Quizz Game

Strătianu Bianca-Ionela, grupa A6, anul II

1 Introducere

Acest raport prezintă detalii despre realizarea jocului Quizz Game in limbajul C++.

Aceasta este o aplicaţie client-server în care clienţii răspund unor întrebări. Aceştia se autentifică cu un nume, iar pe urmă se pot alătura unui joc deja existent(dacă au codul jocului) sau pot să creeze un nou joc la care pot participa doar clienţii care au codul jocului.

Clientul care a creat un joc nou poate decide dacă mai asteaptă alţi jucători sau dacă începe jocul. În cazul în care acesta decide să părăsească jocul înainte ca acesta să înceapă, jocul respectiv se anulează. Iar în cazul în care acesta vrea să părăsească jocul după ce a început, acesta trebuie să continue.

Întrebările sunt puse clienţilor în ordinea în care s-au alăturat în joc. Clienţii au la dispozitie 20 de secunde pentru a alege răspunsul corect din cele patru oferite la o intrebare, serverul verifică răspunsul dat de client şi reţine scorul jocătorilor, iar la final, anunţă trimite tuturor jucătorilor care au rămas în joc clasamentul. Serverul suporta mai multi jucători şi desigur mai multe jocuri consecutive.

Întrebările împreună cu răspunsurile sunt stocate într-o bază de date SQLite.

2 Tehnologii utilizate

Pentru realizarea conexiunii client-server am utilizat modelul TCP[**1**] concurent (multithreading), care permite posibilitatea conectării mai multor clienţi la server. Am utilizat această conexiune deoarece vreau să am siguranţa şi asigurarea transmiterii datelor în ordin, pentru că nu vreau ca un jucator să răspundă corect, iar răspunsul să nu ajungă la server.

Serverul TCP va crea câte un thread[**2**] (fir de execuţie) pentru fiecare client care vrea să se conecteze la el, asigurând astfel posibilitatea conectării mai multor clienţi simultan. Acesta are rolul de a prelua datele de la clienţi, a le procesa si a le transmite înapoi clienţilor.

Pentru realizarea graficii am folosit biblioteca GTK+[**3**] deoarece este o bibliotecă simplă de utilizat, dar ş i pentru că conţinea toate lucrurile de care aveam nevoie. Am reuşit sa fac o interfaţa grafică unde clientul poate iesi oricand vrea el, iar proprietarul jocului stabileşte setările jocului.Iar pentru realizarea interfeţei am folosit aplicaţia Glade [**4**].

Clientul va stabili un port pentru conexiune. Va avea rolul de a prelua datele de la server şi a le afişa în interfaţa grafică, iar pe urmă preia datele introduse de utilizator si le transmite serverului.

Pentru stocarea întrebărilor am folosit o bază de date SQLite[**5**] deoarece este uşor de utilizat, oferă uşurinţa reţinerii datelelor, dar şi actualizarea sau ştergerea lor, fiind totodata stocate în siguranţă. Astfel, actualizându-mi şi cunoşitinţele de la cursul de Baze de date.

În baza de date am stocat mai multe tabele, de unde vor fi preluate întrebările în funcţie de alegerile jucătorului care creeaza acel joc.

În funcţie de numărul întrebării aceasta este citită din baza de date şi transmisă tuturor clienţilor care sunt in acel joc.

3 Arhitectura aplicaţiei

Am realizat conexiunea dintre server si client cu ajutorul protocolului TCP multi-threading care creează câte un fir de execuţie pentru fiecare client, putând servi clienţii în mod concurent. Serverul preia informaţia de la client, o procesează şi o transmite înapoi clientului.

Am construit o bază de date în care am pus întrebările si răspunsurilor pe care le voi transmite clientilor, dar şi răspunsul corect pentru a putea citi de la client răspunsul şi să îl compar cu cel din baza de date, actualizând punctajele clienţilor în cazul în care au răspuns corect.

