Politechnika
Śląska

PROJEKT INŻYNIERSKI

System do współtworzenia list utworów muzycznych.

Mateusz CUDZIK
Nr albumu: 295620

Kierunek: Informatyka
Specjalność: Bazy danych i inżynieria systemów

PROWADZĄCY PRACĘ
dr. inż. Daniel Kostrzewa
KATEDRA Informatyki Stosowanej
Wydział Automatyki, Elektroniki i Informatyki

Gliwice 2024

Tytuł pracy

System do współtworzenia list utworów muzycznych.

Streszczenie

Praca stanowi analizę, projekt oraz implementację systemu umożliwiającego wspólne zarządzanie listą odtwarzania utworów i sesjami odsłuchowymi podczas wydarzeń towarzyskich. Praca koncentruje się na stworzeniu interaktywnego środowiska, które umożliwi użytkownikom synchroniczne wyszukiwanie oraz dodawanie utworów w czasie rzeczywistym do wspólnej kolejki odtwarzania, jednocześnie zapewniając organizatorowi wydarzenia pełną kontrolę.

Słowa kluczowe

aplikacja internetowa, Spring, muzyka

Thesis title

Co-creation of song lists application

Abstract

The thesis presents an analysis, design, and implementation of a system enabling collaborative management of a playlist and listening sessions during social events. It focuses on creating an interactive environment that allows users to synchronously search and add tracks in real-time to a shared playback queue, while granting the event organizer full control.

Key words

web application, Spring, music

Spis treści

Rozdział 1

Wstęp

1.1 Wprowadzenie w problem/zagadnienie

Muzyka od zawsze towarzyszyła człowiekowi. Jest obecna na każdym spotkaniu towarzyskim czy wydarzeniu grupowym i odgrywa kluczową rolę w tworzeniu panującej atmosfery. Wspólne słuchanie muzyki potrafi integrować grupę i pomaga budować relacje.

Problematyczny może się okazać dobór odpowiedniej muzyki, która powinna angażować wszystkich uczestników i w miarę możliwości odpowiadać preferencjom jak największej liczby osób. Aplikacja, która ma za zadanie umożliwić każdemu uczestnikowi dodawania własnych ulubionych utworów do wspólnej listy, może być odpowiedzią na te wyzwania.

1.2 Osadzenie problemu w dziedzinie

Obecnie jednym z powszechnie wykorzystywanych sposobów odtwarzania muzyki są platformy do strumieniowania treści takie jak, Spotify [9]. Ręczne sterowanie odtwarzaniem lub wybór utworów przez pojedyncze osoby może nie zawsze spełnić oczekiwania wszystkich uczestników. Brak systemów, które efektywnie łączą preferencje wszystkich użytkowników oraz umożliwiają równoczesne i sprawiedliwe dodawanie utworów do wspólnej listy odtwarzania jest głównym powodem powstania niniejszej pracy.

1.3 Cel pracy

Celem tej pracy jest stworzenie aplikacji, która umożliwi spersonalizowanie zarządzania odtwarzanej muzyki podczas spotkań towarzyskich w oparciu o integrację z serwisem Spotify. System ma dać uczestnikom wydarzenia możliwość równoczesnego dodawania utworów do wspólnej kolejki odtwarzania oraz intuicyjny i wygodny interfejs do interakcji z serwisem Spotify, aby zapewnić m.in. prosty sposób przeglądania dostępnych

w bazie utworów. Aplikacja umożliwi administrację sesją, pozwalając przykładowo na ustawienie blokady na treści dla dorosłych oraz kontrolę liczby uczestników. System nie wymaga rejestracji czy logowania od użytkowników, którzy do sesji dołączają, jedynym wymogiem jest znajomość kodu dostępu lub zeskanowanie kodu QR udostępnionego przez właściciela sesji. Wszystkie dodawane utwory w sesji dodawane są do wspólnej listy utworów dostępnej publicznie na koncie Spotify właściciela sesji, jeśli ta opcja nie została wyłączona.

1.4 Zakres pracy oraz określenie wkładu autora

1.4.1 Baza danych

Zaprojektowanie oraz implementacja bazy danych, dostosowanej do potrzeb aplikacji, przechowującej informacje o sesjach, gościach oraz właścicielach sesji. Ustalenie relacji między encjami w celu zapewnienia optymalnej struktury danych.

1.4.2 Serwer aplikacji

Zaprojektowanie struktury aplikacji oraz analiza wymagań. Wdrożenie logiki, na którą składa się obsługa sesji i użytkowników, komunikacja z zewnętrznym API serwisu Spotify oraz implementacja interfejsów RESTful API umożliwiających komunikację z aplikacją klienta. Zapewnienie bezpieczeństwa przez realizację mechanizmów uwierzytelnienia oraz autoryzacji dostępu do zasobów.

1.4.3 Interfejs użytkownika

Zdefiniowanie struktury aplikacji - wyznaczenie komponentów odpowiedzialnych za różne funkcje (logowanie, zarządzanie sesją), zaplanowanie klarownej i intuicyjnej nawigacji pomiędzy widokami. Stworzenie szablonów HTML i ich stylizacja, aby poprawić interaktywność z użytkownikiem przez np. animacje sugerujące ładowanie danych. Implementacja komponentów do obsługi logiki biznesowej takich jak przyciski, panel wyszukiwania utworów czy formularz z walidacją danych.

1.4.4 Komunikacja między klientem a serwerem

Zdefiniowanie interfejsu API, określenie ścieżek URL, parametrów i metod HTTP dla poszczególnych operacji zgodnie z zasadami architektury REST. Implementacja logiki po stronie klienta odpowiedzialnej za wywoływanie odpowiednich żądań do określonych punktów końcowych serwera aplikacji oraz przetworzenie danych otrzymanych w odpowiedzi poprzez np. aktualizację interfejsu użytkownika o odpowiednio

sformatowane wyniki zapytania. Zapewnienie obsługi błędów (brak odpowiedzi ze strony serwera, niekompletne zapytanie, wewnętrzny błąd serwera itp.) oraz walidacji danych zarówno po stronie serwera jak i aplikacji klienta.

1.4.5 Dokumentacja techniczna

Sporządzenie dokumentacji opisującej obiekty uczestniczące w komunikacji między serwerem a klientem oraz punkty końcowe serwera - przyjmowane parametry, ścieżki, zawartość zapytania, sposób autoryzacji oraz zwracane wartości.

1.5 Zwięzła charakterystyka rozdziałów

Rozdział 1 zawiera wprowadzenie w zagadnienie oraz określenie dziedziny i zakresu pracy. Rozdział 2 to wstęp teoretyczny i porównanie istniejących rozwiązań dostępnych obecnie na rynku. W rozdziale 3 zostały określone wymagania, które system powinien spełnić oraz przypadki użycia aplikacji wraz z opisem użytych do implementacji narzędzi. Następnie w rozdziale 4 zawarto wyjaśnienie działania aplikacji od strony użytkowej, sposób instalacji oraz konfiguracji systemu. Dodatkowo przedstawione zostały przykłady oraz scenariusze korzystania z aplikacji. Kolejny rozdział skupia się na przedstawieniu sposobu działania aplikacji ze strony technicznej, przez omówienie architektury, bazy danych i implementacji systemu. Rozdział 6 to omówienie metod i wyników weryfikacji oraz walidacji systemu. Finalnie w rozdziale 7 przedstawione zostało podsumowanie pracy oraz zdefiniowanie przyszłych kierunków rozwoju systemu.

Rozdział 2

Analiza tematu

2.1 Sformułowanie problemu

Brak skutecznego narzędzia do zarządzania listą odtwarzania w czasie rzeczywistym podczas wydarzeń jest problematyczne dla organizatorów. Głównym wyzwaniem jest stworzenie aplikacji, która umożliwi wszystkim uczestnikom wydarzenia czynny udział w tworzeniu wspólnej atmosfery oraz płynne odtwarzanie muzyki zgodnie z ich preferencjami, jednocześnie zapewniając pełną kontrolę organizatorowi.

2.2 Osadzenie tematu w kontekście aktualnego stanu wiedzy o poruszanym problemie

Definicja 1. *Aplikację internetową możemy zdefiniować jako program komputerowy uruchamiany w przeglądarce internetowej, komunikujący się z serwerem udostępniającym usługi i zasoby za pomocą sieci internetowej. [6]*

Aplikacja internetowa rożni się od strony internetowej tym, że kładzie nacisk na interaktywność z użytkownikiem, a strona ma na celu przekazywanie statycznych informacji.

Jedną z częściej stosowanych strategii, planując architekturę takiego systemu jest zastosowanie architektury wielowarstwowej. W tym przypadku zastosowany został model trójwarstwowy, który dzieli system na oddzielne byty:

- warstwa prezentacji - najwyższa warstwa odpowiedzialna za interakcję z użytkownikiem, wyświetla informacje i komunikuje się z niższą warstwą.

- warstwa usług - implementuje logikę biznesową aplikacji, komunikuje się z bazą danych, przetwarza otrzymane dane oraz zwraca klientowi wyniki.

- warstwa danych - przechowuje oraz pozyskuje dane potrzebne do poprawnego funkcjonowania aplikacji, komunikuje się z warstwą usług przez zapytania w języku SQL.

Realizacja takich systemów wymaga infrastruktury w tym jednego lub kilku serwerów, które będą dostarczały funkcjonalność użytkownikom. Zaletą takiego rozwiązania jest wysoka skalowalność, ponieważ każda z warstw to osobna aplikacja mogąca funkcjonować na osobnych serwerach co może też pozytywnie wpłynąć na wydajność systemu. Dodatkowym atutem jest także uniezależnienie od technologii, w której dana warstwa została zaimplementowana.

Komunikacja między warstwą prezentacji a warstwą usług określona jest przez protokół HTTP, który charakteryzuje się tym, że klient wysyła żądania do serwera, który udziela na nie odpowiedzi. Format żądań i odpowiedzi jest ściśle zdefiniowany:

- Sekcja nagłówków - zawiera dodatkowe informacje o żądaniu, takie jak metodę autoryzacji, czy oczekiwany format odpowiedzi itp.
- Rodzaj metody - informuje serwer o rodzaju działania, które klient chce podjąć, przykładowo DELETE usunie dane a POST je wyśle.
- Adres URL - określa adres zasobu, który klient chce pozyskać.
- Kody odpowiedzi - serwer wysyłając odpowiedź dodaje także kod informujący o stanie wyniku żądania, kod 200 oznacza, że operacja zakończyła się pomyślnie a kod 500 informuje o wewnętrznym błędzie serwera.
- Bezstanowość - oznacza to, że realizacja każdego żądania nie wpływa na wynik kolejnego oraz są one realizowane niezależnie od siebie.
- Treść - najczęściej jest to obiekt typu JSON lub XML, nie musi występować w każdym zapytaniu.

