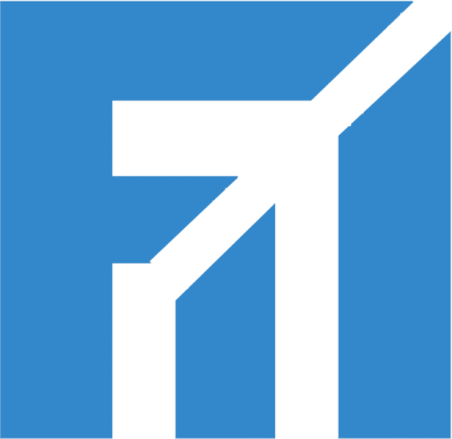
UNIVERSITATEA „ALEXANDRU IOAN CUZA” IAȘI

**FACULTATEA DE INFORMATICĂ**

****

LUCRARE DE LICENȚĂ

**Noiseblend – O nouă experiență muzicală**

Propusă de

**Matei – Șerban Sandu**

Sesiunea**: iulie, 2019**

Coordonator științific

**Conf. Dr. Anca Vitcu**

UNIVERSITATEA „ALEXANDRU IOAN CUZA” IAȘI

**FACULTATEA DE INFORMATICĂ**

**Noiseblend – O nouă experiență muzicală**

**Matei – Șerban Sandu**

Sesiunea**: iulie 2019**

Coordonator științific

**Conf. Dr. Anca Vitcu**

Avizat,

Îndrumător Lucrare

Titlul, Numele şi prenumele Conf. Dr. Vitcu Anca

Data \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Semnatura \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**DECLARAŢIE**

**privind originalitatea conţinutului lucrării de licenţă/disertaţie**

Subsemntatul(a) Sandu Matei – Șerban,domiciliul în Iași, Aleea Mihail Sadoveanu nr. 61D, născut(ă) la data de 13-03-1994 identificat prin CNP 1940313076439 absolvent al Universităţii „Alexandru Ioan Cuza” din Iaşi, Facultatea de Informatică specializarea Informatică, promoţia 2019, declar pe propria răspundere, cunoscând consecinţele falsului în declaraţii în sensul art. 326 din Noul Cod Penal şi dispoziţiile Legii Educaţiei Naţionale nr. 1/2011 art.143 al. 4 si 5 referitoare la plagiat, că lucrarea de licenţă cu titlul: Noiseblend – O altă experiențp muzicală elaborată sub îndrumarea Conf. Dr. Vitcu Anca, pe care urmează să o susţin în faţa comisiei este originală, îmi aparţine şi îmi asum conţinutul său în întregime.

De asemenea, declar că sunt de acord ca lucrarea mea de licenţă/disertaţie să fie verificată prin orice modalitate legală pentru confirmarea originalităţii, consimţind inclusiv la introducerea conţinutului său într-o bază de date în acest scop.

Am luat la cunoştinţă despre faptul că este interzisă comercializarea de lucrări ştiinţifice in vederea facilitării fasificării de către cumpărător a calităţii de autor al unei lucrări de licenţă, de diploma sau de disertaţie şi în acest sens, declar pe proprie răspundere că lucrarea de faţă nu a fost copiată ci reprezintă rodul cercetării pe care am întreprins-o.

Data ………………………… Semnătură student …………………………

DECLARAȚIE DE CONSINȚĂMÂNT

Prin prezenta declar că sunt de acord ca Lucrarea de licență cu titlul „ Noiseblend – O nouă experiență muzicală”, codul sursă al programelor şi celelalte conţinuturi (grafice, multimedia, date de test etc.) care însoţesc această lucrare să fie utilizate în cadrul Facultăţii de Informatică.

De asemenea, sunt de acord ca Facultatea de Informatică de la Universitatea „Alexandru Ioan Cuza” Iași să utilizeze, modifice, reproducă şi să distribuie în scopuri necomerciale programele-calculator, format executabil şi sursă, realizate de mine în cadrul prezentei lucrări de licenţă.

Iași,

Absolvent Matei – Șerban Sandu

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Cuprins

[1. Introducere 1](#_Toc12731454)

[1.1 Motivație 1](#_Toc12731455)

[1.2 Contribuții 1](#_Toc12731456)

[1.3 Cerințe funcționale 2](#_Toc12731457)

[1.4 Abordare tehnică 3](#_Toc12731458)

[2. Proiectare 4](#_Toc12731459)

[2.1 Tehnologii folosite 4](#_Toc12731460)

[2.1.1 Server 4](#_Toc12731461)

[2.1.2 Client 4](#_Toc12731462)

[2.2 Arhitectura soluției 5](#_Toc12731463)

[2.2.1 Structura back-end-ului 6](#_Toc12731464)

[2.2.2 Structura front-end-ului 7](#_Toc12731465)

[3. Funcționalitățile aplicației 9](#_Toc12731466)

[3.1 Blend 9](#_Toc12731467)

[3.2 Discover 13](#_Toc12731468)

[3.2.1 Opțiunile din discover 13](#_Toc12731469)

[3.2.2 Playlist 17](#_Toc12731470)

[3.2.3 Profile 22](#_Toc12731471)

[4. Concluzii 23](#_Toc12731472)

[5. Bibliografie 24](#_Toc12731473)

# 

# Introducere

*Noiseblend* este o aplicație web destinată utilizatorilor Spotify, cu scopul de a le oferi o experiență nouă în materie de muzică. Folosindu-se de un set de date relativ mic din API-ul aplicației de muzică, aceasta reușește să creeze noi metode de a construi playlist-uri, totodată punând la dispoziție noi feature-uri extrem de utile ce devin, pe parcursul utilizării, tot mai indispensabile utilizatorilor.

## Motivație

Mai mult sau mai puțin, fiecare dintre noi iubește muzica și este foarte probabil ca la un moment dat să ajungem să avem nevoie de noi idei sau sugestii. În general, gusturile diferă, așa că nu este indicat să oferim aceleași sugestii mai multor oameni pentru că există riscul de a-i nemulțumi. De aceea, *Noiseblend* reușește sa plaseze sugestiile în funcție de gusturi îmbunătățind calitatea acestora prin simplul fapt că oferă posibilitatea de a le filtra în funcție de o multitudine de atribute precum: acustică, tonalitate, ritm, energie si multe altele.

Această aplicație a fost dezvoltată pornind de la pasiunea mea pentru muzică și totodată din necesitatea de a avea la dispoziție un accesoriu cu adevărat util și satisfăcator in materie de *music discovery.* Bazându-mă pe conceptul de a construi ceva ce îmi lipsește, multe idei noi au fost adăugate pe parcurs, aducând acest produs la o calitate mult mai înaltă.

