Raport Tema 2 RC

Danila Ana-Maria

[1] Universitatea Alexandru Ioan Cuza, Iasi
[2] Facultatea de Informatica https://www.info.uaic.ro

1 Introducere

Proiectul Top Music ce se doreste implementat prin intermediul proiectului la materia Retele de Calculatoare are ca deziderat implementarea unei aplicatii de tipul client-server care sa realizeze intr-o maniera intuitiva managementul unui top muzical, care contine piese ce pot apartine diverselor genuri muzicale. Din punct de vedere tehnic, aplicatia va fi realizata in limbajul C, punand in evidenta utilizarea paradigmei client-server si a comunicarii intre procese ce ruleaza in locatii diferite in retea, folosind socket-uri. Transmisia se doreste a fi una sigura. Aplicatia pusa in discutie va permite utilizatorilor realizarea diferitelor operatiuni dupa ce acestia sunt logati. Pentru utilizatorii ce nu au un cont se va pune la dispozitie optiunea de a-si face unul. Utilizatorii pot fi de doua tipuri principale, si anume utilizatori simpli, respectiv administratori, care vor avea drepturi superioare. Utilizatorii logati vor putea sa realizeze diverse operatiuni pe topul muzical, operatiuni printre care se numara: adaugarea unei melodii la top, voterea unei melodie, adaugarea unui comentariu pentru o anumita melodie. De asemnea, utilizatorul cu drepturi de administrator va putea sa stearga o melodie din acest top sau chiar sa restrictioneze optiunea altor utilizatori de a vota. In ceea ce priveste afisarea topului muzical, aceasta va putea fi facuta fie in general in functie de numarul de voturi, fie in functie de genul muzical de apartenenta. Entitatile de tip melodie vor contine elemente precum nume, descriere, link, respectiv vor apartine unuia sau mai multor genuri muzicale. Elementele tehnice ce sunt utilizate in realizarea proiectului vor fi prezentate in sectiunile urmatoare.

2 Tehnologii utilizate

Pentru implementarea acestei aplicatii au fost utilizate mai multe tehnologii, care vor fi amintite pe scurt in ceea ce urmeaza.

Aplicatia considerata in acest proiect e bazata pe paradigma client-server si a fost implementata utilizand limbajul C si librariile puse la dispozitie.

Unul din aspectele cele mai importante este tipul de server utilizat pentru aplicatie. Aici am considerat folosirea unui server concurent, care manageriaza clietii. Concurenta este asigurata prin crearea cate unui thread care sa deserveasca fiecare din clientii care solicita diverse lucruri de la server.

Comunicarea dintre client si server am realizat-o utilizand protocolul TCP. Am ales sa fac acest lucru, deoarece am dorit conexiunea dintre procese sa fie

realizata intr-o maniera sigura. Totodata, am dorit comunicarea dintre client si server sa fie orientata conexiune. De asemenea, deoarece in cadrul acestei aplicatii informatiile schimbate sunt de tip text, cu o dimensiune realitv scazuta, faptul ca TCP este mai lent decat UDP nu constituie o problema. Un alt motiv ce a stat la baza acestei alegeri a fost dorinta de a avea un protocol care sa fie sigur si sa realizeze verificarea erorilor de transmisie si recuperarea din acestea.

Am ales realizarea mecanismului de concurenta pe server utilizand thread-uri, nu procese, datorita faptului ca resursele consumate in cazul utilizarii thread-urilor sunt mult mai reduse comparabil cu cazul in care s-ar utiliza procese, si totodata comunicarea dintre thread-uri, in caz de nevoie, poate fi realizata mai usor si mai performant.

In ceea ce priveste persistenta datelor, am considerat necesara existenta unei baze de date relationale, SQLite, care sa permita operatiile de baza pe date (CRUD - create, read, update, delete) intr-o maniera cat mai facila. De asemenea, datorita cerintei care presupune diverse relatii intre tabele, am considerat eficient a modela baza de date utilizand SQLite intr-o manierea relationala. Dintre bazele de date relationale existente, am considerat ca SQLite este cea care permita cea mai facila interactiune din limbajul C, avand de asemenea performante bune pentru aplicatii ca cea pusa in discutie.

