MusicDB

A project to store and manage music metadata.

Project Requirements

1. **Add a New Track:** As a user, you should be able to add a new track to an artist's catalogue, capturing attributes such as track title, genre, length, etc.
2. **Edit Artist Name:** As a user, you should be able to edit an artist's name to accommodate instances where artists have multiple aliases.
3. **Fetch Artist Tracks:** As a user, you should be able to fetch all tracks associated with a specific artist.
4. **Artist of the Day:** As a user, you should be able to see a different "Artist of the Day" in a cyclical manner on the homepage each day, ensuring a fair rotation through the entire catalogue of artists. This means if there are n artists, after n days, the cycle restarts with the first artist, ensuring an equal chance for each artist to be the "Artist of the Day".

Possible Solution

Phase 1: Backend

We are going to create following APIs

- Get /track // To get list of all tracks or filter using query
- Get /track/{id} //To get track by id
- Post /track // To create a new track
- Patch /track/{id} // To edit track metadata
-
- Get /artist // To get a list of artist or filter using query
- Get /artist/{id} // To get a list of artist by id
- Post /artist // To create an artist
- Patch /artist // To update an artist
- Get /artist/aotd // To get a artist of the day
- Get /artist/{id}/tracks // To get all tracks of a artist by id
-
- Patch /alias/{artist_id}add/{name} to add a alias
- Patch /alias/remove/{id} to remove a alias by id
-
- Patch /collab/{track_id}/add/{artist_id} // To add a musicians to a track
- Patch /**collab**/{track_id}/add/{artist_id} // To remove a musicians from a track

Phase 2: Frontend

We are going to create following pages

- Home Page: Artist of the day and links to following pages :
- Track Table View: A search bar, A list of tracks
- Track Add/Edit View: A page to edit/add a track
- Artists Table A search bar, View: A list of artists
- Artist Add/Edit View: A page to edit/add artist
-

Software Stack & Requirements for development

Backend

Language: Java 17
Framework: Spring Boot 3.1.3
Database: MySQL

Frontend

Language: TypeScript
Framework: Angular: 16.2.5
Node: 18.17.1
Package Manager: npm 9.6.7

Other Tools

Version Management: Git
Container: Docker

Other Considerations:

1. I assume that we are going to have only few artists in our system (< 300) because with given logic it will take 300 days before last artist's turn.

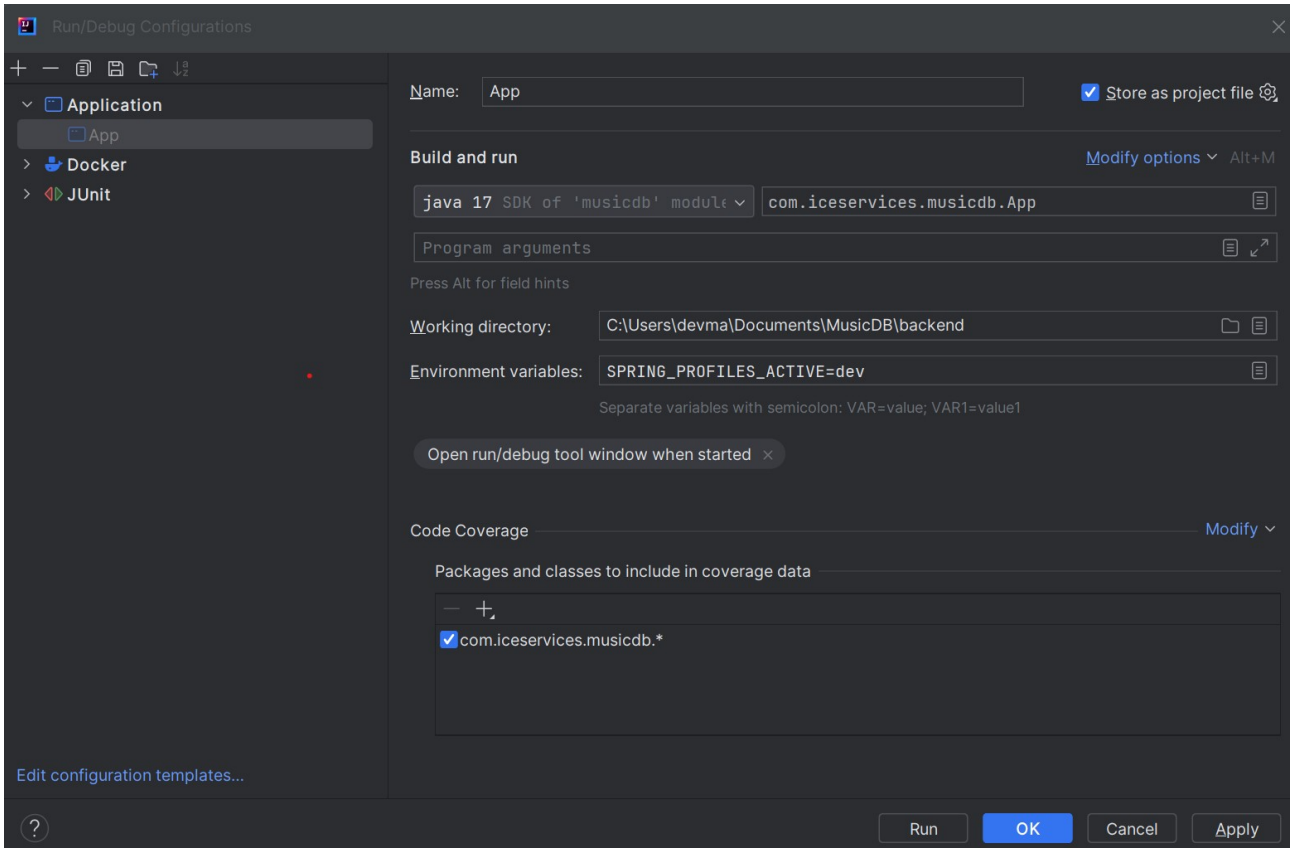
Possible Future Improvements

1. Albums – Options to have albums as entity and group tracks in album.
2. Band – Options to have band as entity and group artists in bands.

Guide for local system development.

Backend

1. Open backend folder in your preferred IDE, this guide follow ***intellij,*** but other IDE should work similarly.
2. Open ***backend/src/main/java/com/iceservices/musicdb/App.java***
3. You should be able to run the file in your IDE using a configuration like this :



4. **Don’t forget to add SPRING_PROFILES_ACTIVE=dev**
5. You can edit ***backend/src/main/resources/application-dev.properties*** to change setting to connect to a local database instead of using H2.

If you just want to run the project and don’t want to open it, you can also use CLI.

1. Open the terminal and go to inside backend folder.
2. Run : ./mvnw spring-boot:run -Dspring-boot.run.profiles=dev

Note : if you are using powershell on windows (default in intellij termianl) then give space after -D

./mvnw spring-boot:run -D spring-boot.run.profiles=dev

Testing:

A postman collection is included in same location as this file to test the APIs.

Frontend

1. [Optional] You don’t have to open the frontend in IDE but if you want to check the code you can use your preferred IDE.
2. Open terminal and go to frontend folder.
3. Run : ng serve --open

Testing:

After above command is successful you can open http://localhost:4200/ to test the application.

Docker

Dockerfiles are included in both backend and frontend and these can be used to create docker images.

docker run -dp 4200:80 music-db-frontend
docker run -dp 8080:8080 music-db-backend