În cazul în care ești pasionat de muzică, rămâi des fără surse de inspirație și dorești ceva cu totul nou, *Noiseblend* îți sare în ajutor si foarte probabil se va ridica la măsura așteptărilor tale.

## Contribuții

Pe parcursul dezvoltării acestei aplicații am recurs la studierea forumului gigantului serviciu digital de muzică – Spotify, am colectat dorințele cele mai populare ale utilizatorilor și le-am inclus într-o manieră cât mai logică și atrăgătoare în această nouă aplicație.

Nu este singura aplicație care se bazează pe API-ul acestui popular serviciu de muzică, dar este categoric singura aplicație de acest tip și față de acesta, printr-o multitudine de noi funcționalități, reușește să se remarce prin stilul unic de a genera liste de piese folosindu-se de o interfață cu un număr foarte ridicat de detalii.

## Cerințe funcționale

Oferind o vastă gamă de funcționalități, voi construi o așa-zisă clasificare a acestora în funcție de importanța considerată de mine, evident această importanță poate diferi de la utilizator la utilizator:

1. Playlisturi instante în funcție de activitatea curentă.

Folosirea acestor playlist-uri este foarte simplă și se comportă ca o aplicație mobile având la bază referințe la aplicația web.

Având genul de muzică potrivit activității desfășurate in acel moment, totodată îmbunătățit de preferințele tale, nivelul de concentrare și eficientizare poate crește considerabil, oferind în același timp o stare de entuziasm, eventual motivație.

1. Combinarea de artiști diferiți

Această funcționalitate pune la dispoziție setul de artiști preferați și prin selectarea până la trei dintre aceștia, se vor genera piese ale acestora, adăugându-se similarități complet relevante selecției.

1. Limitarea la un singur gen de muzică/ țară de proveniență.

Listând majoritatea genurilor de muzică existente, evident prioritizate în funcție de obiceiurile utilizatorului, se pot crea seturi de piese cu o bază strict limitată la acel gen muzical sau acea țară.

1. Eliminarea definitivă a sugestiilor nedorite.

Multe sugestii pot fi considerate nerelevante pentru un utilizator, sau poate un anume artist care este ușor asociat cu unul dintre preferații utilizatorului, nu îi este nicicum pe plac. De aceea, salvând constant nepotrivirile de gusturi in baza de date si efectuând o filtrare a acestora la fiecare generare de playlist de orice natură, se va oferi în mod categoric o experiență mult mai calitativă.

1. Sortarea si filtrarea în funcție de atribute muzicale

Oferind atribute precum acustică, dansabilitate, energie, nivel de instrumental, liveness, volum, intonanță, valență, tempo, durată și notă muzicală pentru selecția utilizatorului, acestea sunt folosite de către algoritmi pentru a sorta sau filtra prin intervalele setate din interfață, schimbând lista în mod direct.

1. Eliminarea dintr-un playlist a pieselor cu caracter vulgar

Această funcționalitate este utilă atât in cazul în care utilizatorul nu este un fan al pieselor cu caracter vulgar, cât și cazului în care playlist-ul dorit este adresat într-un cadru familial și nu este dorită această notă grosolană.

## Abordare tehnică

Proiectul este scris într-un manieră relativ comună si specifică aplicațiilor web, și are la bază un server și un client.

Serverul este dezvoltat in Python si conține toți algoritmii ce stau la baza funcționalităților, rezultatele acestora fiind expuse în urma cererii clientului. Există două comunicări cu două baze de date diferite: cea de la Spotify ce oferă date generale despre utilizatori, și cea necesară aplicației, ce conține date despre artiști, tări de proveniență ale acestora, genuri muzicale, tradiții specifice respective acestor țări și seturi de playlist-uri deja create pentru cazul în care sugestiile inițiale sunt nefondate din cauza lipsei de activitate anterioare ale utilizatorului.

Pornind de la ideea de a crea ceva ce consider că lipsește Spotify-ului, am luat în considerare inițial obiceiurile utilizatorului de aici și am introdus noi algoritmi pentru oferi noi metode de a găsi muzică pe gustul acestora.

Clientul a fost dezvoltat cu Next.js folosind librăria ReactJS, o librărie JavaScript ce s-a dovedit a fi cea mai utilă acestui proiect în ceea ce privește dezvoltarea interfeței. Dorința de a scrie componente incapsulate ce își gestionează singure starea, alături de premiza de a evita codul duplicat, s-au transformat pe parcurs într-o necesitate deoarece aplicația creștea și ar fi devenit imposibil de gestionat cursul acesteia fără a avea la dispoziție astfel de caracteristici specifice librăriei ReactJS.

Pentru partea de ­­­creare, citire, updatare și ștergere am folosit PostgreSQL, o bază de date relațională cu care am reușit să creez relații și să comunic într-o manieră extrem de ușoară cu ajutorul librăriilor asyncpg și PonyORM.

Fiind o aplicație web, nu se vor întâmpina probleme la accesarea acesteia indiferent de sistemul de operare sau browser folosit, dar acest lucru tot nu mi s-a părut indeajuns. Prin folosirea PWA-ului (Progressive Web Apps) am oferit posibilitatea de a accesa *Noiseblend* într-o manieră extrem de similară unei aplicații Mobile. Progressive Web App-ul este un tip de aplicație mobile livrată prin web, și este construită folosind tehnologiile de bază ale acestuia: HTML, CSS și Coffeescript(JavaScript).

# Proiectare

## Tehnologii folosite

În acest subcapitol voi descrie într-un mod minimalist tehnologiile demne de menționat alături de valoarea adăugată la această aplicație.

### Server

Sanic este un server web disponibil pentru utilizare începând cu Python 3.6 și totodată un framework web conceput pentru a gestiona cererile și răspunsurile într-o manieră cât se poate de rapidă. Încă de la versiunea 3.5 de Python, a fost oferită posibilitatea de a utiliza design-ul de programare concurentă (AsyncIO), iar Sanic permite folosirea acestui design reușind cu ușurintă să își atingă țelul, și anume: rapiditate.

AsyncIO, paradigma pe care tocmai am menționat-o, are implementări în mai multe limbaje de programare. Cum își pune aceasta amprenta in Python? Răspunsul este simplu și anume prin punerea la dispoziție a pachetului *asyncio* și a noilor keyword-uri *async/await* pentru definirea corutinelor. Aceste generatoare specializate sunt ceea ce definesc paradigma asincronă în Python.

Urmând modelul de până acum, am devenit fascinat de paradigma async așă că, la momentul alegerii unei librării pentru baza de date, am decis să folosesc asyncpg, o interfață creată strict pentru PostgreSQL și bineînțeles într-o strânsă relație cu AsyncIO din Python.

