Bazy Danych

Projekt:

System rekomendacji na przykładzie utworów, artystów i słuchaczy w grafowej bazie danych Neo4j

Autorzy:

Konrad Siuzdak

Łukasz Wolski

Cel Projektu

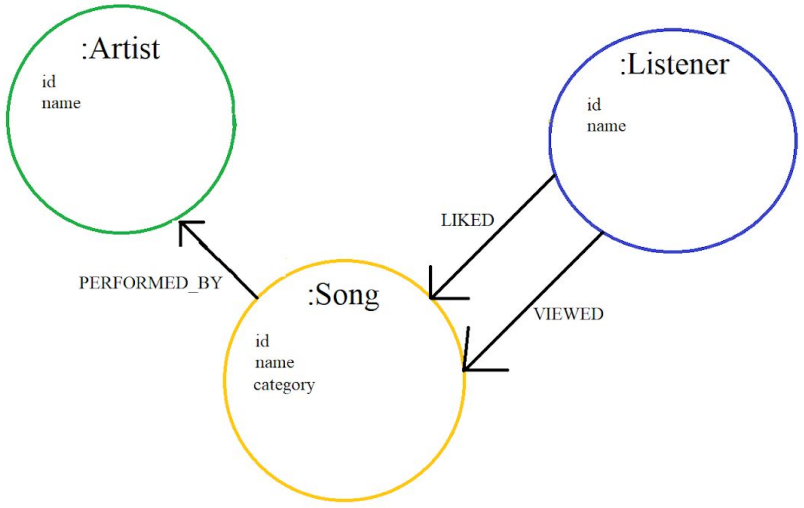
Celem projektu było stworzenie aplikacji, która na podstawie aktualnych preferencji słuchaczy będzie sugerować utwory, które być może im się spodobają.

Użyte technologie

- Baza danych: Neo4j Community Edition
- Język programowania: Java, Cipher
- Sterownik: Bolt
- Maven, Neo4jOGM

Wygląd bazy danych

Wykorzystano grafową bazę danych, której schemat znajduje się poniżej.



Gdzie category jest typem wyliczeniowym o wartościach ze zbioru: Blues,Rock,Pop,Jazz. Ważne jest, aby pola name były unikalne, dlatego nałożono na nie constraint UNIQUE.

Funkcjonalność

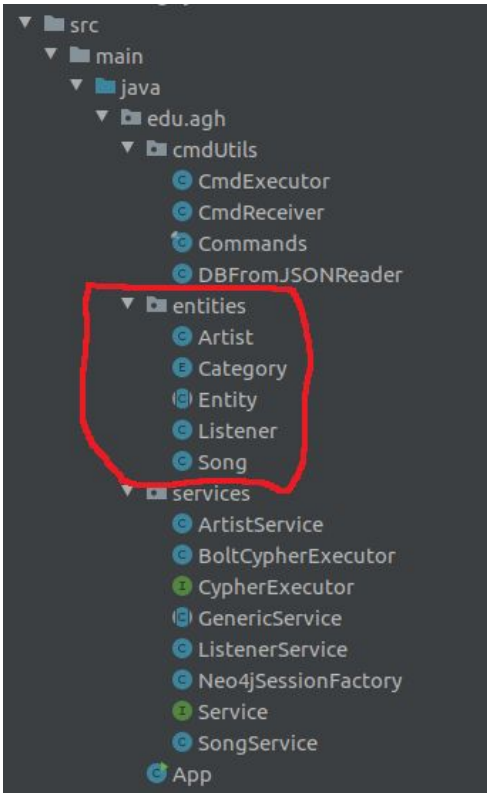
Użytkownik może z poziomu konsoli specyfikować następujące polecenia:

- add_song [tytuł utworu] [kategoria][nazwy artystów]
  Dodaje piosenkę o podanym tytule do podanej kategorii [1].
- add_artist [nazwa artysty]
  Dodaje artystę o podanej nazwie.

