Aplicație de streaming muzical

Proiectul nostru constă într-o aplicație de streaming muzical care permite încărcarea melodiilor, vizualizarea melodiilor existente și descărcarea / ascultarea unor melodii.

Protocolul constă dintr-un schimb de pachete dintre client și server. Buffer-ul are dimensiune 1024 de octeți. Avem următoarele tipuri de mesaje:

- Descărcare melodie: 1 | ID melodie
- Verificare încărcări: 2
- Upload melodie: 3 | ID cerere | nr de octeți de informație | titlul|gen
- Pachet cu număr de pachete: 4 | nr de pachete care urmează | request ID
- Pachet cu octeți: 5 | nr de octeți care urmează cu melodia |request ID | informație din melodie
- Pachet de confirmare: 6 | request ID
- Deconectare clienți: 8
- Pachet care arată conectarea administratorului: 9
- Închidere server: 10
- Vizualizare număr clienți conectați: 11

În funcție de header acestea au semnificații și structuri diferite. În continuare vom enumera operațiile posibile și schimbul de mesaje între client și server.

- Descărcare melodie: clientul dorește sa descarce o melodie de pe server. Clientul trimite un pachet de tip 1, unde specifică melodia dorită; serverul trimite un pachet de tip 4 cu totalul pachetelor care urmează sa fie trimise și ID-ul cererii, după care atâta timp cât avem pachete de trimis, serverul trimite pachete de tip 5, iar clientul pachete de tip 6 pe post de confirmare.

- Verificare încărcări: clientul trimite un pachet de tip 2, prin care cere să i se trimită datele despre toate melodiile; atâta timp cât se vor trimite pachete de tip 7 (adică atâta timp cât avem melodii), clientul va trimite confirmarea pachete de tip 6.

- Încărcare melodie: clientul va trimite un pachet de tip 3, unde va specifica datele despre noua melodie, după care atâta timp cât sunt pachete de tip 5 de trimis cu melodia efectivă, serverul va trimite confirmare la un pachet de tip 6. La final se va trimite un pachet de tip 7 cu o sugestie aleatoare.

- Deconectare clienți: administratorul va trimite un pachet de tip 8, după care serverul va trimite un pachet de tip 6 drept confirmare și va deconecta toți clienții.

- Închidere server: administratorul trimite un pachet de tipul 10, iar serverul se va închide.

- Vizualizare număr clienți conectați: administratorul trimite un pachet de tipul 11, iar serverul returnează număr de clienți conectați în momentul apelului.

Ascultare melodie

Procesul pentru a asculta o melodie se realizează astfel:

- clientul introduce în terminal opțiunea 1, care reprezintă ascultarea melodiei, după care specifică id-ul melodiei pe care o dorește
- programul trimite pachetul corespunzător către server
- serverul trimite un pachet în care specifică numărul de pachete care vor fi trimise pentru a descărca toată melodia
- programul clientului extrage numărul de pachete și id-ul cererii, după care trimite către server o confirmare că a primit pachetul de tip 4
- aplicație client creează și deschide pentru scriere fișierul ".mp3" în care va fi scrisă melodia cerută
- programul client într-o iterație de la 0 la numărul de pachete primite anterior în care citește pachetul primit, verifică câți octeți a primit, după care scrie datele în fișier, sărind header-ul pachetului
- la fiecare iterație clientul trimite către server o confirmare pentru a continua primirea pachetelor
- cand iterația este gata, fișierul este închis, melodia aflându-se acum pe calculatorul personal al clientului pentru a o putea asculta oricând
- melodia este redata automat imediat după descărcare. Pentru acest lucru a fost folosită librăria "vlc"

```
audio = fopen(songTitle, "wb");
int tmp;
for(i = 0; i < songSize; i++){
    if ((tmp = recv(clientSocket, data, dataSize, 0)) < 0){
        fprintf(stderr, "Error reading data");
    }
    fwrite(data+12, sizeof(char), tmp-12, audio);
    snprintf(data, dataSize, "---%d---%d---%c", 6, requestId, data[11]);
    if (send(clientSocket, data, 12, 0) == -1){
        fprintf(stderr, "Error sending data\n");
    }
}
fclose(audio);
```

```
audio = open(songTitle, "wb")
for i in range(0, songSize):
    data = clientSocket.recv(dataSize)
    audio.write(data[12:])
    data = '0'*dataSize
    data = "---6---{reqID}---{reqID}".format(reqID = requestId)
    clientSocket.send(data.encode())

audio.close()
```

```
PrintWriter writer = new PrintWriter(songTitle);
for(i = 0; i < songSize; i++){
    writer.print(data);
    data = String.format("---%d---%d---%d", 6, requestId, requestId);
    data = sendRequestType(headers, data);
}
writer.close();
```

Încărcare melodie

Procesul pentru a încarca o melodie se realizează astfel:

