Local Marketplace Platform

Cochior Daniel, A6

Universitatea Alexandru Ioan Cuza, Iasi, Romania

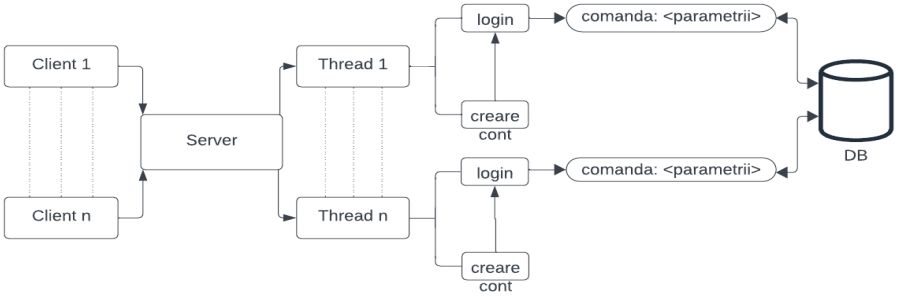
1 Introducere

Local Marketplace Platform este o platforma locala care are ca scop crearea unui spațiu unde utilizatorii pot vinde și cumpăra produse sau servicii. Aceasta platforma le va permite utilizatorilor sa creeze noi oferte sau sa vizualizeze ofertele existente ale celorlalti utilizatori, precum si sa achizitioneze produsele dorite. Pentru facilitarea tranzactiilor, aplicatia pune la dispozitie un portofel pentru fiecare utilizator, iar fiecare tranzactie de vanzare-cumparare se va incheia sau nu in functie de soldul fiecarui utilizator.

2 Tehnologii utilizate

Am ales să implementez o soluție bazată pe tehnologia TCP/IP, utilizând un server concurent cu suport pentru mai multe thread-uri. Această abordare permite conexiunea simultană a mai multor clienți, oferindu-le posibilitatea de a vedea in timp real noile oferte aparute pe platforma, precum si disponibilitatea celor deja existente. Am ales tehnologia TCP/IP deoarece acesta asigura integritatea datelor care este foarte importanta avand in vedere contextul financiar al aplicatiei.
Pentru stocarea datelor despre utilizatori si produse voi folosi o baza de date Sqlite.

3 Arhitectura aplicatiei



4 Detalii de implementare

4.1 Structura bazei de date

In implementarea acestei platforme am folosit urmatoarea baza de date ce contine doua tabele:

- Tabelul asociat userilor ce va contine urmatoarele coloane: id-ul utilizatorului(unic), numele si parola (necesare pentru conectarea la server), soldul si o coloana logged cu care se va face verificarea daca un client deste logat sau nu.
- Tabelul asociat ofertelor ce va contine urmatoarele coloan: id-ul ofertei(unic), numele produsului, pretul produsului, id-ul vanzatorului, id-ul cumparatorului.

4.2 Comenzile disponibile

Comenzile disponibile pentru un utilizator al platformei sunt urmatoarele:

1. login: <username> <parola>
2. creare cont: <username> <parola> <sold>
3. logout
4. lista produse
5. creare oferta: <nume produs> <pret produs>
6. modificare oferta: <id> <nume produs> <pret produs>
7. stergere oferta: <id>
8. istoric achizitii
9. cumparare produs: <id>
10. cautare produs: <nume produs>
11. vizualizare sold
12. modificare sold: <adaugare/retragere> <suma>
13. postari proprii
14. exit

4.3 Bucati de cod

Pentru parsarea comenzilor cu mai multi parametrii mi-am declarat urmatoarea structura:

```
typedef struct {
    char command[50];
    char **params;
    int numParams;
} CommandParams;
```

Astfel, in CommandParams voi retine numele comentii, numarul de parametrii si numele acestora, in acest fel putand sa folosesc aceasta parsare a comenzilor pentru orice comanda.

```
CommandParams parseCommand(char *input) {
    CommandParams result;

    result.params = NULL;
    result.numParams = 0;

    char *token = strtok(input, ":");

    strcpy(result.command, token);

    while ((token = strtok(NULL, " ")) != NULL) {
        result.params = realloc(result.params, (result.numParams + 1) * sizeof(char*));
        result.params[result.numParams] = strdup(token);
        result.numParams++;
    }

    return result;
}
```

Cu ajutorul acestei functii creez o variabila de tip CommandParams pe care o populez astfel:caracterele de dinainte de cele doua puncte vor reprezenta numele comenzii, iar cele de dupa vor fi parametrii care sunt separati de catre spatii.

```
char* login_client(char *cmd){
    char* result = (char*)calloc(MAX_COMMAND_LENGTH, sizeof(char));
    if (result == NULL) {
        perror("Eroare la alocarea de memorie");
        exit(EXIT_FAILURE);
    }

    CommandParams params = parseCommand(cmd);
    if(params.numParams != 2){
        strcat(result, "Numarul de parametrii este incorect!\n");
        strcpy(result, "Comanda trebuie sa fie de forma: \n");
        strcat(result, "login: <username> <password>\n");
        freeCommandParams(&params);
        return result;
    }

    //strcpy(result, "Comanda login a fost receptionata!");
    strcat(result, "S-a receptionat comanda: ");
    strcat(result, params.command);
    strcat(result, "\n Cu urmatorii parametrii: \n");
    for(int i=0; i<params.numParams; i++){
        strcat(result, params.params[i]);
        strcat(result, "\n");
    }

    freeCommandParams(&params);
    return result;
}
```

