Documentație QuizzGame
(Tema 2)

Silitră Gabriel-Florin
2B4

1 Introducere

1.1 Motivația Alegerii

Am ales acest proiect deoarecere se aseamănă în oarecare măsură cu prima temă prezentată, astfel având o bază solidă de pornire și un potențial mai mare de a finaliza în totalitate proiectul. De asemenea, ideea de joc mă atrage și o consider a fi o metodă foarte eficientă prin care un student reușește să îmbine utilul cu plăcutul și anume, să înțeleagă conceptele prezentate în cadrul orelor de curs dar în același timp să construiască un joc interactiv între actori, reprezentați de server și mai mulți clienți.

1.2 Exemple Practice

Cel mai la îndemână exemplu îl reprezintă jocuri precum Triviador, Kahoot sau Trivia Challenge ce funcționează pe un model foarte asemănător cu cel prezentat în proiect. Aceste jocuri au creat comunități impresionante de fani și au generat venituri pe măsură. Trebuie amintit faptul că examenul parțial din cadrul materiei Introducere în Programare (IP) a fost susținut chiar pe platforma Kahoot. Deci iată un exemplu practic ce a facilitat evaluarea rapidă și corectă a studenților în contextul pandemic și al distanțării sociale din acea perioadă.

2 Tehnologii Utilizate

În cadrul proiectului au fost utilizate următoarele tehnologii:

2.1 Server Concurent TCP

Pentru realizarea conexiunii între server și clienți este folosit un server concurent TCP, deoarece funcționalitatea principală a acestui server permite conectarea mai multor clienți la server, în același timp. Spre deosebire de serverul iterativ, cel concurent permite conexiunea mai multor clienți la server și realizează schimbul de informații cu send() si recv() fără a ține cont de o anumită ordine prestabilită de momentele de conectare ale fiecărui client.

Librăria sys/socket.h este folosită pentru implementarea socket-ului ce realizeaza comunicarea dintre client şi server. Mai precis socket-ul permite comunicarea între două procese diferite de pe acelaşi calculator sau de pe doua calculatoare diferite folosindu-se de descriptorii de fişiere din UNIX.

Librăria sys/types.h este folosită pentru realizarea concurenței, mai precis pentru crearea proceselor fii care asigură deservirea mai multor clienți în acelaşi timp

Librăria netinet/in.h este folosită pentru stabilirea conexiunii dintre client şi server, mai precis ”găsirea” serverului de către client.

Librăria errno.h este folosită pentru gestionarea erorilor.

Librăria stdio.h este folosită pentru afișarea în aplicația clientului informații cerute de la server.

Librăria string.h este folosită pentru transmiterea întrebării de la server la client.

3 Arhitectura Aplicației

În prima etapă, aplicația permite clienților să se conecteze la server urmând ca mai apoi, într-o manieră sincronizată din server, clienții să primească o întrebare la care sunt nevoiți să răspundă. Pentru a sincroniza momentul în care întrebările sunt trimise, ar trebui sa ne asigurăm că toți clienții sunt conectați şi gata să primească informații de la server. Din momentul în care întrebarea a fost trimisă către clienți, un temporizator se va asigura că răspunsul este dat într-un timp de n secunde. După verificarea răspunsurilor de către server, se trimite către fiecare client punctajul obținut până în acel moment. Întrebările împreuna cu variantele de răspuns posibile, vor fi stocate prin intermediul unei baze de date SQLite.

De asemenea, în cazul în care un client întrerupe conexiunea cu serverul, acesta va fi eliminat din joc urmând ca restul întrebărilor să fie adresate fără nicio întrerupere celorlalți clienți conectați. După finalizarea jocului, serverul trimite un mesaj cu numele clientului câştigător către toți ceilalți clienți.