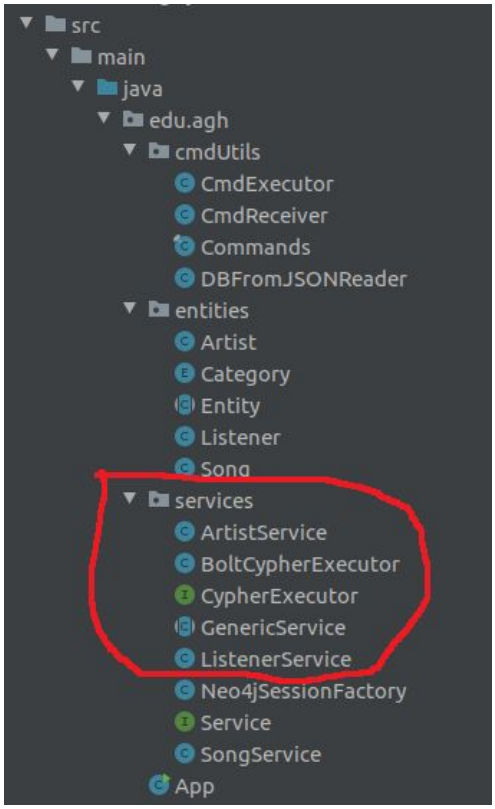
- add_listener [nazwa słuchacza]
  Dodaje słuchacza o podanej nazwie.
- get_recommendation_by_category [nazwa słuchacza]
  Pobiera rekomendację na podstawie słuchanych utworów przez słuchacza. Spośród lubianych(lub jeśli ich nie ma to odwiedzonych) utworów wybierany jest losowo jeden, a następnie jego kategoria, na podstawie której wyświetlana jest propozycja.
- get_recommendation_by_artist [nazwa_słuchacza]
  Pobiera rekomendację na podstawie słuchanych utworów przez słuchacza. Spośród lubianych(lub jeśli ich nie ma to odwiedzonych) utworów wybierany jest losowo jeden, a następnie jego autor(również losowo), na podstawie którego wyświetlana jest propozycja od tego samego autora.
- get_recommendation_by_similar_listeners [nazwa_słuchacza]
  Pobiera rekomendację na podstawie słuchanych utworów przez słuchacza. Na podstawie wyborów innych słuchaczy, którzy słuchali tych samych utworów co słuchacz wyspecyfikowany w poleceniu wyświetlana jest propozycja.
- listener_like_song [nazwa_słuchacza] [nazwa_piosenki]
  Pozwala z poziomu aplikacji “polubić” utwór o podanej nazwie przez podanego słuchacza. Innymi słowy, tworzy relację LIKED między słuchaczem a utworem.
- listener_viewed_song [nazwa_słuchacza] [nazwa_piosenki]
  Analogicznie jak powyżej, jednak tworzy relację VIEWED.
- find_artists_by_song [nazwa_piosenki]
  Wyświetla artystów, którzy stworzyli podaną piosenkę.
- find_songs_by_artist [nazwa_artysty]
  Wyświetla piosenki danego artysty.
- quit
  Zamyka aplikację.

Struktura kodu

W projekcie wykorzystano bibliotekę Neo4j Object Graph Mapper, która służy do rzutowania obiektów z Javy na wierzchołki oraz relacje między obiektami na krawędzie w grafie. Klasy które są rzutowane (Artist, Song, Listener) wraz z klasą abstrakcyjną (Entity) oraz typem wyliczeniowym Enum (Category) stanowiącym atrybut piosenki, znajdują się w katalogu “entities” folderu źródłowego:



Za rzutowanie struktury w Javie na graf, odpowiedzialne są przede wszystkim usługi. To dzięki nim tworzymy obiekty klas z katalogu entities, relacje pomiędzy nimi, a także wyszukujemy istniejących w bazie obiektów jeżeli jest to konieczne. Wszystkie usługi znajdują się w katalogu services:



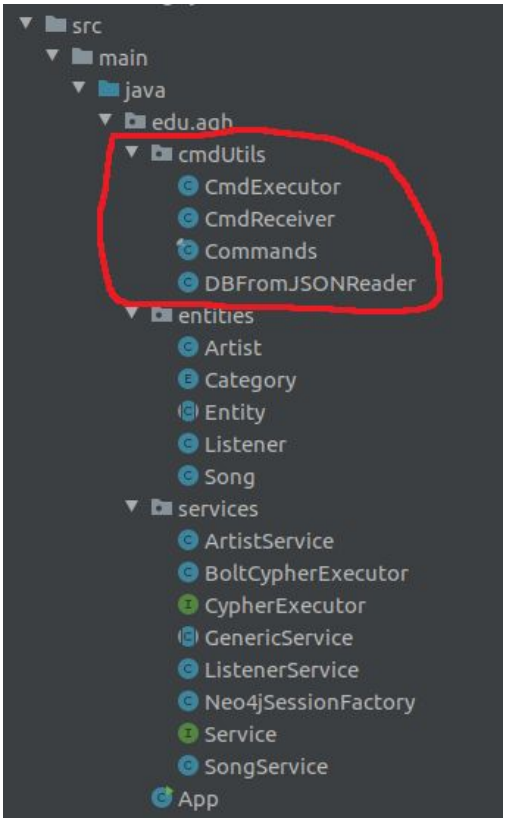
Jak sama nazwa wskazuje, każdy taki service odpowiada za funkcjonalność związaną ściśle z danym obiektem:

- ArtistService umożliwia tworzenie nowych artystów czy też szukanie piosenek wykonywanych przez artystę,
- SongService umożliwia tworzenie nowych utworów oraz wyszukiwanie wykonawców danej piosenki,
- ListenerService pozwala na dodawanie nowego słuchacza do bazy, zwrócenie rekomendacji oraz polubienie piosenki lub odznaczenie jako odsłuchaną.