### Client

Probabil cel mai util pentru randarea aplicațiilor bazate pe React, indiferent de sursa datelor este Next.js. Acesta, fără a fi necesar să înveți să utilizezi o nouă librărie, oferă un posibilitatea de a exporta un site static fără efort. Odată cu Next.js vine și o nouă metodă de a scrie CSS-ul, prin *styled-jsx*, ce permite includerea CSS-ului in JavaScript, oferind totodată suport deplin pentru fiecare soluție din acestea.

*React* estelibrăria Javascript ce se poate caracteriza prin: declarativa sa unică de a reda posibilitatea de a programa mai ușor și de a testa mai ușor, scrierea de componente incapsulate ce-și gestionează singure starea și evitarea codului duplicat.

Un mare ajutor a fost oferit de *Redux*, cu care am reușit să gestionez stările globale alea modului de front-end precum și ale stărilor fiecărei pagini în parte.

*Redux-saga* este o librărie middleware, creată pentru a gestiona efectele colaterale într-o aplicație redux. Se folosește de generatorii puși la dispoziție de ES6, permitând dezvoltarea unui cod asincron și foarte ușor de testat. De menționat că termenul de *saga,* ce are o istorie în lumea informaticii, nu este strict limitat doar pentru JavaScript, și este un pattern folosit pentru a gestiona și împiedica tranzacțiile de lungă durată ce pot genera efecte colaterale sau chiar eșecuri.

Coffeescript este principalul limbaj folosit pentru a scrie componenta web a acestei aplicații și este un limbaj mic ce compilează direct in JavaScript. Acesta a fost creat pentru a expune calitățile și utilitatea JavaScript-ului folosind o sintaxă mult mai atractivă. Acest limbaj este compatibil complet în ambele sensuri cu JavaScript, acesta fiindu-i scopul principal mergând după premiza *It is just JavaScript.*

## Arhitectura soluției

Proiectul este structurat in două mari module și anume: cel ce deține interfața de programare a aplicației și cel ce conține componentele web împreună cu logica caracteristicilor de stare proprietăți și design.

### Structura back-end-ului

Deoarece sunt o mulțime de fișiere de configurare în afara acestui modul, am restrâns prezentarea structurii după cum puteți vedea in următoarea figură.

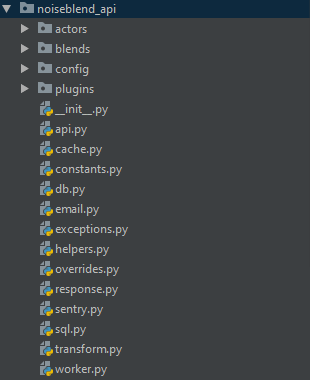
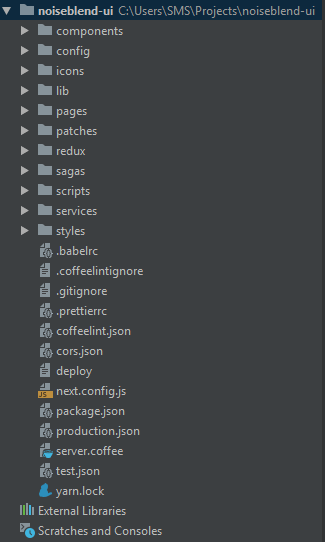


Fig. 1 – Structura back-end-ului

* *Actors –* aici se regăsește un set de actori, fiecare fiind responsabil pentru un grup de acțiuni precum autentificarea prin Spotify, generarea playlist-urilor pentru blend-uri, *listener*-ul de dispozitive și efectuarea redării pe unul dintre acestea, funcția radio, potrivirea de artiști, filtrarea pieselor, efectul de scădere sau creștere a volumui pe un interval setat de utilizator din aplicație.
* *Blends –* directorul ce ține cele opt blend-uri instante ce sunt create printr-un algoritm ce se folosește de obiceiurile utilizatorilor care sunt ponderate.
* *Config –* fișierele ce țin toate configurările necesare back-end-ului se vor regăsi în acest director.
* *Plugins –* folderul ce conține helperi pentru conexiunile la baza de date și folosirea conexiunilor din pool, autentificare, cache control, stabilire de priorități, adăugarea de noi clienți etc.
* Restul fișierelor din directorul părinte ce au o importanță mai ridicată:
  + Api.py – fișierul principal de control al aplicației
  + Db.py – fișierul ce obține date generale din tabela user
  + Exceptions.py – excepții customizate folosite in modulul de back-end
  + Sql.py – interogări folosite foarte des în întreaga aplicație

### Structura front-end-ului



După cum puteți observa in figura alăturată, structura modului de *user interface* este realizată într-un mod pe cât se poate de organizat iar în continuare voi descrie fiecare director în parte și cu ce se ocupă.

* *Components* – este directorul în care sunt amplasate componentele UI, fiecare dintre ele fiind referențiată în cel puțin două locuri în oricare locație a proiectului. Acestea sunt configurate într-un mod dinamic și pot fi mapate cu ușurință în orice pagină din front-end evitând astfel codul duplicat.
* *Config* – este locul în care sunt aflate toate constantele printre care mărimile unor imagini, culorile unor componente inclusiv calcule de gradienți
* *Icons* – un director ce deține toate imaginile vectoriale scalabile folosite atât în componente cât și în pagini
* *Lib* – conține un set de fișiere coffee ce exportă funcții utilitare aplicației.
* *Pages* – cel mai complex director al aplicației și conține un fișier coffee desemnat paginilor, fiecare dintre ele având implementările componentelor de lifecycle din

Fig. 2 – Structura front-end-ului

React și logica din spatele fiecărei pagini web.

* *Patches –* conține patchfiles pentru conversia CSS-ului pur în styled-jsx
* *Redux –* aici este locul în care se mențin state-urile pentru fiecare pagina din *pages*. După cum am menționat mai sus, toate aceste state-uri pot crea efecte secundare și pot altera cursul dorit al aplicației, astfel folosirea acestui middleware a devenit esențială.
* *Sagas –* în strânsă legătură cu redux, efectuează apelurile către API precizând endpoint-ul, interogarea dorită și diferiți parametri necesari finalizării cererii.
* *Scripts –* diferite scripturile de tip shell/fish pentru optimizarea noilor imagini.
* *Services –* oferă logica pentru transmiterea token-urilor de autentificare către Spotify si serverul de back-end, obținerea detaliilor despre utilizatori, a recomandărilor, efectuarea cererilor de redare, oprire, schimbare, salvare, modificare, clonare, filtrare a unui playlist pe unul dintre dispozitivele selectate, aducerea caracteristicilor audio, a genurilor muzicale, a dispozitivelor ce au instalate aplicația Spotify și sunt gata să redea muzica pregătită și multe altele.
* *Styles –* conține animații ale imaginilor vectoriale și logica acestora pe lângă toate referințele la importările de imagini din extern.
* Restul fișierelor din home demne de menționat:
  + .babelrc – presetări pentru compatibilități cu anumite versiuni de browsere și referințe la anumite plugin-uri
  + Cors.json – conține toate adresele ce au acces în mod direct la aceasta aplicație.
    - Allowed-origin | response-header | allowed-methods
  + Next.config.js – aici se găsesc toate configurările serverlui next: adrese, chei sentry, referințe la API, node-modules etc.
  + Server – manifest, caching, handlers

# Funcționalitățile aplicației

## Blend

Acest modul al aplicației reprezintă cea mai importantă funcționalitate a aplicației și redă posibilitatea utilizatorului de a crea un playlist instant prin selectarea unuia dintre cele opt blend-uri afișate în pagina /blend a aplicației.

