UNIWERSYTET ŚLĄSKI
W KATOWICACH

Wydział Nauk Ścisłych i Technicznych

Kamil Kasprzak

327735

Aplikacja rekomendująca muzykę

PRACA DYPLOMOWA
INŻYNIERSKA

dr Aleksander Lamża

Sosnowiec 2022

Spis treści

3:8126664394

1. Wstęp

Obecnie coraz bardziej popularne są aplikacje webowe (inaczej zwane aplikacjami internetowymi), które oferują o wiele większe możliwości, niż klasyczne strony internetowe, które często mają charakter wyłącznie informacyjny. Tego typu aplikacje działają w środowisku przeglądarki, dzięki czemu nie ma konieczności ich instalacji oraz późniejszej aktualizacji. Są one dostępne z każdego urządzenia, które posiada dostęp do internetu oraz zainstalowaną dowolną przeglądarkę. Najczęstszym ich wykorzystaniem są serwisy społecznościowe, a także sklepy internetowe.

Celem pracy jest zaprojektowanie i realizacja aplikacji webowej rekomendującej muzykę. Typ aplikacji, czyli aplikacja webowa został wybrany dzięki opisanej wcześniej ich popularności oraz ich łatwej dostępności na różnych urządzeniach. Do stworzenia aplikacji zostaną wykorzystane współcześnie używane narzędzia oraz metody ich tworzenia. Głównym narzędziem będzie popularny obecnie framework Angular (według strony internetowej ImpiCode). Jest on rozbudowaną platformą do tworzenia aplikacji webowych. Dodatkowo do zalogowania użytkownika zostanie wykorzystany framework Java Spring Boot, jako wsparcie i zabezpieczenie procesu logowania użytkownika do aplikacji. Główną funkcją aplikacji będzie rekomendacja muzyki użytkownikowi, do czego zostanie wykorzystany jeden z najpopularniejszych na świecie serwisów muzycznych, czyli serwisu Spotify (według raportu firmy analitycznej Counterpoint Research). Oznacza to, że aplikacja będzie służyła celom rozrywkowym. Rekomendacja będzie spersonalizowana na podstawie kryteriów wybranych przez użytkownika takich jak utwór czy wykonawca. Oprócz rekomendacji użytkownik będzie mógł zobaczyć swoje najczęściej odtwarzane utwory.

4:5977136864

2. Analiza problemu

Celem pracy jest stworzenie aplikacji, która będzie w przystępny i łatwy sposób rekomendować użytkownikowi z niej korzystającemu muzykę na podstawie jego wybranych przez niego kryteriów. Przed rozpoczęciem prac nad aplikacją konieczne było rozwiązanie kilku problemów, które uniemożliwiłyby spełnienie podstawowych założeń aplikacji lub mogły powodować istotne problemy podczas tworzenia lub późniejszego używania aplikacji.

Duże znaczenia dla projektu będzie mieć typ aplikacji. Ze względu na rozrywkowy charakter zdecydowano się na stworzenie aplikacji webowej. Dla użytkownika będzie to najwygodniejsza forma i będzie mógł z niej korzystać z dowolnego urządzenia z dostępem do internetu. Z tego typu aplikacji najczęściej korzysta się na urządzeniach mobilnych i komputerach, dlatego strona powinna być responsywna, w czym pomoże aplikacja webowa. Do stworzenia tej aplikacji wybrano framework Angular, który ułatwi proces jej stworzenia.

Aplikacja będzie korzystać z serwisu Spotify. Jest to popularny serwis muzyczny, który udostępnia dane konieczne do działania aplikacji. Udostępnia on dane o utworach, wykonawcach, a także dane użytkownika oraz rekomendowane mu utwory. Bez sprawnej komunikacji z serwisem nie będzie możliwe spełnienia podstawowego założenia omawianego projektu, jakim jest rekomendacja muzyki. Komunikacja będzie odbywać się za pomocą REST API, czyli stylu architektonicznego umożliwiającego pobranie odpowiedniego zasobu. Umożliwia ono łatwą komunikację między frontendem a backendem aplikacji za pośrednictwem internetu, w tym wysłanie i otrzymanie koniecznych danych. Jednak w tym przypadku, wiele żądań wymaga dodatkowo autoryzacji.

Kolejnym ważnym problemem w niniejszej pracy jest wspomniana w poprzednim akapicie autoryzacja użytkownika. W przypadku wykorzystywanego do niniejszej pracy serwisu Spotify, do każdego żądania w nagłówku należy podać token, indywidualny dla każdego użytkownika. Aby możliwe było zalogowanie się użytkownika, należy zarejestrować aplikację w celu otrzymania odpowiednich kluczy, których konieczne jest ukrycie ich przed użytkownikiem. Nie jest możliwe ukrycie danych zapisanych po stronie klienta, ponieważ użytkownik ma dostęp do kodu aplikacji działającej po stronie klienta. W tym celu za pomocą frameworka Java Spring Boot stworzono pośrednika między klientem a serwisem Spotify, którego zadaniem jest uzupełnienie żądania o elementy utajnione. Użytkownik loguje się do swojego konta za pomocą specjalnie przygotowanej strony przez serwis, a następnie

5:1495647762

zostaje przekierowany do strony aplikacji z odpowiednim kodem w adresie, który jest konieczny do otrzymania tokenu. Do otrzymania tokenu należy podać sekretny klucz aplikacji, który został ukryty dzięki wspomnianemu wcześniej pośrednikowi napisanego w frameworku Spring Boot.

Konieczne jest także zabezpieczenie aplikacji przed nieautoryzowanym dostępem do zasobów. Dzięki Spotify API nieautoryzowany użytkownik nie otrzyma danych, jednak konieczne będzie uniemożliwienie także wywołanie zabezpieczonych żądań. Dostęp do widoku gdzie konieczne jest zalogowanie, nie będąc zalogowany spowoduje wystąpieniem błędów. Rozwiązaniem tego problemu jest oferowany przez Angular route guards, który decyduje czy użytkownikowi można wyświetlić dany widok w aplikacji. Dzięki niemu niezalogowanemu użytkownikowi nie zostanie wyświetlona nieodpowiednia strona.

6:4463772185

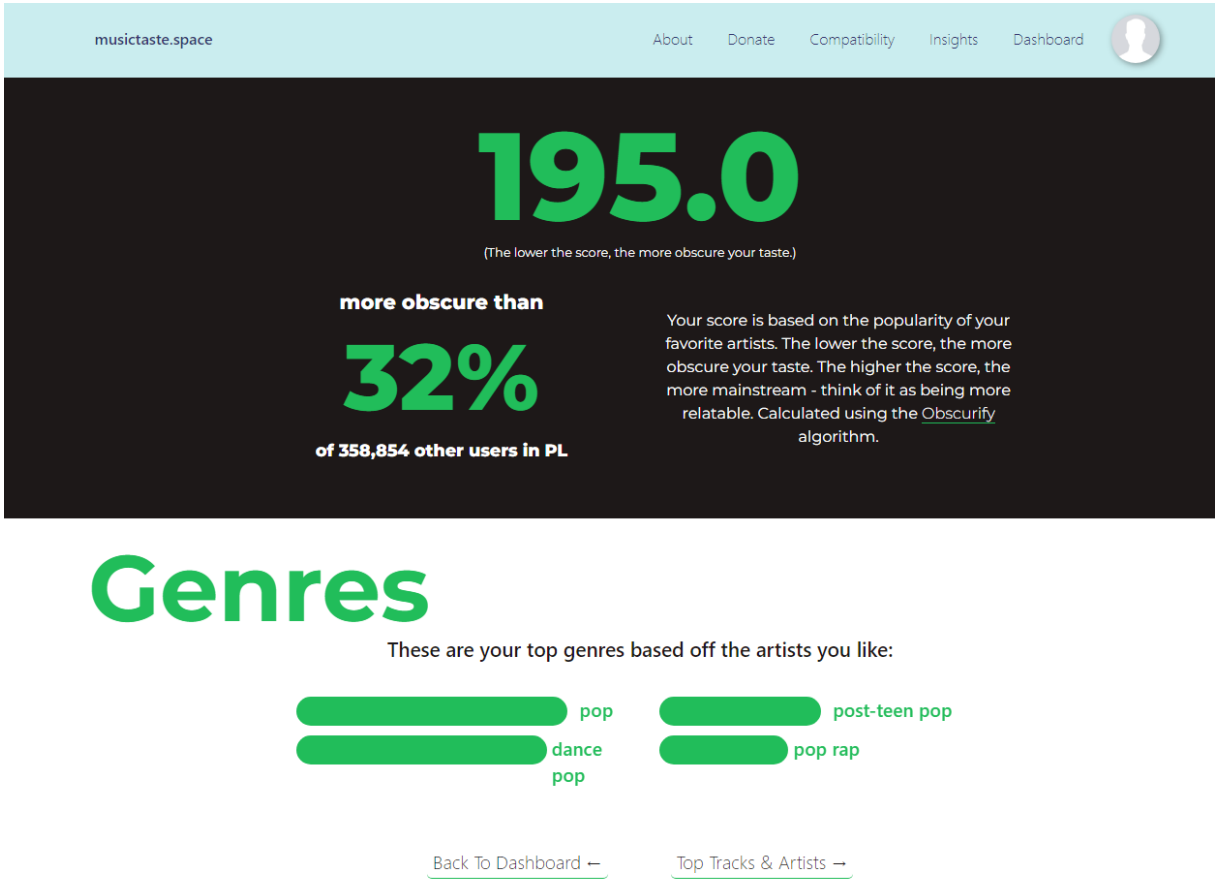
3. Przegląd istniejących rozwiązań

W tym rozdziale zostaną przedstawione trzy aplikacje webowe, które mają podobne możliwości do planowanej aplikacji oraz także korzystają z usług Spotify API. Każda z nich zostanie krótko opisana oraz zostaną wyjaśnione różnice i podobieństwa do planowanej aplikacji.

3.1. Musictaste.space

Musictaste.space jest to aplikacja webowa (dostępna po adresem https://musictaste.space) korzystająca z serwisu Spotify. Podstawową funkcją tej aplikacji jest podgląd na najczęściej wysłuchiwane utwory w różnych przedziałach czasowych. Użytkownik ma też podgląd na najczęściej słuchanych wykonawców. Każdy utwór w serwisie Spotify posiada własne zdefiniowane wartości, przykładowo jego taneczność i akustyczność. Serwis ten wylicza średnią z tych danych dla zalogowanego użytkownika na podstawie jego utworów oraz średnią dla jego kraju, w celu porównania z użytkownikiem. Musictaste.space oferuje także możliwość porównania tych statystyk oraz słuchanych utworów czy artystów ze znajomymi użytkownika za pomocą wygenerowane w aplikacji kodu. Istnieje też możliwość znalezienia osób z podobnym gustem muzycznym.

7:1943276567



Rysunek 1: Aplikacja Musicspace.taste

Aplikacja została zaprojektowana minimalistycznie i w stonowanych kolorach. Jest ona intuicyjna oraz łatwo się w niej poruszać. W niektórych miejscach ta aplikacja nie została dostosowana do urządzeń mobilnych.

Aplikacja będąca tematem tej pracy będzie bardziej skoncentrowana na rekomendowaniu muzyki, ale użytkownik także będzie mieć możliwość podglądu swoich najczęściej wysłuchiwanych utworów lub artystów. Planowana aplikacja będzie także bardziej dostosowana do aplikacji mobilnych.

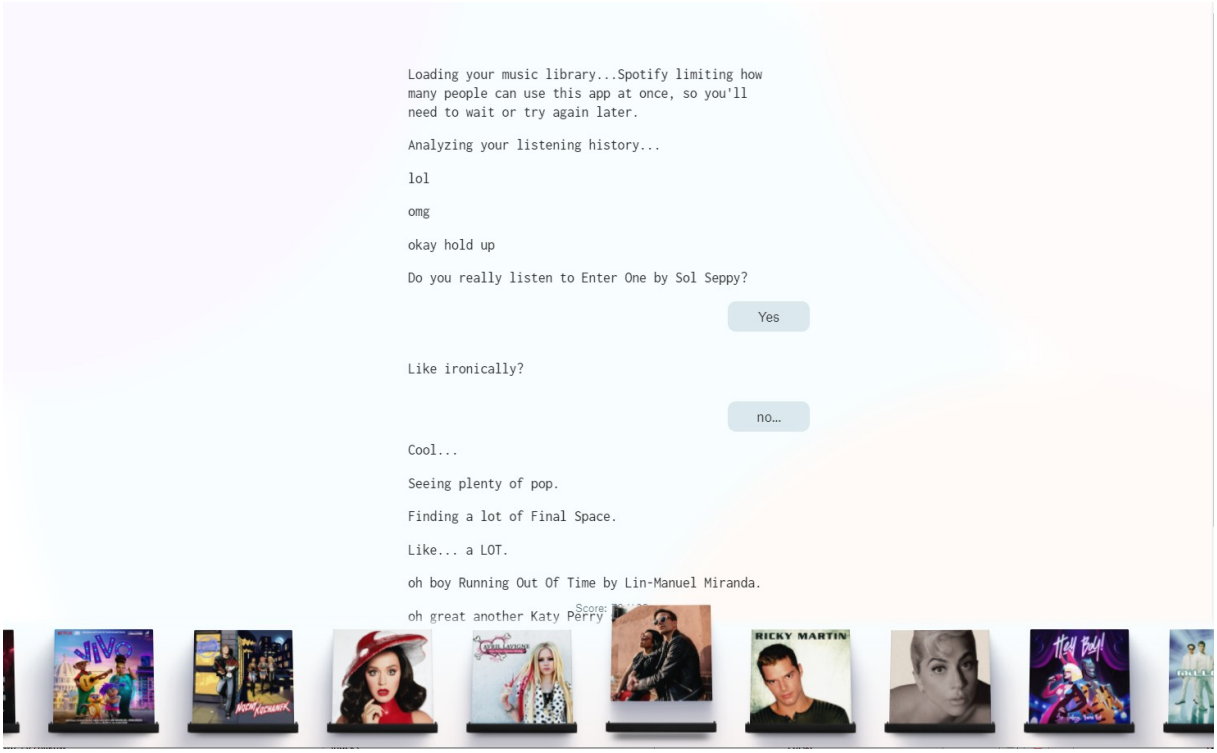
3.2. Judge-my-spotify

Judge-my-spotify to aplikacja webowa (dostępna pod adresem https://pudding.cool/2021/10/judge-my-music) wykorzystująca sztuczną inteligencję w celu oceny gustu muzycznego użytkownika. Aplikacja ocenia go na podstawie ilości i częstości słuchanych utworów danego artysty oraz z jakiego okresu pochodzą najczęściej słuchane

8:9410184725

utwory. Aplikacja jest w stanie ocenić, jakiego wykonawcę użytkownik słucha najczęściej, czy z jakiej dekady najczęściej słucha muzyki.

