Proof of concept (z użyciem Neo4j)

Aplikacja z danymi o Formule 1 (2021)

Tomasz Gajda

SPIS TREŚCI

1. Problem i założenia projektu

Celem projektu było stworzenie aplikacji korzystającej z grafowej bazy danych Neo4j, która udostępniłaby prosty interfejs dla użytkownika służący do manipulacji danymi przechowywanymi w bazie.

Stos technologiczny:

- Neo4j
- Node.js
- Express
- Typescript
- React

2. Krótki wstęp teoretyczny

Do stworzenia aplikacji wykorzystałem Node.js z frameworkiem express, bazę grafową Neo4j, Typescript wraz z biblioteką React. Rest API zaimplementowane na serwerze Node pozwala na wywoływanie zapytań do bazy z aplikacji klienta, które pozwalają na manipulację danymi o kierowcach, torach i drużynach Formuły 1 sezonu 2021.

3. Zakres funkcjonalności

Poprzez zestaw metod API pozwala użytkownikowi na dodawanie/usuwanie **kierowców, torów i drużyn** oraz tworzenie wybranych **raportów**. Aplikacja udostępnia poniżej przedstawione endpointy:

DRIVER - Kierowca F1

1. **[POST] Dodawanie kierowcy** - createDriver()

Parametry

- id <<int>>
- name <<string>>
- surname <<string>>
- image_url <<string>>

Typ zwracany

- <<void>>

2. **[DELETE] Usuwanie kierowcy po id** - findByIdAndDelete()

Parametry

- id <<int>>

Typ zwracany

- <<void>>

3. **[GET] Zwróć wszystkich kierowców** - findById()

Parametry

- id <<int>>

Typ zwracany

- <<Driver>>

4. **[GET] Zwróć wszystkich kierowców** - findAll()

Parametry

- <<void>>

Typ zwracany

- <<List<Driver>>>

5. **[PUT] Aktualizuj kierowcę po id** - findByIdAndUpdate()

Parametry

- id <<int>>
- name <<string>>
- surname <<string>>
- image_url <<string>>

Typ zwracany

- <<Driver>>

RACETRACK - Tor F1

1. **[POST] Dodawanie toru** - createRacetrack()

Parametry

- id <<int>>
- name <<string>>
- image_url <<string>>

Typ zwracany

- <<void>>

2. **[DELETE] Usuwanie toru po id** - findByIdAndDelete()

Parametry

- id <<int>>

Typ zwracany

- <<void>>

3. **[GET] Zwróć wszystkie tory** - findById()

Parametry

- id <<int>>

Typ zwracany

- <<Racetrack>>

4. **[GET] Zwróć wszystkich torów** - findAll()

Parametry

- <<void>>

Typ zwracany

- <<List<Racetrack>>>

5. **[PUT] Aktualizuj tor po id** - findByIdAndUpdate()

Parametry

- id <<int>>
- name <<string>>
- image_url <<string>>

Typ zwracany

- <<Racetrack>>

6. **[GET] Znajdź zwycięzcę na torze po ID** - findWinner()

Parametry

- id <<int>>

Typ zwracany

- <<Driver>>

6. **[POST] Ustaw zwycięzcę dla toru po ID** - setWinner()

Parametry

- id kierowcy <<int>>
- id toru <<int>>

Typ zwracany

- <<Racetrack>>

TEAM - Drużyna F1

1. **[POST] Dodawanie drużyny** - createTeam()

Parametry

- id <<int>>
- name <<string>>
- image_url <<string>>

Typ zwracany

- <<void>>

2. **[DELETE] Usuwanie drużyny po id** - findByIdAndDelete()

Parametry

- id <<int>>

Typ zwracany

- <<void>>

3. **[GET] Zwróć wszystkie drużyny** - findById()

Parametry

- id <<int>>

Typ zwracany

- <<Team>>

4. **[GET] Zwróć wszystkich drużyn** - findAll()

Parametry

- <<void>>

Typ zwracany

- <<List<Team>>>

5. **[PUT] Aktualizuj drużynę po id** - findByIdAndUpdate()

Parametry

- id <<int>>
- name <<string>>
- image_url <<string>>

Typ zwracany

- <<Team>>

6. **[GET] Zwróć wszystkich kierowców jeżdżących dla tej drużyny** - findDrivers()