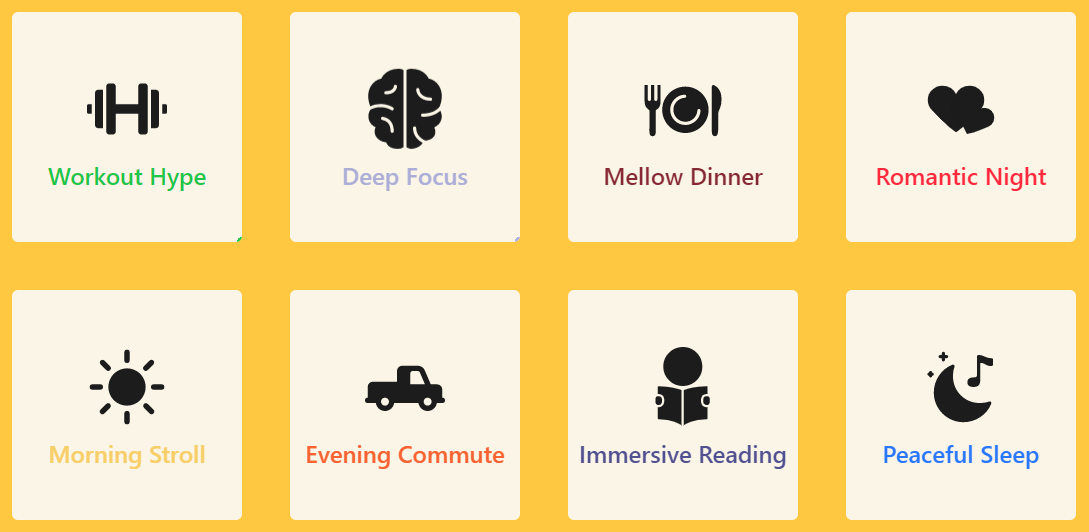


Fig. 3 – Tipuri de blend-uri

Fiecare dintre acestea constă într-un playlist cu anumite caracteristici bazate pe tipul de activitate pentru care este deservit playlist-ul, și anume:

1. Workout Hype, generează un playlist cu un caracter energic, aplicând atribute precum *energy, speechiness, tempo* adăugând în același timp piese preferate ale utilizatorului ce se încadrează în intervalele acceptate de algoritm.
2. Deep focus este un playlist cu un caracter complet diferit față de Workout Hype, bazându-se mai mult spre o gamă de muzică înclinată spre genul clasic, ce prezintă un nivel de *instrumentalness* ridicat, o valență medie, *danceability* scăzut și un tempo într-un inteval restrâns, menit să ofere ascultătorului o stare cât mai potrivită învățatului sau să îi sporească capacitatea de a se concentra.
3. Mellow Dinner, la dispoziție pentru a crea o atmosferă liniștită și perfectă unei cine ce se dorește a fi servită în liniște, eventual cu un fundal ideal pentru face simțită comunicarea mai plăcută
4. Romantic Night, după cum spune și numele, este un playlist destinat serilor romantice în care un fundal muzical corespunzător este esențial. Acesta se bazeaza pe atribute precum o acustică puțin sporită, un instrumental evidențiat și eventual un nivel de dansabilitate deasemenea puțin peste medie.
5. Morning Stroll reprezintă un playlist la bază energic, menit să inducă o stare potrivită pentru începutul unei zile noi.
6. Evening Commute este un playlist ce se este destinat momentului de tranziție de la întoarcerea de la job spre casă.
7. Immersive Reading are un caracter asemănător playlist-ului Deep Focus dar cu câteva atribute schimbate precum *loudness* ce aici este mai scăzut, cel puțin o nota mai joasă și o acustică mai ridicată.
8. Peaceful Sleep se remarcă prin caracteristi cum ar fi *loudness* extrem de scăzut, tempo ce tinde spre zero, și energie deasemenea scăzută.

De menționat că la fiecare utilizare de blend, playlist-ul va fi mereu unul parțial nou.

Realizarea acestei funcționalități și a acestui proces va fi descrisă în continuare pornind de la conceptul de PWA (Progressive Web App), ce este menit să ofere utilizatorilor o experiență foarte asemănătoare cu cea a folosirii unei aplicații mobile dar la bază totul se realizează pe concepte web.

Așadar, calitățile acestei funcționalități ar fi: viteza, care este imediat observată îmbunătățind calitatea animațiilor și a efectelor generale precum scroll și senzația captivantă ce face utilizatorul să uite că nu folosește o aplicație mobile.

Manifest-ul pentru Web App este un simplu fișier JSON care transmite browser-ului detalii despre aplicație și mai ales cum să se comporte când este instalat pe un dispozitiv mobile sau desktop. Mai jos regăsim un exemplu de implementare și modul ușor prin are acest concept PWA se poate folosi.

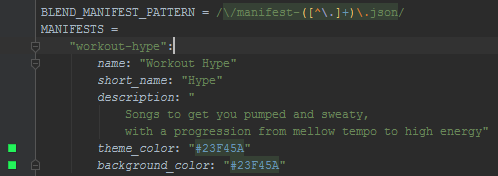


Fig. 4 – Exemplu de manifest file pentru cazul Workout Hype

Așadar, la baza unui acest tip de fișier, stau atributele nume și url, dar fiecare se poate customiza folosind alte atribute ca în exemplul din figura 3: short\_name, description, theme\_color, background\_color.

În continuare vom prezenta tot un exemplu de implementare al blend-ului, de data aceasta însă mergând pe logica execuției acestuia.

