Spotify 2.0

Dokument Projektowy

Ogólny opis wykonywanego projektu

Spotify 2.0 to platforma streamingowa oferująca darmowy dostęp do muzyki oraz podcastów, udostępnianych przez twórców z całego świata. Korzystanie z serwisu jest możliwe za pośrednictwem aplikacji przeglądarkowej. Oferuje ona możliwość swobodnego przeglądania oraz wyszukiwania utworów bez konieczności instalacji dodatkowego oprogramowania. Użytkownik po zalogowaniu ma możliwość personalizacji profilu oraz tworzenia playlist'y z wybranymi utworami. Utworzenie konta oraz korzystanie z serwisu jest całkowicie darmowe. Platforma skierowany jest głównie dla użytkowników szukających darmowego oraz intuicyjnego rozwiązania do słuchania ulubionych utworów.

Autorzy projektu

- Damian Moczybroda
- Filip Krasiński
- Kacper Witek
- Patryk Wojtiuk

Opis motywacji powstawania projektu

Powstanie projektu rozwiązuje problem z publikowaniem utworów muzycznych przez ludzi jak i również ich odsłuchiwaniem przez każdego z użytkowników aplikacji. Codziennie miliony osób słucha utworów muzycznych które nie zawsze można łatwo odszukać, stworzenie projektu pozwalałoby na łatwiejszy dostęp do źródeł co poprawiłoby komfort użytkowania oraz samopoczucie wielu ludzi jak i również możliwość składowania własnych utworów co poprawiłoby łatwiej osiągnąć sukces muzyczny, ale również pomogłoby osobom dopiero zaczynającym, ponieważ mogliby opublikować swoje dotychczasowe lub przyszłe utwory muzyczne, w związku z tym każdy z użytkowników mógłby ich odsłuchać oraz docenić ich twórczość. Na podstawie przeprowadzonego sondażu w 2018 roku w Polsce wyniki prowadzą do tego, iż 65% osób dorosłych słucha muzyki codziennie co pokazuje prawdziwy zakres możliwego występowania problemu. Nasze rozwiązanie usprawniłoby codzienne życie ludzi, poprzez ułatwienie do ich rutynowych nawyków słuchania muzyki.

W realizowanym projekcie kolaboranci uzyskają wiele pozytywów w zakładanej części pozafinansowych, korzyści z uczestnictwa w projekcie pokazują możliwość rozwoju jak i samodoskonalenia swoich umiejętności np. w zakresie wytwarzania oprogramowania, kompetencji społecznych, jak i również nauczenia się nowych np. współpracy z zespołem projektowym. Członkowie projektu odczują poprawę swoich możliwości realizacji projektów w zakresie IT poprzez dyscyplinarne ocenianie prac przez członków zespołu, co umożliwi wielopoziomowe podejście do przedstawionego rozwiązania problemu oraz pokaże możliwe sposoby realizacji zdefiniowanego problemu przez innych członków zespołu.

Cel projektu

Celem projektu jest zapewnienie łatwy dla użytkowników dostęp do muzyki, odtwarzania jej jak i wrzucania i dzielenia się swoimi utworami. Nasz serwis ma na celu udostępnić prosty w obsłudze kreator playlist i odtwarzacz muzyki, ma też udostępnić prosty interfejs przesyłania własnych utworów.

Opis problemu

Głównym problemem który chcemy rozwiązać to brak prostych w obsłudze serwisów do odtwarzania muzyki. Chcemy stworzyć serwis który jest przeznaczony do tego czego miał być od początku bez zbędnych funkcjonaliści dzięki temu będzie banalnie prosty a użytkownik szybko nauczy się jego obsługi. Wiele serwisów również nie daje prostego sposobu na dzielenie się swoimi utworami, nasza aplikacja udostępni taką możliwość.

Opis wykorzystywanych technologii

Nasza aplikacji będzie korzystać z technologii takich jak Spring Boot, Angular oraz PostgresSQL.

Spring Boot jest frameworkiem bazującym na Springu który pozwala na tworzenie prostych jak i rozległych aplikacji w prosty sposób. Dostarcza szereg rozwiązań które mają pomóc w szybkim i wydajnym tworzeniu oprogramowania. Posiada wbudowany serwer i do samego uruchomienia zasadniczo nie jest wymagana żadna konfiguracja. Opiera się on na kontenerze IoC który pozwala na łatwe przechowywanie obiektów i zarządzanie nimi w całym cyklu życia aplikacji.

Angular to JavaScript'owy framework służący do łatwego i szybkiego budowania aplikacji typu SPA (Single Page Application). Jest on napisany w języku TypeScript. Tak jak w przypadku Spring Boot'a nie wymaga on żadnej konfiguracji i zawiera szereg paczek posiadających różne funkcjonalności które można dodać w dowolnym momencie do projektu.

PostgresSQL jest połączeniem relacyjnej i obiektowej bazy danych, pozwala na tworzenie tabel w których zawarte są dane i relacji między nimi.

Cele użycia technologii

- Spring Boot zostanie wykorzystany na backend'zie do tworzenia endpoint'ów i zarządzania danymi, tam zostanie zawarta główna logika aplikacji.
- Angular zostanie wykorzystany jako frontend do przedstawienia danych w przyjazny sposób.
- PostgresSQL zostanie wykorzystany jako kontener do przechowywania danych aplikacji.

Opis motywacji dla wybranych technologii w projekcie

Spotify 2.0 jest budowany w oparciu o stosunkowo nowe oraz popularne technologie. Zwykle głównym powodem wyboru technologii była dobra dokumentacja. Uważamy, że jest to główna zaleta, gdy trzeba wybierać między różnymi technologiami.

Spring Framework – kompleksowy framework, które znacznie skraca proces tworzenia aplikacji. Jego głównym założeniem jest dostarczenie programiście mechanizmów, które sprawią, że tworzenie aplikacji będzie sprawniejsze. Jest to obecnie najlepiej rozwinięty oraz najpopularniejszy framework dla Java. Spring dostarcza gotowe rozwiązania dla takich aspektów tworzenia systemu jak: tworzenie WebAPI, autoryzacja oraz autentykacja użytkowników, realizacja dostępu do bazy danych, tworzenie testów jednostkowych oraz wiele innych.

Angular - frontendowy framework, służący do budowania dynamicznych stron internetowych. Sam w sobie dostarcza wszystko, co jest niezbędne do stworzenia aplikacji internetowej. W porównaniu do innych frameworków, gdzie często musimy skorzystać z dodatkowych rozwiązań.

MongoDB – nierelacyjny system zarządzania bazą danych. Został użyty w projekcie ze względu na zaimplementowany w nim system składowania plików GridFS. Jednym z kluczowych wyzwań podczas projektowania systemu było znalezienie sposoby na przechowywanie dużej ilości plików audio przy zachowaniu wysokiej wydajności oraz skalowalności bazy danych.

