UNIVERSITATEA TEHNICĂ
DIN CLUJ-NAPOCA

# *MUSIC APPLICATION*

Remeș Daria-Maria
Strugar Mădălina-Alexandra

Ianuarie 2024

# Contents

# 1. INTRODUCERE

## 1.1 Descriere

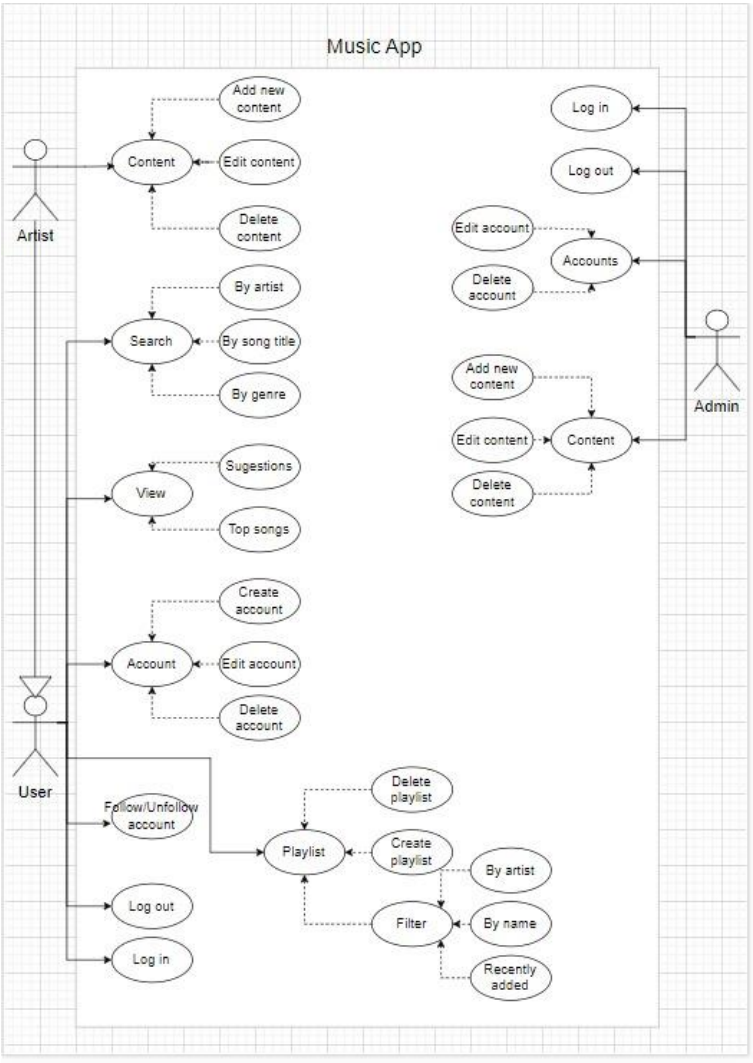
Obiectivul acestui proiect a fost implementarea unei aplicații pentru muzică online. Aceasta oferă posibilitatea înregistrării a două tipuri de utilizatori: artist și utilizator ordinar și existența unui administrator, care au acces la diverse funcționalități.

Administratorul poate să gestioneze crearea, editarea și ștergerea cântecelor cât și editarea și ștergerea conturilor deja existente.

Artiștii și utilizatorii ordinari au câteva funcționalități de bază cum sunt crearea, ștergerea și filtrarea unui playlist, editarea și ștergerea contului, vizualizarea în home page a unor sugestii și topuri muzicale.