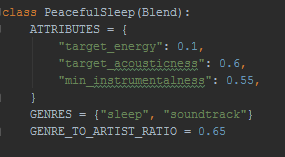


Fig. 5 – Exemple de atribute folosind clasa PeacefulSleep

Atributele pe care le putem asigna unei clase ce moștenesc părintele Blend, pot fi de două tipuri:

* *Target –* se precizează o valoare a atributului spre care se dorește ca algoritmul să tindă
* *Min / max –* aici se precizează valori minime și/sau maxime ale atributului pe care îl dorim luat în considerare de către algorim

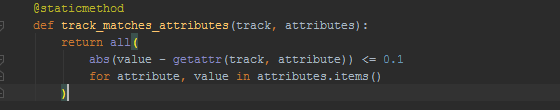


Fig. 6 – Funcție de attribute matching

În figura 5 avem o simplă funcție statică pentru returnarea tuturor track-urile ce îndeplinesc condițiile stabilite în figura 4. Această funcție este folosită în toți algoritmii ce oferă sugestii fie pe bază de top\_tracks, fie pe bază de top\_genres. În continuare vom oferi un exemplu de recomandare bazată pe top\_tracks.

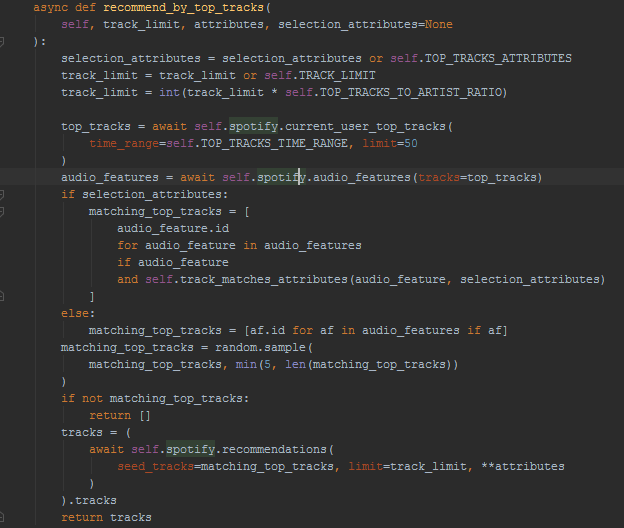


Fig. 7 – Exemplu de recomandare bazată pe top\_tracks

Se iau piesele cele mai ascultate ale utilizatorului, se verifică relevanța acestora luând în considerare atributele și apoi se vor genera recomandări, respectiv playlist-ul în sine, acestea sunt trimise mai departe către worker-ul ce este responsabil cu redarea sunetului și astfel playback-ul este pornit.

Un alt mod de a genera track-uri este prin folosirea top\_genres. Această modalitate presupune verificarea apartenenței anumitor artiști la genurile muzicale preferate de către utilizator și încadrarea acestora, după executarea filtrării pe atributele blend-ului curent, în playlist-ul final.

## Discover

### Opțiunile din discover

Discover reprezintă cel de-al doilea modul al aplicației și pune la dispoziție patru metode de a crea playlist-uri.

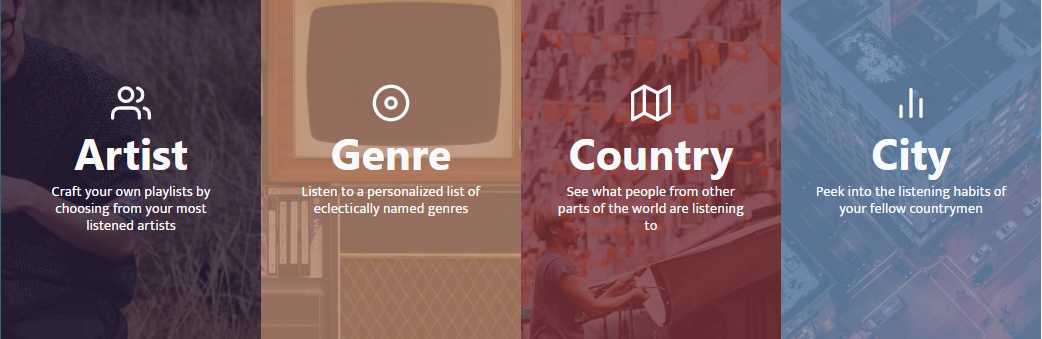


Fig. 8 – Modulul discover

***Artists*** este modulul ce oferă utilizatorului posibilitatea de a crea playlist-uri prin selectarea de la unu până la trei artiști, generându-se piese nu numai ale acelor artiști, ci și piese cu același caracter sau gen muzical, artiști cu care au avut eventuale colaborări anterioare inclusiv albume.

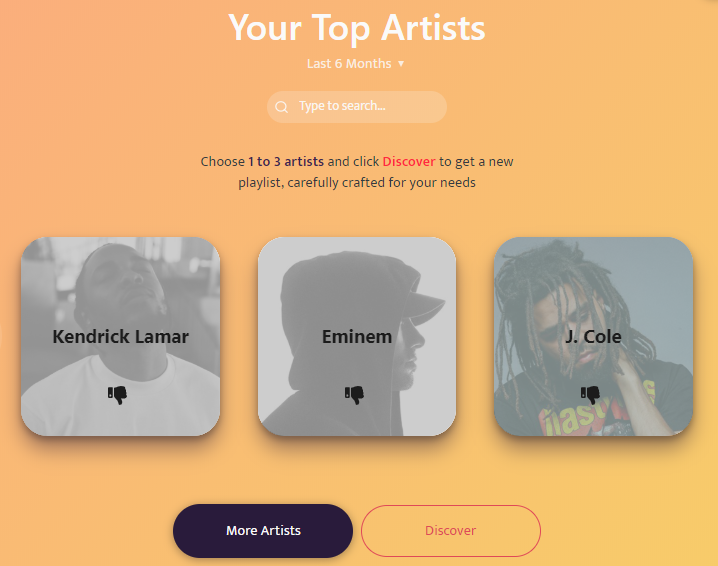


Fig. 9 – Top artists

Cursul acestei pagini este unul pe cât se poate de simplu și constă in selectarea a cel puțin unuia dintre artiștii afișați, ulterior, apăsându-se butonul *Discover*, playlist-ul urmând să fie generat. Privind Fig. 9, se pot observa alte câteva butoane, carduri și un drop-down fiecare având o logică bine definită și fiind responsabilă cu o funcție diferită.

În primă fază, utilizatorul poate selecta perioada din care artiștii să fie selectați. Pentru concretizare, vom presupune următorul exemplu:

* Un utilizator a început de acum o lună să asculte frecvent *Eminem*, dar acum cinci luni obișnuia să asculte *Kendrick Lamar*. Aici intervine filtrul *time-range*, un filtru ce ia trei intervale: ultima lună, ultimele șase luni sau ultimii ani (de obicei acest caz este reprezentat de dorința utilizatorului de a primi sugestii privind gusturile lui muzicale din totdeauna.

Câmpul de căutare este utilizat în cazul în care un alt artist decât cei prezenți in pagină este dorit și se dorește să fie luat în considerare la momentul *discovering*-ului.

