Facultatea de Informatică
Universitatea Alexandru Ioan Cuza

Proiect: QuizzGame

Constantinescu Stefania-Bianca (anul 2, grupa A2)

December 7, 2022

1 Introducere

Proiectul QuizzGame presupune implementarea unei aplicatii ce se bazeaza pe comunicarea de tip client/server. Mai multi clienti se vor inregistra, conectandu-se la server, dupa care jocul va incepe. Fiecare client va raspunde la intrebarea primita intr-un numar de n secunde. Daca un client raspunde corect, acesta va primi un punctaj, iar daca raspunde gresit sau nu raspunde in timpul alocat, punctajul va ramane neschimbat. In cazul in care un client paraseste jocul, acesta va fi eliminat din rundele viitoare cu intrebari. La sfarsit va fi anuntat un castigator catre toti clientii. [1]

2 Tehnologiile utilizate

- Am folosit protocolul de comunicare TCP (Transmission Control Protocol), un protocol orientat conexiune, deoarece TCP oferă incredere, asigura livrarea ordonata a unui flux de octeti de la un program de pe un computer la alt program de pe un alt computer aflat in retea. Pe langa sarcinile sale de gestionare a traficului, TCP controleaza marimea segmentului de date, debitul de informatie, rata la care se face schimbul de date, precum si evitarea congestionarii traficului de rețea. Comunicarea dintre server si client va folosi socket-uri, intrucat acestea sunt canale bidirectionale.
- Proiectul QuizzGame este implementat în limbajele de programare C/C++, iar pentru a gestiona concurenta proceselor se vor crea thread-uri (fire de executie) pentru fiecare client, deoarece acestea sunt mai eficiente decat procesele copil din punct de vedere al timpului de executie si al resurselor.
- Pentru crearea unei baze de date, stocarea intrebarilor si variantelor de raspuns si verificarea acestora se va folosi API-ul SQLite.

3 Arhitectura aplicației

Serverul TCP concurent permite conectarea a mai multi clienti simultan, prin functia **registerPlayer()**. De asemenea, acesta va realiza conexiunea la baza de date prin intermediul functiilor **getQuestionFromDB()** si **validateAnswer()**, va valida username-ul primit de la client si va cere reintroducerea sa in cazul in care exista deja prin functia **validateUser()**.

In plus, serverul va crea cate un thread pentru fiecare client care se inregistreaza, va porni jocul pentru jucatorul respectiv, calculeaza scorul iar mai apoi asteapta sa termine si ceilalti jucatori pentru a afisa castigatorul.

Clientul dispune de functia **connectClient()** prin care se va inregistra la quiz, alegandu-si un username (pe care serverul il valideaza) si de functia **sendAnswerToServer()** care va prelua inputul de la tastatura si il va trimite serverului. In cazul in care clientul doreste sa paraseasca jocul, acesta trebuie sa tasteze QUIT si va fi eliminat din urmatoarele runde de intrebari.

4 Detalii de implementare

Server:

- **createDB()**
  Creeaza baza de date in care vom stoca intrebarile,raspunsurile si variantele de raspuns corecte. [3]

```
char * sql="DROP TABLE IF EXISTS Questions;"
           "CREATE TABLE Questions(id INT, QUESTION TEXT, answer1 TEXT, answer2 TEXT, answer3 TEXT, answer4 TEXT, right_answer TEXT);"
           "INSERT INTO intrebari VALUES(1,'Care era numele zeului soarelui din Egiptul antic?', 'a. Ra','b. Gingo', 'c. Marko', 'd.Eferis', 'a');"
           "INSERT INTO intrebari VALUES(2,'Cati ani a durat razboiul de 100 de ani?', 'a. 100','b. 116', 'c.110', 'd. 120', 'b');"
           "INSERT INTO intrebari VALUES(3,'Care este numărul care vine după 14 în numărul pi?', 'a. 2','b. 9', 'c. 1', 'd. 3', 'c');"
           "INSERT INTO intrebari VALUES(4,'Cine este autorul Odiseei?', 'a. Ion Creanga','b. Oscar Wilde', 'c. Alexandre Dumas', 'd. Homer', 'd');"
```