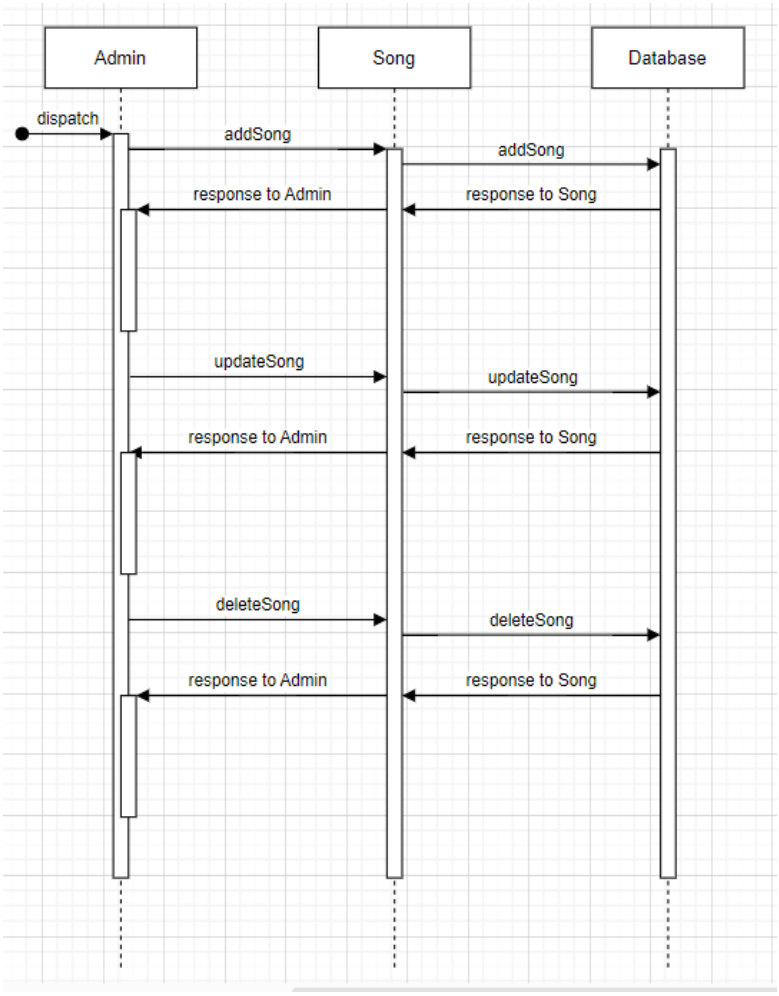
Utilizatorii de tip artist se diferențiază de cei ordinari prin posibilitatea adăugării de conținut: adăugare, editare și ștergere cântec.