2.3 Studia literaturowe

2.3.1 Opis znanych rozwiązań

- Spotify Jam [12]

  Narzędzie umożliwia zakładanie sesji oraz dołączanie do nich, co pozwala użytkownikom wspólnie słuchać muzyki. Dostęp do tego rozwiązania jest oferowany przez platformę Spotify, jednak użytkownik zakładający sesję musi posiadać płatne konto premium. System nakłada ograniczenie, wymagając od dołączających posiadania urządzenia mobilnego (takiego jak telefon czy tablet) z zainstalowaną

aplikacją z zalogowanym kontem Spotify. Dołączenie jest możliwe poprzez zeskanowanie kodu QR, kliknięcie w udostępniony link lub zaakceptowanie zaproszenia od właściciela sesji. Maksymalna liczba uczestników jest ograniczona do 32 osób. Dodatkowo właściciel sesji ma możliwość usuwania użytkowników, którzy już dołączyli oraz nadania gościom uprawnień do sterowania głośnością urządzenia odtwarzającego muzykę.

- Festify [4]

  Festify to aplikacja umożliwiająca zakładanie oraz dołączanie do sesji. Dostęp do platformy wymaga posiadania konta premium w serwisie Spotify przez osobę zakładającą sesję. Dużym atutem tego rozwiązania są szerokie możliwości interakcji gości sesji z kolejką odtwarzania takie jak głosowanie na utwory, aby zostały szybciej odtworzone oraz dodawanie utworów przez gości. Platforma na podstawie utworów w kolejce proponuje podobne utwory.

Rozdział 3

Wymagania i narzędzia

3.1 Wymagania funkcjonalne i niefunkcjonalne

3.1.1 Wymagania funkcjonalne

Wymagania funkcjonalne systemu stanowią kluczową część projektu, definiując precyzyjnie zachowania, funkcje i operacje, których oczekuje się od systemu w celu zaspokojenia potrzeb użytkowników. Poprzez ich klarowne określenie możliwe jest wyznaczenie ram projektowych oraz stworzenie fundamentu dla późniejszego projektowania i implementacji systemu:

- Możliwość zalogowania przy pomocy konta Spotify:
  - Możliwość założenia sesji
  - Możliwość przeglądania założonych sesji
  - Możliwość usuwania sesji
  - Możliwość moderacji sesji:
    * Ograniczenie liczby utworów, które może dodać gość
    * Ograniczenie gatunków muzyki i autorów, które mogą być dodane
    * Zmiana kolejności utworów w kolejce
    * Nadanie priorytetu danemu gościowi, aby jego utwory pojawiały się wcześniej
    * Zablokowanie możliwości dodawania utworów danemu gościowi
    * Ograniczenie liczby gości, którzy mogą dołączyć do sesji
    * Ograniczenie maksymalnej długości utworów, które mogą być dodane
    * Możliwość włączenia funkcji głosowania, aby pomijać obecny utwór oraz ustawienia liczby głosów wymaganych do zatwierdzenia głosowania
  - Możliwość sterowania odtwarzaczem:

    - Zatrzymanie lub wznowienie odtwarzania
    - Zmiana głośności odtwarzania
    - Pomijanie lub cofanie do poprzedniego utworu

- Możliwość dołączenia do sesji jako gość:
  - Dołączenie do sesji odbywa się przez zeskanowanie kodu QR bądź użycie kodu dostępu
  - Możliwość ustawienia pseudonimu widocznego w sesji
  - Możliwość wyszukania utworów z bazy Spotify
  - Możliwość dodania wyszukanego utworu do kolejki
- Możliwość zapisania playlisty z danej sesji przez wszystkich użytkowników w sesji
- Aplikacja podpowiada kolejne utwory bazując na ostatnio dodanych

3.1.2 Wymagania niefunkcjonalne

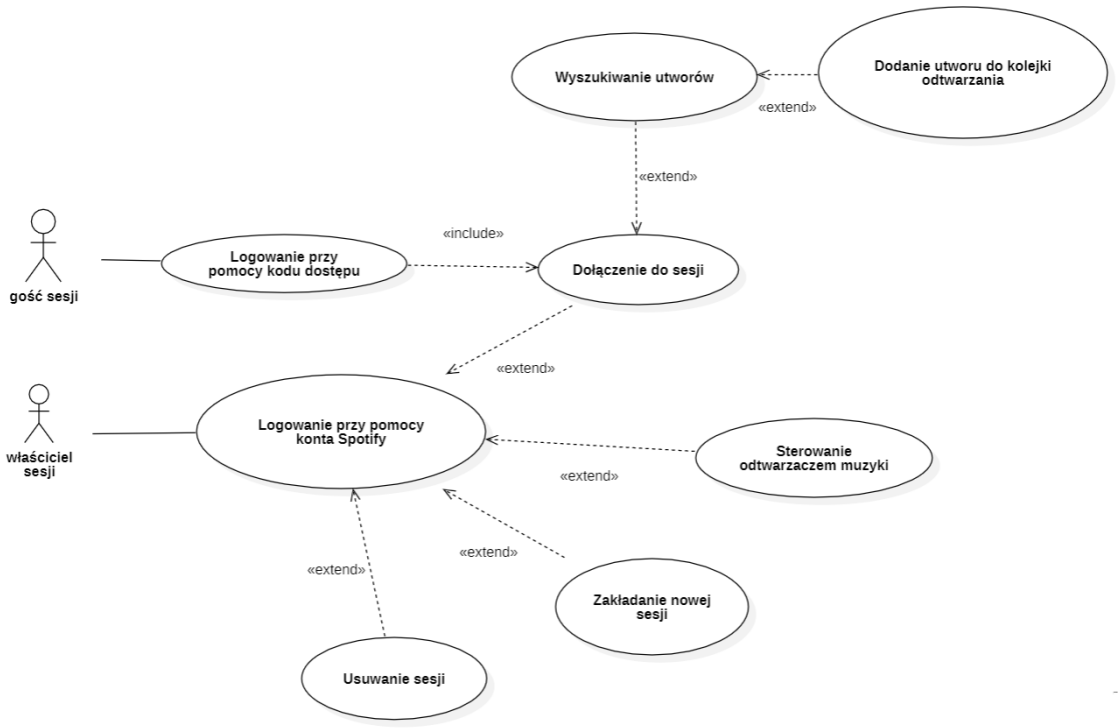
Wymagania niefunkcjonalne to kluczowe kryteria, które określają cechy systemu, takie jak wydajność, bezpieczeństwo czy skalowalność, nie będącymi bezpośrednio związane z jego funkcjonalnością. Definiują one granice, w których system musi działać, obejmując aspekty takie jak wydajność, dostępność czy też wymagania dotyczące interfejsu użytkownika:

- Aplikacja internetowa kompatybilna z różnymi urządzeniami mobilnymi, oraz różnymi systemami operacyjnymi
- Zabezpieczenie systemu przed nieautoryzowanym dostępem
- Użytkownicy nie mają możliwości ingerencji w konto Spotify właściciela sesji za wyjątkiem przewidzianej funkcji (dodawanie utworów do kolejki odtwarzania)
- Aplikacja ma posiadać prosty oraz intuicyjny interfejs użytkownika z osobnymi panelami dla gościa oraz właściciela sesji umożliwiający realizację funkcjonalności aplikacji
- Zapewnienie możliwości dalszej rozbudowy systemu oraz aktualizacji
- System jest zależny od API serwisu Spotify
- System jest zgodny z warunkami użytkowania serwisu Spotify [11]
- System posiada bazę danych SQL, w której przetrzymywane są informacje o użytkownikach, sesjach etc.

- Aplikacja jest ograniczona przez limit zapytań nałożony przez Spotify (w przypadku osiągnięcia takiego limitu użytkownik będzie zmuszony odczekać pewien czas przed kontynuacją użytkowania aplikacji)
- System jest w stanie obsłużyć wielu użytkowników oraz wiele sesji na raz.

3.2 Przypadki użycia

Zgodnie z rys. 3.1, aby skorzystać z systemu należy się najpierw zalogować jednym z dwóch sposobów. Gdy intencją użytkownika jest zakładanie oraz zarządzanie sesją należy użyć do logowania swojego konta Spotify. Następnie użytkownik ma możliwość założenia nowej sesji i ustawienia jej zasad, sterowanie odtwarzaczem muzyki lub dołączenie do jednej z istniejących już sesji lub usunięcie jej. Pozostali użytkownicy (goście) logują się używając kodu dostępu przez co dołączają do sesji. Wszyscy użytkownicy należący do sesji mogą wyszukiwać utwory oraz dodawać wybrane do kolejki odtwarzania.



Rysunek 3.1: Przypadki użycia systemu

3.3 Opis narzędzi

3.3.1 Technologie serwera aplikacji (warstwa usług)

Warunkami doboru technologii do wykonania części pracy działającej na serwerze była przede wszystkim uniwersalność oraz bezpieczeństwo aplikacji. Z tego powodu

wybrany został język Java, który uruchamiany jest na wirtualnej maszynie (ang. *Java Virtual Machine*, JVM) wprowadzającej dodatkową warstwę abstrakcji między kodem Java a sprzętem komputerowym. JVM posiada wiele mechanizmów bezpieczeństwa (np. izolacja kodu czy automatyczne zarządzanie pamięcią) oraz umożliwia bezproblemowe uruchamianie programu na różnych platformach [7]. Dużym atutem jest także ogromna społeczność programistów oraz bogaty ekosystem narzędzi oraz bibliotek, które można wykorzystać w projektach, co znacznie przyspiesza proces tworzenia oprogramowania.

Wraz z językiem Java użyta została platforma programistyczna Spring Boot [15], która znacznie ułatwia tworzenie aplikacji serwerowych, przez dostarczanie gotowych rozwiązań oraz domyślne konfiguracje. Dużym atutem jest także modularność - architektura opiera się na mikro-serwisach, które pozwalają na dzielenie aplikacji na mniejsze części, co znacznie ułatwia skalowanie oraz dalszy rozwój aplikacji. Obecnie do tworzenia oprogramowania w Javie, Spring Boot jest najczęściej wybieraną platformą programistyczną [8].

