Documentație QuizzGame

Galatanu Emilia

December 2023

1 Introducere

Viziunea proiectului este de a crea un joc în care un server gestionează mai mulți clienți care vor răspunde la întrebările adresate de server. Clienții vor avea un timp în care se vor loga, iar după acel timp serverul va începe jocul. Serverul trimite o întrebare către clienți, iar aceștia vor avea un timp de câteva secunde pentru a răspunde la întrebare. Jocul se termină atunci când se termină setul de întrebări sau toți clienții s-au retras din joc.

Obiectivele în realizarea acestui proiect:

- anunțarea câștigătorului
- calcularea punctajelor pentru fiecare client
- sincronizarea clienților
- gestionarea jocului în cazul în care un client părăsește jocul
- desfășurarea jocului(trimiterea întrebării, primirea răspunsului, și validarea acestuia)
- creearea unui fișier XML în care vor fi întrebări, variante de răspuns, dar și varianta corectă la întrebare;
- înregistrarea clientior pentru a participa la joc;

2 Tehnologii Aplicate

Pentru a implementa acest proiect am folosit protocolul TCP pentru comunicarea între client și server. Am selectat acest protocol deoarece asigură o conexiune persistentă și datele sunt livrate fără pierderi și în ordine, ceea ce este necesar într-un astfel de joc pentru a menține coerenta.

Am ales limbajul de programare C++ datorită caracteristicilor sale de performanță, flexibilitate și capacitatea de a gestiona eficient resursele sistemului.Pot implementa structuri de date mai complexe ceea ce este esențial într-o astfel de aplicație unde gestionarea rapidă a comunicării este crucială.

Am ales să transmit datele între clienți și server printr-un socket deoarece datele sunt transmise mai eficient și este posibilă și transmiterea bidirecțională a acestora.

Am folosit un fișier XML pentru înregistrarea întrebărilor, opțiunilor și răspunsurilor corecte, deoarece oferă o structura ierarhică mult mai ușor de gestionat.

Am realizat o mini interfata grafica care doar ilustreaza camera jocului, serverul, intrebarile trimise, dar si cati clienti sunt conectati.

Am folosit mutex-uri pentru sincronizarea thread-urilor.

3 Structura Aplicatiei

Aplicația este structurată pe mai multe module interconectate:

Server:

- Gestionarea conexiunilor: acceptă clienți și îi adaugă în lista de clienți activi;
- Distribuirea întrebărilor si a variantelor de răspuns din fisierul XML;
- Verificarea răspunsurilor:După ce un client trimite un răspuns, serverul îl verifică și acordă sau nu puncte, în funcție de corectitudinea răspunsului
- Actualizarea punctajelor: Serverul gestionează punctajele clienților și le actualizează pe măsură ce aceștia răspund la întrebări.

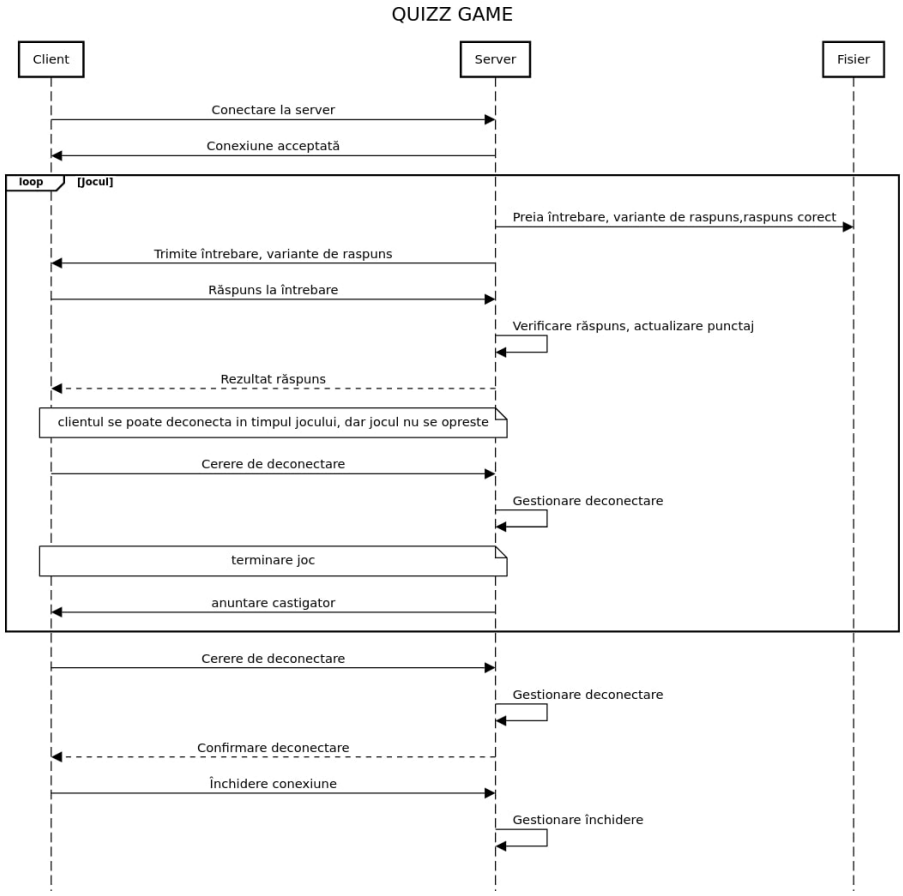
Client:

- Primirea intrebarilor si a optiunilor de raspuns de la server
- Transmiterea răspunsurilor alese înapoi la server

Fisier XML:

- Stochează întrebările
- Optiunile de raspuns
- Răspunsurile corecte

Diagrama implementării jocului.



