QuizzGame

Gherasim Antonie Stefan-E1

Universitatea “Alexandru Ioan Cuza”

1. Introducere

Scopul proiectului este de a simula o aplicatie in cadrul careia utilizatorii primesc un set de intrebari, raspund la acestea iar numarul de raspunsuri corecte pentru fiecare client este stocat intr-o baza de date. Baza de date in care sunt inregistrati clientii si numarul de raspunsuri corecte este accesata de server. La finalul jocului serverul anunta castigatorul.

2. Tehnologii utilizate

2.1 TCP

TCP este un protocol de transport orientat-conexiune care ofera siguranta in trasportul de date. Orientat-conexiune inseamna ca pentru a se asigura transmiterea de date intre server si client trebuie sa exita o conexiune stabila. Pentru a incepe transferul de date, serverul asteapta aparitia unui client.

Am ales TCP in loc de UDP deoarece prefer siguranta transportului de date in schimbul vitezei. Nu sunt permise erori la transferul de date deoarece acest lucru poate face diferenta intre castigatorul jocului si ceilalti concurenti.

2.2 Socket programming

Programarea socket-ului este o modalitate de conectare a două noduri dintr-o rețea pentru a comunica între ele. Un socket (nod) ascultă pe un anumit port la un IP, în timp ce celălalt soclu ajunge la celălalt pentru a forma o conexiune. Serverul formează soclul ascultătorului în timp ce clientul ajunge la server.

2.3 Thread

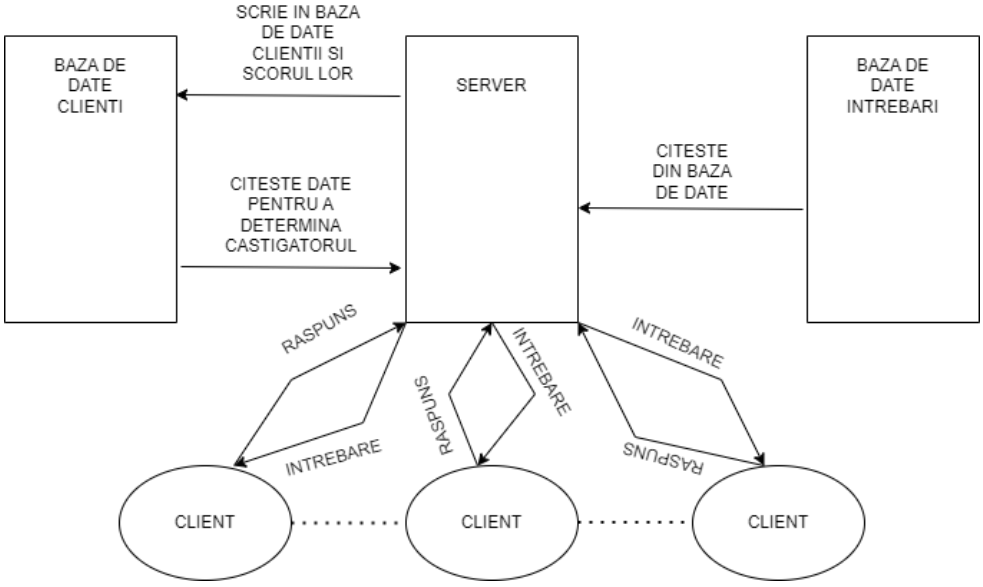
Cu programarea pe calculator, un thread este un mic set de instrucțiuni concepute pentru a fi programate și executate de cpu independent de procesul părinte. De exemplu, un program poate avea un thread deschis care așteaptă să aibă loc un anumit eveniment sau execută o activitate separată, permițând programului principal să efectueze alte activități.

Am ales sa folosesc Threaduri in loc de procese datorita posibilitatii de a crea mai multe fire de executie intr-un timp mai scurt comparativ cu procesele.