- utilizatorul introduce titlul melodiei, genul acesteia și adresa din calculator la care se află fișierul pe care dorește să îl încarce
- aplicația client deschide în modul de citire fișierul de la adresa menționată și returnează numărul de octeți pe care îi conține, după care calculează numărul de pachete necesare pentru a trimite întreaga melodie
- informațiile despre numele, genul melodiei și numărul de pachete sunt trimise către server într-un pachet de tipul 3
- serverul trimite un pachet de confirmare
- aplicația client extrage din pachetul de confirmare id-ul cererii pentru a-l trimite ulterior
- melodia este citită într-o variabilă în cazul clientului C, iar clientul python citește pe rând părți din melodie în iterație
- aplicația intră într-o iterație de la 0 la numărul de pachete care va fi trimis, calculează câți bytes vor fi trimiși (n bytes) și copiază într-o variabilă acei n bytes
- trimite apoi către server un pachet de tipul 5 în care menționează numărul de bytes ce vor fi trimiși, id-ul cererii și apoi datele efective din melodie
- serverul trimite o confirmare, iar după primirea acesteia clientul continuă transmiterea datelor

```
for i in range(0, packetsNo):
    if octSize >= 1012:
        toSendSize = 1012
    else:
        toSendSize = octSize
    octSize -= 1012
    sentAudio = audio.read(1012)

    if toSendSize > 999:
        data = "---5{size}---{id}".format(size = toSendSize, id = requestId)
    elif toSendSize > 99:
        data = "---5-{size}---{id}".format(size = toSendSize, id = requestId)
    elif toSendSize > 9:
        data = "---5--{size}---{id}".format(size = toSendSize, id = requestId)
    else:
        data = "---5---{size}---{id}".format(size = toSendSize, id = requestId)
    data = data.encode()
    data += sentAudio

    clientSocket.send(data)
    clientSocket.recv(dataSize)
```

```
fread(songbytes, sizeof(char), octSize, audio);
for(i = 0; i < packetsNo; i++){
    if(octSize >= 1012){
        toSendSize = 1012;
    }else{
        toSendSize = octSize;
    }
    octSize -= 1012;
    memcpy(sentAudio, songbytes + (i*1012), toSendSize);

    if(toSendSize > 999){ //4 positions
        snprintf(data, dataSize, "---%d%d---%d", 5, toSendSize, requestId);
    }else if(toSendSize > 99){ //3 positions
        snprintf(data, dataSize, "---%d-%d---%d", 5, toSendSize, requestId);
    }else if(toSendSize > 9){ //2 positions
        snprintf(data, dataSize, "---%d--%d---%d", 5, toSendSize, requestId);
    }else{ //1 position
        snprintf(data, dataSize, "---%d---%d---%d", 5, toSendSize, requestId);
    }
    memcpy(data+12, sentAudio, toSendSize);
    if(send(clientSocket, data, toSendSize + 12, 0) ==-1){
        fprintf(stderr, "Error sending data");
    }
    if (recv(clientSocket, data, dataSize, 0) < 0){
        fprintf(stderr, "Error reading data");
    }
}
```

Verificare încărcări

- clientul trimite un pachet de tipul 2 pentru a primi lista cu toate melodiile aflate în baza de date, după care afișează lista și trimite un pachet de confirmare

```
snprintf(data, dataSize, "---%d", 2);
if(send(clientSocket, data, strlen(data) + 1, 0) ==-1){
    fprintf(stderr, "Error sending data");
}
if (recv(clientSocket, data, dataSize, 0) < 0){
    fprintf(stderr, "Error reading data");
}

printf("\nThe available songs: \n");
for(i = 72; i< strlen(data); i++){
    printf("%c", data[i]);
}
```

```
data = "---2"
clientSocket.send(data.encode())
print("\nSent package: \n", data)

data = clientSocket.recv(dataSize)
data = data.decode("utf-8", "ignore")

print("\nThe available songs:\n", data[72:])
```

```
data = String.format("---%d", 2);
sendRequestType(headers, data);
```

```
String response = restTemplate.exchange(path, HttpMethod.POST, entity, String.class).getBody();
System.out.flush();
for(int i = 72; i < response.length(); i++)
    System.out.print(response.charAt(i));
```

Funcții administrator

- atunci când un administrator se conectează, programul trimite către server un pachet de tipul 9 care îl anunță faptul că un admin este conectat și astfel își primește drepturile
- pentru a deconecta clienții, acesta trimite un pachet de tipul 8
- pentru a închide serverul trimite un pachet de tipul 10
- pentru a vedea câți clienți sunt conectați, trimite un pachet de tipul 11, după care serverul returnează informația cerută

```
case 3:
    if(send(clientSocket, "--10", 4, 0) == -1){
        fprintf(stderr, "Error sending data\n");
        exit(EXIT_FAILURE);
    }
    goto exit_admin;
    break;

case 4:
    if(send(clientSocket, "---8", 4, 0) == -1){
        fprintf(stderr, "Error sending data\n");
        exit(EXIT_FAILURE);
    }
    break;

case 5:
    if(send(clientSocket, "--11", 4, 0) == -1){
        fprintf(stderr, "Error sending data\n");
        exit(EXIT_FAILURE);
    }

    if (recv(clientSocket, data, dataSize, 0) < 0){
        fprintf(stderr, "Error reading data");
    }
    fprintf(stderr, "There are %s clients connected\n", data);
    break;
```

Dependențele proiectului

Pentru a putea rula proiectul este necesară instalarea următoarelor librării: vls și python-vlc. Clientul REST necesită Maven, versiunea 2.5.6, iar limbajul de programare folosit este Java 11.