# Proiectul PlayGoC

Mihai Maxim[1][310910401RSL181185]

[1] Universitatea Alexandru Ioan Cuza din Iași, Bulevardul Carol I, Nr.11, 700506, Iaşi, România,
[2] Facultatea de Informatică, Str. General Berthelot, Nr. 16, RO-700483 – Iaşi
secretariat@info.uaic.ro
https://www.info.uaic.ro/

## 1 Introducere

Am ales proicetul PlayGoC deoarece consider ca "Go" este un joc complex,care merita abordat si mai ales implementat cu ajutorul retelelor de calculatoare.
Prezentarea jocului:
Go este jucat pe o matrice patratica ce poate avea urmatoarele dimensiuni:9x9, 13x13,19x19, timpul necesar unei partide fiind direct proportional cu dimensiunea aleasa.
Fiecare jucator foloseste un set de pietre de culoare alba sau neagra. Obiectivul jocului este sa inconjori mai mult teritoriu decat oponentul tau. O mutare la Go se face prin plasarea unei pietre pe o intersectie a liniilor tablei.
In timpul unei partide de Go una sau mai multe pietre pot fi capturate daca sunt inconjurate complet de pietre adverse, adica prin umplerea tuturor punctelor libere din jurul lor. Regulile aditionale ale jocului pot fi gasite la:
https://gobase.org/studying/rules/?id=11ln=ro

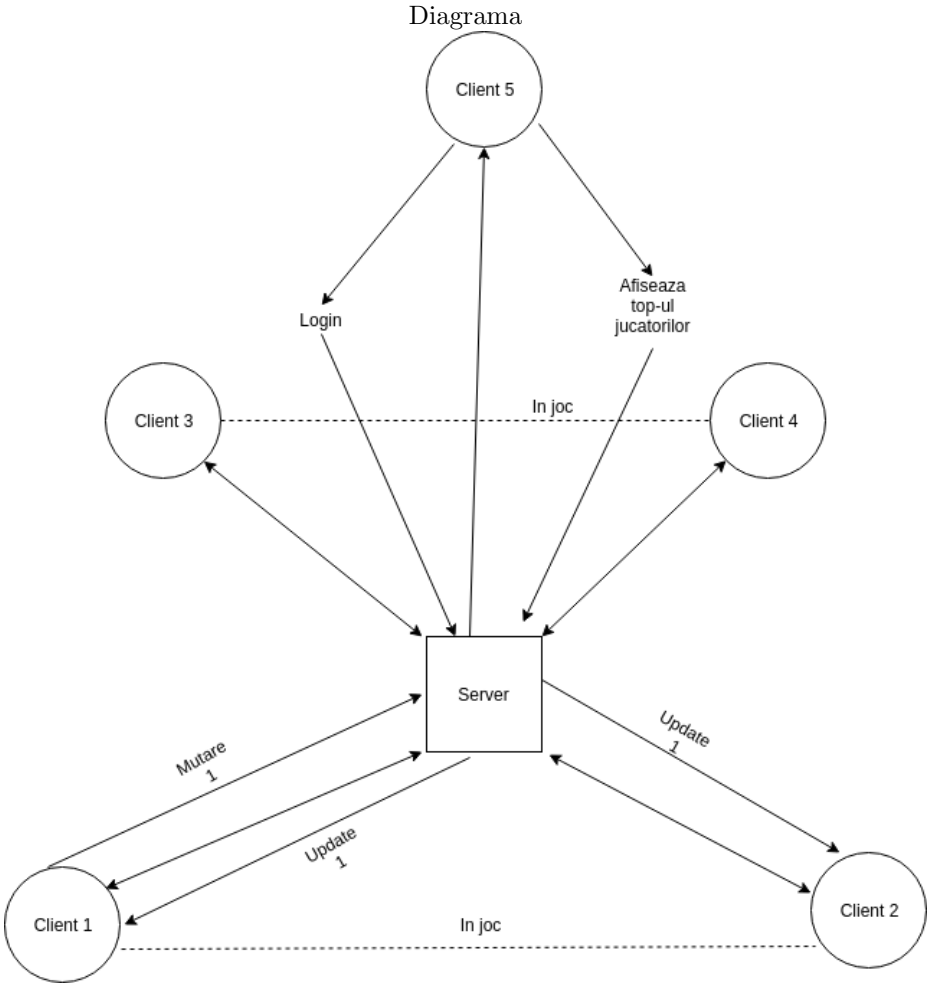
## 2 Tehnologii utilizate

UDP.
Folosit pentru requesturi mai mici:starea serverului,cate jocuri active sunt,etc.
TCP cu multiplexare I/O.
Folosit pentru a putea face lagatura dintre doi jucatori.
Libraria grafica SFML.
Folosita pentru a reprezenta jocul(matricea,piesele,butoanele,etc).