Utilizand SQLite, putem stoca intr-un mod persistent datele corespunzatoare entitatilor implicate in aplicatie. SQLite poate fi utiliat intr-un mod facil impreuna cu limbajul C, indiferent de sistemul de operare pe care se lucreaza. Pentru instalarea SQLite pe Ubuntu se procedeaza astfel:

- se porneste un terminal
- se executa comenzile:
- sudo apt-get install sqlite3
- sudo apt-get install libsqlite3-dev
- sudo apt-get install sqlitebrowser (acest utilitar poate fi folosit pentru o operatii in mod interactiv pe baza de date)

Comunicarea cu baza de date este realizata de catre server. Acesta, dupa primirea comenzilor si a datelor necesare de la client, respectiv procesarea acestora, realizeaza operatiile necesare pe baza de date. Aceste operatii sunt operatii CRUD: de inserare (create), de citire (read), de modificare (update) sau de stergere a datelor (delete). Astfel, logarea unui utilizator este o operatie de citire (se verifica daca in baza de date in tabela User exista utilizatorul cu username-ul introdus, respectiv daca parolele corespund). Inregistrarea unui utilizator este o operatie de inserare (se verifica daca mai exista in baza de date vreun utilizator cu acelasi username, iar in caz ca nu, se va insera utilizatorul introdus in baza de date, in tabela User). Adaugarea unei melodii la top este o operatie de inserare (se introduce o intrare in tabela Melodie, respectiv cate o intrare in tabela GenMuzical pentru fiecare gen muzical al melodiei introduse). Votarea unei melodii presupune o operatie de update (se face update in tabela Melodie la intrarea corespunzatoare melodiei selectate de utilizator, mai exact la numarul de voturi, acesta crescand cu o unitate). Adaugarea unui comentariu la o melodie presupune o operatie de inserare (se insereaza o intrare in tabela Comen-

tariu, asociind astfel un anumit comentariu unui anumite melodii si unui anumit utilizator). Stergerea unei melodii din top de catre administrator reprezinta o operatie de stergere (se sterge melodia respectiva din tabela Melodie, respectiv se sterg comentariile corespunzatoare acelei melodii (tabela Commentariu), cat si intrarile din tabela GenMuzical corespunzatoare acelei melodii). La stergere se aplica principiul "on delete cascade". Restrictionarea votului unui utilizator presupune o operatie de update (se modifica valoarea campului corespunzator posibilitatii de votare a unui utilizator din tabela User). Afisarea topului general presupune o operatie de read (se citesc intrarile din tabela Melodie si se sorteaza in ordinea descrescatoare a numarului de voturi). Afisarea topului pentru un anumit gen de muzica presupune o operatie de read (se citesc intrarile din tabela Melodie, a caror gen muzical specificat in tabela GenMuzical e cel specificat de utilizator).

3 Arhitectura aplicatiei

Aplicatia Top Music este dezvoltata in maniera client-server, avand un server concurent, cu concurenta bazata pe thread-uri, care poate deservi clienti multipli. Comunicarea dintre client si server se realizeaza prin protocolul TCP, intr-un mod sigur. Clientul comunica doar cu serverul, iar accesul la baza de date este asigurata de catre server. Acest mod de dezvoltare a aplicatiei permite folosirea acesteia in mod distribuit, clientul si serverul putand comunica pe dispozitive distincte. Clientul trimite catre server request-uri sub o forma independenta de contextul aplicatiei locale (in cazul de fata sub forma de string-uri), serverul interpreteaza aceste request-uri, executa logica de business ceruta de client, in cazul in care este posibil, apoi returnand un raspuns, tot sub o forma de string-uri.

