Quiz Game

Bodnariu Daniel-Iustin

Facultatea de Informatică, Universitatea Alexandru Ioan-Cuza Iaşi
secretariat@info.uaic.ro
https://www.info.uaic.ro
bodnariu.daniel@info.uaic.ro

Abstract. Proiectul Quiz Game constă în crearea unei aplicaţii care să simuleze un joc de tip MCQ(multiple choice-objective response), alegere multiplă-răspuns obiectiv. Această formă de evaluare este bazată pe alegerea unui singur răspuns dintr-o listă de răspunsuri. Acest format al întrebărilor este foarte utilizat în domenii precum: educaţie, cercetarea pieţei chiar şi în alegerile electorale. În cadrul acestei aplicaţii comunicarea se va realiza între două tipuri de entităţi: serverul şi clientul(utilizator). Jocul va consta în trimiterea unor întrebări ,de către server, tuturor clieţilor conectaţi. La sfârşitul jocului serverul va trimite un mesaj în care va anunţa câştigătorul, la fiecare trei clienţi conectaţi.

Keywords: TCP· server · client· socket · threading · interfaţă · bază de date · concurenţă

1 Introducere

Aplicaţia de tipul server-client pentru Quiz Game va suporta oricât de mulţi clienţi prin intermediul multhithreading-ului. Înainte de a începerea jocului fiecare client va fi înregistrat prin introducerea de către acesta a username-ului. Serverul va sincroniza clienţi înregistraţi şi la fiecare trei clienţi conectaţi va anunţa câstigătorul. După ce fiecare client din cei trei a rspuns la întrebări,vor aştepta anunţarea câstigătorului după care conexiunea cu serverul se va termina. Întrebările vor fi stocate într-o bază de date sub forma de fişier text şi preluate de acolo spre a fi trimise către clienţi. Dacă un client se va deconecta în timpul jocului acesta va fi eliminat din rundele de întrebări iar jocul va continua pentru clienţi rămaşi în joc.

2 Tehnologii utilizate

2.1 TCP

TCP este un protocol de comunicare orientat conexiune care oferă posibiliatea de a realiza comunicaîi full duplex sigure. Protocolul TCP se bazează pe protocolul IP. Acest protocol ne asigură conectivitatea între sursă şi destinaţie iar TCP-ul va transmite informaţiile. TCP/IP ca şi protocol de comunicare reprezintă un set de reguli standardizat care realizeză schimbul de informaţii între nodurile din reţea.

Avantaje ale TCP-ului

1. transmiterea informaţiei se realizează fără pierderea acesteia. Cu alte cuvinte informaţia ajunge la client în siguranţă
2. are mecanism de stabilire a conexiunii(3-way hands shaking). Acest mecanism ajută serverul să fie sigur că receptorul va primi informaţiile trensmise. Dacă conexiunea cu clientul nu se realizează atunci serverul va şti prin acest mecanism şi nu transmite informaţia.
3. protocolul TCP/IP oferă mecanism de control al fluxului de date, utilizând fereastra glisantă în acest sens, Fereastra glisantă asigură o trasnmisie sigură informaţiilor,

Un dezanvantaj al TCP-ului este că are un deplay datorită mecanismului de confirmarea de stabilire a conexiunii şi confirmare a primirii datelor. în comparaţie protocolul UDP este mai rapid însă are foarte multe dezavantaje precum: că este neorientat conexiune, deci gradul de siguranţă nu este prea îmbucurător; nu are mecanism de confirmare iar de aceea mesajul se poate pierde pe parcurs fără a mai putea fi recuperat, nu oferă control al fluxului de date şi nici al erorilor. De obicei UDP-ul este folosit mai predomininant în aplicaţii de tipul control de la distanţă(Remote Procedure Call).

2.2 Concepte la nivel de TCP

Socket Un socket este un mecanism bidirecţional ce poate fi utilizat atât pentru a comunicarea între procese(socketurile trebuie să aibă aceeaşi adresă şi acelaşi tip) de pe acelaşi calculator, dar şi pentru a asigura comunicarea în reţea. Fiecare conexiune este identificată printr-o pereche de forma adresă IP : port, unde adresa reprezintă un set de patru numere între 0 si 225 separate prin punct, iar portul este un număr între 1 şi 65535 (0-1024 fiind rezervate pentru system). Această pereche formează un socket.Un socket mai este numit şi end-point într-o reţea.Un socket este foarte similar cu un descripot de fişiere, diferenţele fiind la nivel de creare şi diferitle opţiuni de control asupra socketurilor.

