

UNIVERSITATEA "ALEXANDRU-IOAN CUZA" DIN IASI

FACULTATEA DE INFORMATICA



LUCRARE DE LICENTA

Songsurf

propusă de

Manuela-Mihaela Berea

Sesiunea: iulie, 2023

Coordonator științific

drd. Olariu Florin

UNIVERSITATEA "ALEXANDRU-IOAN CUZA" DIN IASI

FACULTATEA DE INFORMATICA

Songsurf

Manuela-Mihaela Berea

Sesiunea: iulie, 2023

Coordonator științific

drd. Olariu Florin

Abstract

"Songsurf" este o aplicație muzicală dezvoltată pentru sistemul de operare Android care îmbină calități și funcții a mai multor plaforme de succes asemănătoare. Printre aceste funcționalități se enumeră: navigarea rapidă și interactivă printr-un volum mare de videoclipuri muzicale, filtrarea acestora pe baza unui set divers de caracteristici ale melodiilor, integrarea unui sistem de recomandare bazat pe conținut și pe interacțiunile utilizatorului și crearea de playlist-uri personalizate.

Cuprins

Introducere	2
Motivătie	4
Contribuții	5
1 Aplicații similare	7
1.1 Youtube Music	7
1.2 Spotify	8
2 Arhitectura aplicației	10
2.1 Arhitectura Client-Server	10
2.1.1 Client	12
2.1.2 Server	12
2.2 Modelul arhitectural MVC (Model-View-Controller)	12
2.3 Scenarii de utilizare	14
3 Detalii de implementare	16
3.1 Backend-ul aplicației	16
3.1.1 Tehnologii utilizate	16
3.1.2 Proiectarea bazei de date	18
3.1.3 Popularea bazei de date	21
3.1.4 Procesarea cererilor de la client și returnarea răspunsurilor corespunzătoare	24
3.1.5 Sistemul de recomandare de melodii	27
3.2 Frontend-ul aplicației	29
3.2.1 Tehnologii utilizate	29
3.2.2 Descrierea implementării	31

4 Ghidul de utilizare	35
4.1 Navigarea printre melodii	35
4.1.1 Adaugarea filtrelor	36
4.2 Navigarea printre playlist-uri	37
4.3 Afisarea playlist-urilor	37
4.4 Administrarea contului	39
Concluzii	40
Bibliografie	42

Introducere

Muzica reprezintă mult mai mult decât o simplă formă de divertisment, ea reprezintă un limbaj universal ce depășește granițele culturale și lingvistice. Având puterea de a exprima emoții puternice, deseori muzica devine o oglindă a sufletului uman, reflectând experiențele și trăirile noastre.

Descoperirea de noi melodii reprezintă un scop pentru mulți dintre noi, fiecare având gusturi și pasiuni diferite, însă această călătorie poate deveni dificilă și derulantă în imensitatea oceanului sonor. În acest sens, recomandările de melodii personalizate oferă o oportunitate unică de a modela această călătorie în funcție de gusturile, experiențele și stările noastre de dispoziție.

În loc să căutăm printr-o mare de melodii cu speranța de a găsi una care să ne placă, platformele muzicale personalizate ne ajută să ne lărgim orizonturile muzicale. Prin sugerarea de melodii din genuri sau artiști pe care nu i-am explorat în mod normal, acestea ne încurajează să ieșim din zona noastră de confort și să experimentăm noi peisaje muzicale. Acest lucru nu numai că ne amplifică aprecierea pentru diversitatea muzicii, dar poate duce și la dezvoltarea noastră personală și culturală.

Soluția acestei probleme ar fi crearea unei platforme muzicale care transmite multă informație într-un timp scurt. Mai precis, această aplicație va prezenta diverse fragmente din videoclipuri muzicale mai mult sau mai puțin populare din care utilizatorul își poate alege favoritele.

Aplicația se numește Songsurf, numele făcând referire la sportul nautic „surfing”, sugerând rapiditatea și ușurință cu care utilizatorul navighează printre melodii printr-un simplu gest. Aceasta își poate personaliza singur muzica aplicând diverse filtre, sau poate primi recomandări din partea aplicației pe baza melodiilor apreciate. Aceste recomandări sunt generate prin identificarea similarităților dintre caracteristicile melodiilor apreciate, cum ar fi genul muzical sau limba în care sunt interpretate. Astfel, utilizatorul își poate crea playlist-uri personalizate, care nu doar că îl ajută să își orga-

nizeze melodiile favorite într-un mod convenabil, ci oferă și o modalitate de a redescoperi și de a se bucura de muzică în orice moment.

Platforma îmbină funcționalități din mai multe aplicații muzicale de succes de pe piață, precum Youtube sau Spotify. Principala diferență dintre aplicația mea și cele enumerate anterior este aceea că ele se axează mai mult pe afișarea de playlist-uri personalizate, în timp ce Songsurf se concentrează pe descoperirea rapidă a melodiilor noi printr-un simplu swipe.

O alta caracteristică prin care se destinge Songsurf de alte aplicații este modalitatea aparte de afișare a melodiilor, ce stârnește curiozitatea utilizatorului. În acest sens, pe pagina principală a aplicației sunt afișate videoclipuri muzicale atractive, integrate cu ajutorul platformei Youtube, dispuse într-o ordine arbitrară pentru a facilita descoperirea de noi melodii. Videoclipurile nu încep într-un mod clasic de la început, ci acestea sunt derulate la momentul refrenului sau prerefrenului pentru a prezenta piesa în cel mai dinamic și atractiv moment al său. Acest lucru oferă oportunitatea de a încânta imediat ascultătorul cu partea cea mai captivantă a cântecului, încurajând astfel explorarea mai profundă a melodiei. De asemenea, în acest mod este prezentat un volum mare de melodii într-un timp foarte scurt, acesta fiind unul din scopurile principale ale aplicației, favorizând astfel descoperirea noului conținut.

Motivație

Într-o lume în care fluxul continuu de informații poate deveni copleșitor, desco-
perirea muzicii într-o manieră rapidă și usoară a devenit din ce în ce mai dificilă.
Acest adevăr se simte mai ales în rândul noii generații, care, din cauza tehnologiei
avansate, se confruntă cu o modificare a duratei de atenție și a nivelului de răbdare.

Personal, am resimțit această schimbare, adesea pierzând interesul pentru aplicațiile
mai lente, a căror interfață nu este adaptată la viteza cu care se desfășoară viața în
prezent. Am observat cum am devenit atrasă de acele platforme care îmi pot oferi
informații cât mai rapid și eficient. Un exemplu în acest sens este aplicația TikTok,
care, printr-un simplu gest de swipe, permite accesul la o diversitate impresionantă de
conținut, fie amuzant, fie informativ, personalizat pentru preferințele fiecărui utiliza-
tor.

Fiind o persoană pasionată de muzică, adesea mi-am dorit să existe o platformă
asemănătoare, unde, printr-un simplu gest, să pot descoperi o gamă largă de noi melo-
dii. Desigur, există platforme precum YouTube sau Spotify, care îți recomandă muzică
în funcție de propriele preferințe, dar acestea nu au acea componentă interactivă și
dinamică, care face procesul de descoperire a muzicii la fel de simplu și captivant că
vizionarea clipurilor pe TikTok.

Astfel, am avut ideea de a combina elementele care au contribuit la succesul aces-
tor aplicații gigant și de a crea o platformă nouă și îmbunătățită pentru muzică. Prin
această idee s-a născut Songsurf, o aplicație muzicală care oferă utilizatorului o moda-
litate interactivă și rapidă de a descoperi noi melodii. În acest sens, Songsurf nu este
doar un instrument util, ci și o platformă care stimulează curiozitatea utilizatorului,
făcând procesul de descoperire a muzicii o activitate plăcută în sine, nu doar un mijloc
de a ajunge la un scop.

Contribuții

Ce aduce nou Songsurf este modul interactiv și dinamic de navigare printre melodii doar cu un simplu swipe. Aplicația are scopul de a transforma această căutare, ce poate deveni deseori obosită și derulantă, într-o o acțiune plăcută și distractivă. În acest sens, se stârnește curiozitatea utilizatorului prin afișarea de videoclipuri muzicale de pe platforma Youtube, acestea fiind create într-un mod artistic și intrigant de către artiștii muzicali și echipele lor de producție.

Punctul forte al aplicației este însă rapiditatea cu care poți descoperi noi melodii și modul de recomandare pe baza preferințelor fiecărui utilizator. Cu siguranță toate aceste preferințe trebuie salvate undeva, așa că utilizatorul are opțiunea de a-și salva melodiile preferate în diverse playlist-uri. Mai mult, Songsurf permite filtrarea melodiilor pe baza unui set divers de criterii, precum gen muzical, limbă și anul publicării pe platforma Youtube, permitând astfel utilizatorului să își personalizeze cu atenție playlist-urile cu melodiile favorite.

Astfel, prin integrarea tuturor acestor funcționalități, Songsurf îmbină calitățile mai multor aplicații populare de succes pentru a aduce un plus de dinamicitate și satisfacție utilizatorului în procesul de căutare a noi melodii.

Pentru dezvoltarea acestei aplicații, am îmbinat cunoștințe și idei proprii cu instrumente și tehnologii potrivite, creând astfel setul de pași ideal pentru funcționalitățile ce construiesc aplicația. Printre acești pași se enumeră:

- Crearea unui serviciu REST API cu ajutorul Flask, un framework solid și flexibil. Acesta a acționat ca server al aplicației, asigurând astfel un backend decuplat de frontend pentru o scalabilitate optimă.
- Dezvoltarea unei interfețe prietenoase și intuitive, cu ajutorul limbajului Java în mediul de lucru Android Studio.

- Implementarea unui modul de autorizare și autentificare, care are ca scop asigurarea securității și protejării datelor utilizatorilor. Autentificarea confirmă identitatea utilizatorilor, în timp ce autorizarea permite accesul acestora la resurse specifice.
- Crearea unei baze de date ce lucrează cu un volum mare de date, aceasta stocând zeci de mii de melodii. Astfel, am ales Postgres, un sistem capabil să suporte cantități mari de informație.
- Comunicarea cu baza de date a fost gestionată prin SQLAlchemy, un ORM popular pentru aplicațiile dezvoltate în Python.
- Folosirea unui API extern precum Youtube Data API. Acest instrument este extrem de util pentru extragerea de informații de pe platforma Youtube, informații ce nu sunt foarte ușor de filtrat, melodiile fiind o secțiune mică în imensitatea de date oferite de acesta aplicație.
- Utilizarea unor unelte eficiente pentru a facilita comunicarea dintre backend și frontend. Astfel, am folosit biblioteca Volley pentru gestionarea cererilor și Ngrok pentru furnizarea unui tunel sigur în vederea expunerii serverului pe rețea.
- Implementarea unui algoritm de recomandare a melodiilor cu ajutorul Scikit-learn, o librărie foarte populară în învățarea automată. Aceasta a fost folosită pentru calcularea similarității cosinus între melodiile apreciate de utilizator și restul melodiilor, asigurând astfel recomandări personalizate în cadrul aplicației.

Toate aceste tehnologii m-au ajutat să îmi consolidez și dezvolt cunoștințele acumulate în decursul celor trei ani de facultate. Lucrând la acest proiect, am îmbinat noțiuni teoretice dobândite la materii precum Programare Avansată, Python, Ingineria Programării, Programare Orientată pe Obiect și Baze de date. Astfel, prin muncă și perseverență, mi-am dezvoltat abilitățile practice în procesul creării aplicației Song-surf.

Capitolul 1

Aplicații similare

1.1 Youtube Music

Youtube Music este o aplicație muzicală de streaming dezvoltată de YouTube. Această platformă oferă o experiență de ascultare a muzicii personalizată, centrată pe descoperirea de melodii noi și pe accesul la un catalog extins de înregistrări live, remixuri și videoclipuri muzicale.