3.1 Schema conceptuala a aplicatiei

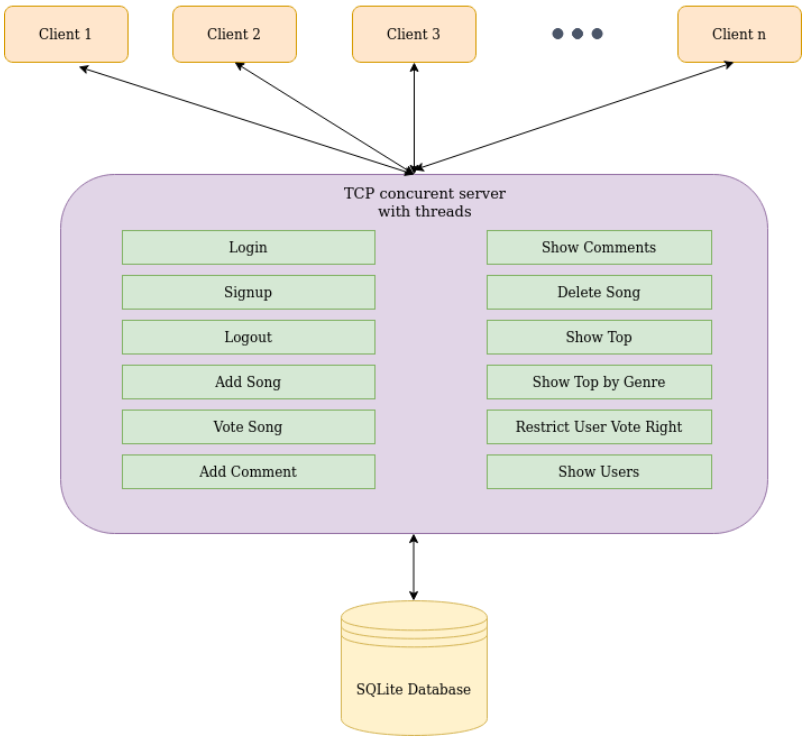

Fig. 1. Arhitectura conceptuala a aplicatiei

3.2 Diagrama bazei de date

Pentru stocarea entitatilor implicate in proiect am ales sa utilizez o baza de date relationala de tip SQLite, iar schema bazei de date folosita e reprezentata in figura 2. Am ales sa fie 4 tabele, dintre care, tabela User cuprinde datele specifice utilizatorului, tabela Song cuprinde principalele date legate de o melodie, in afara genului muzical (care poate fi multiplu) si care am ales a fi reprezentat prin tablea Genre care cuprinde pe langa id-ul unic, id-ul corespunzator melodiei(ca si cheie straina), respectiv genul muzical al acesteia, iar in cele din urma tabela Comment stocheaza comentariile pentru melodii asociindu-le cu userii care le-au adaugat. Pe langa id-ul unic, tabela Comment mai cuprinde si id-ul utilizatorului si cel al piesei (ca si chei straine), respectiv comentariul in sine.

