

QuizzGame - Raport tehnic

Aciocanese Bianca-Ioana (2A1)

January 10, 2024

1 Introducere

Proiectul **Quizz Game** are ca scop crearea unei aplicatii pentru un numar nelimitat de clienti, care, conectati in retea, pot participa la un joc in care pot raspunde la intrebari primite de la server, pe o anumita tematica. Serverul asigura verificarea raspunsurilor primite de la client, in scopul calcularii scorului pe baza numarului de raspunsuri corecte, cat si gestionarea interactiunilor dintre clienti, in sensul in care, dupa ce un client termina quizz-ul, poate sa vada unde se situeaza in clasament cu scorul obtinut, in comparatie cu ceilalti clienti din retea care joaca in acelasi moment de timp (aceeasi runda de joc) ca si el.

2 Tehnologii Aplicate

La baza implementarii proiectului se afla protocolul **TCP - Transmission Control Protocol**, in detrimentul protocolului UDP, datorita avantajelor acestuia. In primul rand, protocolul asigura o comunicare orientata conexiune, adica garanteaza trimiterea datelor de la client la server in ordinea corecta si fara erori, deci nu se realizeaza pierderi de informatii, fapt important pentru asigurarea unei bune desfasurari a jocului. Desi protocolul UDP este mai rapid, in cazul proiectului de fata, transmiterea informatiilor in siguranta este mult mai esentiala decat rapiditatea. Intrebarile, impreuna cu variantele de raspuns, sunt stocate intr-o **baza de date SQLite**, de unde sunt extrase de catre server in momentul in care clientul doreste sa inceapa quizz-ul. In plus, pentru implementarea concurentei in tratarea cererilor de participare la quizz primite de la clienti, serverul va folosi mai multe **thread-uri** - cate unul pentru fiecare client. Pentru asigurarea unei experiente *user - friendly*, s-a utilizat **biblioteca grafica GTK3**.

3 Structura Aplicatiei

Conform celor mentionate anterior, aplicatia functioneaza pe modelul TCP. Dupa stabilirea conexiunii cu clientul prin **three way handshaking**-ul specific TCP-ului, serverul creeaza un fir de executie pentru a furniza quizz-ul respectivului client.