Parametry

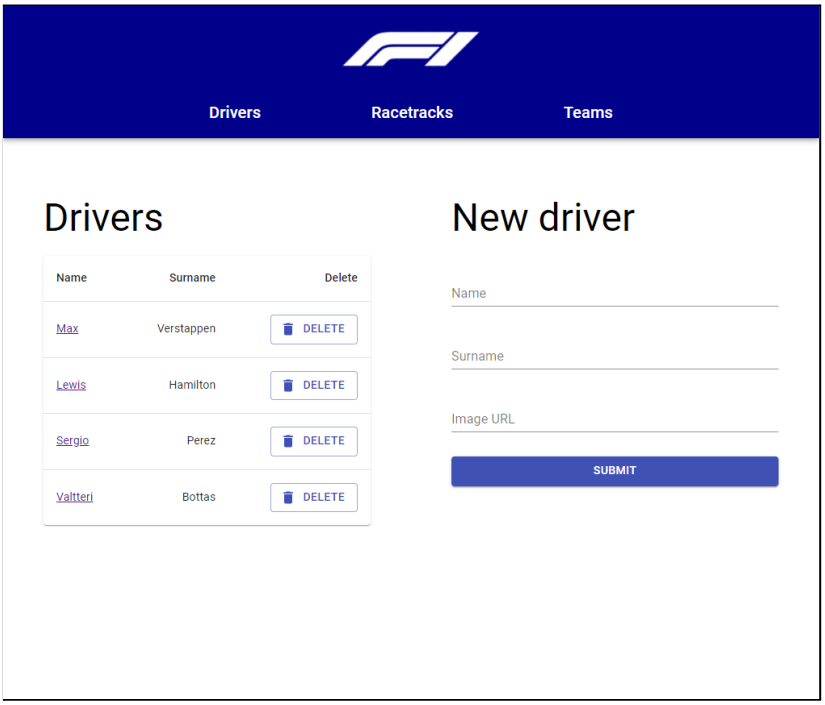
- id <<int>>

Typ zwracany

- <<List<Driver>>>

4. Opis implementacji

Wewnątrz aplikacji dostępne mamy 3 główne podstrony - z kierowcami, torami oraz drużynami. W każdej z nich możemy obejrzeć listę aktualnie znajdujących się w bazie elementów, usunąć każde z nich oraz dodać nowe.

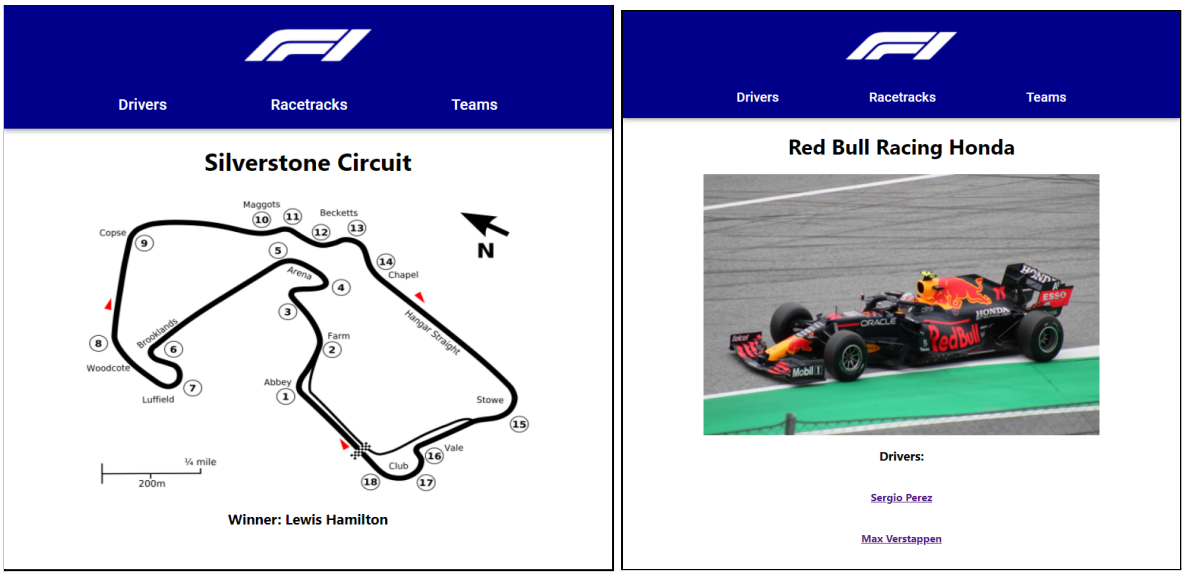


Rys. 1 - Obraz przedstawia interfejs aplikacji demonstracyjnej

W każdej z zakładek możemy również wejść w pojedynczy element i wyświetlić go razem ze zdjęciem.