Pozostałe klasy i interfejsy znajdujące się w tym katalogu są ściśle związane z operacjami jakie należy wykonać w powyżej wymienionych usługach:

- Service to wspólny interfejs dla powyższych usług,
- GenericService to klasa abstrakcyjne, która implementuje wspólne dla wszystkich dziedziczących klas operacje (stanowi szablon),
- CypherExecutor - interfejs odpowiedzialny za wykonywanie query
- BoltCypherExecutor - implementuje wyżej wymieniony interfejs, za pomocą niego wykonujemy query w bazie

Klasa Neo4jSessionFactory jest przede wszystkim odpowiedzialna za tworzenie sesji dla usług wymagających połączenia z bazą. Do obsługi aplikacji bazodanowej służą klasy z katalogu cmdUtils:

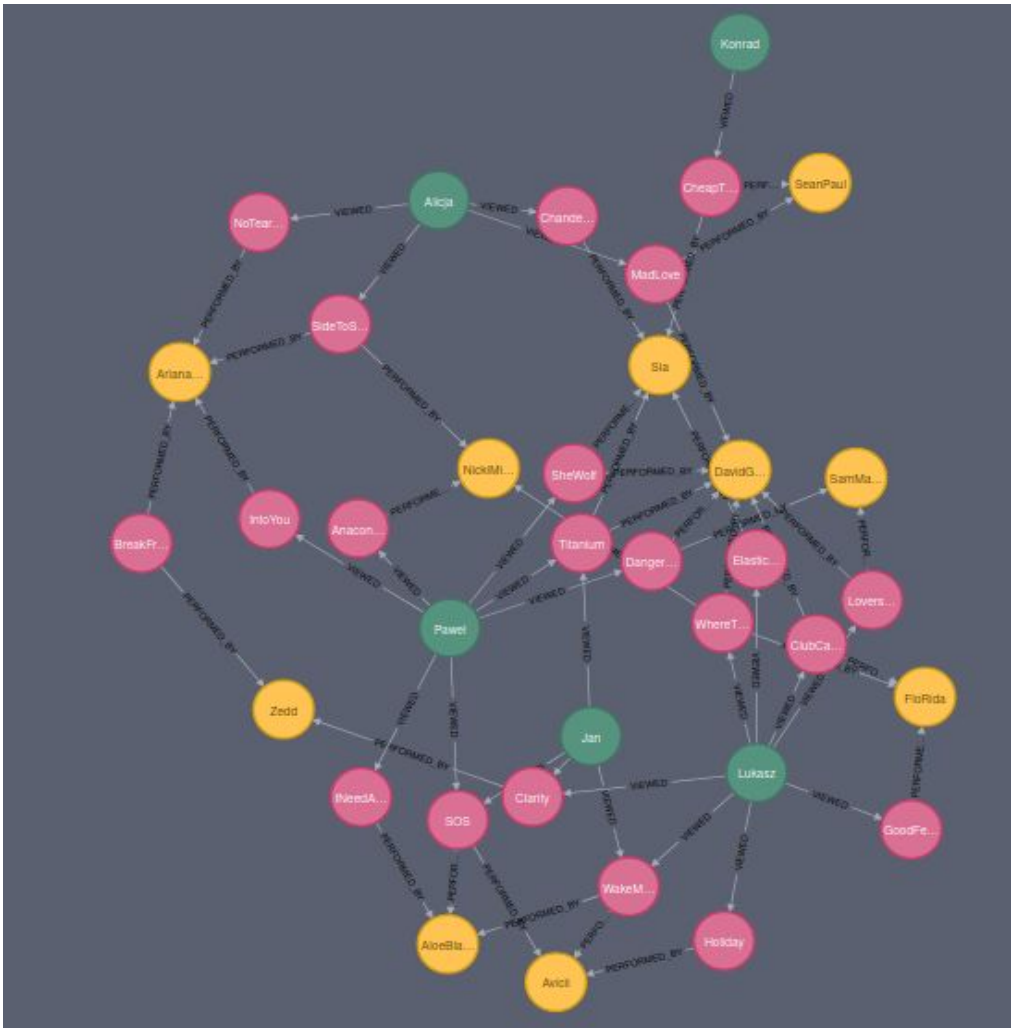


