# Nu te supara, frate!-NTSF

Galan Andrei-Iulian, an 2, grupa A4

Universitatea "Al. I. Cuza", Informatică Iaşi, RO
https://profs.info.uaic.ro/~computernetworks/

**Abstract.** Scopul aceastei a documenţii este de a introduce subiectul abordat, unde se vor prezenta: motivaţia alegerii proiectului, implementarea lui, tehnologiile utilizate în cadrul acestuia, arhitectura aplicaţiei, precum şi detalii de implementare şi scenarii de utilizare, iar în final vom concluziona cu ajutorul motivaţiei alegerii proiectului, precum şi cu îmbunătăţirile viitoare posibile.

## 1 Introducere

Proiectul ales, **NTSF-"Nu te supăra, frate!"**, propune dezvoltarea unei aplicaţii/joc ce se bazează pe tipul client/server ce permite între doi, trei sau patru utilizatori conectaţi să joace bine-cunoscutul joc "Nu te supăra frate!", oferindu-le posibilitatea de a-şi alege un nume, ca la finalul jocului să obţină un număr de puncte, unde in final, toate punctajele obţinute din toate jocurile să fie introduse intr-o bază de date şi formându-se astfel un top al jucătorilor. Jucătorii vor avea totodată şi posibilitatea de a pune pauză la joc în cazul in care unul dintre jucători doreşte să-l întrerupă temporar, iar după terminarea pauzei jocul va reveni de unde a rămas. În cazul în care unul dintre jucători nu ştie regulile jocului, va putea folosi comanda /help si i se vor afişa regulile jocului.

### 1.1 Motivaţia

**Nu te supăra, frate!** este un joc ce poate fi jucat în 2-4 jucători. Pentru a putea juca acest joc este nevoie de o planşă specială, un zar şi câte 4 pioni de aceaşi culoare pentru fiecare jucător. Aşadar, cu ajutorul internetului reuşim ca un joc vechi, jucat pe o tabla de carton, îndragit de mine şi cel mai probabil de alţi oameni, care l-au jucat când erau copii sau poate încă îl joacă şi astązi ocazional, să conecteze maxim 4 jucatori aflaţi la distanţă unul de altul şi să joace in timp real. Astfel că, pentru a pune în practică cunoştinţele acumulate, necesare programării cu ajutorul protocoalelor TCP/IP, precum şi de a dezvolta experienţa de a programa cu ajutorul threadurilor/procese-copil şi cu ajutorul socketurilor, am ales crearea unui joc al copilăriei mele "Nu te supăra, frate!", din cauza nostalgiei si a pasiunii mele de a juca jocuri pe calculator, cât şi fizic.

## 2 Tehnologiile utilizate

Deoarece acest joc, necesită ajutorul internetului, pentru conectarea mai multor jucători aflaţi la distanţă la un anumit server, am ales folosirea protocolului de tip TCP/IP, astfel că utilizatorii pentru a efectua diferite operaţiuni în paralel, serverul va trebui să fie unul concurent.

Însă motivele pentru care am ales protocolul TCP/IP în acest proiect, sunt :

-Oferă încredere şi asigură livrarea ordonată, fără erori a unui flux de octeţi de la un computer la altul aflat în reţea. Acesta este folosit în general de aplicaţii care au nevoie de o confirmare atunci când primesc date.

-Protocolul se ocupă de controlul fluxului de informaţii pentru ca destinatarul să nu fie încarcat de mesaje, astfel încat să nu poată să le proceseze pe toate, astfel că octeţii sunt primiţi în aceeaşi ordine cum au fost fost trimişi.

-Întrucât aplicaţia necesită o comunicare bidirecţională, cu ajutorul nivelului reţea, pachetele sunt transmise de la o maşină la alta cu ajutorul adresei **IP** si a unui **PORT**, comunicarea fiind full-duplex.

Astfel că, pentru acest proiect, am ales utilizarea protocolului TCP, în ciuda faptului că nu este cel mai rapid protocol, oferă singuraţă în transmiterea datelor, deoarece cel mai important lucru, este ca toti jucătorii să-şi execute cu succes mutarea în fiecare tură a jocului, iar dacă o mutare nu s-ar executa cu succes şi ar exista o pierdere de date, acel jucator ar fi dezavantajat. Deci consider că în cazul acestui proiect este mult mai important trasmiterea informaţiilor cu succes decât să fie mai rapid.

