Projekt: Perfect Placements

Tomasz Maczek

Zaawansowane Technologie Internetowe
WFiIS

21 Sierpnia 2023

Spis treści

1. Opis tematu projektu

Tematem projektu jest stworzenie serwisu internetowego, w którym można tworzyć rankingi swoich ulubionych albumów. Funkcjonalność, którą chcemy zapewnić to:

- możliwość rejestracji nowych użytkowników,
- logowanie na stronę,
- tworzenie i usuwanie rankingów,
- dodawanie, usuwanie i zmienianie kolejności pozycji w rankingach,
- wyszukiwanie rankingów korzystając z API Spotify i dodawanie wyszukanych pozycji.

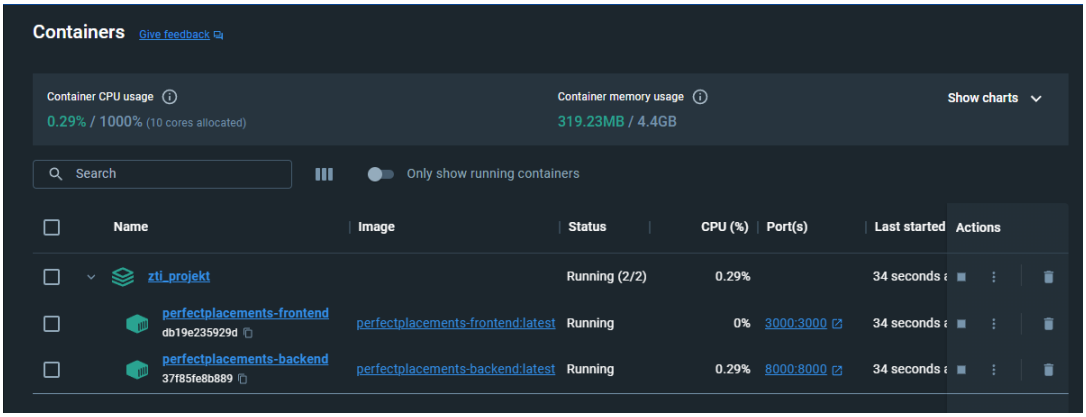
2. Użyte technologie

Technoogie i narzędzia użyte w projekcie:

- język Java i Spring Boot,
- Spring Security oraz Spring Data,
- baza danych PostgreSQL na serwisie chmurowym EleplantSQL,
- biblioteka Spotify Web API,
- JavaScript i React,
- Docker,
- IDE: IntelliJ oraz Visual Studio Code,
- Postman

3. Wdrożenie projektu

Wdrożenie projektu odbywa się poprzez użycie technologii Docker. W głównym folderze projektu znajduje się plik `docker-compose.yml`, który uruchamiamy przez komendę `docker compose up` (upewniając się że *docker daemon* jest uruchomiony - najłtawiej uruchamiając `Docker Desktop`).
Po uruchomieniu serwis dostępny jest pod adresem `http://localhost:3000`.



Rysunek 3.1: Screen z programu Docker Desktop prezentujący uruchomiony kontener compose.

4. Opis kodu programu

Program podzielony jest na dwie części - backendową napisaną w Javie i Springu oraz frontend w JavaScript i React.

4.1 Backend

Pliki `.java` są rozmieszczone w odpowiednich katalogwach według ich funkcjonalności.

- **configuration** - pliki konfiguracyjne
  - `SecurityConfiguration` - klasa zawierająca konfigurację bezpieczeństwa - klucze RSA, enkoder hasła, enkoder i dekoder tokenu JWT, filter żądań http.
- **controllers** - kontrolery odpowiadające za RESTowe endpointy
  - `AuthenticationController` - odpowiada za logowanie i rejestrację użytkownika,
  - `RankingController` - odpowiada za wszystkie metody dotyczące rankingów i ich funkcjonalności.
- **dtos** - Data Transfer Objects - służą do przesyłania najczęściej uproszczonych/ograniczonych danych między serwerem i klientem
  - `EntryDTO` - do tworzenia pozycji w rankingu,
  - `EntryDataDTO` - zwraca wszystkie dane pozycji takie jak nazwa artysty i albumu, pozycję w rankingu oraz URL grafiki okładki,
  - `LoginResponseDTO` - zwracany przez serwer po zalogowaniu użytkownika,

  - `RankingDTO` - wysyłany przez frontend przy tworzeniu rankingu,
  - `RegistrationDTO` - używany przy rejestracji oraz logowaniu - przesyła login i hasło do serwera.