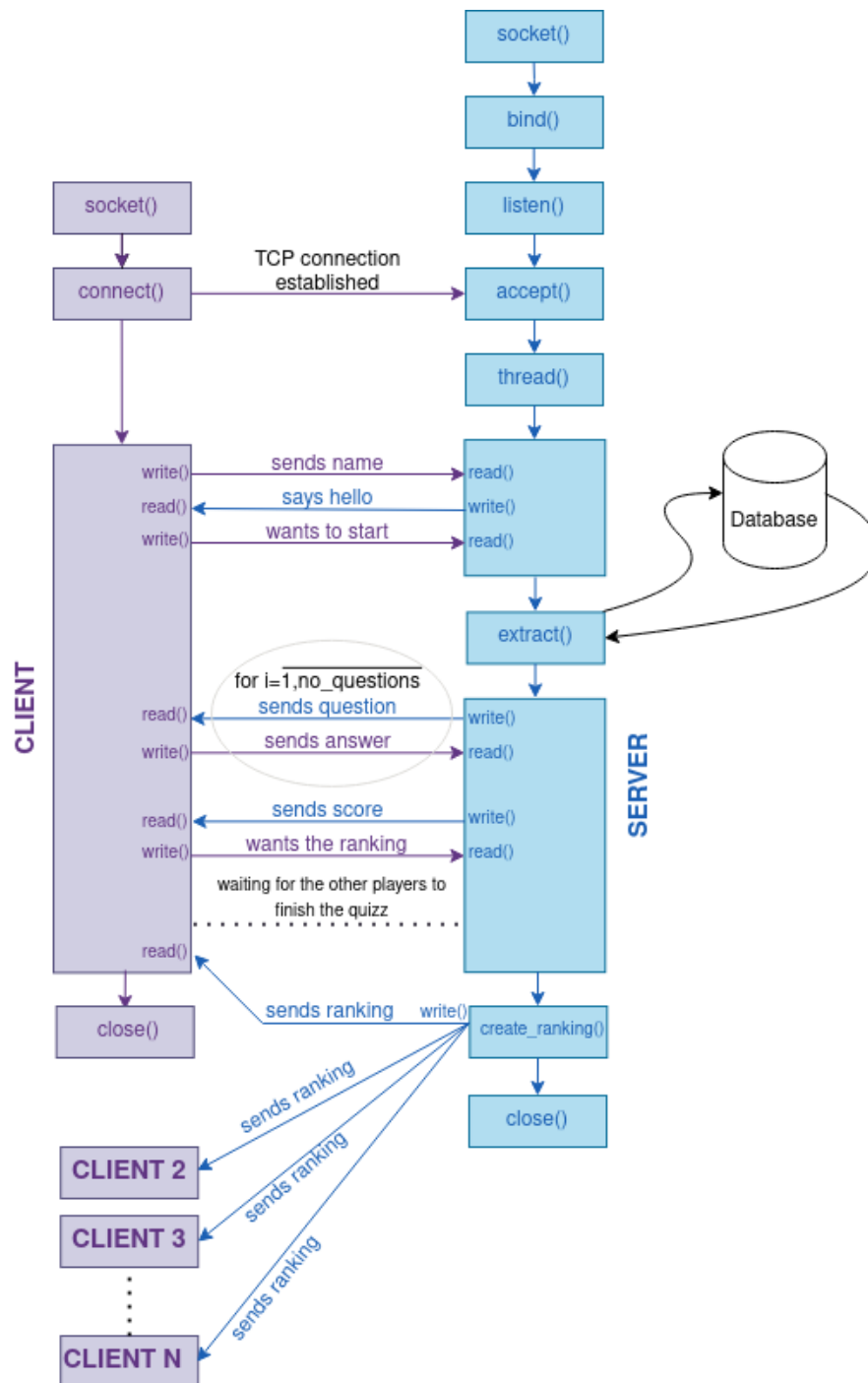
Urmeaza apoi o serie de utilizari ale primitivelor `read()` si `write()`, care permit schimbul de mesaje intre client si server. Clientul trimite mai intai numele, dupa care este nevoit sa astepte pana cand meciul curent de joc se incheie (daca se desfasoara unul in momentul in care acesta doreste sa inceapa quizz-ul). Intre o runda de joc si alta, se realizeaza o pauza de 20 de secunde, in care se mai acorda timp pentru ca alti jucatori sa se conecteze la noua runda.

Urmeaza un alt schimb de mesaje de tip intrebare-raspuns - serverul trimite intrebarea clientului, clientul ii trimite inapoi raspunsul, si tot asa pana cand s-au epuizat toate intrebarile. Pentru fiecare intrebare este setat un timer de 5 secunde, in care clientul poate apasa pe raspunsul pe care il considera corect. Clientul primeste apoi scorul de la server, calculat in functie de numarul de intrebari la care a raspuns corect.

Daca doreste, clientul poate astepta pana cand ceilalti participanti la joc finalizeaza si ei de raspuns la intrebari, pentru a putea primi de la server clasamentul jocului. Daca nu doreste sa mai astepte clasamentul dupa ce a primit scorul propriu, poate apasa `exit` (de altfel, clientul poate parasi jocul inclusiv in timpul rundei de intrebari - utilizarea butonului `exit` va duce la eliminarea clientului din clasamentul final). Apoi, conexiunea dintre client si server este incheiata (ori dupa ce s-a primit si clasamentul, iar runda de joc este gata, ori dupa apasarea comenzii `exit` in timpul jocului), serverul pregatind structurile de date folosite pentru inceperea unei noi runde de joc.

Clientii din aceeași runda de joc sunt sincronizati, in sensul in care toti incep jocul in exact acelasi moment de timp, si dupa ce raspund la intrebari in timpul acordat si aleg sa astepte clasamentul, acesta este trimis in acelasi timp abia cand toti jucatorii au terminat quizz-ul. Daca un meci este deja inceput, un jucator care abia atunci doreste sa intre in joc, trebuie sa astepte pana cand runda curenta este incheiata.