Aby usprawnić implementację warstwy bazy danych, użyty został moduł Spring Data JPA [5], który umożliwia pominięcie pisania dużych ilości kodu SQL, ponieważ podstawowe operacje CRUD są automatycznie generowane. Programista tworząc klasę, która będzie odpowiedzialna za komunikację z bazą danych musi jedynie zaimplementować odpowiedni interfejs programistyczny. Dodatkowo umożliwia generowanie bardziej skomplikowanych zapytań wykorzystując do tego nazwę metody oraz mechanizm refleksji, w praktyce oznacza to, że pisanie jakiegokolwiek kodu SQL może się okazać zbędne. Spring Data JPA jest bardzo elastyczny i umożliwia korzystanie z różnych systemów bazodanowych oraz zapewnia automatyczne mapowanie obiektów Javy na tabele w relacyjnej bazie danych (ang. *Object-Relational Mapping*, ORM).

Komunikacja z API Spotify została uproszczona przez użycie gotowego klienta [13], który jest narzędziem programistycznym wprowadzającym dodatkową warstwę abstrakcji ukrywającą szczegóły komunikacji z zewnętrznym API, co ułatwia tworzenie oprogramowania i pozwala programiście skupić się na logice aplikacji.

3.3.2 Technologie interfejsu użytkownika (warstwa prezentacji)

Kluczowym aspektem w tworzeniu interfejsu użytkownika jest jego przejrzystość, intuicyjność oraz w jaki sposób aplikacja będzie zarządzać stanem. Dodatkowo, zgodnie z założeniami projektu, system powinien być przystosowany do działania na urządzeniach mobilnych. Aby spełnić te wymogi do implementacji użyty został język TypeScript wraz z platformą programistyczną Angular [1], której głównym przeznaczeniem jest tworzenie jednostronicowych aplikacji internetowych (ang. *Single Page Application*, SPA). Dużą zaletą takiego rozwiązania jest pozbycie się konieczności przeładowywania zawartości strony w trakcie użytkowania aplikacji - zwiększa to interaktywność oraz

efektywniej wykorzystuje zasoby. Dodatkowym atutem jest modularność, co minimalizuje powtarzalność tworzonego kodu oraz asynchroniczność wywoływania zapytań, która pozwala użytkownikowi na dalsze korzystanie z aplikacji pomimo oczekiwania na odpowiedź serwera. Angular zapewnia także wsparcie dla przeglądarek internetowych na urządzeniach mobilnych.

Dodatkowo w projekcie użyta została biblioteka Angular Materials [2], która udostępnia gotowe szablony komponentów takich, jak wyskakujące okienka czy gotowe elementy formularzy. Pozwala to przyśpieszyć pracę nad tworzeniem oprogramowania.

3.3.3 Baza danych (warstwa danych)

Do realizacji warstwy danych użyty został system zarządzania bazami danych PostgreSQL. Dużą zaletą tego rozwiązania jest wysoka niezawodność zapewniona przez między innymi system kontroli spójności. Dodatkowym atutem jest skalowalność, która pozwala na zastosowanie tego rozwiązania zarówno w małych jak i w dużych systemach.

3.3.4 Platformy i usługi zewnętrzne

Aby zapewnić spójność między warstwą prezentacji a warstwą usług wykorzystany został generator kodu [14], który na podstawie utworzonego przez programistę pliku opisującego powstające API (punkty końcowe, sposób autoryzacji, przyjmowane oraz zwracane wartości, opis obiektów uczestniczących w wymianie między serwerem a klientem itp.) generuje interfejsy programistyczne, obiekty modelu aplikacji oraz serwisy wykonujące zapytania w docelowym kodzie i wybranej technologii (w tym przypadku Java-Spring Boot oraz TypeScript-Angular), można powiedzieć że tworzony jest szkielet aplikacji pozbawiony logiki. Rolą programisty jest implementacja tych interfejsów oraz zastosowanie gotowych serwisów w odpowiednich miejscach zarówno po stronie klienta jak i serwera. Takie rozwiązanie ułatwia wprowadzanie zmian w wymianie klient-serwer oraz usprawnia tworzenie oprogramowania.

W projekcie użyta została platforma Docker [3], która umożliwia konteneryzację oprogramowania, co oznacza uruchomienie aplikacji z gotowego obrazu (program wraz z zależnościami takimi jak biblioteki, pliki konfiguracyjne etc.) na wirtualnej maszynie, z którą można się komunikować oraz dowolnie zarządzać. Platforma wykorzystana była do symulacji serwera bazy danych.

3.3.5 Środowisko deweloperskie

Do implementacji oraz rozwoju omawianego systemu zostało użyte środowisko IntelliJ IDEA, zapewniające wsparcie dla języka Java oraz umożliwiające korzystanie z wtyczek oraz rozszerzeń, które znacznie ułatwiają pracę programisty. Interfejs użytkownika został

utworzony przy pomocy Visual Studio Code, który wspiera składnię HTML, CSS oraz TypeScript.

Rozdział 4

Specyfikacja zewnętrzna

4.1 Wymagania sprzętowe i programowe

Do działania aplikacji wymagany jest serwer, na którym uruchomiona będzie warstwa usługowa projektu. Bazując na przeprowadzonych testach wydajności działającej aplikacji można stwierdzić, że wykorzystanie zasobów jest niewielkie - średnio 300MB pamięci RAM, 60MB przestrzeni dyskowej oraz wykorzystanie procesora na poziomie 2% (testy uruchamiane na procesorze Intel Core i5-8250U). Aplikacja wymaga stałego połączenia do Internetu aby poprawnie działać (komunikacja z zewnętrznym API Spotify) oraz urządzenie, na którym aplikacja będzie uruchomiona musi być kompatybilne z maszyną wirtualną Javy (JVM). Dodatkowo do uruchomienia oprogramowania wymagana jest Java w wersji co najmniej 21. Warstwa prezentacji także wymaga serwera HTTP, z którego pobierane będą potrzebne dla klienta pliki ze strukturą aplikacji oraz potrzebnymi skryptami. Dodatkowo system wymaga dostępu do silnika bazodanowego PostgreSQL w wersji 16.1. Wszystkie elementy systemu mogą działać na jednym urządzeniu (serwer HTTP, API oraz bazy danych), bądź dowolnie rozłożone na kilka osobnych urządzeń.

Aby klient mógł korzystać z aplikacji wymagana jest przeglądarka internetowa na dowolnym urządzeniu wspierająca obsługę języka JavaScript, HTML5 oraz CSS3 oraz połączenie z serwerem.

4.2 Sposób instalacji

W celu instalacji należy pobrać skompilowane pliki projektu bądź skompilować je we własnym zakresie. Do kompilacji potrzebne będą następujące narzędzia:

- Środowisko Java w wersji 21
- Narzędzie Maven
- Narzędzie wiersza poleceń Angular CLI (Command Line Interface)

- Narzędzie Node Package Manager

Aby skompilować oprogramowanie warstwy usług należy uruchomić wiersz poleceń, odnaleźć plik pom.xml w plikach projektu i przejść do jego lokalizacji (folder "Backend"), a następnie wywołać następujące polecenie:

```
mvn compile
```

Warstwa interfejsu użytkownika kompilowana jest w analogiczny sposób. Należy uruchomić wiersz poleceń, odnaleźć ścieżkę projektu interfejsu użytkownika i przejść do głównego folderu (folder "Frontend"). Następnie należy wywołać polecenie

```
ng build
```

4.3 Sposób aktywacji

Przed uruchomieniem systemu należy skonfigurować aplikację serwera. W folderze z projektem warstwy usług należy zlokalizować plik "application.properites", w którym znajdują się następujące właściwości, które należy ustawić:

- **base.address** - adres serwera HTTP udostępniającego warstwę interfejsu użytkownika
- **qrcode.address.suffix** - przyrostek adresu, który zakodowany w formie kodu QR. Otrzymany adres w kodzie QR będzie złączeniem trzech elementów: adresu serwera HTTP (base.address), przyrostka (qrcode.address.suffix) oraz kodu dostępu do danej sesji.
- **spotify.authorize.callbackUrl.suffix** - przyrostek adresu dodany do adresu serwera HTTP, na który użytkownik będzie przekierowany po pomyślnej autoryzacji kontem Spotify.
- **spotify.client.id**, **spotify.client.secret** - wartości wymagane do komunikacji z zewnętrznym API serwisu Spotify.
- **jwt.secret** - klucz do generowania tokenów do autoryzacji użytkowników.
- **jwt.token.expirationtime** - czas wyrażony w sekundach określający ważność tokenu dostępu.
- **spring.datasource.url** - adres serwera bazy danych.
- **spring.datasource.username** - login użytkownika serwera bazy danych.
- **spring.datasource.password** - hasło użytkownika serwera bazy danych.

Razem z kodem źródłowym załączone zostały dwa pliki wsadowe ze skryptami uruchamiającymi serwer aplikacji oraz serwer udostępniający warstwę prezentacji w trybie deweloperskim. Możliwe jest także uruchomienie obu aplikacji z poziomu wiersza poleceń wywołując polecenie ng serve z głównego folderu z projektem warstwy interfejsu użytkownika (folder "Frontend"). Uruchomienie aplikacji serwera wymaga odnalezienia w folderze projektu aplikacji serwera (folder "Backend") folderu "target", w którym powinny znajdować się skompilowane pliki programu. Z tej lokalizacji należy wywołać polecenie java −jar backend−0.0.1−SNAPSHOT.jar

Aby system mógł działać poprawnie należy dodatkowo zarejestrować swoją aplikację na platformie Spotify Developer [10] - umożliwi to korzystanie z API serwisu. Należy także skonfigurować adres zwrotny taki sam jak w konfiguracji aplikacji serwera oraz przekopiować wygenerowane klucze (spotify.client.id, spotify.client.secret).

4.4 Kategorie użytkowników

W systemie istnieją dwie kategorie użytkowników:

- Właściciel sesji

  Do tej kategorii należą użytkownicy, którzy zostali zweryfikowani za pomocą własnego konta Spotify. Mają oni możliwość zakładania, moderacji oraz usuwania własnych sesji.

- Gość sesji

  W tej kategorii znajdują się użytkownicy, którzy dostęp do systemu uzyskali przez użycie poprawnego kodu dostępu, bądź zeskanowanie kodu QR uzyskanego od właściciela sesji. Użytkownik ten może należeć tylko do jednej sesji, w przypadku której usunięcia jest likwidowany. Ma możliwość wyszukania utworów oraz dodania ich do kolejki, jeśli zostały spełnione warunki ustawione przez właściciela sesji.