Primitive Pentru realizarea conexiunii şi comunicării între un server şi un client,într-un model de server-client bazat pe socket-uri, se folosesc o serie de funcţii numite primitive.

- socket()-crează un nou socket
- bind()-ataşează adresa locală la socket
- listen()-permite unui socket să accepte conexiuni
- accept()- blochează apelantul până la sosirea unei cereri de conectare(utilizată de serverul TCP)
- connect()- este folosită de client pentru a stabili conexiunea cu serverul
- read()-citirea datelor din socket
- write()-scrierea datelor in socket
- close()-eliberează conexiunea(închide un socket).

Adresă IP.Este un set de 4 numere între 0 şi 225 separate prin punct, reprezentând o etichetă atribuită unei entităţi existente într-o reţea de calculatoare.

Port.Este un număr pe 16 biţi care identifică în mod unic procesele ce rulează pe un system de calcul. Orice aplicaţie care realizează o comunicare în reea va ataşa un port conexiunii respective. Porturile cuprinse între 0 și 1024 sunt rezervate unor procese de system. La nivelul TCP-ului se utilizează conexiuni şi nu porturi.

2.3 Alte concepte utilizate în proiect

Server. Serverul este procesul existent pe maşina care găzduieşte fişierele şi oferă seviciul înregistrare şi trmitie întrebările pentru fiecare client conectat. Mediază comunicarea între clienți și ofera servicii acestora.

Client Este partea de aplicaţie care permite conectarea unui utilizator uman la un server. Clientul este cel care iniţeaă conexiunea cu serverul şi solicită un serviciu după care aşteapă răspunsul.

Threading. Thread-urile(fire de execuţie) sunt procese create pentru a putea rula mai multe părţi ale unui program simultan, în paralel. Threading-ul în cadrul proiectului quiz game asigură multiprocesarea clienţilor şi asigură **concurenţa**. Concurenţa asigură servirea clienţilor în mod independent unii de ceilalţi şi fiecare client poate fi servit oricând. În cadrul proiectului s-au folosit **pthread-urile**, un standard ce defineşte un API(aplication programming interface) pentru crearea şi manipularea firelor de execuţie. Un dezavantaj al thread-urilor este acela cpot fi greu de manevrat când procesele ajung la race condition acest lucru fiind posibil deorece în multiprocesarea cu threaduri nu există prorprietatea de izolare(procesele lucrează cu memoria lor astfel că dacă se întâmplă ceva cu un proces celelalte nu sunt influenţate).

Interfaţă.Este partea de afișare a aplicaţiei ce realizează comunicarea efectivă cu utilizatorul conectat prin intermediul clientului,

Baza de date. O bază de date este o colecţie organizată de informaţii sau de date structurate, stocate electronic într-un computer. Pentru aplicaţia Quiz Game s-a folosit o bază de bate in forma fişier text (.txt) în care sunt stocate întrebârile şi răspunsurile ce vor trimise clientului.