Jest ona stworzona jeszcze bardziej minimalistycznie od poprzedniej. Analizowanie konta użytkownika jest jej jedyną funkcją, jednak jest ona mocno rozbudowa.



Rysunek 2: Aplikacja Judge-my-spotify

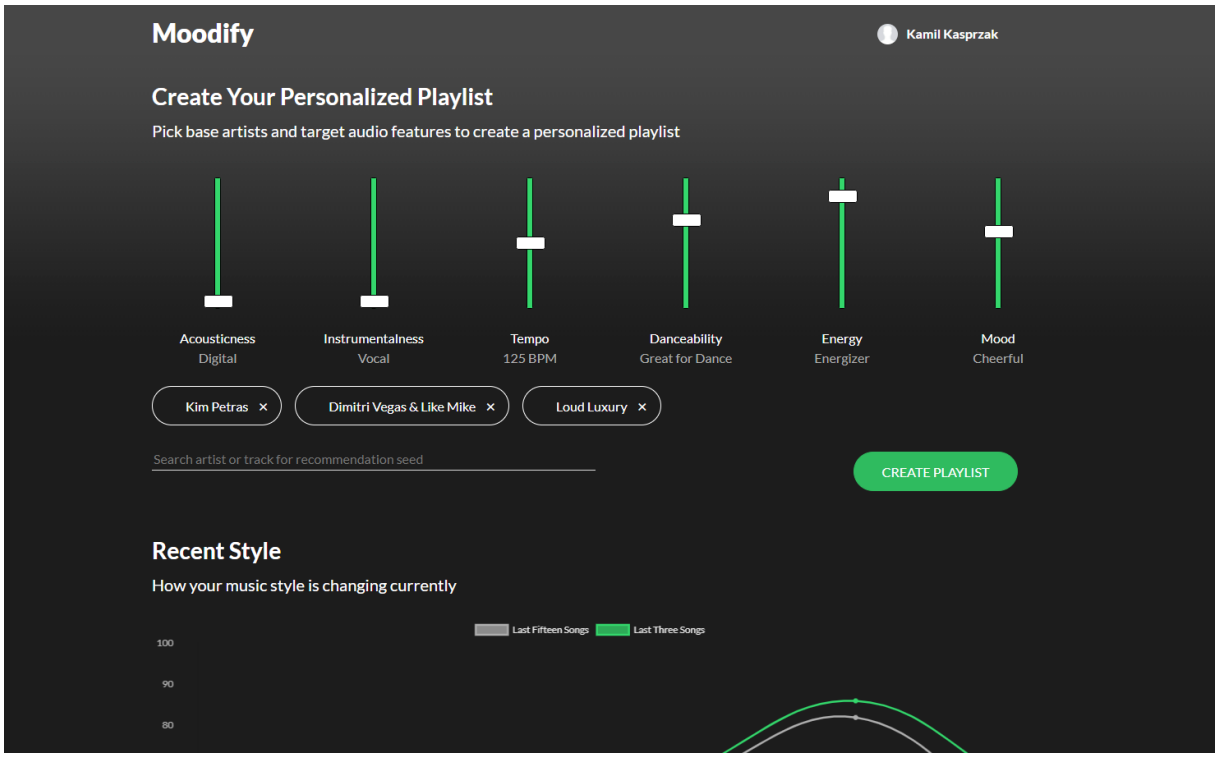
Opisywana aplikacja także korzysta z danych udostępnionych przez serwis Spotify i tak jak w planowanej aplikacji są one wykorzystywane do jej własnych celów. Na korzyść aplikacji przemawia jej czytelność, spowodowana bardzo prostym i nieskomplikowanym interfejsem. Od pozostałych i planowanej aplikacji wyróżnią ją dodatkowo możliwość wykorzystania konta z serwisu Apple Music.

3.3. Moodify

Moodify to aplikacja webowa (dostępna pod adresem https://moodify.app) służąca do generowania playlist w serwisie Spotify. Oprócz tego umożliwia poznanie 5 najczęściej słuchanych utworów w różnych przedziałach czasowych. Generowanie playlisty polega na wybraniu przez użytkownika bazowych artystów, utworów i gatunków, oraz określeniu parametrów takich jak tempo czy nastrój, które przykładowo Musicspace.taste samo ustalało

9:7724190510

bazując na koncie użytkownika. Aplikacja jest mało rozbudowana, dzięki czemu jest prosta w obsłudze. Interfejs jest intuicyjny oraz przyjazny. Aplikacja utrzymana jest w kolorystyce charakterystycznej dla serwisu Spotify.



Rysunek 3: Aplikacja Moodify

Aplikacja ta umożliwia tworzenie playlist na podstawie ustalonych przez użytkownika. Utwory, które generują się na playliście bazują na usłudze od rekomendacji ze Spotify API, która będzie istotnym elementem w planowanej aplikacji.

10:2711514679

4. Koncepcja

W tym rozdziale zostanie przedstawiona ogólna koncepcja aplikacji

4.1 Koncepcja rozwiązania użytkowego

Aplikacja będąca tematem niniejszej pracy jest przeznaczona dla użytkowników serwisu muzycznego Spotify, co oznacza, że ma ona charakter jedynie rozrywkowy. Ze względu na swoje główne założenie, konieczne jest posiadanie konta w tym serwisie i zalogowanie się na te konto w planowanej aplikacji. Dlatego grupą docelową dla aplikacji będzie każdy użytkownik tego serwisu. Po uruchomieniu strony głównej aplikacji przez użytkownika jedyną dostępną opcją jest możliwość zalogowania, ponieważ obecnie nie są przewidziane opcje dla niezalogowanych użytkowników. Po zalogowaniu użytkownik zostanie przekierowany do widoku z główną funkcją aplikacji, jaką jest rekomendacja muzyki. W widoku tym wyświetlony zostanie interfejs, w którym użytkownik będzie mógł wybrać, na jakich utworach, wykonawcach czy gatunkach bazować będzie rekomendacja. Użytkownik musi wybrać przynajmniej jeden, dowolny element z listy bazującej na jego słuchanych utworach, lub dodać go za pomocą wyszukiwarki. Użytkownik musi wybrać od przynajmniej jednego do maksymalnie pięcie elementów. Dodatkowo, za pomocą suwaków użytkownik będzie mógł określić zbliżone tempo, czy taneczność rekomendowanych mu utworów. Wyniki zostaną wyświetlone pod tym interfejsem z podstawowymi informacjami, takimi jak nazwa, czy wykonawca tego utworu. Dodatkowo użytkownik będzie mógł odsłuchać fragmentu utworu, lub całego, jeśli w aplikacji Spotify udostępni strumieniowanie muzyki do aplikacji, dodać go do polubionych na swoim koncie Spotify, czy przejść do strony tego utworu w tym serwisie. Dodatkowo zostanie udostępniony widok, gdzie użytkownik może zobaczyć swoje najczęściej wysłuchiwane utwory. Niezależnie od widoku w lewej części strony wyświetlony będzie panel nawigacyjny, a w górnym, prawym rogu przycisk do wylogowania. Na urządzeniach mobilnych panel nawigacyjny będzie wysuwany za pomocą przycisku umieszczonego w prawym, górnym rogu aplikacji. W dolnej części strony znajdzie się panel służący do sterowania muzyką, gdzie możliwe będzie wznowienie/wstrzymaniu utworu, jego przewinięcie oraz zmiana głośności.

Od podobnych rozwiązań przedstawionych w poprzednim rozdziale będzie większe skoncentrowanie na rekomendowaniu utworów, które te aplikacje nie wykorzystywały, lub

11:1341851338

wykorzystywały je częściowo. Wcześniej omówiona aplikacja Musictaste.space oferowała podgląd na najczęściej wysłuchiwane utwory, a aplikacja Moodify umożliwiała tworzenie playlist z muzyką. Te elementy także zostaną udostępnione w planowanej aplikacji. Dodatkowo planowana aplikacja będzie dostępna w języku polskim, podczas gdy wcześniej omawiane aplikacje są dostępne tylko w języku angielskim.

4.2 Koncepcja rozwiązania technologicznego

Ze względu na charakter aplikacji, najlepszym rozwiązaniem będzie stworzenie aplikacji webowej, która będzie dostosowana do urządzeń o różnej wielkości urządzeniach, jednak pierwszeństwo będą miały urządzenia z szerokim ekranem. Obecnie dostępnych jest duża ilość frameworków do tworzenia aplikacji webowych po stronie klienta. Jednym z nich jest Angular, opracowany przez Google, który został wybrany do stworzenia tej aplikacji. Jego zadaniem jest ułatwienie tworzenia takich aplikacji za pomocą komponentów, czy dostarczonych wraz z nim bibliotek. Został on wybrany z powodu dużego ułatwienia procesu tworzenia oraz swojej popularności (według wspomnianego wcześniej badania serwisu Impicode). W przeciwieństwie do tych dwóch rozwiązań Angular wykorzystuje język TypeScript, będący nadzbiorem języka JavaScript. Innymi popularnymi rozwiązaniami są React oraz Vue. Do projektu zostanie wykorzystany także framework Spring boot z językiem Java, którego celem będzie zabezpieczenie danych przed użytkownikiem, wymaganych do autoryzacji aplikacji po stronie Spotify (zalecenia samego serwisu). Spring Boot jest ułatwionym w użytkowaniu frameworkiem Spring. Innym możliwym rozwiązaniem w tym przypadku było wykorzystanie Java EE (Enterprise Edition). Spring Boot został wybrany dzięki swojej większej popularności i przystępności.

Użytkownik musi się zalogować do aplikacji w celu otrzymania tokenu. Token jest konieczny przy wywoływaniu żądań w celu pobrania danych koniecznych do działania aplikacji. Aplikacja w celu poprawnego działania musi korzystać z tych danych, które zawierają między innymi rekomendacje, dane o utworach, wykonawcach czy dane dotyczące użytkownika.

W samym procesie tworzenia zostanie wykorzystany dodatkowo system kontroli wersji – git. Dzięki któremu będzie łatwiejsza kontrola na plikami i nad ich zmianami. Do pisania kodu po stronie klienta wykorzystany Visual Studio Code, a do części po stronie serwera Intellij IDEA. Obydwa oprogramowania koloryzują składnie języka oraz zawierają

12:8850112070

podpowiedzi do pisanego kodu. Są to mocno rozbudowane edytory, zawierające wiele ułatwień, przykładowo przy korzystaniu z systemu kontroli wersji, czy w poruszaniu się po projekcie.

13:9558151571

5. Projekt ogólny

W tym rozdziale zostaną bardziej rozpisane szczegóły dotyczące tworzenia projektu, które zostały wstępnie opisane w poprzednim rozdziale.

5.1 Wymaganie funkcjonalne

- Logowanie do konta Spotify.
  - Użytkownik loguje się na swoje konto Spotify, żeby móc korzystać z głównych funkcji aplikacji, zwrócony zostaje token, konieczny by móc korzystać z udostępnionych danych.
- Główną funkcją aplikacji jest rekomendacja utworów muzycznych na podstawie wybranych przez użytkownika parametrów.
  - Wymagane jest wcześniejsze zalogowanie się.
- Użytkownik może zobaczyć swoje statystyki takie jak najczęściej słuchane utwory czy wykonawcy, które są wykorzystywane do przez Spotify API do rekomendacji muzyki.
  - Wymagane jest wcześniejsze zalogowanie się.
- Użytkownik może odtworzyć fragment lub całość utworu i zarządzać jego odtwarzaniem.
  - Wymagane jest wcześniejsze zalogowanie się, by móc odtworzyć utwory.
  - Użytkownik może zmieniać głośność utworu, wstrzymać lub wznowić jego odtwarzanie oraz go przewinąć.
  - W celu odtworzenia całego utworu użytkownik musi w aplikacji Spotify udostępnić strumieniowanie muzyki do aplikacji oraz posiadać konto premium.

