TEMA 2

Erhan Alexandru 2A5

1 Introducere

Am ales proiectul QuizzGame de complexitate B. In scurt este vorba despre implementarea unui joc unde clientii raspund la un set de intrebari, intr-un timp stabilit si la final afisandu-se castigatorul jocului(cel care are cele mai multe raspunsuri corecte). Am ales anume acest proiect intrucat utilizeaza mai multe concepte si tehnologii necesare pentru viitor dar si pentru ca este un proiect de nivel mediu si astfel va necesita o informare mai adanca referitor la unele parti. Un alt motiv ar fi ca acest joc pe viitor poate fi imbunatatit astfel incat sa ajunga o aplicatie foarte interesanta.

2 Tehnologii utilizate

2.1 TCP

O tehnologie utilizata este protocolul TCP concurent care acopera nivelul de transport. Am ales anume acest tip de protocol deoarece este orientat-conexiune, adica aceasta legatura este realizata si mentinuta pana atat serverul cat si clientul termina de trimis mesaje.

In cadrul quizzului este nevoie de a fi asigurata transmiterea in ordine a intrebarilor si a optiunilor pentru a ramane jocul corect si egal pentru toti clientii. TCP asigura anume acest fapt, chiar daca viteza poate fi una mai lenta decat la alte protocoale, este sigur ca mesajele ajung intacte si in ordinea corecta.

Un alt avantaj este ca pachetele de date eventual oricum vor ajunge la client (in caz de eroare de conexiune de moment etc.) caci serverul mereu va cunoaste daca informatia a ajuns sau nu la client, astfel fiind capabil sa trimita mesajul urmator in ordines au sa retransmita pachetul anterior.

2.2 SQLite

O alta tehnologie utilizata este baza de date SQLite cu ajutorul librariei <sqlite3.h> ce va fi folosita pentru stocarea intrebarilor, optiunilor de raspuns si a utilizatorilor. Am ales anume acest tip de stocare a datelor (si nu fisiere XML sau TXT) in primul rand pentru performanta, caci operatiile de citire si scriere sunt foarte rapide, iar operatii de citire se vor face multe pentru fiecare intrebare si optiunile acesteia.

De asemenea continuturile bazei de date sunt updatate continuu astfel eliminandu-se riscul de a pierde informatii.

3 Arhitectura aplicatiei

3.1 Server concurent

In acest proiect pentru a putea servi mai multi client odata adica pentru a putea efectiv juca quizul, este necesar ca clientii sa poata fi conectati in paralel. In cadrul proiectului toata logica va fi implementata de catre server, adica toate functionalitatile : logare, creare joc, alaturare joc, afisare clasament, precum si conectarea la baza de date (pentru citire/scriere). Un aspect important este faptul ca la server se vor putea conecta un numar n de client, iar un joc se va incepe avand un minim de 3 clienti conectati. Clientul doar se va conecta la server dupa care va participa la joc. Protocolul utilizat este TCP astfel prin primitivele de creare socket, bind(), listen() si accept() se va realiza conexiunea dintre client si server conform unei adrese IP si a unui port specific.

3.2 Multithreading

Un concept implicat este acel de multithreading, adica clientii sunt creati prin primitiva pthread_create(). Toate threadurile unui proces partajeaza resursele acestuia (memorie, date, fisiere etc) astfel nu se creaza o copie (precum la fork()), acest fapt ajuta la economisirea de spatiu. Alt avantaj al firelor de executie este ca trimiterea de mesaje si comunicare este mai rapida astfel va fi posibila sincronizarea de afisare a intrebarilor.

3.3 Baza de date

Un alt concept implicat este acel de baza de date, cu ajutorul careia vor fi stocate intrebarile si optiunile de raspuns (inclusiv care este cea corecta), vor fi create 2 tabele : Intrebari si Optiuni pentru o stocare mai eficienta si clara a datelor.

Va exista un atribut comun (ID Intrebare) pentru ambele tabele pentru a asigura conexiunea dintre acestea si pentru a fi posibila operatia de JOIN.