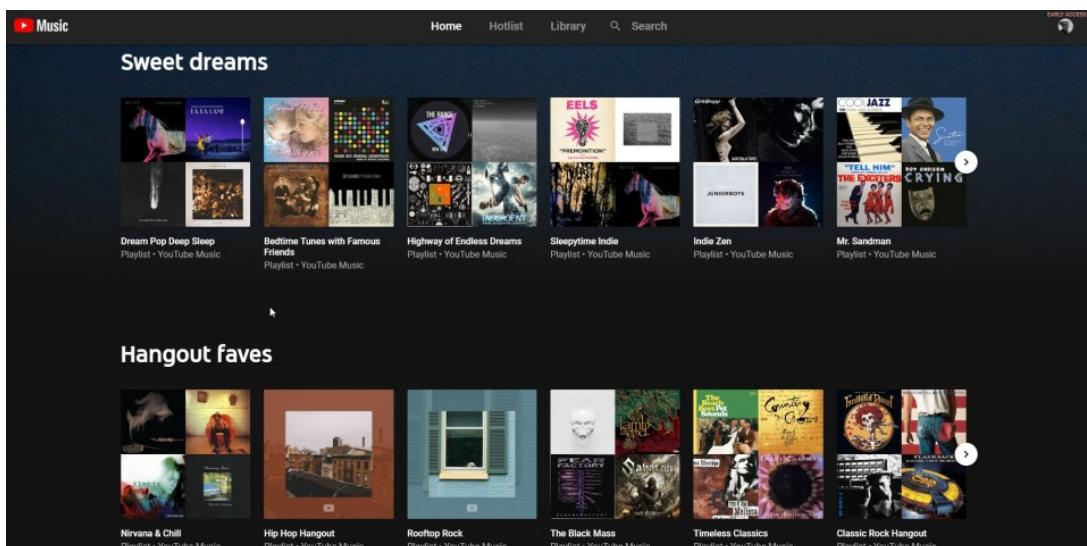


Figura 1.1: YouTube Music

<https://www.pcmag.com/reviews/youtube-music>

YouTube Music oferă utilizatorilor posibilitatea de a căuta artiști, albume sau genuri muzicale preferate, precum și de a găsi muzică nouă folosind listele de redare recomandate care sunt create pe baza istoricului de ascultare al utilizatorului. Aplicația are o funcție de căutare inteligentă care permite utilizatorilor să găsească melodii, chiar dacă nu-și amintesc cu exact titlul sau artistul, prin introducerea unui vers semnificativ

sau al unei descrierii succințe [6].

Unul dintre avantajele majore ale YouTube Music, față de alte servicii de streaming muzical este integrarea strânsă cu platforma YouTube. Astfel, utilizatorii au acces la o vastă colecție de videoclipuri muzicale, create într-un mod estetic și plăcut.

Acest avantaj am dorit să-l integrez și în aplicația mea, fiind principala asemănare dintre Songsurf și YouTube Music, astfel se oferă utilizatorului o experiență audiovizuală unică. O altă asemănare ar fi faptul că ambele aplicații au un sistem de recomandare de melodii. Totuși, diferența între cele două sisteme este aceea că YouTube Music crează playlist-uri personalizate în funcție de listele de redare curente ale utilizatorului, în timp ce Songsurf recomanda melodii în mod dinamic în funcție de ultimele melodii apreciate. De asemenea, ambele aplicații pot aplica filtre pentru videoclipurile muzicale și crea playlist-uri personalizate pe baza lor.

1.2 Spotify

Spotify este una dintre cele mai populare platforme de streaming muzical din lume, oferind acces la o gama largă de muzică și podcasturi. Lansat în 2006 de o companie suedeză, Spotify a schimbat modul în care oamenii ascultă muzică, oferindu-le posibilitatea de a-și crea liste de redare personalizate, de a descoperi muzică nouă și de a se conecta cu prietenii lor.

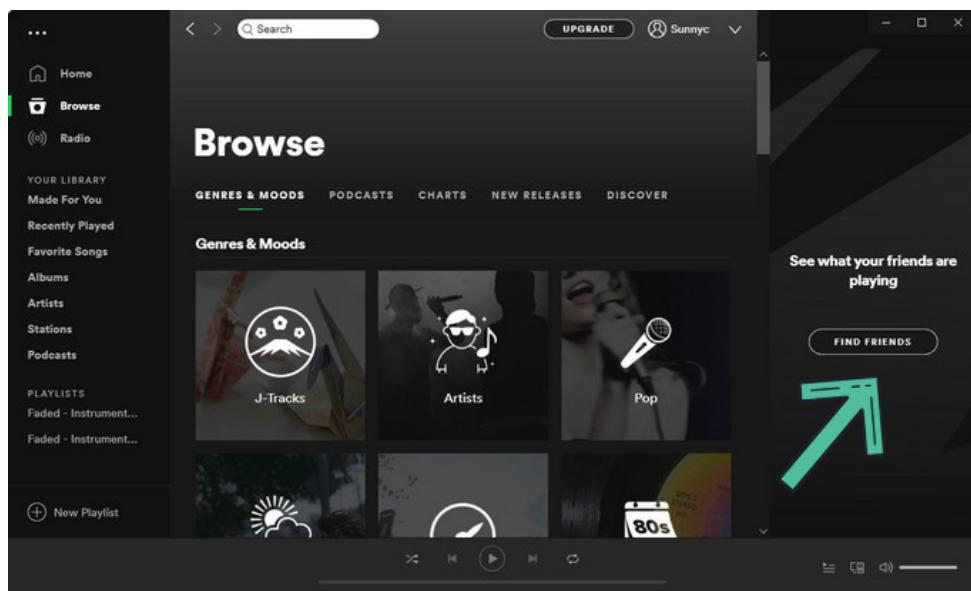


Figura 1.2: Spotify - Căutare prieteni

<https://www.sidify.com/resources/add-friends-on-spotify.html>

Integrarea cu alte servicii și platforme este o caracteristică inovatoare concepută de platforma Spotify. De exemplu, pot fi integrate aplicații de social media precum Facebook, oferindu-se utilizatorilor posibilitatea de a împărtăși ceea ce ascultă cu prietenii lor.

Utilizatorii Spotify pot căuta muzică după artist, album, gen sau după listele de redare ale altor utilizatori. Această aplicație facilitează găsirea de noi melodii prin recomandări personalizate bazate pe istoricul de ascultare al utilizatorului [8].

Aseamanările dintre Songsurf și Spotify sunt evidente, ambele aplicații reușind să ofere sugestii pe baza melodiilor apreciate de utilizator, sau să creeze playlist-uri personalizate pe baza anumitor filtre. Principala diferență însă este modalitatea de afișare a melodiilor, Songsurf punând la dispoziție videoclipuri muzicale atractive, într-o ordine aleatoare în punctul cheie al refrenului sau prerefrenului unei melodii, pe când Spotify afișează doar sonor melodii într-un mod clasic de la început până la final.

Capitolul 2

Arhitectura aplicației

Arhitectura unei aplicații constă în modul în care aceasta este structurată și proiectată pentru a asigura îndeplinirea cerințelor de funcționalitate, performanță, securitate și scalabilitate. Ea descrie modul în care componentele aplicației interacționează între ele și cu mediul extern. De asemenea, o arhitectură solidă facilitează dezvoltarea și întreținerea sistemului software pentru a răspunde la noi cerințe sau schimbări tehnologice.

2.1 Arhitectura Client-Server

În arhitectura client-server, componentele aplicației sunt împărțite în două părți principale: un server care furnizează servicii și un client care accesează și utilizează acele servicii. Această organizare facilitează arhitectura unei aplicații în diferite măduuri:

- Sarcini autonome: serverul se ocupă de gestionarea datelor, procesarea cererilor și implementarea logicii aplicației, pe când clientul se ocupă de comunicarea cu utilizatorii, colectarea cererilor, transmiterea acestora la server și afișarea rezultatelor. Astfel, aplicația devine organizată și ușor de întreținut prin intermediul acestei împărțiri clare a responsabilităților.
- Scalabilitate: aplicația poate fi extinsă, astfel pot fi create mai multe servere ce își împart sarcinile. De asemenea, se pot face modificări la server fară a afecta clientii.
- Consistență: mai mulți clienti pot folosi același server, asigurându-se astfel că

toti clientii primesc aceleasi servicii. În plus, se pot crea clienti pentru diverse dispozitive, toti fiind gestionati de același server (de exemplu un client web și unul mobile).

- Securitate: riscul de expunere a datelor sensibile este redus prin păstrarea acestora pe server, fară a fi trimise și către client.

Având în vedere multitudinea de avantaje oferite, aplicația Songsurf a fost construită pe baza arhitecturii client-server, menținând astfel o separație clară între server și client.

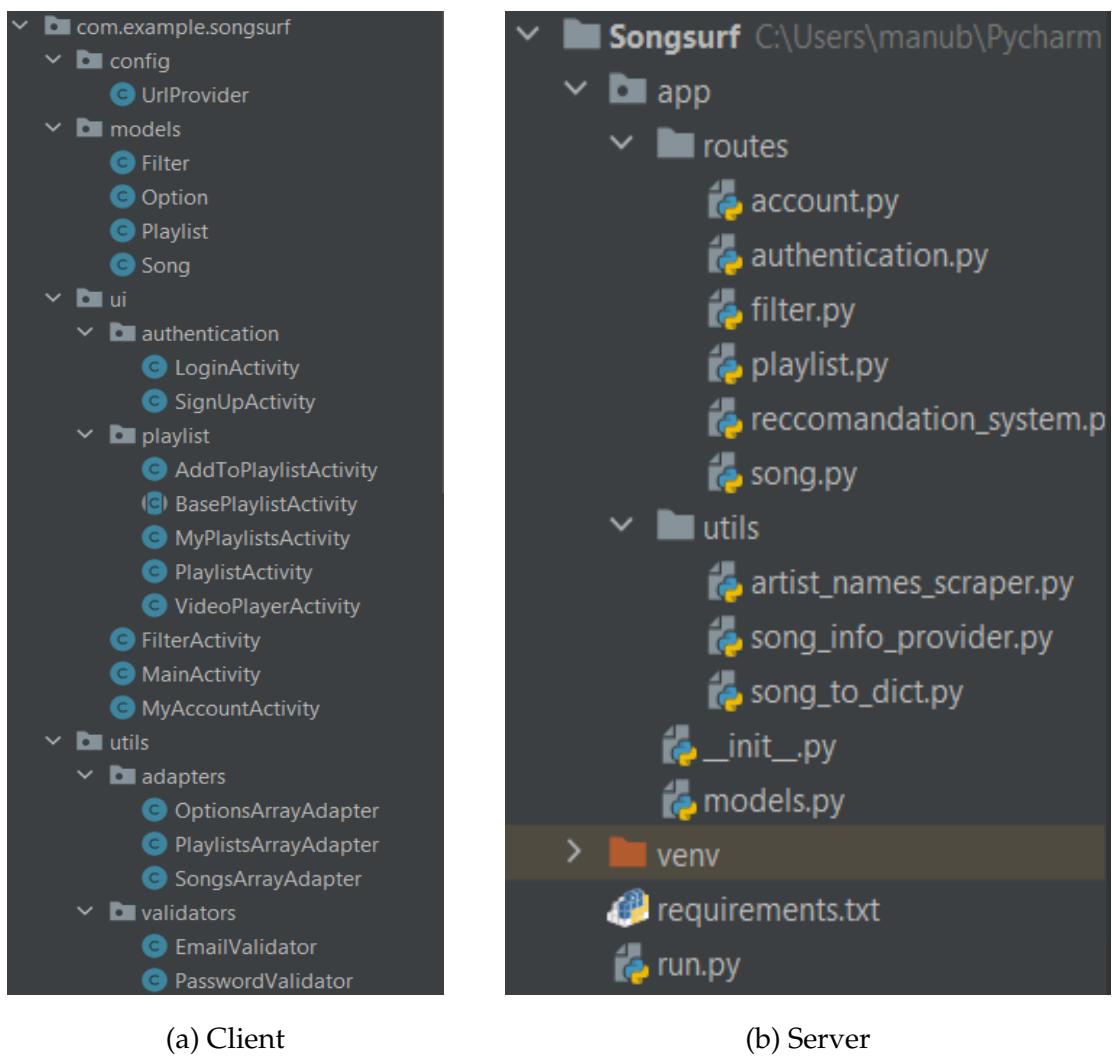


Figura 2.1: Structura Client-Server

În figura 2.1 se poate observa că structura celor două componente se bazează pe o arhitectură modulară, în care funcționalitățile aplicației sunt împărțite în module independente, fiecare având responsabilități diferite. Acest mod de organizare are diverse

avantaje precum reutilizarea și întreținerea codului, oferind o flexibilitate și scalabilitate crescută [7].

2.1.1 Client

Clientul aplicației a fost construit utilizând Android Studio, fiind una dintre cele mai populare platforme în dezvoltarea aplicațiilor mobile. Responsabilitatea clientului este aceea de a interacționa cu utilizatorul, oferind o interfață atractivă și ușor de utilizat. În acest sens, au fost implementate diverse clase în limbajul de programare Java pentru a gestiona solicitările clientului și a le transmite către server. Aceste clase, cunoscute sub numele de "activități" lucrează în tandem cu fișierele XML, utilizând obiecte de tip View, pentru a crea o interfață vizuală plăcută cu care utilizatorul poate comunica.

2.1.2 Server

Server-ul aplicației a fost dezvoltat cu ajutorul limbajului de programare Python și a framework-ului Flask. Rolul acestuia este de a gestiona datele în funcție de solicitările primite de la client și de a trimite răspunsurile corespunzătoare înapoi către acesta. Tot prin intermediul serverului sunt implementați algoritmi ce țin de logica aplicației, cum ar fi sistemul de recomandare a melodii sau procesul de populare a bazei de date prin intermediul unui API extern.

