Documentatie - QuizzGame (B)

Dascălu Rareș-Vasilică, Grupa A5, Anul 2

07.12.2022

Abstract. Acest document are ca scop prezentarea tehnologiilor utilizate, a arhitecturii si a detaliilor de implementare a proiectului propus QuizzGame (B)

Introducere:

Scopul acestui proiect este de a crea un server ce permite unui numar nelimitat de clienti sa se conecteze si sa raspunda la diferite intrebari. Serverul va retine scorurile fiecarui client si le va incrementa pentru fiecare raspuns corect.

Motivatie:

Am ales acest proiect deoarece este un concept foarte asemanator cu site-ul https://kahoot.it/. Fiind un concept cunoscut, am vrut sa aflu cum functioneaza si cum se creeaza o aplicatie de acest tip. Alt motiv pentru care am ales acest proiect este ca imi place sa imi testez creativitatea in crearea jocurilor.

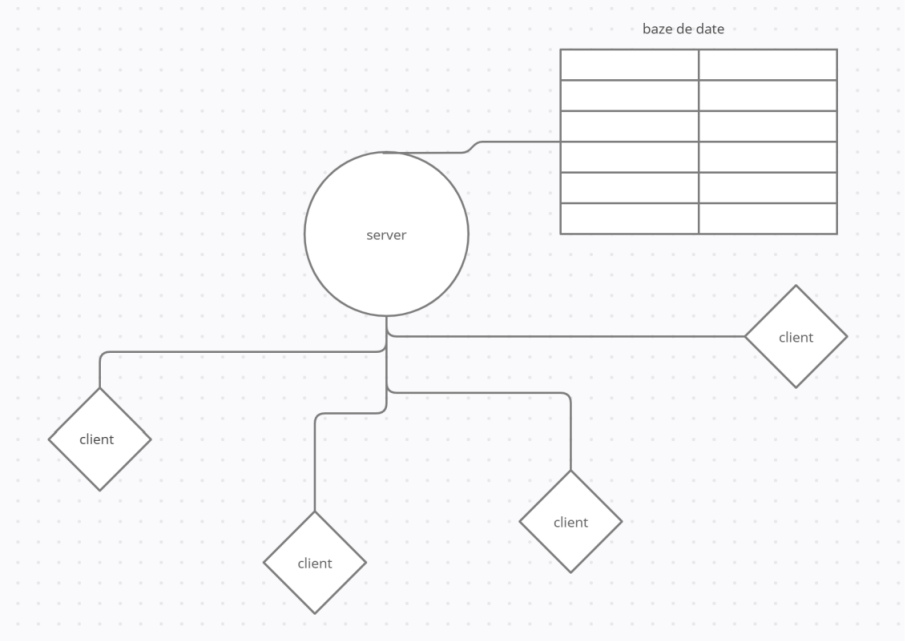
Tehnologii utilizate:

Aplicatia trebuie sa permita un numar nelimitat de clienti, de aceea protocolul folosit este de tip TCP/IP concurent, fiecare client primind la conectare un thread creat de server, in care isi va desfasura intreaga activitate. In comparatie cu protocolul UDP, acesta asigura integritatea mesajelor, precum si faptul ca mesajele ajung la destinatie in ordine cronologica.

Deoarece C/C++ este un limbaj ce se recomanda pentru proiectarea unor astfel de proiecte si ofera o varietate de functii ce ma pot ajuta sa duc la bun sfarsit acest proiect, acesta va fi utilizat pentru crearea proiectului.

Deoarece trebuie sa tinem evidenta clientilor si a intrebarilor, vom folosi o baza de date relationala, SQLite, versiunea a 3-a.

Arhitectura aplicatiei



Detalii de implementare:

In prezent programul contine 4 fisiere (client.c, client.h, server.c, server.h, baza.db). Fisierul client.c contine clientul ce primeste de la server textul afisat de client ce specifica conectarea. Clientul trimite inapoi catre server numele si parola, serverul le verifica, si trimite inapoi un raspuns pentru a spune clientului daca a reusit sa se conecteze cu succes sau nu.