Fiecare card ce reprezintă un artist conține, sub numele acestuia, un icon ce are conectată funcția de *dislike* artist. Această funcție adaugă acel artist în tabela *dislikes*, unde putem regăsi toți artiștii ce nu vor mai fi afișați sau luați în considerare în următoarele creări sau descoperiri.

Butonul intitulat *More Artists* este o alternativă căutării manuale prezentate mai sus și funcționează prin afișarea altor artiști decât cei selectați.

***Genre*** are în primă fază un comportament similar modulului *artists*, lipsindu-i câteva detalii deoarece ulterior acesta are un algoritm complet diferit de generare a playlist-ului.

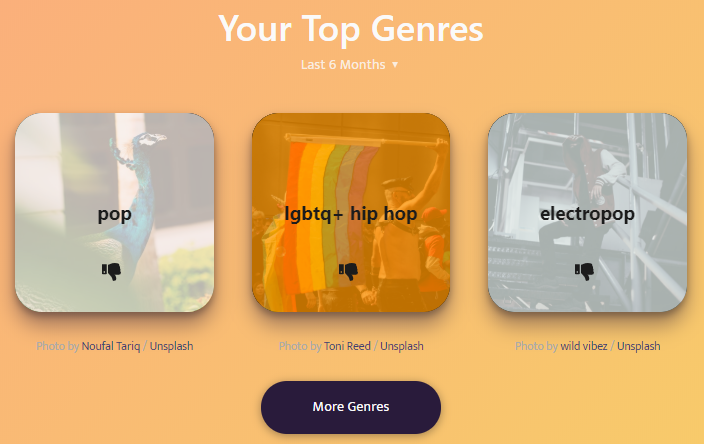


Fig. 10 – Top genres

Spre deosebire de *artists,* în *genres* nu mai regăsim posibilitatea de selecta mai multe genuri muzicale și se merge pe ideea de o singură alegere a unui gen afișat ce va redirecționa către următorul *quiz* de popularitate.

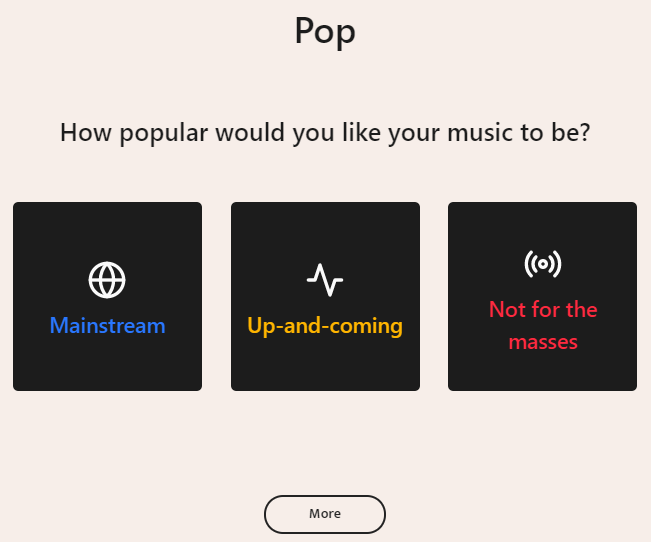


Fig. 11 – Popularity quiz

Acest modal integrat in mod fullscreen oferă utilizatorului posibilitatea de a alege nivelul de popularitate al pieselor ce vor aparține playlist-ului și acestea reprezintă:

* + - *Mainstream –* în principiu aici vor fi piese ce au un nivel de popularitate foarte ridicat, iar playlist-ul va fi unul destul de comun pentru majoritatea celor care selectează acest nivel.
    - *Up-and-coming –* playlist-ul ce se va construi în urma acestei selectări va fi unul populat cu track-uri ce nu sunt încă populare dar cu un potențial ridicat, acest lucru fiind remarcat prin creșterea foarte rapidă a numărului de accesări și ascultări ale pieselor componente
    - *Not for the masses –* este un playlist ce nu este, după cum spune și numele, adresat majorității, fiind astfel opusul *Mainstream*-ului și reprezentând artiștii, genurile muzicale și piesele de nișă.

**Country** are un comportament identic cu *Genre* din punctul de vedere al interfeței, dar din punct de vedere funcțional, aduce șansa utilizatorilor de a explora trend-urile muzicale din majoritatea țărilor, și de ce nu, să studieze cultura și evoluția din țara ce îi stârnește curiozitatea în ceea ce privește planul muzical.

Fiecare *cardview* ce afișează o țară, are ca fundal o imagine ce reprezintă fie un cadru foarte popular, fie o clădire cu impact istoric, fie imagini ce însumează prin unicitate elemente din vegetație sau faună.

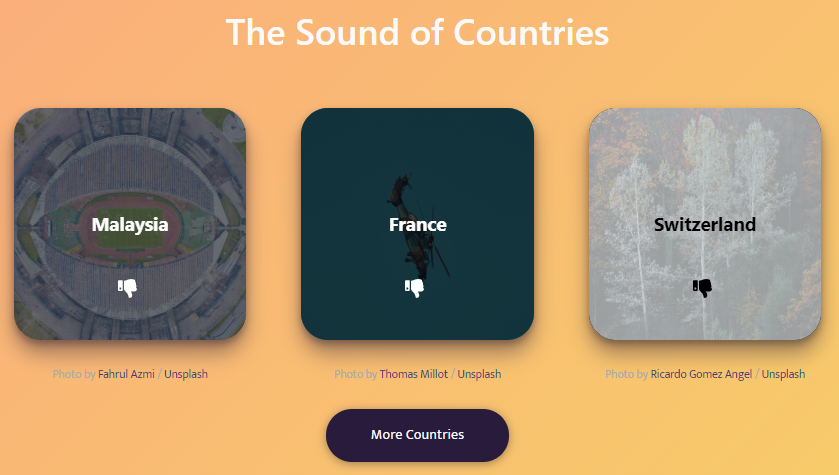


Fig. 12 - Countries

**Cities** oferă funcționalitatea prin care utilizatorul poate studia obiceiurile atât din alte orașe ale propriei tări cât și obiceiurile din orașele altor tări.



Fig. 13 - Cities

Ce se poate observa la prima vedere este înlocuirea *timerange-­ului* cu o listă de țări din care utilizatorul poate selecta nu numai țara din care face parte ci oricare din tările în care Spotify este la dispoziție. Astfel, cu ajutorul acestei funcționalități, se pot observa diferențele de obiceiuri din țară prin crearea unor playlist-uri ce însumează trend-urile momentului din fiecare oraș.

### Playlist

Pagina intitulată *playlist* este cea mai complexă din această aplicație și încheie fiecare curs al selectării preferințelor pe parcursului navigării pe modului *Discover* afișând un playlist ce este compus pe baza artiștilor, genurilor sau trend-urilor preferate.