## 1.2 Diagramă use case

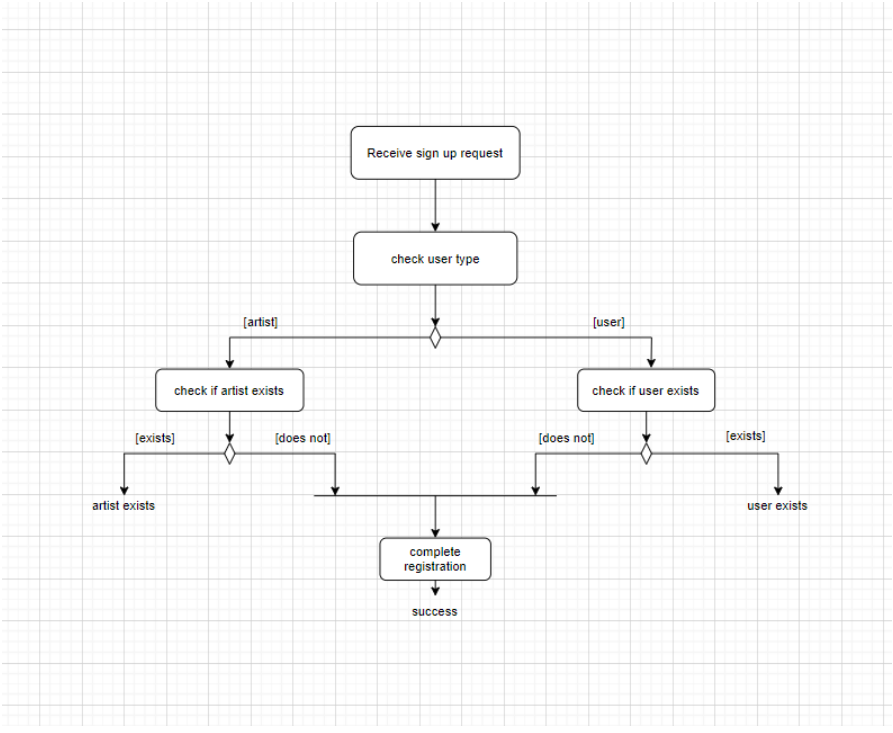


# 2. DIAGRAME

## 2.1 Diagramă secvențială




## 2.2 Diagramă de activitate

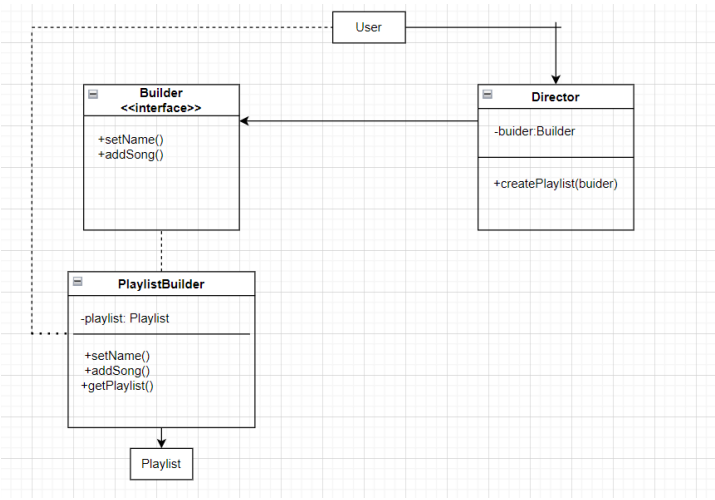


# 3. DESIGN PATTERN

**Builder** este un șablon de proiectare creațional care vă permite să construiți obiecte complexe treptat.

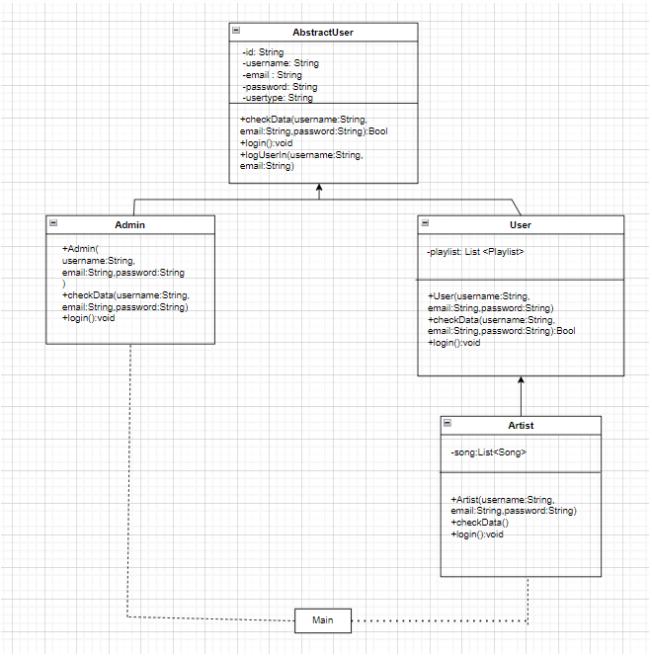
Problema pe care o rezolvă șablonul de proiectare Builder este gestionarea construirii și inițializării obiectelor complexe, în special a celor care necesită o serie de pași laborioși și pot avea o varietate de configurații. Aceasta împiedică construirea unui constructor cu un număr mare de parametri sau răspândirea codului de inițializare în mod dispersat în întregul cod client.

Soluția adusă de șablonul Builder constă în separarea procesului de construcție a obiectului de reprezentarea acestuia. Acesta implică utilizarea unui director care coordonează construcția și un builder care implementează pașii concreți de construcție.




Metoda **Template** se referă la faptu că se utilizează un schelet de algoritm în superclasă care urmează să fie implementat în subclase, fără a modifica structura.

Evită duplicarea codului și consolidează logica algoritmică.



Am ales aceste design pattern-uri în momentul proiectării aplicației, însă pe parcursul dezvoltării acesteia am ajuns la concluzia că nu se potrivesc cu modul în care am gândit funcționalitatea finală, astfel nu am reușit implementarea pattern-urilor.

# 4. PROIECTARE

## 4.1 Tehnologii

Pentru dezvoltarea acestui proiect am utilizat Java, Java Spring, Spring Boot, MySQL, React (JavaScript, CSS, HTML).