Pentru a putea lega baza de date cu jocul, am instalat baza de date compatibilă cu versiunea mea Linux, am instalat librariile necesare in cadrul programului şi am creat legatura dintre cele două.

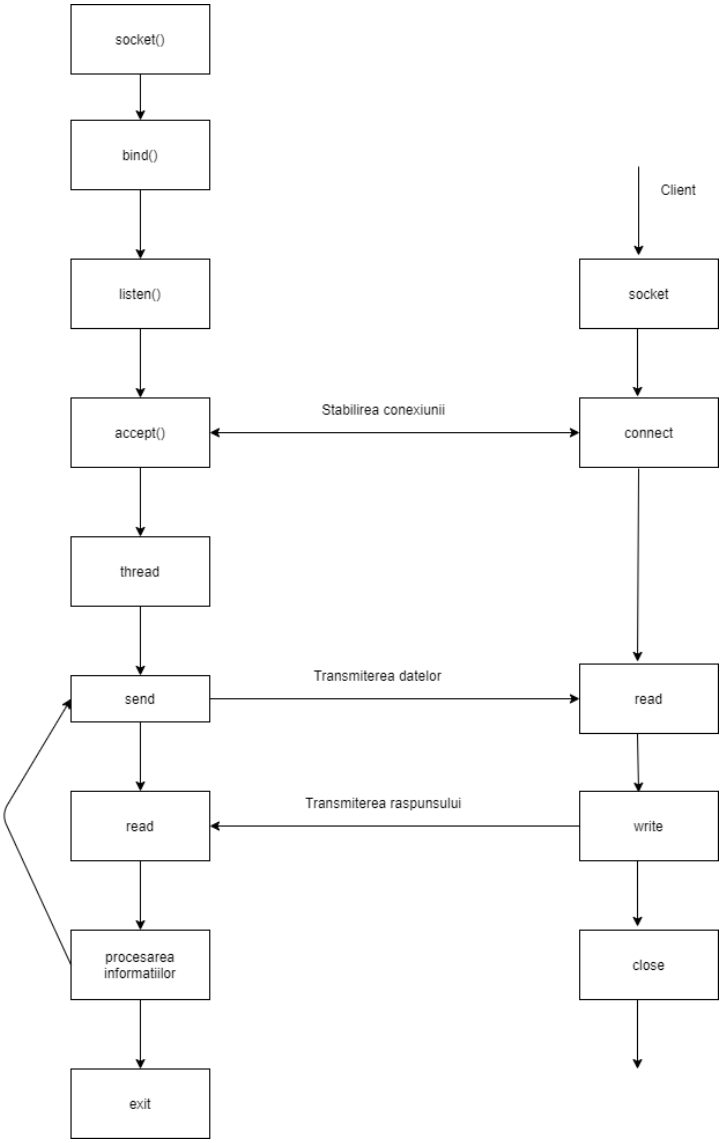

Fig. 1.

