Sprawozdanie Projektowe: Karaoke Machine

Roksana Żyłka, Kamil Kubrak, Mikołaj Wołoszyn,
Kamil Ziemiański, Marcin Szuszko

9 stycznia 2026 r.

Spis treści

1 Opis funkcjonalny systemu

Projekt **Karaoke Machine** to nowoczesna platforma webowa dedykowana miłośnikom śpiewu. Głównym założeniem systemu jest umożliwienie użytkownikom korzystania z ogromnych zasobów serwisu YouTube przy jednoczesnym wsparciu profesjonalnego promptera tekstowego.

Do kluczowych funkcjonalności należą:

- **Wyszukiwanie dynamiczne:** Integracja z YouTube Data API v3 pozwala na przeszukiwanie milionów utworów bezpośrednio z poziomu aplikacji.
- **System synchronizacji napisów:** Wykorzystanie parsera plików .lrc oraz integracja z zewnętrzną bazą LRCLIB zapewnia wyświetlanie tekstu piosenki zsynchronizowanego z podkładem muzycznym.
- **Personalizacja profilu:** Użytkownicy mogą zarządzać własną listą polubionych utworów, które są zapisywane w bazie danych.
- **Panel Administratora:** Moduł dedykowany do zarządzania bazą użytkowników oraz weryfikacji metadanych utworów.

2 Opis technologiczny

System został zaprojektowany w architekturze rozproszonej z wyraźnym podziałem na warstwę prezentacji (Frontend) oraz logiki biznesowej (Backend).

2.1 Backend

- **Framework:** Spring Boot 3.5.7.
- **Bezpieczeństwo:** Spring Security z wykorzystaniem tokenów JWT (JSON Web Token).
- **Baza danych:** MySQL 8.0 z mapowaniem obiektowo-relacyjnym Hibernate.
- **Komunikacja:** REST API zwracające dane w formacie JSON.

2.2 Frontend

- **Technologia:** Vue.js / Vite.
- **Komunikacja API:** Biblioteka Axios z obsługą nagłówków autoryzacyjnych.
- **Stylizacja:** Tailwind CSS.

3 Podział obowiązków i odpowiedzialności

- **Roksana Żyłka (Lider):** Architektura backendu, implementacja zabezpieczeń JWT, konfiguracja polityki CORS oraz zarządzanie dokumentacją techniczną.

- **Mikołaj Wołoszyn (Frontend Lead):** Implementacja odtwarzacza, parsera napisów .lrc oraz integracja z API LRCLIB.
- **Marcin Szuszko:** Logika wyszukiwania lokalnego, implementacja serwisów zarządzania piosenkami oraz optymalizacja zapytań SQL.
- **Kamil Ziemiański:** Tworzenie komponentów interfejsu użytkownika, m.in. paska wyszukiwania oraz widoku Panelu Administratora.
- **Kamil Kubrak:** Testy integracyjne API, walidacja poprawności tokenów w narzędziu Postman oraz przygotowanie scenariuszy testowych.

4 Instrukcja uruchomienia systemu

4.1 Uruchomienie lokalne

1. **Skonfiguruj bazę MySQL:** utwórz schemat `karaoke_db`.
2. **W pliku application.properties** ustaw poprawne dane logowania do bazy.
3. **Wykonaj aktualizację schematu** (wymagane pole `thumbnail_url`):

```
ALTER TABLE songs ADD COLUMN thumbnail_url VARCHAR(255);
```

4. **Uruchom aplikację** backendową za pomocą komendy: `mvn spring-boot:run`.
5. W folderze frontendu wykonaj `npm install` oraz `npm run dev`.

4.2 Uruchomienie zdalne i CORS

System jest przygotowany do pracy zdalnej. Należy upewnić się, że w klasie `SecurityConfig` dozwolone są źródła (AllowedOrigins) odpowiadające adresom IP serwerów produkcyjnych (obecnie skonfigurowane dla localhost:5173 oraz 5174).

5 Wnioski projektowe

Roksana Żyłka: Największym wyzwaniem była stabilizacja autoryzacji JWT przy zapytaniach typu preflight. Poprawna konfiguracja CORS okazała się kluczowa dla integracji frontendu z backendem.

Mikołaj Wołoszyn: Kluczową lekcją była obsługa niespójności w zewnętrznych API. Konieczne było stworzenie mechanizmów obronnych (fallback) na wypadek braku zsynchronizowanych napisów dla niszowych utworów.

Marcin Szuszko: Projekt pokazał, jak ważna jest elastyczność bazy danych. Zmiana podejścia z danych statycznych na dynamiczny import z YouTube wymagała gruntownego przemyślenia encji JPA.

Kamil Ziemiański: Praca nad panelem administratora nauczyła mnie, jak ważne jest projektowanie interfejsu w sposób czytelny, nawet przy dużej ilości danych spływających z różnych źródeł API.

Kamil Kubrak: Testowanie API w Postmanie uświadomiło mi, że nawet drobna literówka w nagłówku autoryzacyjnym może zablokować pracę całego zespołu na wiele godzin.