**Java** este un limbaj de programare orientat pe obiect, cu un syntax simplu și clar, potrivit pentru diferite platforme. De obicei este folosit pentru a scrie parte de server (backend) a software-ului. Câteva caracteristici cheie ale acestui limbaj sunt:

- Tipizarea rigidă: toate variabilele sunt declarate cu specificarea tipului lor. Acest lucru previne erorile de tipizare, permite scrierea de cod clar și identifică erorile în timpul compilării, nu în timpul execuției.
- Gestionarea automată a memoriei. Limbajul Java implementează colecatrea automată a gunoiului, eliberând memoria ocupată de obiecte care nu mai sunt utilizate. Dezvoltatorii nu mai trebuie să facă acest lucru.
- Orientare pe obiect. Java se bazează pe conceptele programării orientate pe obiect, ceea ce înseamnă că totul în java este un obiect, cu proprietățile și metodele sale. Abordarea orientată pe obiect permite dezvoltatorilor Java să creeze aplicații modulare, felxibile și sigure.

**Spring Framework** este o platformă Java opensource care oferă un suport vast pentru crearea unei infrastructuri de dezvoltare a aplicațiilor Java. Spring structurează programul și legăturile dintre entități, ușurând lucrul programatorului. Este bazat pe principiul inversion of control (IoC) care are scop inversia fluxului tradițional de executare a programului astfel fluxul programului fiind setat de framework. Într-un program obișnuit programatorul hotărăște în ce ordine vor fi apelate metodele, pe când în unul controlat de framework programatorul definește și implementează metodele rulând doar funcția principală a framework-ului acesta apelând singur metodele definite de către programator atunci când este nevoie de ele.

Avantaje:

- Template predefinite pentru utilizarea diferitor tehnologii ca JDBC, Hibernate, JPA etc
- Ușor de testat.

**Spring Boot** reprezintă o extensie a framework-ului Spring care permite eliminarea configurărilor de nivel inferior, acestea fiind făcute automat prin alegera unor opțiuni de către programator la inițiarea aplicației. Permite accelerarea creării unei aplicații web, aceasta analizând configurările setate de către programator și injectează toate dependențele necesare pentru satisfacerea cerințelor acestuia. Spring Boot oferă posibilități simple de introducere a modulelor prin pom.xml file, prin Spring Intializr și de pe siteul oficial, permițând ușurarea lucrului programatorului, acesta având nevoie să se asigure doar de funcționalitatea programului.

Avantajele de baza a Spring Boot framework sunt auto-configurarea, independența și autonomia acestuia de a lua decizii pentru a face o aplicație cât mai optimizată.

**MySQL** este un Sistem de Gestiune a Bazelor de Date Relaționale (RDBMS) cu sursă deschisă care permite utilizatorilor să stocheze, să gestioneze și să recupereze datele structurate eficient. Este larg utilizat pentru diverse aplicații, de la proiecte de scară mică la site-uri de scară mare și soluții la nivel de întreprindere.

O bază de date este o structură de date organizate și stocate în tabele. Ea servește ca un depozit central în care informațiile sunt gestionate efficient, permițând utilizatorilor să stocheze, recupereze, actualizeze și șteargă date. MySQL furnizează cadrul software pentru a crea, menține și interacționa cu aceste baze de date, facilitând stocarea și recuperarea datelor fără probleme.

**React** sau React.js este o bibliotecă JavaScript utilizată pentru crearea interfețelor utilizator web dinamice și interactive. A fost dezvoltată de Facebook și este folosită pentru construirea de aplicții web moderne. Este construită în jurul unui concept numit "componente", care reprezintă bucăți de cod reutilizabile și independent gestionate, ceea ce face ca dezvoltarea aplicațiilor wb să fie mai ușoară și mai eficinetă.

**JavaScript** (**JS**) este un limbaj de programare orientat obiect bazat pe conceptul prototipurilor. Este folosit mai ales pentru introducerea unor funcționalități în paginile web, codul JavaScript din aceste pagini fiind rulat de către browser. Limbajul este binecunoscut pentru folosirea sa în construirea siteurilor web, dar este folosit și pentru accesul la obiecte încapsulate (embedded objects) în alte aplicații.