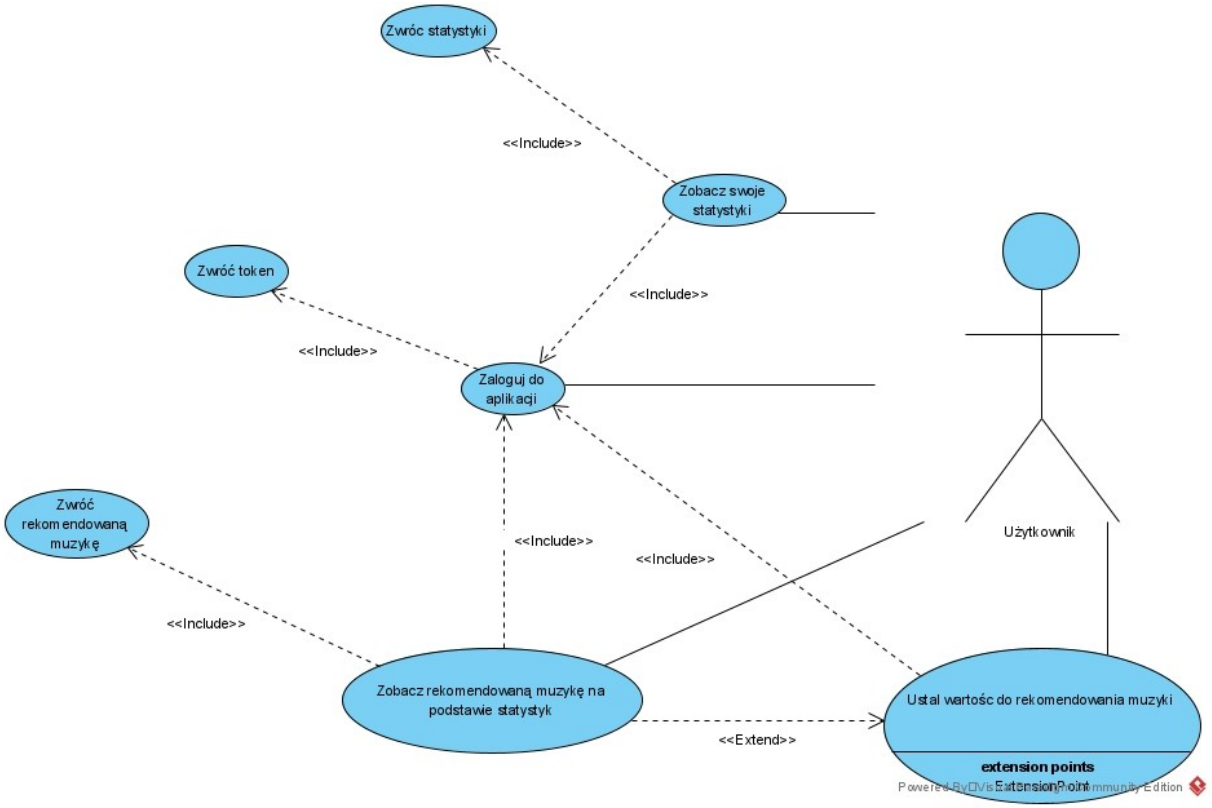
5.2 Wymaganie niefunkcjonalne

- Aplikacja powinna być łatwa w obsłudze.
  - Interfejs powinien być intuicyjny oraz być zaprojektowany w najczytelniejszej możliwie formie.
- Wymagane jest wcześniejsze zalogowanie się, wykorzystując konto użytkownika w serwie Spotify.
  - Warunek konieczny, by móc korzystać ze Spotify API.

14:5395077679

Wymagane jest wcześniejsze zalogowanie się, by móc odtworzyć utwory.

- Aplikacja przeznaczona jest na najnowsze wersje współczesnych przeglądarek z wykorzystaniem współczesnych technologii i rozwiązań.
- Aplikacja musi działać wydajnie.
- Aplikacja musi być zaprojektowana zarówno do wyświetlania na ekranach monitorów komputerów oraz urządzeniach mobilnych.
- Użytkownik nie może mieć wrażenia, że aplikacje nie reaguje na jego polecenia.



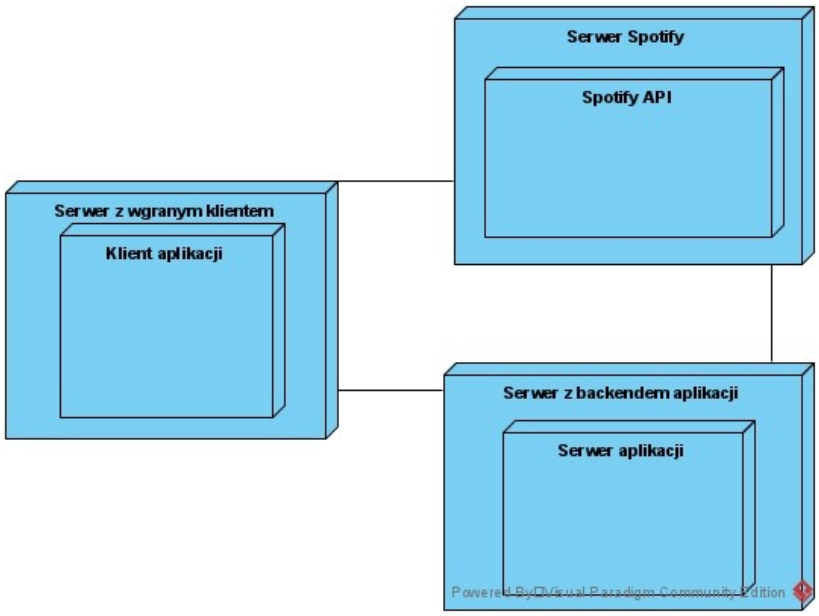
Rysunek 4: Diagram przypadków użycia

5.3 Architektura systemu

Aplikacja będzie składać się z trzech warstw. Dwie z nich klient oraz z serwer, który jest używany będzie tylko do logowania, zostaną stworzone specjalnie na potrzeby aplikacji. Dodatkowo system będzie korzystać z już istniejących usług wystawionych przez Spotify API, który stanowić będzie trzecią warstwę. Klient będzie działać w środowisku przeglądarki, która będzie się komunikować bezpośrednio ze Spotify API. Wyjątkiem będzie logowanie, gdzie będzie się komunikować z usługą wystawioną przez warstwę serwerową. Zdecydowano

15:9275476561

się na to rozwiązanie w celu ukrycia istotnych danych, służących do autoryzacji aplikacji, które nie powinny być w żaden sposób dostępne dla użytkownika, co było już opisane w poprzednich rozdziałach i jest rekomendacją samego serwisu.



Rysunek 5: Diagram architektury aplikacji

5.4 Narzędzia realizacji

W tym podrozdziale znajdują się opisy narzędzi, które zostały wykorzystane do stworzenia tytułowej aplikacji, ze szczególnym naciskiem na elementy tych narzędzi będące znaczące dla procesu tworzenia aplikacji.

5.4.1 Angular

Aplikacja, będąca tematem tej pracy inżynierskiej zostanie stworzona z wykorzystaniem frameworka Angular. Jest to najważniejsze narzędzie używane w tym projekcie, w którym zostanie napisana większa część aplikacji.

Angular – framework do tworzenia frontendu aplikacji webowych napisany i wykorzystujący język TypeScript rozwijany przez Google. Jest on kontynuacją frameworka AngularJS, wydanego w październiku 2010 roku. Oprócz TypeScriptu do tworzenia aplikacji wykorzystuje się on także język znaczników HTML oraz CSS. Istnieje możliwość innego języka kompilowanego do niego. W tej aplikacji zostanie wykorzystany SCSS, czyli jeden z kilku dostępnych preprocesorów CSS. Na Angularze działają strony takich firm, jak Google czy Microsoft.

16:1830788020

Obecna wersja Angulara znana również jako Angular 2.0 została oficjalnie opublikowana w maju 2016. Framework ten został on napisany w języku TypeScript, w przeciwieństwie do pierwszej wersji opartej na języku JavaScript.

Jego zadaniem jest ułatwienie tworzenia dużych aplikacji zgodnych z modelem MVC – model-widok-kontroler. Aplikacje tworzy się przy użyciu komponentów które można wielokrotnie wykorzystywać podczas tworzenia aplikacji. Aplikacje są jednostronne (Single page application), czyli posiadają jeden plik HTML, który dynamicznie zmienia swoją zawartość na podstawie czynności wykonywanych przez użytkownika. Framework ten wprowadza również między innymi dwustronne wiązanie danych, czy bibliotekę HttpClient ułatwiającą tworzenie żądań HTTP. Aplikacje tworzy się za pomocą komponentów, które można wielokrotnie używać.

Framework oparty jest na języku programowania zorientowanym obiektowo TypeScript będący nadzbiorem języka JavaScript. Największą różnicą między tymi dwoma języka jest wprowadzenie typowanych zmiennych, co zostało odzwierciedlone w nazwie języka. Oprócz tego dodano także między innymi klasy oraz interfejsy. TypeScript jest ostatecznie kompilowany do JavaScript, więc każdy kod napisany w JavaScriptcie jest w pełni działającym kodem w języku TypeScript.

Angular jest obecnie jednym z najpopularniejszych frontendowych frameworków obok Vue, czy biblioteki React. Z tego powodu został on wybrany jako główne narzędzie tej pracy. Innymi powodami skłaniającymi do jego wyboru jest opisany już wcześniej język TypeScript oraz liczne udogodnienia, jakie ten framework oferuje.

Serwisy w Angularze

Serwisy w Angularze to klasy z wydzielonym fragmentem kodu, można je wykorzystać w celu wydzielenie kodu z komponentu. Twórcy Angulara zalecają, by w komponentach przechowywać logikę związaną z wyświetlaniem, pozostałą logikę wydzielić do serwisu. Istnieje tylko jeden serwis na całą aplikację.

Serwisy wyróżniają się specjalnym oznaczenie w postaci adnotacji @Injectable. W tej adnotacji znajduje się własność providedIn: ‘root’**,** co oznacza, że komponent można wstrzyknąć do każdego komponentu, jaki tylko istnieje w aplikacji. Wstrzyknięcia dokonuje się w konstruktorze.

17:8224490651

Interceptory

Interceptory to klasy, których zadaniem jest przechwycenie żądania Http i jego modyfikacja. Klasy ten implementują interfejs HttpInterceptor zawierającą metodę intercept(), która jest wykorzystywana przy każdym żądaniu. Można w nim umieścić kod, który powtarza się podczas każdego żądania.

Routing

W aplikacja jednostronnych to co jest wyświetlanie zależy od aktualnego adresu. Należy zadeklarować, dla jakiej strony wyświetlany będzie, jaki komponent w miejscu <router-outlet></router-outlet>. Aplikacje będzie się składać z kilku podstron, więc konieczne będzie wykorzystanie Angularowego routingu.

Żądania HTTP

Kolejnym, bardzo ważnym elementem do działania aplikacji jest wysyłanie żądań http, w celu komunikacje między warstwami aplikacji. Aplikacja napisana w Angularze komunikuje się z serwerem za pomocą protokołu HTTP. Klient wysyła żądanie, na które zostaje wygenerowana odpowiedź.

Żądania składa się z:

- Nagłówka, zawierająca podstawowe dane dotyczące żądania, przykładowo takie jak adres hosta, czy cookie. Dla aplikacji istotny będzie nagłówek zawierający token w celach autoryzacji.
- Ciała, czyli części zawierająca przesyłane dane. Nie każda metoda HTTP zawiera ciało. Podstawowym formą zapisu danych to zapis w postaci pliku JSON.

Metody HTTP służą do ograniczenia możliwości komunikacji. Każdej metodzie można zdefiniować jej działania po stronie serwera, czy zabezpieczyć ją przed nieuprawnionym użyciem.

Najczęściej wykorzystywane metody to:

- GET - Służy do pobrania danych z serwera, nie zawiera ciała.
- POST - Przesyła dane do serwera w ciele, wykorzystywana przy tworzeniu nowych danych.
- PUT – Działa na tej samej zasadzie co metoda POST, najczęściej wykorzystywana do aktualizacji danych.

18:9221431606

- DELETE – Metoda służąca do usuwania danych.

Angular dostarcza klasę HttpClient, która umożliwia wysyłanie żądań http. Metody tej klasy zwracają obiekt Observable, który należy zasubskrybować**,** aby otrzymać wynik. Ta metoda przyjmuje dwie funkcje, pierwsza wywoływana jest w przypadku poprawnego otrzymania danych, druga w przypadku wystąpienia błędu. Działa ona asynchronicznie, odpowiednia funkcje wykonuje się po otrzymaniu odpowiedzi.

Moduły

Aplikacje Angularowe można podzielić na moduły, czyli fragmenty zawierające klasy i inne elementy takie jak komponenty, serwisy czy modele. Każda aplikacja zawiera przynajmniej jeden moduł. Służą one do zwiększenia porządku w kodzie. W tytułowej aplikacji zostaną wydzielone osobne moduły, które zawierają elementy, które wspólnie tworzą jedną funkcję serwisu.

5.4.2 Framework Spring Boot

Spring Boot to framework oparty na języku Java, którego celem jest ułatwienie i szybsze tworzenie aplikacji internetowych w tym języku. W przypadku tego projektu, aplikacja napisana w tym frameworku jest pośrednikiem podczas logowania użytkownika między aplikacją Angularową a Spotify API. Powodem tego jest konieczność ukrycia przed użytkownikiem danych koniecznych do autoryzacji aplikacji. Z tego powodu framework ten nie będzie tak omawiany jak Angular w poprzednim rozdziale.

Java jest to obiektowy język programowania oparty na obiektach, który pojawił się już w 1995. Kod napisany w tym języku kompilowany jest do kodu bajtowego, który wykonywany jest na maszynie wirtualnej. Według serwisu z ofertami pracy No Fluff Jobs w 2019 język ten był najbardziej popularnym językiem programowania na rynku pracy.

5.4.3 Spotify API

Nie jest to narzędzie w takim samym rozumieniu, jak te opisane we wcześniejszych podrozdziałach, jednak jest istotnym elementem opisywanej aplikacji.

