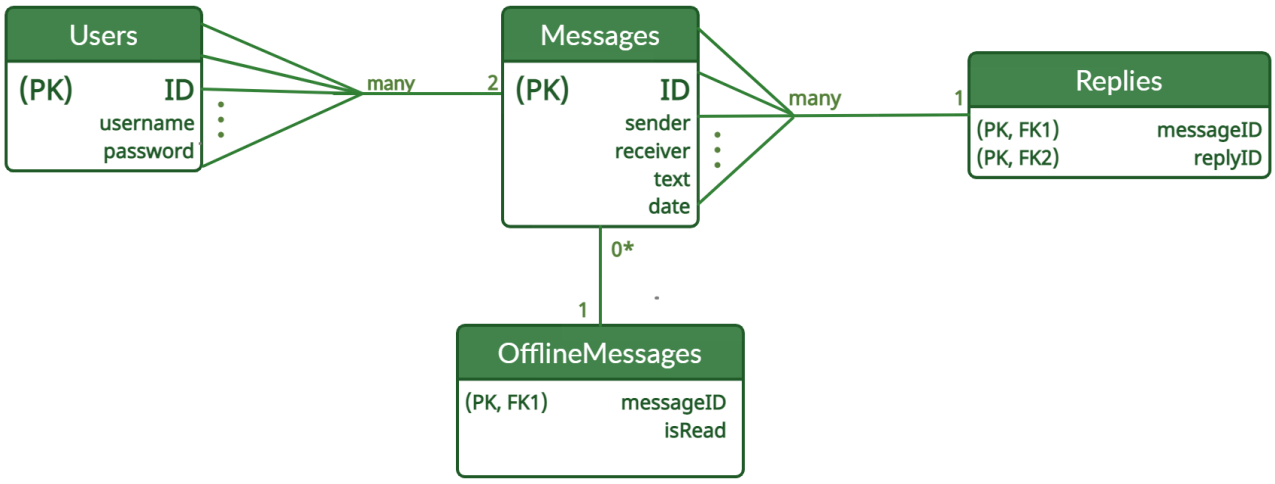
- **Register**
  Userul își va introduce username-ul și parola, iar apoi se va verifica dacă acest username nu există deja. Dacă nu există deja, acesta va fi înregistrat în tabela Users.
- **Login**
  Userul care a fost înregistrat anterior se va autentifica.
- **Send message**
  Userul va putea trimite mesaje altui user, la alegere din lista de useri.
- **Reply to specific message**
  Userul va putea să trimită un mesaj ca răspuns la un mesaj specific.
- **Show Users**
  Userul va putea vizualiza lista de useri folosind aceasta comandă.
- **Show conversation history**
  Această comandă afișează istoricul conversațiilor.
- **Show messages received while being offline**
  Această comandă îi permite userului care nu a fost online să își vizualizeze mesajele primite.
- **Logout**
  Această comandă îl deconectează pe user.
- **Exit**
  Folosind această comandă, userul închide aplicația 'Offline Messenger'.

Pentru a ilustra funcționalitatea proiectului, voi insera următoarele diagrame:

