Documentatia proiectului Local Market Place

Ciobanu, Matei

student in grupa 2A4 la Facultatea de Informatica din Iasi

Abstract. În lucrarea dată am prezentat succint atât viziunea mea asupra Local Market Place, cât și metoda de implementare a proiectului. De asemenea, au fost enumerate operațiunile pe care le poate face utilizatorul pe platformă și s-a explicat succint modul de comunicare dintre server și client,justificandu-se motivele din spatele alegerii protocolului TCP. Totodată, au fost sugerate idei de îmbunătățire a proiectului.

1 Introducere

1.1 Viziunea proiectului

Ne-am propus ca aplicația să reprezinte o implementare a unei platforme prin care utilizatorii conectați să aibă posibilitatea de a publica,de a vedea anunturi in care sunt listate produsele de vanzare si de a negocia oferte in vederea tranzactionarii acestora. În lipsa unei interfețe grafice, am adaptat unele funcționalități ale clientului prin citirea unor parametri dați de utilizator.

Operații efectuate de client Ne propunem să asigurăm utilizatorilor posibilitatea de:

- a-și crea cont pe platformă;
- a se conecta cu ajutorul unui nume de utilizator și al unei parole;
- a-și reseta parola în cazul în care au uitat-o;

Le oferim utilizatorilor posibilitatea de a executa urmatoarele comenzi,cu conditia de a se fi conectat de pe platforma:

- sa propuna oferte de cumparare la produsele listate pentru vanzare;
- sa vizualizeze ofertele primite de la alti utilizatori si propuse pentru alti utilizatori
- sa creeze anunțuri de vânzare;
- sa accepte oferte de cumparare;
- sa șteargă anunțurile postate anterior;
- sa vizualiza toate anunțurile postate;
- sa vizualizeze propriul istoric de tranzacții realizate prin intermediul platformei;
- sa se deconecteze de pe platformă, pentru a se putea reconecta sub un alt nume de utilizator;

2 Tehnologii Aplicate

Comunicarea dintre server și clienți se realizează prin protocolul TCP. Avantajele acestui protocol față de UDP sunt că oferă siguranța transmiterii integrale a informațiilor și că este mai robust în cazul pierderilor de pachete[1].

Serverul tratează comunicările cu clienții în mod concurent, pentru a le asigura acestora un timp de așteptare mai mic. Pentru fiecare client ce accesează platforma se creează un proces fiu în care se vor efectua toate operațiile, iar rezultatul final va fi trimis către client.
Stocarea informatiilor necesare se face cu ajutorul bazei de date SQLite(instalat dupa imodelul prezentat in [2]),iar interogarea ei se face cu ajutorul functiilor definite in biblioteca "sqlite3"[3](am specificat terminalului in momentul compilarii ca folosim bibnlioteca "sqlite3" si versiunea C99 a limbajului de programare C prin comanda gcc ./ex.c -o ./ex -lsqlite3 -std=c99).

3 Structura aplicatiei

Programul este proiectat pentru a stoca datele de care vom avea nevoie ulterior (ex. numele utilizatorului și parola acestuia, aspecte legate de anunț precum prețul și modelul produsului) in baza de date, care este formata din 4 tabele:

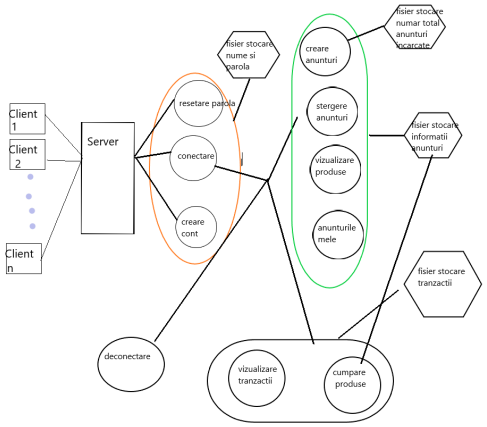
- -un tabel pentru a salva numele și parola utilizatorului;
- -un tabel în care vom scrie informațiile referitoare la anunț (numărul anunțului, categoria produsului, brandul, modelul, prețul de vânzare și numele utilizatorului care a postat anunțul);
- un tabel în care vom salva tranzacțiile efectuate prin aplicație;
- -un tabel in care vom salva ultima oferta din procesul de negociere dintre doi utilizatori pentru un anumit produs;

3.1 Scenarii de utilizare

Logica din spatele aplicației este următoarea:

- clientul trimite server-ului operația pe care utilizatorul dorește să o efectueze;
- server-ul va trimite input-ul procesului fiu creat pentru user-ul respectiv;
- procesul fiu verifică la început dacă utilizatorul este conectat. În caz negativ, el nu va efectua request-ul și va trimite clientului un mesaj prin care îi spune că trebuie să se conecteze sau să își creeze cont;
- după conectare, procesul fiu va efectua request-ul, căutând informațiile necesare în tabelele destinate pentru stocarea informațiilor și , dacă este cazul, modificându-le pe acestea;
- după efectuarea operatiei, vor fi trimise către utilizator rezultatele sau un mesaj legat de succesul/eșecul** actualizării datelor.

