

UNIVERSITATEA NAȚIONALĂ DE ȘTIINȚĂ ȘI TEHNOLOGIE
POLITEHNICA DIN BUCUREȘTI
FACULTATEA DE AUTOMATICĂ ȘI CALCULATOARE
DEPARTAMENTUL DE CALCULATOARE



PROIECT DE DIPLOMĂ

Sleeve Music Library
Aplicație socială de critică muzicală

Nițu David-Gabriel

Coordonator științific:

Dr. ing. Cătălin Negru
Prof. Dr. Ing. Florin Pop

BUCUREȘTI

2024

NATIONAL UNIVERSITY OF SCIENCE AND TECHNOLOGY
POLITEHNICA BUCHAREST
FACULTY OF AUTOMATIC CONTROL AND COMPUTERS
COMPUTER SCIENCE AND ENGINEERING DEPARTMENT



DIPLOMA PROJECT

Sleeve Music Library
Social Application for Music Criticism

Nițu David-Gabriel

Thesis advisor:

Dr. ing. Cătălin Negru
Prof. Dr. Ing. Florin Pop

BUCHAREST

2024

CUPRINS

1	Introducere	1
1.1	Context	1
1.2	Problema	1
1.3	Obiective	2
1.4	Structura lucrării	3
2	Analiza Cerințelor	4
3	Studiu de Piață	7
3.1	Produse similare	7
3.1.1	Spotify	7
3.1.2	Instagram	8
3.1.3	Letterboxd	9
3.1.4	Rate Your Music	10
3.1.5	Album Of The Year	11
3.2	Tehnologii Alternative	11
3.2.1	Firebase	12
3.2.2	Kotlin	12
3.2.3	Swift	12
3.2.4	Flutter	13
4	Soluția Propusă	14
4.1	Arhitectura Aplicației	14
4.1.1	Arhitectura generală a aplicației	14
4.1.2	Baza de date din Strapi	16
4.1.3	Funcționalitățile aplicației	18

5	Detalii de implementare	24
5.1	Conectarea la bazele de date	24
5.1.1	Strapi	24
5.1.2	Spotify API	25
5.2	Implementări notabile	27
5.2.1	Search Page	27
5.2.2	Sort Page și Listened To	28
5.2.3	Follow	29
5.2.4	Interacțiunea cu albumele	30
5.3	Tehnologii Folosite	31
5.3.1	React Native	31
5.3.2	Strapi	31
6	Evaluare	33
6.1	Testarea Funcționalităților	33
6.2	Testarea Performanței	38
6.3	Posibile probleme	40
7	Concluzii	41
7.1	Posibile îmbunătățiri	41
7.2	Experiența personală	41

SINOPSIS

Muzica este un limbaj universal; este iubire, tradusă în cea mai pură formă de artă. Oameni de pretutindeni împărtășesc pasiunea pentru muzică, însă conexiunile între ei pe baza acestui interes comun sunt tot mai greu de găsit sau de întreținut. Lipsa unei modalități de a-ți exprima opiniile și de a le vedea pe cele ale prietenilor este cauza principală.

Sleeve Music Library este o aplicație care dorește să le ofere utilizatorilor un spațiu în care să își exprime opiniile muzicale și să le vadă pe cele ale altor utilizatori, având ca scop crearea unei comunități ce facilitează descoperirea de albume noi și conectarea cu prietenii, pentru a întări legătura prin gusturile muzicale comune.

Aplicația a fost dezvoltată folosind React Native și Expo pentru front-end, iar pentru back-end Strapi, legătura cu baza de date fiind făcută prin cereri de tip REST API. Ea beneficiază de acces la cea mai mare bază de date muzicală din lume – Spotify Database, ceea ce oferă garanția corectitudinii datelor despre albume.

ABSTRACT

Music is a universal language; it is love, translated into the purest form of art. People everywhere share the passion for music, but connections based on this common interest are increasingly difficult to find or maintain. The primary cause is the lack of a way to express opinions and see those of friends.

Sleeve Music Library is an application designed to provide users with a space where they can express their musical opinions and see those of other users. Its main goal is creating a community that facilitates the discovery of new albums and allows people to connect with friends, strengthening their bonds through shared musical taste.

The application was developed using React Native and Expo for the front-end, and Strapi for the back-end, with the connection to database made through REST API requests. It benefits from access to the world's largest music database – the Spotify Database, ensuring the accuracy of album information.

1 INTRODUCERE

1.1 Context

”Conform studiului Nielsen Music 360, în jur de 90 – 93% din populația lumii ascultă muzică. Nu este o surpriză, deoarece cercetările arată numeroasele beneficii ale ascultării muzicii. Aceste beneficii sunt atât fizice, cât și psihologice. Pe scurt, muzica ne face să fim mai fericiți și să ne simțim mai bine. Stimulează creierul să elibereze dopamina și endorfina, ceea ce ne face să ne simțim mai fericiți și mai bine. Beneficiile atribuite ascultării muzicii includ ameliorarea durerii, reducerea anxietății, scăderea tensiunii arteriale, îmbunătățirea somnului, creșterea vigilenței și îmbunătățirea memoriei.” [12]

Toată lumea iubește muzica. Ea este necesară și constantă în viața oricărui om. Din totalitatea ascultătorilor de muzică, un procent semnificativ este pasionat și dorește să poarte discuții despre părerile lor muzicale. O astfel de discuție începe cu clasică întrebare “Tu ce muzică asculți?”, pentru care multora le este foarte greu să găsească un răspuns, unele dintre cauze fiind: timpul limitat, panica de moment, anxietatea, posibilitatea ca interlocutorul să nu fie familiar cu artiștii enumerați, neputința de a face o sinteză a tuturor genurilor de muzică ascultate și simplul fapt că oamenii nu sunt niciodată pregătiți să răspundă la această întrebare, în ciuda relevanței ei. Relațiile construite pe baza pasiunilor comune sunt în general foarte intense și de durată, motiv pentru care nu ar trebui ignorată importanța acestei întrebări. Sleeve Music Library caută să rezolve aceste situații inoportune, oferind un spațiu plăcut care încurajează conexiunile între oameni, crearea de relații noi pe baza pasiunii comune pentru muzică, păstrarea unui catalog al albumelor ascultate, dar totodată și descoperirea de muzică nouă, totul împachetat într-o interfață intuitivă, cochetă și ușor de utilizat.

1.2 Problema

Industria muzicală este într-o creștere constantă, iar această tendință nu pare să se oprească prea curând. Capitalismul și-a lăsat amprenta în această industrie prin desensibilizarea și comercializarea excesivă a muzicii, transformând-o într-o cursă după profit. Acest fenomen a diminuat semnificativ dorința oamenilor de a se aventura în acest domeniu, deoarece este mult mai ușor să asculte pur și simplu ceea ce este popular și promovat de radiourile controlate de corporații. Monopolul asupra preferințelor muzicale este accentuat și de absența unei platforme centralizate în care pasionații să-și poată exprima opiniile, iar cei care doresc să afle mai multe să găsească cu ușurință recomandări de la critici, dar și de la prietenii lor.

Fără un astfel de spațiu, cei cu adevărat pasionați de muzică se simt pierduți într-o lume în care nu prea se mai pune accent pe pasiuni reale, având dificultăți în a se conecta între ei și în a forma relații noi. În același timp, cei care doresc să intre în lumea muzicii se confruntă cu obstacole, deoarece lipsa unei comunități consolidate funcționează ca un zid ce îi ține departe de posibilitățile de a-și aprofunda cunoștințele.

Accesul limitat la o comunitate deschisă, eventual sub forma unei platforme ușor de utilizat, optimizată pentru interacțiune și schimb de opinii împiedică dezvoltarea unei culturi muzicale autentice și îngreunează crearea sau menținerea conexiunilor între indivizi interesați de acest domeniu fascinant. Majoritatea platformelor existente în prezent cu această utilitate se confruntă cu o interfață greoaie, asemănătoare cu cea a unui blog din urmă cu zece ani, furnizând informații neclare și greu de găsit, și oferind o personalizare minimală. Utilizatorii simt nevoia de a-și aduce propriul lor aport și de a simți că profilul lor pe o aplicație le aparține cu adevărat. De asemenea, este esențial ca ei să fie încurajați să revină pe platformă.

În plus, muzica este o artă, iar prin natura sa, cere analiză și critică. Pe lângă criticii de muzică profesioniști, a caror opinie cu siguranță este relevantă și va fi mereu, părerea consumatorului obișnuit este la fel de importantă. În ciuda nevoii muzicii de a evolua și de a ajunge la toată audiența țintă, și a oamenilor de a își exprima opinia și de a simți că aceasta contează, o astfel de aplicație, în adevăratul sens al cuvântului, încă nu există.

1.3 Obiective

Sleeve Music Library are ca scop principal îmbunătățirea modului în care oamenii se conectează atât cu muzica, cât și între ei. Aceasta oferă o platformă cu o interfață intuitivă și prietenoasă, care permite utilizatorilor să-și creeze un profil personalizabil, similar cu cele ale aplicațiilor de tip social media.

Platforma dorește să mute accentul de pe consumul rapid și superficial al muzicii, specific acestei ere dominate de serviciile de streaming și radio, către o relație mai profundă cu arta. Sleeve Music Library permite utilizatorilor să reflecteze asupra albumelor ascultate, să le ofere note și să își împărtășească experiențele muzicale.

Un loc centralizat ca acesta poate să stârnească cu ușurință discuții captivante între prieteni pe baza noilor adiții în topul lor de albume sau a listelor create de ei de genul "albumele mele preferate de jazz", "jazz-rap essentials" sau "city pop I have on vinyl".

De asemenea, platforma oferă un mod simplu și confortabil pentru cei care doresc să descopere muzică nouă și să găsească recomandări, fie de la critici, fie de la prietenii lor. Astfel, utilizatorii nu mai trebuie să navigheze prin milioane de bloguri cu recenzii profesionale care adesea se contrazic, ajungând în final să fie și mai confuzi.

Pe lângă aceste funcționalități, Sleeve Music Library permite utilizatorilor să își monitorizeze gusturile muzicale și evoluția lor în timp, să țină evidența istoricului de ascultare și să aibă acces la toate albumele existente.

1.4 Structura lucrării

Lucrarea este împărțită pe capitole și sub-capitole pentru a păstra o structură cât mai curată și ușor de citit. Fiecare capitol va exemplifica în detaliu ideea principală prezentată mai jos.

Analiza Cerințelor prezintă problemele pe care aplicația încearcă să le rezolve, cu suportul unor statistici realizate în urma unui chestionar online, completat de potențiali utilizatori.

Studiu de Piață exemplifică diverse soluții similare, oferă detalii despre ele, cât și o analiză a lor pentru a arăta problemele pe care aceste soluții le întâmpină. În plus, prezintă câteva dintre tehnologiile cele mai relevante pentru o aplicație de acest tip, care nu au fost alese pentru Sleeve.

Soluția Propusă elaborează arhitectura generală, modul de interacțiune a componentelor, bazele de date și funcționalitățile principale ale aplicației. Acest capitol ajută la înțelegerea modului de funcționare al aplicației și oferă o imagine de ansamblu a acesteia.

Detalii de Implementare explică în întregime modul de conectare la bazele de date și cum comunică acestea cu clientul. Tot aici sunt prezentate părți mai interesante din implementare, dificultăți tehnologice și soluțiile găsite pentru ele. Totodată, prezintă o descriere a tehnologiilor alese pentru dezvoltare.

Evaluare pune la dispoziție diferite teste asupra funcționalităților aplicației și oferă diverse metrice și grafice ce analizează performanțele sale.

Concluzii oferă o privire retrospectivă asupra proiectului realizat și posibile îmbunătățiri. În plus, el descrie călătoria parcursă pentru a dezvolta această aplicație.

2 ANALIZA CERINȚELOR

Pentru a descoperi cerințele posibile ale utilizatorilor, am creat un chestionar online care include 10 întrebări despre conexiunea lor cu muzica, modul lor de interacțiune cu prietenii pe acest subiect, problemele și dorințele lor. Următoarele statistici sunt obținute în urma completării acestui chestionar de aproximativ 50 de oameni cu poziții foarte variate față de acest subiect.



Figura 1: Întrebări generale despre muzică

Din rezultatele primei întrebări se poate observa clar importanța muzicii în viața oamenilor, cu peste 55% alegând nivelul de importanță maximă. A doua întrebare consolidează această opinie, prezentând timpul petrecut în fiecare zi ascultând muzică, ajungând chiar să fie prezentă în mai mult de 16% din ziua lor. A treia întrebare dovedește premisa prezentată în introducere: confuzia și dificultatea de a comunica opiniile cu acuratețe.



Figura 2: Importanța unui profil personalizat

A patra întrebare prezintă părerea pozitivă despre importanța personalizării profilului pe o platformă de social media. Oamenii vor să își lase amprenta și să exprime cine sunt cu adevărat prin profilul lor, lucru cu care sunt de acord peste 90% dintre participanții la chestionar. A cincea întrebare vine în ajutorul celei de-a patra pentru a răspunde direct la problema prezentată la întrebarea cu numărul 3: dificultatea întocmirii unui răspuns complet pe loc la întrebarea "Tu ce muzică asculți?". Participanții tind să creadă aproape în totalitate că un profil personalizat în prealabil conform părerilor lor le-ar permite să encapsuleze mai eficient cine sunt ei și i-ar ajuta să răspundă la această întrebare pe loc, având opțiunea să ofere interlocutorului acest profil.