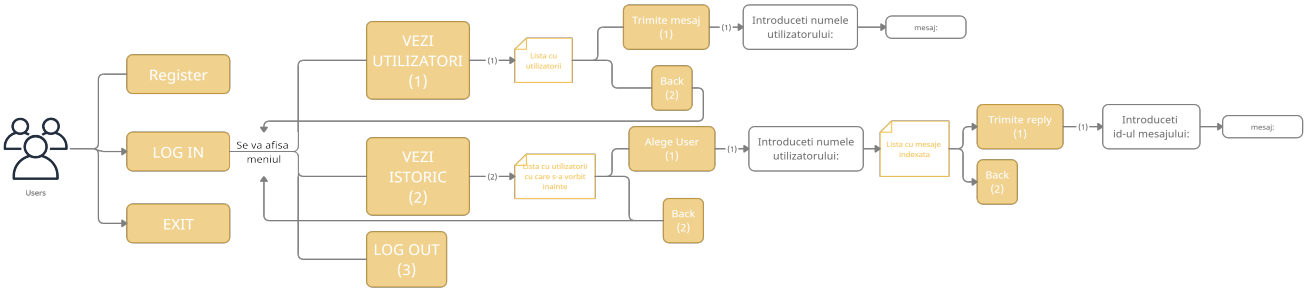
- **Fig. 1** Structura programului



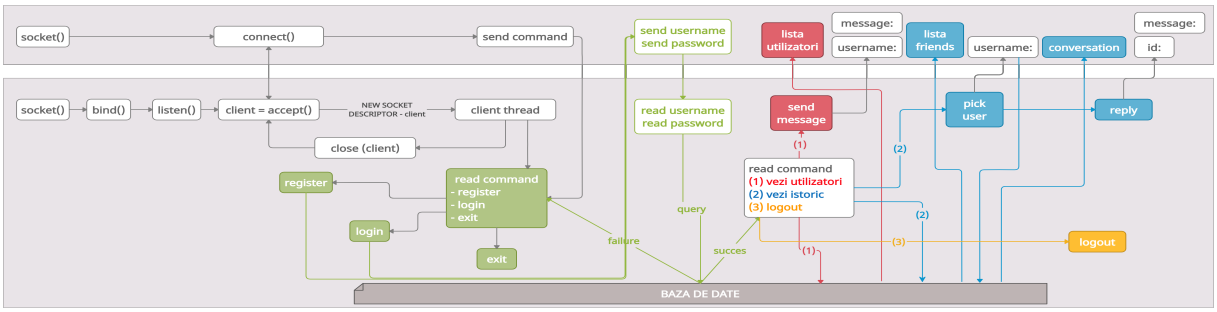
- **Fig. 2** Schemă conceptuală a tabelelor din baza de date



- **Fig. 3** Diagrama Use Case



- **Fig. 4** Schema generala a functionalitatii programului



4 Detalii de implementare

Comunicarea dintre server si client se realizează în primul rând cu ajutorul unui socket. Folosirea unui socket pentru este mult mai avantajoasă datorită bidirecționalității sale (spre deosebire de pipe-uri/fifo care sunt unidirecționale). Acest lucru este avantajos și din prisma faptului că nu trebuie să realizăm și închiderea capetelor canalului de transmisie.

Crearea unui socket în server

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

- In structura proiectului, am implementat funcții care sunt necesare operațiilor efectuate, precum conectarea la baza de date.
- Pentru a gestiona datele din aplicație am implementat o bază de date SQL. Am ales să implementez o bază de date în loc de gestionarea fișierelor din următorul motiv : Folosirea fișierelor .txt ar fi într-adevăr mai simplă, insă accesul la date s-ar produce mai lent, pentru că ar trebui aplicat de fiecare dată un algoritm de citire a fișierului, un algoritm de căutare pe baza informațiilor, care repetitiv aplicate ar putea duce la încetinirea aplicației.
- Stocarea manuală a informațiilor este dezavantajoasă comparativ cu stocarea informațiilor pe bază de interogări întrucât pentru acest tip de aplicație, algoritmul implementat mai jos oferă bazei de date o interogare simplă și rezultatul dorit este afișat în mai puțin timp. Deci, extragerea, modificarea și afișarea datelor se face avantajos cu ajutorul funcțiilor din biblioteca pe care am utilizat-o, funcții pe care le-am implementat în interiorul serverului:

Conectarea la baza de date

```
MYSQL *conn = mysql_init(NULL);
if (conn == NULL)
{
    fprintf(stderr, "%s\n", mysql_error(conn));
    exit(1);
}
if (mysql_real_connect(conn, "localhost", "root", "230700", "OfflineMessenger",
 0, NULL, 0) == NULL)
{
    fprintf(stderr, "Am dat peste o eroare: %s\n", mysql_error(conn));
    mysql_close(conn);
    exit(1);
}
```

Comanda SELECT

```
char select[200] = "SELECT count(name) from Users WHERE name=?";
if (mysql_stmt_prepare(stmt, select, strlen(select)))
{
    fprintf(stderr, " mysql_stmt_prepare(), INSERT failed\n");
    fprintf(stderr, " %s\n", mysql_stmt_error(stmt));
    return true;
}
```

Comanda INSERT

```
char insert[100] = "INSERT INTO Users(name,password) VALUES (?,?)";
 mysql_stmt_prepare(stmt, insert, strlen(insert));
```

Pentru a asigura concurența serverului, în implementarea acestuia am folosit thread-uri, astfel pentru fiecare client care se va conecta, se va crea un nou thread care se va ocupa de comenzile date de user.

Thread

```
  typedef struct thData
{
    int idThread; //id-ul thread-ului tinut in evidenta de acest program
    int cl;       //descriptorul intors de accept
    int idUser;   //id-ul userului logat
} thData;

thData *clients[100];
int nrClients = 0;

static void *treat(void *); /* functia executata de fiecare
 thread ce realizeaza comunicarea cu clientii */
```

- Comanda Register este prima comandă pe care o poate plasa un user nou. Acesta se va autentifica cu un username și o parolă și va fi autentificat.
- Comanda Register va anunța userul atunci când alege un username invalid și îi va cere să introducă datele în mod corect.
- Condiția pentru ca un username să fie valid pentru înregistrarea în aplicație este ca username-ul ales să nu fie ulterior deja utilizat, în acest caz va trebui utilizată comanda Login.
- Atunci când userul va introduce un username valid, va avea acces la al doilea meniu de opțiuni.