3.1 Diagrama UML

Pentru a înțelege mai bine funcționalitățile aplicației, reprezentăm printr-o diagramă UML cum funcționează:

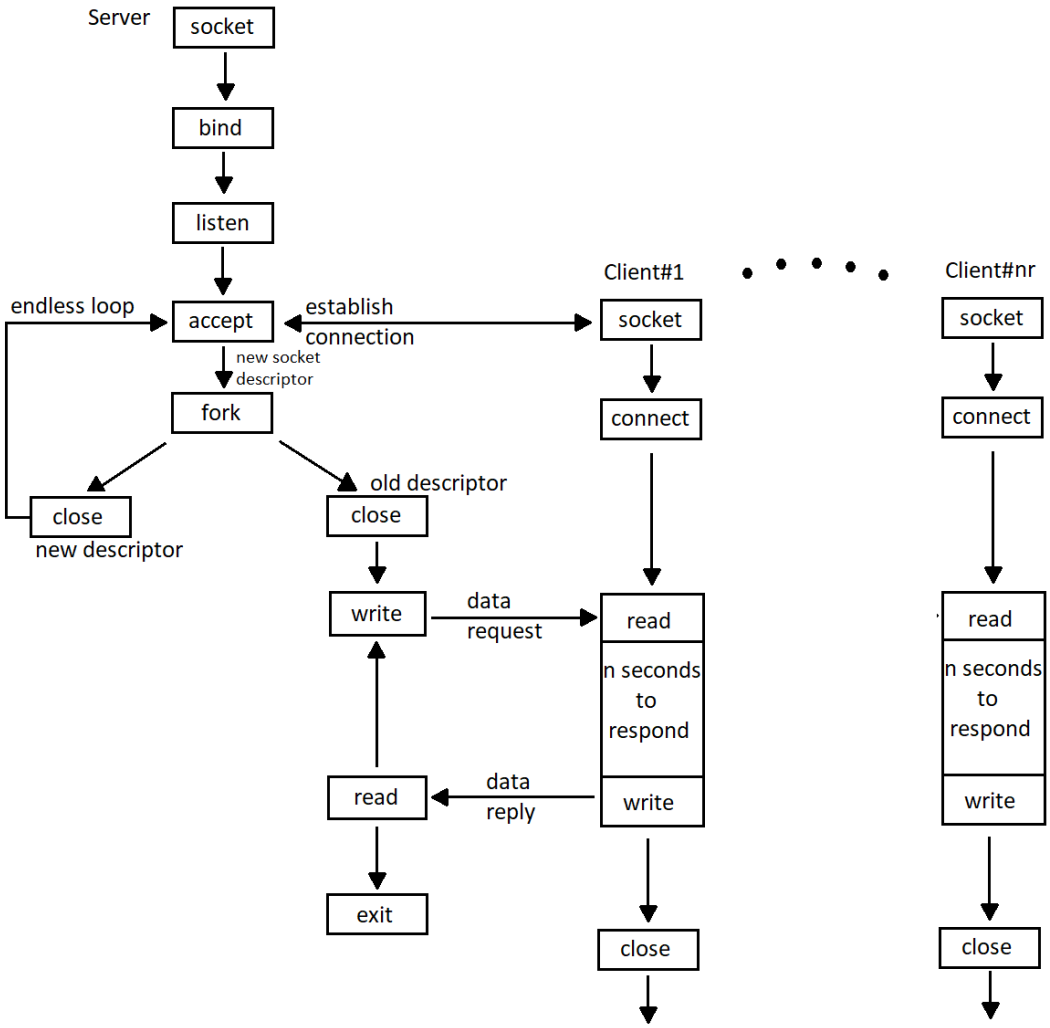

Fig. 1 Diagrama UML QuizzGame

4 Detalii de implementare

4.1 Server concurent TCP

In urmatoarea secventa de cod este implementat serverul concurent TCP:

```
#define PORT 3005
typedef struct thData{
	int idThread;
	int cl;
}thData;
int main ()
{
	struct sockaddr_in server;
	struct sockaddr_in from;
	int sd;
	int pid;
	pthread_t th[100];    //Identificatorii thread-urilor care se vor crea
	pthread_t log_timer_th;
	int i=0;


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

	int on=1;
	setsockopt(sd,SOL_SOCKET,SO_REUSEADDR,&on,sizeof(on));

	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 (struct sockaddr)) == -1)
	{
		perror ("[server]Eroare la bind().\n");
		return errno;
	}
```

```
        if (listen (sd, 2) == -1)
        {
                perror ("[server]Eroare la listen().\n");
                return errno;
        }

        pthread_create(&log_timer_th, NULL, &log_timer, NULL);

        while (1)
        {
                int client;
                thData * td;
                int length = sizeof (from);

                if ( (client = accept (sd, (struct sockaddr *) &from, &length)) < 0)
                {
                        perror ("[server]Eroare la accept().\n");
                        continue;
                }
                if(run_quizz == 1)
                {
                        while(1)
                                if(run_quizz == 0)
                                        break;
                }

                td=(struct thData*)malloc(sizeof(struct thData));
                td->idThread=i++;
                td->cl=client;
                number_cl++;
                cl[td->idThread].ok = -1;

                pthread_create(&th[i], NULL, &treat, td);

        }
};
```

4.2 Lucru cu baza de date SQLite3

```
int sqlite_data()
{
          sqlite3 *db;
           char *err_msg = 0;

           int rc = sqlite3_open("Questions.db", &db);

          if (rc != SQLITE_OK) {

           fprintf(stderr, "Cannot open database: %s\n",
             sqlite3_errmsg(db));
          sqlite3_close(db);

           return 1;
   }

   char *sql = "DROP TABLE IF EXISTS Questions;"
            "CREATE TABLE Questions(Id INT, Question TEXT, Answear TEXT);"