### 2.1 Baza de date

Pentru stocarea informaţiilor de tip user, punctaje si poziţiile pieselor în cazul unei întreruperi a jocului pentru fiecare client în parte, o bază de date va trebui folosită. Astfel ca, la momentul conectării unui jucator, acesta isi poate vizuliza totalul punctajelor obtinute de-a lungul jocurilor si totodată să îşi afişeze un top a tuturor jucatorilor din bază de date.

Am ales folosirea unui tip de bază de date embedded, ce nu necesită un anumit server sau o altă configurare, denumit **SQLite**, care totodată oferă uşurinţă în implementarea unui SGBD.

## 3 Arhitectura aplicaţiei

Componentele proiectului sunt: un server, cu mai multe sesiuni de joc in acelasi timp, care contin 2-4 clienti si o bază de date, ce reţine username-ul fiecărui jucător, punctajele obţinute de fiecare jucător de-a lungul timpului si poziţia fiecărei piese în cazul unei întreruperi a jocului.

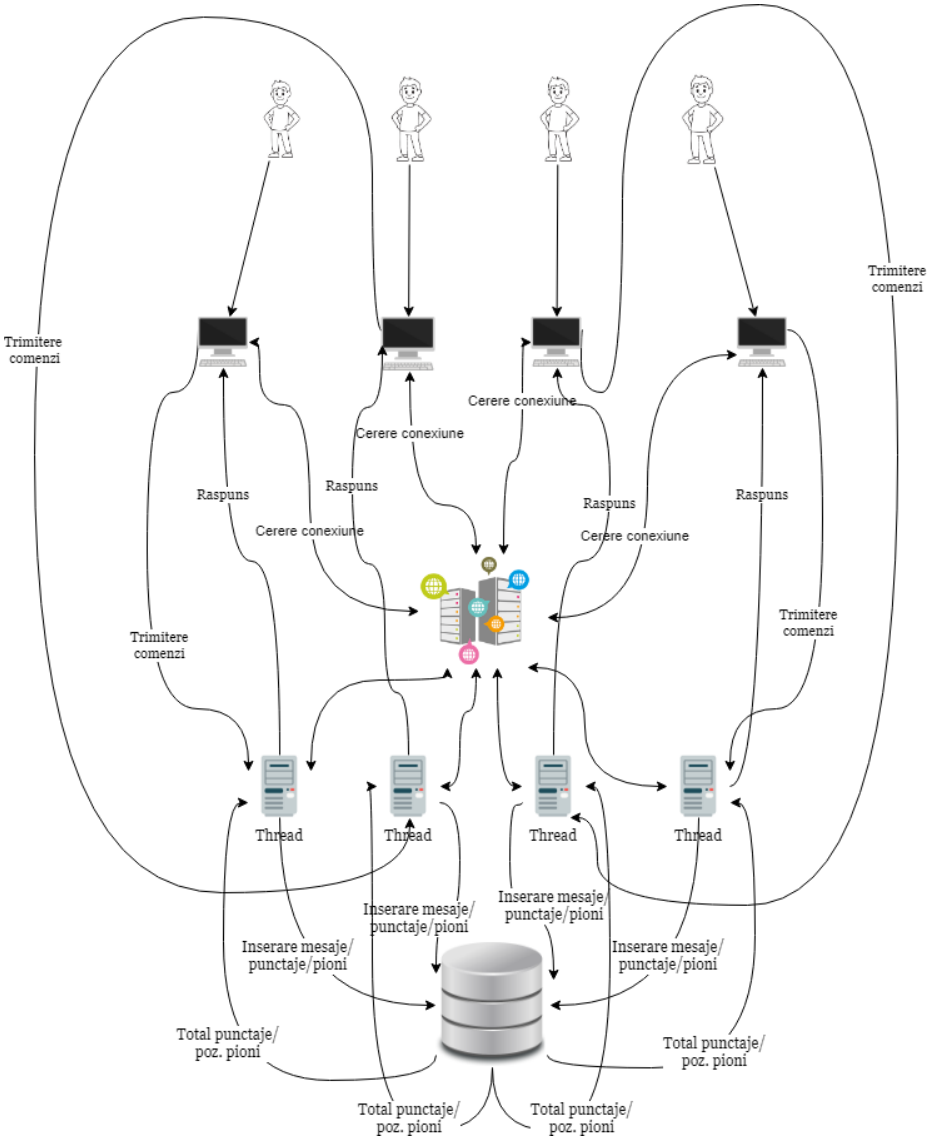



Fig. 1: Diagrama cu arhitectura unei sesiuni de joc