In figura care urmeaza, se pot observa, schematic, descrierile facute mai sus pentru realizarea comunicarii intre server si un singur client (*in aceasta reprezentare, CLIENT 2, CLIENT 3,...,CLIENT N sunt clienti supusi aceluiași sir de schimb de mesaje, dar sunt utilizati in figura pentru a evidentia primirea simultana a clasamentului la finalul rundei de joc - ei reprezinta alti clienti care vor aparea in clasamentul final).



4 Aspecte de Implementare

În completarea prezentării schematice de mai sus a ideii de bază a proiectului, se vor prezenta în cele ce urmează și câteva aspecte esențiale de implementare și secvențe reprezentative de cod, cât și funcțiile utilizate atât în client, cât și în server.

Având în vedere că toată logica proiectului este realizată în server, pe partea de **client** funcționalitățile sunt unele minimale, deoarece singura sarcină a sa este aceea de a răspunde la întrebările primite. Funcționalitățile importante din partea de client sunt:

- introducerea de la tastatură a numelui pe care dorește să îl aibă pe parcursul quizz-ului;
- pornirea quizz-ului în momentul în care apasă butonul de start și așteptarea începerii runde de joc;
- trimiterea către server a răspunsului la fiecare întrebare, pe rand, prin apăsarea pe butonul cu litera pe care o consideră corectă, în timpul acordat; dacă nu se răspunde la întrebare în acel interval de timp, scorul pentru întrebare este 0;
- parasirea jocului oricând dorește prin apăsarea **exit**, fapt care îl va duce la eliminarea din rundă, deci nu va mai putea fi luat în considerare la clasamentul final;

Pe partea de **server**, funcționalitățile sunt următoarele:

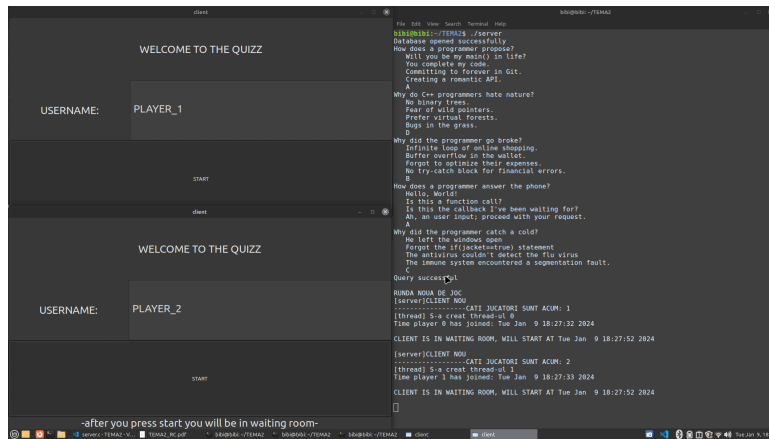
- extragerea din bază de date de către server a întrebărilor, pentru ca acestea să fie trimise spre client - bază de date conține 5 întrebări, fiecare cu câte 4 variante de răspuns, cât și litera răspunsului corect;
- asigurarea sincronizării clienților prin începerea unei runde de joc simultan pentru toți clienții și asigurarea că un jucător, dacă ajunge mai târziu decât începutul unei runde, este introdus în următoarea;
- comenzile de bază pe care serverul trebuie să le execute, și anume: trimiterea întrebărilor către clientul care s-a conectat, cât și a scorului la final de quizz; trimiterea clasamentului la finalul runde de joc, când toți clienții au terminat de răspuns la întrebări — > toate acestea realizate în funcția **void raspunde(void *arg)**;
- asigurarea eliminării unui client în momentul în care primește de la acesta comanda exit, cât și încheierea conexiunii cu el;
- resetarea statusului jocului la finalul unei runde (funcția **void reset()**), cu scopul de a goli informațiile de la runda anterioară de joc despre scorul clienților, numărul de jucători, clasament etc și a pregăti structurile de date pentru primirea altor clienți;
- stocarea organizată a informațiilor, ca în cele ce urmează (cu explicațiile din partea dreaptă):

```
//stocarea intrebarilor extrase din baza de date
struct quizz_questions{
    char q[100]; //intrebarea
    char ans_A[100]; //varianta a de raspuns
    char ans_B[100]; //varianta b de raspuns
    char ans_C[100]; //varianta c de raspuns
    char ans_D[100]; //varianta d de raspuns
    char correct_ans; //litera corespunzatoare raspunsului corect
}questions[no_questions];
```