4.5 Sposób obsługi

Aby skorzystać z systemu należy na dowolnej przeglądarce wpisać skonfigurowany adres URL a następnie postępować dalej z instrukcjami wyświetlonymi na ekranie. Aplikacja w obecnej chwili nie wspiera rozdzielczości ekranów urządzeń mobilnych więc niektóre elementy mogą być niewidoczne lub trudno dostępne w pionowym układzie urządzenia.

Kod dostępu do sesji jest dostępny w widoku sesji, u góry obok nazwy (rys. 4.4), natomiast wygenerowany kod QR zostaje wyświetlony po użyciu przycisku "TV mode"dostępnego przy pasku do zarządzania odtwarzaniem muzyki.

4.6 Administracja serwerem

Możliwa jest ingerencja przez użycie systemu do zarządzania bazą danych. Przykładowo usunięcie użytkownika z odpowiedniej tabeli odbierze mu uzyskany dostęp do systemu i wymusi ponowną autoryzację. Dodatkowo, gdy aplikacja w serwisie Spotify jest zarejestrowana w trybie deweloperskim, użytkownik może zakładać sesję oraz korzystać z systemu jedynie po udzieleniu mu dostępu przez panel administratora udostępniony przez platformę Spotify.

4.7 Kwestie bezpieczeństwa

W celu uwierzytelnienia użytkowników użyty został standard JWT (ang. *JSON Web Token*), który definiuje format bezpiecznego oraz niezmiennego sposobu przesyłania informacji. Token to zaszyfrowany ciąg znaków składający się z trzech części:

- Nagłówek - określa typ tokenu oraz sposób szyfrowania.
- Dane - zawiera informacje o użytkowniku takie, jak kategoria, czy identyfikator oraz informacje o dacie wygaśnięcia tokenu.
- Podpis - jest to część utworzona na podstawie nagłówka oraz ładunku, jego celem jest zapewnienie integralności danych i uwierzytelnienia źródła.

Token zostaje udostępniony użytkownikowi w momencie, gdy jego tożsamość zostanie potwierdzona (przez wprowadzenie poprawnego kodu dostępu do sesji lub pomyślną autoryzację konta Spotify). Bez odpowiedniego tokenu użytkownik nie ma dostępu do żadnej funkcji systemu z wyjątkiem możliwości uwierzytelnienia. Dodatkowo niektóre funkcje są dostępne tylko dla użytkowników odpowiedniej kategorii - przykładowo gość nie ma możliwości sterowania odtwarzaczem muzyki. Generacja tokenu wykorzystuje klucz, który nie może być udostępniony poza serwerem, należy więc w pliku konfiguracyjnym serwera wygenerować oraz ustawić własny klucz (pole jwt.secret).

Aby zapewnić bezpieczeństwo kont Spotify użytkowników, odpowiednie tokeny umożliwiające dostęp do zewnętrznego API Spotify, identyfikujące użytkownika, są przechowywane w bazie danych i nigdy nie są przekazywane bezpośrednio do klientów. Zamiast tego, wykorzystywane są do komunikacji z API, a otrzymane wyniki są zwracane do klienta przez warstwę usług systemu co uniemożliwia przechwycenie i ingerencję przez osoby trzecie w konta Spotify użytkowników.

Kody dostępu do sesji są generowane losowo i składają się z 6 znaków (wszystkie cyfry oraz litery alfabetu łacińskiego bez rozróżnienia na wielkości liter). Zastosowanie takiego mechanizmu znacząco utrudnia próbę zgadnięcia kodu i zwiększa ochronę przed atakami typu brute-force, ponieważ liczba wszystkich kombinacji wynosi 36^6, czyli ponad 2 miliardy.

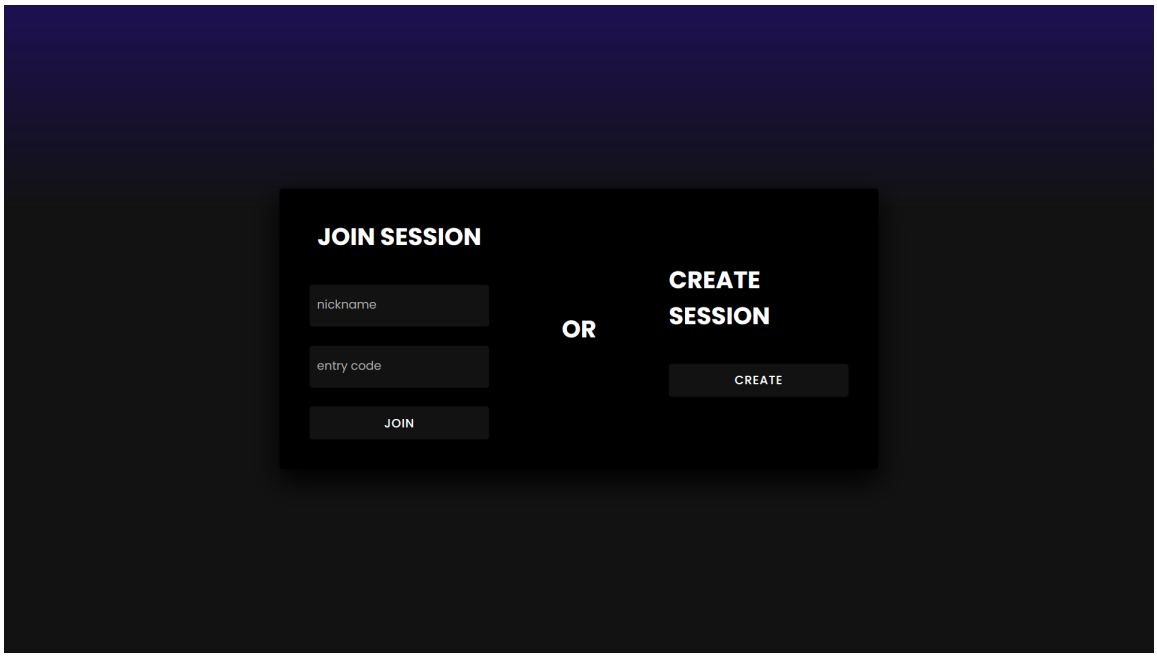
4.8 Przykład działania

Aby zobrazować przykład działania systemu przedstawiony zostanie scenariusz wydarzenia towarzyskiego, na którym powinna być odtwarzana muzyka w tle. Użytkownik decyduje się skorzystać z systemu więc wpisuje adres w przeglądarkę i loguje się używając swojego konta Spotify. Następnie zostaje przeniesiony do menu, gdzie może utworzyć nową sesję lub uruchomić własną już istniejącą. Użytkownik następnie uruchamia odtwarzanie muzyki używając do tego na przykład dedykowanej aplikacji Spotify i udostępnia wszystkim uczestnikom wydarzenia kod dostępu. Uczestnicy otwierają aplikację na swoich urządzeniach, gdzie mogą dołączyć do sesji korzystając z kodu. Po pomyślnej autoryzacji użytkownicy zostają przeniesieni do menu, gdzie mogą wyszukać oraz dodać utwory do kolejki odtwarzania, sprawdzić jakie utwory są następne w kolejce oraz kto jest uczestnikiem sesji. Dodatkowo, jeśli właściciel nie zablokował tej funkcji na koncie Spotify właściciela sesji, tworzona jest publiczna playlista, dostęp do której posiada każdy gość w sesji.

4.9 Scenariusze korzystania z systemu

4.9.1 Zakładanie oraz dołączanie do sesji

1. Przypadek użycia rozpoczyna się gdy użytkownik uruchomi aplikację w przeglądarce internetowej.
2. Użytkownik wybiera z menu opcję zakładania sesji (rys. 4.1).



Rysunek 4.1: Wygląd interfejsu użytkownika - główne menu logowania

3. Użytkownik jest przeniesiony na stronę autoryzacji Spotify, gdzie ma możliwość zalogowania się do swojego konta oraz proszony jest o wyrażenie zgody na określone warunki (rys. 4.2).

4. System przenosi użytkownika do menu gdzie udostępniony jest formularz zakładania sesji (rys. 4.3).

5. Po wypełnieniu i zatwierdzeniu przez użytkownika formularza, system sprawdza poprawność wypełnionych danych.

6. Dodana sesja pojawia się z lewej strony menu właściciela sesji.

7. Jeśli użytkownik zdecyduje się dołączyć do jednej z sesji wybiera ją z listy.

8. System przenosi użytkownika do widoku sesji w wersji właściciela (rys. 4.4).

4.9.2 Dołączenie do istniejącej sesji

1. Przypadek rozpoczyna się, gdy użytkownik uruchomi aplikację w przeglądarce internetowej przez wpisanie poprawnego adresu lub zeskanowanie kodu QR z adresem.

2. System wyświetla użytkownikowi menu (rys. 4.1), w którym proszony jest o wpisanie pseudonimu, którym będzie się posługiwał oraz kodu dostępu do sesji. W przypadku gdy użytkownik zeskanował kod QR kod dostępu będzie automatycznie wpisany w odpowiednie pole.