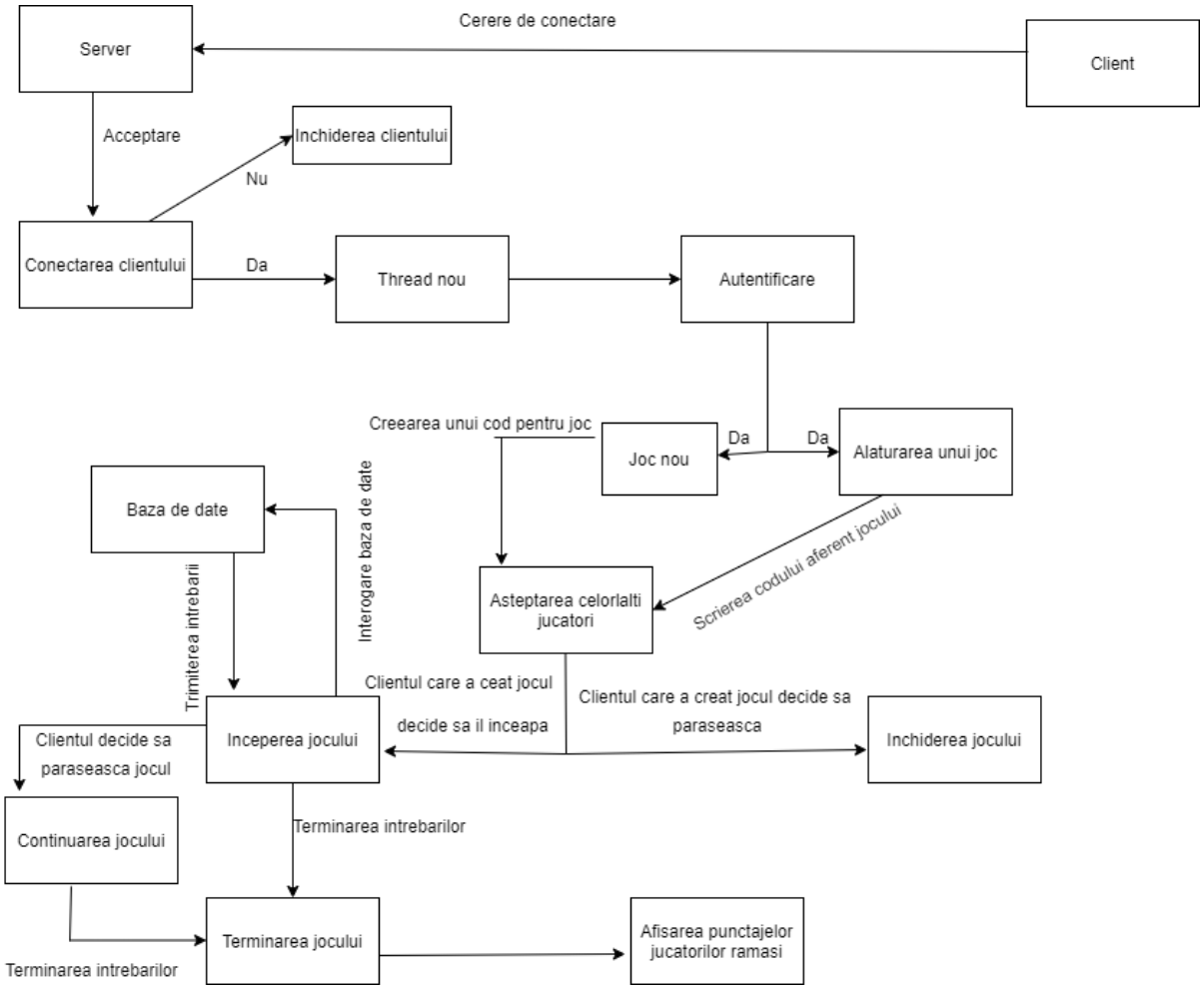

Fig. 2. Diagrama aplicaţiei

4 Detalii implementare

Pentru ca aplicaţia să funcţioneze normal trebuie să respecte diagrama de mai sus. Funcţia de autentificare are rolul a crea un cont clientului. Acesta decide dacă se alatură unui joc deja existent (dacă ştie codul jocului respectiv) sau dacă creează un joc nou cu un cod secret.

Clientul care a decis să facă un joc nou decide când incepe jocul propriu-zis. Dacă aceste decide să părăsească jocul înainte să înceapă, jocul se termină. Însă dacă acesta sau oricare altul din acel joc decide să părăsească jocul după ce a început, ceilalţi clienţi pot continua să răspundă la întrebări.

Întrebările sunt puse clienîlor în ordinea în care s-au alăturat în acel joc. După ce un client a răspuns la întrebarea pe care a primit-o sau dacă a trecut timpul, acesta merge într-o cameră separată unde aşteaptă sa îi vină rândul din nou. În această cameră el poate decide dacă părăseşte jocul.