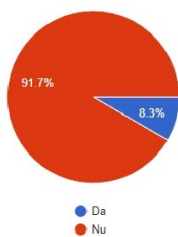
Figura 3: Muzica între prieteni

Întrebarea 6 confirmă popularitatea muzicii ca subiect de discuție între prieteni. Se observă totuși o predominanță puțin mai mică a răspunsurilor pozitive decât la alte întrebări, cu doar 86% de voturi "da". Nu este deloc un procent mic, doar că principala cauză pentru care e puțin mai mic decât la celelalte întrebări este chiar problema prezentată la întrebarea numărul 3: neputința oamenilor de a-și comunica eficient opiniile pe moment, panica și anxietatea socială, lucru pe care Sleeve încearcă să îl rezolve. Întrebarea 7 vine ca pregătire pentru întrebarea 8. Oamenii iubesc să descopere muzică nouă, dar și mai mult iubesc când au o legătură cu muzica pe care o ascultă. Se remarcă astfel cât de vitală este opinia prietenilor în luarea de decizii, prin simpla conexiune pe care o creează cu lucrul recomandat de ei. O recomandare familiară pare să fie mult mai ușor de acceptat decât una profesionistă, cu peste 97% de voturi în acest sens.

În cele din urmă, se poate vedea că platformele tradiționale de streaming chiar sunt văzute doar ca o metodă superficială de consum al muzicii, cu peste 90% dintre votanți care consideră că nu își pot exprima opiniile pe acest tip de aplicație. Ultima întrebare confirmă cererea pentru o platformă care ar încorpora toate ideile prezentate mai sus, în cadrul căreia oamenii să poată să își exprime opiniile.

9

Ti se pare suficient Spotify/alt streaming platform cand vine vorba de exprimarea opiniilor tale despre muzica?



10

Ti s-ar parea de folos sa ai o aplicatie in care sa poti vedea gusturile muzicale ale prietenilor tai sau ale oamenilor noi pe care ii cunosti?



Figura 4: Importanța unei aplicații precum Sleeve

Sleeve dorește să rezolve problemele discutate mai sus prin oferirea unei platforme sociale pe care oamenii să rămână conectați sau să creeze conexiuni noi, să se simtă lipsiți de judecăți și liberi să își exprime gusturile muzicale. Cu acces la aproape orice album existent, ea are ca scop să fie și o librărie în care oamenii să își păstreze un istoric al călătoriei lor în lumea muzicii.

Aplicația permite utilizatorilor să se logheze sau să-și creeze un cont nou, să gestioneze și să descopere liste de albume, să caute albume, artiști și alți utilizatori, și să își personalizeze profilul. Utilizatorii pot pune albumele pe lista de ascultate sau pe lista cu albume ce urmează să fie ascultate, pot să le evalueze, să își modifice topul lor de 6 albume afișat pe profil și să le vadă pe ale altora. Profilurile includ posibilitatea personalizării pozei de profil, vizionarea listei de albume ascultate, pe cea de albume ce urmează să fie ascultate, afișarea topului de albume și gestionarea listelor de urmăritori și urmăriți.

3 STUDIU DE PIAȚĂ

3.1 Produse similare

3.1.1 Spotify

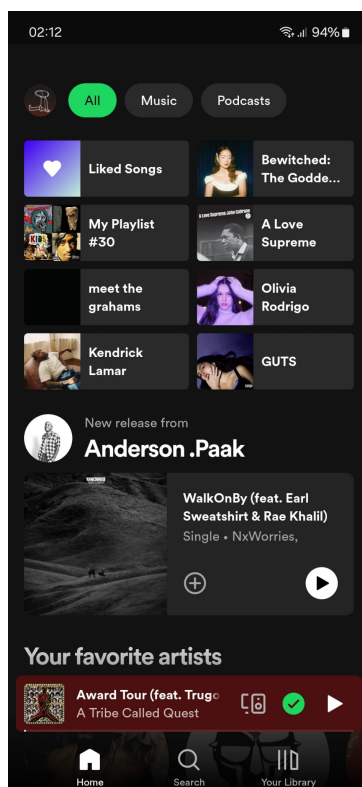


Figura 5: Spotify

”Pe piața serviciilor de streaming muzical, mai multe platforme concurează pentru abonați. În frunte, începând cu 2022, se află Spotify cu o cotă de piață globală de 30,6%, urmat de Apple Music cu 13,7%, Tencent Music cu 13,4% și Amazon Music cu 13,3%.” [8]

Spotify este, fără îndoială, liderul global în industria de streaming muzical, având peste 615 milioane de utilizatori, dintre care 239 de milioane sunt plătitori de abonamente.

Popularitatea Spotify se datorează, în primul rând, catalogului său imens de peste 100 de milioane de melodii și 2.6 milioane de podcasturi, care acoperă aproape orice gen și artist. Un alt plus major al acestei aplicații este reprezentat de recomandările personalizate, care oferă playlist-uri adaptate precum „Discover Weekly” și „Release Radar” bazate pe istoricul fiecărui utilizator.

Aplicația dispune de o interfață intuitivă și foarte ușor de utilizat, care își îndeplinește cu

desăvârșire rolul de a ușura navigarea între albume și artiști, de a crea și partaja playlist-uri și de a oferi utilizatorilor un spațiu familiar în care să își poată desfășura activitatea cu plăcere. Spotify se evidențiază și prin partea sa socială, permițând utilizatorilor să își caute prietenii și să îi adauge pentru a vedea activități precum melodia ascultată în prezent sau playlist-urile lor. Strategiile de marketing și colaborările acestei aplicații cu alte entități sunt lucrurile care o mențin la un standard atât de înalt și care îi asigură o relevanță în continuă creștere. Cu toate acestea, Spotify promovează un consum rapid și superficial de muzică, suprasolicitând utilizatorii pentru a maximiza profitul din play-uri. Aceștia sunt constant expuși la playlist-uri și recomandări automate, ceea ce poate diminua timpul și motivația necesare pentru o explorare și o apreciere profundă a albumelor complete. Personalizarea profilului utilizatorului pe Spotify este limitată. În timp ce utilizatorii pot crea și partaja playlist-uri, posibilitățile de personalizare a profilului lor pentru a reflecta pe deplin gusturile muzicale sunt reduse. Acest lucru face dificilă afișarea preferințelor muzicale într-un mod vizual distinct și personalizat, rezultând în foarte puține prietenii din viața de zi cu zi transpuse în prietenii pe Spotify, în ciuda popularității aplicației.

3.1.2 Instagram

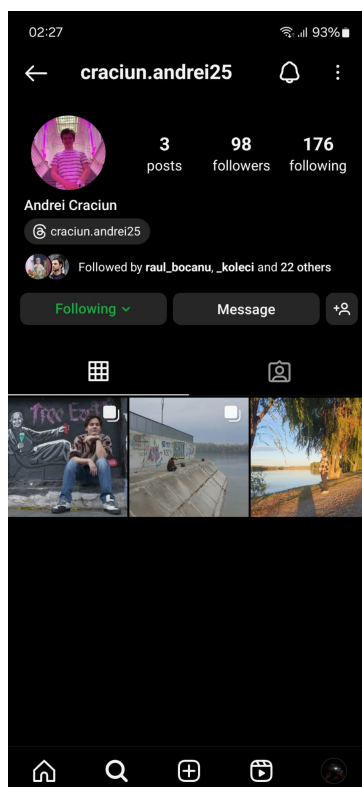


Figura 6: Instagram

La celălalt capăt al spectrului, se află Instagram, care excelează în transformarea prieteniiilor reale în conexiuni online semnificative. Autenticitatea postărilor și posibilitățile de a persona-

liza profilul astfel încât să reflecte identitatea și interesele fiecăruia sunt aspecte esențiale care fac Instagram atât de popular. Platforma oferă utilizatorilor un mediu în care își pot menține și extinde cercurile de prieteni, oferindu-le sentimentul că merită să împărtășească momente și să urmărească activitatea celor apropiați.

Pe lângă toate utilitățile practice, precum partajarea de fotografii, videoclipuri și story-uri, Instagram primează prin faptul că le oferă oamenilor un spațiu în care își pot construi o identitate online care reflectă cine sunt ei cu adevărat. Prin aceste utilități, aplicația le oferă utilizatorilor oportunitatea de a se exprima liber, de a își prezenta experiențele și de a își comunica opiniile despre viață.

3.1.3 Letterboxd

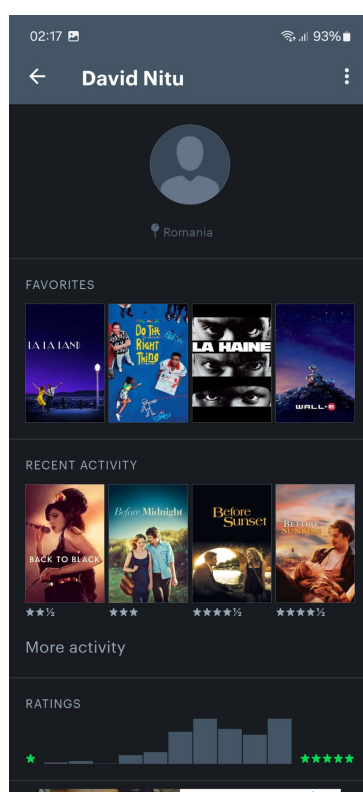


Figura 7: Letterboxd

Letterboxd este o aplicație extraordinară care oferă un spațiu interactiv pentru pasionații de filme, unde aceștia își pot exprima opiniile, pot descoperi recomandări noi și pot interacționa cu alți utilizatori. Ea îmbină aspectul social al lui Instagram, dar la o scară mult mai mică, cu credibilitatea unui site recunoscut precum IMDb. Rezultatul este o aplicație cross-platform, cu o interfață prietenoasă, care le permite utilizatorilor să creeze liste de filme pe care doresc să le urmărească în viitor (watchlist), să acorde ratinguri și recenzii filmelor, și să își personalizeze profilurile pentru a reflecta preferințele lor cinematografice.

Prin crearea acestui mediu confortabil pentru utilizatori, Letterboxd devine o comunitate foarte

strânsă în care oamenii se simt liberi și apreciați, pentru că interacționează cu alți utilizatori care împărtășesc aceeași pasiune cu ei.

”Letterboxd are o funcție de prieteni care îți permite să urmărești persoane și să vezi activitățile lor recente. Pe fiecare pagină de film, aplicația îți arată dacă prietenii tăi au văzut acel film și ce recenzii au lăsat. Există și o pagină principală unde sunt afișate filmele populare ale săptămânii, precum și filmele pe care le-au vizionat prietenii tăi.

„Cred că Letterboxd este un mod frumos de a conecta oamenii care sunt interesați de filme și vor să vadă ce au vizionat alții,” a spus Tucker Bernon, un pasionat de filme.

Îmi place să mă uit la pagina principală pentru a vedea dacă prietenii mei au vizionat filme bune recent, deoarece recenziile lor sunt de obicei amuzante și interesante de citit, și mă ajută să decid dacă vreau să vizionez și eu acel film.” [4]

3.1.4 Rate Your Music

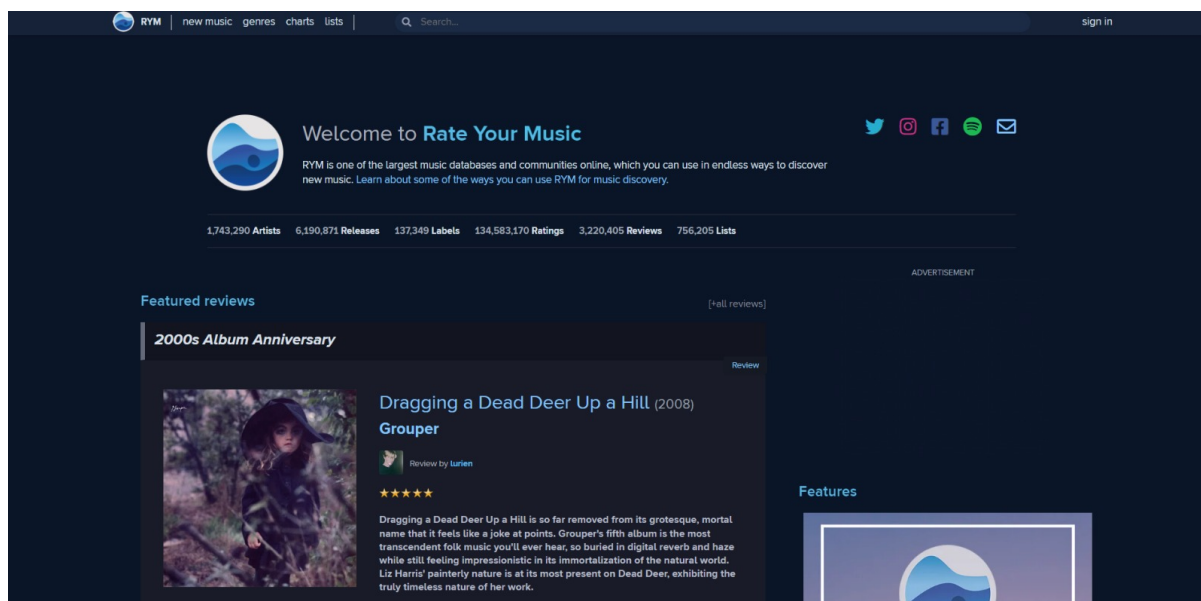


Figura 8: Rate Your Music

Rate Your Music (RYM) este unul dintre cele mai mari site-uri de critică muzicală, însă rămâne destul de necunoscut, cu doar aproximativ 729,000 de utilizatori. Precum Letterboxd, el oferă utilizatorilor posibilitatea de a evalua albume, de a vedea informații despre acestea și despre artiști, de a crea liste și de a face parte dintr-o comunitate de oameni cu pasiuni comune. RYM vine doar sub forma unei aplicații web, cu o interfață destul de robustă și limitată, și cu o personalizare minimală a profilului. Astfel, pentru a cunoaște gusturile muzicale ale unui utilizator, este nevoie de multă energie consumată. Lipsa competenței sociale și a atractivității pentru utilizatori diminuează și aspectul familiarității acestui site, făcându-l să pară doar un blog neinteresant, în ciuda conținutului profund.