Cea mai des întâlnită utilizare a JavaScript este în scriptarea paginilor web. Programatorii web pot îngloba în paginile HTML script-uri pentru diverse activități cum ar fi verificarea datelor introduse de utilizatori sau crearea de meniuri și alte efecte animate.

**HTML** și **CSS** sunt două limbaje ce stau la baza creării design-ului unei pagini web. Cu HTML se creează structura paginilor web, se adaugă texte, imagini, formulare, iar cu CSS se obține stilizarea elementelor HTML. Odată ce un design a fost creat, acesta poate fi legat de o aplicație web implementată în orice limbaj de programare.

## 4.2 Baza de date

Baza de date a fost creată utilizând MySQL. Aceasta conține tabelele admin, artist, user, songs, playlist. Pentru a conecta serverul la baza de date am folosit aceasta depedinta si in application.properties am notat datele despre baza de date, care se numeste musicapp2 si utilizatorul din MySQL Workbench. Singurul lucru pe care l-am facut in MySQL Workbench a fost sa creem baza de date.

```
<dependency>
    <groupId>com.mysql</groupId>
    <artifactId>mysql-connector-j</artifactId>
    <version>8.2.0</version>
    <scope>runtime</scope>
</dependency>
```

```
create schema musicapp2;
```

application.properties

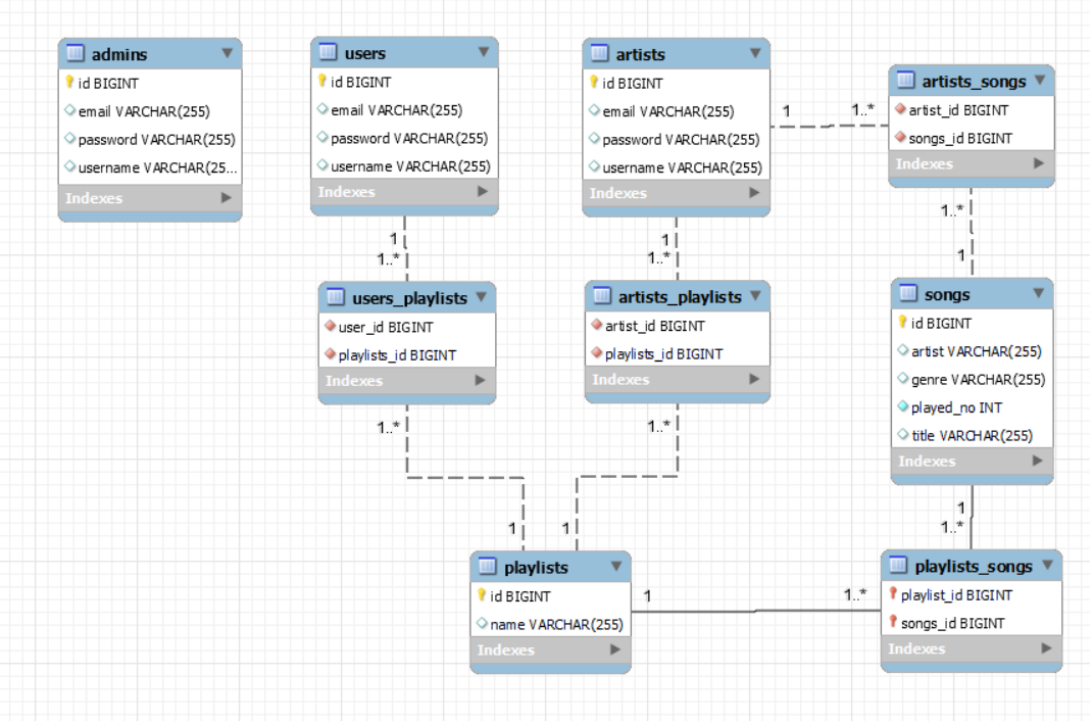
```
server.port=8080

spring.datasource.url=jdbc:mysql://localhost:3306/musicapp2
spring.datasource.username=root
spring.datasource.password=Inginer2023*

spring.jpa.hibernate.ddl-auto=update
```