PostgreSQL - jeden z najpopularniejszych otwartych systemów zarządzania relacyjnymi bazami danych. Rozwiązanie zostało wybrane jako główny magazyn do przechowywania danych, ponieważ jest dobrze znane przez osoby pracujące przy projekcie.

Heroku - platforma chmurowa (PaaS) obsługująca wiele języków programowania oraz technologii. Zdecydowaliśmy się na to, ponieważ chcieliśmy wypróbować tę bardzo popularną platformę.

Docker - platforma służąca do tworzenia i wdrażania aplikacji kontenerowych oraz zarządzania nimi. Rozwiązanie zostało użyte jako wsparcie dla wdrażania systemu na platformę Heroku.

Opis przedziału czasowego do realizacji projektu

Data rozpoczęcia projektu – 04.10.2021

Okres: 04.10.2021 - 24.10.2021 (prace przygotowawcze)

Podczas owego okresu została ustalona koncepcja projektu, osoby w zespole otrzymały określone role (lider projektu, inżynier, tester). W celu realizacji projektu została założona Jira, w związku z metodologią SCRUM grupa określiła pracę w cotygodniowych sprintach oraz zdefiniowała milestony, epiki oraz taski które będą wykonywane w trakcie trwania projektu.

Pierwsza część projektu została podzielona na 3 milestony:

Milestone 1 - 25.10.2021 - 05.12.2021

Głównym celem milestona jest sporządzenie głównego opisu projektu oraz wykonanie schematów projektu. W tym celu praca została podzielona na dwa wynikające epiki

Milestone 2 – 06.10.2021 – 02.01.2022

Realizacja owego milestona polega na zaprojektowaniu intefejsu działania aplikacji oraz zdefiniowanie mockupów/widoków projeku, następstwem również było podzielnie prac na dwa epiki.

Milestone 3 – 03.01.2022 - 23.01.2022

Ostatni milestone jest poświęcony na przygotowanie narzędzia CI/CD oraz również przygotowanie do części implementacyjnej. Milestone został podzielny na dwa epiki.

Opis o użytkownikach (aktorach)

Aktorami systemu będą ludzie chcący skorzystać z możliwości wysłuchania utworów muzycznych stworzonych przez wszelakich twórców znajdujących się na platformie. Aplikacja jest również skierowana dla takich użytkowników, którzy pragną rozpowszechnienia własnych utworów, aby stać się w przyszłości rozpoznawalnymi artystami muzycznymi.

Diagram przypadków użycia

Diagram przypadków użycia reprezentuje specyficzne spojrzenie na system z pozycji aktorów, którzy będą z niego korzystać, symbolizuje interakcję pomiędzy systemem a użytkownikiem.

W przedstawionym poniżej diagramie przypadków użycia przedstawione zostały funkcjonalności systemu udostępnione użytkownikowi takie jak: rejestracja, logowanie, zmiana hasła, stworzenie kolekcji, dodanie/usunięcie utworu dźwiękowego, możliwość przeglądania artystów/utworów, słuchania utworów, interakcji poprzez obserwowanie/polubienie artystów/utworów, przeglądanie obserwowanych/polubionych artystów/utworów oraz odpowiednie cofnięcie operacji obserwowania/polubienia.

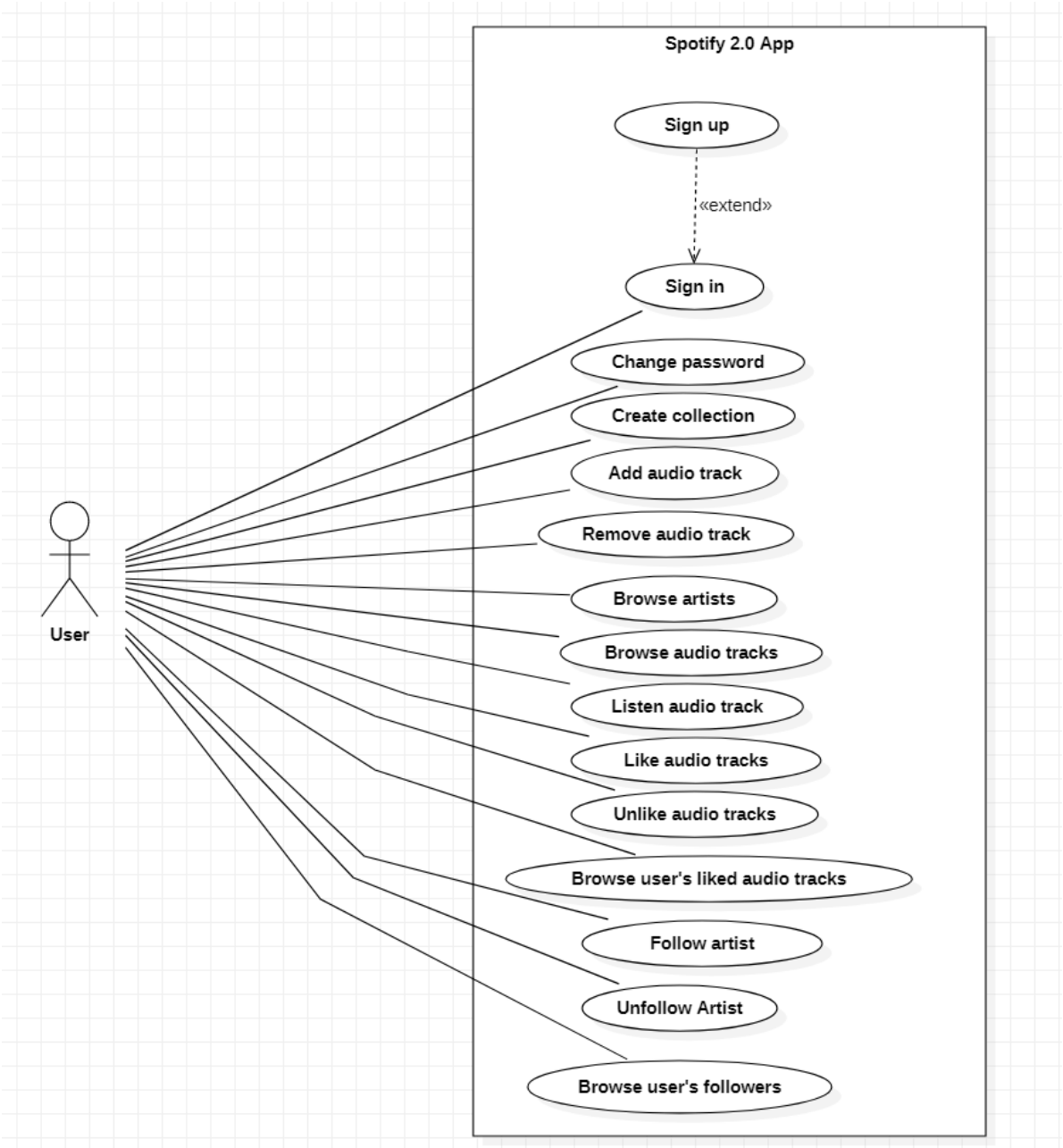


Diagram przypadków użycia został wygenerowany przy użyciu narzędzia StarUML Version 4.1.6