Spotify jest to serwis strumieniowy umożliwiający wygodny sposób słuchania muzyki oraz podcastów. Obecnie jest to jedna z najpopularniejszych tego typu usług. Na początku 2019 serwis posiadał 217 milionów użytkowników.

19:4052999934

Spotify oferuje liczne usługi, umożliwiającą pobranie danych w formacie typu JSON. Pobrać można między innymi dane użytkownika, jego polubione utworu, czy utworzone playlisty wraz z możliwością edycji ich. Udostępnione są także usługi związane z utworami czy wykonawcami. Te usługi są podstawą działania aplikacji.

5.4.4 Docker

Docker to narzędzie umożliwiający konteneryzację aplikacji. Każdy kontener aplikacji posiada pliki projektu, który ma obsługiwać oraz środowisko potrzebne to jego uruchomienia. Kontenery działają niezależnie od siebie, posiadając własne środowisko, system czy adres sieciowy. Ich głównym celem jest ułatwienie uruchamiania aplikacji na dowolnym urządzeniu. W projekcie wykorzystano Dockera do ułatwienia procesu tworzenia klienta aplikacji, umożliwiając szybkie uruchomienie obrazu Dockera z serwerem.

5.4.5 Git

Git to rozproszony system kontroli wersji, czyli oprogramowaniem, którego celem jest śledzenie zmian w kodzie i ułatwienie dzielenia się kodem z innymi. Tworzone jest repozytorium, które po umieszczeniu na odpowiednim serwerze umożliwia łatwą wspólną pracę wielu osobom przy jednym projekcie. W tym projekcie wykorzystano serwer gita - GitHub, jednego z dostępnych na rynku serwisu hostingowego.

20:1214559018

6. Implementacja

W tym rozdziale zostanie opisana implementacja aplikacji.

6.1 Implementacja w Angularze

W tym podrozdziale zaprezentowanie zostanie praktyczne zaimplementowania części działającej po stronie klienta, czyli napisanej za pomocą frameworka Angular. Zostały tu omówione najciekawsze fragmenty. Do najciekawszych fragmentów zostały także dołączone fragmenty kodu.

6.1.1 Struktura projektu w Angularze

Wszystkie pliki projektu zostały uporządkowane w projekcie zgodnie z przyjętymi podczas tworzenia aplikacji zasadami:

- W folderze components znajdują się pliki komponentów.
- W folderze models znajdują się klasy obiektów używane w aplikacji.
- W folderze modules znajdują się wydzielone moduły, których struktura jest podobna do widocznej struktury.
- W folderze pages znajdują się komponenty, które są wykorzystywane jako główne komponenty podstron.
- W folderze security są klasy związane z bezpieczeństwem i autoryzacją, które nie klasyfikują się do innych folderów, przykładowo są to interceptory.
- W folderze services znajdują się serwisy.

Struktura modułu jest bardzo podobna do głównej struktury projektu, wyjątkiem jest brak folderu security i modules. W pozostałych folderach znajdują się elementy, które są używane tylko w tym module.

6.1.2 Logowanie

Ważnym elementem działania logowania jest serwis loginService. W tym serwisie znajduje się cała logika związana z logowaniem do aplikacji oraz metody sprawdzające, czy użytkownik jest zalogowany.

21:4575855603

Na początku użytkownik zostaje przekierowany do odpowiedniej strony w serwisie Spotify. W linku z przekierowaniem znajduje się link powrotu do aplikacji, oraz lista praw, która aplikacja zyska do konta użytkownika. Po poprawnym zalogowaniu użytkownik zostaje przekierowany z powrotem do aplikacji.

Listing 1: Metoda przekierowująca do logowania w Spotify

```
showWindowlogin(): void {
    let scope = environment.spotifyApp.scope;
    let redirect = encodeURIComponent(environment.spotifyApp.redirect_uri);
    window.location.href =
      'https://accounts.spotify.com/authorize?client_id=' +
      environment.spotifyApp.client_id +
      '&response_type=code&redirect_uri=' +
      redirect +
      '&scope=' +
      scope +
      '&show_dialog=true';
  }
```

Po poprawnym zalogowaniu użytkownik zostaje przekierowany z powrotem do aplikacji wraz z kodem, który zostaje następnie wymieniony na token. Token ten wykorzystywany jest do autoryzacji użytkownika. Sama metoda wywołuje żądanie do serwisu Spotify metodą post, która w ciele zawiera kod oraz link powrotu, który służy tylko autoryzacji. Po otrzymaniu odpowiedzi za pomocą metody handleAuthentication token, jego ważności oraz token odświeżający są zapisywane w przeglądarce. Dzięki temu, po odświeżeniu strony token nie zostaje utracony. Dodatkowo zostaje wyemitowana wiadomość do zalogowaniu do innych komponentów w aplikacji.

22:2793379266

Listing 2: Metoda do otrzymania tokena

```
getloginToken(code: string): Observable<User> {
  return this.http
    .post<User>(`${environment.spotifyApp.api}/getToken`, {
      code: code,
      redirect_uri: environment.spotifyApp.redirect_uri,
    })
    .pipe(
      tap((resData) => {
        this.handleAuthentication(
          resData.access_token,
          resData.expires_in,
          resData.refresh_token
        );
      })
    );
}

private handleAuthentication(
  access_token: string,
  expires_in: number,
  refresh_token: string
): void {
  const date = new Date().getTime() + expires_in * 1000;
  localStorage.setItem('access_token', access_token);
  localStorage.setItem('expire_date', date.toString());
  localStorage.setItem('refresh_token', refresh_token);
  this.loginEmitter.next(true);
}
```

Oprócz tych metod znajdują się dodatkowo dwie proste metody. Pierwsze służy do sprawdzenia, czy zapisany jest token oraz datę jego wygaśnięcia. Druga służy do wylogowania użytkownika. Usuwa ona zapisane informacje związane z tokenem z aplikacji, oraz przekierowuje użytkownika do strony głównej.

6.1.3 Interceptor tokenInterceptor

Klasa ta implementuje interfejs HttpInterceptor z metodą intercept. Interceptor w aplikacji przechwytuje każde wysłane żądanie HTTP i dołącza do niego w nagłówku token użytkownika, jeśli jest on zalogowany, dzięki czemu nie ma potrzeby powtarzania fragmentów kodu w wielu serwisach. Oprócz tego ułatwia globalne modyfikowanie zasady działania dodania tokena oraz zwiększa przejrzystość i czytelność kodu. Wykorzystuje on omawiany w poprzednim rozdziale serwis.

23:5391848735

Listing 3: Metoda interceptora

```
intercept(
  request: HttpRequest<unknown>,
  next: HttpHandler
): Observable<HttpEvent<unknown>> {
  if (this.authService.isLogin()) {
    request = request.clone({
      setHeaders: {
        Authorization: `Bearer ${localStorage.getItem('access_token')}`,
      },
    });
  } else {
    this.authService.logout(true);
  }
  return next.handle(request);
}
```

6.1.4 AuthGuard

Ta klasa odpowiedzialna jest za sprawdzenie czy można wyświetlić podstronę użytkownikowi. Wykorzystywany jest on w podstronach, w których konieczne jest bycie zalogowanym. Dzięki czemu nie wystąpią niepożądane błędy. Jeśli użytkownik nie ma pozwolenia na wejście na daną stronę, zostaje przekierowany do strony z logowaniem.

Listing 4: Metoda AuthGuarda

```
  canActivate(): boolean {
    if (this.authService.isLogin()) {
      return true;
    }
    this.authService.logout(true);
    return false;
  }
}
```

6.1.5 Strona z rekomendacjami

Jest to wydzielony moduł aplikacji odpowiedzialny za wszystkie elementy, które odpowiedzialne są za wyświetlenie użytkownikowi rekomendowanych utworów. Zawiera on trzy serwisy. Pierwszy służy do pobrania z serwisu Spotify odpowiednich rekomendacji. Jego główna metoda dodatkowo wykorzystują inną metodę, której celem jest stworzenie linku wraz z ustawieniami wybranymi przez użytkownika.

24:8652108377

Listing 5: Metoda pobierająca rekomendacje dla użytkownika

```
getRecommend(): Observable<{ tracks: SongInfo[] }> {
  return this.http.get<{ tracks: SongInfo[] }>(this.createLink().href).pipe(
    tap(
      (data) => {
        this.recommendSongs = data.tracks;
        this.recommendChanged.next(this.recommendSongs);
      },
      (error) => {
        this.recommendSongs = [];
        this.recommendChanged.next(this.recommendSongs);
      }
    )
  );
}
```

Drugi serwis zawiera proste metody wspomagający zarządzaniem danymi. Trzeci serwis służy do wyszukania z serwisu Spotify wykonawców i utworów, na podstawie wpisanego przez użytkownika teksty w odpowiedniej wyszukiwarce.

Listing 6: Metoda wysyłająca żądanie do wyszukania

```
public search(query: string) {
  const url = new URL(this.SEARCH_LINK);
  url.searchParams.append('q', query);
  url.searchParams.append('type', 'track,artist');
  url.searchParams.append('limit', '3');
  return this.http.get<SearchResponse>(url.href);
}
```

Moduł ten zawiera główny komponent odpowiedzialny za wyświetlenie interfejsu oraz rekomendowanych utworów. Interfejs został podzielony na kolejne dwa komponenty. Pierwszy służy do wybrania zbliżonych wartości parametrów, takich jak energiczność czy tempo utworu. Drugi umożliwia wybranie utworów, wykonawców, oraz gatunki, na jakich bazować będą rekomendowane utwory. Wybór możliwy jest z przygotowanej listy lub z samodzielnego wyszukania.

Na podstawie wyników rekomendacji, użytkownik może utworzyć playlistę, zawierającą rekomendowane utwory. Metody związane z tą funkcją znajdują się w osobnym serwisie.

25:8551326848

Listing 8: Metody serwisu z tworzeniem playlist muzycznych

```
public createPlaylist() {
  let id = this.userService.user.id;
  let link = `https://api.spotify.com/v1/users/${id}/playlists`;
  return this.http.post<{ id: string }>(link, { name: 'Twoje rekomendacje' });
}

public addTracksToPlaylist(tracks: SongInfo[], playlistId: string) {
  const ids = tracks.map(track => track.uri);
  return this.http.post(
    `https://api.spotify.com/v1/playlists/${playlistId}/tracks`,
    {uris:ids}
  );
}
```

Dodatkowo moduł który, oraz moduł z następnego rozdziału korzystają z współdzielonego modułu, który zawiera serwisy wykorzystywane w tych dwóch modułach. Przykładowo zawiera on serwis z metodami związanymi z dodawaniem utworów do polubionych, a także serwis z tworzeniem playlist muzycznych.

6.1.6 Strona z najpopularniejszymi utworami i wykonawcami

Jest to kolejny moduł, który zawiera dwie strony. Pierwsza służy do wyświetlenia listy najczęściej słuchany utworów, druga do najczęściej słuchanych wykonawców. Na obydwóch stronach użytkownik może wybrać przedział czasu, w jakim najczęściej słuchał utworów, czy wykonawców.

Moduł ten korzysta z serwisu, które ze względu na wykorzystanie go w poprzednim module, zostały przeniesiony do kolejnego, współdzielonego modułu.

Listing 7: Metody pobierające najpopularniejsze utwory i wykonawców

```
getTopTracks(timeRange: string): Observable<{ items: SongInfo[] }> {
   return this.http.get<{ items: SongInfo[] }>(`https://api.spotify.com/v1/me/top/tracks?time_range=$
{timeRange}`);
 }
```


26:1834487144

```
  getTopArtists(timeRange: string): Observable<{ items: Artist[] }> {
    return this.http.get<{ items: Artist[] }>(`https://api.spotify.com/v1/me/top/artists?time_range=$
{timeRange}`);
  }
```

6.1.7 Odtwarzacz muzyki

Jedną z możliwości, jakie oferuje aplikacja jest odtwarzanie 30 sekundowych fragmentów muzyki, lub jeśli użytkownik posiada konto premium w serwisie Spotify to całych utworów. W rzeczywistości składa się on z prostego komponentu, który zawiera przyciski do sterowania, pasek do przewijania oraz do ustawiania głośności. Zdecydowanie ciekawszą jego stroną są serwisy, pierwszy wykorzystywany do uruchamiania fragmentów utworów, drugi do łączenia się z serwisem Spotify z odtwarzaniem w nim strumieniowanych do aplikacji utworów muzycznych.

Pierwszy serwis korzysta z linków udostępnionych przez serwis Spotify z fragmentami utworów. W tym serwisie wykorzystany jest obiekt HTMLAudioElement, który służy do uruchomienia i zarządzania fragmentem utworu.

Listing 8: Przykładowo metoda z pierwszego serwisu od odtwarzania muzyki

```
private loadAndPlay(newTrack: SongInfo) {
  this.isLoading.next(true);
  this.audio.load();
  this.audio.onloadeddata = () => {
    this.trackData.next(newTrack);
    this.trackDuration.next(this.audio.duration);
    this.audio.volume = this.volume;
    this.trackVolume.emit(this.volume);
    this.audio.play();
    this.isPaused.next(false);
    this.playInterval();
    this.isLoading.next(false);
  };
}
```

Drugi serwis inicjuje połączenie z aplikacją Spotify. Dzięki temu możliwe jest odtwarzanie w aplikacji całych fragmentów utworów. Jeśli użytkownik posiada konto premium może w kliencie aplikacji Spotify odtwarzać i sterować w tytułowej aplikacji.