2.2 Modelul arhitectural MVC (Model-View-Controller)

Model-View-Controller (MVC) este un design pattern foarte popular în dezvoltarea aplicațiilor web. Acesta împarte logica unei aplicații în trei componente, fiecare având roluri și responsabilități diferite [11].

Modelul este nucleul aplicației ce gestionează datele și logica asociată acestora. Acesta accesează direct baza de date și furnizează informațiile necesare folosite de controller pentru a efectua anumite sarcini și de view pentru a afișa rezultatele.

Controller-ul este intermediarul dintre model și view. Acesta preia cererea și instruiește modelul să execute anumite acțiuni, de exemplu să comunice cu baza de date.

View-ul este responsabil pentru prezentarea informațiilor utilizatorului. Acesta ia datele procesate de model, cu ajutorul controller-ului, și le afișează utilizatorului într-un format atractiv.

Comunicarea între cele 3 componente este exemplificată în figura următoare:

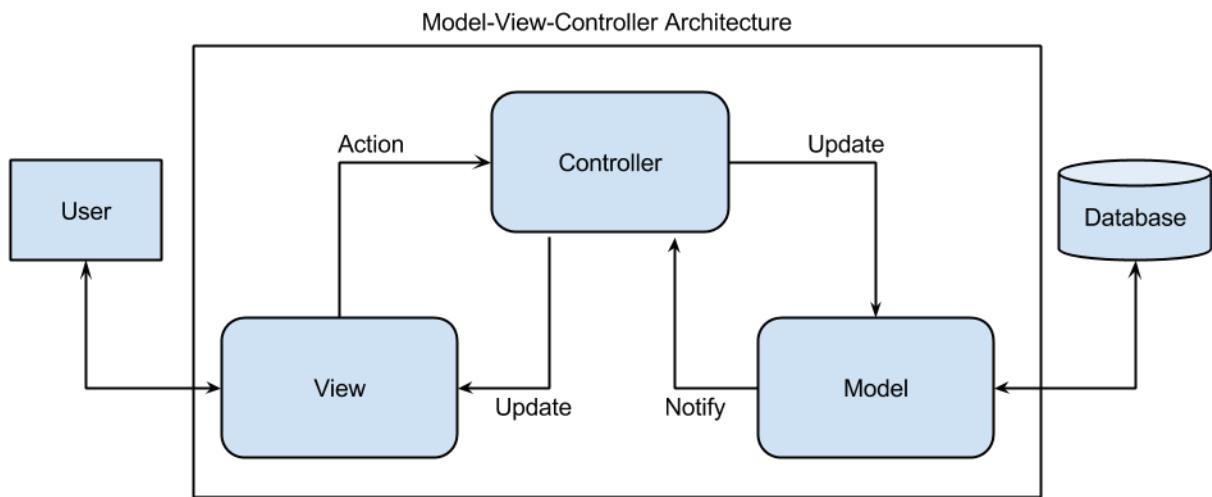


Figura 2.2: Modelul arhitectural MVC

În acest mod, design pattern-ul Model-View-Controller aduce deverse beneficii în dezvoltarea aplicațiilor precum:

- Claritate: codul este mai ușor de înțeles având o structură clară.
- Reutilizare: componentele Model, View și Controller pot fi utilizate în mod repetat, ceea ce îmbunătățește productivitatea.
- Scalabilitate: facilitează dezvoltarea și testarea paralelă a celor trei componente, astfel îmbunătățind procesul de scalare în proiectele mari.
- Flexibilitate: pot fi făcute modificări asupra unei componente fară a fi afectate și celelalte.

În aplicația Songsurf, componentele MVC sunt împărțite astfel: la nivelul serverului aplicației (figura 2.1, secțiunea a) sunt definite modelele în fișierul "models.py", iar funcționalitatea de controller este asigurată de pachetul "routes". View-urile se regăsesc în partea de client a aplicației (figura 2.1, secțiunea b), fiind implementate prin intermediul activităților din modulul "ui" și a layout-urilor de tip XML asociate.

2.3 Scenarii de utilizate

O etapă esențială în dezvoltarea arhitecturii unei aplicații o reprezintă definirea interacțiunii utilizatorului cu sistemul. Pentru aplicația Songsurf, vom ilustra aceste scenarii prin diagrama use-case prezentă în figura 2.3 de mai jos.

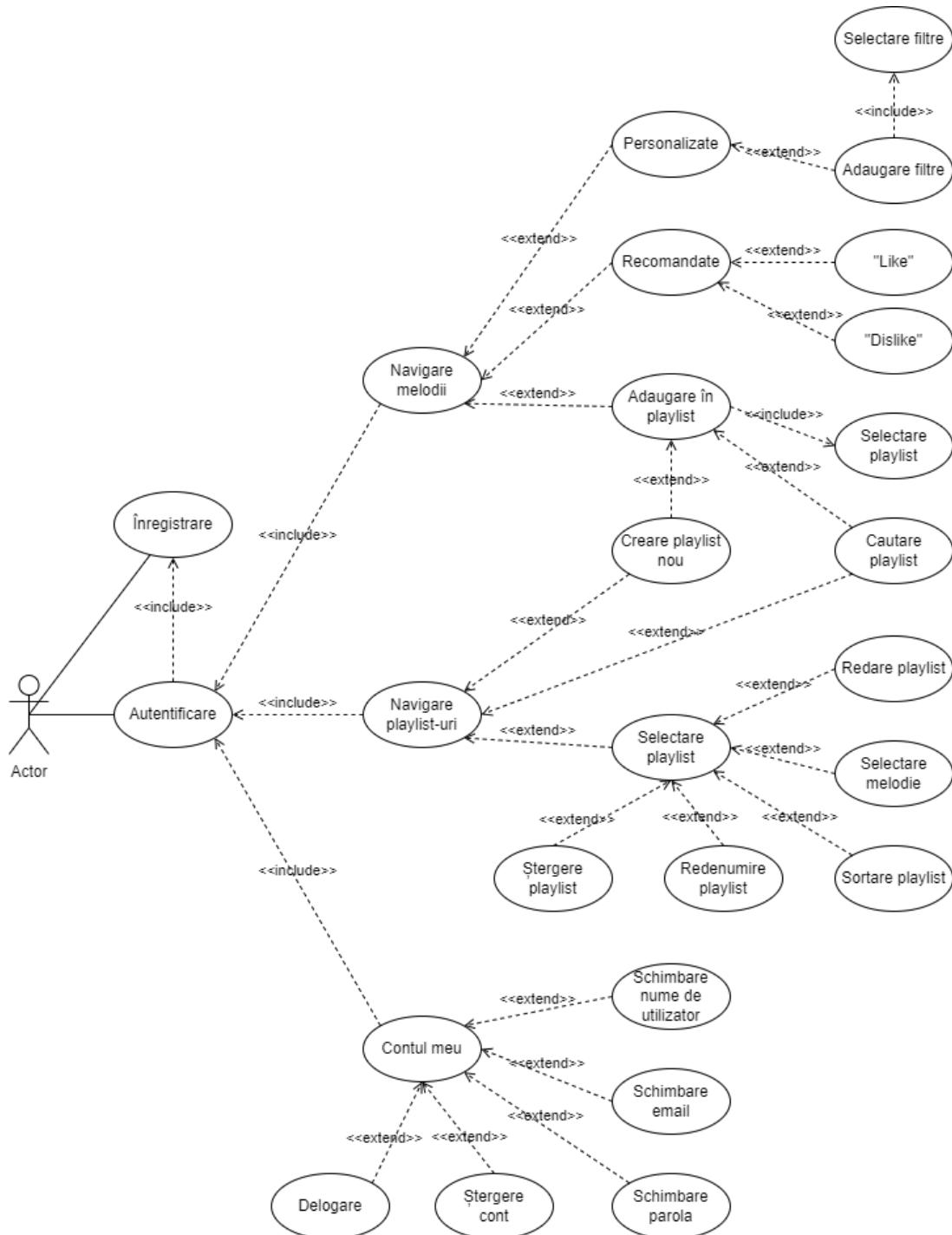


Figura 2.3: Diagrama Use-Case

Un utilizator ce nu este autentificat are două opțiuni, cea de înregistrare în care își crează un cont nou și cea de logare în care își introduce datele corespunzătoare pentru

a intra în cont.

După autentificare acesta va intra automat în secțiunea de navigare printre melodii numită precum aplicația, "Songsurf". În aceasta secțiune, utilizatorul are posibilitatea de a selecta dacă vrea să primească melodii personalizate pe baza unui set de filtre alese de el, sau dacă vrea să obțină videoclipuri muzicale recomandate în funcție de "rating"-ul oferit acestora. Tot aici acesta poate alege obținerea de a adăuga o melodie în intr-un playlist.

Următoarea opțiune pe care utilizatorul o poate alege din meniul de navigare este afișarea playlist-urilor sale, unde are de asemenea un set de operații din care poate alege. Printre acestea se enumera: căutarea, crearea sau afișarea unui playlist. Dacă utilizatorul alege opțiunea de afișare, atunci vor apărea melodiile din el împreună cu o serie de opțiuni de modificare, sortare sau ștergere a playlist-ului sau conținutului acestuia. Desigur, acesta poate selecta o melodie care va fi ulterior redată prin intermediul videoplayer-ului oferit de platforma Youtube.

Ultima secțiune este cea de administrare a contului, numită "My Account", unde se pot modifica datele de utilizator, șterge contul sau se poate ieși din cont.

Capitolul 3

Detalii de implementare

Acest capitol își propune să scoată în evidență modul în care a fost dezvoltată aplicația, evidențiind tehnologiile folosite, deciziile de proiectare și strategiile alese pentru realizarea acesteia.

Proiectul se bazează pe o arhitectură client-server, cuprinzând un server Flask, care servește ca „backend” al aplicației, responsabil pentru gestionarea datelor, și un client Android, „frontend”-ul aplicației, care creează interfață grafică pentru utilizator. Serverul furnizează un serviciu REST API clientului, oferind acces la date și posibilitatea de a efectua diverse solicitări conform interacțiunilor utilizatorului.

În acest sens, vor fi prezentate componentele principale ale aplicației, modulele de frontend și de backend, principiile pe baza cărora au fost construite, tehnologiile utilizate și modul prin care acestea interacționează între ele. În cele din urmă, vor fi expuse obstacolele care au apărut în timpul dezvoltării proiectului și modurile prin care acestea au fost depășite.

3.1 Backend-ul aplicației

3.1.1 Tehnologii utilizate

- **Python:** este un limbaj de programare de nivel înalt, orientat pe obiecte, cu o sintaxă intuitivă și usoară, ceea ce îl face unul dintre cele mai populare limbaje de programare. În cadrul aplicației Songsurf, am ales Python pentru dezvoltarea backend-ului datorită sintaxei sale ușor de înțeles și a gamei largi de biblioteci și framework-uri disponibile, precum Flask și SQLAlchemy. De asemenea, acest limbaj este deseori ales pentru proiectarea algoritmilor de învățare auto-

mata grație simplității și flexibilității sale, astfel am integrat librăria Scikit-learn.

- **Flask:** reprezintă un micro-framework web care nu necesită unelte sau librării adiționale, oferind dezvoltatorilor libertatea de a alege instrumentele potrivite pentru fiecare proiect în parte. Am folosit Flask în crearea aplicației deoarece a permis dezvoltarea unui server web performant și flexibil, ce raspunde cu succes cererilor venite de la client. În plus, un alt avantaj al utilizării Flask este simplitatea și curățenia codului, făcând dezvoltarea aplicațiilor mult mai intuitivă și mai puțin predispusă la erori.
- **PostgreSQL:** este un sistem de gestionare a bazelor de date obiect-relaționale puternic și scalabil. Am ales PostgreSQL pentru stocarea datelor aplicației Songsurf deoarece este stabil, gratuit și suportă lucrul cu un volum mare de date [9].
- **SQLAlchemy:** definește o tehnologie foarte utilă din Python care oferă un set de instrumente SQL și un sistem de mapare obiect-relațională (ORM). ORM-ul permite dezvoltatorilor să creeze clase Python care corespund tabelelor din baza de date, precum și instanțe ale acestor clase care corespund liniilor din tabel. Operațiunile de bază de date precum interogările, inserările, actualizările și ștergerile pot fi efectuate direct în Python cu ajutorul acestei biblioteci. Acest lucru reduce probabilitatea de erori de sintaxă SQL și face codul mai ușor de înțeles și de întreținut [3].
- **Flask-RESTful:** este o extensie Flask ce ajută la crearea de API-uri REST. Cu ajutorul acesteia funcționalitățile aplicației sunt expuse că servicii web, ceea ce ne permite să tratăm cererile de la frontend într-un mod optim sau să interacționăm cu alte aplicații dacă este nevoie /citebib10.
- **Flask-JWT Extended:** reprezintă o extensie ce oferă suport pentru utilizarea tokenurilor web de tip JSON (JSON Web Tokens) în aplicațiile Flask. Am utilizat Flask-JWT Extended în proiect pentru a implementa autentificarea și autorizarea utilizatorilor. Acest lucru ne permite să asigurăm că doar utilizatorii care au fost autenticați pot accesa anumite funcționalități ale aplicației.
- **Scikit-learn:** este o bibliotecă din Python care oferă implementări eficiente pentru o varietate de algoritmi de învățare automată și de analiză a datelor. Această bibliotecă oferă tehnici diverse pentru analiza textului și calculul similarităților,

care pot fi foarte utile în crearea unui sistem de recomandare pe bază de conținut. Un astfel de sistem se regăsește și în aplicația Songsurf, cu ajutorul căruia se recomandă melodii utilizatorilor pe baza asemănărilor dintre melodiile apreciate de acesta [12].