3.1.5 Album Of The Year

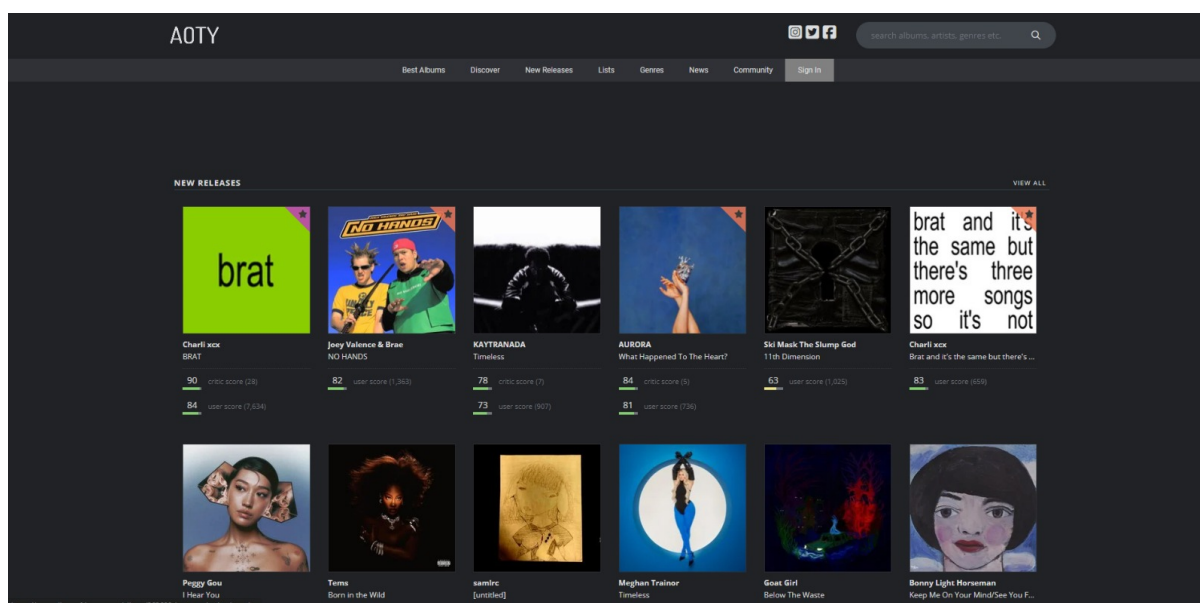


Figura 9: Album Of The Year

Album Of The Year (AOTY) este competitorul direct al lui Rate Your Music (RYM). Chiar dacă este inferior ca număr de utilizatori, când un pasionat de muzică caută o platformă pe care să-și petreacă timpul, una dintre ele este de obicei soluția. Problema este că aceste site-uri destul de obscure ajung doar la pasionații desăvârșiți de muzică, din cauza accesibilității lor scăzute. Având doar pagini web cu o interfață greoaie și veche, ele tind să sperie oamenii care abia încep să asculte muzică și caută un loc în care să vadă opiniile prietenilor lor sau ale altor utilizatori. Lipsa unei aplicații mobile și a personalizării reale prin care utilizatorii să-și exprime gusturile fac aceste platforme să fie învechite și sunt motivele principale pentru numărul lor atât de scăzut de utilizatori. Muzica este unul dintre cele mai discutate și dezbătute subiecte și cu siguranță oamenii își doresc o platformă prietenoasă pe care să facă asta.

3.2 Tehnologii Alternative

Pentru a alege arhitectura dorită, este nevoie de o analiză completă a tuturor variantelor. Acest subcapitol va prezenta pe scurt cele mai relevante tehnologii pentru realizarea unei aplicații de acest tip. Aceste tehnologii nu au fost totuși alese pentru Sleeve, ci doar ajută la înțelegerea opțiunilor curente de pe piață, pentru a putea explica de ce au fost alese tehnologiile prezentate în capitolul "Soluția Propusă", în detrimentul acestora.

3.2.1 Firebase

Firebase este o platformă completă de dezvoltare a aplicațiilor mobile și web, dezvoltată de Google. Este foarte folosită și înțelegerea modului său de funcționare este esențială în dezvoltarea unei aplicații care folosește un astfel de serviciu, lucru explicat foarte bine în articolul "Application of Firebase in Android App Development-A Study" [9]. Ea oferă o gamă largă de servicii, precum: bază de date în timp real, autentificare, găzduire, stocare în cloud și altele, toate integrate într-o singură platformă. Firebase face dezvoltarea backend-ului mult mai ușoară cu ajutorul SDK-urilor și API-urilor, permițând programatorilor să dezvolte aplicații calitative mai rapid. Popularitatea sa enormă se datorează capacităților sale de bază de date în timp real și a posibilității de a ține o bază de date fără server prin serviciul lor de cloud. Pe lângă un back-end dezvoltat în totalitate de programator, Firebase este cam cea mai populară opțiune, și pe bună dreptate: este de încredere, ușor de utilizat, intuitivă și completă. Însă pentru unii programatori poate fi o metodă puțin învechită care nu oferă foarte multe posibilități de dezvoltare în viitor. Constant apar platforme noi la fel de sau chiar mai performante decât Firebase, însă ele sunt incapabile să concureze cu Firebase în popularitate prin prisma faptului că Firebase este dezvoltată de un gigant tehnologic precum Google.

3.2.2 Kotlin

Kotlin este un limbaj de programare dezvoltat de JetBrains și lansat în 2011. În 2019, Google a anunțat că Kotlin este limbajul preferat pentru dezvoltarea aplicațiilor Android [10], depășind Java. În prezent, Kotlin are integrare cu Android Studio - IDE-ul oficial pentru dezvoltarea aplicațiilor Android. Datorită acestei strânse legături cu Android, Kotlin este predominant utilizat pentru dezvoltarea aplicațiilor native pentru această platformă. Astfel, pentru programatorii care doresc să dezvolte aplicații cross-platform într-un singur limbaj, Kotlin nu este printre opțiunile principale. Cu toate acestea, el are o rată de utilizare foarte mare în aplicațiile existente pe Android.

3.2.3 Swift

Swift este un limbaj de programare dezvoltat de Apple, lansat în 2014, care dispune de o sintaxă concisă, dar foarte puternică. Este folosit exclusiv pentru dezvoltarea aplicațiilor destinate ecosistemului Apple. Viteza sa incredibilă, scalabilitatea și managementul de memorie automat sunt doar câteva motive pentru care Swift a schimbat complet modul de dezvoltare a aplicațiilor iOS.

Una dintre problemele pe care le întâmpină un dezvoltator care folosește Swift este lipsa de compatibilitate cu versiunile vechi, ceea ce face rescrierea codului pentru o versiune nouă să devină o necesitate. De asemenea, Swift este un limbaj nativ, astfel încât o aplicație mobilă

menită să fie folosită și pe iOS și pe Android necesită două dezvoltări alternative.

"Încă de la introducerea sa în 2014, Swift a câștigat o popularitate imensă printre dezvoltatori, și nu fără motiv. Datorită sintaxei concise, funcțiilor puternice și integrării perfecte cu ecosistemul Apple, Swift a făcut dezvoltarea de aplicații iOS mai rapidă, mai eficientă și mai plăcută ca niciodată. Potrivit unor statistici recente, Swift este acum limbajul cel mai folosit pentru dezvoltarea de aplicații iOS, cu peste 85% dintre dezvoltatori care îl folosesc ca limbaj principal." [1]

3.2.4 Flutter

Flutter este un cadru de dezvoltare open-source dezvoltat de Google și lansat în 2017, fiind în prezent unul dintre cele mai folosite pentru dezvoltarea aplicațiilor mobile cross-platform. El se bazează pe limbajul Dart și vine cu un set propriu de componente UI, care oferă un control foarte bun asupra aspectului și comportamentului aplicației. Datorită acestor componente de sine stătătoare, aplicațiile dezvoltate în Flutter nu sunt afectate de actualizările de software, spre deosebire de cele dezvoltate în React Native, care doar transformă componente JavaScript în componente proprii. Un alt mare plus al acestei platforme de dezvoltare este performanța. Flutter folosește Skia pentru randarea grafică, care este mai eficientă decât metoda de adaptare a interfeței native a React Native-ului 5.3.1.

Ceea ce poate fi considerat un minus al acestei platforme este dificultatea de învățare. Fiind un limbaj complet diferit, spre deosebire de React Native care se bazează pe JavaScript, poate crea probleme programatorilor care nu au mai lucrat cu așa ceva. Un alt aspect poate fi lipsa de experiență și suport comunitar comparativ cu alte platforme mai consacrate, chiar dacă statisticile arată ca Flutter a ajuns chiar să depășească React Native în popularitate în anul 2022, aspect prezentat în figura 10.

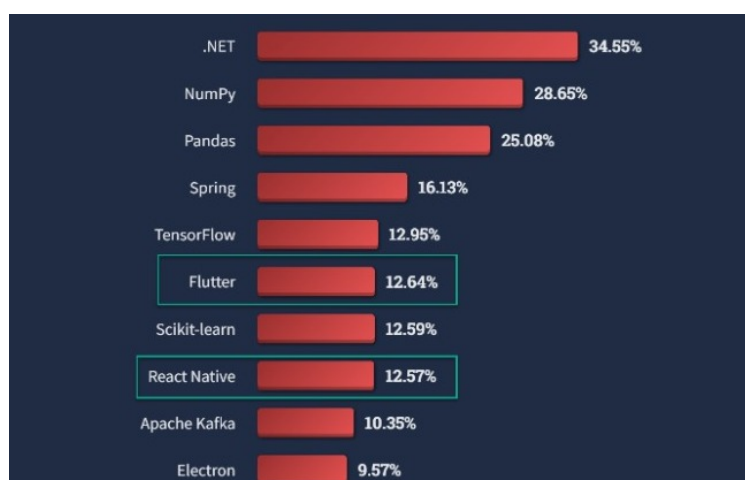


Figura 10: 2022 Cross-Platform Popularity [7]

4 SOLUȚIA PROPUȘĂ

Ca o rezolvare la cerințele enumerate și analizate mai sus, Sleeve Music Library dorește să ofere acest spațiu cu o interfață prietenoasă în care oamenii să se simtă liberi să își exprime opiniile și să le vadă pe cele ale celorlalți utilizatori.

4.1 Arhitectura Aplicației

4.1.1 Arhitectura generală a aplicației

Aplicația este dezvoltată în mediul de testare facilitat de Expo Go, care oferă compilarea ușoară și rapidă a codului pe un telefon mobil printr-o simplă scanare a unui cod QR. Frontend-ul este realizat în React Native folosind JavaScript. Tot de aici sunt realizate și cererile către backend. Sunt folosite două baze de date:

Cea oferită de **Spotify** pentru a obține informațiile despre albume și artiști, în care nu se pot introduce informații noi, ea joacă doar rolul unui set de date.

Una ținută local în **Strapi** 5.3.2, în care se stochează toate datele despre utilizatori și face autentificarea.

Ambele baze de date folosesc token-uri pentru autorizare, deci sunt securizate. Capitolul Detalii de implementare va dezvolta modul în care aceste token-uri de acces sunt obținute.

Pentru ca un album să fie adăugat la "Listened To" prima oară de un utilizator, este nevoie de o interacțiune între frontend și ambele baze de date, exemplificată în figura 11. Mai întâi, pentru a obține informațiile necesare despre album, blocul Client Frontend, care reprezintă componenta scrisă în React Native cu care interacționează direct utilizatorul, trimite cererea GET către blocul Spotify API Backend, care procesează cererea și caută informația cerută în blocul Spotify Database, care reprezintă setul de date despre albume oferit de Spotify. Baza de date întoarce un răspuns către API-ul de la Spotify, care este apoi întors sub formă de JSON (JavaScript Object Notation), ce conține toate informațiile despre album, precum titlul, artistul căruia îi aparține, data de lansare; către client.

Clientul parsează apoi acest JSON și separă informațiile de care are nevoie (în acest exemplu titlul, id-ul unic al albumului, numele artistului, id-ul unic al artistului, URL-ul către poza de copertă a albumului și data de lansare) și le trimite, alături de informații suplimentare precum id-ul utilizatorului care face cererea, către backend-ul oferit de Strapi sub forma unui request REST de tip POST. După primirea cererii, backend-ul inserează în baza de date din Strapi informațiile necesare, și întoarce rezultatul sub formă de JSON clientului, care afișează informațiile cu modificările făcute.