Pagina este compusă din patru mari componente, fiecare ocupându-se cu un set de funcționalități bine definite și grupate într-un mod ce este plăcut, ușor de înțeles și totodată la îndemână pentru utilizator.

Analizând figura 14, se va prezenta fiecare componentă din pagină și se vor detalia funcționalitățile ce o determină.

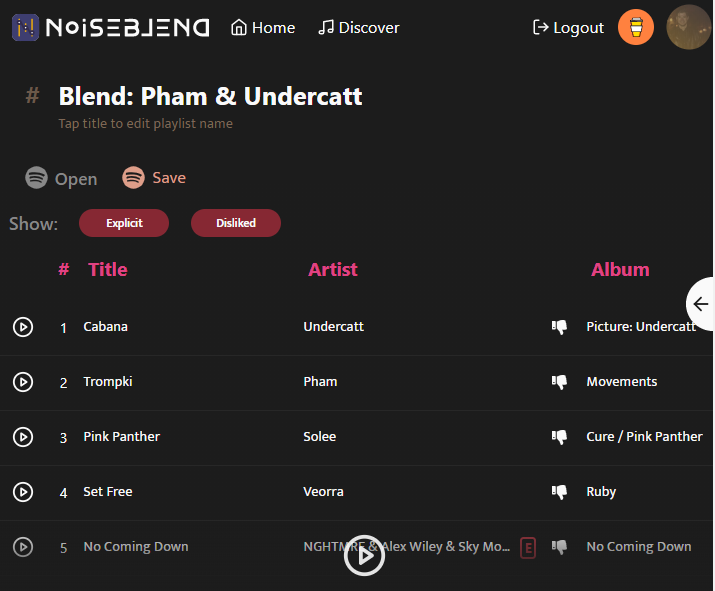


Fig. 14 – Playlist

* Prima componentă a acestei pagini începe cu funcționalitățile de redenumire, salvare și deschidere și salvare a playlistului precum și următoarele filtre reprezentate prin butoanele *Explicit* și *Disliked*:
  + *Open* este butonul ce odată apăsat, va începe redarea pieselor în aplicația web a Spotify-ului.
  + *Save* salvează playlist-ul afișat în contul utilizatorului, astfel acesta avându-l la dispoziție în aplicația Spotify în orice moment.
  + *Explicit*, este un filtru activ a cărui rol este de a elimina elementele din playlist cu un caracter vulgar, și este util momentelor în care se petrece timp cu familia și copiii sunt prin preajmă.
  + *Disliked*, reprezintă butonul ce este responsabil cu eliminarea pieselor ce provin de la un artist ce nu este agreat sau pe placul utilizatorului. În cazul în care o piesă este produsă de mai mulți artiști, un *dropdown menu* va fi afișat ce va conține lista de artiști ai piesei respective și astfel se poate elimina doar artistul sau artiștii ce nu prezintă interes.
* Cea de-a doua componentă a paginii este reprezentată de playlist-ul în sine și este compusă dintr-un tabel cu șase coloane: preview, număr, titlu, explicit, artist și album.
  + *Preview* este acel *play-button* din dreptul fiecărui track și prin apăsarea lui va începe redarea a 30 de secunde din refrenul unei piese. O numărătoare inversă cu startul de la 30 va înlocui butonul de play pe parcursul redării.
  + *Explicit* este flag-ul din dreptul fiecărui titlu ce indică prezența caracterului vulgar. Absența acelui *E* de culoare magenta reprezintă lipsa acestui tip de caracter în piesa de pe acea linie.
* A treia componentă este acel *sidebar* din partea dreaptă a paginii, care în figura 14 este închis și este afișat prin acea săgeată orientată spre stânga.

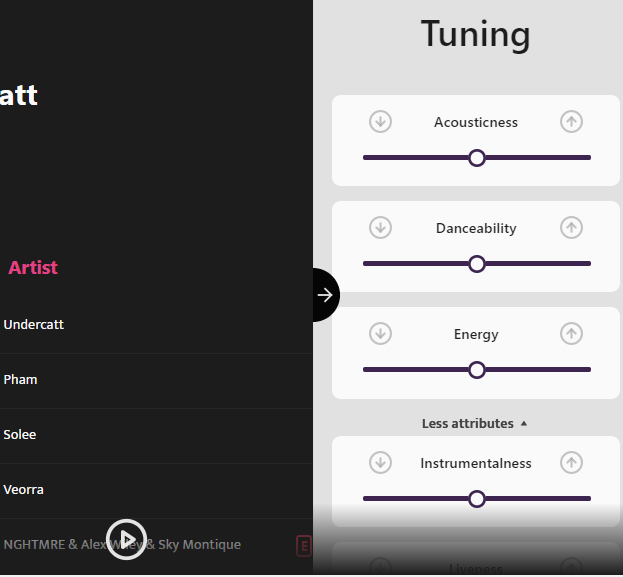


Fig. 15 – Tuning attributes

Acest *siderbar* are la rândul lui două mari funcționalități: cea de sortare bazată pe atributul selectat și cea de reajustare a playlist-ului, eliminând sau adăugând piese ce respectă intervalul selectat. Inițial vor fi afișate primele trei cele mai relevante atribute, acestea fluctuând în funcție de obiceiurile utilizatorului, iar ulterior, butonul *More attributes* extinde lista adăugând restul de încă nouă atribute după care se poate realiza sortarea sau reajustarea playlist-ului.

* Ultima componentă a paginii este disponibilă în partea de jos a paginii și este accesată prin activarea butonului de play in figura 14.

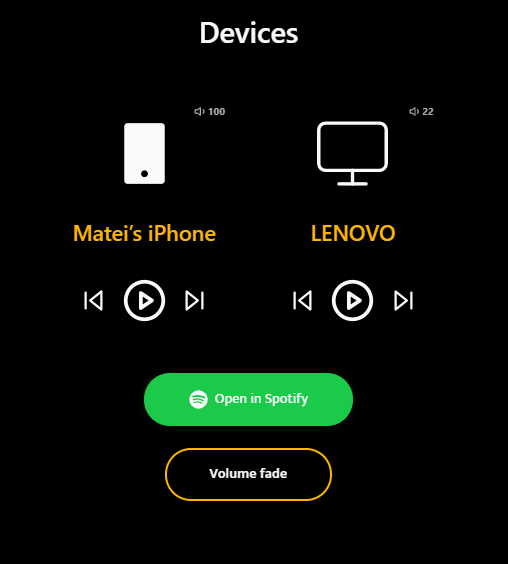


Fig. 16 – Dispozitivele pregătite pentru redare

Prin apăsarea butonului de play se deschide modalul din fig. 16 unde vor fi afișate toate dispozitivele ce au aplicația Spotify deschisă și pregătită pentru redarea playlistului creat. Dispozitivele acceptate pot fi telefoane, laptop-uri/desktop-uri, televizoare sau orice alte dispozitive cu un sistem de operare ce suportă instalarea aplicației Spotify. Fiecare dintre acestea este reprezentat printr-un *cardview* cu o imagină cât mai reprezentativă a dispozitivului disponibil, în partea dreaptă sus fiind prezentă valoarea volumului său, iar în partea de jos cele trei butoane ce permit redarea sau schimbarea pieselor înainte și înapoi. *Open in Spotify* este butonul ce deschide aplicația web nefiind necesară folosirea unuia dintre dispozitivele afișate.