Scenariusze testowe

Scenariusz rejestracji:

Przypadek użycia	Rejestracja nowego użytkownika w aplikacji.
Scenariusz	Prawidłowa rejestracja nowego użytkownika, dodanie do bazy nowego użytkownika.
Użytkownik	słuchacz / artysta
Warunki wstępne	Nowy użytkownik chcący posiadać nowe konto, który nie jest zarejestrowany w aplikacji.
NIezmienniki	Użytkownik żąda rejestracji w aplikacji.
Opis	Użytkownik który nie posiada konta w systemie może się zarejestrować podając swój login i hasło które musi utworzyć podczas rejestracji. Po prawidłowej rejestracji użytkownik może się zalogować do aplikacji.
Warunki końcowe	Użytkownik został dodany do bazy danych / założył konto
Źródła ew. błędów	Dane do rejestracji nie spełniają wymagań aplikacji.

Scenariusz logowania:

Przypadek użycia	Użytkownik loguje się do aplikacji.
Scenariusz	Prawidłowe zalogowanie się użytkownika do aplikacji oraz uzyskanie dostępu.
Użytkownik	słuchacz / artysta
Warunki wstępne	Użytkownik ma dostęp do aplikacji. Użytkownik jest już zarejestrowany.
NIezmienniki	Użytkownik żąda zalogowania się w aplikacji.
Opis	Użytkownik wpisuje login oraz hasło. Aplikacja sprawdza czy podany login istnieje oraz czy hasło jest poprawne. Zalogowany użytkownik otrzymuje uprawnienia do odpowiednio przyznanych mu operacji oraz do wylogowania się.
Warunki końcowe	Użytkownik uzyskuje dostęp do aplikacji. Może słuchać muzyki, tworzyć playlisty, wrzucać utwory, obserwować i polubić inne utwory, playlisty, artystów.
Źródła ew. błędów	Podanie błędnego loginu lub hasła.

Scenariusz zmiany hasła:

Przypadek użycia	Zmiana hasła przez użytkownika.
Scenariusz	Użytkownik zmienia hasło w aplikacji.
Użytkownik	słuchacz / artysta
Warunki wstępne	Użytkownik jest zalogowany w aplikacji.
NIezmienniki	Użytkownik żąda zmiany hasła.
Opis	Użytkownik wpisuje stare hasło i dwa razy potwierdza nowe hasło. Aplikacja zmienia hasło użytkownika.
Warunki końcowe	Hasło użytkownika zostaje zmienione.
Źródła ew. błędów	Dane do zmiany hasła nie spełniają wymagań aplikacji. Wpisane stare hasło jest niepoprawne.

Scenariusz Stworzenia kolekcji:

Przypadek użycia	Stworzenie kolekcji przez użytkownika.
Scenariusz	Użytkownik tworzy nową kolekcję w aplikacji.
Użytkownik	słuchacz / artysta
Warunki wstępne	Użytkownik jest zalogowany w aplikacji.
NIezmienniki	Użytkownik żąda utworzenia kolekcji.
Opis	Użytkownik wybiera opcję stworzenia kolekcji, wpisuje nazwę tworzonej kolekcji. Aplikacja stwarza nową kolekcję użytkownika
Warunki końcowe	Kolekcja została utworzona.
Źródła ew. błędów	Kolekcja o takiej samej nazwie już istnieje.

Scenariusz Stworzenia kolekcji:

Przypadek użycia	Stworzenie kolekcji przez użytkownika.
Scenariusz	Użytkownik tworzy nową kolekcję w aplikacji.
Użytkownik	słuchacz / artysta
Warunki wstępne	Użytkownik jest zalogowany w aplikacji.
NIezmienniki	Użytkownik żąda utworzenia kolekcji.
Opis	Użytkownik wybiera opcję stworzenia kolekcji, wpisuje nazwę tworzonej kolekcji. Aplikacja stwarza nową kolekcję użytkownika
Warunki końcowe	Kolekcja została utworzona.
Źródła ew. błędów	Kolekcja o takiej samej nazwie już istnieje.

Scenariusz dodania utworu:

Przypadek użycia	Dodanie utworu przez użytkownika.
Scenariusz	Użytkownik dodaję nowy utwór w aplikacji.
Użytkownik	słuchacz / artysta
Warunki wstępne	Użytkownik jest zalogowany w aplikacji.
NIezmienniki	Użytkownik żąda dodania utworu w aplikacji.
Opis	Użytkownik wybiera opcję dodania utworu, uploaduje plik z utworem. Aplikacja zapisuje utwór w bazie danych.
Warunki końcowe	Utwór został dodany.
Źródła ew. błędów	NIeprawidłowy format pliku. Plik ma zbyt duży rozmiar.

Scenariusz usunięcia utworu:

Przypadek użycia	Usunięcie utworu przez użytkownika.
Scenariusz	Użytkownik usuwa istniejący utwór w aplikacji.
Użytkownik	słuchacz / artysta
Warunki wstępne	Użytkownik jest zalogowany w aplikacji. Utwór istnieje w bazie danych.
NIezmienniki	Użytkownik żąda usunięcia utworu w aplikacji.
Opis	Użytkownik na liście swoich wrzuconych utworów wybiera opcję usunięcia. Aplikacja usuwa utwór z bazy danych.
Warunki końcowe	Utwór został usunięty.
Źródła ew. błędów	Usuwany utwór nie istnieje w bazie danych.

Scenariusz przeglądania artystów:

Przypadek użycia	Przeglądanie artystów przez użytkownika.
Scenariusz	Użytkownik przegląda artystów w aplikacji.
Użytkownik	słuchacz / artysta
Warunki wstępne	Użytkownik jest zalogowany w aplikacji. Artyści istnieją w bazie danych.
NIezmienniki	Użytkownik żąda przeglądania artystów
Opis	Użytkownik na odpowiednio zaprojektowanym GUI może przeglądać i wyszukiwać istniejących w aplikacji artystów.
Warunki końcowe	Artyści/artysta zostali/ał znalezieni/ona.
Źródła ew. błędów	Szukany artysta nie istnieje w bazie danych.

Scenariusz przeglądania utworów:

Przypadek użycia	Przeglądanie utworów przez użytkownika.
Scenariusz	Użytkownik przegląda utwory w aplikacji.
Użytkownik	słuchacz / artysta
Warunki wstępne	Użytkownik jest zalogowany w aplikacji. utwory istnieją w bazie danych.
NIezmienniki	Użytkownik żąda przeglądania utworów.
Opis	Użytkownik na odpowiednio zaprojektowanym GUI może przeglądać i wyszukiwać istniejące w aplikacji utworu.
Warunki końcowe	Utwór/utwory zostały/ał znaleziony/ona
Źródła ew. błędów	Szukany utwór nie istnieje w bazie danych.