## 3 Arhitectura aplicatiei

Aplicatia va fi formata dintr-un server si clienti.
Server-ul va accepta clienti si ii va grupa in perechi de doi pentru a-i pune in joc.
Server-ul nu implementeaza o interfata grafica.
Clientii vor transmite mutarile server-ului ,care va face update la starea jocului.

Fiecare client va prelua starea jocului de la server si o va reprezenta in interfata sa grafica.
Vor fi implementate si:
O functie de log-in.
Un leaderbord cu scorul jucatorilor,in functie de categorie(dimensiunea matricii).
Abilitatea de a juca offline(pe acelasi pc) cu tine insuti sau un prieten.




## 4 Detalii de implementare

Server udp la care se pot conecta n clienti ,acestia fiind serviti in paralel.
In implementarea de mai jos fiecare client trimite un mesaj la server si acesta ii intoarce mesajul primit.

Aceasta ar putea fi folosita pentru a transmite informatii generale clientilor,inainte de a alege sa se conecteze la un joc.

Snippet UDP

```
serverAddr.sin_family = AF_INET;
serverAddr.sin_port = htons(PORT);
serverAddr.sin_addr.s_addr = inet_addr("127.0.0.1");

ret = bind(sockfd, (struct sockaddr*)&serverAddr, sizeof(serverAddr));
if(ret < 0){
        printf("Eroare la bind\n");
        exit(1);
}

if(listen(sockfd, 10) == 0){
        printf("Serverul asculta!\n");
}else{
        printf("Eroare la bind\n");
}


while(1){
        newSocket = accept(sockfd, (struct sockaddr*)&newAddr, &addr_size);
        if(newSocket < 0){
                exit(1);
        }
        printf("S-a acceptat conexiunea cu %s:%d\n", inet_ntoa(newAddr.sin_addr), ntohs(newAddr.sin_port));

        if((childpid = fork()) == 0){
                close(sockfd);

                while(1){
                        int rv=recv(newSocket, buffer, 1024, 0);
                        buffer[rv]='\0';
                        if(strcmp(buffer, "quit") == 0){
                                printf("%s:%d S-a deconectat\n", inet_ntoa(newAddr.sin_addr), ntohs(newAddr.sin_port));
                                break;
                        }else{

                                printf("Client: %s\n", buffer);
                                send(newSocket, buffer, strlen(buffer), 0);
                                bzero(buffer, sizeof(buffer));
                        }
                }
        }

}

close(newSocket);


return 0;
```

Codul afiseaza un window in care este desenat un cerc,atunci cand se face click pe cerc acesta isi schimba culoarea si notifica server-ul.

Snippet Client cu interfata grafica

```
{   sf::RenderWindow window(sf::VideoMode(200, 200), "SFML works!");
    sf::CircleShape shape(100.f);
    int x=0;
    shape.setFillColor(sf::Color::Green);
        window.clear();
        window.draw(shape);
        window.display();

    int clientSocket, ret;
        struct sockaddr_in serverAddr;
        char buffer[1024];

        clientSocket = socket(AF_INET, SOCK_STREAM, 0);
        if(clientSocket < 0){
                printf("Eroare la socket.\n");
                exit(1);
        }

        memset(&serverAddr, '\0', sizeof(serverAddr));
        serverAddr.sin_family = AF_INET;
        serverAddr.sin_port = htons(PORT);
        serverAddr.sin_addr.s_addr = inet_addr("127.0.0.1");

        ret = connect(clientSocket, (struct sockaddr*)&serverAddr, sizeof(serverAddr));
        if(ret < 0){
                printf("Eroare la connect.\n");
                exit(1);
        }
        printf("Conexiune reusita.\n");
        while(window.isOpen()){
        bool change=false;
         while(!change){
        if(sf::Mouse::isButtonPressed(sf::Mouse::Left))
        {
            if(x==0){shape.setFillColor(sf::Color::Red);x=1;}
            else
            {shape.setFillColor(sf::Color::Green);x=0;}
            change=true;
            window.clear();
            window.draw(shape);
            window.display();
        }
```

## 5 Concluzii

Cel mai mari provocari le aduc realizarea conectarii si gruparii clientilor(cine joaca cu cine,cum reuseste server-ul sa lucreze cu perechi de clienti).
Apoi se pune problema implementarii logicii jocului,care trebuie sa fie destul de complexa,avand in vedere numarul de reguli care trebuie luat in considerare.

## References

1. Computer-networks,FII
   https://profs.info.uaic.ro/ computernetworks/index.php
2. Regulile joculi
   https://gobase.org/studying/rules/?id=11ln=ro
3. SFML
   https://www.sfml-dev.org/
4. OVERLEAF
   https://www.overleaf.com/