Raport tehnic al proiectului Offline Messenger

Harton Amalia-Nicola, grupa 2A4

6 Decembrie 2022

1 Introducere

Proiectul Offline Messenger dezvoltă o aplicație client/server care permite schimbul de mesaje între utilizatorii conectați la server. Atât utilizatorii conectați, cât și cei offline pot trimite mesaje. Utilizatorii offline au opțiunea de a vizualiza mesajele atunci când se conectează la server.

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.

Utilizatorii pot trimite răspunsuri specifice la anumite mesaje primite și pot vizualiza istoricul conversațiilor cu fiecare utilizator cu care au comunicat. 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.

1.1 Motivație

Am ales proiectul *Offline Messenger*, deoarece acest proiect poate sta la baza realizării unei aplicații mai mari, precum tipul chaturilor. Totodată prin realizarea proiectului, consinder că voi înțelege mai bine noțiunile legate de această materie.

1.2 Cuvinte cheie

- protocol de comunicare
- TCP
- client
- server
- online
- offline
- reply
- istoric conversații
- **fork()**
- **socket()**

2 Tehnologii utilizate

Aplicația "Offline Messenger" este implementată în limbajul de programare 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 concurent 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.

Implementarea server-ului se va realiza cu ajutorul funcției **fork()**, asigurând concurența. Voi folosi funcția **fork()** pentru a crea câte un proces copil pentru fiecare client care se conectează la server.

Pentru înregistrarea, logarea și delogarea utilizatorilor, dar și pentru a reține istoricul conversațiilor, voi folosi SQLITE. SQLITE este un sistem de gestiune a bazelor de date relaționale, fiind compatibil cu o varietate de limbaje de programare și tehnologii și totodată, fiind favorabil pentru această aplicație, deoarece prin simplitatea lui pune la dispoziție toate resursele necesare dezvoltării acesteia.

Utilizarea unei baze de date în defavoarea unui fisier text este mai eficientă, iar gestionarea datelor este mai ușoară.

3 Arhitectura aplicației

3.1 Conceptele implicate

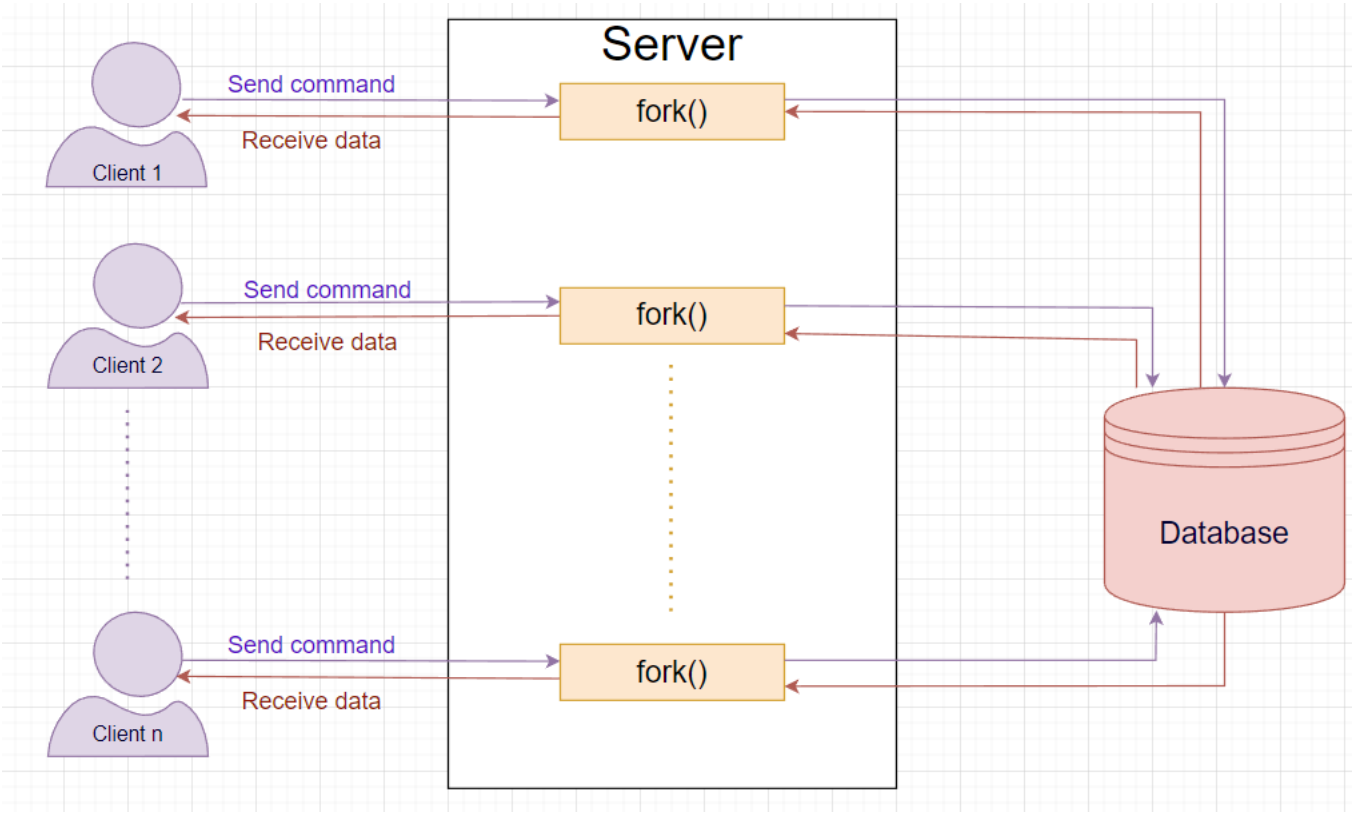
Pentru funcționalitatea aplicației, voi folosi următoarele noțiuni: **server**, **client** și **baza de date**.Un server poate gestiona mai mulți clienți în același timp. Acesta utilizează baza de date pentru a valida cererile primite de la clienți. Dacă totul este în regulă, baza de date returnează informațiile solicitate și serverul returnează un răspuns clientului.