Scenariusz polubienia utworu:

Przypadek użycia	Polubienie utworu przez użytkownika.
Scenariusz	Użytkownik postanawia polubić dany utwór.
Użytkownik	słuchacz / artysta
Warunki wstępne	Użytkownik jest zalogowany w aplikacji. utwór istnieje w bazie danych.
NIezmienniki	Użytkownik żąda polubienia utworu.
Opis	Użytkownik może polubić utwór na liście z utworami lub aktualnie odtwarzany utwór. Fakt polubienia utworu jest zapisana w bazie danych.
Warunki końcowe	Utwór został polubiony
Źródła ew. błędów	Utwór został już polubiony.

Scenariusz usunięcia polubienia utworu:

Przypadek użycia	Usunięcia polubienia utworu przez użytkownika.
Scenariusz	Użytkownik postanawia usunąć polubienie danego utworu.
Użytkownik	słuchacz / artysta
Warunki wstępne	Użytkownik jest zalogowany w aplikacji. utwór istnieje w bazie danych i został wcześniej polubiony
NIezmienniki	Użytkownik żąda usunięcia polubienia utworu.
Opis	Użytkownik może usunąć polubienie utworu na liście z utworami lub na aktualnie odtwarzanym utworze. Fakt usunięcia polubienia utworu jest odzwierciedlony w bazie danych.
Warunki końcowe	Utwór przestał być lubiany.
Źródła ew. błędów	Utwór nie był wcześniej polubiony.

Scenariusz przeglądania ulubionych utworów przez użytkownika:

Przypadek użycia	Użytkownik przegląda polubione przez siebie utwory.
Scenariusz	Użytkownik Przegląda swoje ulubione utwory.
Użytkownik	słuchacz / artysta
Warunki wstępne	Użytkownik jest zalogowany w aplikacji. utwory istnieją w bazie danych i zostały wcześniej polubione
NIezmienniki	Użytkownik żąda przeglądania swoich ulubionych utworów.
Opis	Użytkownik na odpowiednio zaprojektowanym GUI może przeglądać i wyszukiwać swoje ulubione utwory.
Warunki końcowe	Ulubiony utwór/utwory zostały/ał znaleziony/ona
Źródła ew. błędów	Utwór nie był wcześniej polubiony.

Scenariusz zaobserwowania artysty:

Przypadek użycia	Użytkownik zaobserwował artyste.
Scenariusz	Użytkownik postanawia zaobserwować artystę.
Użytkownik	słuchacz / artysta
Warunki wstępne	Użytkownik jest zalogowany w aplikacji. Artysta istnieje w bazie danych.
NIezmienniki	Użytkownik żąda zaobserwowania artysty.
Opis	Użytkownik na profilu artysty może go zaobserwować. Fakt zaobserwowania artysty został zapisany w bazie danych.
Warunki końcowe	Artysta został zaobserwowany.
Źródła ew. błędów	Artysta został już wcześniej zaobserwowany

Scenariusz usunięcia obserwacji artysty:

Przypadek użycia	Użytkownik usunął obserwacje artysty.
Scenariusz	Użytkownik postanawia usunąć obserwacje artysty.
Użytkownik	słuchacz / artysta
Warunki wstępne	Użytkownik jest zalogowany w aplikacji. Artysta istnieje w bazie danych i został wcześniej zaobserwowany.
NIezmienniki	Użytkownik żąda usunięcia obserwacji artysty.
Opis	Użytkownik na profilu artysty może usunąć swoją obserwację. Fakt usunięcia obserwacji jest odzwierciedlony w bazie danych.
Warunki końcowe	Artysta został zaobserwowany.
Źródła ew. błędów	Artysta został już wcześniej zaobserwowany

Scenariusz przeglądania obserwatorów użytkownika:

Przypadek użycia	Użytkownik przegląda swoich obserwatorów.
Scenariusz	Użytkownik postanawia przeglądać swoich obserwatorów.
Użytkownik	słuchacz / artysta
Warunki wstępne	Użytkownik jest zalogowany w aplikacji. Inni użytkownicy aplikacji muszą wcześniej zaobserwować naszego użytkownika.
NIezmienniki	Użytkownik żąda przeglądania swoich obserwatorów.
Opis	Użytkownik na odpowiednio zaprojektowanym GUI może przeglądać swoich obserwatorów.
Warunki końcowe	Użytkownik przeglądnął swoich obserwatorów.
Źródła ew. błędów	Użytkownik nie ma żadnych obserwatorów.

Diagram klas

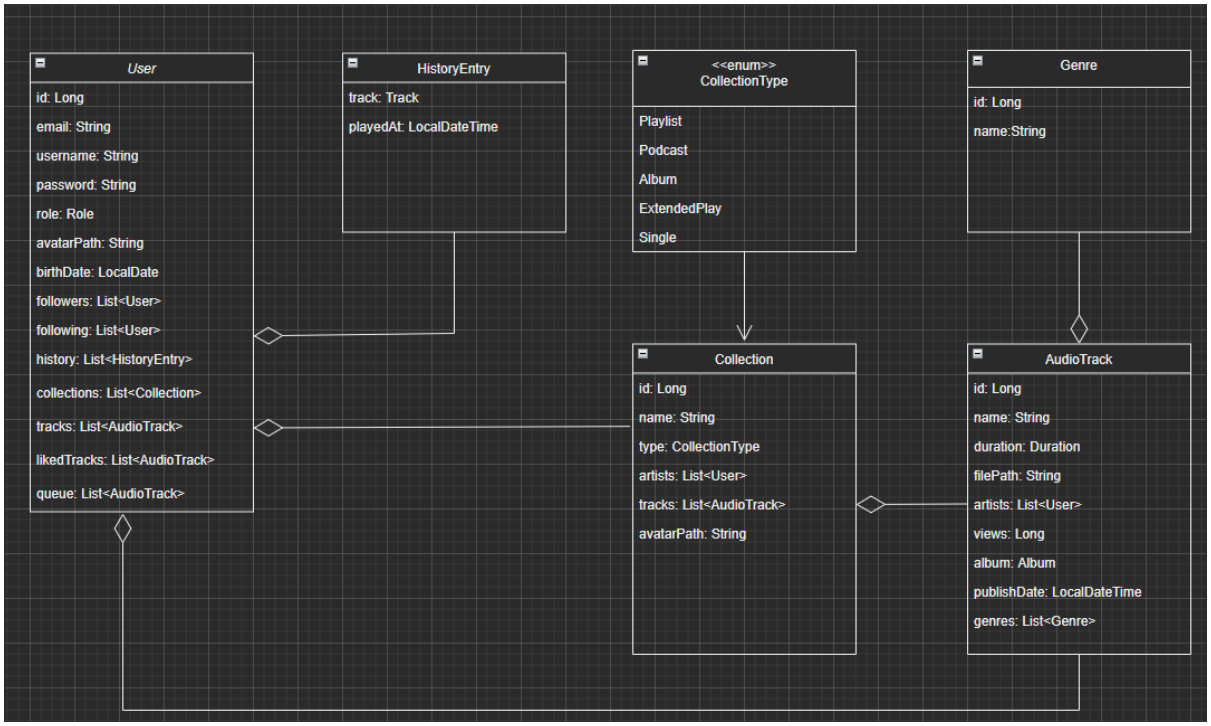


Diagram wykonany przy użyciu draw.io

Opis diagramu klas i relacji

Na powyższym diagramie zawierają się główne klasy reprezentujące obiekty w naszej aplikacji