În orice moment al jocului clientul poate părăsi si jocul, acest lucru nu afectează ceilalti jucători, fiind şters şi din clasamentul final

Clienţii dintr-un joc răspund la întrebări în ordinea în care aceştia s-au alăturat jocului. Ei răspund la întrebări, iar interogând din nou baza de date, verificăm dacă răspunsul este corect. Punctajul clientului fiind actualizat. Dacă a trecut timpul şi acesta nu a răspuns clasamentul rămâne acelaşi, jocul se continuă cu următorul jucător,

După ce toate întrebările din baza de date au fost parcurse, este calculat şi trimis clasamentul tuturor clienţilor care au rămas în joc.

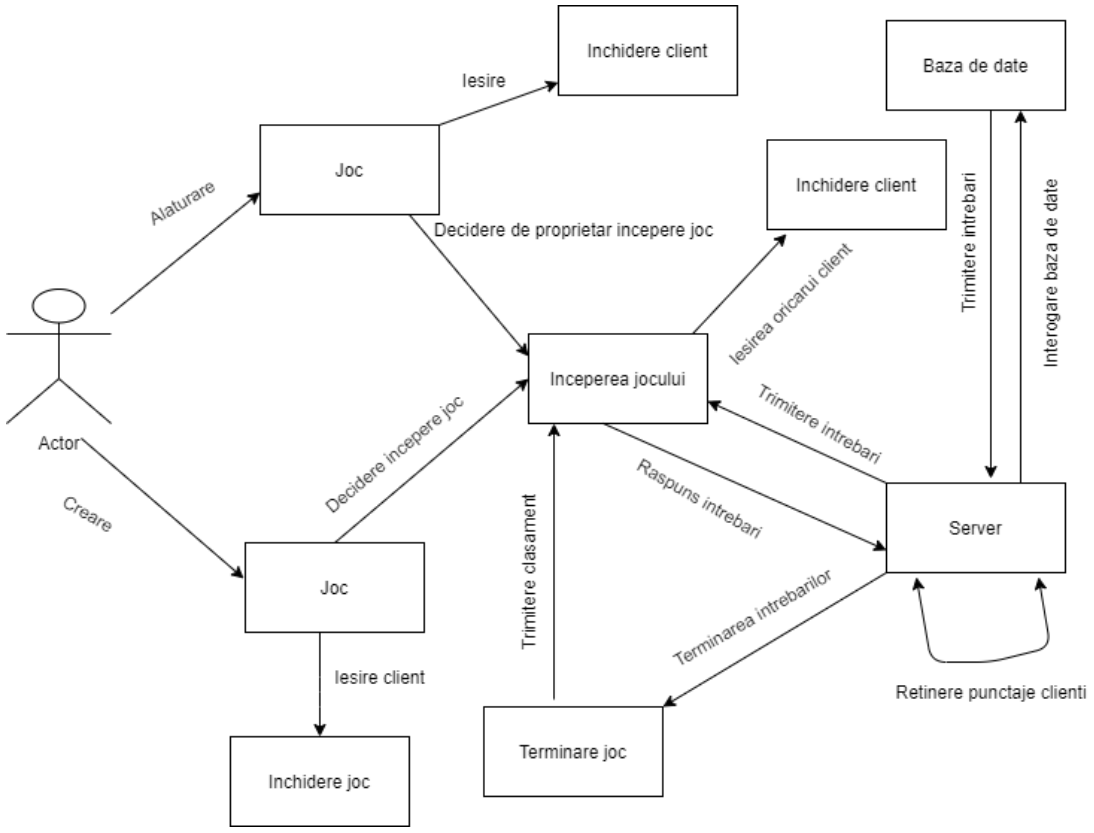

Fig. 3. Uses cases

Un scenariu nefericit ar fi atunci când un client care care aşteaptă să înceapă un joc, decide să îl părăsească, dar pentru asta am o funcţie care îl şterge pe acest jucător din lista juaţorilor care deţin acel cod.

Un alt scenariu nefericit ar fi atunci când clientul care a creat jocul îl părăsește după ce a început, dar ceilalți jucători pot continua jocul deoarece doar statutul clientului inițial se schimbă, ceilalți clienți primind în continuare întrebări.