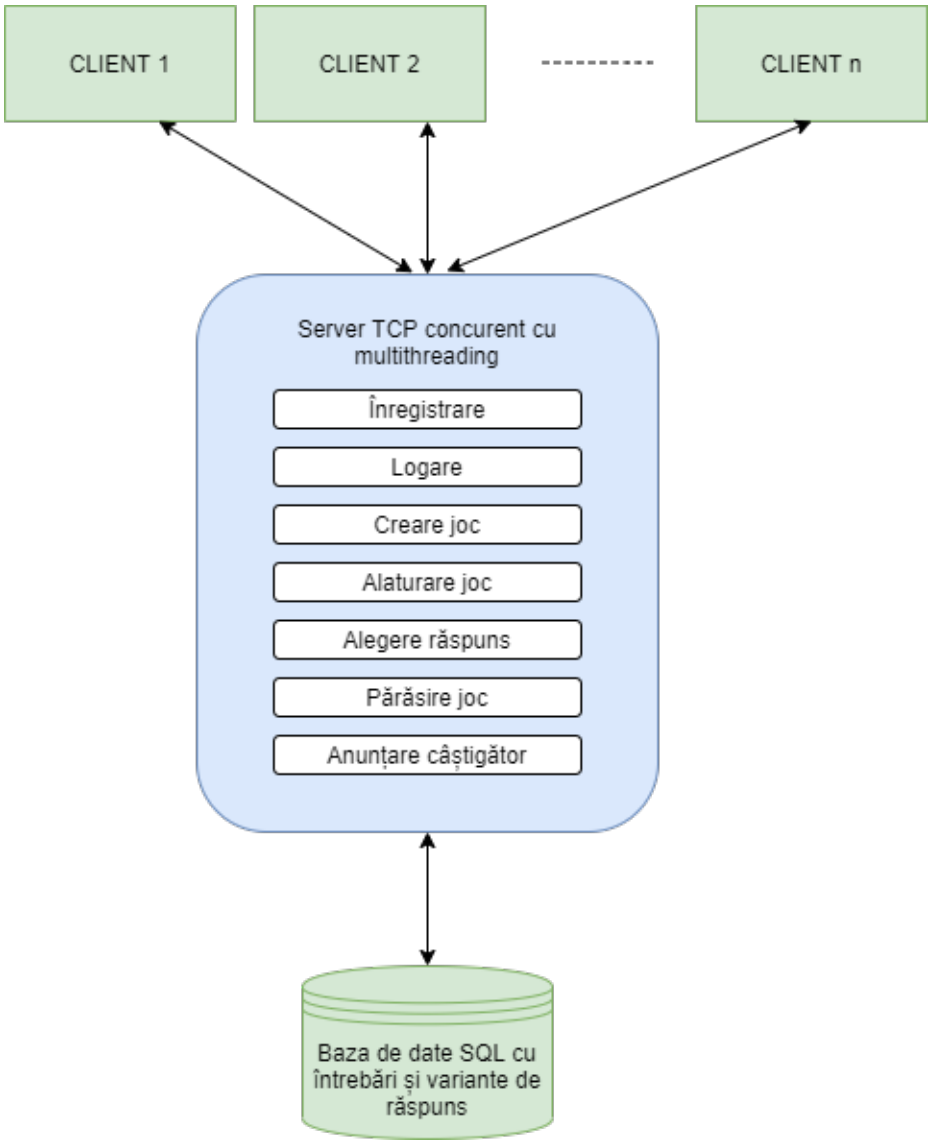

Fig. 1. Asa arata arhitectura aplicatiei la care se pot conecta oricati client, pentru fiecare se va crea un thread nou. Serverul va fi de tip TCP concurent cu functionalitatile descrie mai sus unde fiecare client poate crea un joc (codul va fi generat random) se poate alatura unui joc pe baza codului si poate parasi jocul in caz de necessitate. Serverul va fi conectat la baza de date ce va contine intrebarile, variantele de raspuns, useri si jocurile existente la moment.