Pentru centralizarea utilizatorilor, am ales să creez o bază de date cu patru câmpuri: id, username, password, online (1-online, 0-offline).

ID	Jsername	Password	Online
Fi...	Filter	Filter	Filter
1	Amalia	amalia	1
2	Maria	maria	1
3	Maya	maya	0

Baza de Date a utilizatorilor

3.2 Diagrama aplicației detaliată



4 Detalii de implementare

4.1 Cod relevant proiectului

Comunicarea dintre server și client se va realiza cu ajutorul unui **socket**.

În comunicare, utilizarea unui **socket** este avantajoasă, deoarece acesta este bidirecțional și poate fi utilizat atât pentru a comunica între procese de pe același calculator, dar în special pentru a asigura comunicarea in rețea.

În funcția *main()*, voi crea baza de date și tabelele necesare stocării și prelucrării informațiilor, voi declara variabilele, voi crea socket-ul utilizat în comunicare, iar socket-ului îi voi atașa și partea de "ascultare" a clienților care doresc să se conecteze.

```
void initializareBaza()
{
    int rc = sqlite3_open("users.db",&db);
        if(rc != SQLITE_OK)
    {
        printf("Eroare\n");
    }
    sqlite3_exec(db,"CREATE TABLE Users(ID INTEGER PRIMARY KEY, Username TEXT,Password TEXT,Online INT);",0,0,&err);
    sqlite3_exec(db,"Update Users SET Online=0;",0,0,&err);
}
```

Inițializarea Bazei de Date

```c
int sd;

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

Crearea unui socket în server

În dezvoltarea aplicației voi folosi primitiva **fork()** pentru a crea paralelismul între clienții conectați la server.

```c
int pid;
if ((pid = fork()) == -1)
{
    close(client);
    continue;
}
else if (pid > 0)
{
    close(client);
    while(waitpid(-1,NULL,WNOHANG));
    continue;
}
else if (pid == 0)
{
    close(sd);
    while(1)
    {
        bzero (msg, 256);
        printf ("Asteptam mesajul...\n");
        fflush (stdout);

        if (read (client, msg, 256) <= 0)
        {
            perror ("Eroare la read() de la client.\n");
            close (client);
            continue;
        }

        printf ("Mesajul a fost receptionat...%s\n", msg);
```

Utilizarea primitivei fork în server

Comanda sendMessage

Dacă clientul este un utilizator nou al aplicatiei, își va crea un cont cu ajutorul comenzii *inregistrare*. Dacă acesta este creat cu succes, server-ul îi va permite următoarele funcționalități: *seeNewMessage*, *sendMessage*, *reply*, *seeHistory*, *showUsers*, *logout* și *exit*.

Dacă utilizatorul are deja un cont, iar prin comanda *login* reușește să se conecteze, va putea utiliza urmatoarele comenzi: *seeNewMessage*, *sendMessage*, *reply*, *seeHistory*, *showUsers*,*seeInbox*, *logout*, *exit*.

Atunci când clientul selectează comanda *seeHistory*, i se va afișa tot istoricul cu o anumită persoană, iar atunci când clientul selectează comanda *seeNewMessage*, i se vor afișa mesajele noi primite și le va marca ca citite.

