QuizzGame

Marcu Eduard 2A3

Universitatea Alexandru Ioan Cuza, Facultatea de Informatica

Abstract. In acest raport este prezentata aplicatia QuizzGame si logica din spatele ei: tehnologiile utilizate, arhitectura si detalii de implementare.

1 Introducere

QuizzGame este un proiect de tip client/server unde server-ul furnizeaza intrebari catre jucatori (clienti), iar acestia trebuie sa raspunda la intrebarile date intr-un interval de timp. La finalul jocului serverul anunta jucatorul cu cele mai multe raspunsuri corecte, adica castigatorul.

Motivul pentru care am ales acest proiect este libertatea pe care ti-o ofera. Cerinta este simpla, jucatorul raspunde la intrebarile primite, deci iti ofera o anumita autonomie in privinta modalitatii de realizare a acestui proiect si numeroase mijloacei de a imbunatati si crea o aplicatie interesanta.

2 Tehnologii utilizate

Aceasta aplicatie presupune existenta a mai multor jucatori ce vor dori sa aiba acces la joc simultan, deci am realizat o comunicare server-client ce permite aceasta actiune.

Tehnologia utilizata in aceasta comunicare este TCP (Transmission Control Protocol)[1] deoarece dorim ca informatiile primite si trimise intre client si server sa fie receptionate in siguranta.

Pentru memorarea intrebarilor, variantelor de raspuns si a raspunsurilor corecte am ales sa folosesc un fisier XML[2] deoarece este simplu și accesibil (sunt fișiere text create pentru a structura, stoca și a transporta informația).