- **Ngrok:** reprezintă un instrument care facilitează expunerea unui server local pe internet prin furnizarea de tuneluri securizate. Un server ce rulează local pe mașina dezvoltatorului nu poate interacționa cu o aplicație Android care rulează pe un dispozitiv mobil dacă nu se află în aceeași rețea locală. Ngrok rezolvă această problemă prin crearea unui tunel securizat care conectează serverul la internet. Astfel se generează o adresă URL publică pe care clientul Android o poate folosi pentru a comunica cu backend-ul.
- **YouTube Data API:** este un set de servicii web furnizate de Google care permite interacțiunea cu YouTube în diverse moduri precum încărcarea, căutarea și afișarea de conținut video. În aplicația noastră, am folosit acest API pentru a căuta și a reda videoclipuri muzicale direct de pe platforma YouTube. Acest lucru ne permite să construim o bază de date bogată și diversă de melodii.

3.1.2 Proiectarea bazei de date

Baza de date este un element fundamental într-o aplicație fiind responsabilă pentru stocarea, organizarea și gestionarea datelor într-un mod eficient. Aceasta asigură o multitudine de beneficii precum accesul rapid la informații, scalabilitatea aplicației și securitatea crescută a datelor.

Songsurf folosește o bază de date formată din 6 tabele în care sunt stocate informațiile necesare pentru realizarea tuturor funcționalităților aplicației. În procesul de dezvoltare a aplicației s-a acordat o atenție deosebită modului în care sunt organizate și gestionate datele. În acest sens, tabelele au fost proiectate într-un mod normalizat pentru a evita redundanță și pentru a utiliza informațiile într-un mod cât mai eficient. În cele se urmează va fi explicitat rolul fiecărei tabele în parte, iar mai apoi modul în care acestea au fost create și asociările dintre ele.

- **users:** conține datele despre utilizatori. Colonele includ următoarele atrbute: id (care este cheia primară), nume de utilizator, email și parola. Acest tabel este esențial pentru gestionarea autentificării și autorizării utilizatorilor în aplicație.

- **songs**: stocază informații despre melodii. Contine colone pentru id (cheia primară), id-ul corespunzător melodiei de pe youtube, titlul melodiei, gen, an, limbă și numele artistului. Aceasta este conectat cu tabelul songs_playlists, ceea ce permite fiecărei melodii să fie asociată cu un anumit playlist.
- **playlists**: reține playlist-urile create de utilizatori. Include colone pentru id (cheia primară), nume și id-ul utilizatorului care detine playlist-ul (cheie străină care face referire la atributul id din tabelul users). Acest tabel este conectat cu tabelul users, ceea ce permite fiecărui utilizator să creeze și să dețină mai multe playlist-uri.
- **songs_playlists**: este un tabel de asociere între songs și playlists. Ea conține id-ul unei melodii, id-ul unui playlist (acestea fiind chei străine ce fac referire la atributul id din tabelele songs și playlists) și momentul în care a fost adăugată o melodie într-un playlist. Acest tabel este conectat cu ambele tabele songs și playlists, permitând astfel asocierea melodiiilor cu playlist-urile în care au fost adăugate.
- **user_song_preferences**: are rolul de a stoca preferințele muzicale ale utilizatorilor. Contine câmpuri pentru id-ul unic al preferinței, id-ul utilizatorului, id-ul melodiei apreciate, o valoare booleană care indică dacă melodia a fost apreciată de utilizator sau nu și momentul în care melodia a fost apreciată. Acest tabel este conectat atât cu tabelul users, cât și cu tabelul songs, permitând astfel salvarea preferințelor utilizatorilor, ceea ce ajuta la crearea unui sistem de recomandare personalizat.
- **recommended_songs**: este folosit pentru a reține recomandările de melodi pentru fiecare utilizator. Contine câmpuri pentru id-ul unic al recomandării, id-ul utilizatorului și id-ul melodiei recomandate. Acesta este conectat cu tabelele users și songs și are rolul de a salva melodiile care au fost recomandate către un utilizator pentru a nu le afișa din nou.

Toate aceste tabele au fost implementate în serverul aplicației prin crearea unor modele ale bazei de date în ORM cu ajutorul tehnologiei SQLAlchemy. ORM (Object-Relational Mapping) transformă codul Python în comenzi SQL, pentru a interacționa cu baza de date [2].

```

class Playlist(db.Model):
    __tablename__ = "playlists"
    id = db.Column(db.Integer, primary_key=True)
    name = db.Column(db.String(80), nullable=False)
    user_id = db.Column(db.Integer, db.ForeignKey('users.id'), nullable=False)

    user = db.relationship("User", backref="playlists", lazy=True)
    songs = db.relationship('SongsPlaylists', backref='parent_playlist', lazy=True)

    def __repr__(self):
        return f"{self.name} {self.id} {self.user_id}"

```

Figura 3.1: Modelul tabelului "playlists"

În Figura 3.1 de mai sus este prezentat modul în care a fost conceput tabelul "playlists". Similar au fost construite și restul modelelor explicate anterior. Se poate observa că fiecare clasă reprezintă un tabel din baza de date, fiecare instanță a unei clase reprezintă un rând în acea tabelă, iar atributele claselor sunt coloanele din tabele.

Tot prin intermediul acestor clase sunt stabilite și relațiile dintre tabele, care asigură utilizarea eficientă și corectă a datelor. Aceste relații sunt de mai multe tipuri:

- 1. Relații "One-To-Many":** acestea se regăsesc cel mai frecvent în modelele noastre și sunt esențiale pentru a evita redundanța datelor. De exemplu, există o relație "One-To-Many" între User și Playlist, reprezentând faptul că un utilizator poate avea mai multe playlist-uri, în timp ce fiecare playlist aparține unui singur utilizator.
- 2. Relații "Many-To-Many":** modelele aplicației demonstrează utilizarea corectă a relațiilor "Many-To-Many", care sunt reprezentate prin tabele de asociere precum "songs_playlists". De exemplu, relația dintre tabelele Song și Playlist este o relație "Many-To-Many", deoarece o melodie poate apărea în mai multe playlist-uri și un playlist poate conține mai multe melodii.

Această relație este gestionată prin tabela de asociere SongsPlaylists, care ajută la reprezentarea melodiilor dintr-un playlist. Astfel, atunci când se adaugă o melodie într-un playlist, se creează un nou rând în tabelul SongsPlaylists cu id-ul melodiei și id-ul playlist-ului. În plus, fiind o tabelă separată, aceasta permite stocarea unor informații suplimentare despre acea relație, cum ar fi atributul "timestamp", care indică momentul în care a fost creată.

3. Utilizarea cheilor străine: Fiecare tabelă care face referire la o altă tabelă o face prin utilizarea cheilor străine. De exemplu, atributul "user_id" din clasa "Playlists" (Figura 3.1) este o cheie străină care face referire la atributul "id" din tabelul "users". Astfel, se poate accesa utilizatorul care a creat playlist-ul și melodiile din playlist direct dintr-o instanță a modelului Playlist.

În diagrama următoare (Figura 3.2), se poate observa utilizarea tuturor relațiilor dintre tabele explicate anterior ce au condus la o dezvoltare eficientă și optimă a bazei de date.

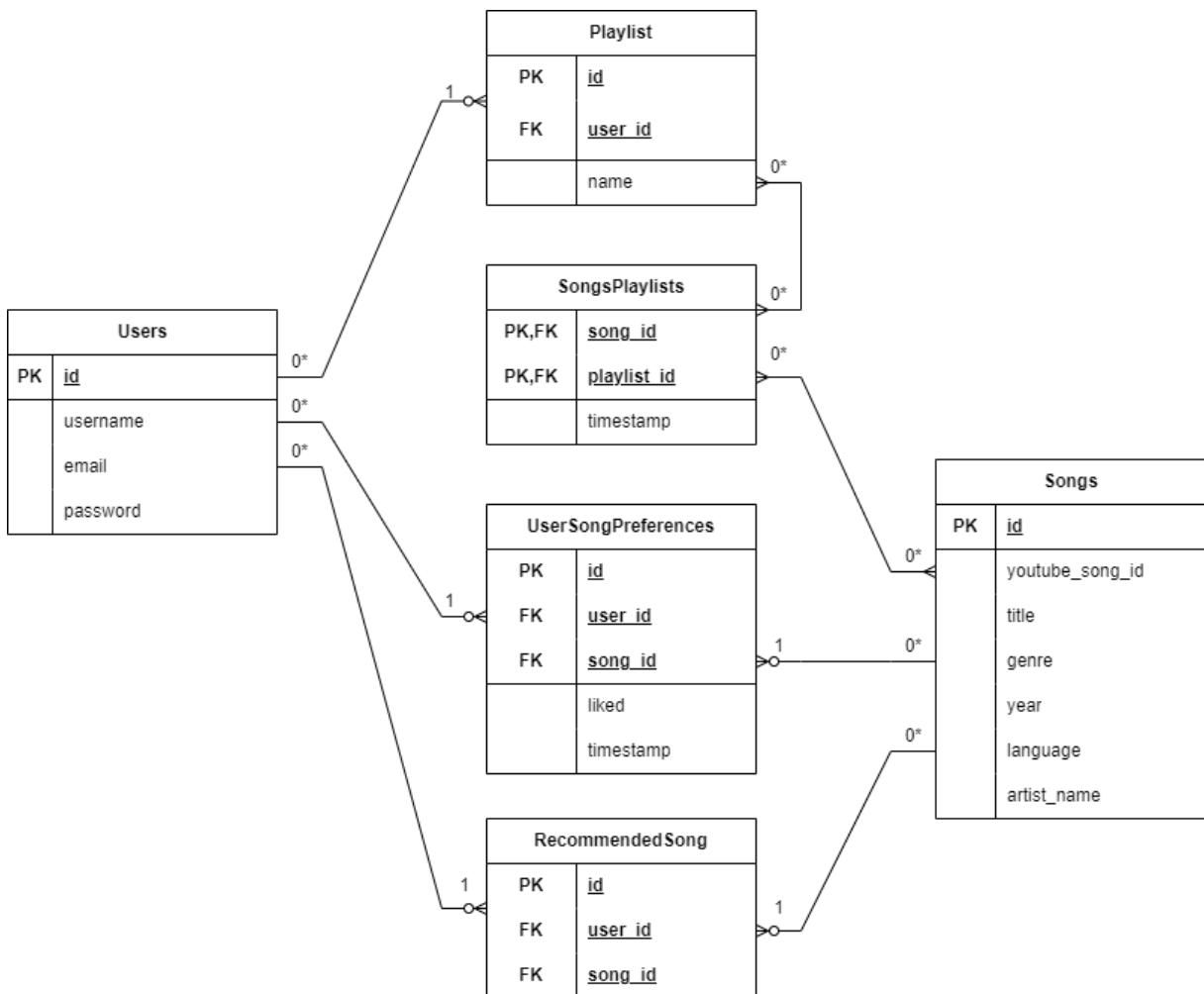


Figura 3.2: Diagrama Bazei de Date

3.1.3 Popularea bazei de date

Popularea bazei de date a reprezentat unul dintre pașii esențiali în dezvoltarea aplicației Songsurf, caracteristica principală a acesteia fiind navigarea printr-o multitudine de videoclipuri muzicale cât mai diverse.

Astfel am utilizat serviciul web Youtube Data API oferit de Google care pune la dispozitie o serie de "endpoint"-uri ce permit programatorilor să interacționeze cu datele de pe platforma YouTube. Cererile pot fi făcute pe baza unei chei de acces securizată oferită de Google pentru fiecare utilizator care își face cont pe platforma de Cloud destinată dezvoltatorilor.