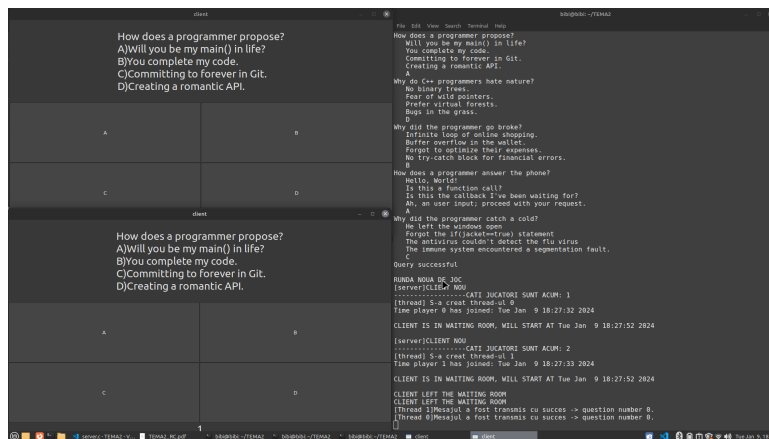
```
//stocarea informatiilor despre clientii participanti
struct info_quizz{
    char name[20]; //numele jucatorului
    int score; //scorul obtinut
    int done; //variabila care retine daca jucatorul a terminat quizz-ul
    int out; //variabila care retine daca jucatorul a parasit quizz-ul
    int descriptor; //descriptorul de socket asociat clientului
    char answers[no_questions]; //literele de raspuns primite de la client
}clients[MAX_CLIENTS];
```

```
//retinerea informatiilor legate de starea jocului
struct status_game{
    int players_number; //numarul de jucatori participanti
    int players_done; //cati dintre acestia au terminat quizz-ul
    int players_out; //cati au parasit jocul
    char ranking[MAX_CLIENTS]; //clasamentul care va fi realizat la final de joc
}status;
```

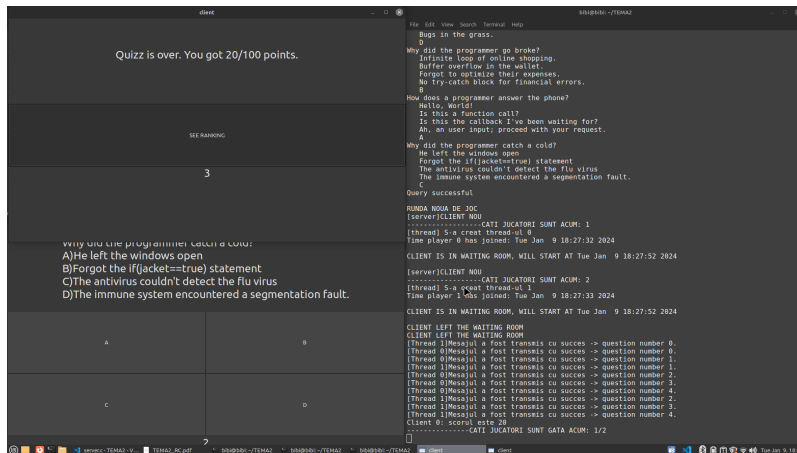
In cele ce urmeaza, se va descrie un scenariu de utilizare. Sa presupunem ca se conecteaza la server 2 clienti. Se observa ca, atunci cand acestia apasa tasta enter, serverul creeaza un thread pentru fiecare si actualizeaza cati jucatori sunt la fiecare aparitie a unui jucator nou. Totodata, seteaza timpul de inceput al runde (se verifica mai intai daca o runda este deja inceputa prin intermediul variabilei game_started, care, daca este 1, blocheaza clientul intr-un while, asteptand ca un thread sa o modifice la incheierea unei runde de joc si sa permita clientului sa iasa din while si sa inceapa comunicarea cu serverului si crearea threadului pentru acesta).