- **model** - klasy reprezentujące obiekty w bazie danych
  - `ApplicationUser` - model użytkownika aplikacji, implementuje `UserDetails` aby działać zgodnie z SpringSecurity,
  - `Entry` - pozycja w rankingu,
  - `Ranking` - obiekt rankingu,
  - `Role` - role użytkowników - admin i user (domyślnie każdy nowy użytkownik ma rolę user).

- **repository** - repozytoria zarządzająca encjami Spring Data
  - `EntryRepository`,
  - `RankingRepository`,
  - `RoleRepository`,
  - `UserRepository`.

- **services** - serwisy zarządzające logiką aplikacji
  - `AuthenticationService`- odpowiedzialny za metody dot. logowania i rejestracji i zapisywanie tych użytkowników do bazy danych,
  - `RankingService` - metody dot. funkcjonalności rankingów,
  - `SpotifyService` - korzystający z biblioteki **Spotify Web API Java** odpowiada za metody łączące się z tym API,
  - `TokenService` - zarządzanie tokenami JWT,
  - `UserService` - metody dot. pobierania użytkowników z bazy.

- **utils** - pomocnicze klasy
  - `EntryComparator` - implementuje metodę pozwalającą na porównywanie pozycji w rankingu między sobą.
  - `KeyGeneratorUtility` - klasa pomocnicza odpowiedzialna za tworzenie kluczy RSA,
  - `RSAKeyProperties` - odpowiedzialny za klucze RSA.

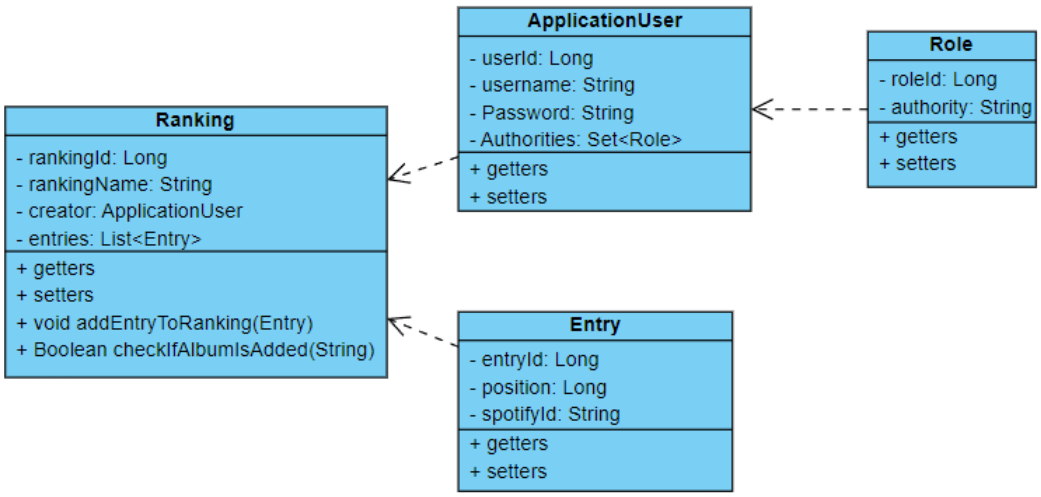
4.2 Frontend

Część frontendowa wykonana jest za pomocą JavaScript i biblioteki React. Wszystkie istotne pliki `.js` znajdują się w katalogu `frontend/src`.

- `App` - główna klasa, zawiera metody dodawania i usuwania pozycji,
- `Login` - zawiera komponenty i metody odpowiedzialne za logowanie i rejestrację jak i również dodawanie i zarządzanie rankingami,
- `MainBody` - główna część aplikacji wyświetlająca ranking,
- `Menu` - komponent dodawania rankingu,
- `Search` - komponent i metodyka wyszukiwania albumów,
- `SearchAlbum` - komponent wyświetlający album na liście wyszukanych,
- `Sidebar` - komponent paska bocznego strony,
- `SortableItem` - komponent pozycji w rankingu, którą będzie można przesuwać myszką,
- `SortableList` - komponent listy rankingu,
- `utils` - metody pomocnicze dot. sesji zalogowanego użytkownika