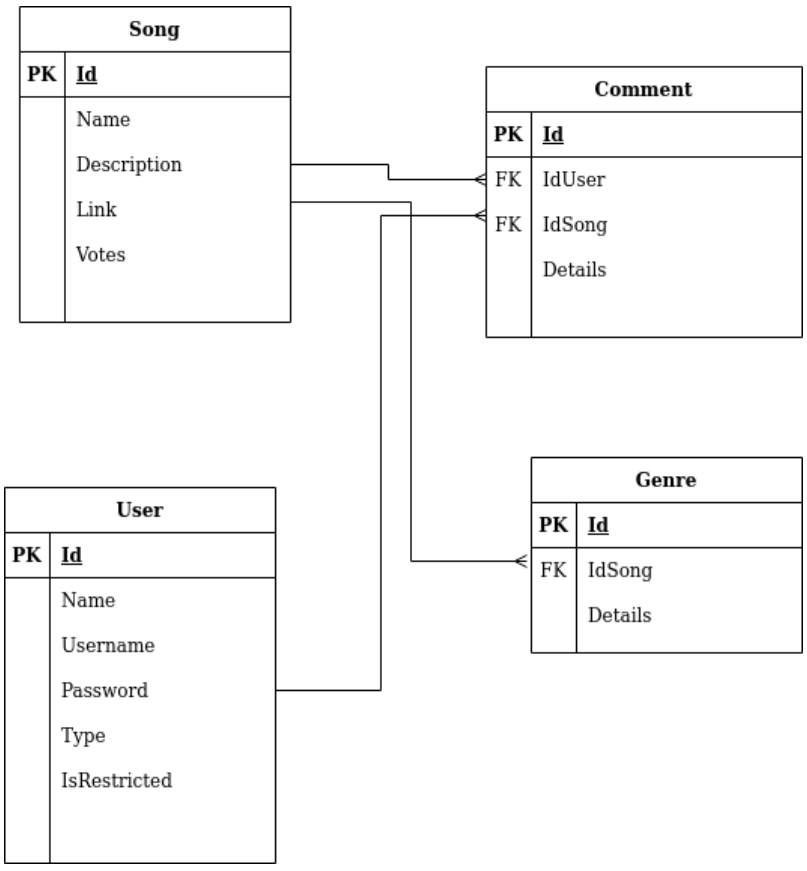

Fig. 2. Diagrama bazei de date

4 Detalii de implementare

4.1 Sectiuni relvante de cod

Pe partea de server e necesar a se crea initial un socket, respectiv a-l configura corespunzator, alaturi de pregatirea structurilor de date ce vor fi utilizate.

```
// crearea unui socket
// configurarea optiunii pentru socket
//pregatirea structurilor de date
  if ((sd = socket (AF_INET, SOCK_STREAM, 0)) == -1)
    {
      perror ("[server]Eroare la socket().\n");
      return errno;
    }
  int on=1;
  setsockopt(sd,SOL_SOCKET,SO_REUSEADDR,&on,sizeof(on));

  bzero (&server, sizeof (server));
  bzero (&from, sizeof (from));
```

Fig. 3. Cod server

Se configureaza familia de socket-uri, adresele acceptate si port-ul considerat. Se incearca atasarea socketu-ului utilizand metoda bind, iar in caz in care totul decurge bine serverul va putea asculta daca vin clienti (metoda listen).

```
// stabilirea familiei de socket-uri
// stabilirea adreselor acceptate (orice adresa)
// utilizarea unui port utilizator
  server.sin_family = AF_INET;
  server.sin_addr.s_addr = htonl (INADDR_ANY);
  server.sin_port = htons (PORT);

// atasarea socket-ului si verificarea posibilelor erori
if (bind (sd, (struct sockaddr *) &server, sizeof (struct sockaddr)) == -1)
  {
    perror ("[server]Eroare la bind().\n");
    return errno;
  }

// serverul asculta daca vin clienti
if (listen (sd, 2) == -1)
  {
    perror ("[server]Eroare la listen().\n");
    return errno;
  }
```

Fig. 4. Cod server

Se mentine serverul intr-o bucla infinita (while(1)), iar la sosirea unui client il acceptam. Se creeaza, de asemenea, cate un thread pentru fiecare client sosit.

```
// servirea concurenta a clientilor
// utilizam thread-uri
while (1)
  {
    int client;
    thData * td; //parametru functia executata de thread
    int length = sizeof (from);

    printf ("[server]Asteptam la portul %d...\n",PORT);
    fflush (stdout);

      // acceptarea clientului si verificare posibilelor erori
    if ( (client = accept (sd, (struct sockaddr *) &from, &length)) < 0)
  {
    perror ("[server]Eroare la accept().\n");
    continue;
  }

  // dupa realizarea conexiunii se astepata
     //id-ul thread-ului si descriptorul intors de accept

  td=(struct thData*)malloc(sizeof(struct thData));
  td->idThread=i++;
  td->cl=client;

  // crearea thread-ului
  pthread_create(&th[i], NULL, &treat, td);

  }
```

Fig. 5. Cod server

Functia treat mangeriaza modul in care sunt alocate thread-urile, respectiv apelul functiei ce va permite comunicarea cu clientul.

```
// functie pentru tratarea unui client
static void *treat(void * arg)
{
        struct thData tdL;
        tdL= *((struct thData*)arg);
        printf ("[thread]- %d - Asteptam mesajul...\n", tdL.idThread);
        fflush (stdout);
        pthread_detach(pthread_self());
        raspunde((struct thData*)arg);
        /* am terminat cu acest client, inchidem conexiunea */
        close ((intptr_t)arg);
        return(NULL);

};
```

Fig. 6. Cod server

Dupa citirea operatiunii pe care clientul doreste sa o execute, se va selecta aceasta operatiune si se va apela functia dedidacta pentru a o implementa.

```
while(1){

    // citirea operatiunii pe care clientul doreste sa o execute
    if (read (tdL.cl, &operatiune, sizeof(int)) <= 0)
            {
              printf("[Thread %d]\n",tdL.idThread);
              perror ("Eroare la read() de la client.\n");

            }

if (operatiune == 0){
//login
    login(tdL);
}
else if(operatiune == 1){
// signup
signup(tdL);
}
else if(operatiune ==2){
// insert melodie
addSong(tdL);
}
```