27:4812287717

Listing 9: Metoda inicjująca połączenie z aplikacją Spotify

```
private loadAndPlay(newTrack: SongInfo) {
  this.isLoading.next(true);
  this.audio.load();
  this.audio.onloadeddata = () => {
    this.trackData.next(newTrack);
    this.trackDuration.next(this.audio.duration);
    this.audio.volume = this.volume;
    this.trackVolume.emit(this.volume);
    this.audio.play();
    this.isPaused.next(false);
    this.playInterval();
    this.isLoading.next(false);
  };
}
```

6.2 Implementacja w Java Spring Boot

W tej części zostanie przedstawiony sposób implementacji niewielkiej części aplikacji po stronie Javy.

Głównym powodem wykorzystania Javy w tym projekcie była konieczność ukrycia danych, z których należało korzystać w celu zdobycia tokenu, które są wymagane do autoryzacji użytkownika i korzystania z aplikacji. Backend jest pośrednikiem między Angularem i Spotify API, gdzie w żądaniu dopisywane są odpowiednie dane i przesyłane dalej.

6.2.1 Struktura projektu

Projekt obecnie z powodu swoich niewielkich rozmiarów nie posiada skomplikowanej struktury.

28:9729466662

6.2.2 CORS filter

Klasa ta implementuje interfejs Filter**,** z której nadpisana zostaje metoda doFilter. Metoda ta określa, jaki adres, jakie metody HTTP oraz jakie nagłówki są dopuszczone poprzez dodanie odpowiednich nagłówków do odpowiedzi.

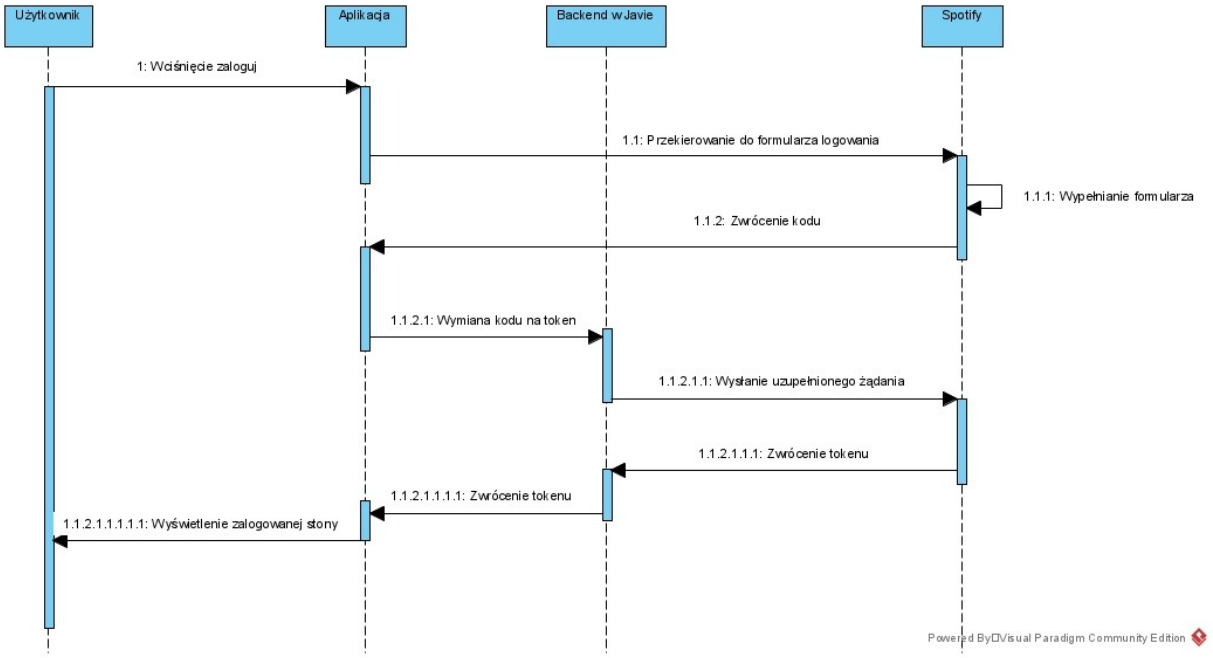
Listing 10: Metoda filtrująca zgodnie z mechanizmem CORS

```
public void doFilter(ServletRequest req, ServletResponse res, FilterChain chain)
        throws IOException, ServletException {
    HttpServletResponse response = (HttpServletResponse) res;
    HttpServletRequest request = (HttpServletRequest) req;
    response.setHeader("Access-Control-Allow-Origin", "*");
    response.setHeader("Access-Control-Allow-Methods", "POST");
    response.setHeader("Access-Control-Max-Age", "3600");
    response.setHeader("Access-Control-Allow-Headers", "Content-Type");
    if ("OPTIONS".equalsIgnoreCase(request.getMethod())) {
        response.setStatus(HttpServletResponse.SC_OK);
    } else {
        chain.doFilter(req, res);
    }
}
```

6.2.2 Usługa do otrzymania tokena

Usługa ta odpowiedzialna jest za dodanie do żądania wysłanego z Angulara w celu otrzymania odpowiedniego klucza. Klucz ten umożliwia autoryzację aplikacji po stronie serwisu. Z tego powodu musi być przed użytkownikiem ukryty. Wykorzystano tutaj bibliotekę OkHttp oraz bibliotekę GSON,

29:5072617385



Rysunek 6: Diagram przepływu logowania

Zawiera on metodę otrzymującą żądania, która tworzy nowe żądanie wysyłane do serwisu Spotify.

Listing 11: Metoda obsługująca żądanie do otrzymania tokena

```
  @PostMapping("/getToken")
    public TokenData getToken(@RequestBody Request request, HttpServletRequest servletRequest)
throws IOException {
        var post = new HttpPost(LINK);
        var origin = servletRequest.getHeader("origin") + "/login/";
        post.setEntity(new UrlEncodedFormEntity(prepareEntity(request.getCode(),origin)));
        try (CloseableHttpClient httpClient = HttpClients.createDefault();
             CloseableHttpResponse response = httpClient.execute(post)) {
            return new Gson().fromJson(EntityUtils.toString(response.getEntity()), TokenData.class);
        }
    }
```

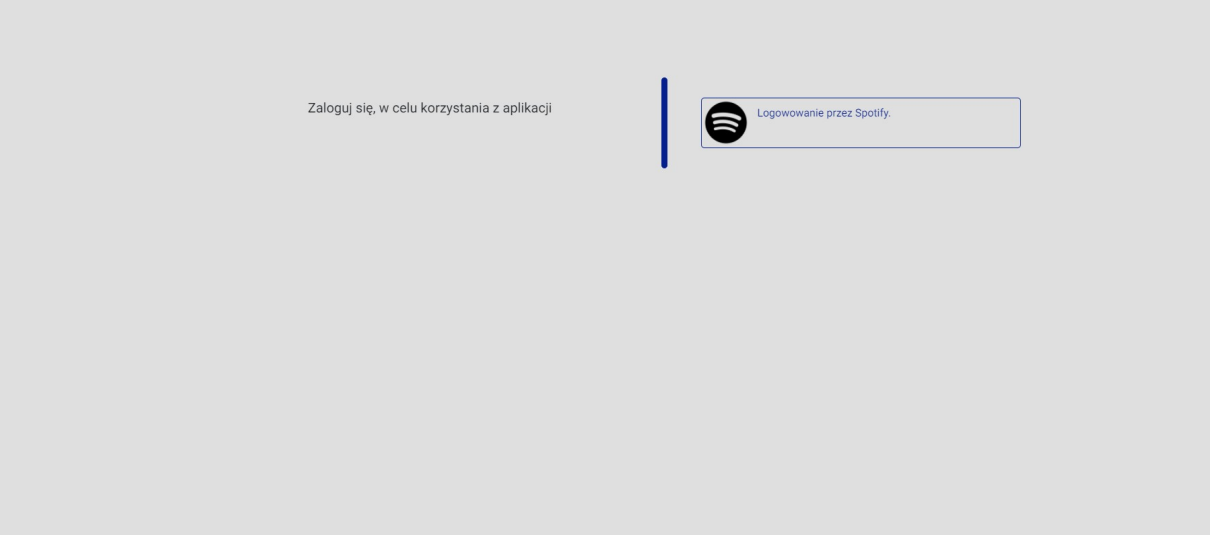

30:3074193282

7. Przykład użycia aplikacji w praktyce

W tym rozdziale zostanie przedstawiona finalna aplikacja będąca tematem tej pracy z punktu widzenia użytkownika.

7.1 Logowanie

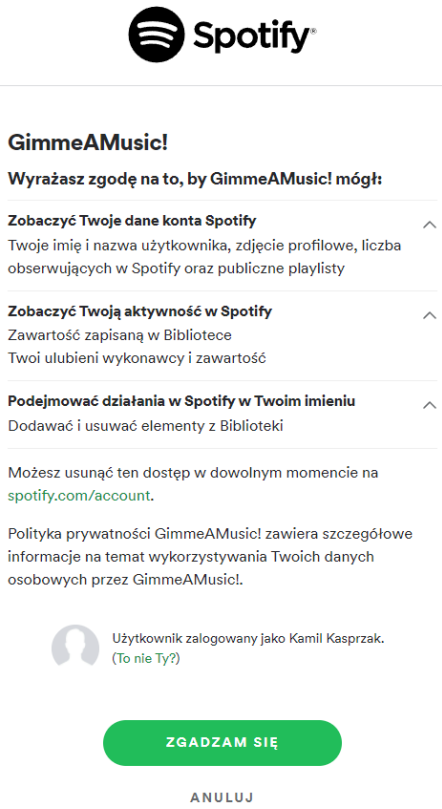
Po uruchomieniu aplikacji użytkownik zobaczy widok z informacją o konieczności zalogowania oraz przycisk z przekierowaniem do logowania w serwisie Spotify. Na urządzeniach z szerszym ekranem elementy te są umieszczone poziomo, a na urządzeniach z węższym ekranem pionowo.



Rysunek 7: Widok dla niezalogowanego użytkownika

Po kliknięciu „logowanie przez Spotify” użytkownik przenosi się do ekranu logowania, który jest już częścią samego serwisu Spotify, a nie utworzonej aplikacji. Spotify zapamiętuje, czy użytkownik logował się już wcześniej, dlatego istnieją dwa różne warianty tej strony. Na tej stronie znajdują się także informacje dotyczące praw, jakie zyska aplikacja do konta użytkownika.

31:1034702576



Rysunek 8: Logowanie do serwisu Spotify

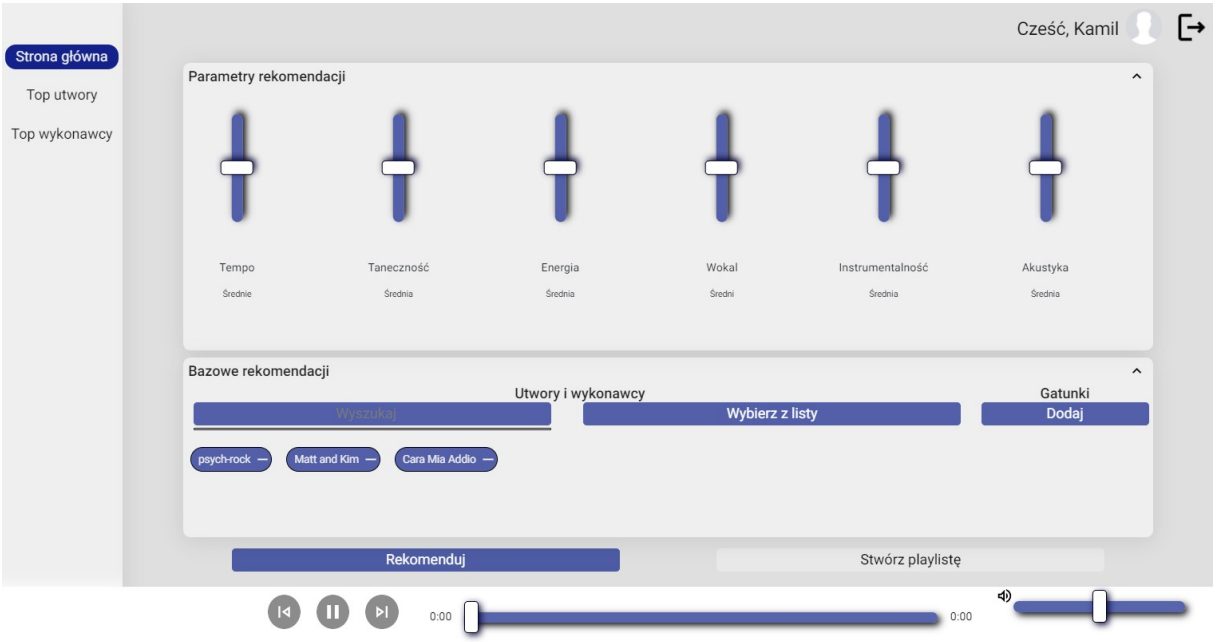
7.2 Rekomendowanie utworów

Po udanym zalogowaniu użytkownik zostanie przekierowany do strony z główną funkcją aplikacji, czyli rekomendowaniem utworów muzycznych. Niezależnie od strony, zalogowany użytkownik zawsze ma dostęp do nawigacji znajdującej się w lewej części aplikacji oraz do panel służącego do sterowania odtwarzaniem muzyki.

Widok do rekomendowania utworów po wejściu na niego składa się z dwóch rozwijanych sekcji. Pierwsza zawiera suwaki dotyczące ustalenia zbliżonych parametrów utworów. Przykładowe parametry to tempo, czy akustyka utworu. Druga sekcja służy do wybrania elementów, na których bazować będzie rekomendacja. Użytkownik może do nich dodać maksymalnie pięć elementów, wybierając z wykonawców, utwory oraz gatunki.