**Volume fade**are în spate implementarea cu rolul de a crește sau a scade nivelul volumului pe un interval setat de utilizator. În figura 16 acest dialog este închis, iar in figura 17 putem observa structura acestuia când este deschis.

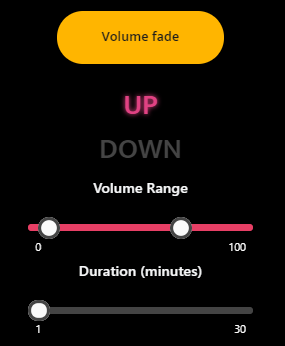


Fig. 17 – Volume fade menu

Prin simpla apăsarea a butonului *Volume fade* se va deschide acest meniu ce pune la dispoziție setarea valorilor următoare:

* Up/down – indică creșterea sau scăderea volumului
* Volume Range – un *range slider* ce preia valorile de început și încheiere a efectului de volume fading
* Duration – valoarea exprimată în minute ce reprezintă durata pe care acest efect să fie efectuat.

Din punct de vedere al implementării, închiderea acestui meniu reprezintă ignorarea valorilor setate și a întregului efect de *volume fade.*

### Profile

În header-ul acestei pagini se regăsesc imagini despre utilizator, nume, țara configurată în Spotify și email-ul asignat contului.

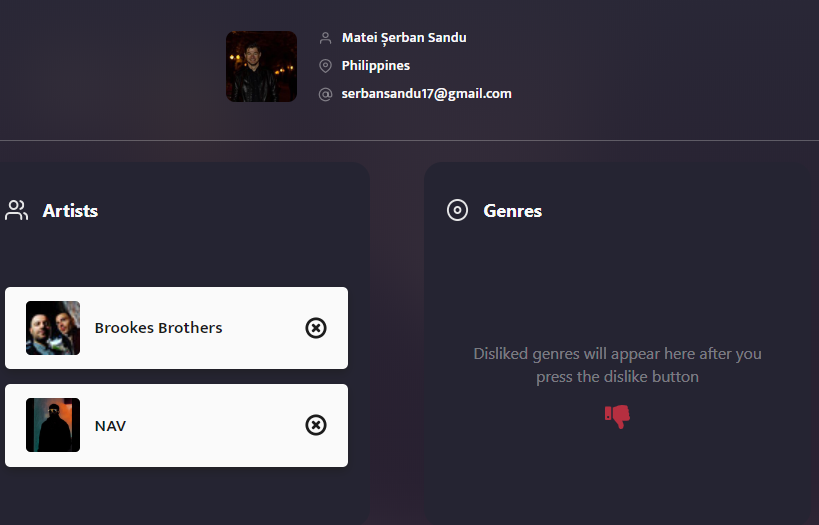


Fig. 18 – User profie

În partea de jos a acestei pagini se regăsesc patru liste cu numele celor patru module din *Discover* și sunt populate cu artiștii, genurile, țările respectiv orașele ce nu sunt pe gustul utilizatorului (în figura 18 sunt afișate doar două dintre acestea pentru a evita atașarea unei imagini de o dimensiunea prea mare).

# Concluzii

Dezvoltarea aplicației *Noiseblend* a fost concepută strict cu scopul de a oferi noi metode de success în căutarea muzicii, iar pe parcursul dezvoltării acesteia am considerat că ar fi util să consult și alte idei, astfel am ascultat opiniile populare ale utilizatorilor forumului Spotify, unele dintre ele fiind implementate astăzi chiar în această aplicație*.*

În dezvoltarea acestei aplicații am întâlnit o multitudine de dificultăți, atât pe back-end cât și pe front-end. Cele mai importante dintre acestea au fost întâmpinate la crearea algoritmului pentru *blend*, unde aplicarea atributelor pe playlist, ce trebuia să respecte caracterul și scopul blend-ului, a necesitat un efort deosebit. O altă provocare a fost întâlnită la crearea paginii *playlist*, unde erau necesare o multitudine de detalii și toate acestea trebuiau implementate cu multă atenție și pe partea de logică dar și design.

Atât din perspectiva implementării algoritmilor, cât și din perspectiva design-ului, consider că aplicația este un succes total și datorită caracterului modularizat, aceasta are un potențial foarte ridicat de dezvoltare, adăugarea de noi funcționalități devenind realizabilă fără efort.

O îmbunătățire a acestui proiect, cel puțin din punctul meu de vedere, ar fi crearea unui nou algoritm de învățare automată ce categorizează toată gama de gusturi ale utilizatorului pe genuri sau grupuri de artiști și oferă astfel sugestii independente și neinfluențate de restul genurilor sau artiștilor.

Dacă sunteți un utilizator al aplicației Spotify, vă considerați pasionat de muzică și adesea rămâneți fără idei în căutarea muzicii potrivite, *Noiseblend* este exact ceea ce urmăriți.

# Bibliografie

Cărți:

[1] Beazle David, Jones K. Brian(2013) „Python Cookbook 3” O’Reilly Media,Inc

[2] Dr. Rauschmayer Axel (2014) „Speaking JavaScript: An In-Depth Guide for Programmers” O’Reilly Media, Inc

Link-uri:

[1] Spotify Web API Reference - <https://developer.spotify.com/documentation/web-api/reference/>

[2] Asyncio - <https://docs.python.org/3/library/asyncio-task.html>

[3] Sanic - <https://sanic.readthedocs.io/en/latest/>

[4] Redux - <https://redux.js.org/>

[5] Asyncpg - <https://magicstack.github.io/asyncpg/current/>

[6] Redux-saga - <https://engineering.universe.com/what-is-redux-saga-c1252fc2f4d1>

[7] Next.js - <https://nextjs.org/>

[8] Coffeescript - <https://coffeescript.org/>

[9] React.js - <https://reactjs.org/>

[10] Progressive Web Apps - <https://developers.google.com/web/progressive-web-apps/>

[11] Imagini de fundal: <https://unsplash.com/>

[12] Icons/SVGs :

* <https://origin.fontawesome.com/>
* <https://realfavicongenerator.net/>