Fig. 7. Cod server

Conectarea la baza de date sqlite.

```
rc = sqlite3_open(DB, &db);

if (rc)
{
    fprintf(stderr, "Can't open database: %s\n", sqlite3_errmsg(db));
}
else
{
    fprintf(stderr, "Opened database successfully\n");
}
```

Fig. 8. Cod server

Crearea tabelelor din baza de date. Se creaza query-ul corespunzator si se executa.

```
// table comment

sql = "CREATE TABLE COMMENT("
      "ID INTEGER PRIMARY KEY     AUTOINCREMENT,"
      "IDUSER           INT    NOT NULL,"
      "IDSONG       INT NOT NULL,"
      "DETAILS           TEXT,"
      "FOREIGN KEY (IDSONG) REFERENCES SONG (ID) ON DELETE CASCADE, "
      "FOREIGN KEY (IDUSER) REFERENCES USER (ID) ON DELETE CASCADE);";

rc = sqlite3_exec(db, sql, 0, 0, &zErrMsg);

if (rc != SQLITE_OK)
{
    fprintf(stderr, "SQL error: %s\n", zErrMsg);
    sqlite3_free(zErrMsg);
}
else
{
    fprintf(stdout, "Table COMMENT created successfully\n");
}
```

Fig. 9. Cod server

Query-ul corespunzator pentru a extrage datele necesare pentru realizarea login-ului. Dupa scrierea query-ului, acesta se pregateste si se extrag datele din randul obtinut, in cazul in care acesta exista.

```
sprintf(sqlQuery, "select isrestricted, type from user where USERNAME='%s' and PASSWORD='%s'", username, password);

sql = sqlQuery;

puts(sql);

strcpy(loginResponse, "");

sqlite3_prepare_v2(db, sql, -1, &stmt, NULL);

int step = sqlite3_step(stmt);

if (step != SQLITE_ROW)
{
    fprintf(stderr, "SQL error: %s\n", zErrMsg);
    sqlite3_free(zErrMsg);
    strcpy(loginResponse, "Bad login");
}
else
{
    fprintf(stdout, "Operation done successfully\n");
    sprintf(loginResponse, "%s;%s", sqlite3_column_text(stmt, 0), sqlite3_column_text(stmt, 1));
}

sqlite3_finalize(stmt);
sqlite3_close(db);
```

Fig. 10. Cod server

Generare topului pentru un anumit gen muzical utilizand un query complex (cu join), rezultatele de pe fiecare rand returnat de baza de date fiind salvate pentru a putea fi transmise clientului.

```
sprintf(sqlQuery, "select s.id, s.name, s.description, s.link, s.votes from SONG s join GENRE g on (s.id=g.idsong) where g.details='%s' order by s.votes desc", genMuzical);

sql = sqlQuery;

puts(sql);

sqlite3_prepare_v2(db, sql, -1, &stmt, 0);

strcat(topResponse, "id|name|description|link|votes\n_____\n");

while (sqlite3_step(stmt) != SQLITE_DONE)
{
    //char raspPartial[1000]="";
    int i;
    int num_cols = sqlite3_column_count(stmt);

    for (i = 0; i < num_cols - 1; i++)
    {

        strcat(topResponse, sqlite3_column_text(stmt, i));
        strcat(topResponse, "|");
    }
    strcat(topResponse, sqlite3_column_text(stmt, i));

    strcat(topResponse, "\n");
}

sqlite3_finalize(stmt);

puts(topResponse);

sqlite3_close(db);
```

Fig. 11. Cod server

Clientul la apel trebuie sa precizeze atat numele serverului cu care doreste sa se conecteze, cat si portul pe care ruleaza aplicatia considerata.

```
// preluare argumente de conectare din linia de comanda
if (argc != 3)
  {
    printf ("Sintaxa: %s <adresa_server> <port>\n", argv[0]);
    return -1;
  }

//stablim portului
port = atoi (argv[2]);
```

