Offline Messenger (B) - Raport Tema 2

Tanase Alexandru-Ionut

Universitatea Alexandru Ioan Cuza, Facultatea de Informatica - Iasi

1 Introducere

Acest raport este bazat pe proiectul Offline messenger, un proiect ce face parte din categoria B.

Offline messenger este o aplicatie de tip text-chat ce permite comunicarea real-time intre 2 utilizatori. La baza aplicatiei stau 2 componente principale : **clientul** prin intermediul caruia utiliztatorii se vor conecta la **server**. Serverul va realiza "comunicarea" propriu-zisa intre persoanele conectate, dar va putea oferi si informatii despre acestea, de exemplu datele de logare. Clientul le va pune la dispozitie utilizatorilor posibilitatea de a da reply-uri la anumite mesaje si totodata de a vizualiza istoricul conversatiei dintre acestia si alte persoane.

2 Tehnologii Utilizate

Aplicatia are la baza protocolul de comunicare TCP. Acesta ofera o conexiune sigura intre server si client. Am ales modelul TCP in detrimentul celui UDP deoarece in acest mod avem siguranta in ceea ce priveste integritatea datelor pe parcursul transimterii acestora, un factor decisiv in realizarea acestei alegeri. Din punct de vedere al limbajelor utilizate, am folosit C/C++ in realizarea aplicatiei. Tot ce tine de memorarea datelor de conectare ale utilizatorilor si de istoricul conversatiilor dintre acestia s-a realizat folosind simple fisiere cu extensia ".txt".

3 Arhitectura Aplicatiei

Din punct de vedere al modului de programare, in acest proiect am impartit atat functionalitatea serverului cat si pe cea a clientului in mai multe subprograme ce desevrvesc diferite roluri(citirea din fisiere, transmiterea informatiilor intre client si server etc.), in acest mod imi voi asigura un mai bun workflow si totodata este o metoda buna pentru a organiza proiectul in mai multe subsectiuni, structurand si eficientizand dezvoltarea aplicatiei. Am considerat faptul ca acest proiect nu detine o complexitate suficient de ampla incat sa fie nevoie de a utiliza o baza de date si de aceea am ales sa imi stochez toate datele cu privire la aplicatie in simple fisiere .txt, asa cum am mentionat si in sectiunea anterioara.

