Akademia Górniczo-Hutnicza
im. Stanisława Staszica w Krakowie

Wydział Fizyki i Informatyki Stosowanej
Kierunek Informatyka Stosowana

Przetwarzanie Danych w Chmurach Obliczeniowych

Music Wiki

dokumentacja projektu zaliczeniowego

Anna Nowak

8.12.2025, Kraków

1. Założenia

Celem projektu było stworzenie aplikacji internetowej typu Single Page Application (SPA), służącej do wizualizacji i zarządzania bazą wiedzy muzycznej. Głównym założeniem architektonicznym było wykorzystanie grafowej bazy danych, która w naturalny sposób odzwierciedla powiązania w świecie rzeczywistym, w tym przypadku pomiędzy artystami, wydanymi przez nich utworami oraz albumami.

Kluczowe założenia obejmowały:

- wykorzystanie chmurowej, grafowej bazy danych **Neo4j AuraDB**
- zastosowanie języka **GraphQL** do komunikacji między klientem a serwerem
- użycie **React** do budowy aplikacji internetowej
- implementacja interaktywnej wizualizacji grafu przy pomocy modułu *react-force-graph-2d*.

2. Temat projektu

Aplikacja webowa Music Wiki pozwala na budowanie biblioteki muzycznej w formie grafu, przez co oferuje przejrzystą wizualizację powiązań między artystami, albumami i piosenkami.

Frontend aplikacji został zbudowany przy użyciu React oraz Vite. Do komunikacji z serwerem GraphQL wykorzystano bibliotekę Apollo Client. Wizualizacja grafu zrealizowana jest za pomocą *react-force-graph-2d*.

Wykorzystane biblioteki:

- *@apollo/client* - klient GraphQL do pobierania danych z serwera Apollo i zarządzania stanem
- *react-force-graph-2d* - komponent do renderowania grafu, który wizualizuje dane w postaci węzłów i krawędzi.

3. Grafowa baza danych

W aplikacji użyto bazy danych **Neo4j** w wersji chmurowej AuraDB.

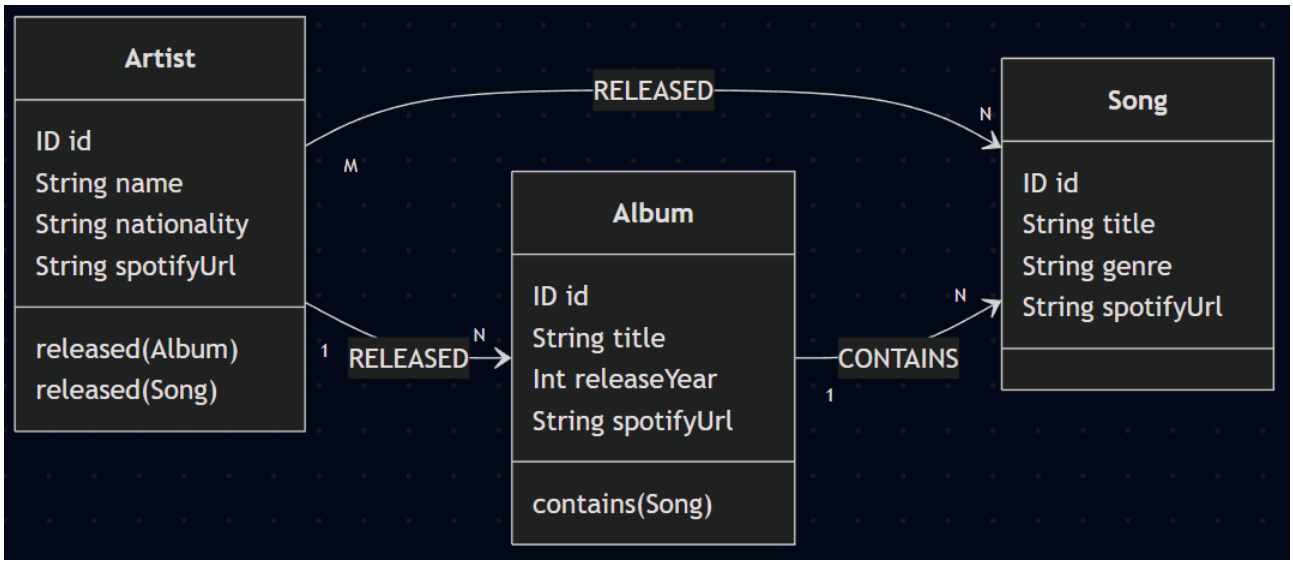
Poniżej przedstawiono definicję typów zawartych w bazie danych:

```
type Artist {
    id: ID! @id
    name: String!
    nationality: String
    spotifyUrl: String
    albums: [Album!]! @relationship(type: "RELEASED", direction: OUT)
    songs: [Song!]! @relationship(type: "RELEASED", direction: OUT)
}

type Album {
    id: ID! @id
    title: String!
    releaseYear: Int
    spotifyUrl: String
    artist: [Artist!]! @relationship(type: "RELEASED", direction: IN)
    songs: [Song!]! @relationship(type: "CONTAINS", direction: OUT)
}

type Song {
    id: ID! @id
    title: String!
    genre: String
    spotifyUrl: String
```

```
    artists: [Artist!]! @relationship(type: "RELEASED", direction: IN)
    album: [Album!]! @relationship(type: "CONTAINS", direction: IN)
}
```