- **User** – jest obiektem użytkownika, oprócz typowo podstawowych pól zawiera on listę użytkowników których obserwuje/przez których jest obserwowany, historię odtwarzania, listę polubionych/dodanych ścieżek audio, listę utworzonych kolekcji (playlist) oraz listę ścieżek audio dodanych do kolejki
- **AudioTrack** – klasa reprezentująca ścieżkę audio, zawiera ona nazwę oraz długość ścieżki, listę twórców, ilość odsłuchań, datę dodania, listę gatunków do których się zalicza oraz album do którego należy.
- **Genre –** obiekt reprezentujący gatunek, zawiera unikalne id i nazwę
- **HistoryEntry** – składowa historii użytkownika, zawiera ścieżkę audio i datę w której została odsłuchana.
- **CollectionType –** klasa typu ENUM, zawiera typy kolekcji (playlist)
- **Collection –** jest to klasa reprezentująca playlisty, podcasty itp., zawiera pola takie jak: nazwa, typ, lista twórców, lista ścieżek audio składających się na ta kolekcję, ścieżkę do avataru.

Diagram stanów

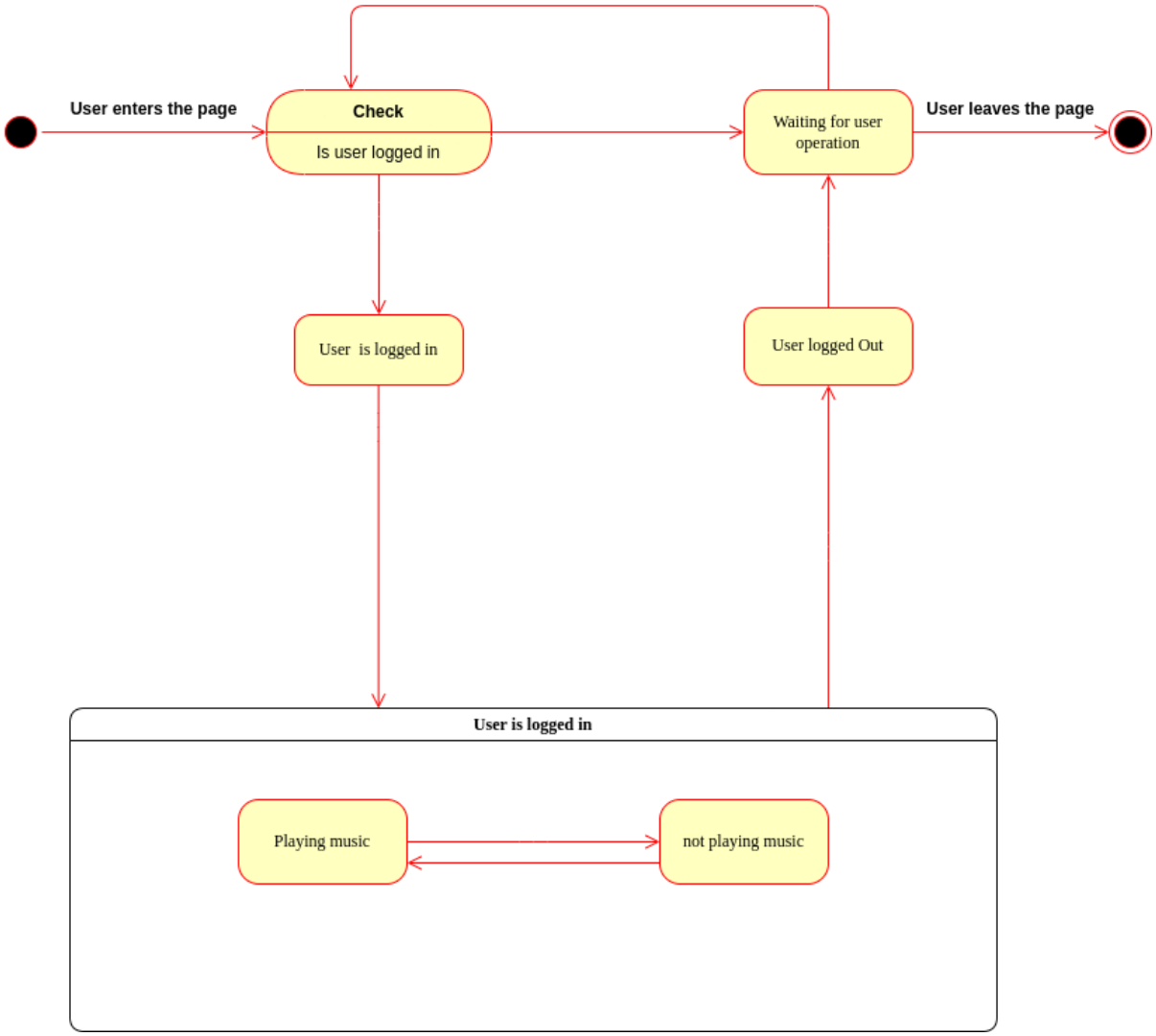
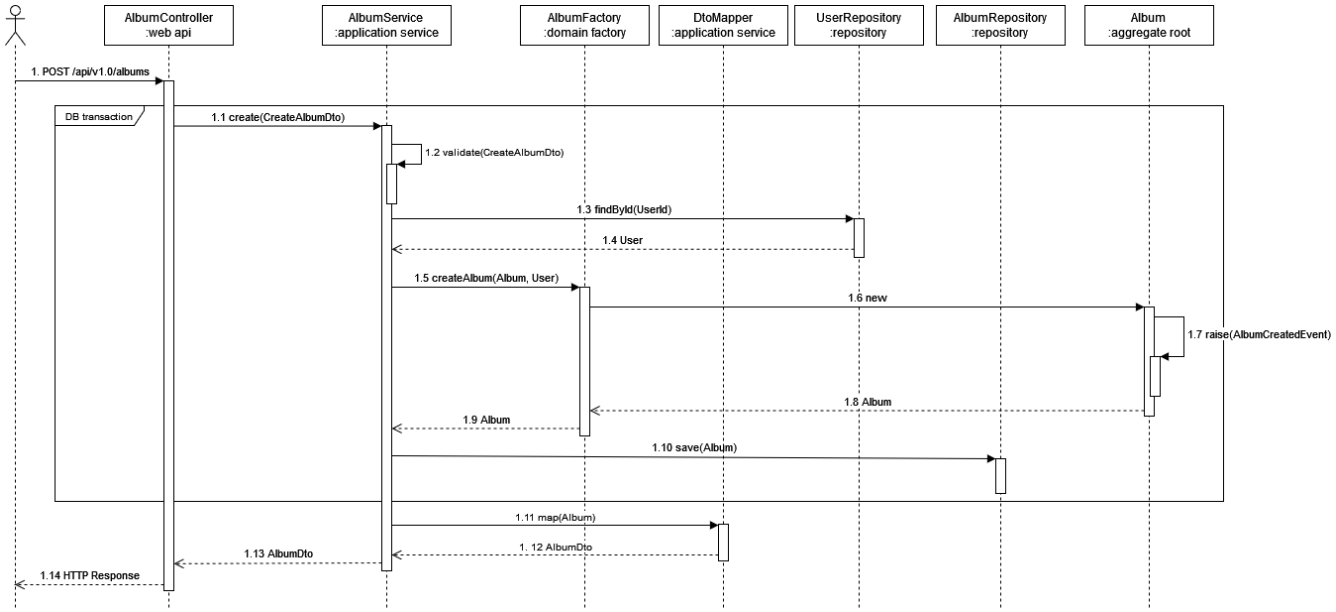
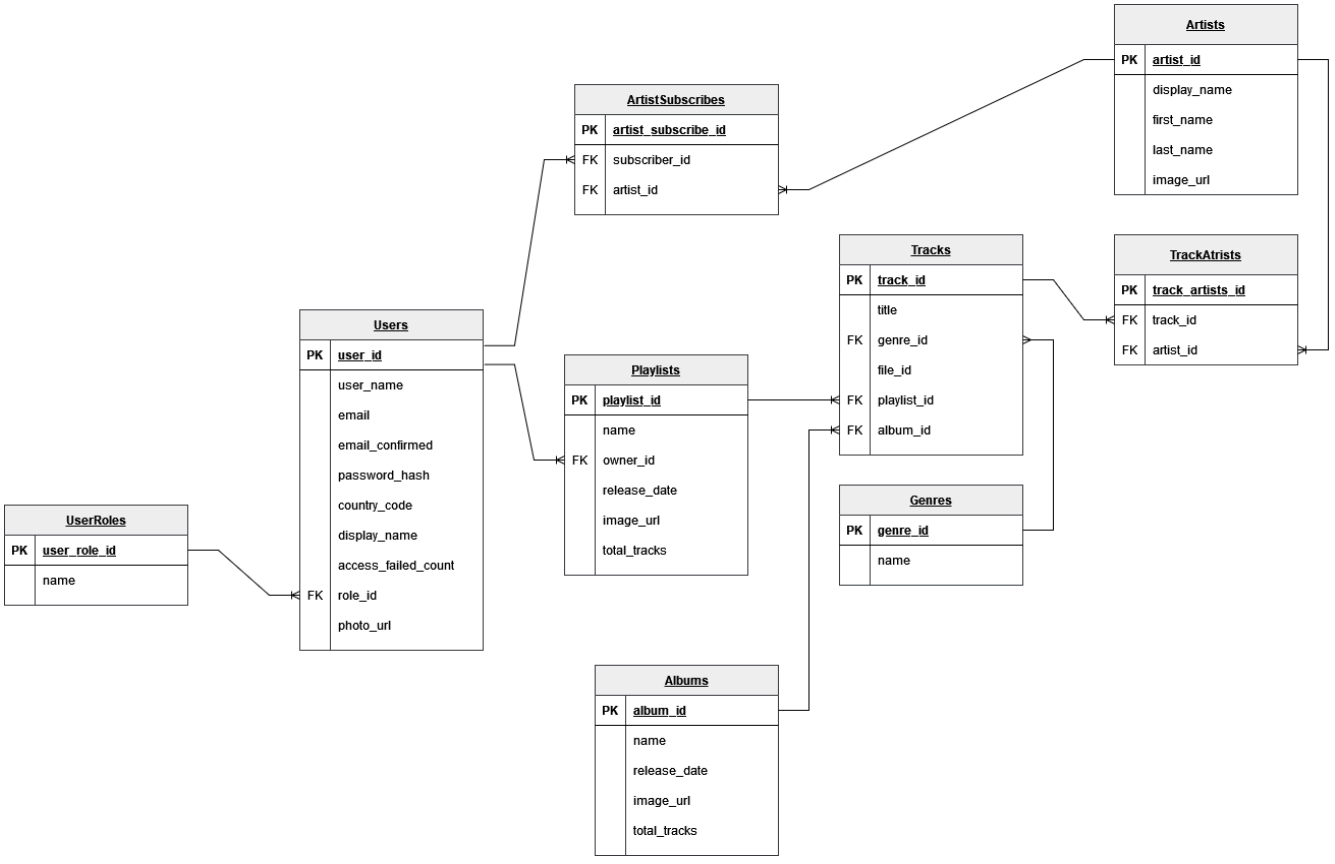