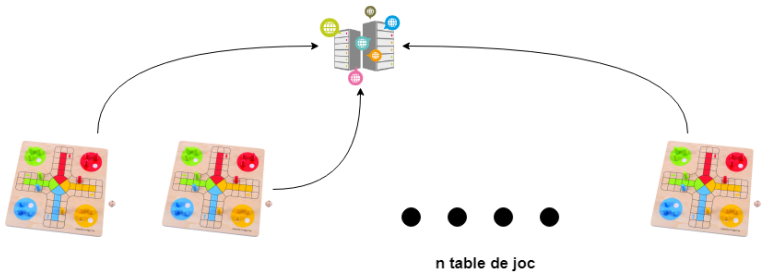



Fig. 2: Diagrama cu arhitectura aplicaţiei

În prima diagrama de mai sus, este prezentată comunicarea celor 4 jucatori prin intermediul serverului si a thread-urilor. Deci, dacă un client trimite o cerere de conectare utilizând protocolul TCP/IP si socketurile, serverul , dacă acceptă cererea de conectare, se foloseşte de thread-uri, îndeplinind astfel cererile fiecărui jucător, dar în acelaşi timp asigură şi stocarea puntajelor totale fiecărui jucător şi poziţiile pionilor la fiecare moment de timp. Iar in cea de-a doua diagrama se poate observa faptul ca fiecare tabla reprezinta cate o sesiune de joc, aflata pe acelasi server.

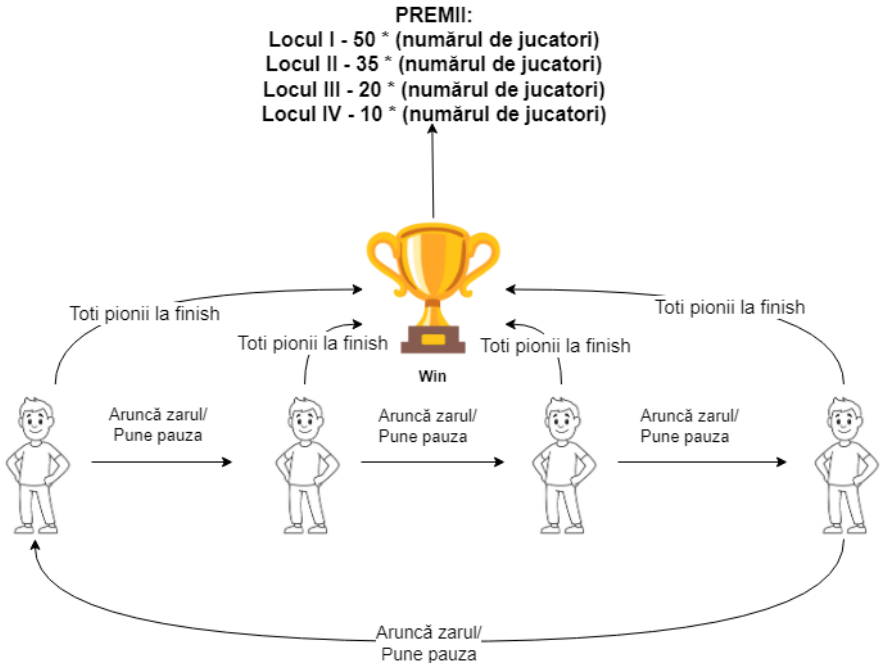



Fig. 3: Diagrama cu desfăşurarea jocului

În diagrama de mai sus, este prezentată desfăşurarea jocului între 4 playeri, comenzile posibile în timpul jocului şi momentul în care jocul se va termina, fiecare jucător obţinând un număr de puncte aferent locului său, care sunt adăugate în baza de date.