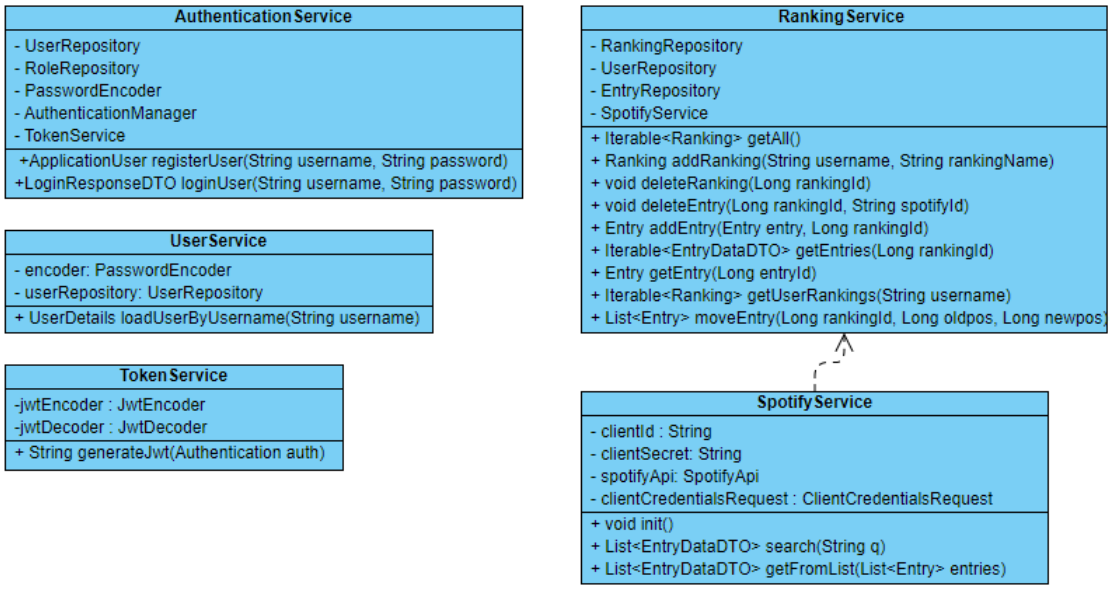
Poza tym są jeszcze pliki `index.js` - główny plik renderujący stronę oraz arkusz stylów `index.css`.

5. Opis działania



Rysunek 5.1: Diagram modeli.

Modele encji reprezentowanych przez Spring Data w bazie danych są zapisane w folderze model. Dla każdego modelu stworzone jest repozytorium rozbudowywujące JPARepository. Repozytoria te są odpowiedzialne za operacje CRUD, które są realizowane przes Spring Data.
Repozytoria używane są przez serwisy, które realizują konkretne operacje potrzebne w aplikacje. Serwisy są następnie wykorzystywane przez kontrolery, które definiują endpointy aplikacji.



Rysunek 5.2: Diagram serwisów.

6. Endpointy

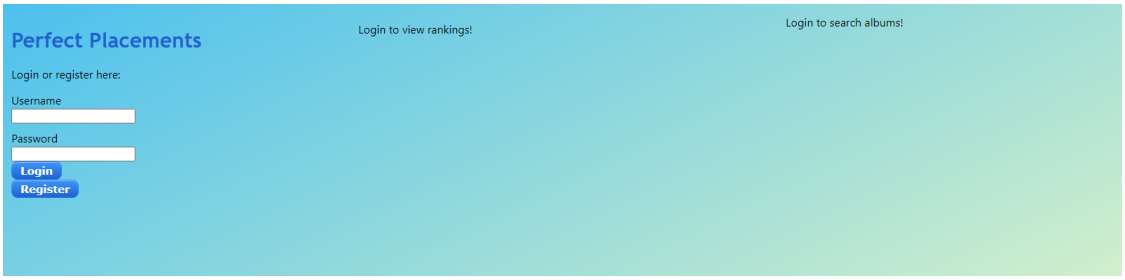
AuthenticationController		
Endpoint	Typ	Opis
/auth/register	POST	Rejestracja użytkownika
/auth/login	POST	Logowanie użytkownika
RankingController		
/ranking/all	GET	Zwraca wszystkie rankingi w bazie
/ranking/{id}	GET	Znajdowanie rankingu o zadanym id
/ranking	POST	Dodawanie rankingu
/ranking/{id}	POST	Dodawanie pozycji do rankingu
/ranking/{id}/sorted	GET	Posortowane pozycje rankingu o id
/ranking/{id}/{entryId}	GET	Zwraca pozycję z rankingu
/ranking/search/{q}	GET	Wyszukiwanie w bazie Spotify
/ranking/user	GET	Wszystkie rankingi użytkownika
/ranking/{id}/{oldpos}/{newpos}	PUT	Zmiana pozycji w rankingu
/ranking/{id}	DELETE	Usuwanie rankingu
/ranking/{id}/{entry}	DELETE	Usuwanie pozycji z rankingu

7. Funkcjonalność strony

Demo z zaprezentowaną działalnością strony znajduje się pod linkiem: `https://www.youtube.com/watch?v=z4pwkf0EIJ8`.

