Quiz Game

Blaj Marius-Marian A1

1 Introducere

Proiectul "Quiz Game" presupune implementarea unui model client-server prin intermediul caruia sa se poate obtine un server pe care se pot desfasura mai multe partide de quiz. Fiecare client, dupa conectarea la server va astepta ca oponentii lui sa se conecteze pentru a incepe o partida. Apoi tuturor clientilor li se dau intrebarile la care vor raspunde. Serverul va tine evidenta raspunsurilor iar la finalul rundei acesta va trimite tuturor clientilor cine este jucatorul castigator. Serverul suporta multiple parte desfasurate in paralel.

2 Tehnologii utilizate

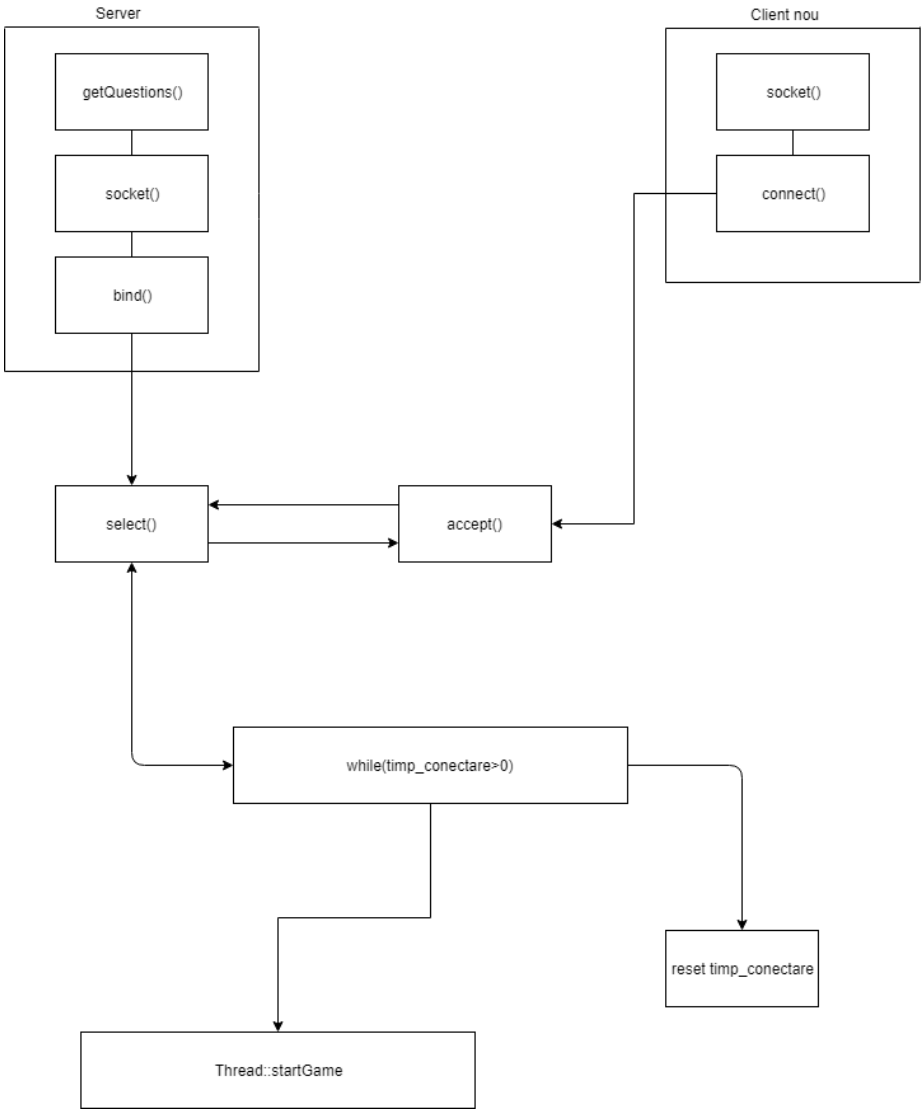
In acest proiect pentru transmiterea de mesaje intre client si server am folosit protocolul TCP. Protocolul TCP prezinta avantaje in comparatie cu protocolul UDP deoarece protocolul TCP asigura transmiterea mesajelor fara pierdere de informatie,astfel nu vom pierde raspunsul dat de clienti asigurand integritatea jocului.