3. Po zatwierdzeniu formularza system sprawdza poprawność danych.

4. Użytkownik przeniesiony jest do menu widoku sesji w wersji gościa (rys. 4.5). Przypadek użycia się kończy.

4.9.3 Dodawanie utworu do kolejki

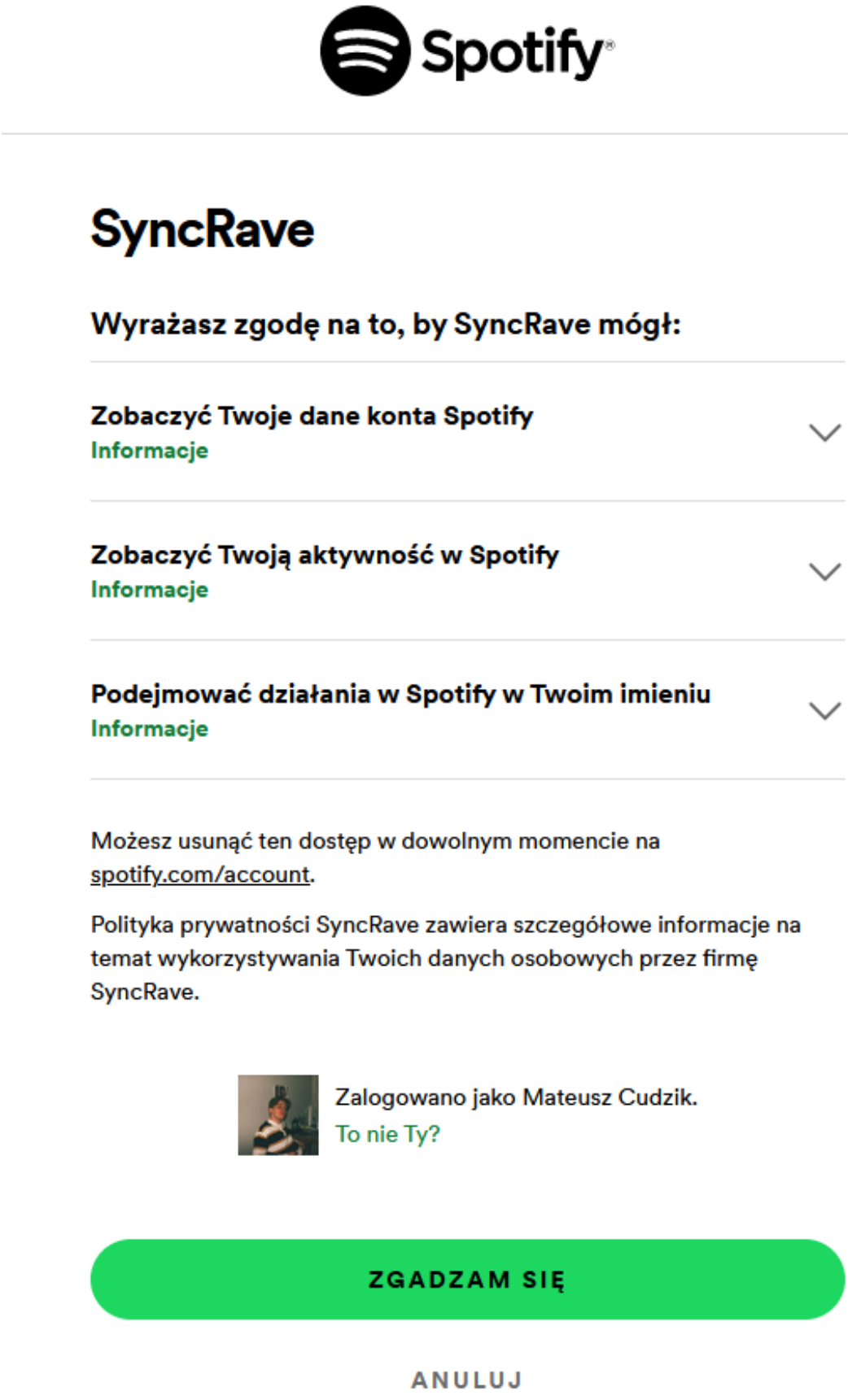
1. Przypadek rozpoczyna się gdy użytkownik znajduje się w menu widoku sesji rys. 4.5 lub rys. 4.4.

2. Użytkownik wpisuje w pole wyszukiwania odpowiednią frazę. Może to być artysta, tytuł utworu lub albumu.

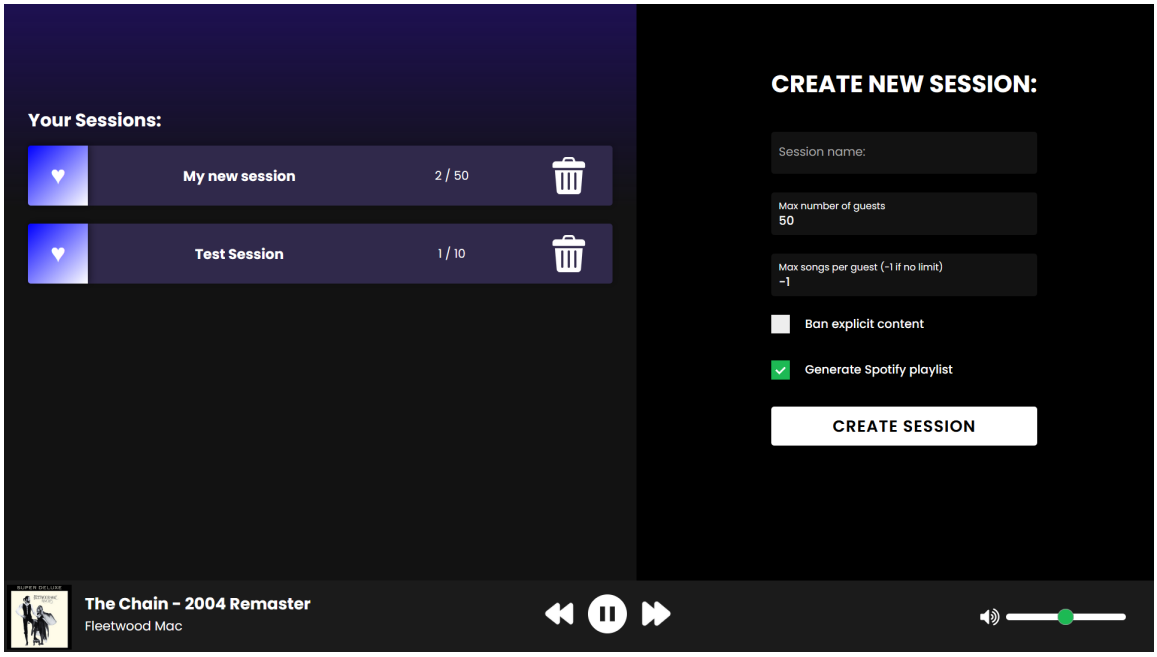
3. Po zatwierdzeniu system zwraca wyniki wyszukiwania.

4. Użytkownik może dodać znaleziony utwór używając odpowiedniego przycisku obok wyniku.

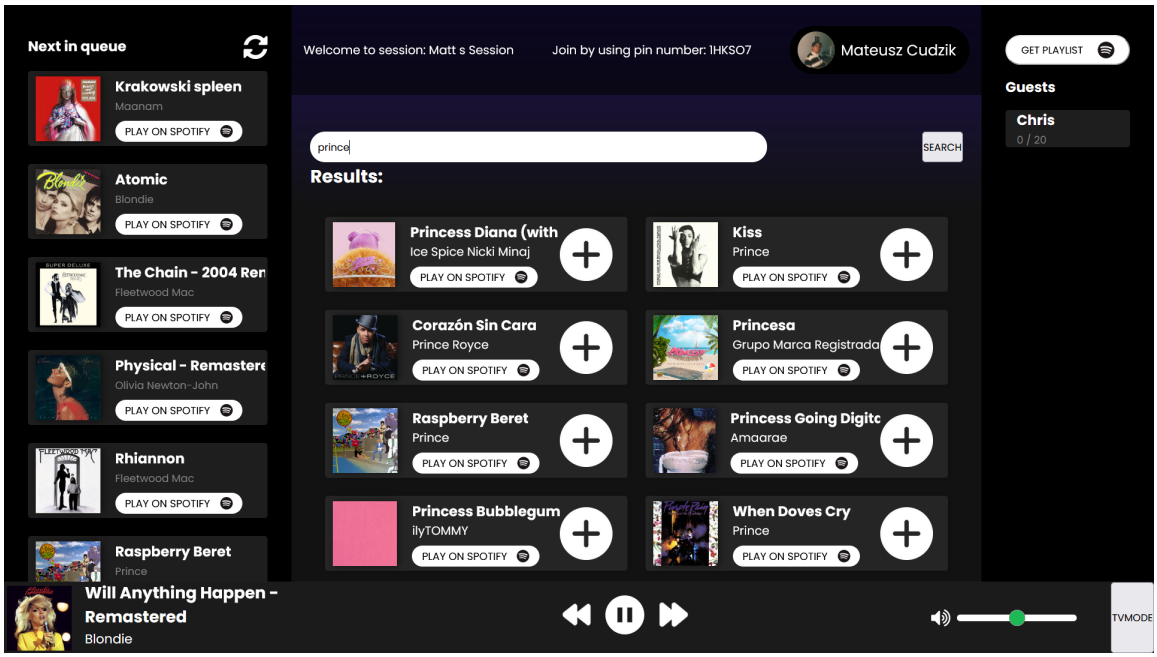
5. Jeśli utwór spełnia zasady sesji oraz użytkownik nie osiągnął limitu utwór zostaje dodany na koniec kolejki. Limity te nie dotyczą właściciela sesji. Przypadek użycia się kończy.



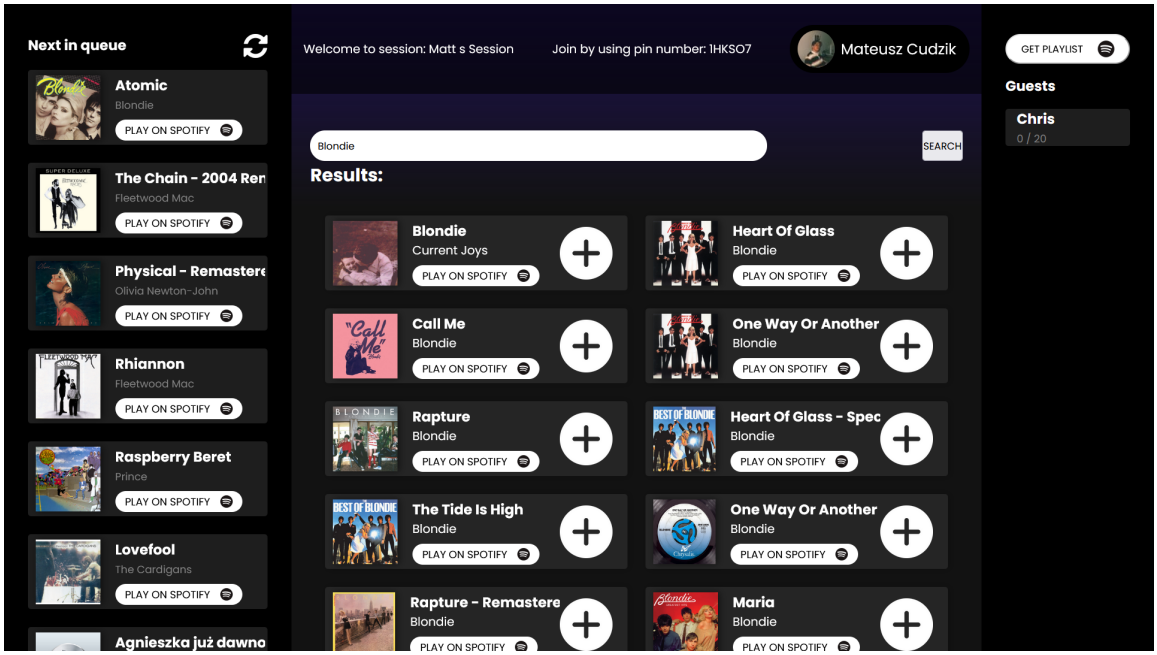
Rysunek 4.2: Wygląd interfejsu użytkownika - menu wyrażenia zgody w serwisie Spotify



