Offline Messenger (B)

Nicșan Raluca-Maria

Facultatea de Informatică, Universitatea "Alexandru Ioan Cuza", Strada General Berthelot, nr. 16, Oras IAŞI, Cod Postal 700483, ROMÂNIA
secretariat@info.uaic.ro
https://www.info.uaic.ro/

Abstract. O aplicație client/server care permite schimbul de mesaje între utilizatori care sunt conectați și oferă funcționalitatea trimiterii mesajelor și către utilizatorii offline, acestora din urmă apărându-le mesajele atunci când se vor conecta la server. De asemenea, utilizatorii au posibilitatea de a trimite un răspuns (reply) în mod specific la anumite mesaje primite. Aplicația oferă și istoricul conversațiilor pentru și cu fiecare utilizator în parte.

Introducere

Offline Messenger este o aplicație de chat între utilizatorii conectați în mod sincron sau asincron la același server.

Am ales acest proiect, întrucât aplicațiile de mesagerie se folosesc zilnic, neau făcut viața mai ușoară de atâția ani și m-am gândit că este foarte interesant să încerc să îmi creez o proprie astfel de aplicație, înțelegând astfel în timpul trecerii prin întregul proces de creare, o parte dintre conceptele funcționalității unei astfel de aplicații, făcând deseori o paralelă cu cazurile de utilizare din viața reală.

Tehnologiile utilizate

În cadrul proiectului am ales să folosesc un protocol care are control la nivelul transmisiei, întrucât oferă o serie de garanții în timpul comunicării. În contextul comunicării între doi clienți este important ca mesajele trimise să ajungă la destinatar în mod corect. Dacă informația transmisă nu ar ajunge în ordine sau ar avea erori, atunci în cadrul conversației nu ar mai exista o logică a mesajelor, firul replicilor crescând difcultatea de înțelegere și răspuns, crescând și posibilitatea de a percepe informații eronate. Astfel, alegerea comunicării prin intermediul unei conexiuni directe în contextul TCP-ului, se pliază în cadrul nevoilor la nivelul aplicației.

Totodată, caracterul concurent al serverului este realizat prin multithreading. Am ales această variantă și nu cea a proceselor copil în care se creează un nou proces propriu-zis, întrucât oferă mai multă flexibilitate în cadrul programului și threadurile împart aceeași memorie cu procesul principal, lucru util în cazul variabilelor care devin astfel accesibile tuturor threadurilor.

Arhitectura aplicației

Conceptele implicate

În cadrul aplicației, interacțiunea dintre client-server este la modul comandă-confirmare/infirmare.

După conectarea clientului, serverul creează un nou fir de execuție pentru acesta, pentru a se loga clientul trebuie să introducă numele și parola.
Serverul verifică datele primite în fișierul de logare. Dacă datele sunt corecte atunci serverul trimite confirmarea, verifică apoi dacă utilizatorul a primit mesaje cât timp nu a fost logat, în fișierul temporar cu numele său, iar în caz afirmativ, trimite mesajele clientului.
Un client care trece de filtrul logării și primește confirmarea din partea serverului, primește mesajele menționate mai sus, apoi serverul îi oferă acces la un nou set de comenzi.

Clientul poate trimite mesaje căre orice alt utilizator. Serverul verificând existența acestuia și mai apoi salvând mesajul în fișierul destinat conversației dintre cei doi. În cazul în care un utilizator nu este logat, serverul creează un fisier temporar, cu mesajele primite de acesta cât timp nu a fost logat, ștergându-l în momentul în care utilizatorul se conectează.
În final, serverul îi răspunde clientului, precizându-i dacă mesajul a fost trimis iar utilizatorul este conectat, sau dacă mesajul a fost trimis și utilizatorul nu este conectat, sau dacă au existat probleme la trimiterea mesajului.