Diagram przedstawia dwa możliwe stany w którym może znaleźć się aplikacja. Stan kiedy nie jesteśmy zalogowani do aplikacji oraz stan kiedy jesteśmy zalogowani. Dodatkowo możemy wyszczególnić 2 stany odtwarzacza muzyki kiedy jesteśmy zalogowani do aplikacji, a mianowicie stan w którym odtwarzana jest utwór i stan w którym nie jest odtwarzany utwór.

Diagram sekwencji



Schemat bazy danych



Wymogi funkcjonalne

Wymóg	**opis**	**Kryteria akceptacji**
Rejestracja	Użytkownik ma możliwość rejestracji w serwisie	Rekord użytkownika jest w bazie danych i może się już zalogować
Logowanie	Użytkownik jeśli jest zarejestrowany może się zalogować do aplikacji	Serwer zwrócił token którym to klient może uwierzytelniać użytkownika.
Zmiana hasła	Użytkownik w każdej chwili ma możliwość zmiany hasła	Rekord użytkownika w bazie danych został zaktualizowany o nowe hasło
Stworzenie kolekcji	Użytkownik może utworzyć swoją kolekcje	Nowa kolekcja w bazie danych powiązana z użytkownikiem
Dodanie utworu	Użytkownik może uploadowac utwór dźwiękowy	Utwór dźwiękowy zapisany w bazie danych i dostępny dla wszystkich zalogowanych użytkowników
Usunięcie utworu	Użytkownik może usunąć wcześniej uploadowany utwór	Utwór dźwiękowy usunięty z bazy danych i niedostępny dla użytkowników
Przeglądanie artystów	Użytkownik może przeglądać i wyszukiwać artystów	Lista artystów dostępna w GUI użytkownika po zalogowaniu
Przeglądanie utworów	Użytkownik może przeglądać i wyszukiwać utwory	Lista utworów dostępna w GUI użytkownika po zalogowaniu
Słuchanie utworów	Użytkownik może odtwarzać utwór	Aplikacja umożliwia strumieniowanie utworów z bazy danych
Polubienie utworów	Użytkownik może polubić utwór	W bazie danych zapisany fakt że użytkownik polubił utwór
Cofnięcie polubienia utworu	Użytkownik może cofnąć polubienie utworu	W bazie danych zapisany fakt że użytkownik cofnął polubienie utworu

