Offline Messenger

Ionela Său
Anul 2 Grupa A3
Facultatea de Informatica Iasi

December 2022

1 Introducere

Acest document prezinta aplicatia Offline Messenger, care permite schimbul de mesaje intre utilizatorii conectati si ofera posibilitatea trimiterii mesajelor si catre utilizatorii offline. Celor offline le vor aparea mesajele cand se conecteaza la server. Utilizatorii pot trimite un raspuns (reply) in mod specific la mesajele primite prin intermediul unui id. Aplicatia va oferi totodata istoricul conversatiilor pentru fiecare utilizator, atat pentru el insusi cat si pentru conversatiile cu ceilalti utilizatori. Punctul forte al acestei aplicatii este minimul de resurse necesare pentru a se putea realiza comunicarea intre doi sau mai multi utilizatori.

Am ales aceast proiect deoarece consider ca in zilele noastre comunicarea de la distanta a devenit indispensabila si este folosita de din ce in ce mai multa lume. Drept urmare, aceasta aplicatie este una utila societatii si consider ca este foarte interesant de implementat, pentru a vedea cum functioneaza de fapt o mare parte din aplicatiile de comunicare din ziua de azi.

2 Tehnologii utilizate

Aplicatia are la baza modelul client-server. Modelul client server este un mod de structurare al unui program care ajuta la transmiterea de informatii intre un server (provider) si un client (requester). Rolul serverului este de a se fixa (bind) pe un port specific (port folosit de client pentru a localiza serverul) si sa asculte noi conexiuni. Spre deosebire de client care este temporar, serverul trebuie sa ruleze incontinuu, chiar daca nu exista clienti conectati la el.

Fiindca aplicatia dezvoltata are nevoie de garantia ca mesajele trimise ajung la destinatar fara a se pierde informatie si in ordinea corespunzatoare, am ales sa folosesc protocolul TCP/IP. Aceesta are rolul de a controla transferul de informatii astfel incat acesta sa fie unul de incredere. In cadrul intregului internet, informatiile sunt transmise in pachete. Pachetele sunt bucati de date transmise

individual si sunt reasamblate cand ajung la destinatie, pentru a reveni la forma originala.

Stiva de protocoale de comunicare TCP/IP este utilizata pentru a interconecta la distanta (prin intermediul internetului) dispozitive diferite. Cu toate ca ea este folosita in general pentru a conecta dispozitive remote, stiva poate fi utilizata si intr-o retea privata.

Programul functioneaza pe principiul de multithreading. Avantajul principal, in comparatie cu crearea de noi procese prin fork(), este viteza de executie sporita deoarece se pot executa mai multe thread-uri in acelasi timp.