Desi acest serviciu oferă un volum imens de informații, el are și anumite limitări de care a trebuit să ţin cont. Una din limitările principale este legată de cota impusă de YouTube pentru utilizarea API-ului său. Mai exact, există un număr limitat de cereri pe care o aplicație le poate face într-o anumită perioadă de timp. Operația de căutare este cea folosită în dezvoltarea bazei de date, prin intermediul căreia sunt căutate melodii și canale de YouTube ale artiștilor. O operație de căutare "costă" 100 de unități, iar într-o căutare poți primi maxim 50 de rezultate. Limita zilnică este 10.000 de unități per utilizator, deci pot fi făcute maxim 100 de cereri pe zi. Astfel, cantitatea maximă de melodii care pot fi extrase cu ajutorul acestui API este de 5.000 de videoclipuri pe zi.

Desi sună promițător, acesta modalitate de căutare directă este ineficientă dacă nu are o logică bine definită în spate, deoarece YouTube oferă o cantitate enormă de videoclipuri de toate tipurile și dimensiunile. Când am început să învăț despre utilizarea acestui serviciu, am dat peste o serie de obstacole.

Primul dintre ele este acela că ruta destinată pentru căutare nu oferă foarte multe detalii despre videoclipurile afisate. Printre detaliile adiționale care ar putea fi incluse se enumeră: numărul de vizualizări pentru a decide popularitatea videoclipurilor, lungimea acestora pentru a selectarea videoclipurilor ce se încadrează în intervalul de 2-5 minute pentru a filtra melodii. Aceste informații sunt oferite însă de "endpoint"-ul numit "videos" care oferă informații despre un anumit videoclip dat. Așadar, pentru a afla aceste informații adiționale destul de importante în filtrarea videoclipurilor, trebuie făcută o cerere separată pentru fiecare videoclip în parte, această operație fiind costisitoare conform limitării de care am discutat anterior.

Astfel a trebuit să aleg dintre două scenarii: popularea bazei de date cu un număr considerabil mai mare de melodii pentru a oferi o diversitate de melodii, filtre dar și recomandări mai precise, și încărcarea a unui numai mai mic de date dar cu câteva caracteristici în plus ce pot aduce valoare aplicației. Am ales primul scenariu din cauza timpului limitat, deoarece am considerat mai importantă cantitatea și diversitatea melodii din baza de date și mai utilă această abordare pe termen scurt. Pe termen lung însă, scenariul al doilea poate fi implementat pentru a îmbunătăți aplicația.

Un alt obstacol este acela că parametrii specifici rutei de căutare, deși sunt diversi și într-un număr mare, nu oferă acuratețe atunci când sunt combinați pentru a filtra conținutul de pe YouTube. Mai exact, pentru implementarea filtrelor din aplicația Songsurf aveam nevoie de anumite informații precum genul muzical, limbă sau anul publicării videoclipului respectiv.

Deși există anumiți parametri care ar trebui să faciliteze această filtrare de date, atunci când sunt combinați nu oferă răspunsul așteptat. Că să dau un exemplu mai clar, ruta de căutare oferită de YouTube API poate avea ca parametru "topicId" care se ocupă cu identificarea unui set prestabilit de genuri. Atunci când acesta este combinat cu parametrul "relevanceLanguage" responsabil să identifice limba relevantă a unui videoclip, rezultatele oferite nu respectau criteriile de filtrare. Astfel, a trebuit să caut o metodă diferită pentru căutarea melodiiilor, astfel încât să pot beneficia de un set minimal de criterii de filtrare.

```
def get_song_info(self, channel_id, artist_name):
    url = f"https://www.googleapis.com/youtube/v3/search?part={self.snippet}&maxResults={self.max_results}" \
          f"&type={self.type}&videoCategoryId={self.video_category}&videoDefinition={self.video_definition}" \
          f"&videoDuration={self.video_duration}&key={self.api_key}&order={self.order}&channelId={channel_id}"
    try:
        response = requests.get(url)
        if response.status_code == 200:
            data = response.json()

            for item in data["items"]:
                song_id = item["id"]["videoId"]
                title = unescape(item["snippet"]["title"])
                year = item["snippet"]["publishedAt"][:4]
                language = "romanian"
                genre = "pop"

                song = Song.query.filter_by(youtube_song_id=song_id).first()
                if song:
                    continue
                if artist_name.lower() not in title.lower():
                    continue

                song = Song(youtube_song_id=song_id, title=title, genre=genre, year=year, language=language,
                            artist_name=artist_name)
                db.session.add(song)
                db.session.commit()
```

Figura 3.3: Metoda "get_song_info"

Soluția acestui obstacol a fost implementată în 3 pași. Primul a fost obținerea unei liste de artiști muzicali consacrați pentru fiecare gen și limbă în parte, iar acest lucru a fost posibil prin dezvoltarea unei unelte de "scrapping" pentru un site web care oferă aceste date. Astfel, am găsit platforma "Chosic" care detine câte o rută pentru aproape

fiecare combinație de gen și limbă existente, fiecare url oferind o listă cu top 200 artiști populari ce se încadrează în parametrii de căutare [15].

Cu ajutorul acestei liste, trecem la pasul doi, acela în care am construit o metodă ce efectuează o cerere GET către YouTube Data API pentru a obține informații despre canalul asociat cu un nume de artist. Astfel, am obținut top 10 canale care se găsesc atunci când este introdus numele artistului în bara de căutare, iar dintre aceste canale l-am selectat pe cel care are procentul de apariție cel puțin 50% pentru a ne asigura că este întradevar canalul de YouTube al artistului căutat.

Al treilea pas a fost acela de a căuta în ordinea popularității top 50 de melodii pentru fiecare canal de YouTube găsit anterior. În acest sens am creat metoda "get_song_info" prezentă în Figura 3.3 care efectuează o cerere GET către YouTube API pentru a obține piesele muzicale a unui anumit canal. Apoi, aceasta extrage informațiile relevante pentru fiecare piesă: id-ul piesei, titlul, anul publicării, limba și genul și apoi le adaugă în baza de date.

Acestă metodă nu este eficientă 100% deoarece se pot găsi canale care să conțină numele artistului dar să nu fie cele oficiale, sau se pot adăuga videoclipuri care nu reprezintă tocmai melodii, artiștii punând la dispoziție și alt tip de conținut pe canalele lor. Astfel, am îndepărtat manual videoclipurile din baza de date care nu reprezentau informațiile dorite. Însă aceste scenarii sunt scăzute și le-am diminuat cât se poate de mult în proiectarea algoritmilor, soluția prezentată fiind cea mai bună opțiune pentru a face rost de datele necesare.

Prin implementarea algoritmilor discuți mai sus, am reușit să creez o bază de date impresionantă de aproape 30.000 de melodii care îmbină artiști din diverse culturi și genuri muzicale. Datorită numărului mare de conținut pe care aceasta îl conține, navigarea printre melodiile din aplicația Songsurf este una diversă și incitantă, fiind îndeplinit astfel unul din scopurile principale ale platformei.

3.1.4 Procesarea cererilor de la client și returnarea răspunsurilor corecte de la server

În cele ce urmează voi prezenta procesul prin care clientul (aplicația android) trimite o cerere către server și modul în care acesta îi oferă răspunsul potrivit înapoi. Astfel, voi ilustra pas cu pas procedeul de adăugare a unei melodii într-un playlist.

```

private void addSongToPlaylist(int playlistId) {
    String url = UrlProvider.PLAYLIST_URL + "/" + playlistId + "/add";
    Map<String, String> params = new HashMap<>();
    params.put("song_id", String.valueOf(songId));

    JsonObjectRequest jsonObjectRequest = new JsonObjectRequest(Request.Method.POST, url, new JSONObject(params),
        response -> {
            Toast.makeText(getApplicationContext(), text: "Song added to playlist successfully", Toast.LENGTH_LONG).show();
            finish();
        },
        error -> {
            // handle error
        }
    ) {
        @Override
        public Map<String, String> getHeaders() {
            Map<String, String> headers = new HashMap<>();
            SharedPreferences sharedpreferences = getSharedPreferences( name: "com.example.songsurf", Context.MODE_PRIVATE);
            String jwt_token = sharedpreferences.getString( key: "accessToken", defaultValue: null);
            headers.put("Authorization", "Bearer " + jwt_token);
            return headers;
        }
    };
    RequestQueue requestQueue = Volley.newRequestQueue( context: this);
    requestQueue.add(jsonObjectRequest);
}

```

Figura 3.4: Metoda “addSongToPlaylist”

1. Inițierea cererii de către client: Acest proces începe în momentul în care utilizatorul decide să adauge o melodie într-un playlist. Atunci, metoda “addSongToPlaylist” din Figura 3.4 este apelată din aplicația Android. Aceasta construiește un obiect de tip JSON care conține id-ul melodiei și inițiază o cerere POST către server, la un URL specific care include și id-ul playlist-ului. Această cerere este adăugată în coada de cereri a bibliotecii Volley, care se ocupă cu gestionarea cererilor de rețea în Android.

2. Procesarea cererii de către server: Pe partea de server a fost definit un endpoint special care ascultă cererile POST către URL-ul la care clientul trimite cererea. Acest endpoint este definit în clasa “AddToPlaylist” din Figura 3.5, care este o resursă Flask-RESTful ce se ocupa de tratarea cererilelor de la frontend într-un mod optim.

Când o cerere ajunge la acest endpoint, metoda POST a acestei clase este apelată. Aceasta preia id-ul melodiei din cererea primită și verifică dacă acest ID există și dacă este valid. De asemenea, verifică dacă playlist-ul există și dacă melodia nu a fost deja adăugată în acesta. Dacă toate condițiile necesare sunt îndeplinite, creează o nouă instanță a clasei de asociere “SongsPlaylists” ce gestionează melodiiile adăugate într-un playlist, că mai apoi să o salveze în baza de date.

```

class AddToPlaylist(Resource):
    @jwt_required()
    def post(self, playlist_id):
        song_id = request.json.get('song_id')

        if not song_id:
            return {"error": "Song id is required"}, 400

        playlist = Playlist.query.get(playlist_id)
        song = Song.query.get(song_id)

        if not playlist or not song:
            return {"error": "Song or Playlist not found"}, 404

        if any(sp.song_id == song.id for sp in playlist.songs_in_playlists):
            return {"error": "Song already in this playlist"}, 400

        association = SongsPlaylists(song_id=song.id, playlist_id=playlist.id)
        db.session.add(association)
        db.session.commit()

    return {"message": "Song added to playlist successfully"}, 200

```

Figura 3.5: Resursa “AddToPlaylist”

3. **Crearea și transmiterea răspunsului:** Dacă totul decurge cum trebuie, serverul creează un răspuns de tip JSON care include mesajul de succes și un “status-code” HTTP 200, care semnifică faptul că operațiunea s-a terminat cu succes. Acest răspuns este trimis înapoi către client.
4. **Primirea răspunsului de către client:** Când clientul primește răspunsul, îl procesează și, dacă este de succes, afișează un mesaj corespunzator către utilizator (în cazul nostru faptul ca playlist-ul a fost creat cu succes). În caz contrar, va fi gestionat mesajul de eroare.

Un factor esențial îl reprezintă asigurarea securității în procesul de schimb de informații dintre client și server. Aceasta este asigurată prin utilizarea de token-uri JWT (JSON Web Token). Astfel, fiecare solicitare de la client către server este însoțită de un token de autentificare care validează identitatea clientului.

Aceste token-uri sunt create pe server în momentul în care utilizatorul se autentifică cu succes. Aplicația client primește token-ul JWT din răspunsul HTTP și îl salvează în container-ul “SharedPreferences” pentru a fi utilizat în cererile ulterioare. Aceasta are rolul de a depozita datele private pe dispozitivul utilizatorului pentru a pu-

tea fi accesate cu ușurință. Atunci când clientul trimite o cerere, el include token-ul JWT în header-ul "Authorization" (ex. Figura 3.), urmând că acesta să fie validat de către server.

Acest lucru este asigurat prin folosirea decoratorului "@jwt_required()" (ex. Figura 3.3) înaintea fiecărei resurse definite în urma căreia utilizatorul trebuie să fie autentificat. Astfel se asigură că procesul de autentificare funcționează corect și doar utilizatorii logați care au un token JWT valid au acces la informațiile aplicatiei.

3.1.5 Sistemul de recomandare de melodii

Un sistem de recomandare cu filtrare pe bază de conținut face recomandări pe baza preferințelor utilizatorului și a caracteristicilor obiectelor. În cazul nostru, sistemul recomandă melodii în funcție de interacțiunile utilizatorului cu acestea.

Într-un sistem de filtrare pe bază de conținut, fiecare melodie este reprezentată printr-un set de caracteristici precum genul, limba, anul și artistul. Pe baza acestor caracteristici, sistemul învață care sunt preferințele utilizatorului cu ajutorul interacțiunilor de apreciere sau nu a unei melodii. Mai exact, o acțiune de "like" indică sistemului că utilizatorul preferă melodia respectivă, iar în caz contrar "dislike"-ul arată faptul că acesta nu-și dorește să primească un videoclip asemănător.