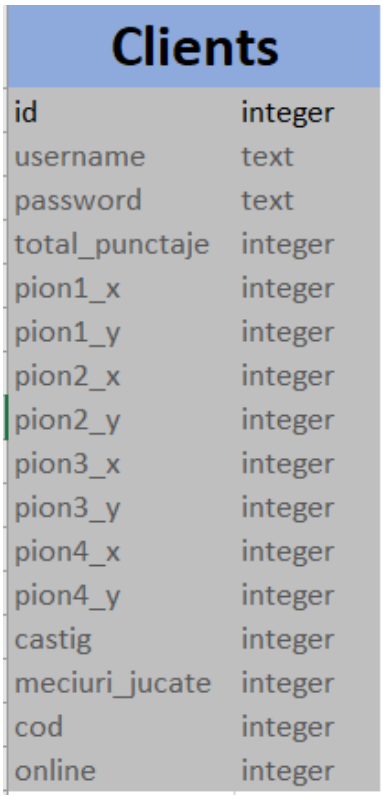



Fig. 4: Tabelul bazei de date

## 4 Detalii de implementare

Pentru început, se vor prezenta paşii de conectare pentru a participa la jocul "Nu te supăra, frate!":

1. Serverul este pornit şi aşteaptă conexiuni la IP-ul propriu şi PORT-ul desemnat;

2. Clientul este pornit, iar adresa şi PORT-ul sunt specificate;

3. Clientul trimite o cerere de conectare către server, cerere ce poate fi acceptată sau respinsa, iar in caz de acceptare serverul va crea un thread pentru comunicarea cu clientul. O data ce sesiunea de joc a ajuns la un numar de 2 jucatori, va incepe un timer de 30 de secunde pentru conectarea celorlalti jucatori. Daca sesiunea de joc nu va ajunge la 4 jucatori, iar timpul s-a scurs, jocul va incepe. In schimb daca se vor conecta alti jucatori la serverm acestia vor fi pusi intr-o alta sesiune de joc diferita de prima.

```c
void setTimeout(int milliseconds)
{
    if (milliseconds <= 0) {
        fprintf(stderr, "Count milliseconds for timeout is less or equal to 0\n");
        return;
    }
    int milliseconds_since = clock() * 1000 / CLOCKS_PER_SEC;
    int end = milliseconds_since + milliseconds;
    do {
        milliseconds_since = clock() * 1000 / CLOCKS_PER_SEC;
    } while (milliseconds_since <= end);
}
```

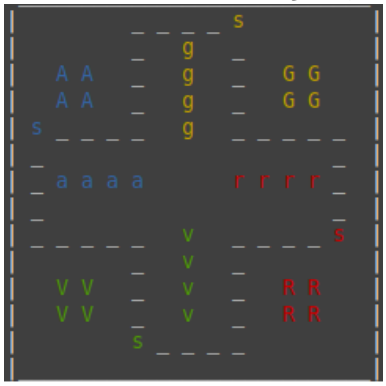
4. După ce toti clienţii s-au conectat la server cu succes, pentru fiecare jucător în parte va apărea un mesaj de introducere, unde il va intreba daca doreste sa vizualizeze regulile jocului, iar apoi va trebui sa isi aleaga username-ul şi parola pentru a se conecta la joc sau să se înregistreze în cazul în care nu se află deja in baza de date.(acesta va trebui sa isi introduca un username, parola si un cod de 4 caractere, care il va ajuta in cazul in care va dori sa isi schimbe parola). In cazul in care un jucator si-a uitat parola sau doreste sa si-o schimbe, acesta va avea posibilitatea de a introduce una noua, cu ajutorul codului introdus de el la prima inregistrare. Aceasta operatie este posibila cu ajutorul functiei **updateparola()**. Daca in cazul in care un jucator este incearca sa se conecteze cu un username deja conectat intr-o sesiune de joc, acesta va primi o eroare si va trebui sa incerce alt username.

```c
void updateparola(char nume[20],char parola[100], sqlite3 *db,int fd)//in cazul in care un jucator isi reseteaza parola, atunci cand isi introduce o parola noua, acesta functie
//va face update in baza de date
{
    char comanda[1024];
    int bdd;
    char *errmsg = 0;
    char msg[100];
    struct sqlite3_stmt *selectstmt;
    const char* data = "Callback function called";
    fflush(stdout);
    bzero(comanda,1024);
    strcpy(comanda , "UPDATE JUCATORI SET PAROLA='parola' WHERE USERNAME='user';");
    sprintf(comanda,  "UPDATE JUCATORI SET PAROLA='%s' WHERE USERNAME='%s';",parola,nume);
    bdd=sqlite3_exec(db, comanda, callback, (void*)data, &errmsg);
    if (bdd == SQLITE_OK)
        printf("Operation done successfully\n");
    sqlite3_close(db);
}
```