3. Arhitectura Aplicatiei

Aplicatia este un quizzGame, ce reprezinta un server care comunica cu mai multi client prin intermediul socket-urilor TCP. Serverul ruleaza intr un thread separate iar fiecare client este gestionat de catre un thread propriu. Aplicatia stocheaza intrebari si raspunsuri intr-o lista simplu inlantuita si poate primi raspunsuri de la client prin intermediul socket ului TCP. Aplicatia comunica cu un numar nelimitat de client prin intermediul unui set de threaduri care este stocat intr o lista simplu inlantuita si socket uri TCP.

Diagrama Aplicatiei



4.Detalii implementare

Acest program este un server de tipul Quiz Game, care poate primi mai multe conexiuni de la clienți prin intermediul unui socket TCP. Programul folosește un fișier "questions.txt" pentru a citi întrebările și răspunsurile posibile ale jocului. Acestea sunt stocate într-o listă simplu inlantuita de tip quiz_question.

Server-ul are un thread principal care se ocupă de acceptarea conexiunilor de la clienți și de crearea unui nou thread pentru fiecare client care se conectează. Fiecare thread are sarcina de a comunica cu clientul respectiv și de a-i oferi întrebările și a primi răspunsurile acestuia.

Server-ul are un mecanism de timeout pentru oprirea sa după o perioadă de timp specificată. Acest lucru este realizat prin intermediul semnalului SIGALRM și a funcției setitimer.

În plus, programul conține o structură de tip thread care este folosită pentru a ține evidența thread-urilor create și a informațiilor despre fiecare client conectat, cum ar fi numele de utilizator, scorul și timpul de răspuns. Această structură este adăugată într-o listă simplu inlantuita in care este parcursă pentru a se actualiza informațiile despre fiecare client și pentru a le trimite către toți clienții conectați la server.

În program sunt incluse următoarele biblioteci:

<sys/types.h>, <sys/socket.h>, <netinet/in.h> - pentru crearea socket-ului și utilizarea acestuia pentru comunicarea prin rețea

<errno.h>, <unistd.h>, <stdio.h>, <string.h>, <stdlib.h> - pentru gestionarea erorilor, citirea și scrierea în fișiere, alocarea dinamică de memorie și alte operații de bază

<signal.h>, <pthread.h>, <sys/time.h> - pentru gestionarea semnalelor, crearea de thread-uri, gestionarea timeout-ului.

Se creează un socket TCP și se setează adresa și portul server-ului. Apoi, se setează un handler pentru semnalul SIGALRM și se setează timeout-ul pentru server cu ajutorul funcțiilor set_handler() și set_timeout().

Server-ul așteaptă conexiuni de la clienți prin intermediul apelului la accept(). Dacă o conexiune este primită, se creează un thread nou pentru a se ocupa de comunicarea cu acest client și se adaugă informațiile despre acesta în lista de thread-uri. Dacă server-ul

primește semnalul de oprire prin intermediul variabilei server_should_stop, timpul pentru acceptarea clientilor s a terminat si incepe jocul.

Funcția treat() este executată de fiecare thread și are sarcina de a comunica cu clientul respectiv. Întâi, se trimite clientului un mesaj de bun venit și se primește numele de utilizator al acestuia. Apoi, se trimite clientului prima întrebare din lista de quiz_question și se așteaptă răspunsul acestuia. Dacă răspunsul este corect, se adaugă 1 la scorul clientului și se trimite următoarea întrebare.

Raspunsul pe care il trimite clientul va fi format din cifre scrise fara spatii in cazul in care sunt intrebari cu alegere multipla.

5. Concluzii

In raport se creioneaza o structura a proiectului unde se descriu functiile si modalitatea de implementare a jocului. Implementarea poate fi imbunatatita daca se va adauga o biblioteca grafica , unde user ul isi poate alege varianta de raspuns apasand pe solutie fara sa fie nevoie sa scrie. Acest lucru poate aduce un plus de viteza si cateva secunde de timp de gandire pentru cei ce nu stiu sigur raspunsul la intrebare.

6. Referinte

1. Course&Laboratory - Computer Networks 2021-2022 (uaic.ro)

2. www.geeksforgeeks.org

3. man page