Acesta este un exempul in care utilizez aceasta structura si unde fac si verificarea dupa numarul de parametrii.

```
char* manager_comenzi(char *comanda)
{
    if (strstr(comanda, "login"))
    {
        return login_client(comanda);
    }
    else if (strstr(comanda, "creare cont"))
    {
        return creare_cont(comanda);
    }
    else if (strstr(comanda, "lista produse"))
    {
        return lista_produse(comanda);
    }
    else if (strstr(comanda, "creare oferta"))
    {
        return creare_oferta(comanda);
    }
    else if (strstr(comanda, "modificare oferta"))
    {
```

Pentru a returna raspunsul dorit de client, am implementat un manager de comenzi care apeleaza functia care se va ocupa cu cererea clientului.

Pentru a interactiona cu baza de date, am functiile din biblioteca sqlite3.h. Cu functia sqlite3_mprintf() se construieste o interogare SQL, care este urmata de functia sqlite3_prepare_v2 care pregătește interogarea pentru a fi executată ulterior. Daca rezultatul acestei functii este SQLITE_OK inseamna ca interogarea poate fi executata, iar acest lucru este facut cu functia sqlite3_step. Aceasta functie returneaza SQLITE_ROW daca interogarea returneaza cel putin un rand de date, SQLITE_DONE daca interogarea s-a efectuat cu succes, sau poate returna alte coduri de eroare. Daca functia returneaza cel putin un rand de date la o interogare de citire, datele pot fi accesate din pointerul la structura de tipul sqlite3_stmt, coloana cu coloana si sunt folosite in functie de nevoi. La finalul unei interactiuni cu baza de date folosesc sqlite3_finalize pentru a eliberea resursele asociate instrucțiunii pregătite si sqlite3_close pentru a inchide baza de date SQLite. Mai jos este un exempul in care am folosit aceste functii pentru a

citi din baza de date soldul unui utilizator.

```
sqlite3_stmt *stmt;
sqlite3 *db;
int open_db = sqlite3_open("mkDB.db", &db);
if(open_db != SQLITE_OK){
    printf("Eroare la deschiderea bazei de date!\n");
    exit(EXIT_FAILURE);
}
int res_db;
char *query1 = sqlite3_mprintf("SELECT sold FROM Useri WHERE id = '%d';", *id);
res_db = sqlite3_prepare_v2(db, query1, -1, &stmt, 0);
if(res_db != SQLITE_OK){
    printf("Eroare la pregatirea interogarii!\n");
    exit(EXIT_FAILURE);
}
else{
    if(sqlite3_step(stmt) == SQLITE_ROW){
        char *sold = (char*)calloc(10, sizeof(char));
        strcpy(sold, sqlite3_column_text(stmt, 0));
        strcat(result, "Soldul dumneavoastra este: ");
        strcat(result, sold);
        strcat(result, "\n");
        free(sold);
    }
    else{
        strcat(result, "Eroare la vizualizarea soldului!\n");
    }
}
sqlite3_finalize(stmt);
sqlite3_close(db);
```

4.4 Scenarii de utilizare

- Un utilizator nou va trebui sa isi creeze un cont cu comanda: creare cont
- Odata ce utilizatorul are cont, el se va putea loga pe platforma pentru a o putea utiliza
- Utilizatorul va putea cumpara un produs daca soldul lui este mai mare sau egal cu pretul produsuli, in caz contrar, acesta nu va putea achizitiona produsul
- Un utilizator isi va putea adauga sau retrage bani din portofelul implementat in aplicatie
- Un utilizator va putea creea noi oferte si sa le modifice pe cele existente(daca produsul nu a fost deja achizitionat)
- Fiecare utilizator va putea vedea si un istoric al achizitiilor
- Fiecare utilizator va putea vedea propriile oferte si statusul acestora

5 Concluzii

In concluzie, aplicatia creeaza un spatiu in care userii pot vine si cumpara produsele disponibile pe baza soldului curent din aplicatie. Imbunatatirile care ar putea fi aduse aplicatiei sunt criptarea parolelor din baza de date pentru o securitate mai mare si crearea unei interfete grafice pentru a aduce o exoerienta mai placuta utilizatorilor.

References

1. $https://profs.info.uaic.ro/\ computernetworks/cursullaboratorul.php$
2. $https://www.andreis.ro/teaching/computer-networks$
3. $https://profs.info.uaic.ro/\ gcalancea/laboratories.html$
4. $https://www.sqlite.org/cintro.html$