5. După logare, fiecare client îşi va alege pe rand o culoare a pionilor săi(pentru a evita evenimentul de suprapunere, astfel că dacă un jucător şi-a ales o culoare deja aleasă va primi un mesaj de eroare şi acelaşi lucru v-a primi şi în cazul în care introduce o culoare invalidă).

6. După alegerea culorii, clientul va crea un thread ce se va ocupa de citirea mesajelor de la server, ce i se vor afişa pe ecran(acestea fiind în general, îndemnul folosirii unor comenzi, cum ar fi: aruncarea zarului, întreruperea temporară a jocului sau a unor mesaje de anunţare, atenţionare: "Este rândul tău!").

După alegerea culorii, jocul va începe şi se va afişa o tablă de joc, care se va actualiza la fiecare mişcare, tablă creată cu ajutorul funcţiei **Createtable()**.



-caraterul '_' semnifică spatiu liber
-caracterul 's' seminfică poziţia de start a pieselor
-caraterele 'G','A','V','R' seminifică piesele jucătorilor
-caracterele 'g','a','v','r' seminifică poziţiile finale unde trebuie să ajungă piesele

La fiecare aruncare de zar(cu ajutorul functiei **aruncazar()**), după ce jucatorul şi-a ales ce pion să îşi mute, cu ajutorul funcţiei **play()**, se verifică dacă mişcarea este posibilă, iar în caz afirmativ pionul ales îşi va actualiza poziţia pe tablă, iar în cazul în care pionul jucătorului, trece peste un pion al unui adversar("îl mănâncă") se va actualiza şi poziţia pionului "mâncat", fiind pus pe locul lui iniţial pe tablă.

```
int aruncazar(){
    srand (time(NULL));
    int zar = rand() % 6 + 1;
    return zar;
}
```

```
int parcurgere_Gi[50] = {1, 1, 1, 1, 0, 0, 0, 0, 1, 1, 1, 1, 0, 0, 0, 0, 1, 1, 1, 1, 0, 0, 0, 0, -1, -1, -1, -1, 0, 0, 0, 0, -1, -1, -1, -1, 0, 0, 0, 0, -1, -1, -1, -1, 0, 0, 1, 1, 1, 1};
int parcurgere_Gj[50] = {0, 0, 0, 0, 2, 2, 2, 2, 0, 0, 0, 0, -2, -2, -2, -2, 0, 0, 0, 0, -2, -2, -2, -2, 0, 0, 0, 0, -2, -2 , -2, -2, 0, 0, 0, 0, 2, 2, 2, 2, 0, 0, 0, 0, 2, 2, 0, 0, 0, 0};
int parcurgere_Ri[50] = {0, 0, 0, 0, 1, 1, 1, 1, 0, 0, 0, 0, -1, -1, -1, -1, 0, 0, 0, 0, -1, -1, -1, -1, 0, 0, 0, 0, -1, -1, -1, -1, 0, 0, 0, 0, 1, 1, 1, 1, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0};
int parcurgere_Rj[50] = {-2, -2, -2, -2, 0, 0, 0, 0, -2, -2, -2, -2, 0, 0, 0, 0, -2, -2, -2, -2, 0, 0, 0, 0, 2, 2 , 2, 2, 0, 0, 0, 0, 2, 2, 2, 2, 0, 0, 0, 0, 2, 2, 2, 2, 0, 0, -2, -2, -2, -2};
int parcurgere_Vi[50] = {-1, -1, -1, -1, 0, 0, 0, 0, -1, -1, -1, -1, 0, 0, 0, 0, -1, -1, -1, -1, 0, 0, 0, 0, 1, 1, 1, 1, 0, 0, 0, 0, 1, 1, 1, 1, 0, 0, 0, 0, 1, 1, 1, 1, 0, 0, -1, -1, -1, -1};
int parcurgere_Vj[50] = {0, 0, 0, 0, -2, -2, -2, -2, 0, 0, 0, 0, 2, 2 , 2, 2, 0, 0, 0, 0, 2, 2, 2, 2, 0, 0, 0, 0, -2, -2, -2, -2, 0, 0, 0, 0, -2, -2, -2, -2, 0, 0, 0, 0};
int parcurgere_Ai[50] = {0, 0, 0, 0, -1, -1, -1, -1, 0, 0, 0, 0, 1, 1, 1, 1, 0, 0, 0, 0, 1, 1, 1, 1, 0, 0, 0, 0, 1, 1, 1, 1, 0, 0, 0, 0, -1, -1, -1, -1, 0, 0, 0, 0, -1, -1, -1, -1, 0, 0, 0, 0};
int parcurgere_Aj[50] = {2, 2, 2, 2, 0, 0, 0, 0, 2, 2, 2, 2, 0, 0, 0, 0, 2, 2, 2, 2, 0, 0, 0, 0, -2, -2, -2, -2, 0, 0, 0, 0, -2, -2, -2, -2, 0, 0, 0, 0, -2, -2, -2, -2, 0, 0, 2, 2, 2, 2};
```