7.1 Podręcznik użytkownika

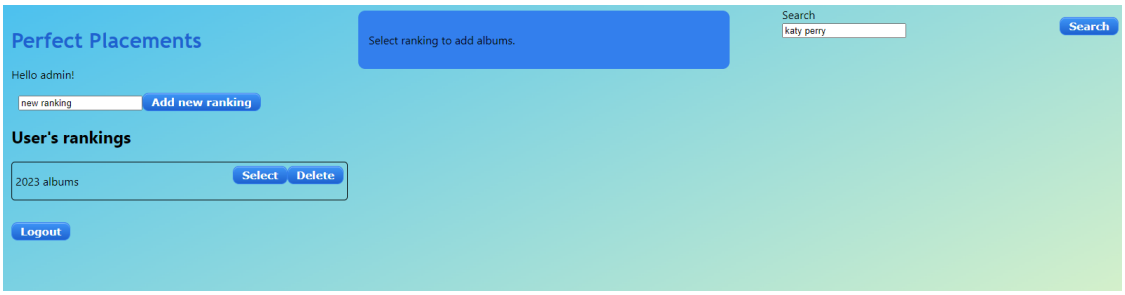
Po wejściu do aplikacji widzimy prostą stronę główną. Funkcjonalność dostępna dopiero po zalogowaniu, co możemy zrobić przez formularz po lewej stronie strony. Przez ten sam formularz możemy się zarejestrować.
Aplikacja posiada zawsze użytkownika **admin** o haśle **password**, na którym można testować funkcjonalność.



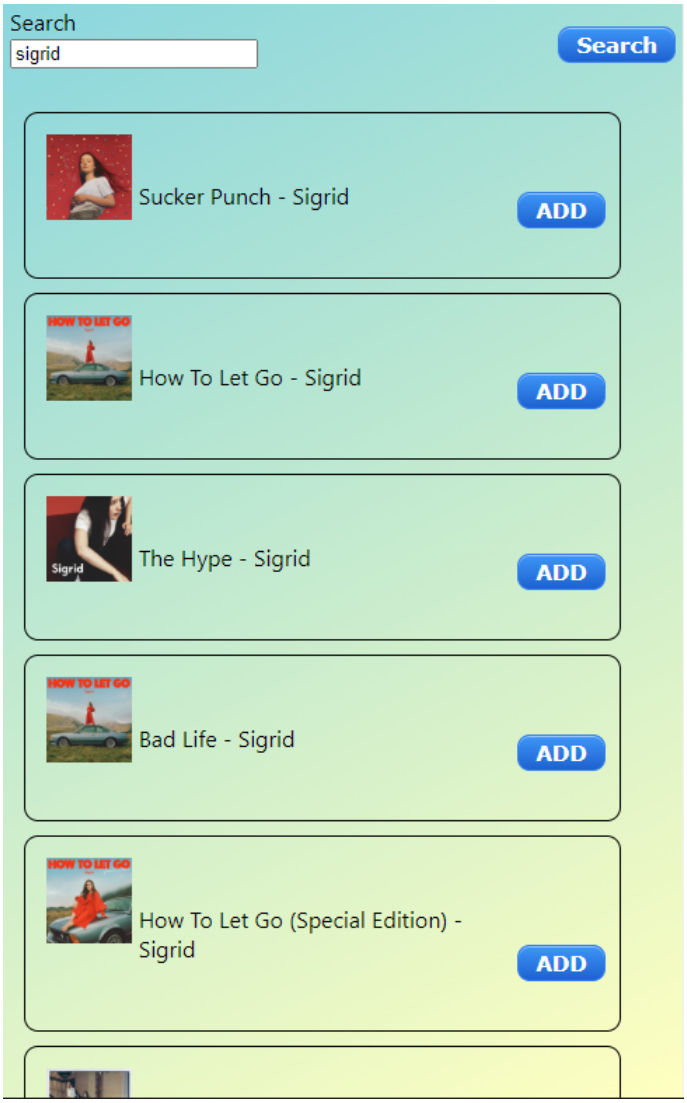
Rysunek 7.1: Strona główna przed zalogowaniem

Po zalogowaniu dostępna jest funkcjonalność: dodawanie rankingów, lista rankingów użytkownika, które można wybierać i usuwać oraz wyszukiwanie. Aby wyszukać wpisujemy tytuł albumu lub nazwę artysty i klikamy przycisk `Search` obok. Aby dodać album do rankingu wcześniej musimy wybrać któryś z rankingów przyciskiem `Select`, wtedy na środku strony pojawi się jego nazwa - oznacza to, że został on poprawnie wybrany.