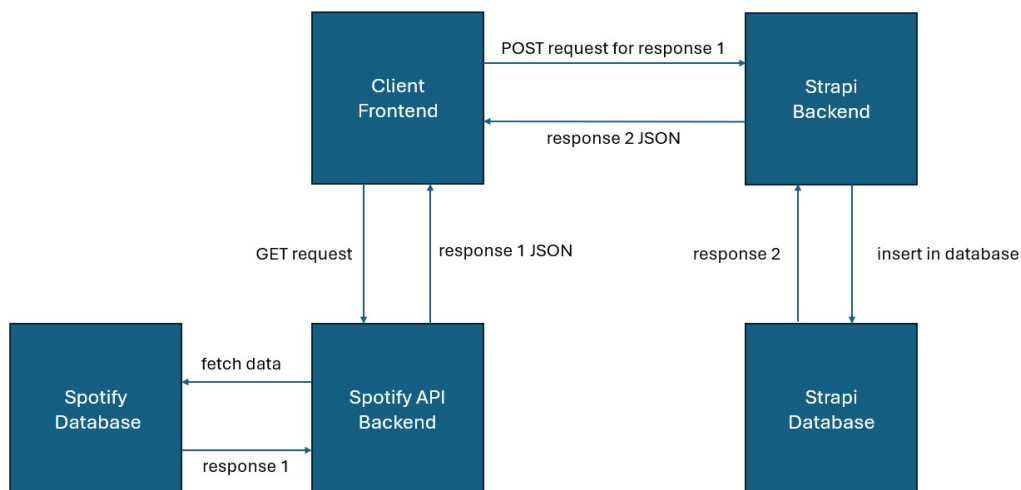


Figura 11: Cerere POST

Clientul poate totuși comunica cu o singură bază de date, ceea ce îmbunătățește timpul de răspuns al aplicației cu mult. Acest caz este exemplificat în figura 12, în care clientul dorește să actualizeze o intrare deja existentă sau doar să primească informații despre ea. Aici baza de date de la Spotify nu trebuie accesată pentru că aceleași informații sunt deja salvate în cea din Strapi.

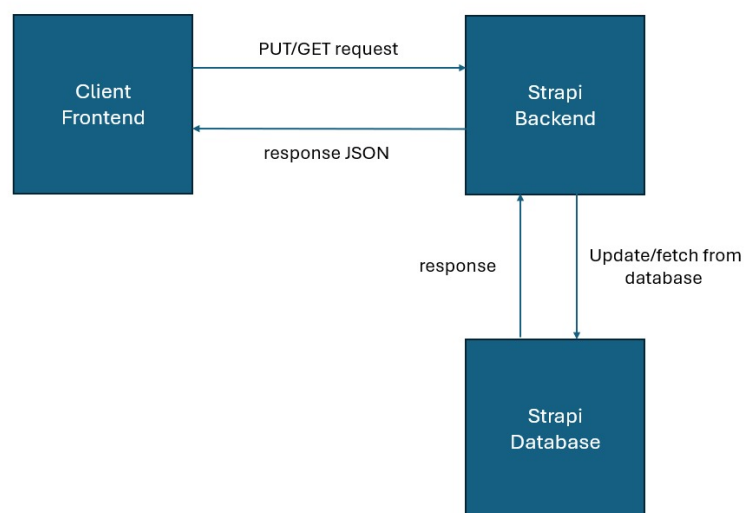


Figura 12: Cerere PUT/GET către Strapi

Dacă utilizatorul vrea doar să vadă date despre un album sau despre un artist, fără ca el să îi dea vreo notă albumului sau să-l adauge pe vreo listă, este nevoie exclusiv de o cerere de tip GET către Spotify API, așa cum este prezentat în figura 13. Astfel, nu este nevoie să interacționeze deloc cu baza de date ținută local.

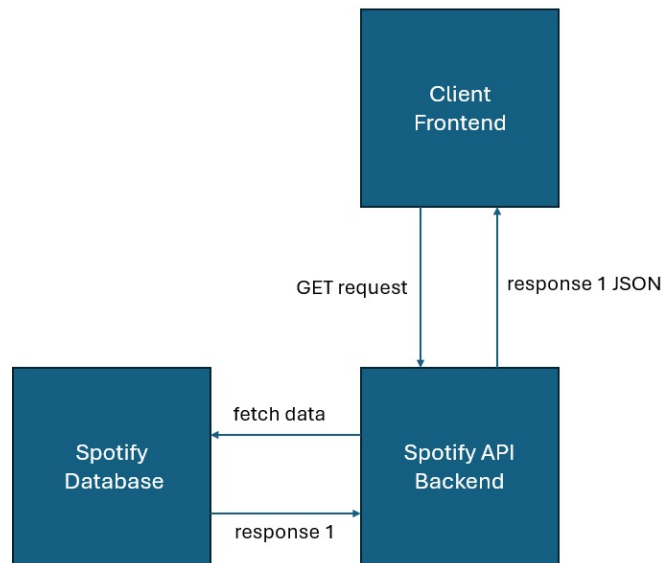


Figura 13: Cerere GET către Spotify

4.1.2 Baza de date din Strapi

În figura 14 se poate observa diagrama relațională a bazei de date a aplicației. Principalele 3 tabele sunt:

User, care ține date despre un utilizator, cum ar fi: ID-ul unic folosit pentru identificare, username-ul, email-ul, numele, prenumele și parola, introduse sau primite la autentificare, createdBy, updatedBy care aparțin adminului care a creat user-ul respectiv, rolul și încă câteva câmpuri necesare pentru administrarea bazei de date. În plus, această tabelă are un câmp pentru poza de profil și 2 relații de tip many-to-many: following și followers, care permit unui utilizator să urmărească mai mulți utilizatori și să aibă mai mulți urmăritori. Ele sunt practic liste de id-uri, care fac referință tot către tabela Users. Ea are și 2 câmpuri relaționale de tip one-to-many: User's Albums și Lists, care fac legătura cu celelalte 2 tabele și permit unui utilizator să aibă mai multe liste și albume salvate.

User's Albums, unde este păstrat tot despre un album salvat de un user a cărui referință este făcută prin userID. Datele despre album: id, titlul albumului, url-ul ce conține link-ul către poza albumului, data de lansare, id-ul și numele artistului; sunt cele oferite de Spotify API, luate și postate aici. User-ul poate adăuga un album la lista sa de "Listened To", și astfel câmpul boolean listenedTo devine adevărat. La fel și pentru lista de "Future Listens", doar că aici câmpul wantToListenTo devine adevărat. Nota este câmpul în care se salvează nota dată de user. Ultimul câmp din această tabelă este cel de favorite, care face posibilă existența unui top cu 6 albume preferate ale user-ului. Astfel în acest câmp va fi trecut un număr de

la 1 la 6 care va arăta pe ce loc este albumul respectiv. Acest câmp este unic, întrucât două albume nu pot avea același loc în top în același timp. Albumele salvate încep cu acest câmp null, și doar 6 pot avea un număr simultan.

Lists, în care sunt stocate listele create de user. Ele au un nume, un id unic, id-ul și username-ul user-ului asociat. În plus ele au un câmp relațional de tip one-to-many cu User's Albums, care permite adăugarea mai multor albume salvate de user.

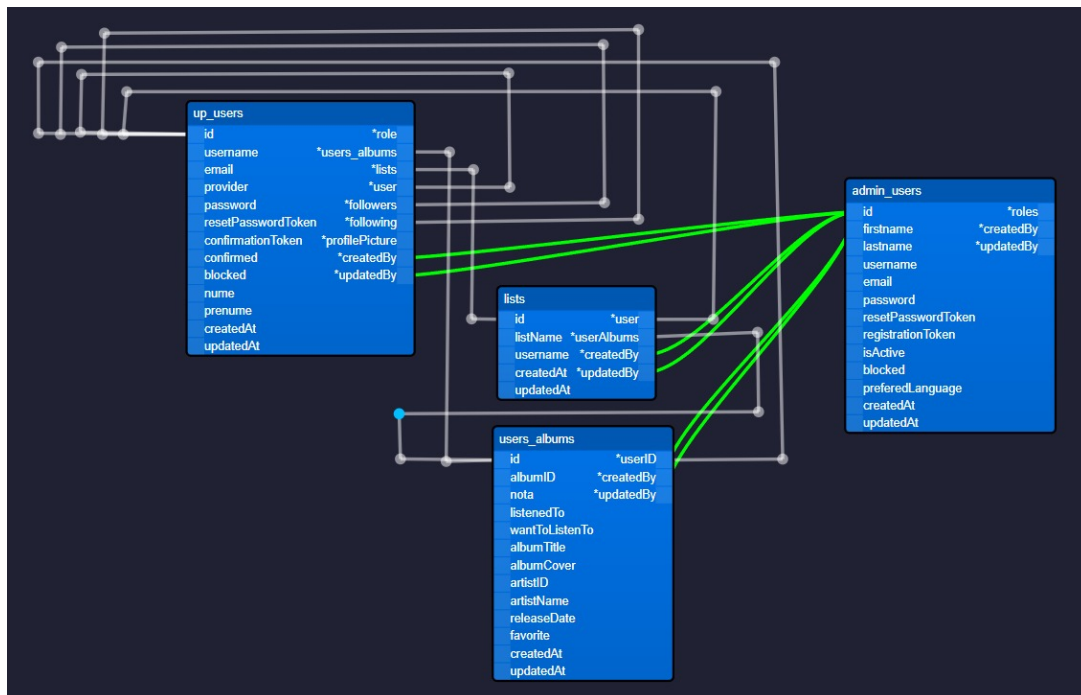


Figura 14: Diagramă a Bazei de Date

Strapi se ocupă automat de backend și păstrează informații adiționale pentru administrare. Autentificarea este făcută pe baza de JWT tokens.

Setul de date folosit de Sleeve este cel oferit de Spotify. Această decizie a fost făcută pentru a oferi utilizatorilor o bibliotecă cât se poate de completă. Astfel în aplicație se poate căuta și interacționa cu aproape orice album existent. La request-ul de GET, API-ul de la Spotify întoarce un JSON cu date despre album sau artist, depinzând de tipul de căutare.

Sleeve se folosește în principal de ID-ul unic al albumului sau al artistului, cu ajutorul căruia face cererea de GET pentru celelalte date. Un utilizator logat poate căuta albume și artiști după nume, și poate intra pe pagina lor care va fi randată automat doar cu ajutorul ID-ului, fără ca tabela de User's Albums să fie populată cu vreo informație.

Odată ce utilizatorul face o acțiune pe pagina albumului, datele enumerate în descrierea listei User's Albums sunt salvate în baza de date ținută în Strapi, producându-se astfel o unificare între cele două baze de date. Toate paginile pot afișa informațiile corect indiferent de la ce bază de date este primit ID-ul. Datele suplimentare precum titlul sau poza de copertă sunt

salvate pentru a ușura afișarea tuturor paginilor, întrucât doar pe paginile de search, cea a albumelor și cea a artiștilor sunt făcute cereri la API-ul de la Spotify. Acest lucru asigură că nu va fi depășit niciodată rate limit-ul acestui serviciu.

4.1.3 Funcționalitățile aplicației

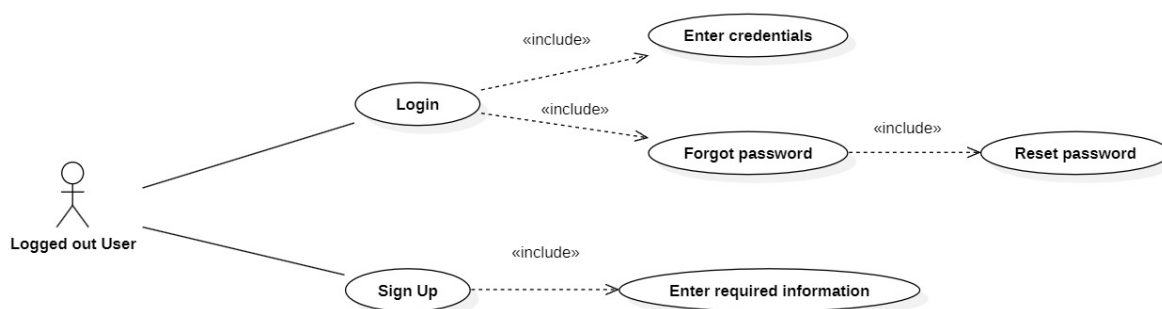


Figura 15: UML pentru utilizatorul neautentificat

Când un utilizator accesează aplicația pentru prima dată sau după deconectare, el este considerat utilizator neautentificat. În figura 15 este prezentată diagrama UML pentru această stare. El poate introduce email-ul sau username-ul în primul câmp și parola în al doilea câmp pentru a se autentifica. Dacă credențialele sunt corecte, el va primi permisiunea de a accesa aplicația. În cazul în care și-a uitat parola, el poate solicita un cod pe email, pe care îl va introduce ulterior în aplicație, și pe baza căruia i se va oferi permisiunea de a-și actualiza parola, fără a i se cere pe cea veche.

Dacă utilizatorul nu are cont, el își poate crea unul, apăsând pe textul care îl va redirecționa către pagina de Sign Up, unde își va introduce datele personale, username-ul și parola dorite. Dacă toate câmpurile sunt completate corespunzător, contul său va fi creat și va primi acces în aplicație.

Odată autentificat, utilizatorul primește acces complet la aplicație și va rămâne autentificat chiar și după închiderea și repornirea aplicației. Figura 16 prezintă diagrama UML pentru această stare. Aplicația oferă navigație prin rutare, care îl poate duce pe utilizator de la o pagină la alta în mod obișnuit, iar la apăsarea butonului de back, el este dus pe pagina precedentă, iar navigarea prin tab-uri afișate în partea de jos a paginii facilitează o navigare ușoară și intuitivă între paginile principale ale aplicației.



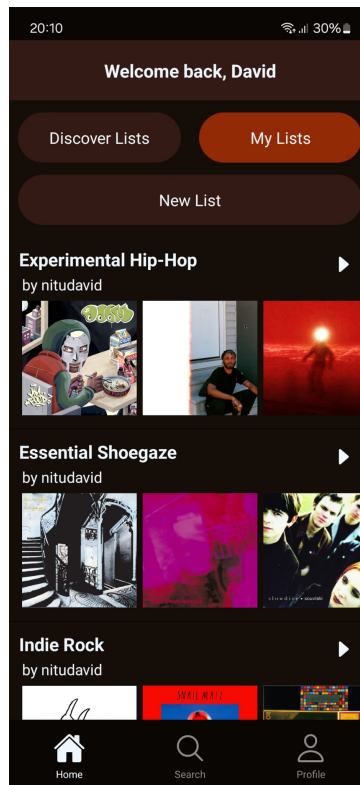
Figura 16: UML pentru utilizatorul autentificat

Primul tab este **Home**, unde utilizatorul este întâmpinat cu un mesaj de bun venit personalizat cu numele său, pe care l-a introdus în formularul de autentificare. Pe această pagină, el poate vedea listele create de el sau poate descoperi liste create de alți utilizatori.