Imaginea de mai sus, reprezinta miscarile pionilor pe axa X si pe axa Y(pentru fiecare culoare), pentru a ajunge la finish.

Funcţia **Pauza()**, are rolul de a opri temporar starea jocului, astfel ca toate informaţiile legate de poziiţile pionilor pe tablă sunt salvate cu ajutorul functiei **updatepozitii()**, în baza de date. După ce jocul este reluat, informaţiile legate de fiecare pion sunt luate din baza de date a fiecărui jucător, cu ajutorul functiei **restabilire_pozitii()**, iar toţi pionii sunt plasaţi exact în aceleaşi poziţii. Astfel, rezultă faptul că nicio informaţie nu a fost pierdută în această pauză.

```c
void updatepozitii(piesa p, sqlite3 *db,int fd,char nume[20])//introducerea pozitiilor pieselor in baza de date
{
    char comanda[1024];
    int bdd;
    char *errmsg = 0;
    char msg[100];
    struct sqlite3_stmt *selectstmt;
    const char* data = "Callback function called";
    fflush(stdout);
    bzero(comanda,1024);
    strcpy(comanda , "UPDATE JUCATORI SET P1_X=p1_x,P1_Y=p1_y,P2_X=p2_x,P2_Y=p2_y,P3_X=p3_x,P3_Y=p3_y,P4_X=p4_x,P4_Y=p4_y WHERE USERNAME='user';");
    sprintf(comanda, "UPDATE JUCATORI SET P1_X=%d,P1_Y=%d,P2_X=%d,P2_Y=%d,P3_X=%d,P3_Y=%d,P4_X=%d,P4_Y=%d WHERE USERNAME='%s';", p.piesa[0].x,p.piesa[0].y,p.piesa[1].x,p.piesa[1].y,p.piesa[2].x,p.piesa[2].y,p.piesa[3].x,p.piesa[3].y,nume);
    bdd=sqlite3_exec(db, comanda, callback, (void*)data, &errmsg);
        if (bdd == SQLITE_OK)
        printf("Operation done successfully\n");
    sqlite3_close(db);
}
```

```c
void restabilire_pozitii(piesa p, sqlite3 *db,int fd,char nume[20])//in cazul in care se pune pauza acesta functie este apelata pentru a restabili pozitiile pieselor
//luand informatiile din baza de date
{
    sqlite3_stmt* stmt;
    char* errMesg = 0;
    char sql_stmt[200];
    strcpy(sql_stmt,"SELECT P1_X,P1_Y,P2_X,P2_Y,P3_X,P3_Y,P4_X,P4_Y from JUCATORI WHERE USERNAME='user';");
    sprintf(sql_stmt,"SELECT P1_X,P1_Y,P2_X,P2_Y,P3_X,P3_Y,P4_X,P4_Y from JUCATORI WHERE USERNAME='%s';",nume);
    int bdd;
    bdd = sqlite3_prepare_v2(db, sql_stmt, -1, &stmt, 0);

    if (bdd != SQLITE_OK) {
        printf("\nUnable to fetch data");
        sqlite3_close(db);
        //return 1;
    }
    char result1[100];
    bzero(result1,100);
    char result2[100];
    bzero(result2,100);
    int i=0;
    int j=0;
    while (sqlite3_step(stmt) == SQLITE_ROW) {
        while (j<4) {
            bzero(result1,100);
            bzero(result2,100);
            strcat(result1,(char*)sqlite3_column_text(stmt, 0));
            strcat(result1,"\0");
            strcat(result2,(char*)sqlite3_column_text(stmt, i+1));
            strcat(result2,"\0");
            p.piesa[j].x=result1[0]-'0';
            p.piesa[j].y=result2[0]-'0';
            i+=2;
            j++;
        }
    }
    sqlite3_finalize(stmt);
    sqlite3_close(db);
}
```