Klasa DBFromJSONReader jak sama nazwa wskazuje wczytuje dane z plików JSON do lokalnej bazy, zaś Commands zawiera wszystkie możliwe komendy do wpisania przez użytkownika w konsoli. Pozostałe dwie klasy są odpowiedzialne za odczytywanie komend z konsoli (CmdReceiver) oraz ich wykonanie (CmdExecutor). Wszystkie wyżej wymienione klasy stanowią silnik aplikacji bazodanowej i znajdują się w katalogu źródłowym (src). Efekt

działania jest możliwy do zobaczenia dzięki ręcznie napisanym dwóm wersjom (większa i mniejsza) mock danych:

Przykładowa Baza Danych

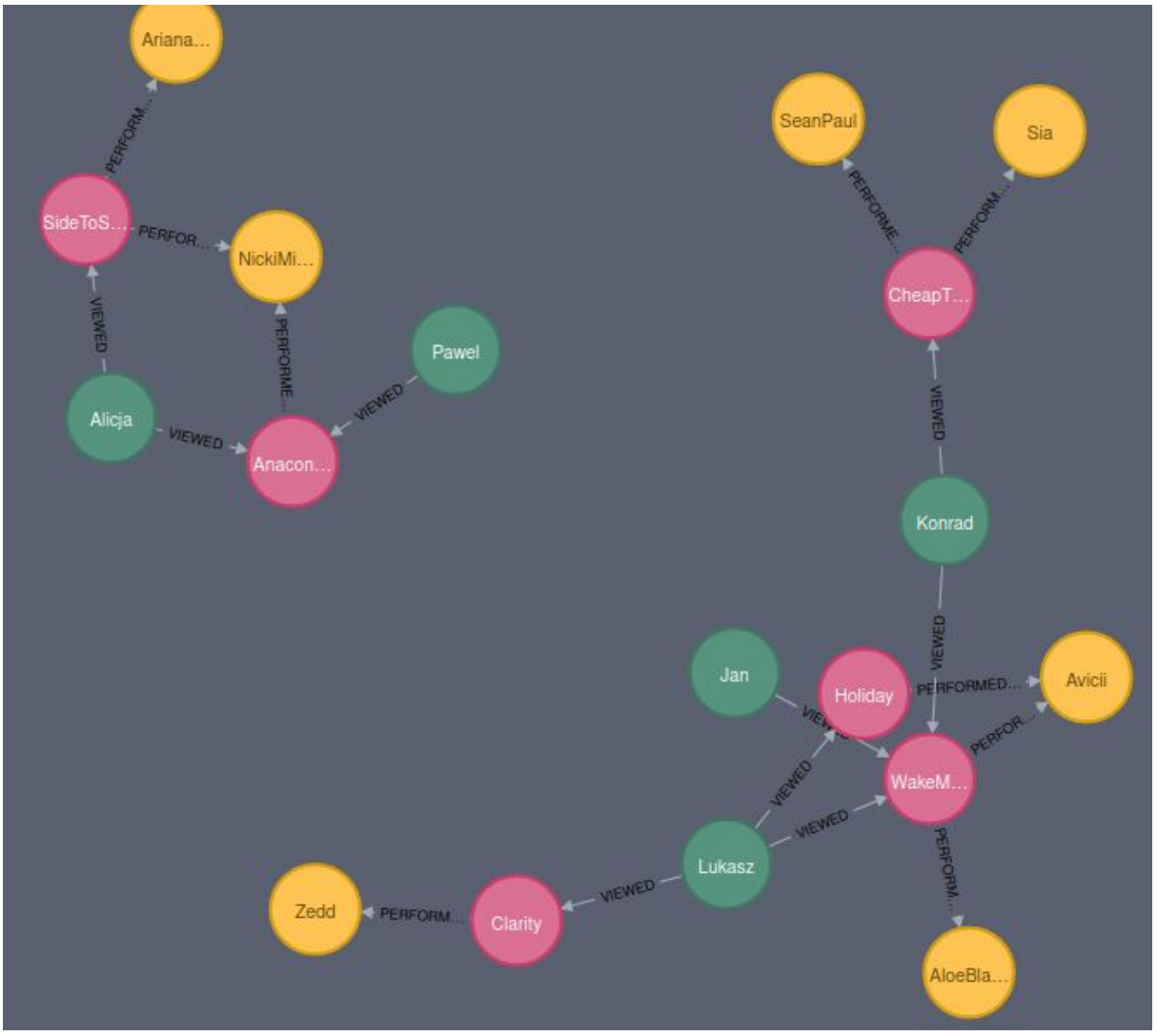
Stworzono przykładową bazę danych, której struktura pobierana jest z plików JSON i prezentuje się następująco:



Na zielono są słuchacze, na czerwono utwory, na żółto autorzy.