W systemie zdefiniowano trzy główne typy węzłów:

- (Artist) –> (Album) (1:N)
- (Artist) –> (Song) (M:N)
- (Album) –> (Song) (1:N).



Grafika 1: Diagram ERD.

4. Funkcjonalność aplikacji

Aplikacja oferuje szereg interaktywnych funkcji do zarządzania grafem muzycznym:

1. **tworzenie nowego węzła** - użytkownik może dodawać nowe elementy do grafu za pomocą formularza. Po wybraniu typu węzła i wypełnieniu wymaganych pól, nowy element jest dodawany do bazy danych i pojawia się na grafie
2. **edycja i usuwanie węzła** - kliknięcie na dowolny węzeł na grafie powoduje wyświetlenie panelu bocznego z jego szczegółami. Użytkownik ma możliwość edycji informacji o węźle lub jego całkowitego usunięcia z bazy danych
3. **dodawanie krawędzi** - użytkownik może zdefiniować relacje, co pozwala na budowanie sieci powiązań
4. **usuwanie krawędzi** - istnieje również możliwość usuwania istniejących relacji między węzłami
5. **manipulacja grafem** - graf jest interaktywny. Użytkownik może przesuwać węzły oraz przybliżać i oddalać widok. Aplikacja udostępnia również wyszukiwarkę, która podświetla pasujące węzły.

5. Obsługa aplikacji

Do uruchomienia programu wymagane jest środowisko **Node.js** oraz menedżer pakietów **NPM**.

5.1. Instalacja

W celu przygotowania środowiska do uruchomienia aplikacji konieczne jest obranie wymaganych zależności zarówno dla serwera jak i dla klienta:

- w katalogu głównymi aplikacji: *npm install*
- w katalogu *frontend*: *npm install*

5.2. Uruchomienie

Należy uruchomić dwa niezależne procesy:

1. uruchomienie backend: *npm run dev*
2. uruchomienie frontend: *npm run frontend*

Po uruchomieniu aplikacja i serwer dostępne są pod adresami:

- **Serwer GraphQL:** `http://localhost:4000`
- **Aplikacja:** `http://localhost:5173`

6. Podsumowanie i wnioski

W przypadku projektu Music Wiki grafowa baza danych okazała się bardziej intuicyjna i elastyczna do budowy oraz przedstawienia schematu danych niż baza relacyjna. Wykorzystanie biblioteki `@neo4j/graphql` znacznie przyspieszyło pracę nad funkcjonalnościami związanymi z komunikacją z bazą danych. Aplikacja spełnia założenia SPA pozwalając użytkownikowi na płynne użytkowanie programu, interakcje z UI oraz, co najważniejsze, manipulowanie wizualizacją danych przy pomocy grafu w czasie rzeczywistym.