Dupa conectare, clientul primeste de la server un meniu, oferindu-i optiunea de a iesi din joc sau de a incepe jocul. Odata ce jocul este pornit, clientul primeste de la server intrebarile si raspunsurile, avand timp cate 10 secunde pentru fiecare intrebare. Fiecare client isi calculeaza propriul scor, ce este ulterior trimis la server pentru a nominaliza un castigator.

```c
void joc()
{
        asteapta1();
        i=0;
        while(i<10)
        {

        if(read( socket_descriptor,&question,sizeof(question)) < 0){
            eroareCitire();
            exit(0);
        }

        system("clear");
        printf("%s \n",question.question);
        fflush(stdout);
        printf("%s\n",question.r1);
        fflush(stdout);
        printf("%s\n",question.r2);
        fflush(stdout);
        printf("%s\n",question.r3);
        fflush(stdout);
        printf("%s\n",question.r4);
        fflush(stdout);

        sec10();

        if(poll(&timer, 1, 10000))
            {
                scanf("%s", answer);

                if(strcmp(answer,"exit")==0)
                    parasit();
            }
        else
            {
                timp0();
                sleep(1);
```

```c
        }
        while(strcmp(answer,"a")!=0 && strcmp(answer,"b")!=0 && strcmp(answer,"c")!=0 && strcmp(answer,"d")!=0 && strcmp(answer,"")!=0)
            {
                formulatGresit();
            }


        if(strcmp(answer,"")!=0 && strcmp(answer,question.rc)!=0)
            {
                gresit();
                sleep(1);
            }
        else if(strcmp(answer,"")!=0 && strcmp(answer,question.rc)==0)
            {
                corect();
                sleep(1);
            }
        answer[0]='\0';
        i++;
        }

        if(write( socket_descriptor,&scor,sizeof(scor)) <= 0)
        {
            system("clear");
            eroareScriere();
            exit(0);
        }

        asteapta2();

        if(read(socket_descriptor,castigator,sizeof(castigator)) < 0)
        {
            system("clear");
            eroareCitire();
            exit(0);
        }
        loc1(castigator);
}
```

Fisierul server.c contine serverul ce creaza socket-ul pentru comunicarea cu clientii. Am atasat sererul cu functia bind(), iar cu functia listen() am pus serverul sa asculte clientii ce doresc sa se conecteze. Pentru fiecare client a fost creat cate un thread pentru conectarea clientilor si executia jocului in sine pentru fiecare client in parte.

In momentul in care este apasata comanda ”CTRL + C” serverul te va intreba daca doresti sa inchizi serverul pentru a nu fi inchis din greseala.

```c
void  ctrlC(int sig)
{
    char  Z;
    signal(si (char [55])"\nSUNTETI SIGUR CA DORITI SA INCHIDETI SERVERUL? [Y/N]\n"
    printf("\nSUNTETI SIGUR CA DORITI SA INCHIDETI SERVERUL? [Y/N]\n");
    Z = getchar();
    if (Z == 'y' || Z == 'Y')
        exit(0);
    else
        signal(SIGINT, ctrlC);
    getchar();
}
```

Inainte de crearea socket-ului in server, acesta citeste intrebarile din baza de date, pentru a fi sigur ca poate incepe un joc dupa conectarea clientilor.

```c
void citireIntrebari()
{
    char *msgEroare = 0;
    struct sqlite3 *dataBase;
    int result = sqlite3_open("baza.db", &dataBase);

    if(result)
    {
        fprintf(stderr, "(intrebari) N-AM PUTUT DESCHIDE BAZA DE DATE %s\n",
        sqlite3_errmsg(dataBase));
        sqlite3_close(dataBase);
        exit(0);
    }
    else
    {
      fprintf(stderr, "(intrebari) AM DESCHIS BAZA DE DATE CU SUCCES\n");
    }

    result = sqlite3_exec(dataBase, "SELECT * FROM INTREBARI", callback, 0, &msgEroare);


    if(result == SQLITE_OK)
    {
       fprintf(stderr, "AM SELECTAT INTREBARILE CU SUCCES\n");
       sqlite3_close(dataBase);
    }
    else
    {
        fprintf(stderr, "NU AM REUSIT SA SELECTAM INTREBARILE CU SUCCES\n");
        fprintf(stderr, "EROARE : %s\n", msgEroare);
        sqlite3_free(msgEroare);
        sqlite3_close(dataBase);
        sleep(1);
        exit(0);
    }
}
```