Un scenariu fericit este atunci când clientul intră intr-un joc deja creat, proprietarul jocului decide când acesta începe jocul, fiecare client din acel joc primind întrebările în ordinea în care au venit în joc.

Un alt scenariu fericit este atunci când clientul creează un nou joc pentru care creeaza si o parola unică. El așteaptă clienții care au această parolă să se alature jocului, primind întrebări, iar la final fiind afișat clasamentul tuturor jucătorilor.

În figura 1 avem:

Prima structură reţine pentru fiecare client detalii despre acesta cum ar fi: numele, dacă a creat un joc sau s-a alaturat unui joc, scorul acestuia, descriptorul, dar şi timpul când a primit o întrebare pentru a sti când a trecut timpul să putem trece la următorul client.

Cea de a doua structură reţine pentru fiecare joc nou creat, clienţii care sunt în acel joc, numărul de jucători, dar şi numărul de întrebări atât puse cât şi numărul de întrebări pe care le-a ales jucătorul care a creat jocul, pentru a şti cand s-a terminat acea sesiune.

Serverul extrage întrebarea din baza de date, dar şi răspunsul corect pentru a-l putea compara cu răspunsul clientului.

```
struct a
{
    char nume[1002];
    int joc, id, scor;
    int incepe_joc;
    int descriptor;
    int nrjuc, nrint, nr_cod;
    char cod[1002];
    time_t begin, end;
    char raspuns[100];
    char alegere[10];
    int intrebare, timp, acum;
    int jucatori[10000], in_timp_int;
    int nrint0[10], nrint1[10], nrint2[10], nrint3[10], nrint4[10];
    char raspbun[100];
} th[10000];

struct c
{
    char intreba[1000];
    int jucator, nr_jucatori, nr_intre, numar, nr_intpuse;
    int juca[10000];
    int jucatori[10000];
    char cod[1100];
} coduri[10000];
```

Fig. 4. Structuri

```
static int callback(void *data, int argc, char **argv, char **arg)
{
    int i;
    char *str = (char *)data;
    for (i = 1; i < argc - 1; i++)
    {
        strcat(str, argv[i]);
        strcat(str, "\n");
    }
    return 0;
}
static int callbackrasbun(void *data, int argc, char **argv, char **arg)
{
    int i;
    char *str = (char *)data;
    for (i = argc - 1; i < argc; i++)
    {
        strcat(str, argv[i]);
    }
    return 0;
}
```

Fig. 5. Fucntie care apeleaza baza de date

```
int initializare2(int client)
{
    int nr = th[client].nrjuc;
    int nr2 = th[client].nr_cod;
    for (int ii = 0; ii <= nr; ii++)
    {
        th[coduri[th[client].nr_cod].jucatori[ii]].incepe_joc = -1;
        th[coduri[th[client].nr_cod].jucatori[ii]].nrint = -1;
        th[coduri[th[client].nr_cod].jucatori[ii]].scor = 0;
        th[coduri[th[client].nr_cod].jucatori[ii]].joc = -1;
        bzero(th[coduri[th[client].nr_cod].jucatori[ii]].alegere, 10);
        th[coduri[th[client].nr_cod].jucatori[ii]].nr_cod = -1;
    }
    for (int ii = 0; ii <= nr; ii++)
    {
        coduri[nr2].jucatori[ii] = -1;
    }
}
```