Interacțiunile sunt folosite pentru a crea un profil al utilizatorului, acesta putând fi mai apoi comparat cu profilurile altor melodii. Sistemul folosește tehnici precum similaritatea cosinusului pentru a calcula scorul de similaritate între profilul utilizatorului și profilul fiecărei melodii. Astfel, algoritmul oferă recomandări în funcție de scorurile de similaritate calculate. Melodia cu scorul de similaritate maxim va fi cea recomandată de sistem în momentul în care utilizatorul cere o melodie nouă prin metoda "getRecommendedSong" apelată în client.

```
liked_song_ids = [preference.song_id for preference in all_user_preferences if preference.liked]
disliked_song_ids = [preference.song_id for preference in all_user_preferences if not preference.liked]

songs_df = pd.DataFrame([{'id': song.id, 'artist_name': song.artist_name, 'genre': song.genre,
                           'language': song.language, 'year': song.year} for song in unrated_songs])

songs_df['combined_features'] = songs_df['genre'] + ' ' + songs_df['genre'] + ' ' + songs_df[
    'language'] + ' ' + songs_df['language'] + ' ' + songs_df['year'].astype(str)

tfidf = TfidfVectorizer()
tfidf_matrix = tfidf.fit_transform(songs_df['combined_features'])
```

Figura 3.6: Combinarea caracteristicilor unei melodii

Preferințele utilizatorului sunt salvate cu ajutorul metodei "sendSongPreferenceToServer" din activitatea principală a aplicației de fiecare dată când acesta oferă un "rating" unei melodii. Pe partea de backend, resursa "RecommendSong" este cea responsabilă de recomandarea de melodii pentru un utilizator. Aceasta preia toate preferințele utilizatorului și construiește un DataFrame pandas (songs_df) cu melodiile care încă nu au fost evaluate de utilizator. Pentru fiecare melodie, se stochează id-ul, numele artistului, genul, limba și anul. Acestea sunt reprezentate ca siruri de caractere ce vor fi concatenate în coloana "combined_features" a structurii de date (Figura 3.6) pentru a combina caracteristicile melodiei, acordând o importanță mai mare caracteristicilor de gen muzical și limbă.

În cele din urmă, se utilizează biblioteca scikit-learn pentru a crea un Vectorizer TF-IDF. Acesta este folosit pentru a transforma caracteristicile combinate într-o matrice de numere. Fiecare rând din această matrice va reprezenta o melodie, iar fiecare coloană o anumită caracteristică a acesteia. Valoarea din fiecare celulă reprezintă importanța acelei caracteristici pentru melodia respectivă, pe baza rarității sale în descrierea tuturor melodiilor. Acest criteriu al rarității este important în conceperea recomandărilor, prin acordarea unei ponderi mai mari caracteristicilor ce apar mai rar. De exemplu, dacă majoritatea melodiilor din baza de date sunt "rock", iar utilizatorul arată o preferință pentru genul "jazz", care este mult mai rar în setul nostru de melodi, acesta devine o caracteristică mai importantă în conceperea recomandărilor pentru acel utilizator.

După ce au fost combinate caracteristicile melodiilor apreciate, acestea vor fi folosite în calcularea similarităților dintre melodii. Mai întâi, se initializează cu zero un vector de lungime egală cu numărul de melodii. Acesta va fi folosit pentru a stoca scorurile de similaritate pentru fiecare melodie. Apoi, pentru fiecare melodie pe care utilizatorul a apreciat-o, se calculează similaritatea între acea melodie și toate celelalte, folosind coeficientul de similaritate cosinus (Figura 3.7). Aceasta reprezintă o măsură în procesarea limbajului natural care determină cât de similare sunt două documente, în cazul nostru, două melodii. În cele din urmă, vor fi atașate scorurile de similaritate în tabelul cu melodii și vor fi sortate în funcție de scor. Astfel, melodia cu scorul maxim va fi trimisă către client pentru a o afișa utilizatorului.

```

similarity_scores = np.zeros(tfidf_matrix.shape[0])

for i, song_id in enumerate(liked_song_ids):
    idx = songs_df[songs_df['id'] == song_id].index
    if idx.empty:
        continue
    liked_song_vector = tfidf_matrix[idx[0]]
    cosine_sim = cosine_similarity(liked_song_vector, tfidf_matrix)
    decay_factor = (5 - i) / 5
    similarity_scores += decay_factor * cosine_sim.flatten()

for i, song_id in enumerate(disliked_song_ids):
    idx = songs_df[songs_df['id'] == song_id].index
    if idx.empty:
        continue
    disliked_song_vector = tfidf_matrix[idx[0]]
    cosine_dissim = 1 - cosine_similarity(disliked_song_vector, tfidf_matrix)
    decay_factor = (5 - i) / 5
    similarity_scores -= decay_factor * cosine_dissim.flatten()

songs_df['similarity_score'] = similarity_scores
songs_df = songs_df.sort_values('similarity_score', ascending=False)

```

Figura 3.7: Calcularea similarităților

3.2 Frontend-ul aplicației

3.2.1 Tehnologii utilizate

- **Android:** este un sistem de operare mobil creat de Google, destinat pentru dispozitivele cu ecran tactil. Am ales Android că platformă pentru aplicația Songsurf, deoarece este foarte popular și oferă multă flexibilitate pentru dezvoltarea de aplicații mobile personalizate [1].
- **Java:** este un limbaj de programare de nivel înalt, orientat pe obiecte, devenind unul dintre cele mai apreciate limbi datorită portabilității, securității și performanței sale. Sistemul oferă o serie largă de beneficii, cum ar fi gestionarea automată a memoriei prin intermediul sistemului de garbage collection, prevenirea erorilor printr-un puternic sistem de excepții și suportul pentru programarea orientată pe obiecte. Prințipiu "scrie o dată, rulează oriunde", oferit de mecanismul Java Virtual Machine (JVM) este una din caracteristicile definitorii ale acestui limbaj de programare. Cu ajutorul acestuia codul Java compilat poate fi rulat pe orice dispozitiv care are JVM instalat, indiferent de sistemul de operare sau de

arhitectura hardware. De-a lungul timpului, Java a devenit unul dintre cele mai utilizate limbaje de programare pentru dezvoltarea aplicațiilor mobile, în special pentru platforma Android [5].

- **Fisierele XML de tip "layout":** sunt elemente esențiale în dezvoltarea aplicațiilor Android. Acestea oferă dezvoltatorilor posibilitatea de a crea și de a personaliza interfețele utilizatorului într-un mod declarativ și ușor de înțeles. Fisierele XML de layout permit separarea logicii aplicației (codul Java sau Kotlin) de interfața cu utilizatorul. Elementele de interfață, numite și "view"-uri, cum ar fi butoane, câmpuri de text și imagini sunt definite în aceste fișiere, alături de atrbute precum dimensiuni, culori sau margini. În Android Studio, dezvoltatorii pot utiliza un editor vizual pentru a manipula și a previzualiza aceste layout-uri, în vederea creării unei interfețe cât mai atractive.
- **Android YouTube Player:** este o bibliotecă ce implementează player-ul oficial YouTube în aplicațiile Android. Aceasta ne permite să integrăm direct funcția de redare a videoclipurilor muzicale în aplicație. Ea a fost concepută pentru a înlocui biblioteca YouTube Android Player API oferită de Google, care s-a dovedit a fi nesigură și acum este scoasă din funcțiune. Google încurajează în prezent utilizarea API-ului IFrame Player în interiorul unui WebView, exact ce face această bibliotecă. Așadar, utilizând această tehnologie nu există conflicte cu Termenii și Condițiile platformei YouTube. În plus, ea oferă o interfață Java nativă pentru interacțiunea cu player-ul web ceea ce facilitează integrarea acestuia cu aplicația Songsurf [10].
- **Volley:** reprezintă o bibliotecă de rețea HTTP dezvoltată de Google pentru platforma Android. Această tehnologie a fost construită pentru a ajuta dezvoltatorii de aplicații Android să gestioneze mai ușor și mai rapid operațiunile de rețea. Utilizând Volley, am putut asigura o comunicare eficientă și rapidă cu serverul, aceasta fiind capabilă să gestioneze o cantitate mare de solicitări simultane. Astfel se asigură un timp de răspuns rapid și o performanță crescută, factori importanți în aplicația Songsurf, utilizatorul fiind nevoit să navigheze ușor și rapid printr-o multitudine de videoclipuri muzicale [13].

3.2.2 Descrierea implementării

Activitățile reprezintă elemente de bază în structura aplicațiilor Android. Acestea au rolul de a oferi utilizatorului o interfață plăcută și o interacțiune fluidă. Fiecare activitate are asociat un fișier xml de tip "layout", acesta reprezentând interfață grafică care prezintă datele utilizatorului. În cele ce urmează, vor fi explicate activitățile care implementează funcționalitățile principale ale aplicației Songsurf.

Aplicațiile Android au anumite metode de bază ce sunt implementate de fiecare activitate în parte, cu ajutorul cărora se face conexiunea cu interfață vizuală. Metoda "onCreate" este apelată de fiecare dată când o nouă activitate este creată [4]. În acel moment sunt inițializate toate componentele cu care utilizatorul urmează să interacționeze, prin intermediul metodei "findViewById". Aceasta returnează id-ul specific vizualizării de care aparține, pentru că aceasta să poată fi tratat ca o instanță ce poate fi accesată și prelucrată în metodele de activitate. Metoda "setContentView" definește layout-ul care va fi folosit pentru activitatea curentă. Acesta este definit într-un fișier XML și dictează cum arată interfața utilizator. O altă metodă comună a mai multor activități este "showBottomNavigationMenu" care se ocupă cu afișarea meniului de navigație din partea de jos a ecranului.

Activitatea principală a aplicației, numită "MainActivity", este și cea mai importantă, aceasta oferind funcționalitatea de navigare prin melodii într-un mod interactiv și fluid.

Melodiile sunt afișate pe ecran prin intermediul unui videoplayer YouTube integrat în aplicație cu ajutorul librăriei "androidyoutubeplayer". Pentru a reda o melodie, este preluat id-ul piesei respective din obiectul Song, care este folosit pentru a încărca videoclipul în "YouTubePlayer" cu ajutorul metodei "loadOrCueVideo()". În acest mod, cu ajutorul metodei "playNextSong" care este apelată când se detectează un gest de swipe, se trece la melodia următoare și începe redarea ei.

Gesturile de glisare care facilitează navigarea melodii sunt gestionate de clasa "SwipeGestureDetector" care moștenește "GestureDetector". Cu ajutorul acesteia, este creată metoda "OnFling" unde sunt calculate diferențele pe axa X și Y între punctul de start și punctul de sfârșit al gestului de swipe. Aplicația detectează o glisare orizontală în momentul în care valoarea absolută a diferenței pe axa X este mai mare decât cea a diferenței pe axa Y. Gestul de swipe este considerat valid dacă acesta depășește o anumită distanță limită predefinită și o viteză minimă. Astfel, în funcție de direcția

glisării, se vor apela metodele "onSwipeRight()" sau "onSwipeLeft()" pentru a oferi funcționalitățile așteptate.

```
private void initYouTubePlayerView() {
    IFramePlayerOptions options = new IFramePlayerOptions.Builder().controls(0).rel(0).build();
    getLifecycle().addObserver(youTubePlayerView);

    youTubePlayerView.initialize(new AbstractYouTubePlayerListener() {
        @Override
        public void onReady(@NotNull YouTubePlayer youTubePlayer) {
            MainActivity.this.youTubePlayer = youTubePlayer;

            songFetchExecutor = Executors.newSingleThreadExecutor();
            for (int i = 0; i <= BUFFER_SIZE; i++) {
                fetchAndQueueSong();
            }
        }
    }, handleNetworkEvents: true, options);
}
```

Figura 3.8: Metoda "initYouTubePlayerView"

Melodiile primite de la server sunt stocate într-o coadă care funcționează pe principiul „primul venit, primul servit” (FIFO), ceea ce înseamnă că melodia care a fost adăugată prima în coadă va fi redată prima. ExecutorService este un serviciu care gestionează un set de fire de execuție și permite executarea asincronă a task-urilor. Cu ajutorul acestuia este implementată logica de preluare a melodii din coadă. Atunci când se preia o melodie nouă, se adaugă un task în "songFetchExecutor". Acesta va cere un videoclip muzical de la server prin intermediul metodelor "getSong" sau "getRecommendedSong" și îl va adăuga în coada cu priorități.

Pentru a oferi o experiență fluidă și rapidă, a fost implementat un "buffer" de stocare a melodii. Scopul acestuia este de a asigura că există întotdeauna melodii preîncărcate disponibile pentru redare. Astfel, utilizatorul nu va fi nevoie să aștepte ca o nouă melodie să fie preluată de la server. În Figura 3.6 este prezentat modul în care este initializat player-ul de YouTube, prin metodele amintite anterior.