- **setupServer()**
  Creeaza socket-ul TCP, face bind la acesta si asculta pentru noi conexiuni cu primitiva listen(). [2]

```
void setupServer()
{
  struct sockaddr_in server;    // structura folosita de server
  struct sockaddr_in from;
  int nr;       //mesajul primit de trimis la client
  int sd;       //descriptorul de socket
  int pid;
  pthread_t th[100];    //Identificatorii thread-urilor care se vor crea
    int i=0;

  /* crearea unui socket */
  if ((sd = socket (AF_INET, SOCK_STREAM, 0)) == -1)
    {
      perror ("[server]Eroare la socket().\n");
      return errno;
    }
  /* utilizarea optiunii SO_REUSEADDR */
  int on=1;
  setsockopt(sd,SOL_SOCKET,SO_REUSEADDR,&on,sizeof(on));

  /* pregatirea structurilor de date */
  bzero (&server, sizeof (server));
  bzero (&from, sizeof (from));

  /* umplem structura folosita de server */
  /* stabilirea familiei de socket-uri */
    server.sin_family = AF_INET;
  /* acceptam orice adresa */
    server.sin_addr.s_addr = htonl (INADDR_ANY);
  /* utilizam un port utilizator */
    server.sin_port = htons (PORT);

  /* atasam socketul */
  if (bind (sd, (struct sockaddr *) &server, sizeof (struct sockaddr)) == -1)
    {
      perror ("[server]Eroare la bind().\n");
      return errno;
    }
  /* punem serverul sa asculte daca vin clienti sa se conecteze */
  if (listen (sd, 2) == -1)
    {
      perror ("[server]Eroare la listen().\n");
      return errno;
    }
}
```

- **registerPlayer()**
  Accepta clientul si creeaza un thread nou care va porni jocul pentru utilizatorul respectiv (cu functia playGame()), dupa care va calcula scorul acestuia (functia calculateScore()) si va astepta ca si ceilalti jucatori sa termine intrebarile pentru ca mai apoi sa le transmita tuturor jucatorilor cine a castigat.

  In interiorul functiei playGame() vom apela urmatoarele functii:

- **getQuestionANDAnswerFromDB() -din fisierul in care se afla baza de date**
  Preia intrebarile si raspunsurile din baza de date.

- **sendQuestionToPlayer()**

- **readAnswerFromPlayer()**
  Serverul asteapta sa primeasca un raspuns de la oricare dintre clienti folosind primitiva select() in care ultimul parametru este timpul dat pentru a raspunde la intrebare.

```
int ok;
ok=select( /*descriptor*/ + 1, /*descriptorii de citire*/, NULL, NULL, /*timpul ales*/)
if(ok==-1)
{
  perror ("[server]Eroare la select().\n");
  return errno;
}
else if (ok == 0)
{
    /*1. tratare caz timeout + mesaj de la server catre client "THE TIME IS UP!" */

}
else
{
    /* 2.1 Tratare caz QUIT*/
    /* 2.2 Tratare caz raspuns + validare raspuns*/
}
```

- **validateAnswer()**
  Preia raspunsul corect din baza de date si il valideaza.

5 Concluzii

- Solutia propusa ar putea fi imbunatatita prin:
- Implementarea unei interfete grafice.
- Implementarea unei functii prin care sa se aleaga intrebarile intr-o ordine aleatorie, in loc de cea prestabilita.

  Solutia prezentata este una partiala, dar cu siguranta va fi completata si imbunatatita.

6 Bibliografie

[1] https://ro.wikipedia.org/wiki/Transmission_Control_Protocol
[2] https://profs.info.uaic.ro/~computernetworks/cursullaboratorul.php
[3] https://zetcode.com/db/sqlitec/