Przykładowe działanie

Do zilustrowania przykładowego działania aplikacji przygotowano mniejszą bazę danych, aby widać było poprawność działania.




```
Provide instructions:
get_recommendation_by_similar_listeners Pawel
Jun 09, 2020 8:16:40 AM org.neo4j.driver.internal.logging.JULogger info
INFO: Direct driver instance 589363823 created for server address localhost:7687
Getting recommendation for "Pawel" by similar listeners.
SideToSide of Blues PERFORMED BY NickiMinaj, ArianaGrande,
Jun 09, 2020 8:16:40 AM org.neo4j.driver.internal.logging.JULogger info
INFO: Closing driver instance 589363823
Jun 09, 2020 8:16:40 AM org.neo4j.driver.internal.logging.JULogger info
INFO: Closing connection pool towards localhost:7687
```

Dla Pawła wywołanie powinno pokazać piosenki, których słucha Alicja i tak właśnie jest.

```
Provide instructions:
get_recommendation_by_category Konrad
Jun 09, 2020 8:19:59 AM org.neo4j.driver.internal.logging.JULogger info
INFO: Direct driver instance 2138005960 created for server address localhost:7687
Getting recommendation for "Konrad" from "Blues"
SideToSide of Blues PERFORMED BY NickiMinaj, ArianaGrande,
Jun 09, 2020 8:19:59 AM org.neo4j.driver.internal.logging.JULogger info
INFO: Closing driver instance 2138005960
Jun 09, 2020 8:19:59 AM org.neo4j.driver.internal.logging.JULogger info
INFO: Closing connection pool towards localhost:7687
```

Konrad jest połączony z CheapThrills, które jest z kategorii Blues, więc wyświetlony jest inny utwór z tej kategorii, który ponadto nie należy do tej samej składowej spójnej, więc wywołanie zwraca prawidłowy wynik dla tego przykładu.

```
Provide instructions:
get_recommendation_by_artist Lukasz
Jun 09, 2020 8:23:55 AM org.neo4j.driver.internal.logging.JULogger info
INFO: Direct driver instance 2101527076 created for server address localhost:7687
Getting recommendation for "Lukasz" by artist named "Avicii"
Jun 09, 2020 8:23:55 AM org.neo4j.driver.internal.logging.JULogger info
INFO: Closing driver instance 2101527076
Jun 09, 2020 8:23:55 AM org.neo4j.driver.internal.logging.JULogger info
INFO: Closing connection pool towards localhost:7687
```

```
Provide instructions:
get_recommendation_by_artist Lukasz
Jun 09, 2020 8:24:10 AM org.neo4j.driver.internal.logging.JULogger info
INFO: Direct driver instance 1601935322 created for server address localhost:7687
Getting recommendation for "Lukasz" by artist named "Zedd"
Jun 09, 2020 8:24:10 AM org.neo4j.driver.internal.logging.JULogger info
INFO: Closing driver instance 1601935322
Jun 09, 2020 8:24:10 AM org.neo4j.driver.internal.logging.JULogger info
INFO: Closing connection pool towards localhost:7687
```

Łukasz słucha już wszystkich piosenek Aviciiego oraz Zedda, więc nie zostaje wyświetlona żadna rekomendacja na podstawie słuchanych artystów, ponieważ takowa nie istnieje.

Z racji na pewną losowość wyboru artystów i kategorii do rekomendacji nie da się dokładnie przetestować aplikacji, jednak przykładowe powyższe wywołania działają dobrze.

Implementacja

https://github.com/wolski0420/MusicDBApp

Wnioski

Grafowa baza danych pozwala na bardziej intuicyjne tworzenie relacji między obiektami niż za pomocą tabel. Kierunek relacji ma znaczenie tylko semantyczne, tzn. człowiekowi ma się lepiej czytać taką bazę. Stworzone są biblioteki, które pozwalają na wykonanie mapowania z modelu obiektowego na grafową bazę danych. Problemem może być inny język zapytań - Cypher, jednak po jego krótkiej nauce okazuje się być łatwiejszy niż SQL, a zapytania w nim odnoszą się bezpośrednio do dziedziny grafów.