Clientul are posibilitea de a deschide conversația pe care a avut-o cu unul dintre ceilalți utilizatori, primind istoricul tuturor mesajelor. Serverul caută fișierul care conține conversația respectivă și trimite informația clientului.
În conținutul fișierului fiecare mesaj este numerotat, oferind astfel posibilitatea clientului de a da reply la oricare dintre mesaje, specificându-i serverului nr. mesajului și conținutul răspunsului. În cazul în care clientul nu are încă mesaje în cadrul conversației sau specifică un nr. incorect, serverul îi va trimite mesaje de eroare, menționând problema.

Clientul se poate întoarce la meniul anterior (cu/fără a da reply), unde în plus față de cele două comenzi menționate, acesta poate verifica noile mesaje primite din partea celorlați, sau îi poate cere serverului deconectarea.

În ultimul caz menționat, serverul actualizează lista utilizatorilor conectați, eliminându-l. Serverul păstrează informațiile despre ultizatorii conectați într-un tablou de elemente de tip structură, în care este stocat numele și adresa utilizatorului, informații utilizate pentru trimiterea și primirea mesajelor.

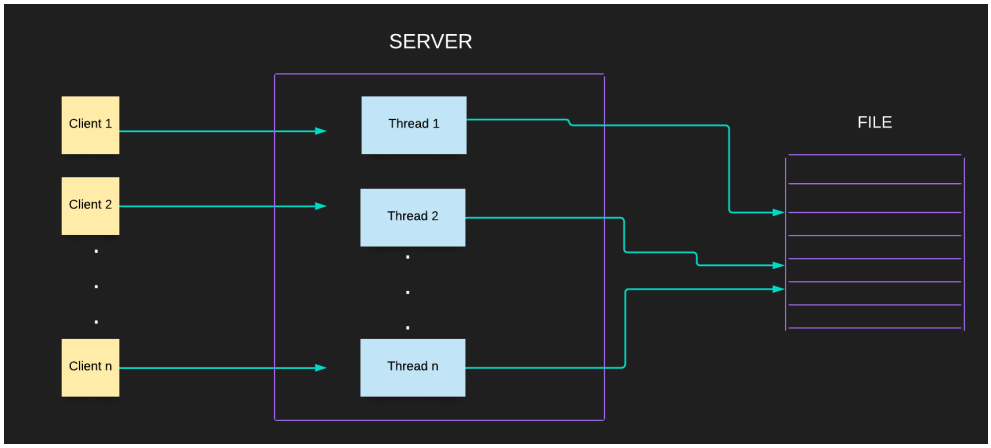

Fig. 1. Diagrama aplicatiei

Detalii de implementare

Cod relevant

funcție apelată dacă datele de logare sunt corecte:

```
int logare(char comanda[100],int cl)
{
    strcpy(comanda,comanda+6);
    comanda[strlen(comanda)-1]='\0';
     for(int i=0;i<nr_clienti;i++)
          if(strcmp(client_date[i].nume,comanda)==0)
              return 0;
    strcpy(client_date[nr_clienti].nume,comanda);
    client_date[nr_clienti].adresa=cl;
    nr_clienti++;
    printf("%s\n",comanda);
    for(int i=0;i<nr_clienti;i++)
    printf("-> %s <-    >> %d <<\n",client_date[i].nume,client_date[i].adresa);

    return 1;

}
void write_to(char comanda[100],char mesaj[100],int client_curent)
{
if(user_corect(comanda)){
    for(int i=0;i<nr_clienti;i++)
    {
```

```
    if (strcmp(client_date[i].nume, comanda) == 0)
        aici = client_date[i].adresa;
    if(client_date[i].adresa==client_curent)
        strcpy(nume_client_curent, client_date[i].nume);
}
    if(strcmp(comanda,nume_client_curent)>0){
        strcat(nume_fisier,comanda);
    strcat(nume_fisier,nume_client_curent);
    }
    else{
        strcat(nume_fisier,nume_client_curent);
        strcat(nume_fisier,comanda);
    }
    strcat(mesaj_complet, nume_client_curent);
    strcat(mesaj_complet, "~");
    strcat(mesaj_complet, mesaj);

     strcat(nume_tempo2,comanda);
     strcat(nume_tempo2,"_tempo");
    if(aici==-1)
    {
        if((fisi_tempo2=fopen(nume_tempo2,"a+"))==NULL)
        {
            printf("unable to create file\n");
            exit(EXIT_FAILURE);
        }
        fputs(mesaj_complet,fisi_tempo2);
        fclose(fisi_tempo2);
    }
    if((fisi_tempo=fopen(comanda,"a+"))==NULL)
    {
        printf("unable to create file\n");
        exit(EXIT_FAILURE);
    }
    fputs(mesaj_complet,fisi_tempo);
    fclose(fisi_tempo);
    {
        if((fisi=fopen(nume_fisier,"a+"))==NULL)
        {
            printf("unable to create file\n");
            exit(EXIT_FAILURE);
        }
        printf("[Thread %s]Mesajul a fost trasmis cu succes:\n", mesaj);