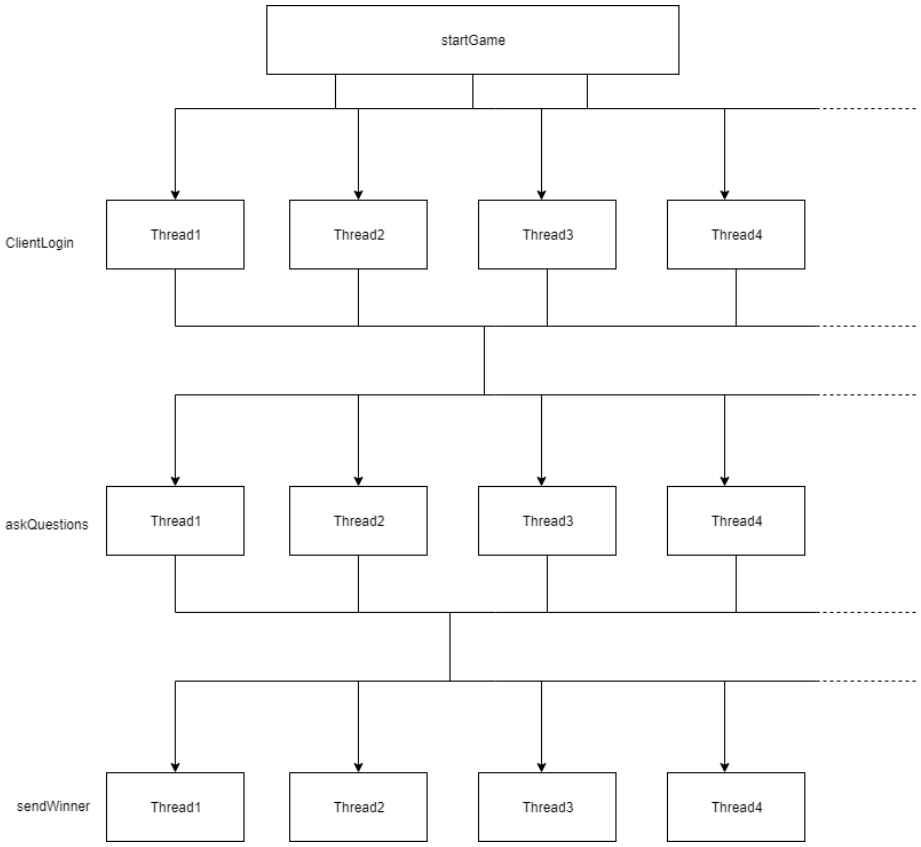
3 Arhitectura aplicatiei

3.1 Concepte implicate

Clientul si serverul vor folosi multiplexarea canalelor de intrare cu ajutorul functiei select(). Dupa conectarea clientilor, acestia asteapta sa intre concurenti in joc iar apoi jocul va fi administrat de un thread,permitand astfel desfasurarea altor runde in paralel. Pentru fiecare client din runda se va crea cate un thread care se ocupa cu trimiterea intrebarilor catre clienti si sincronizarea acestora. Intrebarile sunt stocate intr-un fisier XML "intrebari.xml" ce contine intrebarile, variantele de raspuns si varianta corecta.