Aici stocam datele despre utilizatori(admin, utilizatori si artisti), cantece si despre playlist-urile utilizatorilor. Pentru a crea legaturi intre tabele, precum faptul ca un playlist poate contine mai multe cantece, am folosit relatia one-to-many care ne spune ca un playlist poate avea mai multe cantece. Legatura propriu zisa este realizata cu ajutorul unui tabel intermediar care contine perechi de chei straine care reprezinta cheia primara a playlist-ului si al cantecului, astfel ca stim ce cantece contine fiecare playlist. Acelasi concept se aplica si relatiei intre playlist-uri si utilizatori sau intre artisti si cantecele proprii.

Pentru a realiza relatiile one-to-many am utilizat notatie @OneToMany oferita de Spring Data JPA.




## 4.3 Backend

Backend-ul reprezintă partea dintr-un sistem software responsabilă de gestionarea logicii de afaceri, stocarea și manipularea datelor, precum și comunicarea cu componentele frontend și alte servicii externe. Aceasta parte a proiectului a fost scrisa in limbajul de programare Java si a fost dezvoltat cu framework-ul Spring.

Pentru a organiza codul sursă al proiectului, am împartit clasele în pachete (packages) pentru a menține o structură ordonată și ușor de gestionat. În cadrul acestui proiect, pachetele includ categorii precum model, repository, service, serviceimpl, și controller.

Pachetul model este destinat să conțină clasele care definesc structura obiectelor de date ale aplicației. Aceste clase reprezintă entitățile din domeniul de business al aplicației și pot fi asociate cu tabelele din baza de date în cazul unui proiect Java Spring cu persistență în baza de date. Spre exemplu clasele Song, Playlist, User, etc.

Pachetul repository conține interfețele care se ocupă de accesul la baza de date. Acestea oferă metode pentru a realiza operații CRUD (Create, Read, Update, Delete) asupra obiectelor stocate în baza de date.

Pachetul service găzduiește interfețele care definesc operațiile de business ale aplicației. Acestea acționează ca un nivel intermediar între repository și controller, implementând logica de afaceri.

Pachetul serviceimpl conține implementările concrete ale interfețelor din pachetul service. Aceste clase implementează logica specifică pentru operațiile de business și pot utiliza repository pentru a accesa datele din baza de date.

Pachetul controller conține clasele care gestionează cererile HTTP și interacțiunea cu frontend-ul. Aceste clase utilizează serviciile pentru a obține și a manipula datele și returnează răspunsuri HTTP.

Pachetul dto contine clasele care ne ajuta sa separam obiectele folosite in logica de business de cele utilizate pentru a primi date de la front end.

## 4.4 Frontend

Frontend-ul unei aplicații este responsabil pentru descărcarea datelor de la utilizator în diferite forme și transferarea acestora către backend. Apoi backend-ul bazat pe aceste date îndeplinește o sarcină specifică. Opțional, partea frontal poate afișa utilizatorului rezultatele obținute de la backend.

Această parte a proiectului a fost implementată utilizând React, o bibliotecă JavaScript eficientă, pentru construirea componentelor reutilizabile ce facilitează dezvoltarea unei interfețe coerente și dinamice. Axios a fost integrat pentru gestionarea cererilor HTTP, asigurând comunicarea eficientă cu serverul și actualizarea continuă a datelor. Designul vizual al aplicației a fost definit cu ajutorul CSS și HTML, aducând o prezentare atractivă și responsivă. Dezvoltarea în Visual Studio a oferit un mediu integrat puternic, facilitând procesul de codare, testare și depanare.

Structura src a proiectului cuprinde două două substructuri importante: components și services.

Folderul components conține implemenatrea pentru componentele aplicației. Acestea sunt entități UI reutilizabile, cum sunt bara de navigare, homepage și toate celelete elemete de interfață vizuală.