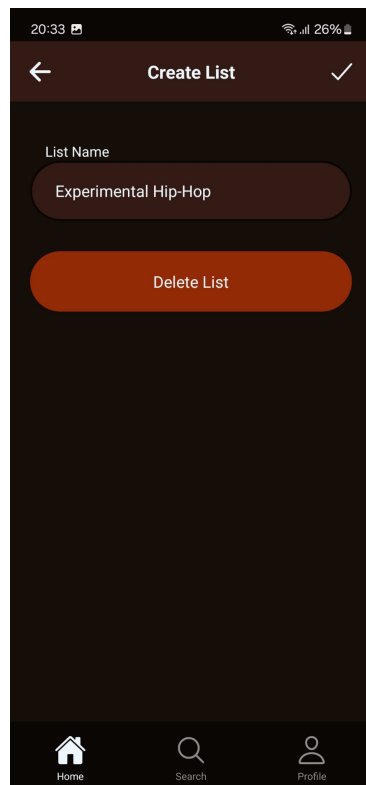
O listă conține numele listei, username-ul utilizatorului care a creat-o și o listă scroll-abila orizontală cu toate albumele pe care le conține, pe care se poate apăsa pentru a fi redirectionat către pagina albumului respectiv. Selectarea tipului de liste afișate se face prin apăsarea unuia dintre cele două butoane de sub header-ul paginii: Discover Lists și My Lists. Dacă tipul selectat este My Lists, o iconiță în formă de triunghi apare în partea dreaptă a fiecărei liste, ceea ce semnifică faptul că el poate apăsa pe orice listă pentru a o edita:

- îi poate schimba numele;
- poate adăuga albume apăsând pe butonul cu un plus, care îl va redirectiona în lista sa de Listened To, unde dacă apasă pe un album, acesta va fi adăugat direct în acea listă, iar utilizatorul va fi întors pe pagina anterioară pentru a putea continua să modifice conținutul listei în continuare;
- poate șterge albume din ea, apăsând lung pe poza de copertă a oricărui album;
- poate șterge lista cu totul din baza de date.

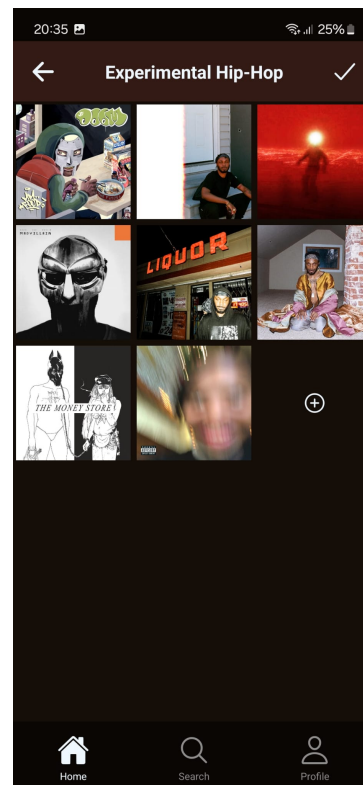
Dacă el vizualizează listele afișate de alți utilizatori, poate apăsa pe username-ul de sub numele listei pentru a fi redirectionat către pagina utilizatorului care a creat lista. În plus pe această pagină există și un buton de New List care îi permite să creeze o listă nouă. Acest proces este foarte similar cu cel de editare a unei liste, doar că aici nu are opțiunea de a o șterge pentru că lista nu există încă. În figura 17 sunt prezentate paginile de home, și cele pentru editarea listelor.



(a) Home Page



(b) Edit List Name



(c) Edit List Albums

Figura 17: Home Tab și Paginile pentru Edit List

Cel de-al doilea tab este **Search**, unde căutarea implicită este cea de albume. Sub bara de căutare sunt 3 butoane care schimbă modul de căutare între albume, artiști și utilizatori. Totodată, se schimbă și textul implicit din bara de căutare, pentru a reprezenta corect tipul corespunzător butonului selectat. Căutarea este făcută în timp real, fără a fi nevoie ca utilizatorul să apeleze vreun buton care să proceseze.

Când bara de căutare nu mai este goală, aplicația afișează rezultatele relevante pentru parametrii căutării curente sub forma unei liste scroll-abile ce afișează imaginea pentru toate cele 3 variante de căutare, numele albumului și artistul căruia îi aparține pentru albume, numele pentru artiști și username-ul pentru utilizatori. Când utilizatorul apasă pe unul dintre rezultate este redirecționat pe pagina albumului, artistului sau a utilizatorului care corespunde rezultatului apăsător. Figurile 18 și 19 prezintă un exemplu de căutare de albume și de utilizatori. Primele două tipuri de căutare sunt făcute prin request-uri REST de tip GET la API-ul de la Spotify, iar cel pentru utilizatori este făcut printr-un request REST de tip GET la baza de date din Strapi. Acest lucru face o separare corectă și fluidă a celor două baze de date și permite căutarea oricărui album și artist, și nu doar a celor salvați deja în baza de date din Strapi.

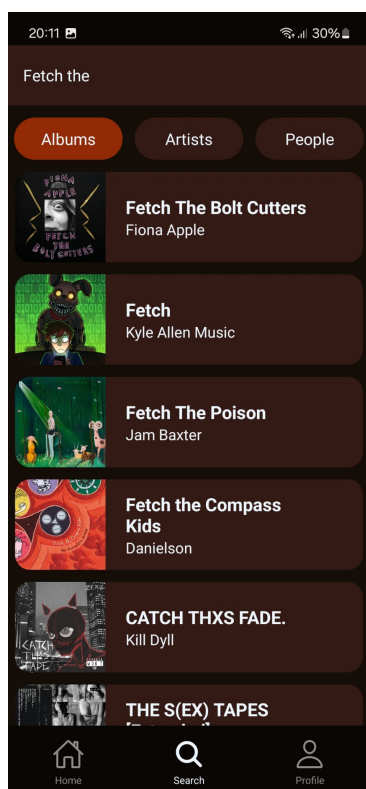


Figura 18: Search Album

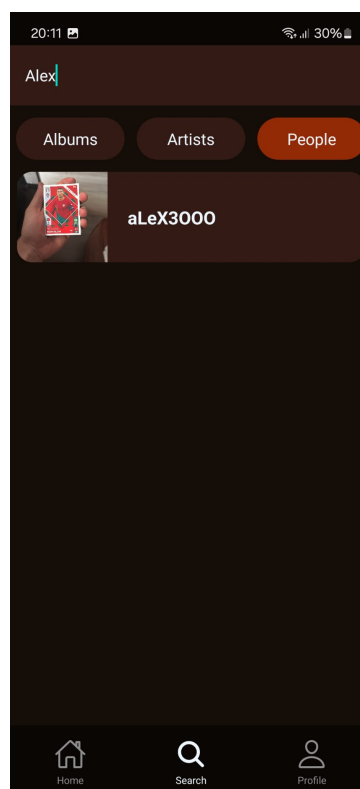


Figura 19: Search User

Cel de-al treilea tab este **Profile**, unde se află toată identitatea utilizatorului, prezentat în figura 20. El are o poză de profil implicită, pe care dacă apasă, o poate schimba cu una din galeria telefonului.

Aici sunt 4 butoane care îl redirecționează pe alte pagini:

Listened To, unde îi apare lista cu albumele care au listenedTo adevărat în baza de date. El poate schimba tipul de afișare între listă și grid. Totodată, poate apăsa pe butonul de sortare, pentru a alege ordinea în care sunt afișate albumele. Opțiunile sunt: după nota acordată, după nume, după numele artistului, după data lansării sau după data adăugării în această listă. Toate aceste opțiuni pot fi selectate ascendent sau descendent. Dacă el apasă pe oricare dintre albume va fi redirecționat pe pagina sa.

Future Listens, care este la fel ca Listened To, dar câmpul wantToListenTo trebuie să fie adevărat în loc de cel listenedTo.

Following, unde poate vedea ce utilizatori urmărește.

Followers, unde poate vedea cine îl urmărește.

În plus, pe pagina cu profilul sunt afișate albumele preferate ale utilizatorului. Implicit sunt 6 casete goale, care odată apăsate îl duc pe utilizator în lista sa de Listened To, iar atunci când el apasă pe un album, acel album este pus în caseta butonului apăsător la început. Pentru a scoate un album de pe una dintre cele 6 poziții, utilizatorul poate ține apăsat lung pe el.

În plus, utilizatorul își poate face log out tot de pe această pagină.

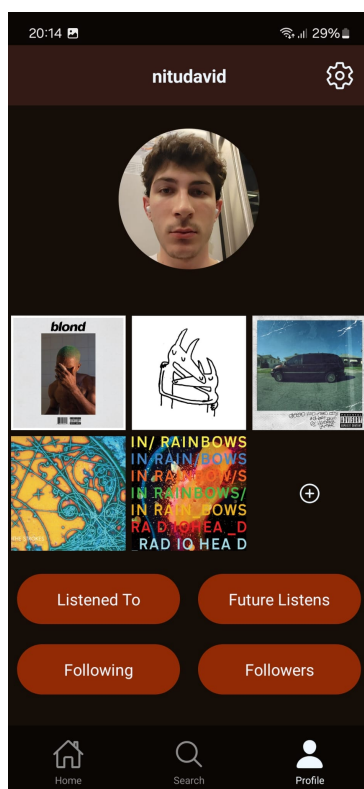


Figura 20: Profile Page

Pe pagina unui album, prezentată în figura 21, apare numele albumului, poza sa de coperta, plasată peste o versiune foarte blurată a acesteia, la care se aplică un gradient pentru a oferi o experiență imersivă utilizatorilor când accesează această pagină; artistul cărui îi corespunde albumul respectiv și sistemul de interacțiune cu el: butonul de Listened To, cel de Future Listen și 5 discuri care reprezintă sistemul de notare. Acestea sunt inițial goale, dar în urma apăsării pe unul dintre ele, toate cele dinaintea lui devin pline, semnificând nota acordată. Numele artistului poate fi apăsat pentru a-l redirecționa pe utilizator pe pagina artistului. Pagina artistului, prezentată în figura 22, conține numele acestuia, poza de profil și o listă cu toate albumele care îi aparțin, pe care utilizatorul le poate apăsa pentru a fi redirecționat către paginile lor.

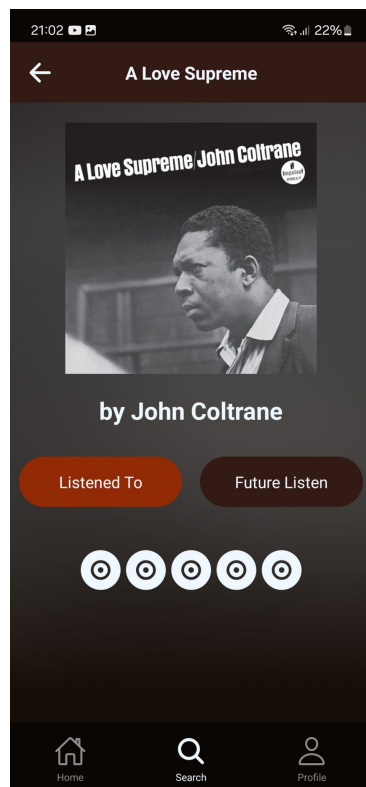


Figura 21: Album Page

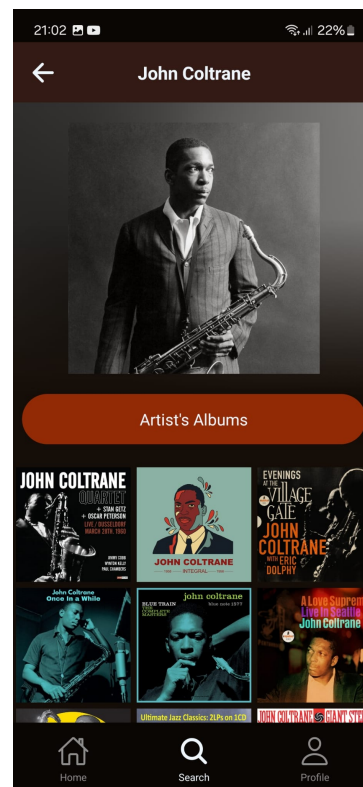


Figura 22: Artist Page

Pagina altui utilizator conține numele său, poza de profil, topul celor 6 albume preferate ale sale, lista cu următorii săi și cea cu utilizatorii pe care îi urmărește. În plus, are un buton de urmărire sau de scoatere de la urmărire.

5 DETALII DE IMPLEMENTARE

5.1 Conectarea la bazele de date

5.1.1 Strapi

```
1 export const BASE = "http://localhost:1338";
2 const apiClient = axios.create({
3   baseURL: `${BASE}/api`,
4 });
5 apiClient.interceptors.request.use(async (config) => {
6   let jwt = await AsyncStorage.getItem("jwtToken");
7   if (config.url === "/auth/local") {
8     jwt = null;
9   }
10  if (jwt) {
11    config.headers["Authorization"] = `Bearer ${jwt}`;
12  }
13  return config;
14 });
15 export default apiClient;
```

Listing 5.1: JSON Web Token

Serverul Strapi este gazduit local, deci baseURL-ul pentru cererile REST este format din IP-ul local al serverului, urmat de portul pe care rulează serverul, și /api pentru a simplifica sintaxa cererilor.