Rysunek 4.3: Wygląd interfejsu użytkownika - główne menu właściciela sesji



Rysunek 4.4: Wygląd interfejsu użytkownika - widok sesji w wersji właściciela



Rysunek 4.5: Wygląd interfejsu użytkownika - widok sesji w wersji gościa

Rozdział 5

Specyfikacja wewnętrzna

5.1 Przedstawienie idei

Celem aplikacji, jest umożliwienie wielu użytkownikom wspólnego zarządzania listą odtwarzania utworów. Aby to osiągnąć wykorzystane zostało API serwisu Spotify, które umożliwia ingerencję w konto użytkownika, który wyraził na to zgodę (logując się do systemu). Dzięki udostępnionym funkcjom aplikacja ma dostęp do bazy utworów serwisu, możliwe jest także sterowanie odtwarzaczem muzyki, w tym dodawanie utworów do kolejki odtwarzania. Po poprawnym zalogowaniu przez użytkownika Spotify system zapisuje w bazie jego tokeny umożliwiające dostęp do zapytań, które ingerują w jego konto. W momencie, kiedy gość sesji doda utwór do kolejki, aplikacja serwera używając odpowiednich tokenów dodaje utwór w imieniu właściciela.

Aby zapewnić prostotę używania aplikacji, system nie wymaga od gości sesji rejestracji, a jedynie podania poprawnego kodu oraz pseudonimu. Serwer w odpowiedzi na udaną autoryzację odsyła token dostępu, bez którego wszystkie zapytania, które nie należą do punktu końcowego autoryzacji zostaną odrzucone. Tokeny dostępu mają datę ważności, po której przestają działać a konto zostaje dezaktywowane, użytkownik musi się wtedy ponownie uwierzytelnić.

5.2 Architektura systemu

Zgodnie z wielowarstwowym modelem aplikacji internetowej system składa się z 3 osobnych elementów: warstwy danych, usług oraz prezentacji. Warstwy te zostaną opisane w kolejnych podrozdziałach.

5.2.1 Warstwa danych

Warstwa odpowiedzialna za przetrzymywanie oraz pozyskiwanie danych została zrealizowana przy pomocy gotowych już narzędzi. Serwer, na którym działa silnik bazy

danych (PostgreSQL) obsługuje zapytania SQL. W trakcie implementacji serwer bazy danych został utworzony lokalnie na maszynie wirtualnej przy pomocy narzędzia Docker. Interakcję z bazą danych implementuje mechanizm dostępny jako moduł szkieletu Spring. Udostępnia on przejrzysty i łatwy w obsłudze interfejs programistyczny, który wprowadza pewną warstwę abstrakcji, aby ułatwić realizację interakcji z bazą danych.

5.2.2 Warstwa usług

Część systemu realizująca logikę biznesową została zaimplementowana korzystając ze szkieletu programistycznego Spring oraz języka Java. Aplikacja implementuje architekturę opartą na wzorcu model-widok-kontroler, natomiast warstwa widoku została zrealizowana w osobnej aplikacji. Dodatkowo istnieje jeszcze warstwa zapewniająca bezpieczeństwo systemu.

Po otrzymaniu zapytania przez serwer trafia ono do łańcucha filtrów, które sekwencyjnie sprawdzają żądanie, między innymi w celach bezpieczeństwa. W aplikacji zaimplementowane zostały dwa filtry, które sprawdzają tożsamość użytkownika (uwierzytelnianie) na podstawie tokenu JWT obecnego w nagłówku żądania. Kolejnym etapem jest autoryzacja, czyli potwierdzenie czy uwierzytelniony użytkownik ma dostęp do zasobu, którego żąda. Następnie żądanie jest mapowane do odpowiedniego kontrolera, który wywołuje odpowiednie serwisy implementujące logikę biznesową. Serwisy korzystają z repozytoriów, których głównym zadaniem jest komunikacja z bazą danych. Dodatkowo wykorzystują inne serwisy, które przykładowo realizują połączenie z zewnętrznym API platformy Spotify. Otrzymane wyniki są odpowiednio przetwarzane przez serwisy i zwracane do kontrolera, który odsyła odpowiedź do klienta. Serwisy stanowią część modelu aplikacji.

5.2.3 Warstwa prezentacji

Jest to część systemu odpowiedzialna za interakcję z użytkownikiem, została zaimplementowana przy użyciu szkieletu programistycznego Angular oraz języka TypeScript. Aplikacja została utworzona w architekturze SPA co oznacza, że aplikacja raz pobrana z serwera WWW nie wymaga przeładowania w celu zmiany wyświetlanych treści, lecz jest to wykonywane asynchronicznie w tle. Aplikacja składa się z komponentów, które reprezentują każdą część interfejsu użytkownika i realizują daną funkcję oraz określają styl i rozmieszczenie poszczególnych elementów. Przykładowo komponent, który implementuje funkcję odtwarzacza muzyki, może być ponownie używany w innych komponentach bez konieczności niepotrzebnego powtarzania kodu. Komponenty korzystają z serwisów, których zadaniem jest komunikacja z warstwą usług. Serwisy udostępniają funkcje, które odpowiednio tworzą, wysyłają oraz odbierają zapytania. Obiekty, które uczestniczą w wymianie między warstwą prezentacji a warstwą usług są zdefiniowane w modelu

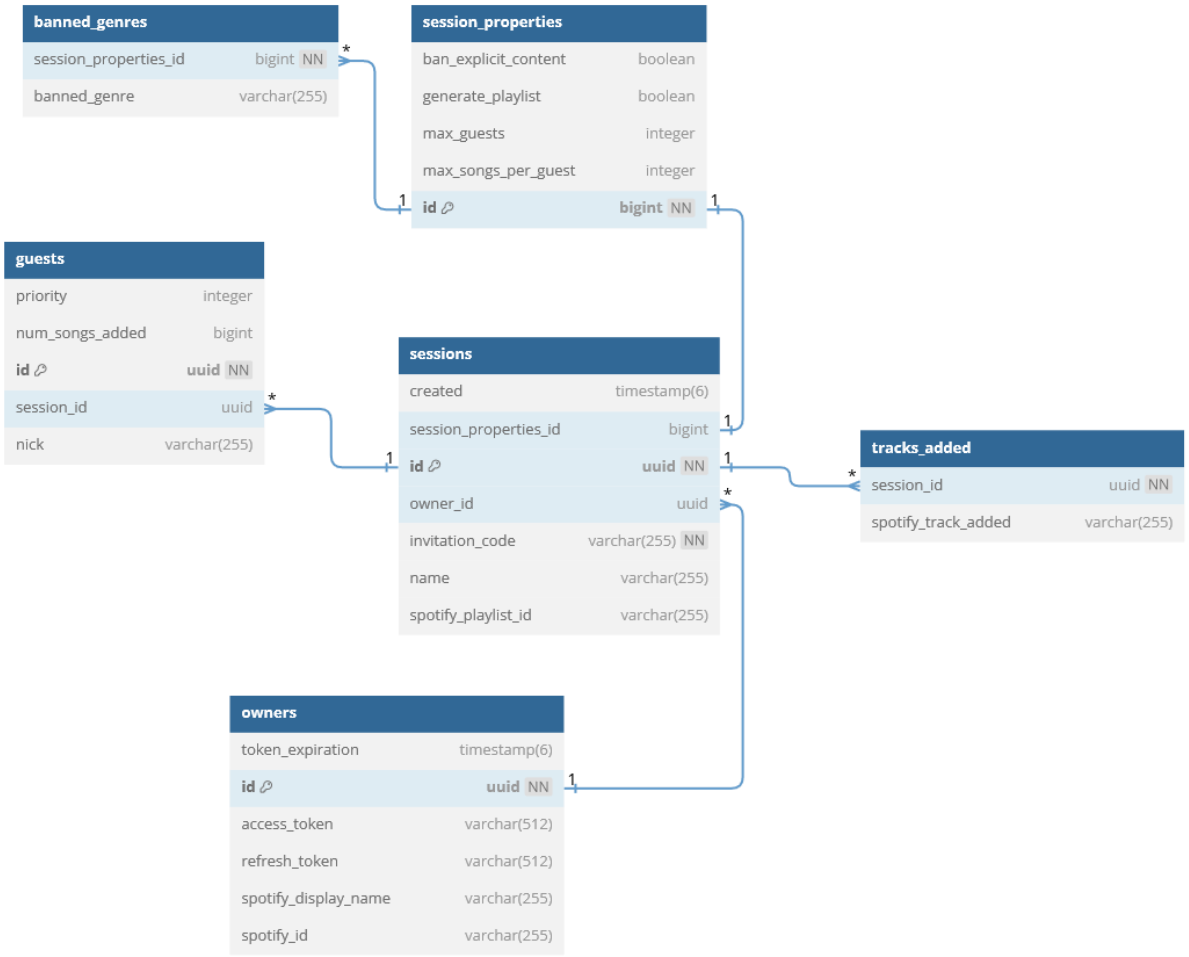
aplikacji.