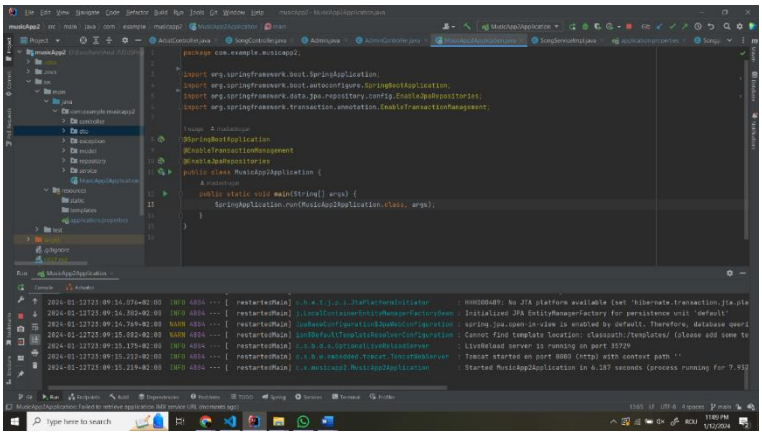
Folderul services cuprinde fișierele care gestionează interacțiunea cu backend-ul prin metodele Axios. Include funcționalități precum preluarea și gestionarea datelor din server. Organizarea acestui folder ajută la separarea clară a responsabilităților între componentele UI și logica de comunicare cu backend-ul.

Fișierul App.jsx funcționează ca punct de intrare principal al aplicației. Conține rute necesare pentru inițializarea și funcționarea aplicației.

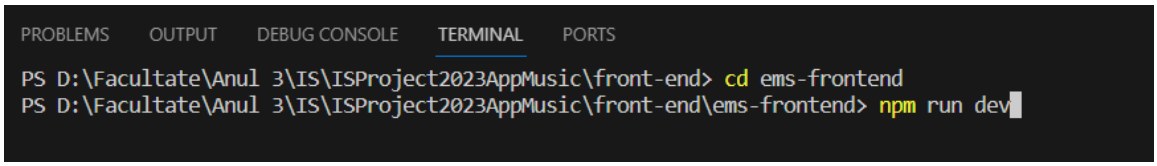
Această structura modulară și organizată este esențială pentru o dezvoltare eficientă și ușor de întreținut.

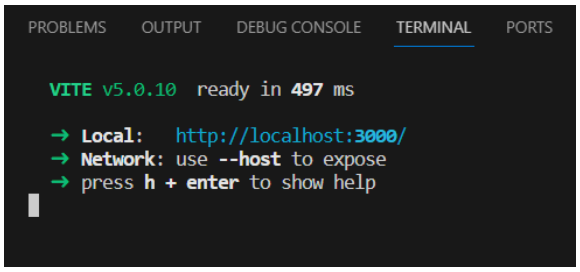
# 5. Read me

Partea de backend a aplicației a fost realizată in Java cu framework-ul Spring, astfel ca avem nevoie de un IDE specific Java, spre exemplu Intellij IDEA. Pentru a rula partea de server a aplicatiei, vom rula clasa MusicApp2Application, care contine metoda run specifica Spring.

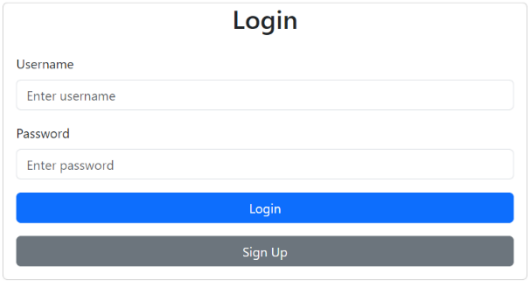


Partea de frontend a fost facuta in React JS, iar ca si IDE am utilizat Visual Studio Code. Pentru a rula partea de client a aplicatiei, vom deschide terminalul si vom introduce aceste doua comenzi




Acum putem deschide aplicatia intr-un browser web.


## 6. Concluzii și îmbunătățiri ulterioare

Parcursul implementării a fost unul interesant, am întâmpinat dificultăți datorită faptului că nu am mai lucrat cu unele dintre tehnologiile alese, însă am reușit să le depășim împreună.

Funcționalitatea finală a aplicației include posibilitatea utilizării unori melodii reale cu sunet, parte la care nu am reușit să ajungem. De asemenea, ramura de search și follow/unfollow între useri mai trebuie dezvoltată., iar userii încă nu au acces la editarea și ștergerea contului.