3 Arhitectura aplicaţiei

3.1 Diagrama aplicaţiei

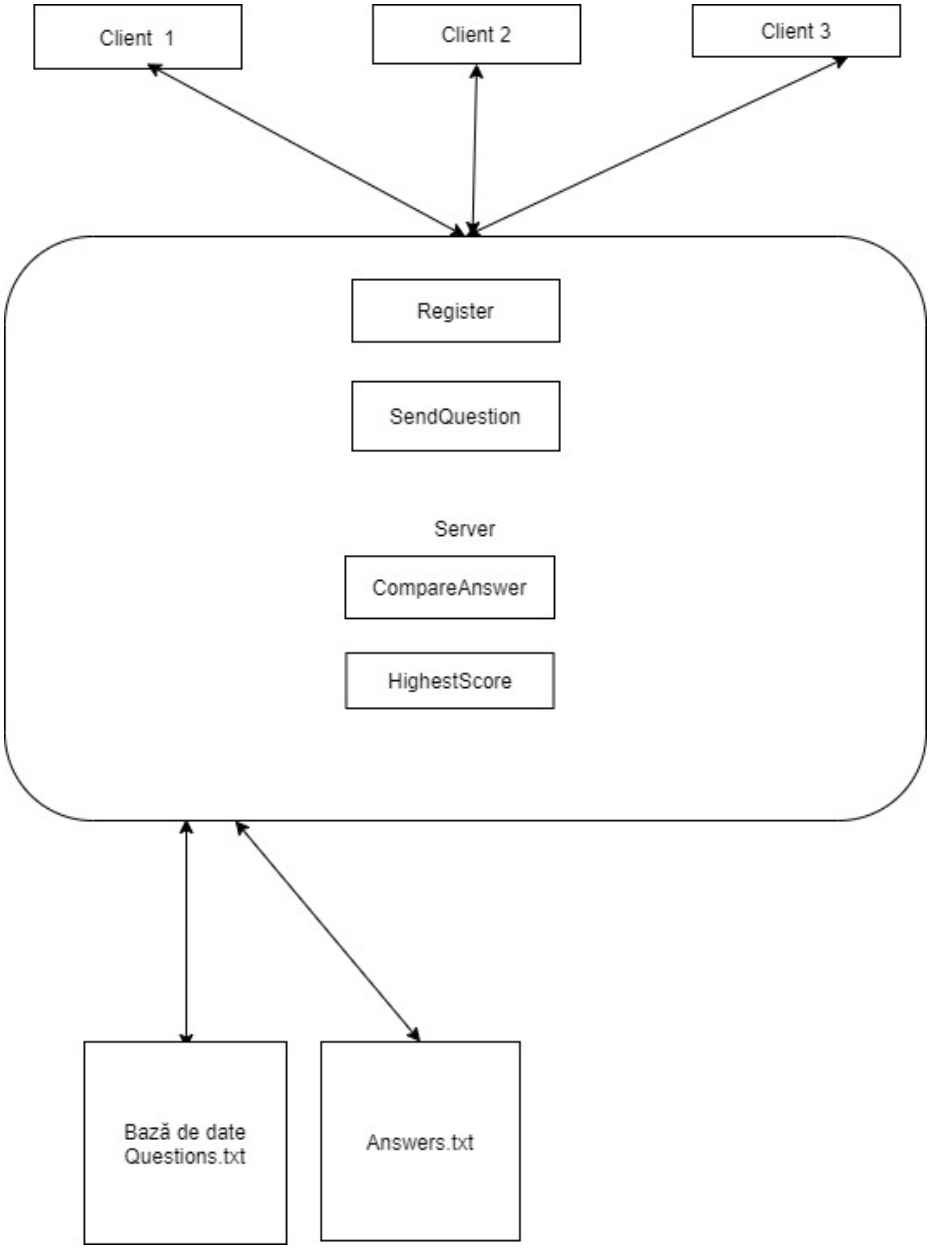

Fig. 1. Schema aplicaţiei

Diagrama user-case

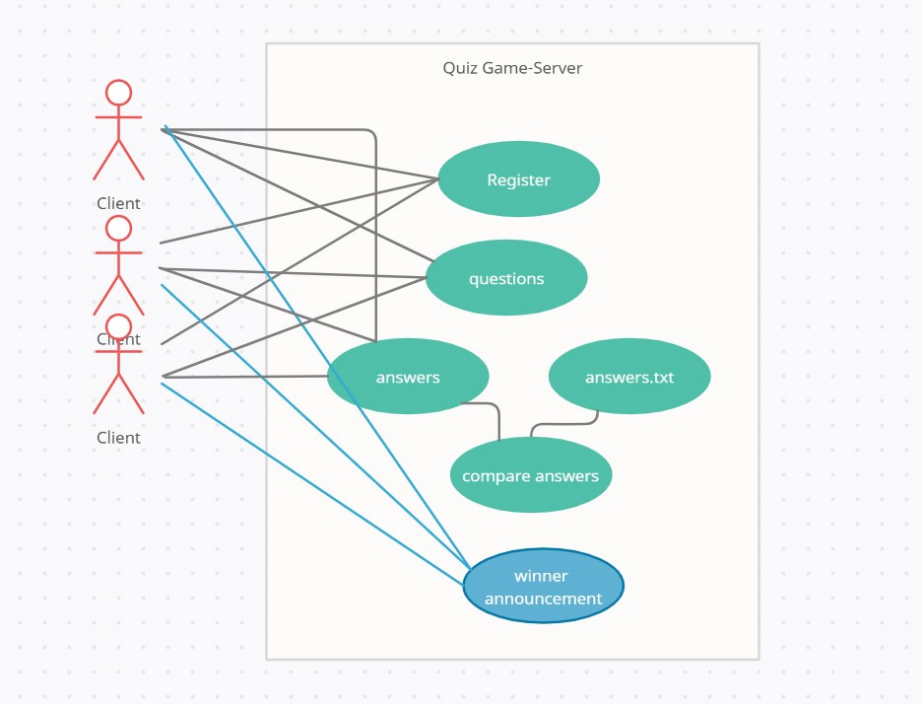

Fig. 2. Schema aplicaţiei

3.2 Concepte implicate în arhitectura aplicaţiei

Register
Când un utilizator se conectează la server pe portul şi adresa specificată, acesta va trebuie să se înregistreze introducându-şi username-ul.