5.3 Organizacja bazy danych

Aby zrealizować założenia projektu konieczne było zaprojektowanie oraz implementacja bazy danych. Diagram przedstawiony na rys. 5.1 pokazuje jakie relacje zachodzą między poszczególnymi tabelami. W centrum diagramu widnieje kluczowy element systemu - sesja, przechowuje ona podstawowe informacje takie, jak data utworzenia czy kod dostępu oraz nazwę. Sesja jest ściśle powiązana z tabelą właścicieli, gdzie właściciel może posiadać wiele sesji natomiast sesja tylko jednego właściciela. Tabela właścicieli przechowuje informacje niezbędne do komunikacji z zewnętrznym API platformy Spotify - tokeny dostępu oraz nazwę i identyfikator użytkownika w serwisie. Ustalone zasady sesji są unikalne dla każdej sesji i przechowują informacje o ustalonych przez właściciela wartościach takich, jak maksymalna liczba utworów, które może dodać gość czy maksymalna liczba gości w sesji oraz czy dozwolone będą utwory oznaczone jako zawierające treści dla dorosłych i czy powinna zostać utworzona lista z dodanymi utworami. W bazie danych przetrzymywane są także informacje o gościach, którzy dołączyli do sesji, takie jak liczba dodanych przez nich utworów oraz pseudonim, który ustawili dołączając do sesji. Gość może należeć tylko do jednej sesji, natomiast w sesji może znajdować się wielu gości. Baza danych zawiera także informacje o tym jakie utwory zostały dodane do kolejki w danej sesji oraz jakie gatunki utworów właściciel zablokował, funkcja ta jest obecnie dostępna jedynie z poziomu API aplikacji i nie jest możliwa do ustawienia używając interfejsu użytkownika.

5.4 Użyte biblioteki

- Lombok - biblioteka generująca powtarzalny i rutynowy kod taki jak konstruktor czy metody pobierające i ustawiające wartości obiektu przez użycie odpowiednich adnotacji, co poprawia czytelność kodu.
- io.jsonwebtoken - biblioteka obsługująca tokeny JWT, za pomocą której możliwe jest tworzenie, walidacja oraz parsowanie tokenów dostępu.
- ZXing - biblioteka umożliwiająca generowanie kodów QR.
- Jakarta Bean Validation - narzędzie służące do walidacji danych, przez odpowiednie adnotacje, dzięki czemu reguły dotyczące pól obiektów są wyraźnie zdefiniowane.
- ngx-cookie-service - moduł biblioteki obsługującej pliki cookie, w których trzymany jest token dostępu po autoryzacji użytkownika.



Rysunek 5.1: Diagram modelu bazy danych

5.5 Ważniejsze klasy

- Interfejsy kontrolerów API

  Są to automatycznie wygenerowane klasy przez narzędzie OpenAPI generator. Tworzą one szkielet aplikacji i definiują sygnatury metod, które należy zaimplementować. Każda z funkcji opatrzona jest odpowiednimi adnotacjami, które określają ścieżkę, parametry oraz ich walidację, metody autoryzacji, rodzaj żądania odpowiadającemu danemu punktowi końcowemu itp.

- Serwisy

  Klasy implementujące logikę biznesową korzystając z innych serwisów oraz bazy danych poprzez metody udostępniane przez repozytoria. Każda klasa implementuje odpowiedni interfejs aby zapewnić niezależność między deklaracją metod a ich implementacją. Większość serwisów dodatkowo mapuje otrzymane wyniki na obiekty, które uczestniczą w komunikacji z klientem.

- Klasa do generacji kodów dostępu i kodów QR

Generowanie nowych kodów dostępu odbywa się podczas tworzenia nowych sesji. Na rys. 5.2 przedstawiony jest kod, który jest za to odpowiedzialny. Z tablicy dozwolonych znaków w sposób losowy wybrane jest kolejno 6 znaków. Kluczowa tutaj jest klasa SecureRandom, która jest dużo bardziej bezpieczna pod względem kryptografii. Do generacji liczby losowej wykorzystane jest kilka źródeł losowości takich jak ruch sieciowy, ruch dyskowy czy czas systemowy.

```
private static final String candidateChars = "ABCDEFGHIJKLMNOPQRSTUVWXYZ1234567890";
public static String generateRandomCode(int length) {

    StringBuilder sb = new StringBuilder();
    SecureRandom random = new SecureRandom();

    for(int i = 0; i < length; i++){
        sb.append(candidateChars.charAt(random.nextInt(candidateChars.length())));
    }

    return sb.toString();
}
```

Rysunek 5.2: Fragment kodu - generowanie losowych kodów dostępu

5.6 Szczegóły implementacji

5.6.1 Zastosowane wzorce projektowe

- Pełnomocnik

  Aby zapewnić kontrolę dostępu do danych z serwisu Spotify oraz dalszą ich modyfikację i korzystanie z nich w logice biznesowej zastosowany został wzorzec pełnomocnika. Oznacza to, że otrzymane wyniki zanim zostaną zwrócone do klienta są dodatkowo modyfikowane i używane w implementacji logiki, aby przykładowo sprawdzić czy utwór, który użytkownik chciał dodać spełnia zasady sesji. Dodatkowo chronione są dzięki temu dane użytkowników Spotify, ponieważ ich tokeny dające dostęp do niektórych funkcji konta w serwisie nie są nigdy udostępniane na zewnątrz.

- Model-Widok-Kontroler

  Zastosowany został podział, który rozdziela logikę od warstwy prezentacji w systemie, zapewniając jednocześnie niezależność obu komponentów, co pozwala dowolnie zmieniać ich sposób realizacji. Komunikację między modelem a widokiem zapewnia kontroler.

- Architektura wielowarstwowa (dokładnie trójwarstwowa)

  Zastosowany został podział, dzielący system na trzy warstwy: prezentację, czyli interfejs użytkownika, logikę biznesową oraz dane. Każdy z wymienionych modułów to osobna aplikacja, dzięki czemu zachowana jest niezależność.

- Wstrzykiwanie zależności

  Mechanizmy szkieletu programistycznego Spring implementują ten wzorzec, aby przekazywać zależności między obiektami w prosty sposób bez konieczności ingerencji w przekazywany komponent. Zależności są dostarczane do obiektu co poprawia łatwość utrzymania aplikacji.

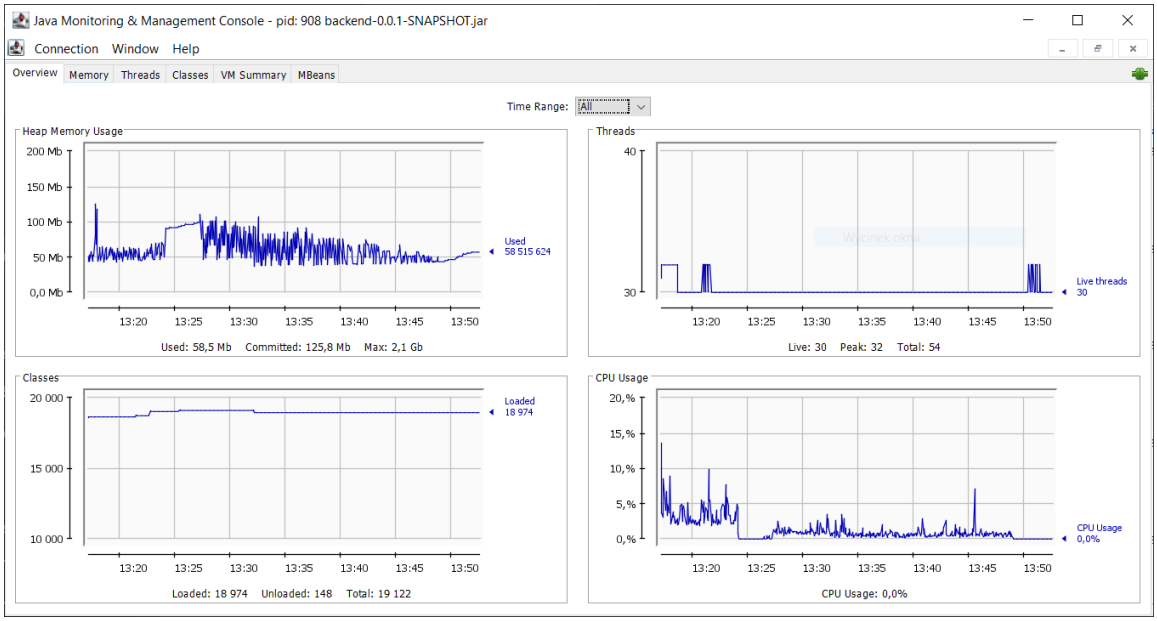
Rozdział 6

Weryfikacja i walidacja

6.1 Sposób testowania w ramach pracy

Podczas rozwoju systemu na bieżąco były wykonywane testy integracyjne oraz testy jednostkowe mające na celu walidację poprawności działania zaimplementowanych komponentów.

Dodatkowo przeprowadzony został test wydajnościowy, w którym zmierzona została ilość użytej pamięci RAM przez uruchomioną aplikację, do której trafiały 4 różne zapytania co 20ms. Wyniki zaprezentowane na rys. 6.1 pokazują, że aplikacja najwięcej zasobów zużyła podczas uruchomienia, a w ciągu normalnego działania zużywane są niewielkie ilości - zużycie procesora na poziomie 2% oraz wykorzystanie pamięci



Rysunek 6.1: Wyniki testu wydajnościowego

6.2 Organizacja eksperymentów

Aby przeprowadzić testy użyte zostało narzędzie Postman, które umożliwia tworzenie oraz wysyłanie żądań HTTP do serwera z ustawionymi danymi, nagłówkami oraz parametrami i pozwala odczytać otrzymane wyniki. Narzędzie to pozwoliło na bieżące testy API bez konieczności implementacji interfejsu użytkownika. Platforma została także użyta w teście wydajnościowym, aby symulować ruch w sieci.

Pomiary wydajnościowe zostały wykonane przy użyciu konsoli monitorowania i zarządzania w Javie, która umożliwia monitorowanie aplikacji oraz zbieranie danych diagnostycznych.

Dodatkowo podczas implementacji interfejsu użytkownika, wykorzystane zostało narzędzie inspektora dostępne w przeglądarce internetowej.

6.3 Przypadki testowe, zakres testowania

W aplikacji serwera przetestowane zostały wszystkie punkty końcowe pod względem poprawności przyjmowanych danych (niepoprawna długość, niedozwolone znaki, puste lub wypełnione białymi znakami łańcuchy znaków itp.), dodatkowo sprawdzone zostało czy zapytania, które odnoszą się do nieistniejących danych są poprawnie odrzucane (nieistniejące sesje, użytkownicy itp.). Punkty końcowe aplikacji serwera zostały przetestowane także pod względem bezpieczeństwa - nieważny token bezpieczeństwa lub jego całkowity brak czy próby dostępu do niedozwolonych dla danej kategorii użytkowników funkcji takiej, jak sterowanie odtwarzaczem muzyki przez gościa sesji. Przetestowana została także poprawność działania zasad sesji, które blokują użytkownikom dodawanie utworów ponad ustawiony limit czy dołączanie do sesji, która jest już pełna.