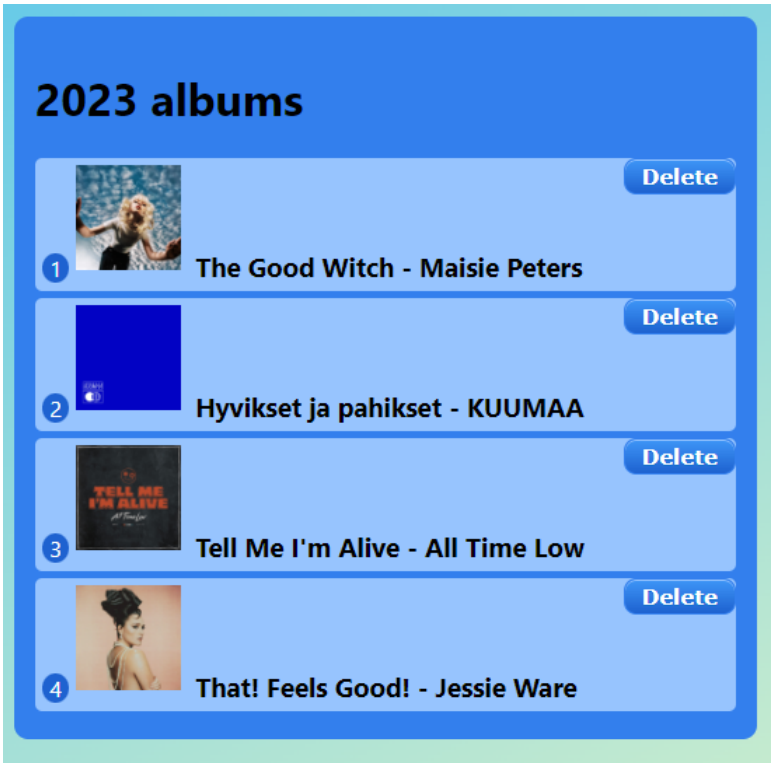
Następnie możemy dodawać dowolną liczbę albumów do rankingu (poza duplikatami), usuwać niechciane i zmieniać ich pozycję poprzez *'drag-and-drop'*. Rankingi możemy usuwać przez przycisk `Delete` przy jego nazwie. Gdy skończymy korzystać ze strony możemy się wylogować.



Rysunek 7.2: Strona główna po zalogowaniu



Rysunek 7.3: Konponent wyszukiwania albumów



Rysunek 7.4: Przykład stworzonego rankingu

8. Wnioski i przemyślenia

- Użycie Spring Data oraz Security bardzo ułatwiło stworzenie części backendowej projektu.
- Jest to mój pierwszy projekt w Reakcie, więc dużo uczyłem się w trakcie pisania i szukania rozwiązań. Z tego powodu struktura komponentów i funkcjonalności nie jest idealna, jednak działa dobrze.
- Próby znalezienia dobrej metody wrzucenia projektu na chmurę nie skończyły się sukcesem - Render nie wspiera docker-compose, rozwiązania, które je wspierają są płatne.
- Docker okazał się dosyć prostym i bardzo wygodnym narzędziem do uruchomienia aplikacji.
- Funkcjonalność aplikacji można rozszerzyć o min.: więcej opcji konta użytkownika, opcję edycji nazwy rankingu, wyświetlanie danych jak data stworzenia i aktualizacji.
- Dodatkowym pomysłem, który można zaimplementować to stworzenie rankingów piosenek w albumie - wtedy wyszukujemy album i od razu pobiera wszystkie piosenki z niego jako pozycje w rankingu, które możemy sortować.
- Po zajęciach seminaryjnych z ZTI prezentujących technologie frontendowe skłaniałem się do zmiany biblioteki na Vue.js, jednak po przeglądnięciu podstaw tych bibliotek zostałem przy React. Wbrew powszechnym opiniom o jego trudności okazał się bardzo łatwy w użyciu - przynajmniej dla takiej prostej aplikacji, gdzie jedyny użyty hook to useState.

- Przez użycie bazy na ElephantSQL pojawiają się czasem kłopoty w ponownym deploy'u aplikacji backendowej, gdy poprzednie połączenie z bazą danych nie zostało poprawnie zamknięte. Po chwili czasu połączenie to się zamknie i Spring uruchomi się poprawnie.