            "INSERT INTO Questions VALUES(1, '2 + 2?', '4');"
            "INSERT INTO Questions VALUES(2, 'Cum se numeste acest proiect?',
'QuizzGame');"
            "INSERT INTO Questions VALUES(3, 'Cum face pisica?', 'Miau');"
            "INSERT INTO Questions VALUES(4, '(int)Pow(10)?', '3');"
            "INSERT INTO Questions VALUES(5, 'Ce zi este maine?', 'Miercuri');";

   rc = sqlite3_exec(db, sql, 0, 0, &err_msg);
   char *sqll = "SELECT * FROM Questions";

   rc = sqlite3_exec(db, sqll, callback, 0, &err_msg);
   if (rc != SQLITE_OK ) {

      fprintf(stderr, "Failed to select data\n");
      fprintf(stderr, "SQL error: %s\n", err_msg);

      sqlite3_free(err_msg);
      sqlite3_close(db);

      return 1;
   }

   sqlite3_close(db);
}
```

```
int callback(void *NotUsed, int argc, char **argv, char **azColName)
{

        NotUsed = 0;
        strcpy(questions[cnt_questions], argv[1] ? argv[1] : "NULL");
        strcpy(answears[cnt_questions], argv[2] ? argv[2] : "NULL");
        cnt_questions++;

         return 0;
}
```

5 Concluzii

În concluzie, consider că aplicația ar putea fi îmbunătățită în ceea ce privește funcționalitatea prin alegerea unui nume din partea clientului înainte de a realiza conexiunea cu serverul, astfel încât serverul să poată reține într-un mod eficient punctajele fiecărui client.

De asemenea, în momentul în care toți clienții reușesc să răspundă în timp util la o întrebare, serverul ar putea să treacă la următoarea întrebare, în acest fel câștigând secundele rămase până la finalizarea timpului de răspuns. După expirarea celor n secunde puse la dispoziție pentru oferirea unui răspuns la întrebare, în cazul în care nu reușește să răspundă, clientul ar trebui penalizat cu un anume punctaj pentru întârzierea celorlalți participanți la joc.

5 Bibliografie

1. https://www.springer.com/gp/computer-science/lncs/conference-proceedings-guidelines
2. .https://profs.info.uaic.ro/~computernetworks/cursullaboratorul.php
3. https://profs.info.uaic.ro/~andreis/index.php/computernetworks/
4. <sys/socket.h> (opengroup.org)
5. <sys/types.h> (opengroup.org)
6. <netinet/in.h> (opengroup.org)
7. errno(3) - Linux manual page (man7.org)
8. C Programming/string.h - Wikibooks, open books for an open world
9. https://zetcode.com/db/sqlitec/?fbclid=IwAR35MufY6ggamKEFis4Yihghm4Un35ZRh7DcDFjMKYeIypLQEyOVP0JeBOM