3 Arhitectura aplicatiei

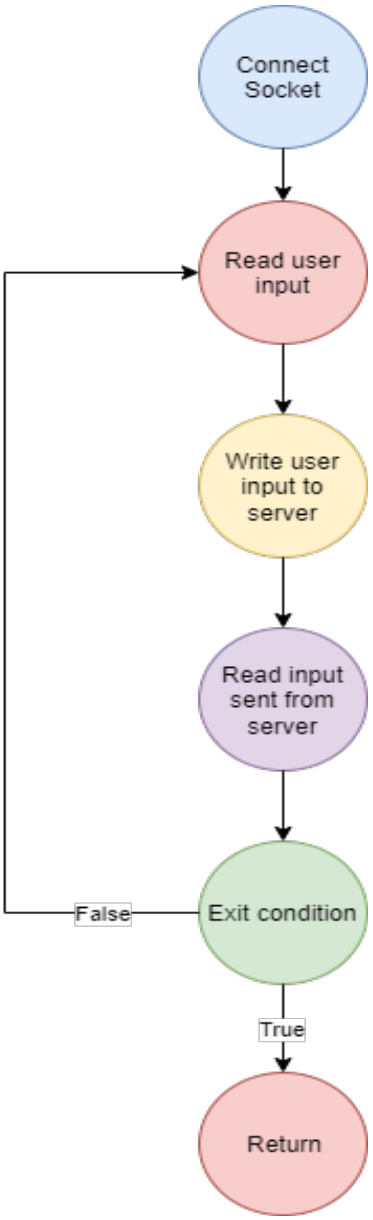

Fig. 1. Diagrama Client

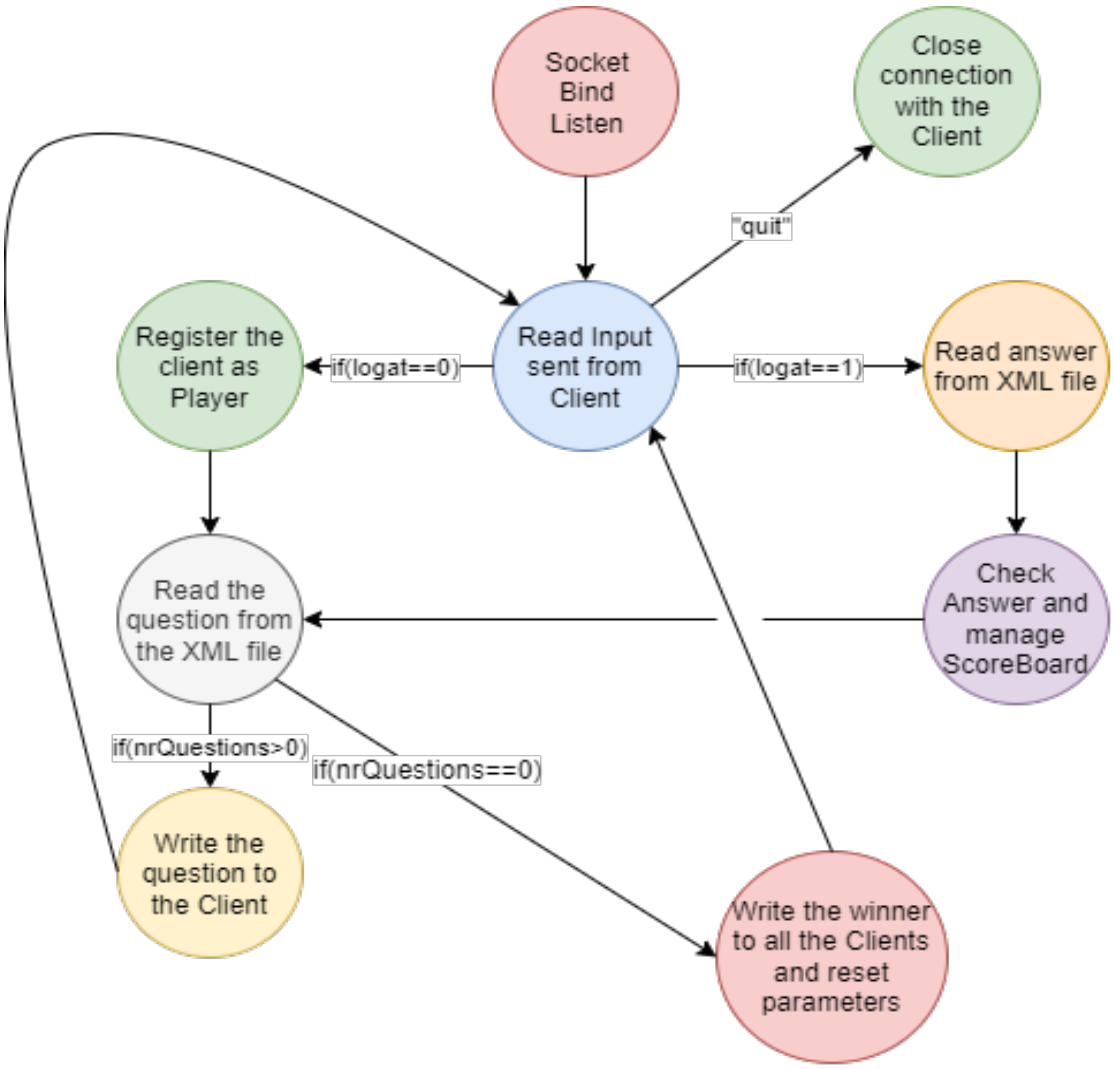

Fig. 2. Diagrama Server

Clientul citeste inputul de la tastatura si il trimite serverului prin socket, asteptand dupa un mesaj de la server.

Serverul citeste mesajul primit de la client. Primul mesaj primit va fi considerat numele clientului. Serverul asteapta ca toti clientii ce sunt conectati sa isi

scrie numele dupa care le trimite prima intrebare. Clientii au timp 10 secunde sa raspunda la intrebare, orice raspuns dat dupa 10 secunde va fi considerat gresit. La final serverul asteapta ca toti clientii sa termine jocul sau sa iasa din el si afiseaza pentru fiecare client cate puncte a luat si cine e castigatorul. In caz de egalitate se considera castigator cel care s-a conectat primul.