SendQuestion.
Se ocupă cu trimiterea întrebărilor către clientţi.
CompareAnswer
Serverul compară răspunsurile primite de la clienţ şi
adaugă punctaj clientului în funcţie de corectitudinea răspunsului. High-Score

Parte a programului care ar trebui să returneze clientul cu cel mai mare scor.

4 Detalii implementare

```
REG.open("config_file",ios::out|ios::app);//register part
REG<<buff;
```

```
while(1){//answers compare
  cout << "Enter in ...";
  getline(FD_Server1, k);
  getline(FD1, cli);
  cout << "Key is :" << k << endl;
  cout << "client ans is :" << cli << endl;
  if(k == "#"){
        break;
        }
        else if(k == cli){
        x += 1;
        }
        k.clear();
        cli.clear();
                }
        cout << "Client score is :" << x << endl;
        cout << "Client name is :" << candidate[curr_iter] << endl;
        candidate_score = x;

if(FD_Server){
  while(1){
        getline(FD_Server, fromFile);
        cout << endl;
        cout << "Question from file :" << fromFile << endl;
        n = 0;
        // copy server message in the buffer
        while ((fromFile[n]) != '\0') {
                buff[n] = fromFile[n];
                n++;
                }
                // send buffer to client
        write(sockfd, buff, sizeof(buff));
        ................................
        cout << "Question send to client :" << buff << '\n';
        read(sockfd, buff, sizeof(buff));
        fromFile.clear();
        getline(FD_Server, fromFile);
        cout << endl;
        cout << "Choice from file :" << fromFile << '\n';
        n = 0;
        // copy server message in the buffer
        while ((fromFile[n]) != '\0') {
                buff[n] = fromFile[n];
                n++;
                }
        // send it to client
```

```
        write(sockfd, buff, sizeof(buff));
        cout << "Choice send to client :" << buff << '\n';
        if(fromFile == "#"){
                cout << "in #";
                break;
                }
        fromFile.clear();
        read(sockfd, buff, sizeof(buff));
        // print buffer which contains the client contents
   printf("From client: %s\t To client : ", buff);

void sendtoall(char * msg) {//function which will send the name of the winner
        int i;
        pthread_mutex_lock( & mutex);
        for (i = 0; i < clients_i; i++) {
                if (send(clients[i], msg, strlen(msg), 0) < 0) {
                        printf("sending failure \n");
                        continue;
                }
        }
        pthread_mutex_unlock( & mutex);
}
```

5 Concluzii

Aplicaţia Quiz Game poate fi o unealtă folositoare în testarea cunoştiinţelor generale ale unei persoane. Iar în funcţie de întrebările stocate poate constitui baza unei aplicaţii de realizare a unui studiu de cercetare etc.

5.1 Cum poate fi îmbunătăţit aplicaţia?

Aplicaţia poate fi îmbunătăţită prin introducerea unui,a unui timer pentru ca fiecare client să aibă un număr de seconde pentru a răspunde la întrebări, un meniu după ce utilizatorul se înregistrează. În acest meniu să aibă opţiunile: Start, High Score,My Answers, Quit. Pentru fiecare ultilizator înregistrat numele acestuia să fie introdus într-o bază de date iar la sfarşitul jocului să se salveze şi scorul acestuia, astfel se va putea ca oricând va fi nevoie un utilizator să poată afla cine a obţinut cel mai mare scor vreodată.

6 Bibliografie

1. Autor, Sabin Corneliu Buraga T.: Introducere în reţele de calculatoare. Editor, Facultatea de Informatică, Universitatea Alexandru Ioan Cuza .

2. Alboaie.L. Reţele de calculatoare - curs. Iaşi, România, 2020-2021, https://profs.info.uaic.ro/ computernetworks/cursullaboratorul.php.
3. Năstase R.,Protocoale,https://www.youtube.com/watch?v=v$_D$gK06JayA.
4. SQLite C tutorial .http://zetcode.com/db/sqlitec/.
5. Diagram creator:https://app.diagrams.net/G1uk$_9$Df8qjHe2uWfh0qBpI − ajmsPw8Rm4
6. Posix Threads Programming,https://computing.llnl.gov/tutorials/pthreads/