Fig. 12. Cod client

Pe client se fac configurari ale socket-ului similare cu cele de pe server, dar se seteaza o adresa specifica si anume cea preluata din linia de comanda.

```
// creare socket si configurare similar cu serverul
// adresa este preluata din linia de comanda
if ((sd = socket (AF_INET, SOCK_STREAM, 0)) == -1)
  {
    perror ("Eroare la socket().\n");
    return errno;
  }

server.sin_family = AF_INET;
server.sin_addr.s_addr = inet_addr(argv[1]);
server.sin_port = htons (port);

// conectare la server si verificarea posibilelor erori
if (connect (sd, (struct sockaddr *) &server,sizeof (struct sockaddr)) == -1)
  {
    perror ("[client]Eroare la connect().\n");
    return errno;
  }
```

Fig. 13. Cod client

Principalele metode utilizate in ceea ce priveste comunicarea client-server sunt:

- socket() - Metoda care creează un nou terminal al conexiunii dintre entitati
- bind() - Metoda ce are rolul de a atasa o entitate la socket
- listen() - Metoda ce permite observarea clientilor care incearca sa se conecteze si acceptarea lor
- accept() - Metoda care efectueaza efectiv acceptarea clientului si blocarea acestuira pana la conectare
- connect() - Metoda pentru realizarea conexiunii

4.2 Cazuri de utilizare

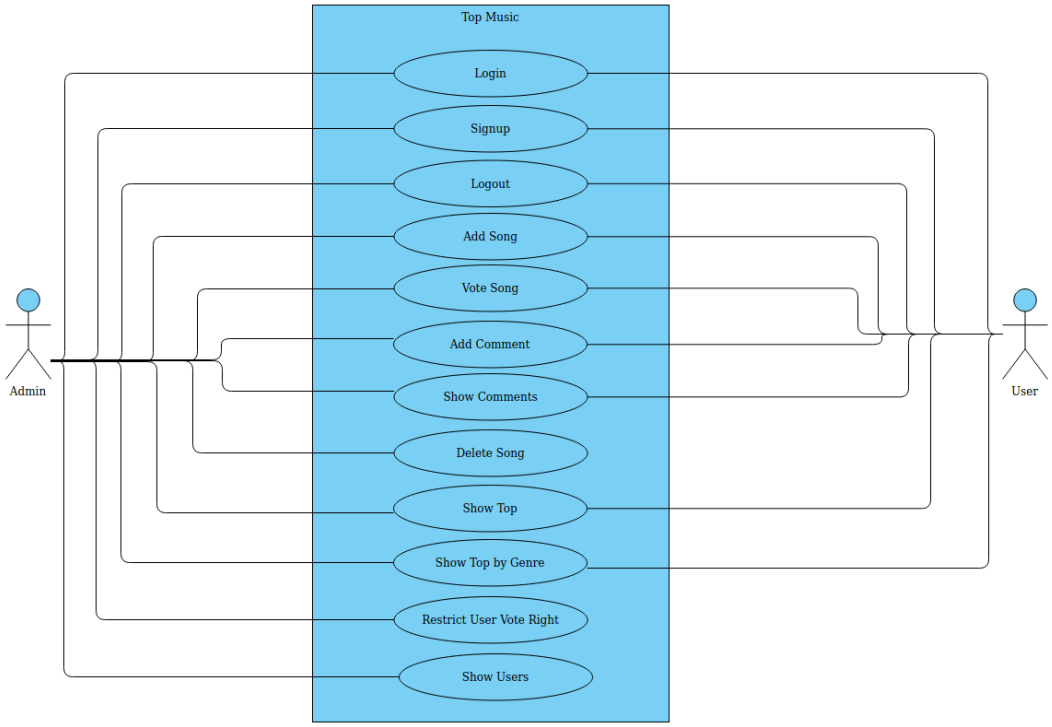

Fig. 14. Diagrama cazurilor de utilizare

Cazurile de utilizare ale acestei aplicatii sunt multiple. Una din principalele diferentiere e cea bazata pe tipul de utilizator care interactioneaza cu sistemul. Conform specificatiilor de utilizare, administratorul va putea realiza cateva lucrui in plus fata de utilizatorul normal, si anume afisarea tuturor utilizatorilor, stergerea unei melodii, respectiv restrictionarea dreptului de vot al utilizatorilor.