4 Detalii de implementare

```
struct Playeri
{
    struct timeval begin, end;
    char nume[100];
    int puncte;
    int bitlogare;
    int bitinitializare;
    int bitLivejoc;
    int bitSfarsitjoc;
    int bitIesit;
    char nrIntrebare[10];
} Player[100];
```

Fig. 3. Struct pentru jucatori

Fiecare client conectat la server va avea un idThread de tip intreg. Astfel pentru orice client Player[idThread] va memora diferite variabile ce ne vor ajuta in rularea programului.

```xml
<?xml version="1.0"?>
<Intrebari>
    <Numar id="I1">
        <Intrebare>Cat este 1+1?</Intrebare>
        <Raspsuns>A)1</Raspsuns>
        <Raspsuns>B)5</Raspsuns>
        <Raspsuns>C)2</Raspsuns>
        <Raspsuns>D)3</Raspsuns>
    </Numar>
    <Numar id="I2">
        <Intrebare>Cat este 2+1?</Intrebare>
        <Raspsuns>A)1</Raspsuns>
        <Raspsuns>B)5</Raspsuns>
        <Raspsuns>C)2</Raspsuns>
        <Raspsuns>D)3</Raspsuns>
    </Numar>
    <RaspunsCorect id="R1">C)2</RaspunsCorect>
    <RaspunsCorect id="R2">D)3</RaspunsCorect>
</Intrebari>
```

Fig. 4. Fisier XML

Fisierul XML va avea structura figurei de mai sus.

```
if (Player[tdL.idThread].bitinitializare = 1 && Player[tdL.idThread].bitlogare != 1)
{
    strcpy(Player[tdL.idThread].nume, msg);
    Player[tdL.idThread].bitlogare = 1;
    Player[tdL.idThread].puncte = 0;
    strcpy(Player[tdL.idThread].nrIntrebare, "I1");

    for (int contor = 0; contor <= returnarenrClienti(); contor++)
        if (Player[contor].bitinitializare == 1 && Player[contor].bitlogare == 0)
            contor = -1;

    Player[tdL.idThread].bitLivejoc = 1;

    char auxIntrebare[500];
    strcpy(auxIntrebare, PrimaxQuestions);

    gettimeofday(&Player[tdL.idThread].begin, 0);
    if (write(tdL.cl, auxIntrebare, 500) <= 0)
    {
        printf("[Thread %d] ", tdL.idThread);
        perror("[Thread]Eroare la write() catre client.\n");
    }
}
```

Fig. 5. Initializare

In prima instanta clientul conectat la server va fi considerat initializat, dar nu si logat. Acesta trebuie sa transmita un mesaj ce va fi considerat numele sau pentru a se loga. Odata logat acesta va astepta si ceilalti clienti conectati, daca exista, sa se inregistreze dupa care incepe jocul.

```
else if (strcmp(msg, "quit\n") == 0)
{
    Player[tdL.idThread].bitIesit = 1;
    char msgrasp[500] = "Ai iesit";
    if (write(tdL.cl, msgrasp, 500) <= 0)
    {
        printf("[Thread %d] ", tdL.idThread);
        perror("[Thread]Eroare la write() catre client.\n");
    }
    okServer = 0;
}
else
```

Fig. 6. Comanda quit

Comanda quit inchide conexiunea cu clientul si memoreaza faptul ca acesta a iesit.

```
gettimeofday(&Player[tdL.idThread].end, 0);
long seconds = Player[tdL.idThread].end.tv_sec - Player[tdL.idThread].begin.tv_sec;
long microseconds = Player[tdL.idThread].end.tv_usec - Player[tdL.idThread].begin.tv_usec;
double elapsed = seconds + microseconds*1e-6;
printf("Time measured: %.3f seconds.Thread %d\n", elapsed,tdL.idThread);
strcpy(xQuestions, "\0");

if(elapsed<=10.0)
{
verificareRaspuns(tdL.idThread, msg);
strcpy(xQuestions, "\0");
}

if (VerificareIncrementare(tdL.idThread) == 1)
{
    incrementare(tdL.idThread);
    primireIntrebare(tdL.idThread);
    char auxIntrebare[500];
    strcpy(auxIntrebare, xQuestions);
    strcpy(xQuestions, "\0");
    gettimeofday(&Player[tdL.idThread].begin, 0);
    if (write(tdL.cl, auxIntrebare, 500) <= 0)
    {
        printf("[Thread %d] ", tdL.idThread);
        perror("[Thread]Eroare la write() catre client.\n");
    }
}
```