Pod sekcjami znajdują się dwa przycisku. Pierwszy „rekomenduj” szuka dla użytkownika utworów na podstawie wybranych opcji, drugi przycisk tworzy z wyników playlistę, która zostaje udostępniona użytkownikowi w aplikacji Spotify.

32:4831670076

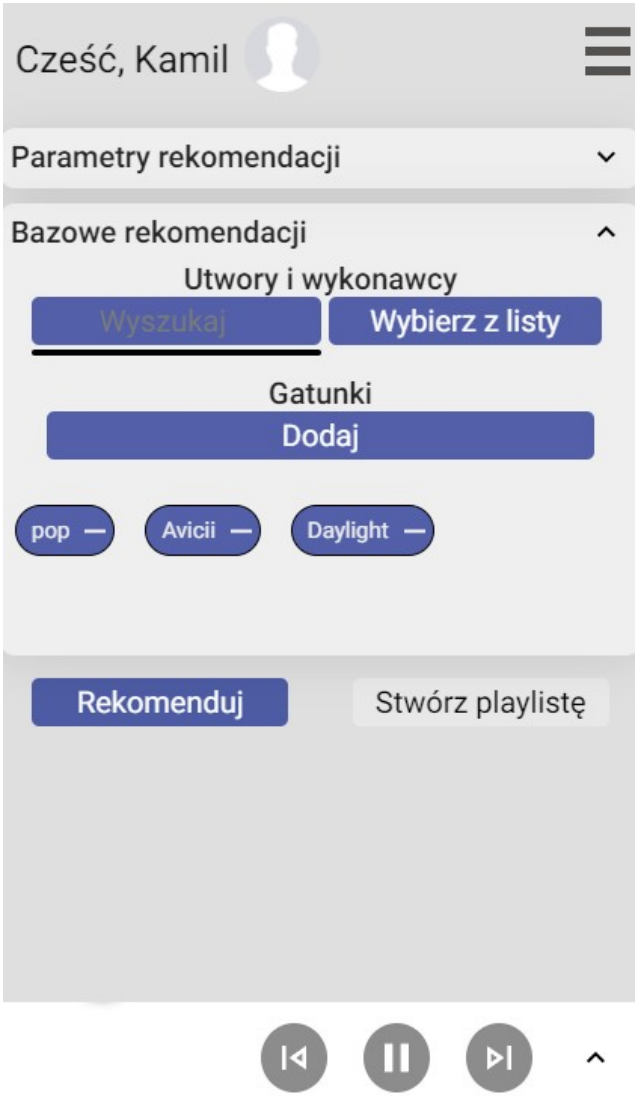


Rysunek 9: Widok po zalogowaniu

W bazowych rekomendacji użytkownik może dodać dowolnych wykonawców i utwory, wybierając ich z listy, przygotowanej na podstawie najczęściej słuchanych utworów i wykonawców, lub samodzielnie te elementy wyszukać. Gatunki użytkownik może dodać tylko z dostępnej listy, z funkcją filtrowania elementów.

Interfejs został dostosowany do urządzeń z węższym wyświetlaczem. Na węższych ekranach suwaki wyświetlone są w trzech lub w dwóch kolumnach w zależności od szerokości ekranu. Przyciski w bazowych rekomendacji, zostały umieszczone pionowo.

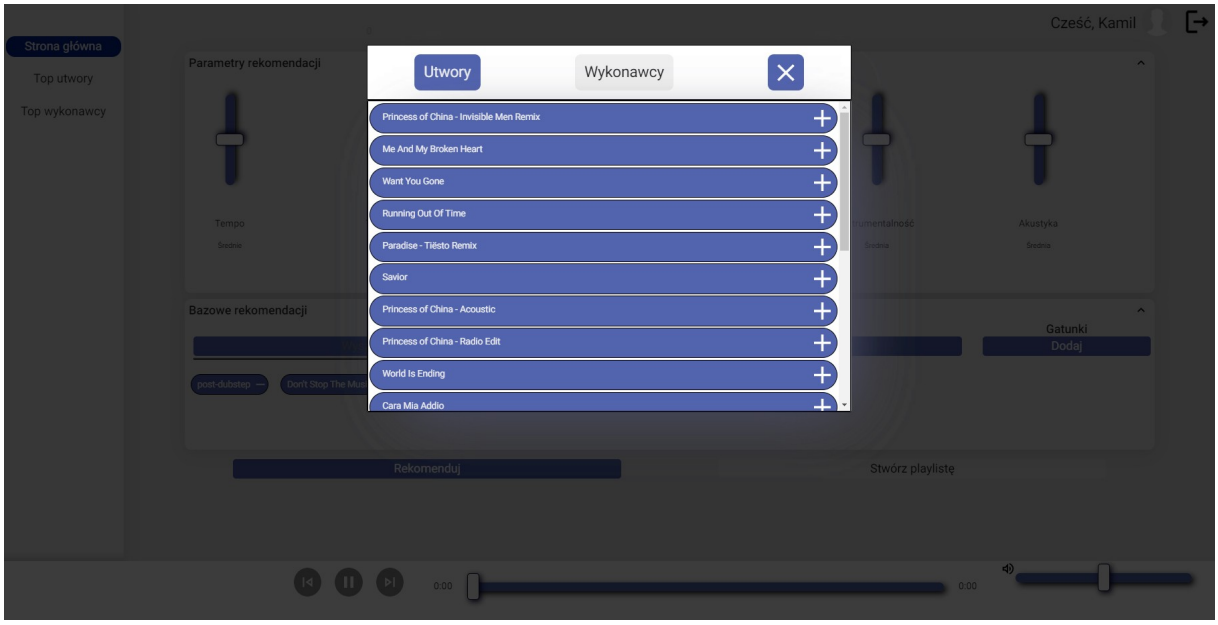
33:1070304208



Rysunek 10: Widok po zalogowaniu na urządzeniu mobilnym

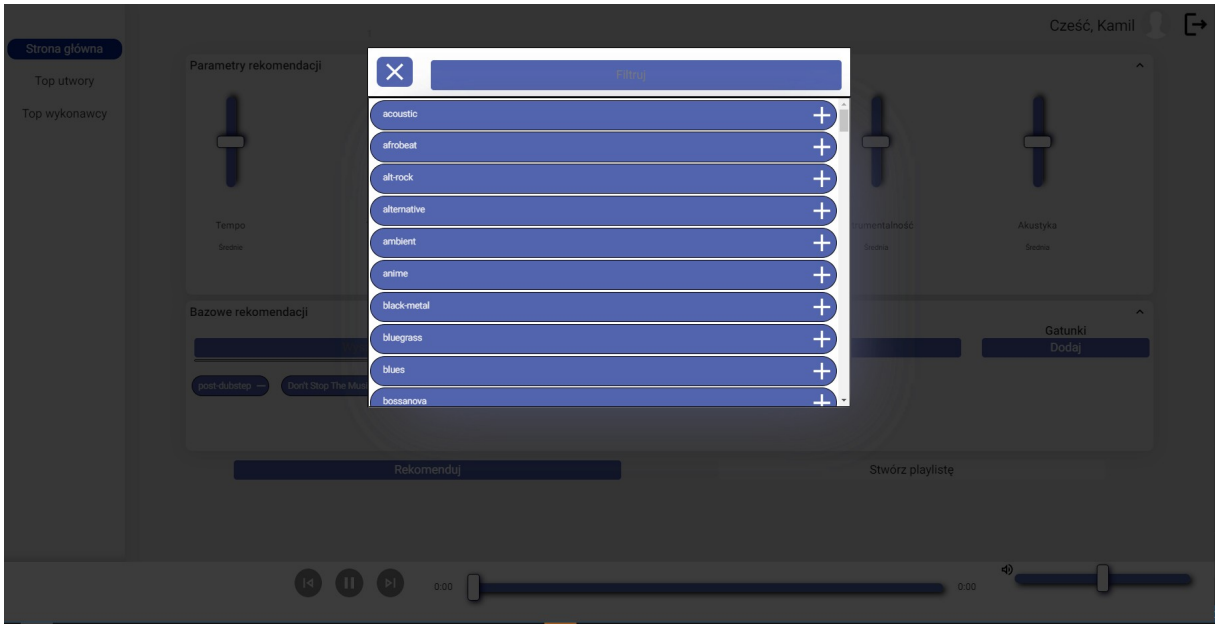
Po kliknięciu przycisku „wybierz z listy” zostanie użytkownikowi wyświetlone okno modalne, gdzie ma on możliwość wybrania elementów z wcześniej omawianej listy. Ma on do wybory listę z utworami bądź wykonawcami. Nowe elementy dodaje się, klikając na dostępne elementy.

34:5886629004



Rysunek 11: Dodanie utwór lub wykonawców

Użytkownik zobaczy także podobne okno, po kliknięciu przycisku „dodaj” znajdującego się pod napisem „gatunki”. Użytkownikowi wyświetli się lista z możliwymi do wyboru gatunkami. Ze względu na liczne możliwe do wyboru gatunki, użytkownik może odfiltrować gatunki, w celu łatwiejszego znalezienia szukanego gatunku muzycznego.

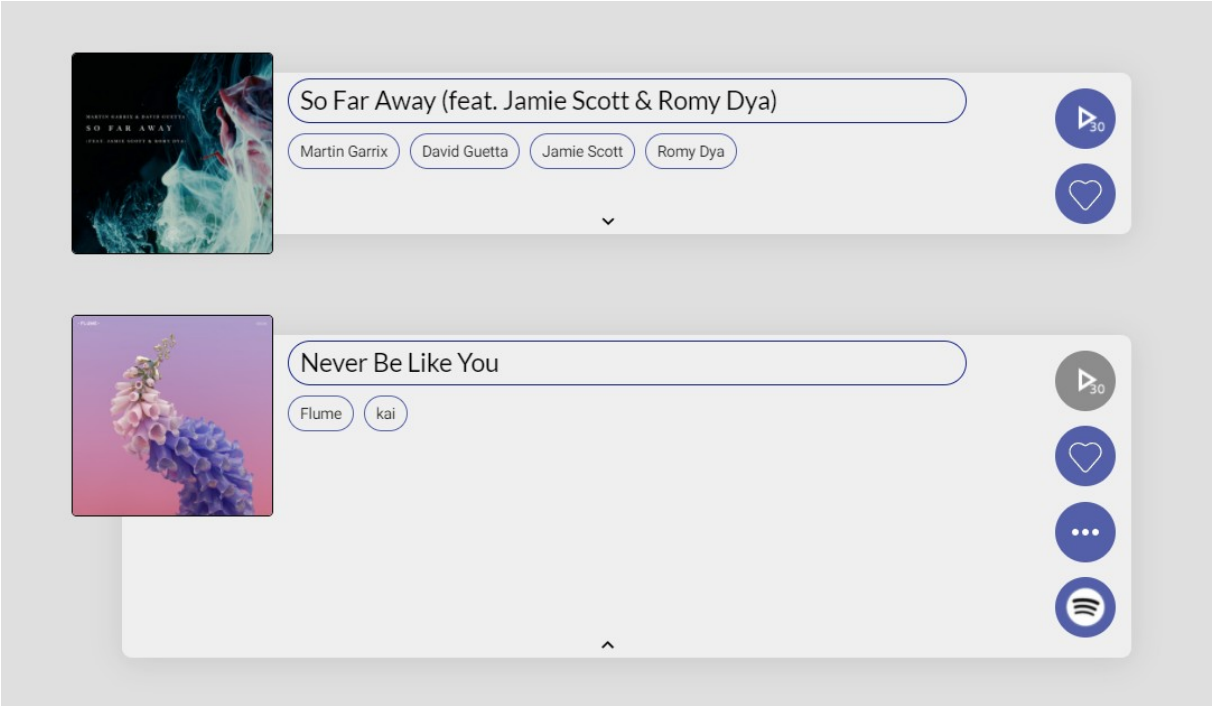


Rysunek 12: Dodanie gatunku

35:4992291410

7.3 Wyniki rekomendacji

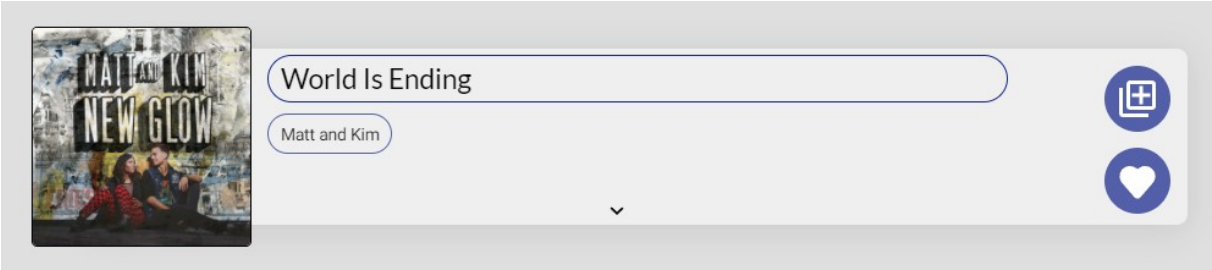
Po znalezieniu utworów dla użytkownika wyniki zostają mu przedstawione pod interfejsem, który zostanie automatycznie zwinięty. Każdy utwór zostaje umieszczony w karcie, na którą składa się zdjęcie albumu, z jakiego pochodzi utwór, tytułu utworu, jego wykonawcę, bądź wykonawców. W karcie znajdują się także dwa przyciski. Po rozwinięciu utworu ich liczba zwiększa się do czterech.