Comunicarea cu serverul aplicației se face prin intermediul metodelor "getSong" și "getRecommendedSong", care fac cereri HTTP pentru a primi datele unei melodii aleatoare sau recomandate. De asemenea, metoda "sendSongPreferenceToServer" trimite preferințele utilizatorului (like sau dislike) către server.

Opțiunile de adăugare a filtrelor sau a unei melodii în playlist au fost gestionate prin clase de tipul "ActivityResultLauncher". Acest tip de clase este folosit pentru a porni o activitate printr-un "Intent" conform unui anumit rezultat. De exemplu,

în cazul filtrorelor, dacă rezultatul activității este "OK", se preiau filtrele selectate și se actualizează melodiile afișate în funcție de acestea.

Pentru organizarea codului într-un mod eficient s-a conceput o clasă abstrasta numită "BasePlaylistActivity" care încorporează toate funcționalitățile comune a tuturor activităților legate de liste de redare. Aceasta include metode pentru crearea de noi playlist-uri, preluarea listei de playlist-uri de la server sau filtrarea playlist-urilor pe baza textului introdus în bara de căutare.

Activitatea "My Playlists" a fost concepută pentru afișarea într-un mod atractiv a tuturor playlist-urilor create de utilizator. Această subclasă a "BasePlaylistActivity" se ocupa cu reprezentarea pe ecran a tuturor listelor de radare și opțiunea de a selecta un playlist.

Aceste liste sunt reprezentate prin "PlaylistsArrayAdapter", o clasa care extinde ArrayAdapter. Utilizând această clasă, utilizatorul își poate personaliza listele de redare, în cazul nostru tratează umplerea listelor pentru activitățile "My Playlists" și "AddToPlaylist", fiecare având un design propriu, determinat prin intermediul unui "ListView" sau "GridView". Prin metoda "getView": este chemat pentru fiecare element din listă sau grilă de playlist-uri și se returnează o vizualizare pentru acel element.

Activitatea "VideoPlayer" are rolul de a afișa un videoclip selectat de utilizator, și de a controla modul de redare a acestuia. Metoda "togglePlayPause()" controlează playback-ul videoclipului. Dacă videoclipul este în curs de redare, acesta poate fi pus pe pauză cu ajutorul butoanelor specifice. Metoda selectSong() schimbă melodia curentă în funcție de direcția selectată de utilizator (următoarea sau precedentă) și începe redarea acesteia.

Metoda "onProgressChanged()" este apelată atunci când timpul curent al videoclipului este actualizat în interfață cu utilizatorul și progresul "SeekBar"-ului se modifică. Pentru a trata cazul în care utilizatorul are o interacțiune mai îndelungată cu bara de actualizare, s-au implementat metodele "onStartTrackingTouch()" și "onStopTrackingTouch()", acestea controlând timpul curent al videoclipului.

Activitatea "My Account" a fost creată pentru a oferi utilizatorilor opțiuni pentru gestionarea contului propriu de utilizator. În acest sens, au fost create metodele ce afișează o fereastră de tip dialog în funcție de operațiunea de modificare sau ștergere aleasă. De exemplu în Figura 3.7, "showChangeUsernameDialog" va afișa caseta de

```
private void showChangeUsernameDialog() {
    View dialogView = LayoutInflater.from( context: this).inflate(R.layout.text_input_dialog, findViewById(android.R.id.content),
        EditText newUsername = dialogView.findViewById(R.id.input);
        newUsername.setInputType(InputType.TYPE_CLASS_TEXT);

        new AlertDialog.Builder( context: this)
            .setTitle("Change Username")
            .setView(dialogView)
            .setPositiveButton( text: "Save", (dialog, whichButton) -> changeUsername(newUsername.getText().toString()))
            .setNegativeButton( text: "Cancel", (dialog, whichButton) -> dialog.dismiss()).show();
}
```

Figura 3.9: Metoda “showChangeUsernameDialog”

dialog care permite utilizatorului să își introducă noul nume de utilizator dorit. Mai apoi, acesta cere confirmarea utilizatorului pentru a efectua operația selectată. Metodele de modificare a datelor de utilizator sunt create prin crearea de cereri specifice cu ajutorul librăriei Volley care vor fi trimise mai apoi către server.

Capitolul 4

Ghidul de utilizare

În capitolul curent vor fi prezentate funcționalitățile principale ale aplicației și modalitățile prin care utilizatorul poate interacționa cu acestea. Astfel, se vor prezenta funcțiile de navigare printre melodii, aplicare de filtre, adăugare în playlist-uri, afișare de playlist-uri și melodii sau personalizarea profilului de utilizator.

4.1 Navigarea printre melodii

După procedeul de autentificare, utilizatorul este redirectionat către meniul principal al aplicației, cel de navigare printre melodii. Aceasta poate alege dintre cele două opțiuni de navigare, mai precis dintre melodii personalizate pe baza filtrelor , sau recomandate în funcție de "rating"-ul oferit ultimelor melodii afișate.

Navigarea printre melodii (Figura 4.1) se face într-un mod interactiv printr-un gest de "swipe". În secțiunea de melodii recomandate, acest gest are rolul de a da un "rating" de "like" sau "dislike" melodiei curente, în funcție de direcția în care s-a produs acest gest. Sunt introduse și butoane specifice pentru această operăriune astfel încât aplicația să devină cât mai intuitivă pentru utilizator. De asemenea, opțiunea de adăugare a filtrelor apare doar în secțiunea afișării de melodii personalizate pentru a face o distincție clară între cele două tipuri de navigare. Ambele au în comun însă operaționarea de adăugare unei melodii într-un playlist.

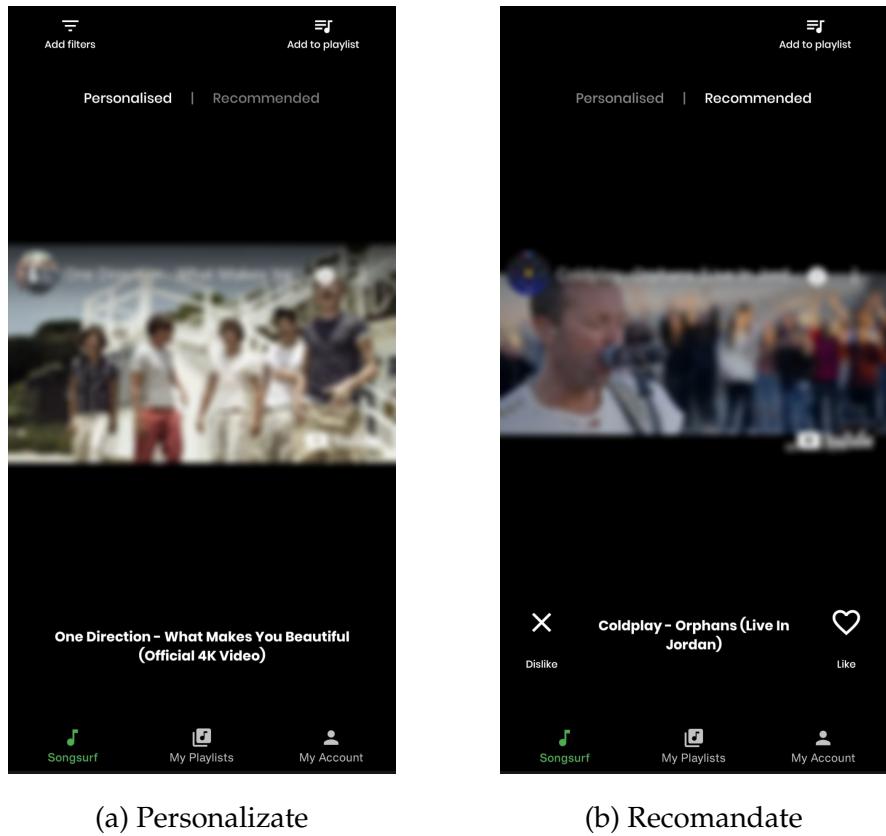


Figura 4.1: Navigarea printre melodii

4.1.1 Adaugarea filtrelor

În figura alăturată (Figura 4.2) este prezentată pagina responsabilă adaugării de filtre a aplicației Songsurf.

Prin apăsarea butonului de "Add filters", vă apărea o listă de filtre pentru diverse genuri muzicale (rock, pop, hip-hop etc.), limbi (română, engleză, franceză etc.) sau anul în care a fost publicată melodia pe platforma Youtube (din 2006 până în prezent).

Filtrele sunt construite într-un mod dinamic astfel încât se pot selecta doar combinații existente în baza de date a aplicației. După aplicarea acestoara, melodiile vor fi filtrate în funcție de opțiunile alese de utilizator.

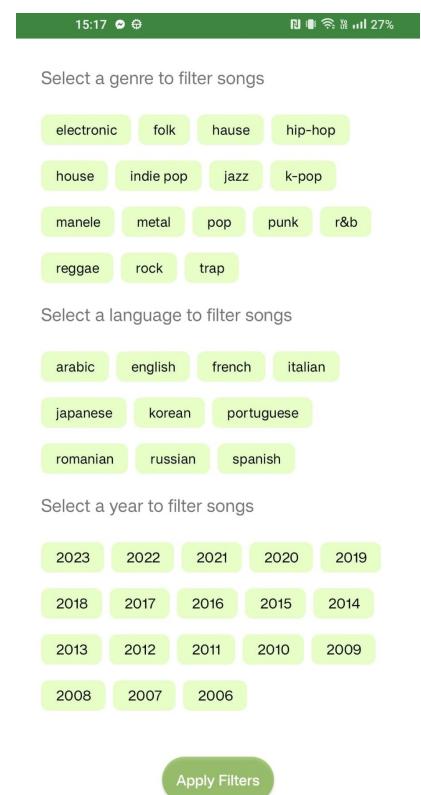


Figura 4.2: Pagina de filtrare

4.2 Navigarea printre playlist-uri

În Figura 4.3 (a) se poate observa interfață optiunii de adăugare a melodiei în diverse playlist-uri, iar în Figura 4.3 (b) afișarea unei liste de redare ce a fost selectată. În ambele secțiuni utilizatorul poate căuta într-un mod dinamic playlist-uri în funcție de numele introdus în bara de căutare, sau poate crea unul nou prin apăsarea butonului "New playlist".

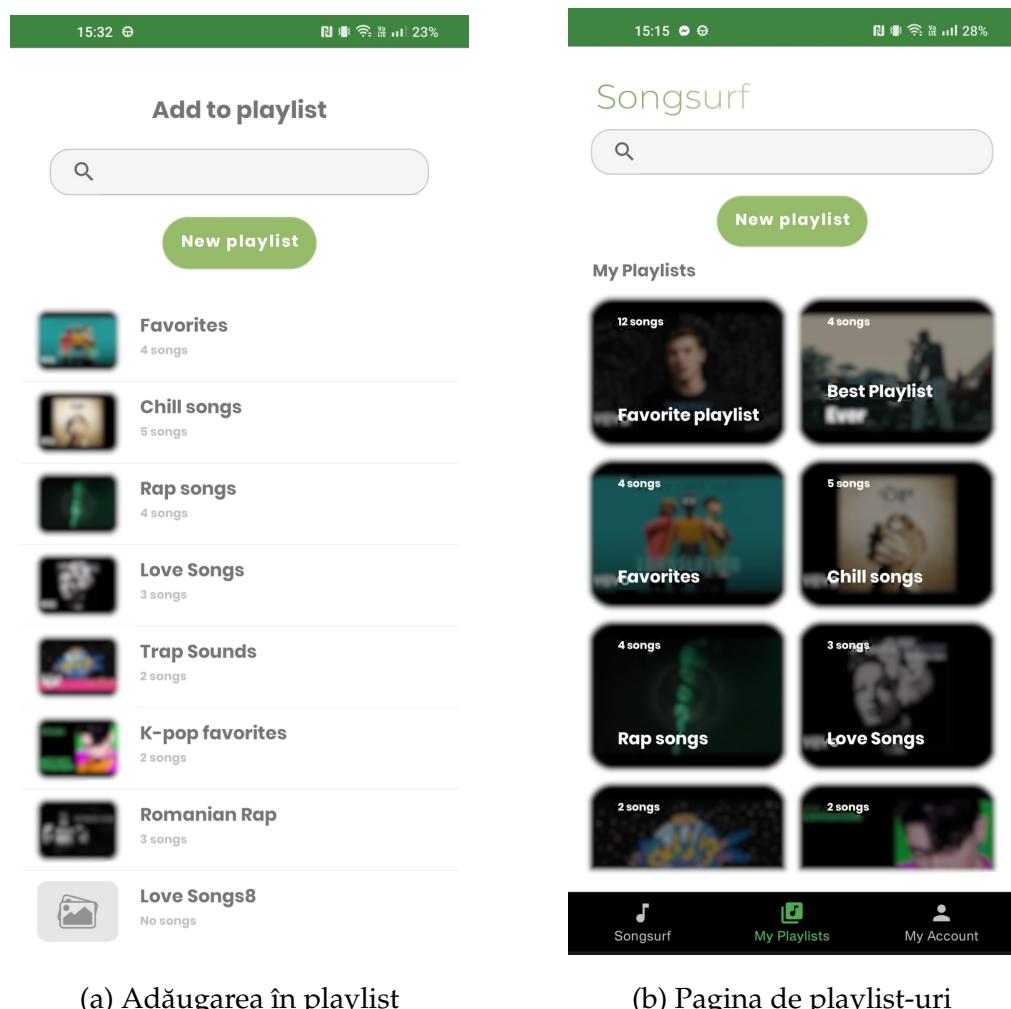


Figura 4.3: Navigarea printre playlist-uri

4.3 Afisarea playlist-urilor

În momentul în care un playlist este selectat, vor fi afișate melodile ce aparțin acestuia, împreună cu o serie de opțiuni de personalizare. În primul rand, utilizatorul poate apăsa butonul de sortare care oferă 3 moduri de aranjare a playlist-ului: alfabe-

tic, în ordinea în care melodiile au fost introduse sau în mod aleatoriu dacă se dorește amestecarea melodiilor pentru sesiuni de redare diverse.

