TopMusic

Griţco Sorin

Facultatea de Informatica, Universitatea Alexandru Ioan Cuza, Iaşi

1 Introducere

Am ales să implementez acest proiect cu scopul de a crea o aplicţie server/client care să facă managementul unui top muzical, ce conţine genuri diverse de muzică. Aplicaţia va putea fi accesată, doar de utilizatorii înregistraţi în sistem, prin intermediul unei parole şi login, ce vor fi memorate într-o bază de date. După autentificare, utilizatorii vor putea accesa aplicaţia prin intermediul logării.

În momentul în care logarea a reuşit, utilizatorul va putea introduce la top o nouă melodie, va putea acorda o notă melodiilor, astfel schimbând media aritmetică a notelor şi influenţând asupra clasamentului melodiilor. Utilizatorul va putea să filtreze melodiile după genul muzical sau/şi după note, la fel va putea accesa melodiile prin intermediul unui link pe youtube sau alte site-uri.

Utilizatorii vor putea să posteze comentarii, în secţiunea comentarii ale melodiei. După utilizarea aplicaţiei, ei vor putea să se delogeze. Doar, administratorul va putea şterge melodiile din top şi să restricţioneze activitatea de votare a utilizatorilor obişnuiţi.

2 Tehnologii utilizate

Pentru implementarea acestui proiect voi utiliza modelul client/server TCP concurent, cu scopul ca mai mulţi utilizatori să poată folosi aplicaţia în acelaşi timp (în mod concurent).

Datele despre utiliztori, login-ul şi parola vor fi memorate într-o bază de date MySQL, login-ul va fi în clar(plaintext), iar parola va fi criptată prin intermediul funcţiei hash SHA-256, pentru a securiza utilizatorii de la o posibilă spargere a bazei de date.

Numele melodiilor vor fi memorate într-o bază de date MySQL, pe lângă numele acestora, vor exista în coloane diferite, genurile aferente, nota generală acestora şi link-ul către youtube sau alte site-uri.

Comentariile şi genurile muzicale vor fi memorate în tabele diferite, corespondenţa dintre melodii şi genuri se va face pe baza cheilor străine, la fel şi pentru comentariile introduse.

3 Detalii de implementare

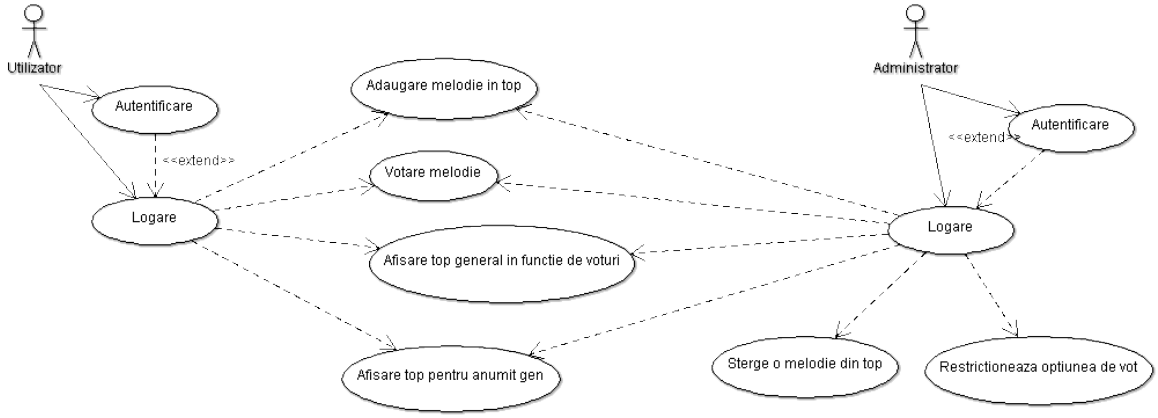
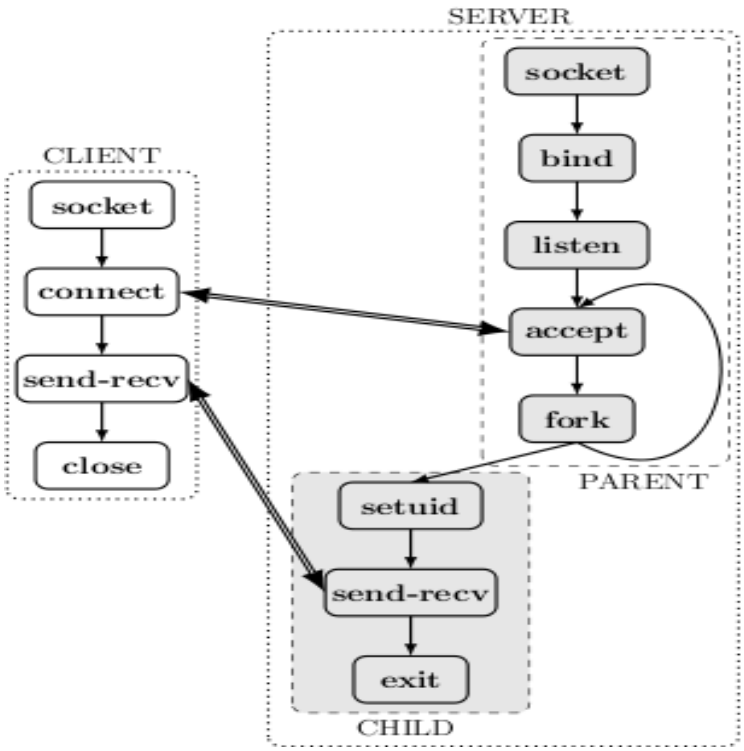