Rysunek 13: Karty z utworami dla użytkownika bez konta premium

Pierwszy przycisk jest zależny od typu konta użytkownika i czy udostępnił on strumieniowanie muzyki do aplikacji. Dla użytkownika bez konta premium możliwe jest tylko wysłuchanie 30 sekundowego fragmentu utworu. Użytkownik z kontem premium może dodać utwór do kolejki odtwarzania. Udane dodanie do kolejki zostanie potwierdzone odpowiednim komunikatem.

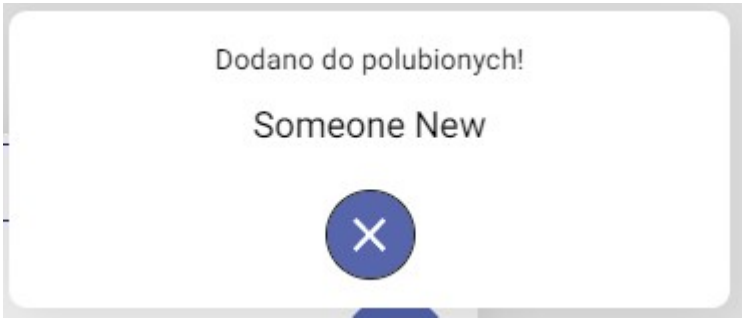
36:3432033577



Rysunek 14: Karta z utworem dla użytkownika premium

Rysunek 15: Powiadomienie o daniu utworu do kolejki

Drugi przycisk służy do polubienia utworu w serwisie Spotify, lub jeśli został on już polubiony, to do jego usunięcia. Tak jak w przypadku dodania do kolejki, użytkownikowi zostanie pokazane powiadomienie o udanym dodaniu.

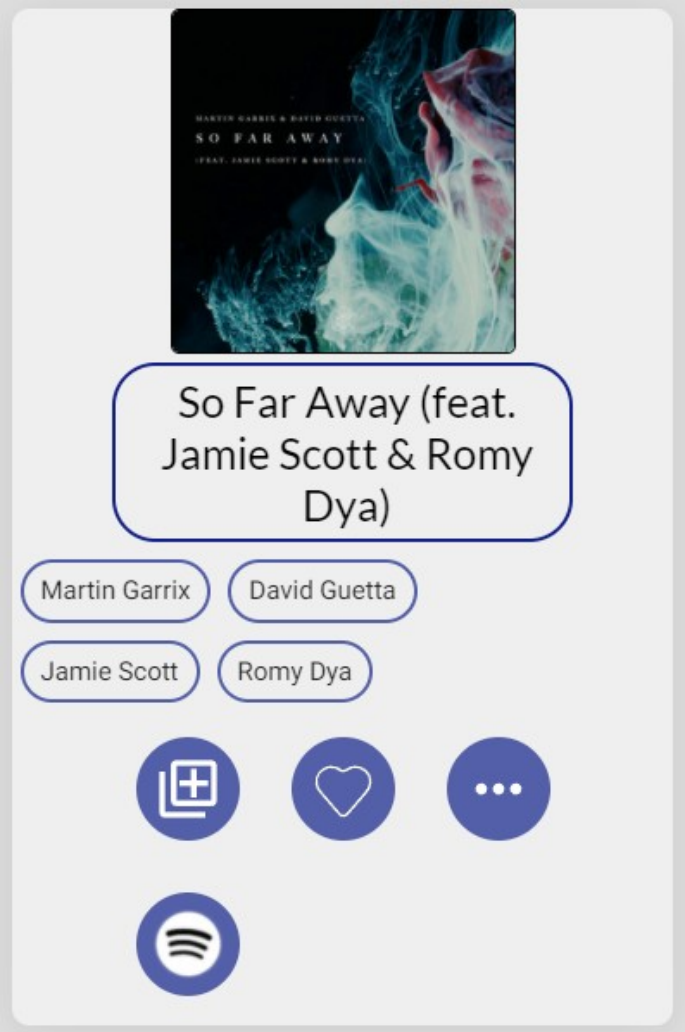


Rysunek 16: Powiadomienie o polubieniu utworu

Trzeci przycisk przekierowuje użytkownika do strony w aplikacji z większymi szczegółami dotyczącym wykonawców danego utworu. Czwarty przycisk przekierowuje do strony w serwisie Spotify z danym utworem.

Karty z utworami zostały dostosowane do urządzeń z węższym ekranem. Elementy budujące kartę zostały umieszczone w tym przypadku pionowo.

37:8194186359



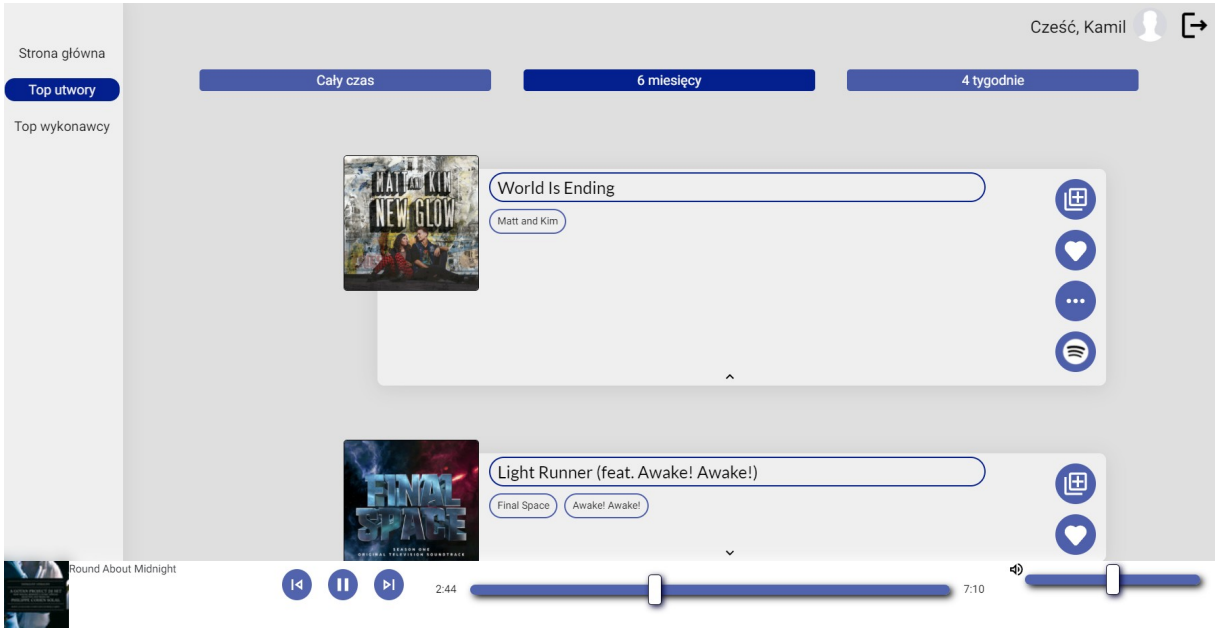
Rysunek 17: Wynik rekomendacji na urządzeniu z węższym ekranem

7.4 Najczęściej słuchane utwory i wykonawcy użytkownika

Użytkownik może zobaczyć swoje najczęściej słuchane utwory, wybierając opcje „top utwory” z panelu nawigacji. Ma on też możliwość zobaczenia najczęściej słuchanych przez siebie wykonawców przez wybranie „top wykonawcy”.

Na stronie z najczęściej słuchanymi utworami użytkownik ma możliwość wybrania przedziału czasowe z trzech możliwych. Wybór odbywa się za pomocą przycisków w górnej części strony. Wyniku są przedstawione w kartach analogicznych do wyników z rekomendacji.

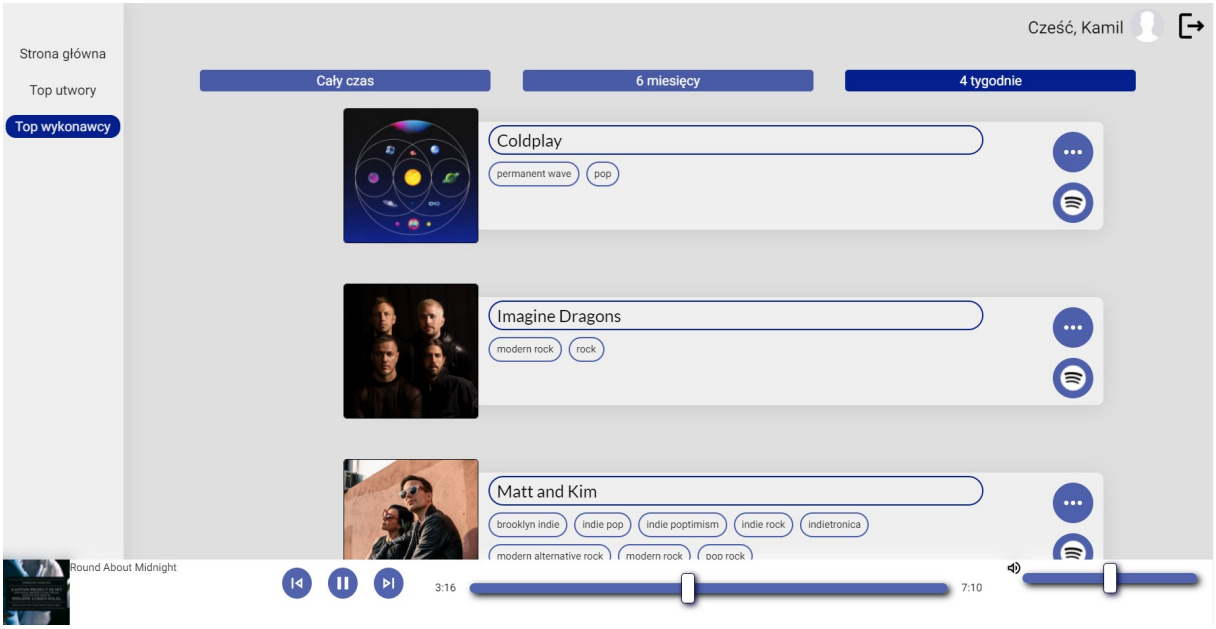
38:7674504721



Rysunek 18: Widok z najczęściej słuchanymi utworami

Bardzo podobnie wygląda widok z najczęściej słuchanych wykonawcami. Jedyną różnicą jest zmiana w wyglądzie kart. W kartach znajduje się zdjęcie danego wykonawcy, jego nazwa, oraz jego gatunki muzyczne. Ich wygląd jest identyczny do kart z utworami. Znajdują się tu dwa przyciski. Pierwszy z przekierowaniem do strony wykonawcy w aplikacji. Drugi z przekierowaniem do serwisu Spotify ze stroną od wykonawcy.

39:1495016784



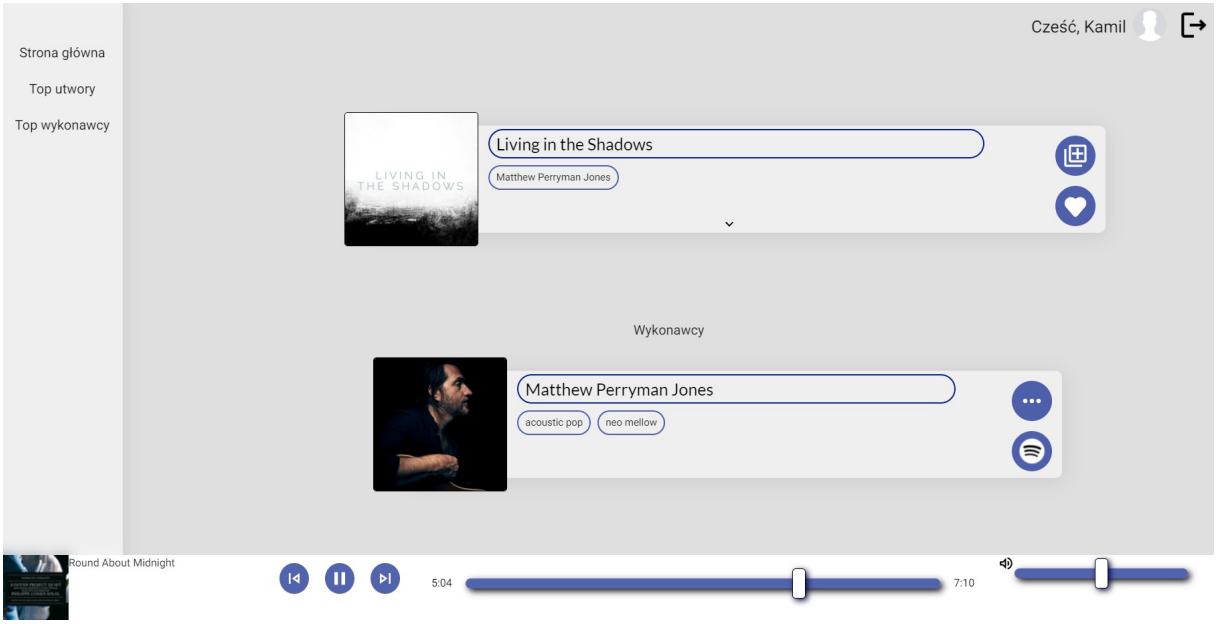
Rysunek 19: Widok z najczęściej słuchanymi wykonawcami

7.5 Dodatkowe informacje o utworach i wykonawcach

Dodatkową funkcją aplikacji jest możliwość zobaczenia dodatkowych szczegółów utworów i wykonawców z wykorzystaniem kart tych drugich.