Functia **verificare_mancat()**, verifica daca pionul mutat este plasat peste alt pion, al unui adversar. In caz afirmativ, acel pion este mutat inapoi in casa.

```
void Verificare_mancat(int X,int Y,int cul,int x)//verifica daca un pion este mancat de altul
{
    int ok=1;
    if(strchr("GRVA",t[x].table[X][Y]))
        for(int i=0;i<4 && ok==1;i++)
            if(t[x].player[i].culoare==t[x].table[X][Y] && i!=cul)
              {
                for(int j=0;j<4;j++)
                    if(t[x].player[i].piesa[j].x ==  X  && t[x].player[i].piesa[j].y == Y)
                    {
                        t[x].player[i].piesa[j].poz=-1;
                        t[x].player[i].piesa[j].x= t[x].player[i].piesa[j].x1;
                        t[x].player[i].piesa[j].y=t[x].player[i].piesa[j].y1;
                        fflush(stdout);
                        t[x].table[t[x].player[i].piesa[j].x][t[x].player[i].piesa[j].y]=t[x].player[i].culoare;
                        int k=0;
                        for(int x=0;x<4;x++)
                            if(t[x].player[i].piesa[x].poz<0)
                                k++;
                        if(k==4)
                            t[x].player[i].iesit=0;
                        ok=0;
                    }
              }
}
```

Funcţiile **verificare_win()** si **verificare_finish()**, după mutarea unui pion verifică dacă jucătorul respectiv a introdus toţi cei 4 pioni în poziţiile de finish. Dacă există un astfel de jucător, acesta se poate inititula caştigător, însă jocul va continua pentru a se stabili şi celelalte locuri. Jocul se va incheia, atunci când va ramane un sigur jucător care nu a terminat de introdus toţi pionii. În final, după ce jocul s-a terminat fiecare player îşi va primi numărul de puncte aferente poziţiei sale, calculate cu ajutorul funcţiei **adaugarePuncte()**.In cazul in care un pion atunci cand este aproape la sfarsit, dar numarul de pe zar nu indica exact cate pozitii mai are pentru a ajunge la finish, jucatorul va primi un mesaj de eroare, care ii va spune ca "Acesta miscare nu este posibila."

```
int Verif_finish(int X,int Y,int cul,int pion,int x)//verifica daca pionul curent a ajuns la finish
{
    if(strchr("grva",t[x].table[X][Y]))
            if(t[x].player[cul].pozfinal[pion].x==X && t[x].player[cul].pozfinal[pion].y==Y)
                return 1;
    return 0;
}
void verificare_win(int x,int id)//verifica daca playerul a ajuns cu toti pionii la finish
{
    int c=j[id].cul;
    int piese_finish=0;
    if(t[x].Clasament[id%4]==0)
        for(int i=0;i<4;i++)
            {
                if(t[x].player[c].piesa[i].x==t[x].player[c].pozfinal[i].x && t[x].player[c].piesa[i].y==t[x].player[c].pozfinal[i].y)
                piese_finish++;//a ajuns in pozitie finala
            }
    if(piese_finish==4)
    {
        t[x].podium++;
        t[x].Clasament[id%4]=t[x].podium;//ii acorda pozitia pe podium
    }
}
```