**



4 Aspecte de Implementare

Conexiunea dintre server și client se face prin protocolul TCP,cu ajutorul socket-descripturilor.

```
struct sockaddr_in from;
char msg[100];      //mesajul primit de la client
char msgrasp[100]=" ";        //mesaj de raspuns pentru client
int sd;         //descriptorul de socket

/* crearea unui socket */
if ((sd = socket (AF_INET, SOCK_STREAM, 0)) == -1)
{
    perror ("[server]Eroare la socket().\n");
    return errno;
}

/* pregatirea structurilor de date */
bzero (&server, sizeof (server));
bzero (&from, sizeof (from));

/* umplem structura folosita de server */
/* stabilirea familiei de socket-uri */
server.sin_family = AF_INET;
/* acceptam orice adresa */
server.sin_addr.s_addr = htonl (INADDR_ANY);
/* utilizam un port utilizator */
server.sin_port = htons (PORT);

/* atasam socketul */
```

Server-ul este concurent,iar pentru fiecare utilizator ce acceseaza platforma se creeaza un proces fiu(idee preluata din codul [4]), in care vor fi executate toate interogarile facute de client.

```
if (client < 0)
{
    perror ("[server-procesparinte]Eroare la accept().\n");
    continue;
}

int pid;
if ((pid = fork()) == -1) {
    close(client);
    continue;

} else if (pid > 0) {
    // parinte
    close(client);
    /*while(waitpid(-1,NULL,WNOHANG));
    continue;*/
} else if (pid == 0) {
    // copil
    close(sd);
    int deconectat=0,comandaGasita=0;
   while(deconectat==0){
    /* s-a r;ealizat conexiunea, se astepta mesajul */
    bzero (msg, 100);
    printf ("[server-procesfiu]Asteptam mesajul...\n");
    fflush (stdout);
```

Conectarea la baza de date se face prin apelarea functiei connectDB si crearea tabelelor in care vom stoca datele se face prin apelul functiei setupDB(cod inspirat dupa modelul prezentat in [3]).

```
void setupBD()
{
    sqlite3 * db;
    char * err;
     int status=sqlite3_open("baza.db",&db);
     if(status!=SQLITE_OK)
     {perror("conectare nereusita");
     sqlite3_close(db);
     }
      char *sql1 ="CREATE TABLE IF NOT EXISTS Users(Nume Varchar(30),Parola Varchar(30),Gmail Varchar(30),LastCo
        char *sql2="CREATE TABLE IF NOT EXISTS Tranzactii(Nume Varchar(30),Actiuni Varchar(30),Brand Varchar(30)
    char *sql3="CREATE TABLE IF NOT EXISTS Anunturi(Nranunt INTEGER PRIMARY KEY,Categorie Varchar(30),Brand Varc
    char *sql4="Create Table IF NOT EXISTS Of(NrAnunt INT,Cumparator Varchar(30) ,Vanzator Varchar(30) ,Pret Var
    status=sqlite3_exec(db,sql1,0,0,&err) ;
  status=sqlite3_exec(db,sql2,0,0,&err) ;
  status=sqlite3_exec(db,sql3,0,0,&err);
   status=sqlite3_exec(db,sql4,0,0,&err);
    if(status!=0)
    {
            fprintf(stderr, "SQL error: %s\n", err);

        sqlite3_free(err);
        sqlite3_close(db);
    }
```

Pentru a evita problemele de concurenta,am limitat accesul proceselor la tabelele din baza de date prin folosirea unor variabile de tip ”semafor” desinate fiecaruia din acestea[5].

```
sem_init(&semaforUsers,1,1);
sem_init(&semaforAnunturi,1,1);
sem_init(&semaforOferte,1,1);
sem_init(&semaforTranzactii,1,1);
```

```
waitSemafor(&semaforOferte);

getField( 4,"Of",acc,10,"NrAnunt",nranunt,"Cumparator",cumparator);
getField(4,"Of",Model, 6,"NrAnunt",nranunt,"Cumparator",cumparator);
getField(4,"Of",Brand, 5,"NrAnunt",nranunt,"Cumparator",cumparator);
getField(4,"Of",Pret, 3,"NrAnunt",nranunt,"Cumparator",cumparator);
postSemafor(&semaforOferte);
```

Pentru a modifica continutul tabelelor sau pentru a afisa informatii din acestea, am creat functiile :insertDB,getField, selectDB,updateDB si deleteDB.Functia insertDB ne ajuta sa inseram date intr-un tabel dat ca argument.Ea primeste ca parametri un numar variabil de argumente[6], care depinde de numarul de coloane in care dorim sa inseram date.

```
if(statusBD==SQLITE_OK)
{
  va_list args;
  va_start(args,nrOFArgs);
  int i;
  for(i=1;i<=nrOFArgs/2;i++)
    {
        const char *c=va_arg(args,const char *);
        strcat(task,c);
        if(i!=nrOFArgs/2)strcat(task,",");
    }
    strcat(task,") Values(");

    for(i=1;i<=nrOFArgs/2;i++)
    {
        const char *c=va_arg(args,const char *);
        strcat(task,"\"");
        strcat(task,c);
        strcat(task,"\"");
        if(i!=nrOFArgs/2)strcat(task,",");
    }
    strcat(task,");");
     va_end(args);
        char * err;
        statusBD=sqlite3_exec(db,task,0,0,&err);
        printf("\n%s",task);
    sqlite3_close(db);
```

5 Îmbunătățiri propuse

Proiectul va implementa funcționalitățile de bază ale unei platforme online de postat anunțuri online.Câteva sugestii de a îmbunătăți aplicația ar fi de a trimite pe adresele de mail notificari legate de ofertele recente și de a asigura o interfață grafică user-friendly.

6 Referinte

1. Caracteristici protocol TCP

2. https://linuxhint.com/install-sqlite-ubuntu-linux-mint/Instalare baza de date SQLite
3. Interogarea bazei de date cu ajutorul librariei sqlite3
4. Model cod server concurent
5. Utilizarea semafoarelor in C
6. Functii cu numarul variabil de argumente in C++