Songify Design Document

Issue Record

Version	Date	Notes
0.1	18/3/21	Added ERD, frontend framework research, explanation on design choices and how SOLID is guaranteed

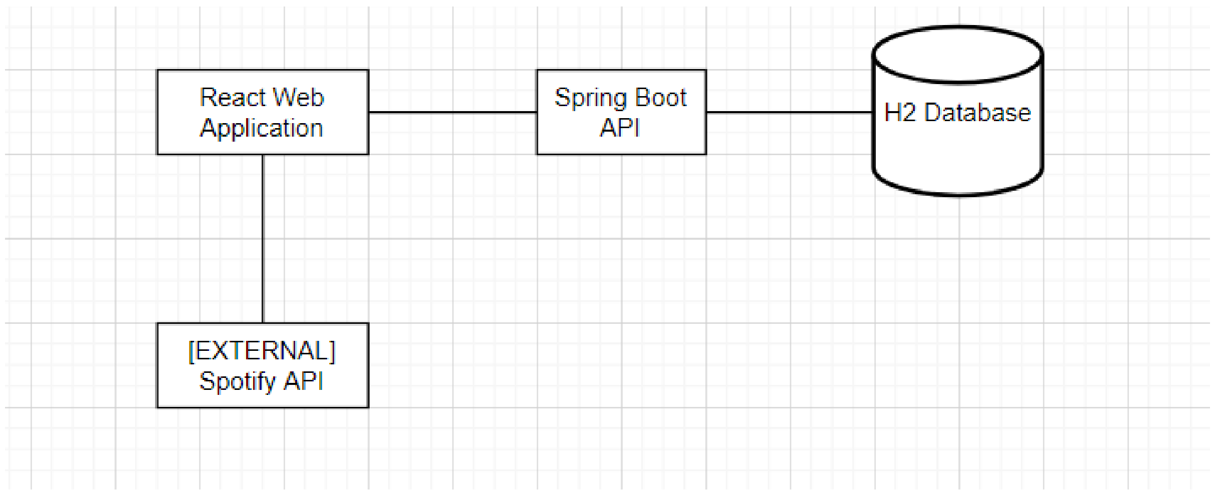
Table of Contents

Introduction

The project Songify aims to create a web application that anyone can use to create and share playlists with friends. In this document the architecture of the applications involved will be discussed, and the choices made during the making of them will be justified.

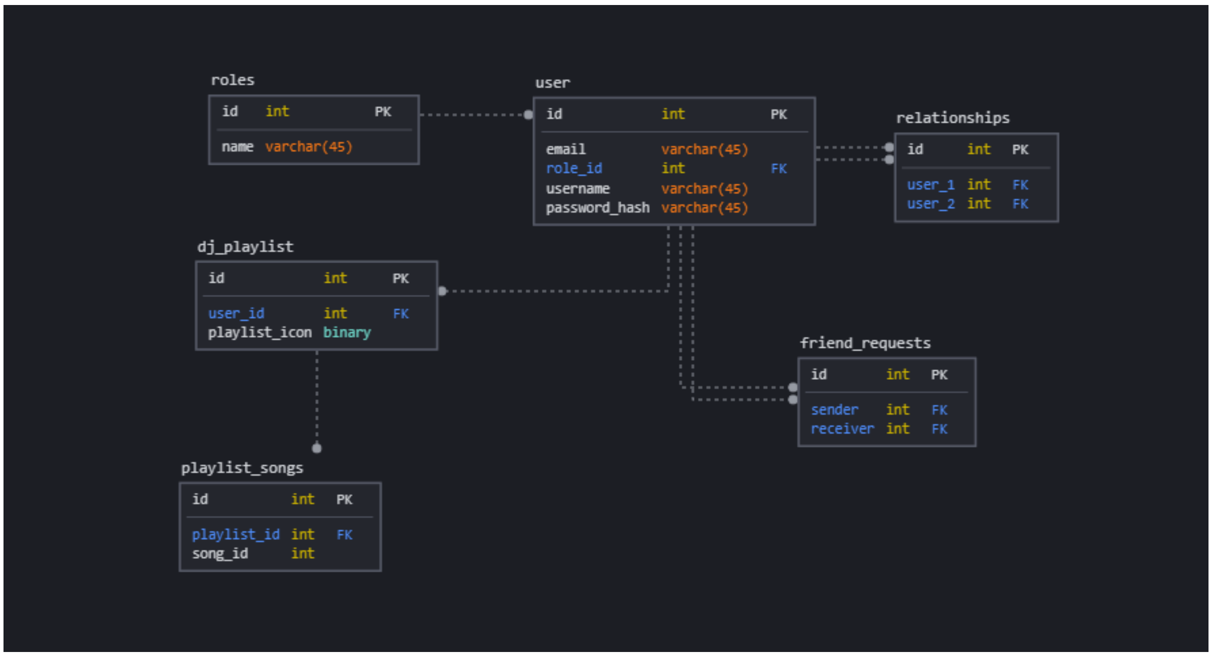
Architecture

Basic Architecture



In the diagram above the basic architecture of the project is illustrated. The reasons behind this particular structure will be discussed in this document

Entity Relationship Diagram



In the ERD above the entities currently in the system are shown. Because the songs are retrieved via the Spotify API, they are not listed as a foreign key anymore in the playlist_songs table. The roles table will for now suffice in handling the difference between regular users, admins and DJ's, but in later iterations bearer authentication could replace this system. Lastly, the relationships table will for now only store friendships (coming from the friend_requests table) but in later iterations could also be used to store other relationships such as followers.

SOLID Principles

To ensure solid, a 3 layered design has been implemented. The presentation, logic and data access layer. The logic and data access layers are within the Spring boot application while the presentation layer is the react application.

Design Choices

Backend

Why Spring Boot?
Spring Boot significantly decreases development time because it has a default setup for tests and it comes with many dependencies that can be plugged into the Spring application. Another option was to create the API with ASP.NET Core. I have previous experience with this but because of this I decided to take on a new challenge in Spring boot.

Why H2 Database?
H2 is in-memory therefore it will not persist. Because it is embedded it is a great option for development and testing. Once the application is ready for production/release, H2 database will be a less practical option and will likely be swapped for another database engine. When the time comes I plan to use MySQL because of previous experience and it is reliable.

Frontend

For this project I have chosen React.

	Angular	React	Vue
Ease of Learning	Medium	Medium	Easy
Performance	Good	Good	Good
State Management	✓	✓	✓
Form validation & handling	✓	✗	✗
Routing	✓	✗	✓
Language	TS	JS/JSX	JS/JSX
Personal experience	Minor	None	None

In the table above are the most important factors for me when choosing a framework, with performance and previous experience being the least important. At times, Angular can feel a bit big, because there are many features built into Angular which from personal experience I have never needed. Because of this and the fact that Vue has a small number of components/plugins, I chose React. In the table you may have seen that React lacks form validation and routing, but this isn't an issue because third-party libraries such as Redux and Formik can be used to implement these features. The option to add these third-party libraries is another good reason I chose React.