Fig. 6. Functia care iniţializează toţi clienţii după ce jocul s-a terminat

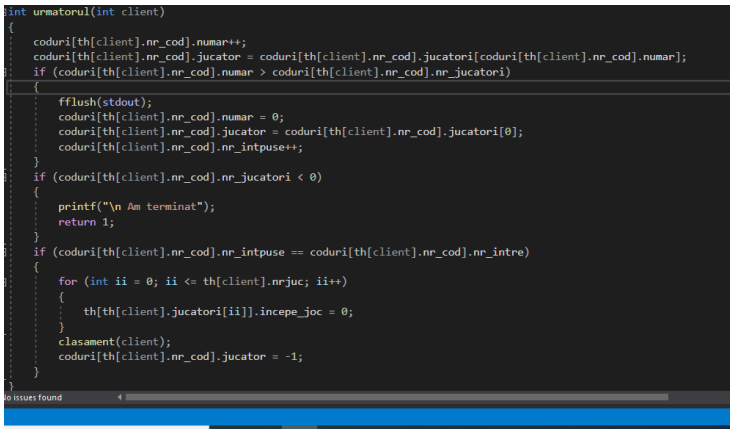

Fig. 7. Functia care trimite întrebarea următorului client

5 Concluzii

Acest joc poate fi îmbunătăţit prin crearea unei baze de date în care este memorat fiecare jucător, iar acesta îşi poate vedea clasamentul jocurile de până acum, având diagrame în care este prezentat progresul sau regresul acestuia, dar şi domeniile la care se pricepe cel mai bine, câte răspunsuri corecte, dar şi greşite are.De asemenea, o altă îmbunătăţire a soluţiei propuse ar putea fi faptul că fiecare client sa aibe pentru fiecare joc un număr de trei soluţii ajutătoare şi anume: eliminarea a două variante de răspuns, schimbarea întrebării, dar si arătarea statisticii răspunsurile tuturor clienţilor care au primit această întrebare.

Această aplicaţie ar putea fi utililă în viaţa realaă deoarece este un mod plăcut de petrecere a timpului liber, învăţând totodată si lucruri noi din diverse domenii.

6 Bibliografie

References

1. https://profs.info.uaic.ro/~computernetworks/files/NetEx/S12/ServerConcThread/servTcpConcTh2.c
2. https://profs.info.uaic.ro/~computernetworks/files/7rc_ProgramareaInReteaIII_Ro.pdf
3. https://profs.info.uaic.ro/~computernetworks/files/5rc_ProgramareaInReteaI_ro.pdf
4. https://profs.info.uaic.ro/~computernetworks/files/NetEx/S12/ServerConcThread/cliTcpNr.c
5. https://profs.info.uaic.ro/~gcalancea/Laboratorul_9.pdf
6. https://stackoverflow.com/questions/31146713/sqlite3-exec-callback-function-clarification?fbclid=IwAR2b5kFpUOZx-0on6M7HLp9ALWPv_S4t5GkLMe6qSPcL9krn3fZv2Pi8EeM
7. https://www.geeksforgeeks.org/sql-using-c-c-and-sqlite/?fbclid=IwAR2ULns4il5r21LsOlLl69h6Ax7gaA4Yssz6NbKfQIY_ouWQqwaehDTe_I
8. https://www.youtube.com/watch?v=dFzJ4UPNL1w&fbclid=IwAR3oQu7bM6Kwmn4karP7B84BQ7u2QukVagrK2MfUM-lG9sE57yL9Zd
9. https://sqlite.org/cli.html
10. https://www.linuxjournal.com/article/1363
11. https://www.youtube.com/watch?v=g-KDOH_uqPk
12. https://prognotes.net/gtk-glade-c-programming/
13. https://prognotes.net/2016/03/gtk-3-c-code-hello-world-tutorial-using-glade-3/
14. https://unix.stackexchange.com/questions/129355/what-is-difference-between-gtk-and-qt-applications
15. https://www.reddit.com/r/linuxmasterrace/comments/6ee9sy/gtk_vs_qt_what_do_you_prefer_and_why_2017_edition/
16. https://stackoverflow.com/questions/43078613/c-non-blocking-reading

Referinţe

[1]https://www.extrahop.com/resources/protocols/tcp/

[2]https://en.wikipedia.org/wiki/Thread_(computing)

[3]https://ro.wikipedia.org/wiki/GTK

[4]https://glade.gnome.org/

[5]https://www.sqlite.org/about.html