```
void sendMessage(char sender[256], char receiver[256], char mesaj[8192], char msgrasp[8192])
{
    char sql[1024];
    char senderSign[256];
    char first[256];
    char second[256];
    char msg[8192];
    bzero(sql, 1024);
    bzero(first, 1024);
    bzero(second, 1024);
    bzero(senderSign, 1024);
    showUsers(msg); // retinem lista cu numele utilizatorilor
    if (strstr(msg, receiver) != 0)//verific ca cel catre care trimite mesajul sa fie in lista de utilizatori
    {
        creareInbox(sender);
        creareInbox(receiver);
        // Creează o conversație între sender și receiver
        creareConversatie(user, receiver);

        if ((verificareTabel(sender, receiver) == 1) || (verificareTabel(sender, receiver) == 0))
        {
            strcpy(first, sender);
            strcpy(second, receiver);
        }
        else if (verificareTabel(sender, receiver) == 2)
        {
            strcpy(first, receiver);
            strcpy(second, sender);
        }

        inserareInbox(receiver, sender);
        sprintf(sql, "INSERT INTO %s%s(Sender, Receiver, Mesaj,Citit,Timp) VALUES ('%s', '%s', '[%s]%s','0',datetime('now','localtime'));", first, second, sender, receiver, sender, mesaj);//inserez mesajul
        sqlite3_exec(db, sql, 0, 0, &err);
        sqlite3_close_v2(db);
        RedeschidereBaza();
        sprintf(msgrasp, "[server]Mesaj trimis cu succes catre %s", receiver);
    }
    else
    {
        strcpy(msgrasp, "[server]Utilizatorul caruia incercati sa-i trimiteti un mesaj nu se afla in baza de date");
    }
}
```

Comanda *sendMessage* permite utilizatorilor să trimită mesaje către anumite persoane. Dacă utilizatorul va dori să utilizeze comanda *logout*, acesta se va deloga, iar prin comanda *exit* va părăsi sesiunea curentă.

4.2 Scenarii de utilizare

Pentru început, un utilizator nou care dorește să utilizeze aplicația "Offline Messenger", trebuie să își creeze un cont. Cu ajutorul comenzii inregistrare, acesta va fi nevoit să introducă un **username** și o **parolă**, care vor fi stocate în baza de date, având un unic **id**.

Pentru un utilizator care are deja un cont și dorește să folosească aplicația, va trebui să se ajute de comanda login. Cu ajutorul acesteia, va introduce datele sale, iar dacă sunt corecte, server-ul îl va recunoaște și i se va oferi șansa să utilizeze și celelalte comenzi: sendMessage, seeNewMessage, reply, seeHistory, showUsers,seeInbox și logout.