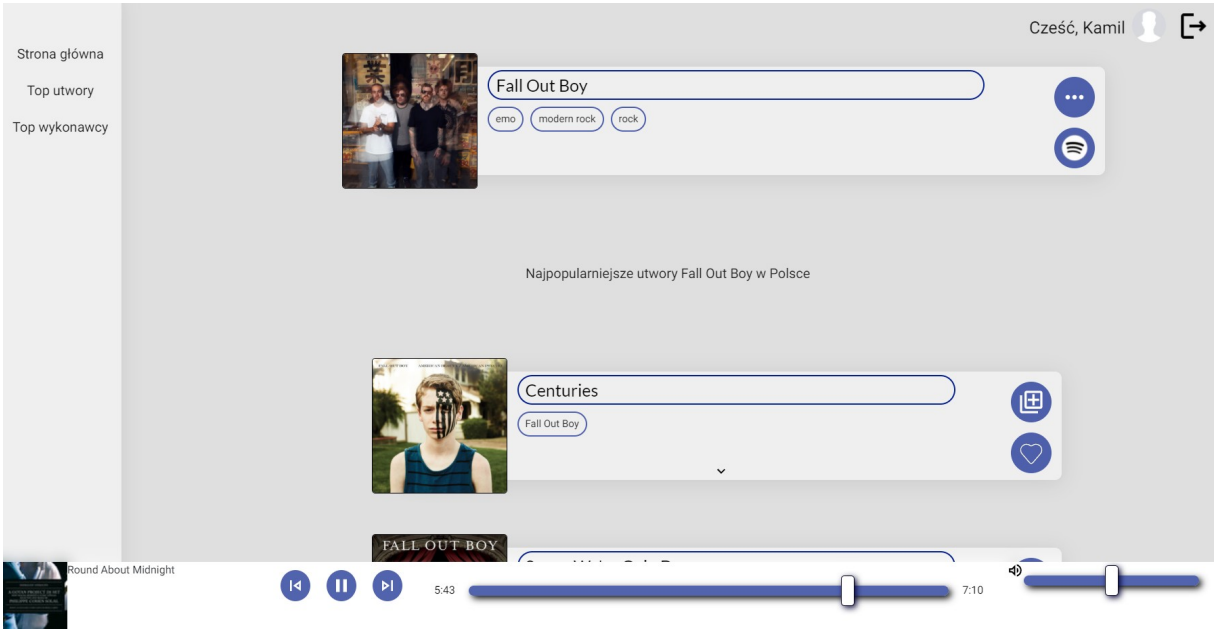
Na dodatkowym widoku dla utworów znajduje się karta z danym utworem oraz ponownie zostaje wykorzystana karta bądź karty z wykonawcą/wykonawcami danego utworu. Są to te same komponenty znajdujące się we wcześniej omawianych widokach.

40:2960153784



Rysunek 20: Dodatkowy widok dla utworu

Dodatkowy widok posiadają także wykonawcy utworów. Na tym widoku znajduje się karta z wykonawcą danego utworu oraz karty z najpopularniejszymi dla niego utworami w Polsce.



Rysunek 21: Dodatkowy widok dla wykonawcy

41:5806165195

7.6 Sterowanie odtwarzaniem muzyki

Niezależnie od widoku aplikacji znajduje się panel do sterowania muzyką. Od lewej strony zawiera on zdjęcie aktualnie odtwarzanego utworu wraz z jego nazwą. Następnie znajdują się trzy przyciski. Pierwszy odtwarza poprzedni utwór, drugi wstrzymuje lub wznawia go, trzeci odtwarza następny utwór z kolejki. Przy braku strumieniowaniu muzyki z aplikacji Spotify pierwszy i trzeci przycisk jest niedostępny. Następnie znajduje się suwak z czasem aktualnie odtwarzanego utworu. Użytkownik ma możliwość przewinięcia utworu. Na końcu znajduje się suwak do zmiany głośności utworu. Na urządzeniach mobilnych suwaki pojawiają się po rozwinięciu panelu nad pozostałymi utworami.

Rysunek 22: Panel do sterowania muzyką

42:1010805322

8. Testy i weryfikacja aplikacji

Ważnym elementem podczas tworzenia każdego systemu jest zadbanie o napisanie testów. Odpowiednie zadbanie o testy potrafi przyśpieszyć wykrywanie błędów powstałych podczas zmiany już istniejącego kodu. We frameworku Angular zostały wbudowane testy jednostkowe. Testy w Angularze bazują na frameworku Jasmine. W tym rozdziale zostanie pokazane kilka przykładowych testów napisanych w tytułowej aplikacji.

Pierwszy test sprawdza, czy dla niezalogowanego użytkownika zostanie wyświetlony odpowiedni komponent.

Listing 12: Test sprawdzający czy wygeneruje odpowiedni komponent

```
it('should render app-home-not-logger', () => {
  component.isLogin = false;
  let debug = fixture.debugElement.nativeElement;
  fixture.detectChanges();
  expect(debug.querySelector('app-home-not-logged')).toBeTruthy();
})
```

Test sprawdzający, czy zostanie wyświetlone odpowiednie imię użytkownika.

Listing 13: Test sprawdzający poprawność wyświetlania imienia użytkownika

```
it('should get name', async () => {
  let userService = fixture.debugElement.injector.get(UserService);
  spyOn(userService, 'getUserInfo').and.returnValue(
    of({ display_name: 'John Doe', images: [], id: '0', product: 'product' })
  );
  fixture.detectChanges();
  component.ngOnInit();
  fixture.whenRenderingDone().then(() => {
    expect(component.user.display_name).toEqual('John Doe');
  });
});
```

Kolejny test sprawdza, czy jedna z sekcji w interfejsie do rekomendacji zawiera właściwy opis.

43:4659428676

Listing 14: Test sprawdzający poprawną treść nagłówka

```
it('has right desctiption', () => {
  fixture.detectChanges();
  const compiled = fixture.nativeElement;
  expect(compiled.querySelector('h1').textContent).toEqual('Bazowe rekomendacji');
});
```

Następny test sprawdza, czy tablica z elementami dla panelu nawigacyjnego zawiera trzy elementy.

Listing 15: Test sprawdzający poprawną liczbę elementów w panelu nawigacyjnym

```
it('should has 3 elements', () => {
  expect(component.links.length).toBe(3);
});
```

Kolejny test sprawdza czy ekran logowania zawiera panel, który po kliknięciu przekierowuje do strony logowania.

Listing 16: Test sprawdzający poprawność widoku logowania

```
it('has div with redirect to Spotify login website',() =>{
  expect(fixture.nativeElement.querySelector('.spotify')).toBeTruthy();
})
```

Kolejne dwa testy sprawdzają poprawność przeliczanie czasu z milisekund na minuty i sekundy w odtwarzaczu muzycznym.

Listing 17: Test sprawdzający poprawność liczenia czasu

```
it('should count seconds correct', () => {
  let result = component.countSecondsSpotify(100000);
  expect(result).toEqual('40');
});

it('should count minutes correct', () => {
  let result = component.countMinutesSpotify(100000);
  expect(result).toEqual('1');
});
```


44:2873433853

Następny test sprawdza poprawność działania metody do resetu czasu odtwarzacza.

Listing 18: Test sprawdzający poprawność ponownego odtwarzania

```
it('replay should set time to 0', () => {
  service.audio.currentTime = 1;
  service.replay();
  expect(service.audio.currentTime).toBe(0);
 });
```

Ostatni test pochodzi z serwerowej części aplikacji, który sprawdza poprawność tworzonego żądania wysyłane do serwisu Spotify.

Listing 19: Test sprawdzający poprawność tworzonego żądania post

```
@Test
  void shouldHasPropertyFromApplicationProperties(){
     when(environment.getProperty(any())).thenReturn(RESULT);
     var result = tokenController.prepareEntity(CODE,ORIGIN);
     assertAll(
           () -> assertEquals("client_id", result.get(0).getName()),
           () -> assertEquals(RESULT, result.get(0).getValue()),
           () -> assertEquals("client_secret", result.get(1).getName()),
           () -> assertEquals(RESULT, result.get(1).getValue()),
           () -> assertEquals("code", result.get(2).getName()),
           () -> assertEquals(CODE, result.get(2).getValue()),
           () -> assertEquals("grant_type", result.get(3).getName()),
           () -> assertEquals(RESULT, result.get(3).getValue()),
           () -> assertEquals("redirect_uri", result.get(4).getName()),
           () -> assertEquals(ORIGIN, result.get(4).getValue())
     );
  }
```


45:5148612930

Podsumowanie

Celem pracy było stworzenie aplikacji, której głównym celem będzie rekomendowanie użytkownikowi utworów muzycznych za pomocą Spotify API. Zdecydowano, że będzie to aplikacja webowa do której do stworzenia został wybrany jeden z najbardziej popularnych obecnie frameworków – Angular. Framework ten jest mocno rozbudowanym narzędziem, które elementy w wygodny sposób umożliwiły realizację wielu kluczowych elementów, przykładowo kwestie związane z bezpieczeństwem, dzięki czemu napisany kod jest czytelniejszy i łatwiejszy w przyszłym modyfikowaniu. Framework zapewnił wiele własnych rozwiązań ułatwiających stworzenie aplikacji, przykładowo bibliotekę do wywoływania żądań HTTP. Dodatkowo, z powodu konieczności ukrycia pewnych danych przed użytkownikiem, które potrzebne są do autoryzacji aplikacji, stworzono dodatkowo mniejszy projekt w Java Spring Boot. Framework ten umożliwił wygodne stworzenie prostego serwera, który obecnie służy tylko do wspomagania procesu logowania użytkownika. Projektując wygląd aplikacji starano się stworzyć czytelny i interesujący dla użytkownika interfejs. Stworzenie funkcjonalnego interfejsu nie jest prostym zadaniem, i z tego powodu podczas tworzenia aplikacji dochodziło do zmian w jego wyglądzie.

Głównym celem aplikacji jest rekomendacja muzyki, co zostało zaimplementowane w niej. Użytkownik do tego celu posiada interfejs, dzięki któremu ma on wpływ na rekomendowane mu utwory. Wybiera on zbliżone parametry utworu oraz elementy, które wykorzystane będą do wybrania rekomendowanych utworów. Dodatkowo użytkownik ma podgląd na utwory, które najczęściej słuchał. Zarówno w widoku najczęściej słuchanych utworów oraz w rekomendacjach, użytkownik może zobaczyć podstawowe informacje o każdym utworze na liście, dodać go do polubionych czy wysłuchać jego fragmentu lub cały utwór, jeśli w aplikacji Spotify wybierze on strumieniowanie muzyki do tytułowej aplikacji. W dolnej części strony znajduje się panel do sterowania muzyką, gdzie użytkownik może przykładowo przewinąć utwór lub go wstrzymać.

Spotify API zapewnia wiele możliwości, dzięki czemu w przyszłości można rozbudować tę aplikację o kolejne funkcje. Przykładowo można by zaimplementować możliwość tworzenia playlist, wybierając poszczególne utwory lub rozbudować znajdujący się w aplikacji panel do sterowania muzyką, między innymi o możliwość powtarzania utworów. Dzięki Angularowi takie rozbudowanie aplikacji będzie znacznie łatwiejsze. Część serwerowa

46:6838691614

w Spring Boot obecnie służy tylko do logowania użytkownika, jednak w przyszłości można wykorzystać inne, nieomawiane w tej pracy możliwości tego frameworka. Serwis Spotify nie jest jedynym serwisem muzycznym udostępniającym rozbudowane API, w przyszłości aplikację można także rozbudować o inne serwisy. Przykładowym takim serwisem jest Apple music.

47:5881755407

Bibliografia

[1] Badanie dotyczące popularności aplikacji webowych [2022-01]

https://www.webchefs.pl/blog/rosnaca-popularnosc-aplikacji-webowych-w-polsce/

[2] Najpopularniejsze serwisy muzyczne [2022-01]

https://www.komputerswiat.pl/aktualnosci/aplikacje/muzyczne-serwisy-streamingowe-zyskuja-popularnosc-spotify-zdecydowanym-liderem/5bv09rg

[3] Najpopularniejsze frameworki [2022-01]

https://impicode.pl/blog/najpopularniejsze-frameworki-javascript/

[4] Aplikacje webowe [2022-01]

https://kissdigital.com/pl/blog/czym-sa-aplikacje-webowe-aplikacje-internetowe

[5] Popularność Spotify [2022-01]

https://www.businesswire.com/news/home/20190429005261/en/Spotify-Technology-S.A.-Announces-Financial-Results-Quarter

[6] Historia frameworka Angular [2022-01]

https://geek.justjoin.it/historia-angular?gclid=CjwKCAiA0KmPBhBqEiwAJqKK4yloidECxK15rvH09zeRx9Cvds3wSCPxLzbnzJjymLfVsvzP44r5vhoCB1EQAvD_BwE#Angular_12_-_najnowsza_wersja_Angulara

[7] Informacje dotyczące JavaScript

JavaScript. Przewodnik. Poznaj język mistrzów programowania. Wydanie VII

Autor: David Flanagan

[8] Dokumentacja Angulara [2022-01]

https://angular.io/docs

[9] Protokół http [2022-01]

https://sekurak.pl/protokol-http-podstawy/

[10] Informacje dotyczące Javy

Java. Podstawy Wydanie XI

Autor: Horstmann Cay S.

[11] Dokumentacja Java Spring [2022-01]

https://docs.spring.io/spring-framework/docs/current/reference/html/

[12] Popularność języka Java [2022-01]

https://nofluffjobs.com/blog/backend-najpopularniejsze-jezyki-programowania-w-2019-roku-infografika/

48:7608412465

Spis listingów

49:1259748067

Wykaz rysunków

50:1332797585