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

/* s-a realizat conexiunea, se astepta mesajul */

// int idThread; //id-ul threadului
// int cl; //descriptorul intors de accept

td = (struct thData *)malloc(sizeof(struct thData));
td->idThread = i++;
td->cl = client;

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

Fig. 2. Un fragment de cod ce demonstreaza cum va avea loc comunicarea dintre server si client, cu ajutorul primitive accept(), si partea de multithreading creandu-se pentru fiecare nou client un thread nou, care este tratat cu ajutorul functiei treat().

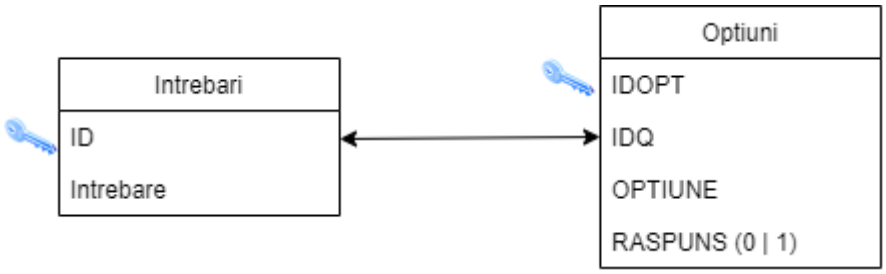

Fig. 3. O schema ce arata continutul bazei de date si adica Tabelele Intrebari si Optiuni, cheile primare fiind ID si IDOPT, iar IDQ este cheia straina pe baza careia va fi posibil de facut operatia Join.

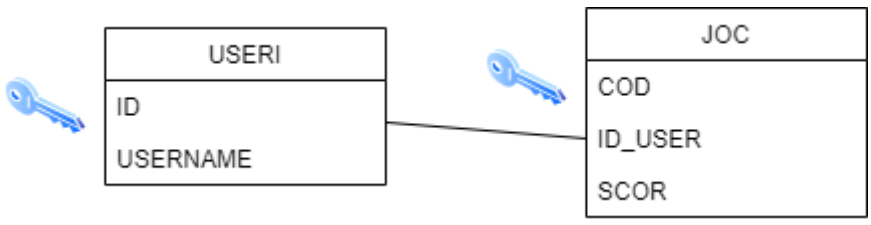

Fig. 4. O schema ce arata celelalte tabele ale bazei de date si anume USERI si JOC. Tabela useri va contine username-ul si un ID (cheie primara), iar tabela JOC va contine codul unic al unui joc si de asemenea toate idurile userilor ce participa la acel joc Atributul scor va fi folosit pentru a pastra punctajul fiecarui jucator participant la un anumit joc.