Comanda seeHistory cere utilizatorului să introducă username-ul persoanei către care este trimis mesajul. Dacă totul este în regulă, utilizatorul va putea vizualiza istoricul conversației.

Prin accesarea comenzii seeNewMessage, unde va trebui să introducă numele persoanei de la care crede ca a primit mesaj si va putea vizualiza noile mesaje pe care le are cu acea persoana.

Dacă utilizatorul alege comanda sendMessage, va trebui să introducă numele persoanei la care va dori sa ajunga mesajul și în final, mesajul dorit. Când receptorul se va loga, și va introduce comanda seeNewMessage si va observa ca are un nou mesaj.

Comanda showUsers afiseaza utilizatorii care sunt stocati in baza de date.

Accesand comanda seeInbox poti vedea de la care utilizatori ai mesaje noi.

În final, prin accesarea comenzii logout, revine în meniul anterior, din care se poate (re)loga pentru a avea acces la funcționalitățile menționate anterior.

Prin accesarea comenzii exit, se va închide conexiunea clientului cu serverul, iar dacă clientul era cumva logat, prin accesarea comenzii de exit, acesta se va și deloga.

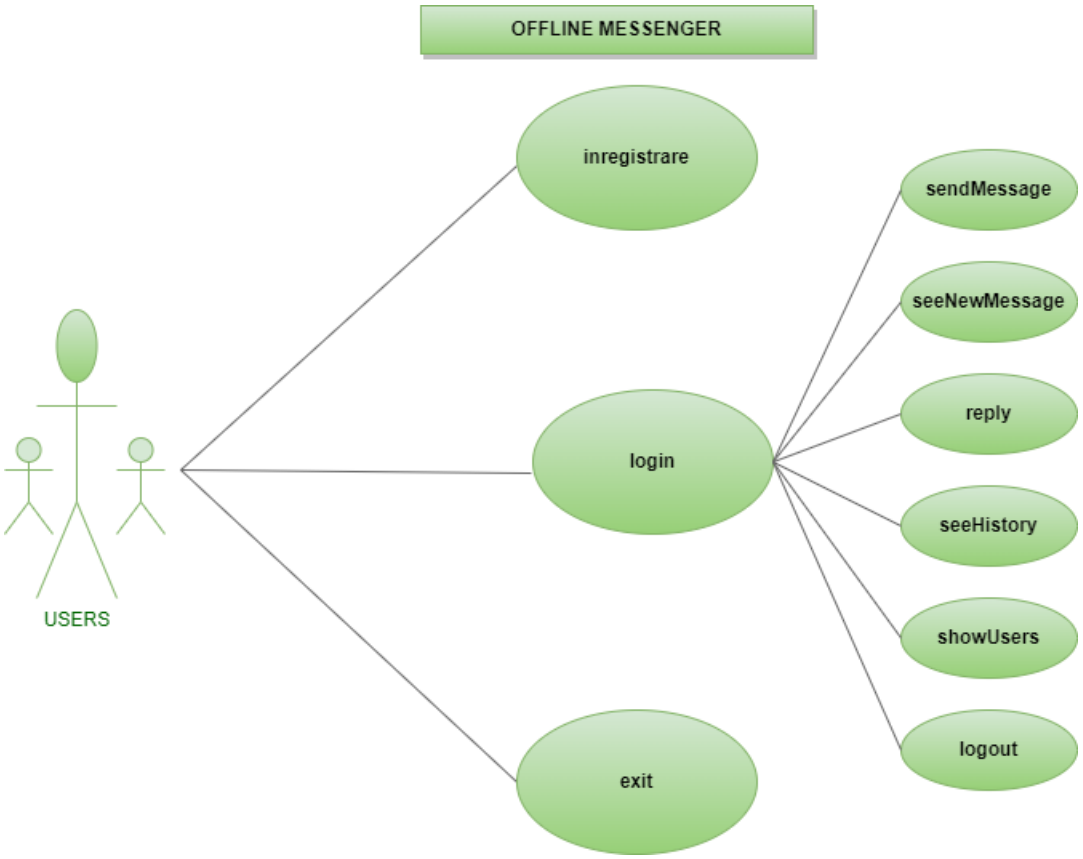


Fig.2 Diagrama Use Case

5 Concluzii

Una dintre ideile de îmbunătățire a soluției pe care am găsit-o este realizarea unei interfețe grafice, care ar putea oferi clienților o experiență plăcută.

O altă idee de îmbunătățire a aplicației ar fi posibilitatea utilizatorilor să ofere reacții mesajelor.

O altă idee a aplicației ar putea fi adăugarea contactelor noi, respectiv ștergerea contactelor sau realizarea unui grup de messenger, unde pot participa cat mai mulți utilizatori.

O ultimă idee ar fi posibilitatea ștergerii unei conversații.

6 Bibliografie

- https://profs.info.uaic.ro/ computernetworks/cursullaboratorul.php
- https://profs.info.uaic.ro/ ioana.bogdan/
- https://www.geeksforgeeks.org/differences-between-tcp-and-udp/
- https://www.geeksforgeeks.org/socket-in-computer-network/

• https://app.diagrams.net/?fbclid=IwAR1sthrM52oMy1wvexCpkiXBCyK0PbfrHm17wOvn7a8VKxGK-Gdc7yyAjpg

• https://www.sqlite.org/cintro.html

• https://stackoverflow.com/