3 Arhitectura aplicatiei

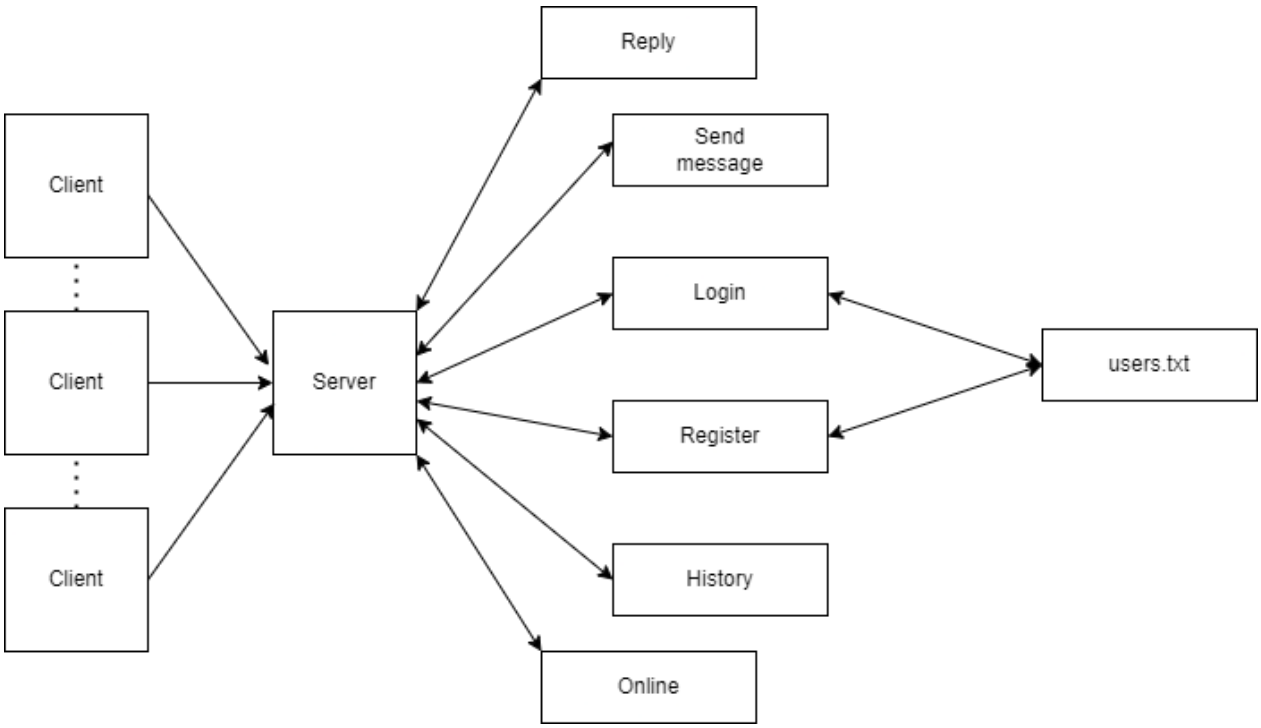

Figure 1: Diagrama

4 Detalii de implementare

Odata cu conectarea la server clientul este intampinat de 3 comenzi principale: <Login>, <Register> si <Exit>. Pentru a putea accesa restul comenzilor, utilizatorul trebuie mai intai sa se inregistreze folosind <Login> daca deja are un cont existent sau <Register> pentru a-si creea cont nou. Dupa acest pas vor fi disponibile toate comenzile: <Lista utilizatori>, <Lista utilizatori online>, <Trimiteti mesaj>, <Reply>, <Vizualizati mesajele primite>, <Vizualizati istoricul mesajelor>, <Logout> si <Exit>. Fiecare comanda este prezentata in randurile ce urmeaza:

Login: Utilizatorul trebuie sa introduca un email si o parola, care vor fi verificate prin intermediul fisierului "users.txt"; daca acestea corespund unuia dintre conturilor existente, clientul va fi intampinat cu un mesaj de conectare si ii vor fi prezentate comenzile disponibile, alaturi de numarul de mesaje primite cat a fost offline. In caz contrar ii va fi adus la cunostinta ca datele de conectare sunt gresite. De asemenea, daca utilizatorul este deja conectat, fie pe acelasi terminal, fie pe altul, va fi avertizat.

Register: Utilizatorul trebuie sa introduca o adresa de email si o parola pentru a-si putea creea contul. Acestea vor fi adaugate in fisierul "users.txt" astfel incat urmatoarea data se va putea conecta accesand comanda <Login>.

Lista utilizatori: Aceasta comanda afiseaza userii existenti, astfel incat sa stii cui poti sa-i trimiti mesaj sau cu cine poti vizualiza istoricul mesajelor.

Lista utilizatori online: Aceasta comanda afiseaza userii conectati in momentul actual la server.

Trimiteti mesaj: Cu ajutorul acestei comenzi un user poate trimite mesaj catre altul folosindu-se de username-ul acestuia. Serverul va solicita username-ul si mesajul care se doreste a fi trimis. In cazul in care username-ul nu exista sau user-ul incearca sa-si trimita mesaj singur, va fi avertizat. Toate mesajele trimise si primite sunt stocate in fisiere in formatul .txt, astefel avem fisiere de tipul: "user_sent.txt", "user_received.txt" si "user1-user2.txt".

Reply: Aceasta comanda poate fi folosita pentru a raspunde la un mesaj primit, prin intermediul unui id. Fiecare mesaj receptionat se identifica printr-un id, iar cand utilizatorul foloseste comanda <Reply> tot ce trebuie sa faca e sa introduca id-ul mesajului la care vrea sa trimita reply si textul pe care vrea sa-l trimita. In rest, aceasta este similara comenzii <Trimiteti mesaj>.

Vizualizati mesajele primite: Aceasta comanda afiseaza mesajele primite de catre user-ul conectat, iar fiecarui mesaj ii este atribuit un id (pentru a se putea da reply). In cazul in care nu exista mesaje primite se va afisa mesajul "Nu exista mesaje primite". Daca user-ul a vizualizat mesajele primite, in momentul

in care se delogheaza mesajele primite se vor sterge.

Vizualizati istoricul mesajelor: La aceasta comanda, utilizatorul este rugat sa introduca numele persoanei cu care vrea sa vada istoricul, sau "eu" daca vrea sa vada doar mesajele trimise de catre el. La fel ca pana acum, daca user-ul pe care il introduce nu exista, va primi o comanda de eroare.