Przeglądanie ulubionych utworów użytkownika	Użytkownik może przeglądać utwory które wcześniej polubił	W GUI użytkownika jest lista gdzie użytkownik ma ulubione utwory
Zaobserwowanie artysty	Użytkownik może Zaobserwować artystę	W bazie danych zapisany fakt że użytkownik zaobserwował artystę.
Cofnięcie obserwacji artysty	Użytkownik może cofnąć obserwacje artysty	W bazie danych zapisany fakt że użytkownik cofnął obserwacje artysty.
Przeglądanie obserwowanych arytsów	Użytkownik może przeglądać artystów których obserwuje	W GUI użytkownika jest lista gdzie użytkownik ma swoich zaobserwowanych artystów

Wymagania niefunkcjonalne

Wydajność - Aplikacja będzie umożliwiać korzystanie z niej przez przy min. 10 osobach równocześnie. Struktura aplikacji wraz z danymi 100 rekordów wraz z załącznikami nie będzie większa niż 5 GB

Dostępność / niezawodność - Aplikacja będzie dostępna dla wszystkich użytkowników 24h i 7 dni w tygodniu, min. Do końca lipca 2022.

Wsparcie - Wszyscy użytkownicy mogą zgłaszać błędy na dedykowany adres e-mail lub jako issue na platformie github. Wszystkie błędy krytyczne aplikacji zostaną skutecznie naprawione w ciągu 7 dni roboczych.

Bezpieczeństwo - Hasła użytkowników w bazie danych są hashowane. Endpointy zwracające wrażliwe dane użytkowników muszą być zabezpieczone przed niepowołanym dostępem.

Wdrożenie - Wdrażana na produkcję jest tylko stabilna wersja aplikacji. Jeśli zajdzie taka potrzeba przy wdrażaniu należy dostarczyć projekt migracji bazy danych. Migracje bazy danych są przeprowadzane tylko wtedy kiedy mamy backup aktualnej wersji.

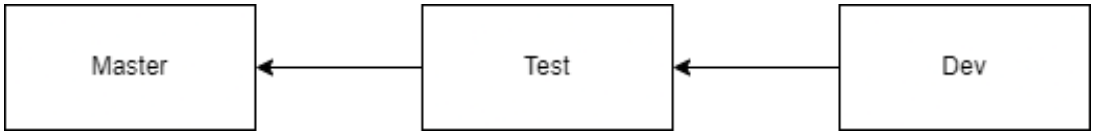
Użyteczność - Aplikacja jest intuicyjna i łatwa w użytkowaniu. Strumieniowanie muzyki działa również w tle poza focusem na przeglądarkę. Instrukcje dla użytkowników aplikacji zostanie stworzona w formie pdf dostępnego do pobrania.

API Interface

Method	Path	Definition
POST	auth/signup	registers user
POST	auth/signin	sign in user
---	---	---
GET	users	gets list of users
PUT	users/:id	updates user info
PUT	users/:id/password	updates user password
GET	users/:id	gets specific user
PUT	users/:id/follow	follows/unfollows user
GET	users/:id/followers	gets list of user followers
GET	users/:id/following	gets list of users followed
GET	users/:id/collections	gets list of user collections
GET	users/:id/liked	gets list of liked items
---	---	---
GET	users/:id/history	gets tracks history of user
POST	users/:id/history	adds to user history
---	---	---
GET	users/:id/queue	gets user queue
POST	users/:id/queue	adds to user queue
DELETE	users/:id/queue	removes from user queue
---	---	---
GET	collections/:id	gets specific collection
PUT	collections/:id	updates collection info
POST	collections/:id	creates collection
POST	collections/:id/tracks	adds track to collection
DELETE	collections/:id	removes collection
DELETE	collections/:collection_id/tracks/:track_id	removes track from collection
---	---	---
GET	tracks/:id	returns track info
GET	tracks/:id/stream	returns audio stream
PUT	tracks/:id	updates track info
POST	tracks/:id	adds tracks
DELETE	tracks/:id	removes track

Opis środowisk projektowych

W celu określenia pracy nad projektem zostały zdefiniowane trzy środowiska projektowe:



Master - środowisko produkcyjne które zostanie udostępnione klientowi.

Test - środowisko testerskie zdefiniowane w celu przeprowadzenia testów przez testerów. Na utworzonym środowisku znajduje się aplikacja przeznaczona do przeprowadzania testów biznesowych informujących czy system spełnia wszystkie założenia oraz testy które zostały uprzednio zaakceptowane na środowisku Dev.

Dev - środowisko deweloperskie przeznaczone do wykonywania testów inżynierskich, testowania architektury jak również zmian które mogę zagrozić stabilności systemu. W tym środowisku występują testy poszczególnych funkcjonalności systemu oraz konfiguracji.

Opis systemu wersjonowania

Naszym oprogramowaniem do kontroli wersji będzie GIT, natomiast nasz system wersjonowania będziemy opierać na wersjonowaniu semantycznym 2.0.0 (https://semver.org/lang/pl/). Czyli nasza wersja będzie się składać z trzech części oddzielonych kropką: **Major**.**Minor**.**Patch**.

Wersję Major będziemy zmieniać tylko wtedy kiedy zmiany łamią kompatybilność wsteczną z poprzednimi wersjami. Taką zmianą może być np. przebudowa naszych endpointów. Przykładem może być: 2.9.0 → 3.0.

Wersję Minor będziemy zmieniać tylko wtedy kiedy dodajemy nową funkcjonalność ale też nie łamiemy kompatybilności wstecznej z poprzednimi wersjami. Taką zmianą może być np. dodanie nowego endpointu lub dodatkowego pola w zwróconej odpowiedzi. Przykładem może być 2.8.2 → 2.9.0

Wersję Patch będziemy zmieniać tylko wtedy kiedy będziemy poprawiać błędy, ale naprawienie takich błędów nie spowoduje złamania kompatybilności z wcześniejszymi wersjami. Przykładem może być 2.8.1 → 2.8.2