Utilizatorul poate șterge melodii printr-o apăsare mai lungă pe una din ele, poate redenumi playlist-ul după bunul plac sau îl poate șterge. Nu în ultimul rand, acesta poate selecta butonul de "Play" care va reda playlist-ul începând de la prima melodie. Toate aceste operațiuni sunt prezentate în Figura 4.4 (a).

În Figura 4.4 (b) este afișată pagina de redare a unei melodii. Aici utilizatorul poate naviga spre melodia precedenta sau următoare, poate pune pauză videoclipului sau îl poate derula la un unumit timp specific prin intermediul unui "seekbar". În momentul în care melodia se termină, videoplayer-ul vă trece automat la următoarea prezentă în lista de redare.

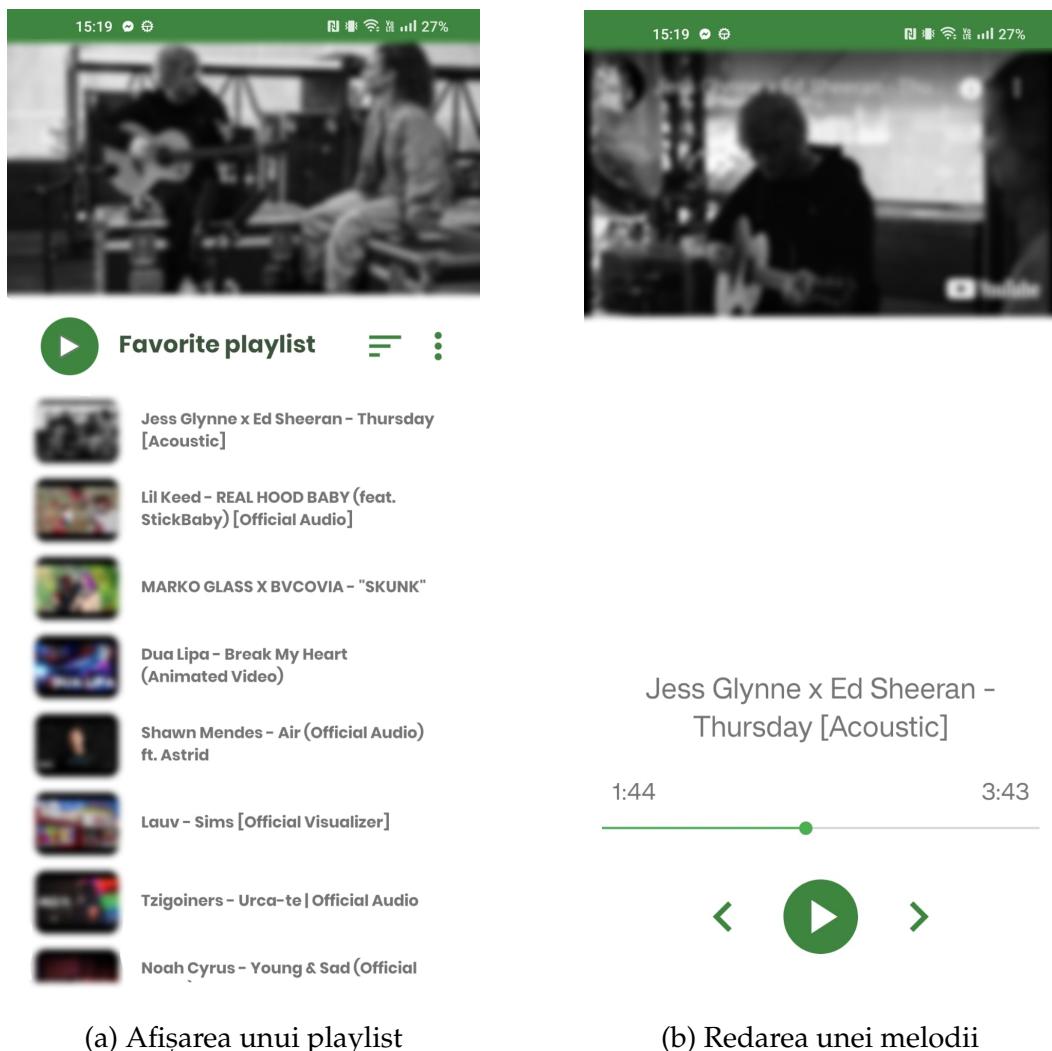


Figura 4.4: Afișarea playlist-urilor

4.4 Administrarea contului

Sectiunea "My Account" a fost creată pentru a permite utilizatorului să își personalizeze contul, oferind o serie de opțiuni de modificare a datelor din cont precum: nume de utilizator, adresă de email sau parolă.

Pentru asigurarea securității contului de utilizator, acesta trebuie să își introducă parola curentă atunci când alege să o modifice sau atunci când dorește să își schimbe email-ul. Dacă acesta va introduce o adresă de email deja existentă în baza de date sau un username care a mai fost folosit, atunci utilizatorul vă fi rugat să adauge alte informații.

Tot în acest meniu, utilizatorul își poate șterge contul sau poate ieși din el prin apăsarea butonului "Sign Out".

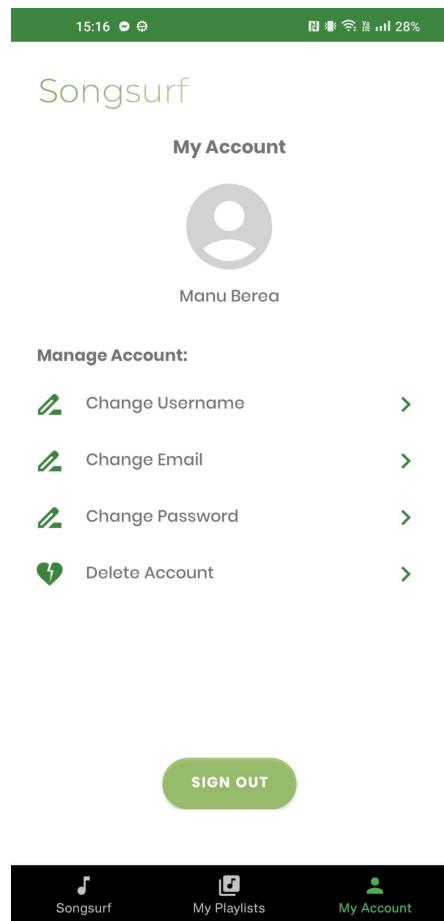


Figura 4.5: Pagina "My Account"

Concluzii

De-a lungul timpului, am dezvoltat o pasiune puternică pentru muzică și am realizat câte beneficii ne aduce stării noastre de spirit și experiențelor de zi cu zi. Am observat însă că platformele actuale, deși sunt foarte performante, nu includ toate cerințele mele, în special ce-a de rapiditate în descoperirea de noi melodii. Astfel, mi-am dorit să creez o aplicație care să facă din acesta căutare o activitate eficientă, plăcută și interactivă.

Songsurf reprezintă rezultatul încercării de a crea aplicația muzicală perfectă, care să fie intuitivă și ușor de folosit, în care utilizatorul își poate petrece ore întregi navigând printre melodii, dezvoltându-și astfel cultura muzicală. Am reușit astfel să creez o aplicație mobile în care poți descoperi melodii rapid printr-un singur gest, care te ține curios afișând videoclipuri muzicale create într-un mod artistic, dar care îți oferă și un grad de personalizare ridicat. Spre exemplu, deseori mi-am dorit să ascult melodii și în alte limbi, dar nu știam exact ce artiști să caut. În acest sens am adăugat filtre ce pot rezolva această problemă având opțiunea să filtrez melodii în funcție de limba, gen muzical sau an.

Totuși, există multe modalități prin care aplicația Songsurf poate fi îmbunătățită, aşadar voi enumera câteva dintre direcțiile viitoare în care se poate dezvolta aplicația.

- Îmbunătățirea sistemului de recomandare de melodii pentru a se potrivi mai bine cu preferințele utilizatorului. În prezent algoritmul de recomandare cu filtrare pe baza conținutului este unul limitat și nu are cea mai puternică acuratețe. În acest sens, aplicația ar putea fi îmbunătățită pe viitor implementând un algoritm de filtrare colaborativă, în care recomandările să fie calculate și în funcție de asemănările dintre preferințele a diversi utilizatori.
- Adăugarea funcționalității de a partaja playlist-urile cu alți utilizatori sau cu alte platforme. Mi-am dorit să adaug funcționalitatea de a partaja playlist-urile create

cu platforme precum Youtube sau Spotify, fiind foarte populare și utilizate de o masă largă de oameni, însă timpul nu mi-a permis acest lucru.

- Adăugarea suportului pentru alte sisteme de operare precum iOS sau web, în prezent aplicația putând fi rulată doar pe dispozitive Android.
- Popularea bazei de date cu mai multe melodii și adăugarea de noi caracteristici pentru acestea, astfel încât filtrele să fie mai diverse și recomandările mai eficiente și calitative.

În concluzie, deși există loc de îmbunătățiri, aplicația Songsurf aduce o soluție excelentă pentru cei care doresc să descopere rapid conținut nou și să se bucure de muzică într-un mod personalizat și interactiv.

Bibliografie

- [1] Bryan Sills, Brian Gardner, Kristin Marsicano, Chris Stewart, *Android Programming: The Big Nerd Ranch Guide*, Addison-Wesley Professional, 5th edition, 2022
- [2] Michael J. Hernandez, *Database Design for Mere Mortals: A Hands-On Guide to Relational Database Design*, Addison-Wesley Professional, 2nd edition, 2003
- [3] Myers Jason, *Essential SQLAlchemy: Mapping Python to Databases*, O'Reilly Media, 2015
- [4] John Horton, *Android Programming for Beginners*, Ingram short title, 2nd edition, 2018
- [5] Kathy Sierra, Bert Bates, Trisha Gee, *Head First Java*, O'Reilly Media, 3rd edition, 2022
- [6] <https://www.cnbc.com/2018/05/25/youtube-music-review.html>
- [7] What is Client-Server Architecture? Everything You Should Know: <https://www.simplilearn.com/what-is-client-server-architecture-article>
- [8] How Spotify has changed the way we listen to music
<https://audiodee.com/articles/how-spotify-has-changed-the-way-we-listen-to-music/>
- [9] What is PostgreSQL?
https://www.tutorialspoint.com/postgresql/postgresql_overview.htm
- [10] YouTube Player API Reference for iframe Embeds
https://developers.google.com/youtube/iframe_api_reference
- [11] MVC Framework Introduction
<https://www.geeksforgeeks.org/mvc-framework-introduction/>

[12] What is Scikit-Learn in Python?

<https://www.activestate.com/resources/quick-reads/what-is-scikit-learn-in-python/>

[13] Volley Library in Android

<https://www.geeksforgeeks.org/volley-library-in-android/>

[14] Developing RESTful APIs with Python and Flask

<https://auth0.com/blog/developing-restful-apis-with-python-and-flask/>

[15] Main music genres & subgenres

<https://www.chosic.com/list-of-music-genres/>

Mulțumiri

În procesul de realizare a ghidului de utilizator pentru aplicația Songsurf, am utilizat o serie de capturi de ecran pentru a ilustra mai bine funcționalitatea și interfața acesteia. În acest sens, s-a ținut cont de protejarea drepturile de autor ale artiștilor care reprezintă videoclipurile muzicale. Astfel, toate secvențele din capturile de ecran care ar putea afișa videoclipuri muzicale sau imagini de tip "thumbnail" ale acestora, au fost blurate pentru a evita orice potențială încălcare a drepturilor de autor.