Logout: Aceasta comanda va deloga utilizatorul si ii va afisa meniul de la inceput, cu comenzile <Login>, <Register> si <Exit>.

Exit: La executarea acestei comenzi, utilizatorul va fi deconectat complet de la server.

5 Cod relevant

In imaginea de mai jos se pot oberva cateva functii indispensabile programului:

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

int opt=1;
setsockopt(sd, SOL_SOCKET, SO_REUSEADDR, &opt, sizeof(opt)); // utilizarea optiunii SO_REUSEADDR

bzero(&server,sizeof(server));
bzero(&from,sizeof(from));

server.sin_family=AF_INET;
server.sin_addr.s_addr=htonl(INADDR_ANY);
server.sin_port=htons(PORT);

if(bind(sd, (struct sockaddr *)&server, sizeof(server))==-1) // se ataseaza socketul
{
    perror("[server] Eroare la bind(1).\n");
    return errno;
}

if(listen(sd, 10)==-1)  // se asculta daca vin clienti sa se conecteze
{
    perror("[server] Eroare la listen(1).\n");
    return errno;
}
```

Figure 2: Pregatire socket in server.c

Functia socket(): Aceasta creeaza socket-ul. Ea primeste 3 argumente: domeniul de adresa al socket-ului (in cazul nostru AF_INET), tipul socket-ului (SOCK_STREAM deoarece folosim TCP) si protocolul (0).

Functia bind(): Aceasta asociaza o adresa IP si un port socket-ului creat anterior. Ea are tot 3 argumente: descriptorul de fisier al socket-ului, un pointer la struct sockaddr si lungimea lui struct sockaddr.

Functia listen(): Prin intermediul ei serverul este pregatit sa accepte conexiuni de la clienti, pe care le va pune intr-o coada de asteptare. Tot aceasta stabileste numarul maxim de conexiuni acceptate, in cazul nostru 10.

Functia accept(): Ia prima cerere de conexiune din coada si o accepta; va returna un nou desciptor care va fi folosit pentru comunicarea cu clientul respectiv.

```c
int login_function(char email[], char parola[])
{
    int bytes;
    FILE *file=fopen("users.txt","r");
    if(file == NULL)
    {
        perror("[server] Eroare la fopen(2).\n");
        return errno;
    }
    else
    {
        char line[200];
        while(fgets(line,sizeof(line),file) != NULL)
        {
            char *e=strtok(line," ");
            char *p=strtok(NULL,"\n");
            if((strcmp(email,e)==0) && (strcmp(parola,p)==0))
                return 1;
        }
    }
    fclose(file);
    return 0;
}
```

Figure 3: Login

In functia "login_function" se trimit ca paramentrii email-ul si parola iar apoi acestia sunt cautati in fisierul "users.txt". Daca contul asociat exista, functia returneaza 1, altfel returneaza 0 (insemnand ca logarea a esuat).

```
int online_users_function()
{
    char online_users[500];
    bzero(online_users,500);
    strcat(online_users,"\n");
    int i;
    for(i=0; i<10; i++)
        if(clients[i]!=NULL)
        {
            strcat(online_users, clients[i]->username);
            strcat(online_users, "\n");
        }
    strcat(online_users,"\n");
    strcpy(mesaj_pentru_client,online_users);
    return 1;
}
```

Figure 4: Online users

In functia "online_users_function" se trece cu ajutorul unui for prin toti clientii (maxim 10), iar pentru fiecare client conectat (clients[i]!=NULL), i se va copia username-ul in sirul online_users, care mai apoi va fi copiat in mesaj_pentru_client.

6 Concluzii

Aplicatia ar putea fi imbunatatita prin adaugarea unei baze de date unde sa stocam conturile, in loc de un fisier de tipul .txt. De asemenea, ar mai putea fi adaugate comenzi precum stergerea mesajelor, posibilitatea de a adauga alti utilizatori la o lista de prieteni, respectiv stergerea lor din lista de prieteni, posibilitatea de a trimite un mesaj catre mai multe persoane odata si implementarea unei interfate grafice prietenoase.

References

[1] https://profs.info.uaic.ro/~ioana.bogdan/

[2] https://profs.info.uaic.ro/~computernetworks/

[3] https://www.fortinet.com/resources/cyberglossary/tcp-ip

[4] https://www.geeksforgeeks.org/tcp-server-client-implementation-in-c/