Fiecare cerere necesită un token Bearer. Pentru această bază de date se utilizează JWT (JSON Web Token), care, odată obținut, este inclus în antetul de autorizare al cererii pentru a permite utilizatorului cu JWT-ul curent să facă cereri.

În codul de mai jos este prezentat un exemplu de cerere de tip REST în Strapi. Această cerere este puțin mai complexă deoarece include și filtre și sortare. În dezvoltare au existat câteva probleme cu sintaxa pentru aceste două câmpuri, deoarece Strapi pare să genereze JSON-uri destul de diferite pentru fiecare tabelă.

Într-o cerere este necesară exprimarea tipului de cerere și tabela din care urmează să fie făcută cererea.

```
1 export const GET_USER_ALBUM_REVIEW = async (userID, filters = {}, sort
  = []) => {
2   try {
3     const sortQueryParam = Array.isArray(sort) && sort.length > 0 ?
```

```

    sort.join(',') : undefined;
4     const response = await apiClient.get('users-albums', {
5       params: {
6         filters: {
7           userID: userID,
8           ...filters
9         },
10        populate: '*',
11        pagination: {
12          pageSize: 100,
13        },
14        sort: sortQueryParam,
15      }
16    });
17    return response.data;
18  } catch (error) {
19    throw error;
20  }
21 };

```

Listing 5.2: Strapi GET

5.1.2 Spotify API

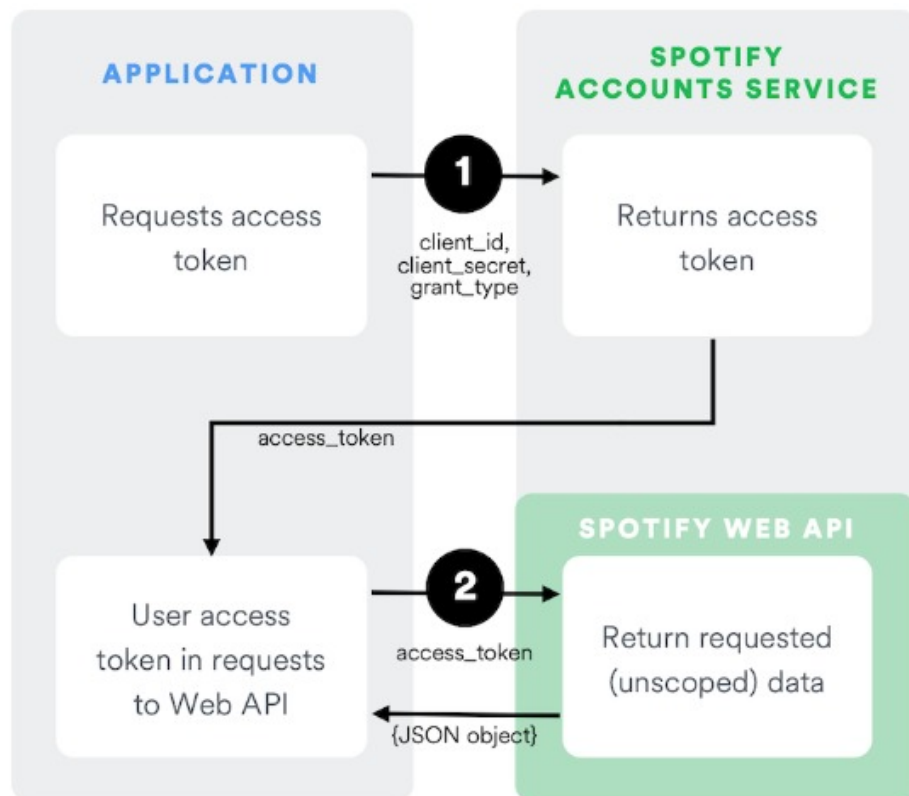


Figura 23: SpotifyAPI [13]

Conectarea la API-ul oferit de cei de la Spotify se poate face prin mai multe modalități, fiecare cu avantajele și dezavantajele sale. Cele mai complexe oferă legătura fiecărui utilizator din aplicația dezvoltată cu contul lor din Spotify, redarea melodiilor în aplicația în care este folosit API-ul și alte beneficii. Însă procesul de conectare are mult mai mulți pași care nu au avut sens pentru acest proiect. Sleeve folosește conectarea de tip "Client Credentials", prezentată în figura 23, care oferă doar date despre muzică, nu și despre utilizatori, însă este complet suficient pentru această aplicație. Obținerea autorizării este puțin mai complexă decât la Strapi, pentru că se folosesc token-uri care expiră. Astfel, este nevoie de o metodă de actualizare când acesta expiră. Acest lucru a fost rezolvat prin reținerea timpului de la primirea token-ului și reactualizarea constantă a acestuia până când ajunge la termenul de expirare, moment în care este cerut un alt token, care este trimis, ca în Strapi, în antetul de autorizare cu sintaxa "Bearer" urmat de token.

```
1 const getAccessToken = async () => {
2   const tokenUrl = 'https://accounts.spotify.com/api/token';
3   const data = {
4     grant_type: 'client_credentials',
5   };
6   const headers = {
7     'Content-Type': 'application/x-www-form-urlencoded',
8     'Authorization': 'Basic ' + Buffer.from(`${CLIENT_ID}:${CLIENT_SECRET}`).toString('base64'),
9   };
10  try {
11    const response = await axios.post(tokenUrl, new URLSearchParams(data).toString(), { headers });
12    accessToken = response.data.access_token;
13    tokenExpirationTime = Date.now() + response.data.expires_in * 1000;
14    return accessToken;
15  } catch (error) {
16    console.error('Error fetching access token:', error);
17    throw error;
18  }
19 };
20 const checkAndRefreshToken = async () => {
21   if (!accessToken || Date.now() >= tokenExpirationTime) {
22     await getAccessToken();
23   }
24 };
25 spotifyApi.interceptors.request.use(
26   async (config) => {
27     await checkAndRefreshToken();
28     config.headers.Authorization = `Bearer ${accessToken}`;
29     return config;
30   },
31   (error) => Promise.reject(error)
```

32);

Listing 5.3: Spotify Access Token

5.2 Implementări notabile

5.2.1 Search Page

```
1  const handleSearch = useCallback(  
2    debounce(async (query) => {  
3      if (query.trim() === '') {  
4        setData(null);  
5        setLoading(false);  
6        return;  
7      }  
8      setLoading(true);  
9      setError(null);  
10     try {  
11       const results = searchType === 1  
12         ? await searchArtists(query)  
13         : searchType === 0  
14           ? await searchAlbums(query)  
15           : await GET_USERS({ username: { $startsWith:  
query }, id: { $ne: myUser?.user.id } });  
16       setData(results);  
17     } catch (error) {  
18       setError('Failed to fetch data. Please try again.');
```

```
19     } finally {  
20       setLoading(false);  
21     }  
22   }, 500),  
23   [searchType]  
24 );  
25 useEffect(() => {  
26   handleSearch(searchQuery);  
27 }, [searchQuery, handleSearch]);
```

Listing 5.4: Search

Una dintre implementările mai interesante este cea pentru pagina de căutare. Inițial, era distribuită pe mai multe pagini, una pentru fiecare tip de căutare, cu valori diferite pentru bara de căutare și cu funcții diferite. În plus, era o căutare clasică care făcea o cerere doar când butonul de căutare era apăsător. După un proces de optimizare și regândire a logicii, activitatea de căutare este acum implementată mult mai eficient. Ea folosește un `searchQuery` care se actualizează cu fiecare schimbare în bara de căutare. Fiecare schimbare a acestui `searchQuery` reapelează funcția `handleSearch`, comportându-se ca un model foarte bun de căutare. În plus, cele trei tipuri de căutare sunt unificate, ele fiind ușor interschimbabile prin apăsarea unui

dintre cele 3 butoane care le poartă numele din antetul paginii. Căutarea se face pe baza flag-ului dat de apăsarea unuia dintre butoane. Primele două tipuri de căutări fac cereri în baza de date de la Spotify, pe când cel de-al treilea face în baza de date din Strapi. În plus, sunt afișate stările intermediare de încărcare și erorile care pot să apară.

Se afișează datele necesare despre obiectul căutat, iar când este apăsât, doar id-ul este pasat paginii sale, pe care se face iar fetch la celelalte date. Asta oferă o paradigmă eficientă care permite comutarea ușoară între cele două baze de date. Totodată asigură lipsa greșelilor în citirea JSON-urilor care pot fi scrise în moduri diferite.

5.2.2 Sort Page și Listened To

Pagina de sort are 2 butoane în partea superioară: Ascending și Descending, și 5 sub acestea: Rating, Title, Artist, Release date și Add date. Ele setează două variabile: sortOrder pentru primele 2, și sortField pentru restul, urmând ca acestea să fie concatenate și ajustate pentru a corespunde structurii cerute de API, de genul "id:desc". Pagina pleacă din sortarea default "id:desc", care arată albumele în ordinea adăugării lor. Ea poate fi apelată și din Listened To, care la rândul ei poate fi apelată de pe profil direct, apăsând pe o căsuță goală din cele 6 din top sau dintr-o listă, dar și din Future Listens, ceea ce face destul de grea păstrarea continuității și întoarcerea pe pagina corectă după ce sortarea e finalizată. În React Native, trecerea de la o pagină la alta nu păstrează parametrii curenți, ei trebuie pasați prin route.params la pagina următoare. Astfel, problema de mai sus a fost rezolvată prin utilizarea flag-urilor pentru pagini și pentru butonul apăsât, toate fiind trimise în pagina de sortare, care la rândul ei le trimite înapoi paginii din care a fost apelată.

```
1  const route = useRoute()
2  const sortParameter = route.params?.sortParameter || 'id:desc'
3  const pageID = 1
4  const selectFavoriteMode = route.params?.selectFavoriteMode || 0
5  const buttonID = route.params?.buttonID
6  const selectListAlbum = route.params?.selectListAlbum || 0
7  const listID = route.params?.listID
8
9  const handleSortPress = () => {
10     navigation.navigate('SortPage', { sortParameter: sortParameter,
        pageID: pageID, selectFavoriteMode: selectFavoriteMode, buttonID:
        buttonID, selectListAlbum: selectListAlbum, listID: listID});
11  };
```

Listing 5.5: Navigare

La fel se întâmplă și cu Listened To, întrucât și ea poate fi apelată din locuri diferite cu funcționalități diferite:

De pe pagina de profil, ea nu primește niciun parametru, iar funcționalitatea ei când unul

dintre albume este apăsat este să îl ducă pe user pe pagina albumului respectiv cu albumID-ul corect.

În procesul de adăugare a albumelor în liste, unde primește parametrul selectListAlbums, iar atunci când este apăsat un album, ea adaugă acel album în lista din care a fost apelată, folosind o cerere de tip PUT.

Apăsând pe unul dintre locurile din top 6 de pe profil, ea primește parametrii selectFavorite și buttonID, urmând să îi dea albumului pe care utilizatorul apasă numărul din variabila buttonID în câmpul favorite, și apoi să îl întoarcă pe utilizator pe profilul său.

Această navigare între pagini trebuie făcută cu mare atenție, și toți parametrii necesari trebuie trimiși de la o pagină la alta de fiecare dată, pentru a nu se încurca datele.

5.2.3 Follow

Pentru că aplicația se bazează în mare parte pe id-uri când se identifică proprietarul unei pagini, datorită unicității lor, pe pagina altui user, datele sale trebuie mai întâi cerute pentru a putea fi afișate, la fel ca cele ale user-ului logat. După primirea ambelor seturi de date, se verifică dacă user-ul este pe lista de following a user-ului logat, și se setează flag-ul isFollowing, care decide funcționalitatea butonului de follow/unfollow.

```
1  const handleFollow = async () => {
2    try {
3      ...
4      // Se obtine lista de urmariri a user-ului curent si cea de
      // urmaritori a user-ului tinta prin id-urile lor
5      if (isFollowing === 0) {
6        // Pentru a actualiza ambele liste, id-ul user-ului curent
        // trebuie concatenat la lista existenta.
7        const updatedFollowing = [...currentUserFollowingIDs,
        userID];
8        const updatedFollowers = [...targetUserFollowerIDs,
        myUser.user.id];
9        await UPDATE_FOLLOW(myUser.user.id, { "following":
        updatedFollowing });
10       await UPDATE_FOLLOW(userID, { "followers":
        updatedFollowers });
11       setIsFollowing(1);
12     } else {
13       const updatedFollowing = currentUserFollowingIDs.filter
        (followingID => followingID !== userID);
14       const updatedFollowers = targetUserFollowerIDs.filter(
        followerID => followerID !== myUser.user.id);
15       await UPDATE_FOLLOW(myUser.user.id, { "following":
        updatedFollowing });
16       await UPDATE_FOLLOW(userID, { "followers":
        updatedFollowers });
17     }
18   }
19 }
```

```

18     } catch (error) {
19         console.error("Error updating follow status:", error);
20     }
21 };

```

Listing 5.6: Urmarire utilizatori

5.2.4 Interacțiunea cu albumele

Sunt 3 moduri în care un user poate interacționa cu un album: să îi dea o notă, să îl adauge pe Listened To sau să îl adauge pe Future Listens. Cele trei interacționează între ele după o logică simplă:

Un album nu poate avea notă fără să fie pe lista de Listened To, deci dacă primește notă fără să fie pe nicio listă este adăugat pe Listened To. Dacă primește notă în timp ce este pe Future Listens, Future Listens devine fals și Listened To adevărat. Un album nu poate fi pe ambele liste în același timp, deci ele se exclud una pe alta. Dacă este scos de pe Listened To, nota devine null, la fel și dacă este adăugat pe Future Listens.