4 Detalii de implementare

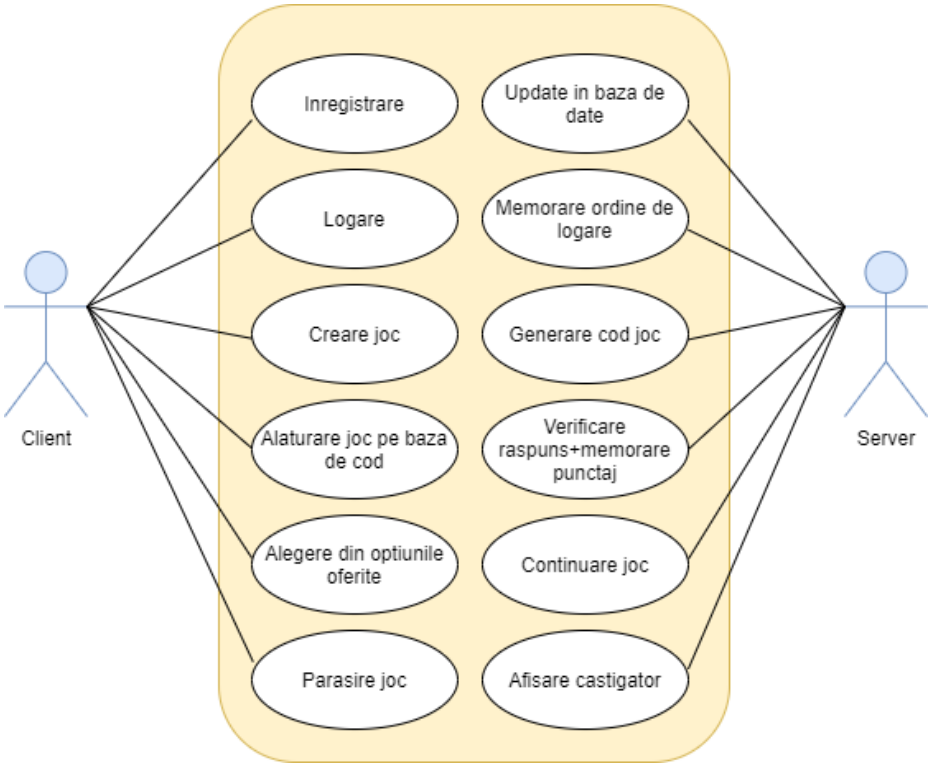

Fig. 5. Este un exemplu de diagrama usecase si arata ce va face serverul la fiecare actiune a clientului. Astfel la inregistrare va face o scriere in baza de date, la logare o citire din tabela Useri. Cand se vac rea un joc se va genera un cod de joc ce va fi pastrat in tabela JOC din baza de date. Daca clientul doreste sa se alature altui joc el va introduce un cod, serverul va verifica daca codul exista in baza de date si daca da, ii va introduce username in tabela. La fiecare raspuns la o intrebare a clientului, serverul va verifica raspunsul si va memora punctajul acestuia. In cazul parasirii jocului de catre un client, jocul va continua fara probleme. La finalul rundei de intrebari serverul va trimite tuturor clientilor castigatorul jocului.