Pentru conectarea clientului am folosit un subprogram ce verifica in baza de date existenta utilizatorului trimis de client, si corectitudinea parolei.

```c
int VerificareLogare(const char* id, const char* parola)
{
    SQL(id,parola);
    char *msgEroare = 0;
    struct sqlite3 *dataBase;
    int result = sqlite3_open("baza.db", &dataBase);

    if(result)
    {
        fprintf(stderr, "N-AM PUTUT DESCHIDE BAZA DE DATE %s\n",
        sqlite3_errmsg(dataBase));
        sqlite3_close(dataBase);
        return 0;
    }
    else
    {
      fprintf(stderr, "AM DESCHIS BAZA DE DATE CU SUCCES\n");
    }

    struct sqlite3_stmt *statement;
    result = sqlite3_prepare_v2(dataBase, SQL_comanda, -1, &statement, NULL);

    if(result == SQLITE_OK)
    {
      if (sqlite3_step(statement) == SQLITE_ROW)
      {
          sqlite3_finalize(statement);
          sqlite3_close(dataBase);
          return 1;
      }
      else
      {
          fprintf(stderr, "EROARE SQL %s\n", msgEroare);
          sqlite3_free(msgEroare);
          sqlite3_close(dataBase);
          return 0;
      }
    }
}
```

Dupa conectarea clientului si selectarea optiunii de incepere a jocului de catre client, serverul va astepta pana toti utilizatorii conectati selecteaza inceperea jocului, in caz contrar, va astepta pana se deconecteaza cei ce nu au selectat o optiune din meniu.

Odata ce a inceput jocul, serverul va trimite aleatoriu catre clienti 10 intrebari din lista de intrebari gasite in baza de date, incluzand si variantele de raspuns, si raspunsul corect. Dupa ce utilizatorii au terminat de raspuns la toate intrebarile, serverul va primi punctajele fiecarui client, urmand sa calculeze clientul cu cel mai mare punctaj, si ii va trimite numele castigatorului fiecarui client in parte.

```
int executie_joc(struct inf_thread thClient)
{
    int folosit;
    int intrebariUzate[11] = {0};
    int m;
    int q;
    punctajMaxim = 0;
    id_punctajMaxim = 0;
    contorIntrebari = 0;
    int punctaj;
    char id[128];

     m=1;
    while(m<=10)
    {
        while(1)
        {
            iar:
            folosit = 0;
            srand(time(NULL));
            n = rand() % 20;

            q=1;
            while(q<=contorIntrebari)
            {
                if(n == intrebariUzate[q])
                    {
                        folosit = 1;
                        break;
                    }
                q++;
            }
            if( folosit == 1 )
                goto iar;
            else
            {
                intrebariUzate[++contorIntrebari] = n;
                break;
            }
        }
        if(write(thClient.descriptor,&intrebare[n],sizeof(intrebare[n])) < 0)
            eroareScriere(thClient.th_id);
    m++;
    }

    if(read(thClient.descriptor,&punctaj,sizeof(punctaj)) <= 0)
    {
                eroareCitire(thClient.th_id);
                return errno;
    }

    punctaje[thClient.th_id] = punctaj;
    ct_finish++;

    ast:
    if(ct_finish == conectati)
    {
        Winner();
        if(write(thClient.descriptor,&castigator[id_punctajMaxim],sizeof(castigator[id_punctajMaxim])) <= 0)
                eroareScriere(thClient.th_id);
    }
    else
        goto ast;
        afisareWinner();
}
```

Concluzii:

Serverul prelucreaza toate datele din baza de date si cele primite de la client, iar in client doar se afiseaza datele primite de la server, trimite inapoi raspunsurile si calculeaza punctele fiecarui client. Pe viitor as dori sa implementez functii care sa permita adaugarea de intrebari direct din server si alta care sa permita inregistrarea si clientilor care nu au conturi memorate in baza de date.

Bibliografie:

1) https://profs.info.uaic.ro/ computernetworks
2) http://zetcode.com/db/sqlitec/
3) https://www.quora.com/How-do-I-put-a-time-limit-to-the-scanf-function