Înainte ca toată această interacțiune automată să se întâmple, trebuie verificat dacă albumul este deja salvat de user sub orice formă, căutându-l după id. Dacă el există deja, doar se actualizează informațiile cu o cerere de tip PUT. Dacă nu există, se face o cerere de tip POST, în care se și salvează informațiile necesare oferite de API-ul de la Spotify.

```

1     const handleWantToListenTo = async () => {
2         try {
3             if (user) {
4                 const data = {
5                     nota: null,
6                     listenedTo: false,
7                     wantToListenTo: !wantToListenTo
8                 }
9                 if (existingReviewId) {
10                     await UPDATE_ALBUM(existingReviewId, data);
11                     setRating('');
12                     setListenedTo(false);
13                 } else {
14                     const newData = {
15                         userID: user.id,
16                         albumID: albumID,
17                         albumTitle: album.name,
18                         albumCover: album.images[0]?.url,
19                         artistID: album.artists[0]?.id,
20                         artistName: album.artists[0]?.name,
21                         releaseDate: album.release_date,
22                         ...data
23                     }
24                     const response = await ADD_ALBUM_REVIEW(newData);

```

```

25         setExistingReviewId(response.data.id)
26     }
27     setListenedTo(false);
28     setWantToListenTo(prevState => !prevState);
29 }
30 } catch (error) {
31     console.error('Error updating review:', error);
32     Alert.alert('Error', 'Failed to update review.');
```

Listing 5.7: Adaugare album la Future Listens

5.3 Tehnologii Folosite

5.3.1 React Native

În detrimentul tuturor platformelor menționate mai sus, Sleeve Music Library este implementată folosind React Native cu JavaScript. Acest cadru de dezvoltare a fost dezvoltat de Meta și lansat în 2015. El permite dezvoltarea aplicațiilor mobile atât pentru iOS, cât și pentru Android. Posibilitatea de a compila același cod pe sisteme diferite este principalul punct forte al acestui cadru, deoarece reduce drastic timpul și efortul necesar pentru a dezvolta o aplicație completă. În plus, având o vechime mai mare decât majoritatea celorlalte opțiuni enumerate mai sus, comunitatea este foarte vastă și, în general, se poate găsi răspuns la aproape orice întrebare.

React Native este un interpretor de JavaScript, ceea ce îl face destul de familiar și ușor de utilizat, mai ales pentru dezvoltatorii web care fac tranziția la dezvoltarea de aplicații mobile [11]. Deși nu este un limbaj nativ, ceea ce poate avea un mic impact asupra performanței aplicațiilor, React Native oferă performanțe foarte apropiate de cele ale aplicațiilor dezvoltate în Swift 3.2.3 sau în Kotlin 3.2.2.

În plus, în dezvoltarea aplicației s-a folosit Expo, un ecosistem construit în jurul React Native, menit să simplifice procesul de creare a aplicațiilor React Native [6]. Unul dintre cele mai mari avantaje ale acestei unelte este Expo Go, o aplicație mobilă care le permite dezvoltatorilor să testeze direct pe telefonul mobil aplicația lor, prin simpla scanare a unui cod QR. Acest proces aduce foarte multă cursivitate dezvoltării și exclude nevoia de a folosi emulatoare de telefon pe calculator.

5.3.2 Strapi

Pentru backend, Sleeve folosește Strapi, în ciuda inferiorității clare față de un serviciu ca Firebase 3.2.1 când vine vorba de popularitate. Unul dintre motivele principale pentru care Strapi a fost ales ca soluția principală de backend este reprezentat de posibilitățile imense

de personalizare, ceea ce permite control complet asupra API-urilor și a structurilor de date. Astfel, în Strapi, îți poți crea cu ușurință orice fel de tabelă și tot la fel de ușor poți dezvolta funcțiile ce vor face cereri.

Strapi folosește un server local, pe când Firebase este serverless, ceea ce limitează și mai tare flexibilitatea Firebase-ului în comparație [5]. Un server ținut local oferă dezvoltatorilor libertatea de a configura și controla mediul după cum doresc. În plus, Strapi este open-source și se integrează foarte ușor cu React Native.

Strapi beneficiază și de un sistem de autentificare superior, folosind JWT (JSON Web Tokens) [2], care asigură autorizarea corectă și sigură a cererilor utilizatorilor, protejand și gestionand astfel eficient datele importante.

6 EVALUARE

6.1 Testarea Funcționalităților

Pentru a garanta corectitudinea implementărilor, s-a folosit metoda Black box, care presupune testarea incrementală a fiecărei funcționalități ale aplicației, fără a ține cont de partea tehnică. Această metodă de testare oferă cel mai apropiat tip de interacțiune cu aplicația cu cea a utilizatorilor. Acest mod de testare este explicat foarte în detaliu în cartea lui Boris Beizer. [3]

Pentru început au fost testate funcționalitățile care țin de autentificare: logarea, atât realizată corect, cât și realizată incorect pentru a observa erorile oferite de aplicație, crearea unui cont nou, și funcția de logout.

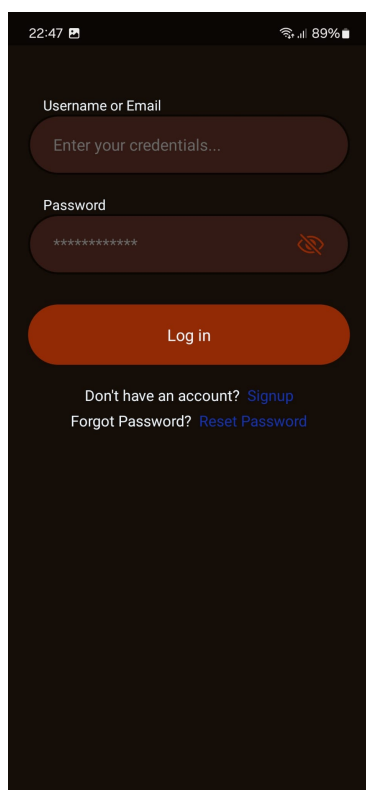
A mobile app screenshot of a login page. At the top, the status bar shows the time 22:47 and 89% battery. The page has a dark background. It features two input fields: 'Username or Email' with a placeholder 'Enter your credentials...' and 'Password' with a placeholder '*****' and a toggle icon. Below these is an orange 'Log in' button. At the bottom, there are links: 'Don't have an account? Signup' and 'Forgot Password? Reset Password'.

Figura 24: Login Page

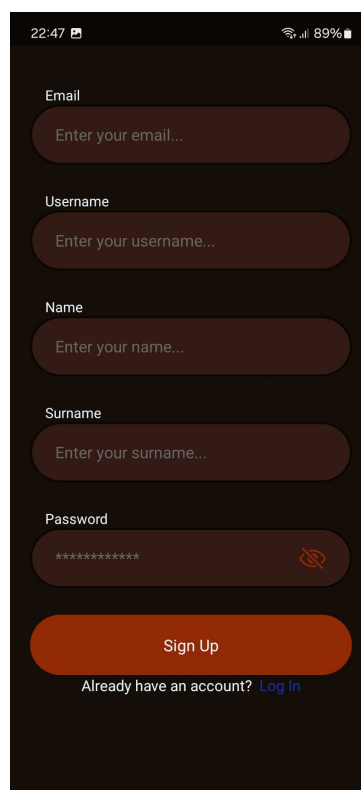
A mobile app screenshot of a sign-up page. The status bar at the top shows 22:47 and 89% battery. The page has a dark background. It features four input fields: 'Email' (placeholder 'Enter your email...'), 'Username' (placeholder 'Enter your username...'), 'Name' (placeholder 'Enter your name...'), and 'Surname' (placeholder 'Enter your surname...'). Below these is a 'Password' field with a placeholder '*****' and a toggle icon. An orange 'Sign Up' button is positioned below the password field. At the bottom, there is a link: 'Already have an account? Log In'.

Figura 25: Sign Up Page

	Numele testului	Funcționalitatea testată	Metoda de testare	Rezultat
1	SignUp	Crearea unui cont nou	Inserarea datelor personale, a unui mail valid, a unui username care nu este deja utilizat și a unei parole ce se ridică la standardele de securitate impuse de aplicație, apoi apăsarea butonului de Sign Up	succes
2	SignUp incorect	Eroarea corectă la încercarea de creare a unui cont cu un email invalid, cu o parolă cu mai puțin de 8 caractere, cu câmpuri necomplete sau cu un username existent	Introducerea datelor greșite pentru a primi o eroare	succes
3	Login	Logarea cu un cont existent	Inserarea credențialelor și parolei care corespund unui cont existent și apăsarea pe butonul de Login	succes
4	Login incorect	Eroarea corectă la autentificarea cu un cont care nu există sau cu credențialele sau parola greșite	Introducerea unui username greșit și a unei parole greșite, apoi apăsarea pe butonul de login. Introducerea unui username existent, dar a unei parole care nu îi aparține, și apăsarea pe butonul de login	succes
5	Resetare Parolă	Resetarea parolei prin mail	Apăsarea pe "Reset Password" urmată de completarea câmpului cu mail-ul. Copierea codului primit pe mail în aplicație pentru a obține autorizarea de a reseta parola, introducerea noii parole	succes
6	Logout	Ieșirea dintr-un cont	Navigarea pe pagina de profil, apăsare pe setări și apoi pe butonul de logout	succes
7	Reținere user	Păstrarea unui utilizator logat la ieșirea din aplicație și intrarea înapoi	Logare, închidere aplicație, repornire aplicație	succes

Tabela 1: Testare autentificare

Apoi au fost testate funcționalitățile ce țin de liste.

	Numele testului	Funcționalitatea testată	Metoda de testare	Rezultat
1	Listă nouă	Crearea unei liste noi	Apăsarea pe butonul de New List, introducerea unui nume, apăsarea pe bifă pentru a finaliza procesul	succes
2	Nume listă	Schimbarea numelui unei liste	Apăsarea pe o listă existentă pentru a o edita, schimbarea numelui care apare în câmpul cu numele listei, confirmare	succes
3	Adăugare album	Adăugarea unui album într-o listă	Apăsarea pe o listă existentă pentru a o edita, confirmarea continuării cu numele pe care lista îl are deja, apăsarea pe plus în pagina cu albumele listei, selectarea unui album din lista de Listened To, confirmare	succes
4	Ștergere album	Ștergerea unui album dintr-o listă	Apăsarea pe o listă existentă pentru a o edita, confirmarea continuării cu numele pe care lista îl are deja, apăsarea lungă pe un album pentru a-l șterge, confirmare	succes
5	Ștergere listă	Ștergerea unei liste existente	Apăsarea pe o listă existentă pentru a o edita, apăsare pe Delete List, confirmare	succes
6	Afișare listele mele	Vizualizarea listelor utilizatorului curent	Apăsarea pe My Lists	succes
7	Afișare alte liste	Vizualizarea listelor celorlalți utilizatori	Apăsarea pe Discover Lists	succes

Tabela 2: Testare liste

În continuare, a fost testată interacțiunea cu albumele și cu artiștii.

	Numele testului	Funcționalitatea testată	Metoda de testare	Rezultat
1	Căutare Album	Căutarea unui album	Pe pagina de căutare, selec-tarea tipului de căutare Al-bum, apoi introducerea nu-melui unui album	succes
2	Pagina Albumului	Vizualizarea paginii unui al-bum din locuri diferite	Apăsarea pe un album de pe pagina de căutare și de pe di-ferite alte pagini, confirmarea că pagina pe care utilizatorul este redirecționat corespunde albumului pe care a apăsăat	succes
3	Notă Album	Oferirea unei note unui album	Pe pagina unui album, se-lectarea unuia dintre discuri pentru a oferi o notă, confir-marea că nota este afișată și că albumul este pus la Liste-ned To în caz că nu era deja, sau scos de la Future Listens în caz că era	succes
4	Adăugare la Listened To/Future Listens	Adăugarea unui album la Lis-tened To sau la Future Lis-tens	Pe pagina unui album, apăsarea pe Listened To sau pe Future Listens, confirma-rea că albumul este adăugat și că câmpurile sunt actua-lizate conform interacțiunii descrise la Detalii de Imple-mentare	succes
5	Căutare Artist	Căutarea unui artist	Pe pagina de căutare, selec-tarea tipului de căutare Ar-tist, apoi introducerea nume-lui unui artist	succes
6	Pagina Artistului	Vizualizarea paginii unui ar-tist din locuri diferite	Apăsarea pe un artist de pe pagina de căutare și de pe di-ferite alte pagini, confirmarea că pagina pe care utilizatorul este redirecționat corespunde artistului pe care a apăsăat	succes

Tabela 3: Testare albume și artiști

Apoi a fost supusa la test interacțiunea cu ceilalți utilizatori.

	Numele testului	Funcționalitatea testată	Metoda de testare	Rezultat
1	Căutare User	Căutarea unui utilizator	Pe pagina de căutare, selectarea tipului de căutare Users, apoi introducerea numelui unui utilizator	succes
2	Pagina unui User	Vizualizarea paginii unui utilizator din locuri diferite	Apăsarea pe un utilizator de pe pagina de căutare și de pe diferite alte pagini, confirmarea că pagina pe care utilizatorul este redirecționat corespunde utilizatorului pe care a apăsat	succes
3	Follow User	Urmărirea unui utilizator	Pe pagina unui utilizator, apăsarea pe butonul de urmărire	succes
4	Unfollow User	Scoaterea unui utilizator de la urmăriti	Pe pagina unui utilizator, apăsarea pe butonul de scoatere de la urmărire	succes
5	Following/Followers ale unui user	Vizualizarea listei de urmăriti sau de urmăritori ai unui utilizator	Pe pagina unui utilizator, apăsarea pe pagina de urmăriti sau de urmăritori	succes
6	Top 6 al unui user	Vizualizarea top 6 al unui utilizator	Pe pagina unui utilizator, confirmarea că top 6 este afișat corect	succes

Tabela 4: Testare utilizatori

În cele din urmă, au fost testate funcționalitățile disponibile pe pagina de profil.