Diagrama Use Case

Modelul TCP de implementat[1]:



Cod relevant:

1. **Serverul TCP concurrent:**

```
#include <sys/types.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <errno.h>
#include <unistd.h>
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
#include <sys/wait.h>

/*port utilizat*/

#define PORT 2025

extern int errno;

int main(){
    struct sockaddr_in server;  //structura folosita
de server
    struct sockaddr_in from;

    char msg[100];              //mesajul primit de
la client
    char msgrasp[100] = " ";    //mesaj de raspuns
pentru client
    int sd;                     //descriptor 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*/

    if (bind(sd, (struct sockaddr *)&server,
sizeof(struct sockaddr))==-1){
        perror("[server]Eroare la bind().\n");
        return errno;
    }

    if(listen(sd, 5)==-1){
        perror("[server]Eroare la listen().\n");
        return errno;
    }

    while(1){
        int client;
        int child_pid;
        int length = sizeof(from);

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

        /*acceptam un client (stare blocanta pana la
realizarea conexiunii)*/
        client = accept(sd, (struct sockaddr *)
&from, &length);

        /*eroare la acceptarea conexiunii de la un
client*/

        if((child_pid = fork())<0){
```

```
            perror("Eroare la fork.");
        }
        else if(child_pid==0){ //suntem in copil
            close(sd);

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

            /*s-a realizat conexiunea, se asteapta
mesajul*/

            bzero(msg, 100);
            printf("[server] Asteptam
mesajul...\n");
            fflush(stdout);

            if(read (client, msg, 100)<=0){
                perror("[server]Eroare la read() de
la client.\n");
                close(client);  /*Inchidem
conexiunea cu clientul*/
                continue;
            }

            printf("[server]Mesajul a fost
receptionat...%s\n", msg);

            /*pregatim mesajul de raspuns*/
            bzero(msgrasp, 100);
            strcat(msgrasp, "Hello ");
            strcat(msgrasp, msg);

            printf("[server]Trimitem mesaju
inapoi...%s\n", msgrasp);

            /*returnam mesajul clientului*/

            if(write(client, msgrasp, 100)<=0){
                perror("[server]Eroare la write()
catre client.\n");
                continue; /*contiunuam sa ascultam*/
            }
```

```
            else
                printf("[server] Mesajul a fost
transmis cu succes.\n");
                close(client);
                exit(1);
        }
        wait(NULL); //parinte
    }
}
```

2. Conectarea la o bază de date MySQL[3]

```
/* Simple C program that connects to MySQL Database
server*/
#include <mysql.h>
#include <stdio.h>

main() {
   MYSQL *conn;
   MYSQL_RES *res;
   MYSQL_ROW row;

   char *server = "localhost";
   char *user = "root";
   char *password = "PASSWORD"; /* set me first */
   char *database = "mysql";

   conn = mysql_init(NULL);

   /* Connect to database */
   if (!mysql_real_connect(conn, server,
         user, password, database, 0, NULL, 0)) {
      fprintf(stderr, "%s\n", mysql_error(conn));
      exit(1);
   }

   /* send SQL query */
   if (mysql_query(conn, "show tables")) {
      fprintf(stderr, "%s\n", mysql_error(conn));
      exit(1);
   }

   res = mysql_use_result(conn);

   /* output table name */
   printf("MySQL Tables in mysql database:\n");
   while ((row = mysql_fetch_row(res)) != NULL)
```

```
        printf("%s \n", row[0]);

    /* close connection */
    mysql_free_result(res);
    mysql_close(conn);
}
```

3. Criptarea cu SHA256[3]:

```
Requires OpenSSL, compile flag: -lssl -lcrypto

#include <stdio.h>
#include <string.h>
#include <openssl/sha.h>

int main (void) {
        const char *s = "Rosetta code";
        unsigned char *d = SHA256(s, strlen(s), 0);

        int i;
        for (i = 0; i < SHA256_DIGEST_LENGTH; i++)
                printf("%02x", d[i]);
        putchar('\n');

        return 0;
}
```

4 Concluzii

Pentru a îmbunătăţi soluţia propusă, aş putea să utilizez o librărie pentru grafică pentru a impelementa o interfaţă prietenoasă pentru utilizator.

5 Bibiliografie

1. https://www.usenix.org/legacy/events/sec08/tech/full_papers/radhakrishnan/radhakrishnan_html/node6.html
2. https://www.cyberciti.biz/tips/linux-unix-connect-mysql-c-api-program.html
3. https://rosettacode.org/wiki/SHA-256#C