6.4 Wykryte i usunięte błędy

Podczas implementacji systemu wykryto oraz usunięto stosunkowo dużo błędów, najczęściej występowały w kodzie aplikacji serwera. Przykładem może być błąd, który powodował odrzucanie wszystkich zapytań pochodzących z aplikacji klienta przez serwer. Błąd nie występował podczas korzystania z narzędzi do testowania API, lecz pojawił się w momencie implementacji interfejsu użytkownika. Przyczyną okazał się źle skonfigurowany moduł odpowiedzialny za kontrolę dostępu. Przeglądarka zanim wyśle do serwera właściwe żądanie wysyła zapytania typu OPTIONS, który jest częścią mechanizmu bezpieczeństwa przeglądarki. Aplikacja serwera odrzucała wszystkie zapytania bez nagłówka autoryzacji w tym wszystkie zapytanie typu OPTIONS, bez których komunikacja jest niemożliwa. Kilka błędów, które się pojawiły były spowodowane

literówkami bądź użyciem złej funkcji o podobnej nazwie, przez co program się kompilował, lecz nie działał poprawnie.

W interfejsie użytkownika wykryto i usunięto wiele błędów związanych ze stylem takie, jak źle ułożone elementy czy niepoprawnie wyświetlające się moduły. Usunięte zostały także błędy związane z komunikacją z aplikacją serwera, które uniemożliwiały poprawną autoryzację czy korzystanie z niektórych funkcji systemu.

Rozdział 7

Podsumowanie i wnioski

7.1 Uzyskane wyniki w świetle postawionych celów i zdefiniowanych wymagań

Aplikacja spełnia większość postawionych założeń. Prawie wszystkie funkcje, które nie zostały zaimplementowane są stosunkowo proste do dodania. Przykładowo, aby możliwość blokady danych gatunków muzyki, która została zaimplementowana po stronie serwera i jest udostępniona przy użyciu odpowiedniego punktu końcowego, ale z powodu braku czasu nie została w pełni wdrożona w interfejsie użytkownika, działała należałoby dodać odpowiedni formularz w warstwie prezentacji. Innym przykładem jest nadanie właścicielowi sesji możliwości usuwania gości z sesji, wymagałoby to jedynie dodania nowego punktu końcowego z prostą funkcją oraz odpowiednio podpiętego przycisku. Istnieje natomiast kilka funkcji, dla których realizacji należałoby przeorganizować strukturę warstwy usług. Przykładem jest możliwość nadawania priorytetu gościowi czy mechanizm głosowania aby pomijać lub przyspieszać pojawienie się utworu. Wymaga to ingerencji w kolejkę, której serwis Spotify nie udostępnia. Możliwe jest tylko dodanie utworu lecz nie zmiana kolejności czy usunięcie z kolejki. Potencjalnym rozwiązaniem byłoby przetrzymywanie kolejki z utworami w bazie danych do momentu gdy powinny być odtworzone i dopiero wtedy wysyłanie ich do serwisu Spotify.

7.2 Możliwe kierunki rozwoju systemu

System można rozwinąć przez dodawanie kolejnych funkcji, przykładowo stworzenie dedykowanej aplikacji na urządzenia mobilne. Dodanie funkcji karaoke znacznie zwiększyłoby interaktywność aplikacji, synchronizacja tekstu z dźwiękiem byłaby możliwa, należałoby jedynie uzyskać dostęp do bazy danych z tekstami utworów, które zawierają znaczniki czasowe kolejnych linijek tekstu. Dodatkowo rozszerzenie funkcji o obsługę innych platform do strumieniowania muzyki umożliwiłoby większej liczbie

użytkowników korzystanie z serwisu.

Ważnym elementem jest usprawnienie interfejsu użytkownika, w szczególności dodanie responsywności na urządzeniach mobilnych, na których obecnie korzystanie z aplikacji jest niewygodne.

System mógłby także skorzystać na poprawie mechanizmów bezpieczeństwa, które nie były priorytetem w tej pracy. Obecnie aplikacje są wrażliwe na wiele rożnych rodzajów ataków ze strony osób trzecich.

Obecnie żądania kierowane do zewnętrznego API Spotify wykonywane są w sposób synchroniczny, może to mieć duży wpływ na optymalizację w przypadku dużej liczby aktywnych użytkowników oraz powodować opóźnienia na serwerze.

7.3 Problemy napotkane w trakcie pracy

W trakcie pracy pojawiły się trudności w implementacji technicznej. Używanie szkieletów programistycznych znacznie ułatwia tworzenie oprogramowania pod warunkiem, że programista posiada dokładną wiedzę o sposobie działania narzędzi, z których korzysta. Podczas implementacji pracy zabrakło momentami tego głębszego zrozumienia sposobu funkcjonowania systemu co stanowiło przeszkodę i wymusiło konieczność pozyskania wnikliwej wiedzy na temat mechanizmów działania tych narzędzi.

Błędem okazało się niedostateczne sprawdzenie możliwości, które udostępnia serwis Spotify przez swoje API. Wynikiem czego początkowe założenia, które system powinien spełniać okazały się zbyt trudne do osiągnięcia w określonym czasie. Etap analizy oraz możliwości jest kluczowy podczas tworzenia projektu i nie powinien być zaniedbany.

Bibliografia

[1] *Angular Docs*. 2023. URL: https://v16.angular.io/docs (term. wiz. 14. 12. 2023).

[2] *Angular Materials*. URL: https://v16.material.angular.io/components/categories (term. wiz. 21. 12. 2023).

[3] *Docker Docs*. URL: https://docs.docker.com/ (term. wiz. 14. 12. 2023).

[4] *Festify*. 2023. URL: https://festify.rocks/ (term. wiz. 29. 12. 2023).

[5] Oliver Gierke, Thomas Darimont, Christoph Strobl, Mark Paluch, Jay Bryant i Greg Turnquist. *Spring Data JPA*. 2023. URL: https://docs.spring.io/spring-data/jpa/reference/index.html (term. wiz. 14. 12. 2023).

[6] Ahmed E. Hassan i Richard C. Holt. „Architecture Recovery of Web Applications". W: ICSE '02. Association for Computing Machinery, 2002, 349–359. ISBN: 158113472X.

[7] Tim Lindholm, Frank Yellin, Gilad Bracha, Alex Buckley i Daniel Smith. *The Java® Virtual Machine Specification*. 2023. URL: https://docs.oracle.com/javase/specs/jvms/se21/html/index.html (term. wiz. 14. 12. 2023).

[8] M. Mythily, A. Samson Arun Raj i Iwin Thanakumar Joseph. „An Analysis of the Significance of Spring Boot in The Market". W: *2022 International Conference on Inventive Computation Technologies (ICICT)*. 2022, s. 1277–1281. DOI: 10.1109/ICICT54344.2022.9850910.

[9] International Federation of the Phonographic Industry. *Global Music Report*. 2023. URL: https://ifpi-website-cms.s3.eu-west-2.amazonaws.com/GMR_2023_State_of_the_Industry_ee2ea600e2.pdf (term. wiz. 20. 12. 2023).

[10] Spotify. *Web API - Spotify for developers*. 2023. URL: https://developer.spotify.com/documentation/web-api (term. wiz. 21. 12. 2023).

[11] *Spotify Developer Terms*. 2023. URL: https://developer.spotify.com/terms (term. wiz. 21. 12. 2023).

[12] *Spotify Jam*. 2023. URL: https://support.spotify.com/pl/article/jam/ (term. wiz. 29. 12. 2023).

[13] *Spotify Web API Java*. 2023. URL: https://github.com/spotify-web-api-java/spotify-web-api-java (term. wiz. 21. 12. 2023).

[14] *Swagger Codegen*. URL: https://swagger.io/docs/specification/about/ (term. wiz. 14. 12. 2023).

[15] Phillip Webb, Dave Syer, Josh Long, Stéphane Nicoll, Rob Winch, Andy Wilkinson, Marcel Overdijk, Christian Dupuis, Sébastien Deleuze, Michael Simons, Vedran Pavić, Jay Bryant, Madhura Bhave, Eddú Meléndez, Scott Frederick i Moritz Halbritter. *Spring Boot Reference Documentation*. 2023. URL: https://docs.spring.io/spring-boot/docs/current/reference/html/ (term. wiz. 14. 12. 2023).

Dodatki

Spis skrótów i symboli

API	interfejs programowania aplikacji (ang. *application programming interface*)
CRUD	operacje: Tworzenie, Odczyt, Aktualizacja, Usuwanie (ang. *Create, Read, Update, Delete*)
HTTP	protokół komunikacji między klientem a serwerem (ang. *Hypertext Transfer Protocol*)
IDE	środowisko programistyczne (ang. *Integrated Development Environment*)
JSON	notacja obiektów JavaScript (ang. *JavaScript Object Notation*)
JPA	interfejs programistyczny w Javie definiujący sposoby zarządzania relacyjnymi danymi (ang. *Java Persistence API*)
JWT	token uwierzytelniający (ang. *JSON Web Token*)
JVM	maszyna wirtualna Javy (ang. *Java Virtual Machine*)
kod QR	dwuwymiarowy kod zawierający dane (ang. *Quick Response code*)
ORM	mapowanie obiektowo-relacyjne (ang. *Object-Relational Mapping*)
pamięć RAM	pamięć operacyjna (ang. *Random Access Memory*)
REST	architektura reprezentacyjnego transferu stanu (ang. *Representational State Transfer*)
SPA	jednostronnicowa aplikacja internetowa (ang. *Single Page Application*)
SQL	język zapytań strukturalnych (ang. *Structured Query Language*)
URL	adres zasobu sieciowego (ang. *Uniform Resource Locator*)

Źródła

W pracy do implementacji interfejsu użytkownika wykorzystano gotowe projekty graficzne, które zostały zmodyfikowane na potrzeby systemu oraz ikony pochodzące z zewnętrznego serwisu. Linki do poszczególnych projektów i źródeł ikon są następujące:

- `https://www.codewithfaraz.com/content/147/create-a-stunning-spotify-clone-project-with-html-and-css`
- `https://codepen.io/ayush602/pen/mdQJreW`
- `https://loading.io/css/`
- `https://www.flaticon.com/`

Lista dodatkowych plików, uzupełniających tekst pracy

W systemie do pracy dołączono dodatkowe pliki zawierające:

- źródła programu
- dokumentacja techniczna w postaci pliku opisującego API

Spis rysunków

Spis tabel