Cazurile de utilizare sunt prezentate schematic in diagrama, si vor fi detaliate textual in cele ce urmeaza.

Login-ul poate fi realizat de catre oricine. Utilizatorul introduce in consola username-ul si parola. In caz ca acestea nu exista sau a facut o greseala de scriere, acesta va fi notificat printr-un mesaj corespunzator, putand sa aleaga din nou daca doreste sa se logheze sau sa isi faca un cont nou.

Signup-ul poate fi realizat de catre oricine. Prin utilizarea formularului de signup pot fi creati doar utilizatori simpli, nu administratori. Utilizatorul va introduce numele, username-ul, parola. In cazul in care mai exista un utilizator cu acelasi username, se va primi un mesaj de eroare, respectiv utilizatorul va

putea sa aleaga din nou daca sa isi faca un cont nou sau sa se logheze in caz ca si-a amintit contul.

Logout este o functionalitate complementara functiei de login. Orice utilizator logat poate folosi aceasta functionalitate pentru a se deloga de pe contul propriu. Pentru aceasta functionalitate, nu exista scenarii de eroare.

Adaugarea unei melodii poate fi facuta de catre orice utilizator care este logat. Acesta va introduce numele piesei, descrierea acesteia, link-ul de YouTube, respectiv genurile muzicale carora le apartine piesa. In cazul in care acesta introduce o piesa care deja este introdusa, va fi notificat printr-un mesaj de eroare corespunzator, avand posibilitatea sa aleaga din nou ce operatie va realiza.

Votarea unei melodii poate fi realizata de catre orice utilizator logat, care are drept de vot. Acesta introduce numele piesei pe care doreste sa o voteze, iar daca procesul are loc cu succes va primi un mesaj corespunzator. In cazul in care utilizatorul nu are drept de vot va primi un mesaj corespunzator, care sa ii sugereze acest lucru. In cazul in care utilizatorul introduce o piesa care nu exista, acesta va fi atentionat, de asemenea, printr-un mesaj corespunzator. Dupa aparitia oricarui caz de eroare, sau dupa finalizarea cu succes a operatiei, utilizatorul va putea alege din nou ce operatie doreste sa faca.

Adaugarea unui comentariu la o melodie poate fi realizata de orice utilizator. Pentru a adauga un comentariu e necesar a fi introduse numele piesei careia se doreste sa i se adauge un comentariu, respectiv comentariul efectiv. In cazul in care piesa introdusa nu exista utilizatorul va primi mesaj corespunzator, putand sa aleaga din nou ce doreste sa faca.

Afisarea tuturor comentariilor pentru o anumita melodie este un caz de utilizare ce poate fi realizat de catre orice utilizator logat. Se selecteaza optiunea corespunzatoare acestui caz de utilizare, iar apoi utilizatorul trebuie sa introduca numele piesei pentru care doreste sa vada comentariile. Acesta va primi ca raspuns comentariile corespunzatoare acelei piese. In cazul in care acesta introduce o piesa care nu exista, nu va primi ca raspuns niciun comentariu. Lipsa comentariilor pentru o anumita piesa poate semnifica, fie ca nu exista comentarii disponibile, fie ca piesa introdusa nu exista. Dupa realizarea acestui caz de utilizare, utilizator va fi redirectionat la meniul principal pentru a putea realiza alte actiuni dorite.

Stergerea unei melodii poate fi realizata doar de catre administrator. Acesta o data logat va putea sa aleaga sa efectueze stergerea unei melodii. Pentru a face asta este necesar sa introduca numele piesei pe care doreste sa o stearga. In cazul in care numele piesei introduse pentru stergere nu exista, administratorul va fi notificat printr-un mesaj corespunzator, respectiv va fi redirectionat catre meniul de unde va putea alege ce doreste sa faca.

Restrictionarea dreptului de vot al unui utilizator este o operatiune specifica doar administratorului. Acesta, in cazul in care, din motive obiective, considera necesar a suspenda dreptul de vot al unui utilizator, va putea face acest lucru. Acesta, dupa ce va vizualiza utilizatorii existenti, va introduce username-ul utilizatorului caruia doreste sa ii restrictioneze dreptul de vot. In cazul in

care username-ul introdus nu exista, adminsitratorul va fi notificat, respectiv va putea sa aleaga ce operatiune doreste sa realizeze in continuare.