        for(c=getc(fisi);c!=EOF;c=getc(fisi))
```

```
                if(c=='~')
                    count++;
            count++;
            printf("%d\n",count);
             char asta_e_ultimul[150]="";
             char a[10]="";
             while(count!=0)
             {     a[strlen(a)]=(count%10)+48;
                  printf("%c\n",a[strlen(a)]);
                  strcat(asta_e_ultimul,a);
                  count=count/10;
             }
             strcat(asta_e_ultimul,") ");
             strcat(asta_e_ultimul,mesaj_complet);
            fputs(asta_e_ultimul,fisi);
             fclose(fisi);

            if((fisi=fopen(nume_fisier,"a+"))==NULL)
            {
                printf("unable to create file\n");
                exit(EXIT_FAILURE);
            }
             fgets(haida,100,fisi);
            fclose(fisi);
        }
        if (write(client_curent, "mesagge sent succefuly!! ", 100) <= 0) {
            perror("[Thread]Eroare la write() catre client.\n");
        }
}
```

Scenariu de utilizare

Clientul 1 se conectează la server, care creează un thread pentru el.
-se autentifică-primește confirmare și setul de comenzi din partea serverului, care actualizează lista cu clienții autentificați.
-inițiază o nouă conversație cu clientul 2, și îi trimite un mesaj
-serverul verifică dacă există în fișierul de logare:

$->$ dacă nu există: trimite mesaj de eroare și cauza

$->$ altfel, preia mesajul, îl salvează în fișierul cu conversația celor doi, creează fișierul temporar pentru clientul 2 și trimite confirmarea că mesajul s-a trimis clientului dar că acesta nu este logat.

Se conectează clientul 2-serverul creează un nou thread
- când se realizează autentificarea, serverul îi trimite mesajele primite când el nu era logat și setul de comenzi
- îi răspunde la mesaj clientului 1
- serverul salvează răspunsul și actualizează fișierul

Se conectează clientul 3-serverul creează un nou thread
- face login și primește doar confirmarea, pentru că nu are alte mesaje primite.
– trimite un nou mesaj clientului 1
– severul îi trimite confirmarea
Clientul 3 se deconectează

Clientul 1 verifică ultimele mesaje și serverul îi trimite mesajele primite de la cei doi.
– intră în conversația cu clientul 3 și îi dă reply
– serverul preia nr. mesajului de reply și mesajul și actualizează fișierul cu conversația lor, salvând mesajul și trimite confirmarea clientului 1, dacă nr este unul greșit, trimite mesaj de eroare și specifică motivul.
Clientul 1 iese din conversație și după se deconectează.
Serverul îl șterge din lista de ultizatori conectați și îi trimite confirmarea de deconectare.

Clientul 2 verifică ultimele mesaje
- serverul îi răspunde că nu are încă niciun mesaj primit.
clientul 2 se deconectează -serverul confirmă și îl șterge din listă.

Concluzii

În cele ce urmează, urmăresc a implementa în cadrul aplicației posibilitatea clientului de a crea un nou cont în cadrul aplicației, și simularea unui cronometru în server prin intermediul căruia, acesta va trimite pachete de mesaje, regulate clienților.

References

1. Computer Networks Computer Networks, http://www.info.uaic.ro/ computernetworks.