	Numele testului	Funcționalitatea testată	Metoda de testare	Rezultat
1	Poză Profil	Schimbarea pozei de profil	Apăsarea pe poza de profil, selectarea unei poze noi din galerie, confirmare	succes
2	Adăugare top	Adăugarea unui album în top 6	Apăsarea pe unul dintre plusurile care corespund unui spațiu gol în top, selectarea unui album	succes
3	Ștergere din top	Ștergerea unui album din top 6	Apăsarea lungă pe un album din top	succes
4	Listened To/Future Listens	Vizualizarea paginii de Listened To sau Future Listens	Apăsarea pe una dintre pagini	succes
5	Sortare Listened To/-Future Listens	Sortarea unei pagini	Pe una dintre pagini apăsarea pe butonul de sortare, selectarea unei metode de sortare, confirmarea sortării	succes
6	Following/Followers	Vizualizarea listei de urmăritori sau de urmăritori	Apăsarea pe una dintre pagini	succes

Tabela 5: Testare Profil

6.2 Testarea Performanței

Statisticile din tabela 6 arată cât de fluidă este interacțiunea cu aplicația. Metricile sunt oferite de Expo Go. Testarea a fost făcută pe un Samsung S21 FE. Aplicația a avut 0 blocări de interfață în timpul testării, și niciun fel de problemă grafică.

	Tipul de utilizare	FPS (Frames per second)
1	Idle	120
2	Accesarea paginii unui album	118
3	Schimbarea tab-urilor	120
4	Logare	115
5	Urmărire utilizator	120

Tabela 6: FPS utilizare

În continuare, s-au facut teste pentru a analiza timpul de răspuns pentru cele două baze de date, iar comparația dintre ele este exemplificată în figura 26.

	Numărul de cereri	Timpul pentru cereri (ms)
1	2	298
2	4	301
3	8	305
4	16	334
5	32	402
6	64	732
7	128	—

Tabela 7: Timp de raspuns al Spotify API pentru cereri de tip GET

	Numarul de cereri	Timpul pentru cereri (ms)
1	2	206
2	4	207
3	8	210
4	16	243
5	32	320
6	64	502
7	128	987

Tabela 8: Timp de raspuns al bazei de date din Strapi pentru cereri de tip GET

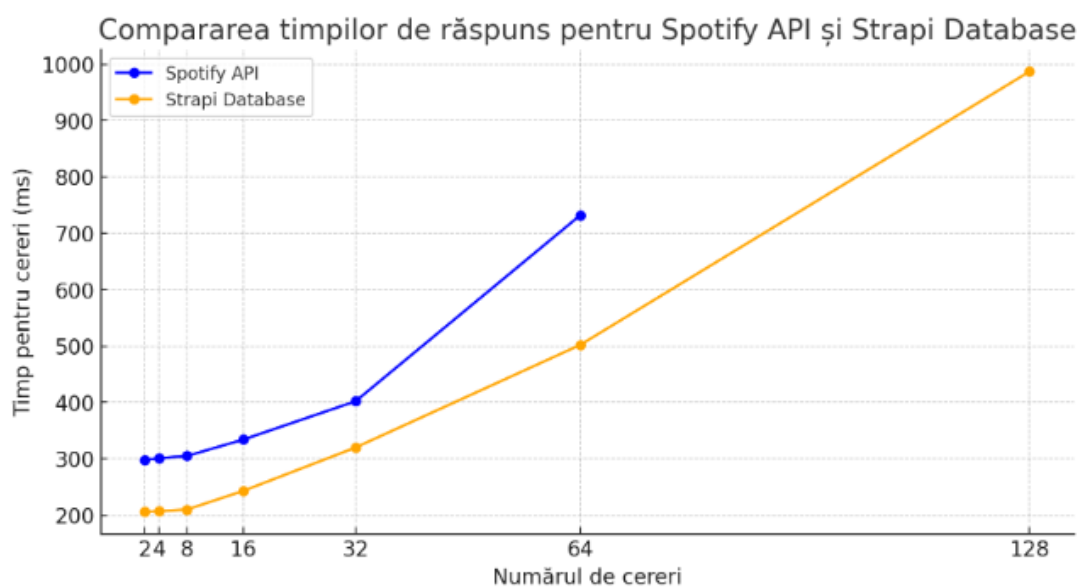


Figura 26: Grafic Comparativ

Se poate observa că baza de date din Strapi are un timp de răspuns mai scurt decât cea de la Spotify, lucru datorat faptului că este ținută local, și că utilizatorul este conectat la aceeași rețea cu ea.

La testarea bazei de date de la Spotify s-a observat o limită de cereri pe secundă, iar orice depășea 70 de cereri rezulta garantat într-o eroare de tip 429. Acest lucru este cauzat de limita pe care Spotify o impune celor care folosesc API-ul lor. [14]

Implementarea aplicației Sleeve se asigură că această limită nu este niciodată atinsă. Singurele pagini care fac mai mult de o cerere către Spotify API sunt pagina de căutare și pagina artistului. Un artist nu poate avea mai mult de 70 de albume, iar pagina de căutare limitează rezultatele afișate. Paginile care necesită mai multe cereri, precum pagina de Listened To, fac aceste cereri către baza de date din Strapi, în care sunt salvate datele despre albumele utilizatorului. Baza de date din Strapi nu are limită de cereri pe secundă.

6.3 Posibile probleme

Una dintre micile erori pe care le întâmpină aplicația apare la **schimbarea pozei de profil**. S-a încercat o implementare care permite alegerea între a selecta o poză din galerie sau a deschide camera telefonului pentru a face una nouă, dar nu s-a reușit o comunicare cu baza de date cu o acuratețe prea mare în acest caz. Abordarea curentă de care aplicația dispune, cea în care utilizatorul primește doar opțiunea de a alege o poză din galerie funcționează, dar în una din zece încercări este întoarsă o eroare de conectare cu baza de date. Acest lucru poate fi cauzat de modul de încărcare a unei imagini în Strapi. Mai întâi ea trebuie pusă în galerie din Strapi, iar apoi trebuie dată o referință a pozei din galerie din baza de date printr-un request API. Partea care întâmpină probleme este cea cu urcarea în galerie.

Aplicația funcționează în mediul oferit de Expo. Acest lucru simplifică foarte tare procesul de dezvoltare, însă are limitările sale. El nu pregătește o aplicație să fie **gata de lansarea în producție**, fiind doar un mediu de dezvoltare. Totuși se poate scoate din acest mediu cu ușurință dacă se dorește asta.

În plus, baza de date este găzduită pe un **server local**, ceea ce înseamnă că utilizatorii trebuie să fie conectați la aceeași rețea ca serverul pentru a accesa aplicația.

Setul de date este cel oferit de Spotify, care este gratuit pentru aplicații care nu sunt lansate în producție, însă pentru o aplicație care dorește a fi monetizată, trebuie găsit un alt set de date sau încheiat un contract cu cei de la Spotify.

7 CONCLUZII

Sleeve Music Library este un răspuns direct la lipsa unei aplicații de tip Letterboxd, dar pentru albume de muzică. Ea rezolvă problemele pe care le au soluțiile curente pentru critica muzicală, abordând aspectul social într-o manieră similară aplicațiilor de genul social media. Arhitectura folosită ajută la o funcționare corectă și eficientă a aplicației, și se supune criteriilor de scalabilitate. Compatibilitatea cu diferite platforme oferită de React Native se asigură că cei care vor dori să folosească aplicația nu vor fi restricționați de sistemul de operare al telefonului lor. Interfața se ridică la cerințele moderne, fiind intuitivă, și foarte ușor de utilizat.

7.1 Posibile îmbunătățiri

O **pagină de tipul "Pentru tine"** pe care utilizatorul ar putea vedea noutăți special selectate pentru el ar oferi ceva de făcut unui utilizator pe aplicație constant.

Recomandări pe baza albumelor ascultate, cu ajutorul inteligenței artificiale [15], ar ajuta mult la îndeplinirea scopului aplicației. Acestea ar oferi utilizatorilor ceva nou de ascultat, ținându-i interesați și captivați.

Detalii suplimentare despre albume ar putea fi oferite, însă pentru această aplicație s-a optat pentru un design mai simplist, pentru a favoriza o utilizare cât mai intuitivă.

Pe lângă notele oferite de utilizatori, o secțiune cu un **review scris** i-ar ajuta să își exprime opiniile în totalitate.

Statistici despre notarea generală a unui album ar oferi utilizatorilor o părere publică despre un album pe care nu l-au ascultat, sau i-ar lăsa să își compare părerea lor cu cea a altor utilizatori.

7.2 Experiența personală

La începutul proiectului, nu aveam nicio cunoștință în domeniul aplicațiilor mobile. Nu știam cum se dezvoltă sau ce ar trebui să conțină pentru a putea concura cu celelalte soluții din prezent.

Pentru acest proiect, a fost nevoie de o analiză foarte detaliată a tehnologiilor disponibile, pentru a alege o arhitectură cât mai adecvată. Componentele tehnologice trebuie să se completeze și să nu își impună limite una alteia.

După alegerea tehnologiilor, a urmat o perioadă lungă de înțelegere a modului de funcționare a fiecăreia și de învățare a limbajului principal de programare: React Native. Neavând experiență

În dezvoltarea aplicațiilor web, faptul că limbajul principal este JavaScript nu mi-a ușurat prea mult acest proces.

Apoi a urmat procesul de dezvoltare propriu-zisă. Acesta nu a fost lipsit de probleme, apariția bug-urilor fiind foarte frecventă. Am fost nevoit să regândesc logica aplicației de câteva ori, să rescriu tabelele din baza de date și să reimplementez componente de la zero. Toate acestea fiind spuse, a fost o experiență foarte educativă.

Acest proiect a fost o metodă foarte bună de a învăța procesul de dezvoltare a unei aplicații mobile și m-a ajutat să înțeleg bazele acestui domeniu.

BIBLIOGRAFIE

- [1] The evolution of swift programming language for ios app development. <https://moldstud.com/articles/p-the-evolution-of-swift-programming-language-for-ios-app-development>. Last accessed: 9 June 2024.
- [2] Salman Ahmed and Qamar Mahmood. An authentication based scheme for applications using json web token. In *2019 22nd international multitopic conference (INMIC)*, pages 1–6. IEEE, 2019.
- [3] Boris Beizer. *Black-Box Testing: Techniques for Functional Testing of Software and Systems*. 1995.
- [4] Journalist Charlotte Wolfstitch. What is letterboxd and why is it becoming so popular? <https://bmgator.org/36215/opinion/what-is-letterboxd-and-why-is-it-becoming-so-popular/>. Last accessed: 5 June 2024.
- [5] Jessica Clark. Strapi vs firebase. <https://blog.back4app.com/strapi-vs-firebase/>. Last accessed: 18 June 2024.
- [6] Ashraf Farid. Building apps with expo react native: Pros cons. <https://upstackstudio.com/blog/expo-react-native/>. Last accessed: 21 June 2024.
- [7] Flatirons. Popularity of flutter vs. react native in 2024. <https://flatirons.com/blog/popularity-of-flutter-vs-react-native-2024/>. Last accessed: 20 June 2024.
- [8] Marie Charlotte Götting. Music streaming services worldwide - statistics facts. <https://www.statista.com/topics/11066/music-streaming-services-worldwide/#topic0verview>. Last accessed: 1 June 2024.
- [9] Chunnu Khawas and Pritam Shah. Application of firebase in android app development-a study. *International Journal of Computer Applications*, 179(46):49–53, 2018.
- [10] Frederic Lardinois. Kotlin is now google's preferred language for android app development. https://techcrunch.com/2019/05/07/kotlin-is-now-googles-preferred-language-for-android-app-development/?guccounter=1&guce_referrer=aHR0cHM6Ly93d3cuZ29vZ2xlLmNvbS8&guce_referrer_sig=AQAAAMZbyyPTzeGqm8tMY9AmlmzLzgOVH19CbEIHNn7ADwDFnRGbWo-w8Txr4jX3v7-8gBRqzGUJW8GtkfDa0hy5Us_eSAEdPWU9mbhUunVXXUjgmJ_

Ta0mHyS03LsSaK6wyhy14d3cW3Wu-OHAe3I2EAW5up1xErUHv1UEJUz3ad. Last accessed: 19 June 2024.

- [11] React Native. React native. *línea*]. Disponible en: <https://reactnative.dev/>. [Último acceso: 2 de noviembre 2019], 2020.
- [12] PhD. Robert M. Brecht. Country music listeners are having a moment. <https://tseentertainment.com/country-music-listeners-are-having-a-moment/>. Last accessed: 25 May 2024.
- [13] Spoify. Spotify client credentials flow. <https://developer.spotify.com/documentation/web-api/tutorials/client-credentials-flow>. Last accessed: 9 June 2024.
- [14] Spoify. Spotify rate limit. <https://developer.spotify.com/documentation/web-api/concepts/rate-limits>. Last accessed: 18 June 2024.
- [15] UL Tupe, Anvit Kulkarni, Gaurav Nimbokar, Prasad Mahajan, and Narayan Raut. Ai based song recommendations system. *Grenze International Journal of Engineering & Technology (GIJET)*, 10, 2024.