3.2 Diagrama aplicatiei





4 Detalii implementare

Functia startGame se ocupa de administrarea thread-urilor pentru fiecare runda.

```
void * startGame(void * arg) {

    //primul si ultimul client din runda
    long start,finish;

    struct param *p=(struct param*) arg;
    start=p->start;
    finish=p->finish;
    //elibereaza resursele la finalul executiei
    pthread_detach(pthread_self());
    //numarul de threaduri necesare pentru actiuni
    pthread_t thread[finish - start];
    //index pentru thread
    int nr = 0;
    //mesaj de eroare pentru threaduri
    int threadErr;
    //pointer pentru a trimite parametrii la alte threaduri
    void * ptr;
```

```
19
20      //facem rost de numele clientilor
21      for(int i=start;i<finish;i++)
22      {
23          ptr = reinterpret_cast < void * > (i);
24          threadErr = pthread_create( & thread[i], NULL, clientLogin2
        , ptr);
25          if (threadErr) {
26                  printf("Unable to create thread\n");
27                  fflush(stdout);
28              }
29      }
30      //asteptam ca toti sa introduca numele
31      for(int i=start;i<finish;i++)
32          pthread_join(thread[i],NULL);
33
34      printf("Clientii s-au conectat\n");
35      fflush(stdout);
36
37      struct cIntrebare *argum=(struct cIntrebare*)malloc(sizeof(
        struct cIntrebare));
38      //trimitem intrebarea actuala clientilor
39      for (int intrebareActuala = 0; intrebareActuala < 10;
        intrebareActuala++) {
40          nr = 0;
41          for (int i = start; i < finish; i++) {
42              (*argum).id=i;
43              //intrebam numai clientii logati
44              if (clienti[i].isLogat() == true)
45              {
46                  (*argum).intrebare=intrebareActuala;
47                  threadErr = pthread_create( & thread[nr], NULL,
        clientPlay, argum);
48                  nr++;
49                  sleep(0.1);
50
51              }
52              if (threadErr) {
53                  printf("Unable to create thread\n");
54                  fflush(stdout);
55              }
56          }
57
58          //asteptam ca toti clientii sa termine intrebarea
59          for (int i = 0; i < finish - start; i++) {
60              pthread_join(thread[i], NULL);
61          }
62      }
63
64      printf("Aflam cine este castigatorul...\n");
65      fflush(stdout);
66
67      int winner = find_Winner(start,finish);
68      char castigator[100] = " ";
69      //mesajul cu castigatorul jocului
70      bzero(castigator, 100);
71      strcat(castigator, "Castigatorul este ");
```

```
72      strcat(castigator, clienti[winner].getName());
73      strcpy(castigator + strlen(castigator) - 1, " cu ");
74      char append[2];
75      sprintf(append, "%d", clienti[winner].getOk());
76      strcat(castigator, append);
77      strcat(castigator, " puncte!");
78
79      sleep(5);//suspans
80
81      printf("Comunicam clientilor cine a castigat runda\n");
82      fflush(stdout);
83      struct win * argument=(struct win*)malloc(sizeof(struct win));
84      strcpy((argument->mesaj),castigator);
85
86      //trimitem rezultatul tuturor clientilor
87      for (int i = start; i < finish; i++) {
88          if (clienti[i].isLogat() == true) {
89              argument->id=i;
90              pthread_create( & thread[i], NULL, printWinner,
    argument);
91              pthread_join(thread[i], NULL);
92          }
93      }
94
95      printf("Joc Terminat!\n");
96      fflush(stdout);
97  }
```

Acesta va creea cate un thread pentru fiecare client pentru a-i loga,a le trimite intrebari si a le trimite mesaj cu castigatorul. Aceste thread-uri sunt clientLogin2, clientPlay si printWinner. Mesajele trimise de server sunt de froma:

```
1  strcat(msgServer, "Asteptam sa se conecteze concurentii...\n");
2  if (send(clienti[tid].getSocket(), msgServer, 100, MSG_NOSIGNAL) <=
       0)
3
4  strcat(msgServer, "Introduceti numele dumneavoastra:");
5  if (send(clienti[tid].getSocket(), msgServer, 100, MSG_NOSIGNAL) <=
       0)
6
7  strcat(msgServer, question[intrebareActuala].detail[1]);
8  if (send(clienti[tid].getSocket(), msgServer, 100, MSG_NOSIGNAL)
```

Mesajele trimise de client vor contine numele lui la logare si in rest o litera reprezentand varianta aleasa. Pentru prelucrarea fisierului XML am folosit libraria "pugixml".Fisierul prelucrat este intrebari.xml la inceputul programului pentru a avea intrebarile in memorie. Exemplu de intrebare din fisier:

```
1  <problem value="1">
2      <question>Which of the following languages could be used in
       both Visual Studio and Unity?</question>
3      <answerA>A. Cobol</answerA>
4      <answerB>B. C#</answerB>
5      <answerC>C.  C    </answerC>
6      <answerD>D. French</answerD>
7      <correct>B</correct>
8    </problem>
```

5 Concluzii

Programul prezentat permite rularea in paralel a mai multor runde de "Quiz Game". Posibile imbunatatiri care ar putea fi aduse acestui program:
-numar variabil de intrebari pentru concurenti
-intrebarile sa fie alese la intamplare astfel jocul ar ramane "fresh"
-un clasament in care vor fi cei mai buni jucatori

References

[1] Informatii despre formatul XML : https://ro.wikipedia.org/wiki/XML

[2] Libraria pugiXML : https://pugixml.org/

[3] Informatii despre libraria Chrono : https://en.cppreference.com/w/cpp/chrono