Dupa ce un jucator a terminat jocul, acestuia ii va aparea un meniu, unde va putea sa isi vizualizeze statisticile sale(puncte,meciuri castigate,meciuri jucate), sau top-ul primilor 10 jucatori. Acest lucru este realizat cu ajutorul functiei **finalmenu()**.

```
void finalmenu(sqlite3 *db,int fd, int i)//afiseaza meniul final
{
    char com[10];
    fd_set actfds;
    char msg2[400];
    putoffline(j[i].nume,db,fd);
    bzero(msg2,400);
    strcat(msg2,"MENU:\nTastati /mystats pentru a va vedea statisticile\nTastati /top pentru a vedea primii 10 jucatori dupa numarul de puncte\nTastati q pentru a iesi din joc\n");
    if (write (fd, msg2, 200) < 0){
        perror ("[Server]Eroare la write() spre client.\n");
    }
    bzero(com,10);
    if (read(fd, com, 10) < 0){
        perror ("[Server]Eroare la write() spre client.\n");
    }
    while(strstr(com,"q")==0)
    {
        if(strstr(com,"/top"))
            top10(db,fd);
        else
            if(strstr(com,"/mystats"))
                mystats(db,fd,j[i].nume);
        bzero(com,10);
        if (read(fd, com, 10) < 0){
            perror ("[Server]Eroare la write() spre client.\n");
        }
    }
    close(fd);
    FD_CLR(fd, &actfds);
}
```

La final după ce toţi clienţii s-au deconectat de la server, thread-urile se vor opri, astfel că nu vor exista procese zombie.

## 5 Concluzii

În concluzie aplicaţia **"Nu te supăra,frate!"** este un proiect ce pune în aplicare bazele deprinse din cadrul reţelelor de calculatoare, ce se referă mai exact la folosirea corectă a proceselor-copil, socket-urilor, protocoalelor de comunicare, thread-urilor, dar în acelaşi timp şi implementarea altor concepte, cum ar fi bazele de date, aflate în strânsă legătură cu aplicaţia, stocând astfel datele clienţilor.

### 5.1 Viitoare Îmbunătăţiri

Pe viitor, aş dori să adaug o interfaţă grafică acestui joc, unde tabla să arate exact ca o tabla de joc din viaţa reală, iar pionii să fie coloraţi şi să arate ca nişte pioni. O altă idee ar fi, cu punctele obţinute, jucătorii să îşi poată cumpără anumite beneficii(cum ar fi diferite rank-uri:VIP,Veteran, etc.) şi un shop, pentru aţi personaliza pionii/tabla. De asemenea, adăugarea inteligenţei artificiale, poate fi o altă idee, unde liderul grupului poate alege dacă nu s-a atins capacitatea maximă de 4 playeri, să se poată adaugă **boţi**. Şi o ultimă idee de implementat, ar fi adaugărea funcţiei de chat între jucători.

## 6 Bibliografie

[1]. **Pagina cursului:**
`https://profs.info.uaic.ro/~computernetworks/cursullaboratorul.php`

[2]. **Pagina seminarului:**
https://profs.info.uaic.ro/~ioana.bogdan/

[3]. **Regulile jocului:**
http://jocuridincopilarie.ro/nu-te-supara-frate/

[4]. **Serverul si clientul din laboratorul 7:**
https://profs.info.uaic.ro/~computernetworks/files/NetEx/S12/ServerPreThread/servTcpPreTh.c
https://profs.info.uaic.ro/~computernetworks/files/NetEx/S12/ServerPreThread/cliTcpNr.c

[5]. **TCP/IP**:
https://ro.wikipedia.org/wiki/TCP/IP

[6]. **Implementarea bazei de date:**
https://www.sqlitetutorial.net/sqlite-limit/
https://www.tutorialspoint.com/sqlite/sqlite_c_cpp.htm