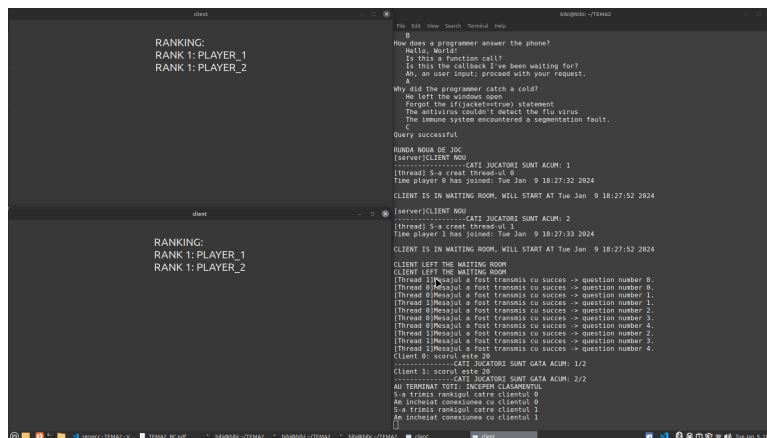
Dupa ce acestia parasesc "waiting-roomul", adica cele 20 de secunde dintre runde au trecut, ei incep sa primeasca intrebarile, alaturi de cele 4 butoane asociate variantelor de raspuns, vazand in partea de jos a ecranului si timpul care le-a ramas pentru a raspunde la intrebare. Daca raspund la o intrebare, sau daca nu raspund si timpul s-a scurs, pot trece la urmatoarea intrebare.



Se observa ca dupa ce player1 termina de raspuns la intrebari, primeste scorul obtinut si poate apasa SEE RANKING pentru a astepta ca si player2 sa termine jocul si sa primeasca in cele din urma clasamentul. Serverul actualizeaza faptul ca un jucator a terminat deja.



Dupa terminarea meciului si de catre player2, clasamentul este trimis spre ambii jucatori in acelasi timp, si conexiunile client-server se incheie. Acum se pot conecta la joc si jucatorii aflati in waiting, care au intrat cand runda de joc era inceputa. 2 jucatori se afla pe acelasi rank daca au obtinut acelasi scor.



Alt scenariu de joc este cel in care un client apasa exit si serverul este anuntat si are grija sa incheie conexiunea cu acesta si sa trimita clasamentul doar spre jucatorii ramasi, pentru a continua runda fara probleme.

5 Concluzii

Solutia propusa si descrisa mai sus pentru proiectul QuizzGame respecta cerintele acestuia, insa poate primi imbunatatiri, precum functionalitati care sa aduca utilizatorului o experienta mai prietenoasa. O idee ar fi aceea de a lasa clientului optiunea de a alege dintr-o lista cu tematici pentru intrebari, pe care serverul i-o va propune inainte de a incepe sa trimita intrebarile. De

asemenea, se poate adauga functionalitatea de a vedea, odata cu clasamentul, si cel mai bun scor obtinut vreodata de cineva la quizz-ul cu tematica aleasa de utilizator. Sau se poate adauga un alt mini-game, pe care utilizatorul in poate juca single-player pana cand timpul de asteptare de inceput de runda se scurge.

Referinte bibliografice

- [1] Site-ul disciplinei pentru exemplu de server concurent care creeaza cate un thread pentru ficare client conectat
<https://profs.info.uaic.ro/~computernetworks>
- [2] Aplicatie pentru crearea diagramei
<https://app.diagrams.net/>
- [3] Transmission Control Protocol. Wikipedia, The Free Encyclopedia
https://en.wikipedia.org/w/index.php?title=Transmission_Control_Protocol&oldid=1189524535
- [4] Utilizarea bibliotecii GTK
https://docs.gtk.org/gtk3/getting_started.html
<https://github.com/RainMark/gtk3-tutorial>