Fig. 7. Raspuns

Cand clientul trimite o varianta de raspuns se va calcula timpul ce a fost necesar pentru aceasta actiune. Daca depaseste mai mult de 10 secunde raspunsul nu va fi validat. Pe urma serverul verifica daca mai sunt intrebari in fisierul XML. Daca mai sunt, acesta le scrie clientului.

```
else
{
    Player[tdL.idThread].bitIesit = 1;
    for (int contor = 0; contor <= returnarenrClienti(); contor++)
        if (Player[contor].bitIesit != 1)
            contor = -1;

    int Xpuncte = Player[tdL.idThread].puncte;
    char message1[500];
    sprintf(message1, "%d", Xpuncte);
    char message[500] = "Numarul tau de puncte acumulate: ";
    strcat(message, message1);
    int maximum = -1;
    int idPlayer;
    for (int contor = 0; contor <= returnarenrClienti(); contor++)
    {
        if (Player[contor].puncte > maximum)
        {
            idPlayer = contor;
            maximum = Player[contor].puncte;
        }
    }
    char message2[500];
    sprintf(message2, "%d", maximum);
    char message3[500] = ". Castigatorul este ";
    strcat(message3, Player[idPlayer].nume);
    char message4[500] = "cu punctajul: ";
    strcat(message4, message2);
    strcat(message3, message4);
    strcat(message, message3);

    if (write(tdL.cl, message, 500) <= 0)
    {
        printf("[Thread %d] ", tdL.idThread);
        perror("[Thread]Eroare la write() catre client.\n");
    }
    ////////pregatire inchidere
    for (int contor = 0; contor <= returnarenrClienti(); contor++)
        Player[contor].puncte = -1;
    okServer = 0;
    ////////
}
```

Fig. 8. Raspuns

Daca nu mai sunt intrebari inseamna ca jucatorul a terminat si va astepta sa termine si ceilalti jucatori. Dupa ce toti au terminat se gaseste castigatorul si clientii vor fi anuntati.

```
void print_xml(xmlNode *node, int indent_len)
{
    strcat(xQuestions, node->name);
    if (node->type == XML_ELEMENT_NODE)
    {
        //printf("%*c%s:%s\n", indent_len*2, '-', node->name, xmlNodeGetContent(node));
        strcat(xQuestions, xmlNodeGetContent(node));
    }
}

xmlNode *find_node(xmlNode *node, char *prop_val)
{

    xmlNode *result;

    if (node == NULL)
        return NULL;

    while (node)
    {
        if ((node->type == XML_ELEMENT_NODE) && xmlGetProp(node, "id") && (strcmp(xmlGetProp(node, "id"), prop_val) == 0))
        {
            return node;
        }

        if (result = find_node(node->children, prop_val))
            return result;

        node = node->next;
    }

    return NULL;
}
```

Fig. 9. Functii pentru afisarea datelor din XML

Functia find_node ne gaseste nodul din fisierul XML cu id-ul dorit. Functia print_xml ne va afista continutul nodului gasit si a copiilor sai.

5 Concluzii

Aplicatia poate beneficia de multe imbunatatiri. Pe langa cea mai evidenta, o interfata grafica, ar fi o idee buna ca intrebarile sa fie adresate clientilor in mod aleatoriu. Mai explicit sa existe 50 de intrebari in total dintre care sa se selecteze

doar un anumit numar de intrebari pentru joc. Unele intrebari, precum cele de matematica, ar putea fi chiar create in mod aleatoriu.

O alta imbunatatire ar putea consta in crearea unui admin ce poate sa scoata sau sa puna intrebari in XML prin diferite comenzi fara a fi nevoie ca ele sa fie scrise de mana in cod.

References

1. TCP.
   https://ro.wikipedia.org/wiki/Transmission_Control_Protocol

2. XML.
   https://ro.wikipedia.org/wiki/XML
   https://qnaplus.com/print-xml-file-tree-form-libxml2-c-programming/
   https://qnaplus.com/search-for-an-xml-node-using-libxml2-in-c/
   http://xmlsoft.org/

3. Comunicare client server.
   https://profs.info.uaic.ro/~georgiana.calancea/Laboratorul_9.pdf

4. Functii de timp.
   https://levelup.gitconnected.com/8-ways-to-measure-execution-time-in-c-c-48634458d0f9