Opis i zdefiniowanie narzędzia CI/CD

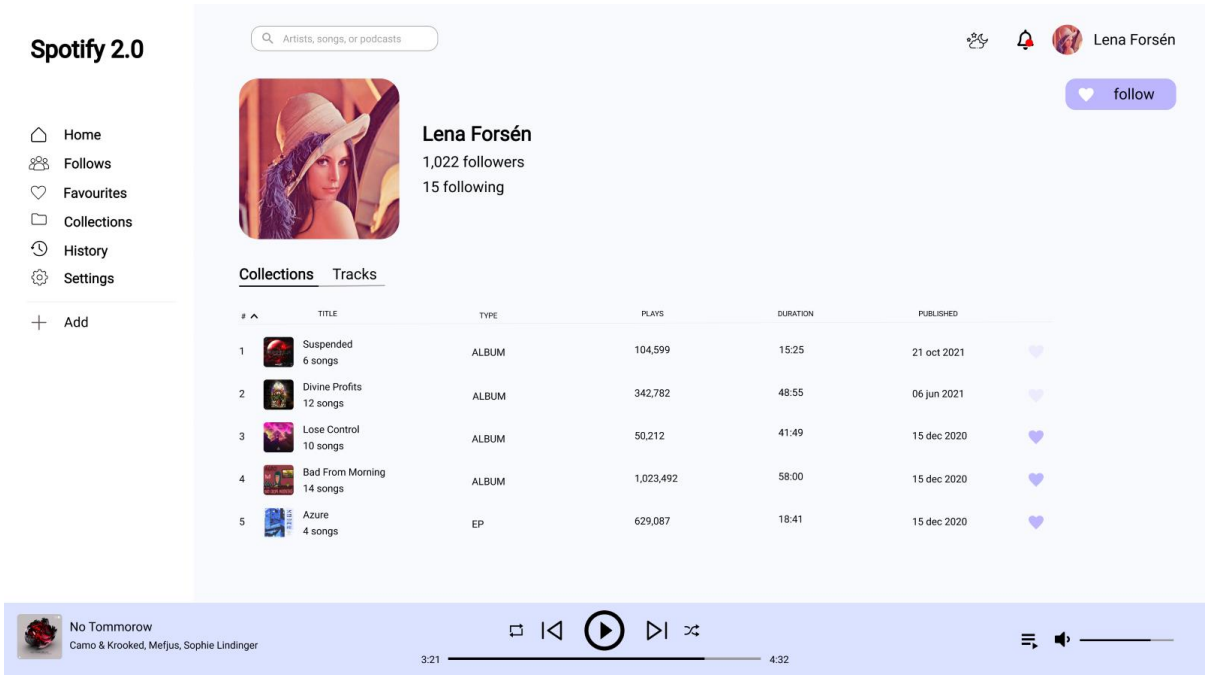
CI - czyli continuous integration to to zbiór zasad, wytycznych, narzędzi służących do zautomatyzowanego sposobu budowania i testowania aplikacji.

CD - czyli continuous delivery to również zbiór zasad, wytycznych, narzędzi które automatyzują proces wdrażania aplikacji i wprowadzanych zmian w kodzie do przygotowanej infrastruktury serwerowej.

Narzędziem który będziemy używać aby zapewnić sprawne CI/CD będzie jenkins. Jenkins będzie automatycznie budował oraz uruchamiał testy na 3 gałęziach w naszym repozytorium odpowiadającym wcześniej zdefiniowanych środowiskach czyli na master, test i dev. Jeśli testy przejdą to odpowiedni build zostanie automatycznie wdrożony na odpowiednie środowisko.

Dodatkowo możemy zintegrować Jenkisna z githubem tak aby blokował pull request jeśli build lub testy się nie powiodą.

Mockup proflu użytkownika



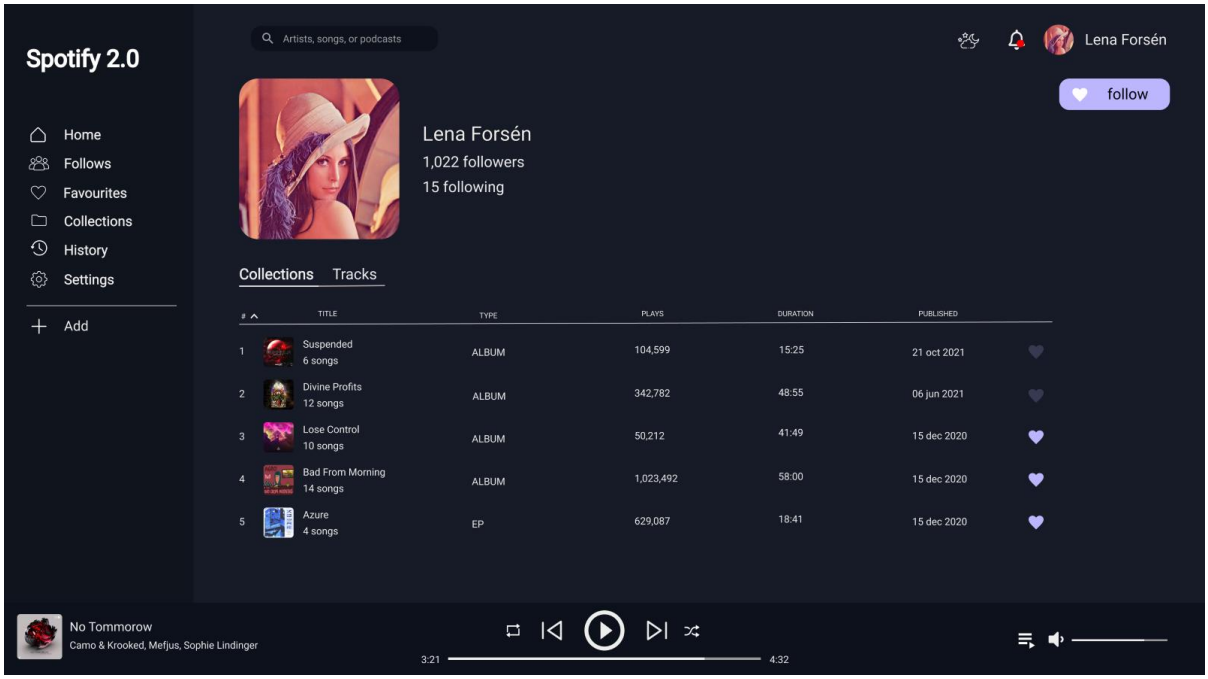


Diagram wykonany przy użyciu figma.com

Na powyższych zdjęciach możemy zobaczyć jak powinien wyglądać profil użytkownika, po lewej stronie znajduje się główne menu z którego możemy przejść do:

- strony głównej,
- widoku osób które nas obserwują/obserwujemy,
- widoku polubionych kolekcji / ścieżek audio
- widoku historii odtwarzanych ścieżek audio
- widoku opcji użytkownika
- widoku dodawania nowych ścieżek audio

Na dole strony możemy znaleźć pasek który zawiera obecnie odtwarzaną ścieżkę audio oraz kontrolki służące do jej zarządzania.

Na głównym ekranie możemy zobaczyć awatar osoby, liczbę osób które ją obserwują/obserwuje, zawarta jest również tabela z listą utworzonych kolekcji / dodanych ścieżek audio.

W prawym górnym rogu zawarty jest awatar oraz nazwa obecnie zalogowanego użytkownika, powiadomienia, jak i kontrolka do przełączania się między ciemnym i jasnym trybem.