Pentru funcționalitatea funcțiilor pe care urmează să le implementez, am făcut o funcție de citire și de scriere pe care le voi folosi pe tot parcursul proiectului.
Aceste funcții îmi vor arăta dacă se produce o posibilă eroare la citirea/scrierea descriptorului.

Read

```
void Read(int sd, int len, char *mesaj, thData td)
{
    int lenBytes = 0, mesajBytes = 0;
    if ((lenBytes = read(sd, &len, sizeof(int))) < 0)
        perror("[Parinte]Err...read");
    if ((mesajBytes = read(sd, mesaj, len)) < 0)
        perror("[Parinte]Err...read");
    if (lenBytes == 0 && mesajBytes == 0)
    {
        close(sd);
        close(clients[td.idThread]->cl);
        clients[td.idThread]->cl = -1;
    }
}
```

Write

```
bool Write(int sd, int len, char mesaj[300])
{
    int a;
    if (write(sd, &len, sizeof(int)) < 0)
        perror("[Parinte]Err...write");
    if ((a = write(sd, mesaj, len)) < 0)
        perror("[Parinte]Err...write");
    return a != 0;
}
```

Funcții implementate

```
 * Funcția checkIfUserExists
  bool checkIfUserExists(char user[40]){}
Se inițializează conexiunea la baza de date. Vom primi eroare dacă
conexiunea nu s-a realizat. Se caută în baza de date cu ajutorul
unei interogări dacă username-ul există deja.Această funcție ne ajută la
Register/Login.
 * Functia getMessageId
 int getMessageId(int sender, int receiver, MYSQL *conn){}
 Aceasta functie returneaza id-ul unui mesaj, se caută în baza de date
 ultimul mesaj dintre doi useri, pentru a se realiza indexarea mesajelor.
 Id-ul mesajelor este, incepand de la 1, terminand cu ultimul mesaj, numarul de
 mesaje in ordine crescatoare.
 * Functia senfOfflineMessages
 void sendOfflineMessages(thData td){}
 Functia verifica daca clientul este conectat, si daca nu, se introduce in baza de
 date pentru a-i fi afisat mai tarziu cand se va conecta.
 * Functia insertMessage
 char *insertMessage(int receiver, int sender, char *mesaj, int isOfflineMessage){}
 Functia actualizeaza baza
 de date atunci cand clientul se conecteaza si citeste mesajul.
```

5 Concluzii

- Un element vital pentru o aplicație de schimb al mesajelor este confidențialitatea comunicării dinte useri, ce poate fi securizată prin adaugarea unui sistem de criptare a mesajelor.
- Un alt element important ar putea reprezenta interfața grafică care ar putea oferi clienților o experiență plăcută.
- Fiind un o aplicatie ce se ocupă cu schimbul de mesaje, poate oferi și o funcție de blocare a anumitor useri, adaugarea unui profil personal.
- Codul încărcat de mine rezolvă structura propusa de proiect în mod parțial și poate fi îmbunătățită, spre exemplu funcția Register nu oferă încă verdictul dacă userul s-a înregistrat în trecut cu același username.

6 Bibliografie

[1] Computer Networks - Faculty of Computer Science Website

- `https://profs.info.uaic.ro/~computernetworks/index.php`
- `https://profs.info.uaic.ro/~computernetworks/files/NetEx/S12/ServerConcThread/servTcpConcTh.c`
- `https://profs.info.uaic.ro/~computernetworks/files/NetEx/S12/ServerConcThread/servTcpConcTh2.c`
- `https://profs.info.uaic.ro/~computernetworks/files/NetEx/S12/ServerConcThread/cliTcpNr.c`

[2] MySQL 8.0 C API Website
`https://docs.oracle.com/cd/E17952_01/c-api-8.0-en/index.html`

[3] MySQL C API programming
`https://zetcode.com/db/mysqlc/`

[4] C API Prepared Statement Function Descriptions
https://dev.mysql.com/doc/c-api/8.0/en/c-api-prepared-statement-function-descriptions.html

- MySQL C API Prepared Statement Function Descriptions - mysql stmt execute()
https://dev.mysql.com/doc/c-api/8.0/en/mysql-stmt-execute.html
- MySQL C API Prepared Statement Function Descriptions - mysql stmt bind param()
https://dev.mysql.com/doc/c-api/5.7/en/mysql-stmt-bind-param.html