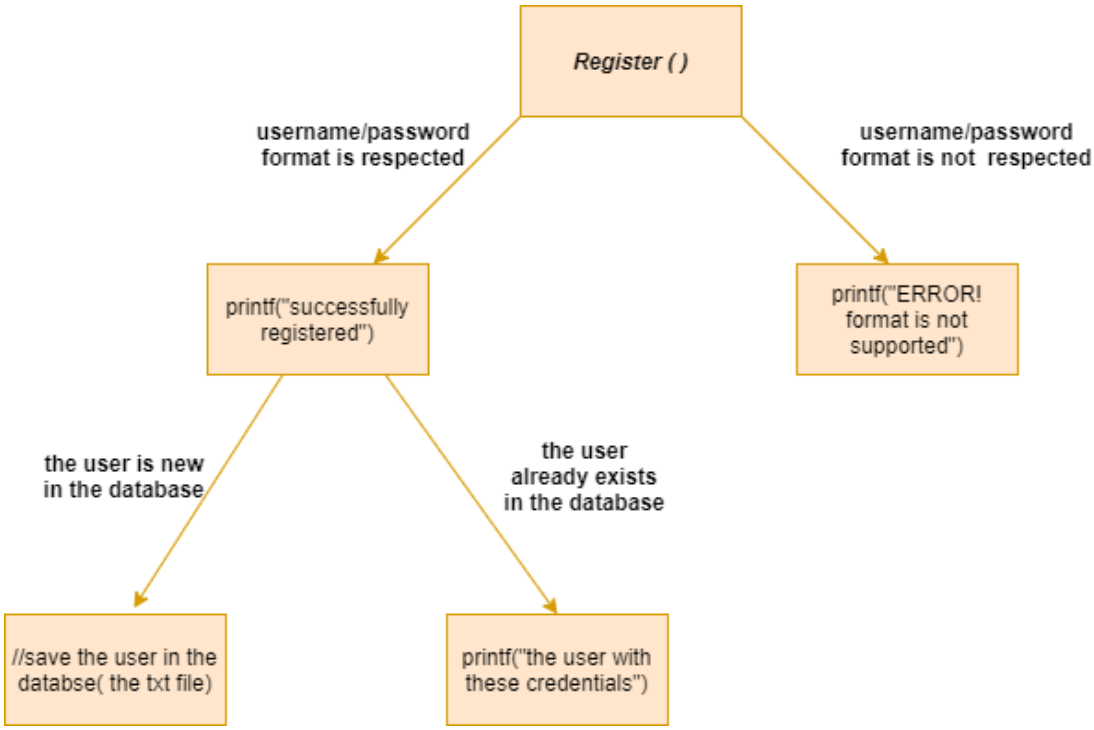

Fig. 1: Diagrama pentru inregistrare - Register ( )