Afisarea tuturor utilizatorilor este un caz de utilizare specific administratorului. Astfel, o data logat, administratorul va putea introduce comanda specifica pentru afisarea tuturor utilizatorilor, fara a fi necesar sa introduca orice altceva. Acesta va putea vizualiza lista utilizatorilor existenti in consola. Acest caz de utilizare nu are scenarii de eroare in functionare.

Afisarea topului general este un caz de utilizare care poate fi realizat de catre orice utilizator. Orice utilizator logat, poate introduce comanda specifica pentru afisarea topului general, fara a mai fi nevoie sa introduca altceva. Acesta va putea vizualiza topul cerut in consola. Pentru acest caz de utilizare nu se identifica scenarii de eroare.

Afisarea topului pentru un anumit gen muzical este un caz de utilizare ce poate fi realizat de catre orice utilizator logat. Dupa selectarea optiunii corespunzatoare acestui caz de utilizare, cel ce foloseste aplicatia va introduce si genul muzical pentru care doreste realizarea topului. Acesta va primi ca raspuns topul solicitat. In cazul in care genul muzical introdus nu exista, utilizatorul va fi notificat printr-un mesaj corespunzator, revenind la meniul principal de unde va putea alege ce doreste sa realizeze in continuare.

Exemplu concret: Maria doreste sa asculte muzica si nu stie ce sa aleaga. Ea are cont pe aplicatia Top Music si doreste sa o utilizeze pentru a afla ce e in trend acum. Porneste aplicatia, introduce optiunea de logare, apoi username-ul (maria) si parola (maria). Dorind sa vada topul melodiilor introduce optiunea de meniu corespunzatoare. Apoi, dupa ce acceseaza link-ul corespunzator primei melodii din topul sugerat de aplicatie, isi da seama ca e o melodie foarte frumoasa si doresteste sa o voteze, pentru ca sa ramana in top si sa poata fi ascultata de cat mai multi utilizatori. Pentru aceasta, va introduce optiunea de meniu corespunzatoare votarii unei melodii, iar apoi va introduce numele piesei pe care doreste sa o voteze.

5 Concluzii

Aplicatiea prezentata in acest raport reprezinta un prototip functional a ceea ce o astfel de aplicatie ar putea fi. Desi aplicatia permite utilizarea corespunzatoare pentru scopul in care a fost creata, totusi aceasta mai permite o serie de imbunatatiri ulterioare, care pot ajuta la sporirea eficientei si sigurantei in utilizarea acesteia. Printre cele mai de seama imbunatari ce se pot aduce acestei apliatii se numara crearea unei interfete utilizator, care sa permita introducerea diverselor date intr-un mod mai intuitiv. O alta imbuatatire ce ar putea fi adusa face referire la modul in care se realizeaza login-ul si signup-ul. Pentru acestea parola ar putea sa fie ascunsa, iar inainte de transmiterea acesteia spre server ar putea sa fie criptata din motive de siguranta a transmisiei. De asemenea, in ceea ce priveste modul de afisare al topurilor muzicale, ar fi utila existenta mai multor modalitati de afisare. De asemenea, aplicatia ar putea sa puna la dispozitie o mesagerie prin care utilizatorii aplicatiei sa poata comunica intre ei, unul cu altul.

Astfel, prin cele expuse anterior, aplictia Top Music implementata in cod si descrisa prin acest raport si-a indeplinit obiectivele propuse, functionarea acesteia fiind comform cerintelor. Desigur, ca aceasta mai poate suporta imbunatatiri ulterioare, pentru a deveni o versiune mai buna si eficienta.

References

1. Server TCP concurent prin thread-uri, https://profs.info.uaic.ro/ computernetworks/files/NetEx/S12/ServerConcThread/servTcpConcTh2.c
2. Client TCP pentru server concurent, https://profs.info.uaic.ro/ computernetworks/files/NetEx/S12/ServerConcThread/cliTcpNr.c
3. SQLite Documentation in C, https://www.tutorialspoint.com/sqlite/sqlite_c_cpp.htm
4. SQLite Tutorial in C, http://zetcode.com/db/sqlitec/
5. SQLite Example in C, https://gist.github.com/jsok/2936764