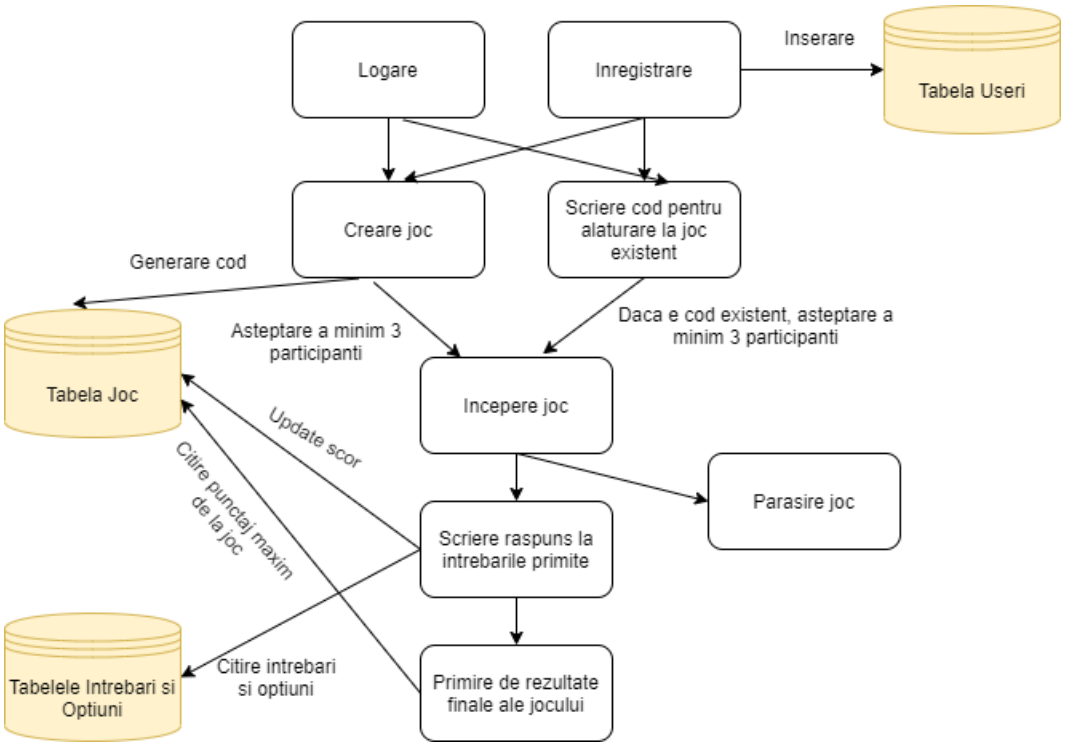

Fig. 6. Intai de toate clientul va trebui sa se logheze sau daca acesta nu are cont sa se inregistreze (nu va putea trece mai departe pana nu se va loga). Serverul va memora ordinea de logare. Dupa care la alegerea jucatorului el poate crea o camera de joc, astfel generandu-se un cod random si va astepta conectarea a inca minim 2 utilizatori pentru a putea incepe jocul. In cadrul jocului vor aparea pe rand intrebarile si fiecare jucator va avea un timp de 30 de secunde pentru a alege varianta corecta. Daca este un raspuns correct va primi un punct, altfel 0. Un alt scenariu este cand un jucator nu doreste sa devina host, astfel el va putea alege din camerele de joc deja existente, va introduce codul si se va putea conecta la joc. In orice moment al jocului un utilizator poate parasi camera, insa asta nu va crea vreo problema jocului, el va continua cu urmatoarele intrebari pentru jucatorii ramasi. Iar la finalul jocului se va afisa utilizatorul cu cel mai mare punctaj obtinut pe baza tabelei JOC existente, in toti clientii care au participat.

```
while (1)
{
  if (read(tdL.cl, &nr, sizeof(int)) <= 0)
  {
    printf("[Thread %d]\n", tdL.idThread);
    perror("Eroare la read() de la client.\n");
  }
  printf("[Thread %d]Mesajul a fost receptionat...%d\n", tdL.idThread, nr);
  if (nr == 1)
  {
    login((struct thData *)arg);
  }
```

Fig. 7. In aceasta bucla infinita din server vor avea loc citirile de la client si efectuarea cazurilor in dependenta de raspunsurile primite.

```
snprintf(sql, 1024, "INSERT INTO USERI (USER)"
                    "VALUES ('%s')",
        username);
rc = sqlite3_exec(db, sql, NULL, 0, &zErrMsg);

if (rc != SQLITE_OK)
{
  fprintf(stderr, "SQL error: %s\n", zErrMsg);
  sqlite3_free(zErrMsg);
}
else
{
  fprintf(stdout, "Records created successfully\n");
  strcpy(rs, "");
  strcat(rs, "Am creat cont si te-am logat cu succes");
  if (write(tdL.cl, rs, sizeof(rs)) <= 0)
```

Fig. 8. In acest fragment de cod se arata cum va avea loc inregistrarea unui user nou si anume printr-o scriere in baza de date, in tabelul USERI dupa care o scriere spre client pentru a-l anunta ca inregistrarea a avut loc si poate continua cu restul comenzilor.

5 Concluzii

Desigur proiectul poate fi imbunatatit prin mai multe modalitati. Una ar fi utilizarea unei interfete pentru utilizatori, pentru a fi mai usor de jucat quizzul si pentru a fi mai clare functionalitatile. O alta imbunatatire posibila ar fi crearea sansei de a juca cu anumiti utilizatori (pe baza de username) adica a crea o camera cu un

cod random si a primi doar utilizatorii doriti de host. De asemenea s-ar mai putea implementa alegerea unui tematici anumite pentru setul de intrebari ce va fi jucat intr-o camera anumita. Daca aplicatia ar fi imbunatatita cu toate aceste feature-uri atunci ar avea o buna aplicabilitate in viata reala fiind o aplicatie interactiva si educativa de quiz.

Bibliografia

1. SQL Online IDE - for Data Science (sqliteonline.com)
2. Course&Laboratory - Computer Networks 2020-2021 (uaic.ro)
3. https://profs.info.uaic.ro/~computernetworks/files/NetEx/S12/ServerConcThread/servTcpConcTh2.c
4. https://profs.info.uaic.ro/~computernetworks/files/NetEx/S12/ServerConcThread/cliTcpNr.c
5. SQLite - C/C++ - Tutorialspoint
6. SQLite.: https://www.sqlite.org/about.html
7. Multithreading in C - GeeksforGeeks