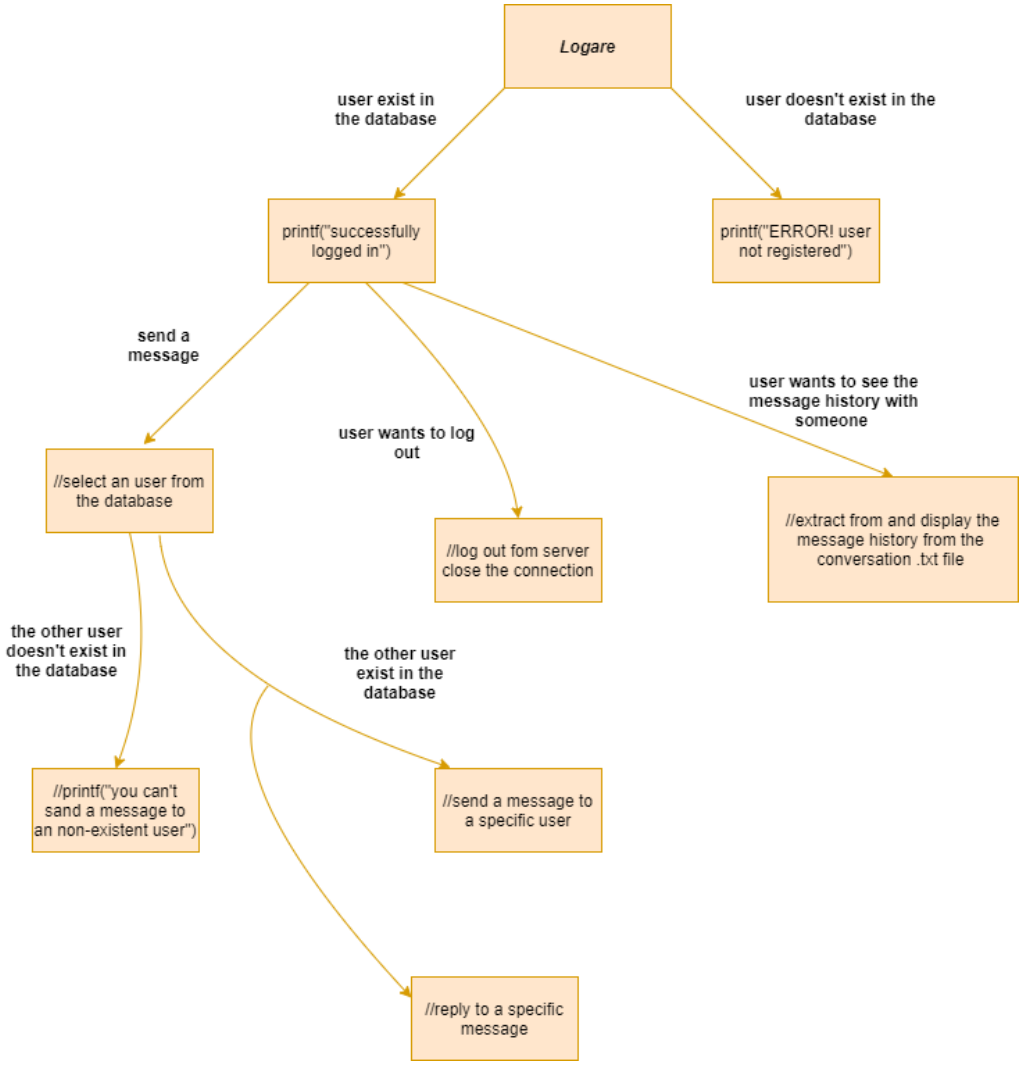

Fig. 2: Diagrama pentru logare -LogIn ( )

4 Detalii de Implementare

Implementarea TCP-ului se va realiza concurent, folosind socket-uri pentru a comunica intre procese. Se va utiliza primitva fork ( ) pentru a crea cate un proces separat ce va gestiona fiecare client.

```
int serverListen(int socketDescriptor)
{

    int listened = listen(socketDescriptor, 5);
    if (listened == -1)
    {
        cout << "[SERVER] Eroare la listen " << endl
             << endl;
        return -3;
    }

    cout << endl
         << endl;
    cout << "**********ASTEPTAM CLIENTI SA SE CONECTEZE LA PORTUL " << PORT << "**********" << endl
         << endl;

    while (1)
    {
        //conectarea cu clientul
        int clientSocketDescriptor = accept(socketDescriptor, NULL, NULL);

        if (clientSocketDescriptor < 0)
        {
            cout << "[SERVER] Eroare la lconectarea la client" << endl
                 << endl;
            continue;
        }

        cout << "**********CONEXIUNE REALIZATA CU CLIENTUL, SE ASTEAPTA COMANDA**********" << endl
             << endl;

        int child = fork();
        if (child == 0)
        {
            int command;
            recieveRSP(clientSocketDescriptor, command); //citim comanda

            cout << "[SERVER] Comanda receptionat de la client: " << command << endl
                 << endl;

            //prelucrarea comenzii
            char response[100];
            executeCommand(command, clientSocketDescriptor, response);
            cout << "[SERVER] Se trimite raspunsul catre server ... " << endl
                 << endl;

            //trimitem raspunsul catre server
            int written = write(clientSocketDescriptor, response, 100);
            if (written <= 0)
            {
                cout << "[SERVER] Eroare la trimiterea raspunsului catre client" << endl
                     << endl;
                return -4;
            }
            else
            {
                cout << response << endl;
                cout << "[SERVER] S-a trimis raspunsul catre server" << endl
                     << endl;
            }

            close(clientSocketDescriptor); //inchidem clientul
        }
    }
}
```

In prima instanta un user fie se va inregistra, fie va incerca sa se conecteze la server. Daca datele de conectare sunt corecte se va afisa mesajul corespunzator si utilizatorului i se va permite o serie de actiuni: trimiterea de mesaje, reply-ul la un anumit mesaj sau vizualizarea istoricului de mesaje cu un anumit utlizator. In orice moment un user poate da log-out. Chiar daca un utilizator este delogat acesta poate inca primi mesaje de la ceilalti useri din sistem, iar acesta le va putea vizualiza cand se va loga din nou.

5 Concluzii

Tinand cont de implementarea actuala a aplicatiei putem spune ca ar mai fi cateva imbunatatiri asupra carora ne-am putea concentra pe viitor: o interfata grafica ce ar imbunatati user-experience-ul, un alt mod de a stoca datele cum ar fi fisiere XML sau JSON, utilizarea threadurilor in loc de fork, etc.

6 Bibliografie

- draw.io
- https://stackoverflow.com
- https://www.springer.com/gp/computer-science/lncs/new-latex-templates-available/15634678
- https://profs.info.uaic.ro/ eonica/rc/
- https://profs.info.uaic.ro/ matei.madalin/retele/laboratory-1