4 Aspecte de implementare

Am definit o structură pentru a reține informații despre threadu-rile create , dar și o clasa Client pentru a seta informațiile unui client.

```
using namespace std;
#define PORT 1112
struct thData{
    int idThread;
    int cl;
    string raspuns;
};
class Client{
    public:
    Client():id(0),score(0),active(true),nume("da"){}
    Client(int id, string nume,bool active):id(id),score(0),active(true),nume(nume){}
    int getId(){return id;}
    int getScore(){return score;}
    int getActive(){return active;}
    void print(){
        cout<<id<<" "<<score<<" "<<" "<<active<<" "<<nume;
    }
    string getName(){return nume;}
    void setInactive(){active=false;}
    private:
    int id;
    int score;
    string nume;
    bool active;
};
```

Am creat câte un fir de execuție și l-am adăugat în vectorul firelor de execuție.

```
while(gamestate==true){
try{
if((client=accept(socketServer,reinterpret_cast<sockaddr*>( &from),&length))==-1)
    throw(25);
}
catch(...)
{
    cout<<"Eroare la accept";
    break;
}


data = new thData();
data->idThread = i++;
data->cl=client;

threads.push_back(thread(raspunde,data));
```

Am trimis serverului în aplicația client comanda citită la tastatură.

```
cout<<"Client: Introduceti o comanda: ";
fflush(stdout);
cin.getline(buf,sizeof(buf));
buf[sizeof(buf)-1]='\0';
cout<<"\n";
cout<<"Am citit comanda:"<<buf<<'\n';
fflush(stdout);
try{
if(send(socketServer,buf,sizeof(buf),0) <= 0)
    throw(22);
}
catch(...)
{cout<<"Eroare la write spre server\n";
exit(1);}
```

Serverul primește comanda.

```
try{
    if(recv(data->cl, comanda, sizeof(comanda),0) <= 0)
        throw(22);
    }
    catch(...)
    {cout<<"[Thread " << data->idThread << "]\n"<<"Eroare la read de la client\n";
    exit(1);}
    int i,j;
```

Dacă serverul primește comanda login prelucrează comanda astfel încât să preia numele de utilizator și să adauge clientul în map-ul pentru clienți.

```
comanda[strlen(comanda)]='\0';
if(strstr(comanda,"login")!=nullptr)
    { for(i=0;i<strlen(comanda);i++)
            if(comanda[i]=='l'&&comanda[i+1]=='o'&&comanda[i+2]=='g'&&comanda[i+3]=='i'&&comanda[i+4]=='n')
                break;
            i=i+5;
        while(isspace(comanda[i])||(!isalpha(comanda[i])&&!isdigit(comanda[i])))
        i++;
        while(isalpha(comanda[i])||isdigit(comanda[i]))
            {utilizator[k]=comanda[i];k++;i++;}
        utilizator[k]='\0';
```

Dacă serverul primește comanda logout serverul trimite un mesaj de confirmare, iar clientul își termină execuția.

```
    if(strstr(comanda,"logout"))
    {char mesaj[256];
    strcpy(mesaj,"V-ati delogat cu succes ");
    strcat(mesaj,"\n");
    try{
 if(send(data->cl,mesaj,sizeof(mesaj),0) <= 0)
    throw(22);
 }
 catch(...)
 {cout<<"Eroare la write spre client\n";
 exit(1);}
```

```
if(strstr(buf, "logout")!=nullptr)
    exit(0);
buf[0]='\0';
mesaj[0]='\0';
```

Pentru orice comandă clientul preia mesajul de la server și îl afișează.

```
char mesaj[256];
try{
if(recv(socketServer, mesaj, sizeof(mesaj),0) <= 0)
    throw(22);
}
catch(...)
    {cout<<"Eroare la read de la server\n";
exit(1);}
cout<<mesaj;
```

5 Concluzii

Am implementat un server multithreading capabil să gestioneze simultan oricâte conexiuni de la clienți, cu o mini interfata grafica care este doar pentru nivel ilustrativ.
Proiectul poate avea îmbunătățirea de a avea selectarea unui domeniu de joc.

6 Referințe Bibliografice

https://profs.info.uaic.ro/ computernetworks/cursullaboratorul.php
https://www.geeksforgeeks.org/multithreading-in-cpp/
-cursurile materiei OOP din anul 1 semestrul 2.
https://en.cppreference.com/w/cpp/thread/mutex