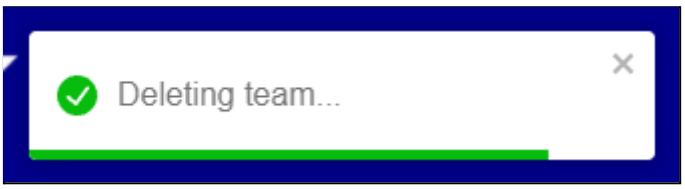
Rys. 2 - Obraz przedstawia ekran pojedynczego elementu z bazy

Pojedyncze elementy z zakładek z torami i drużynami mają również dodatkowe informacje wynikające z relacji w bazie danych. W zakładce z torami możemy zobaczyć kto na nim **zwyciężył**, natomiast w zakładce z drużynami możemy zobaczyć jacy **kierowcy** dla danych drużyn jeżdżą.



Rys. 3 - Ekrany pojedynczego elementu dla toru i drużyny (na dole widać dodatkowe informacje)

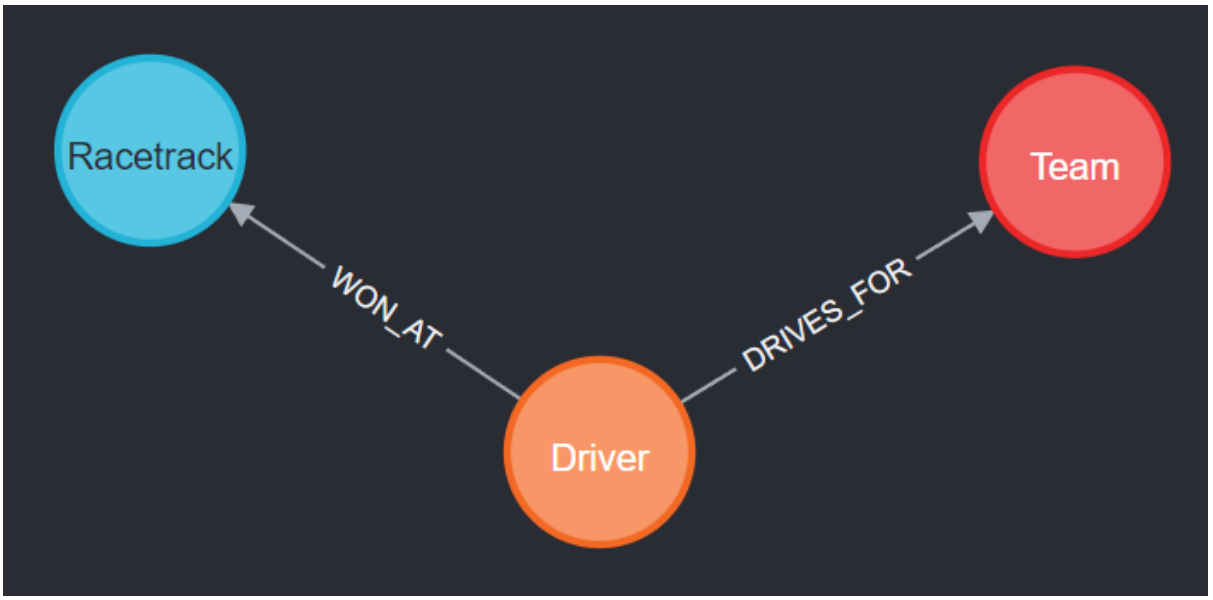
Przy dodawaniu i usuwaniu danych towarzyszą nam notyfikacje potwierdzające udane operacje. Momentami operacje trwają trochę dłużej, dlatego warto chwilę poczekać przed wykonywaniem kolejnych operacji (pomimo tego że akcje wykonują się asynchronicznie).



Rys. 4 - Przykład notyfikacji

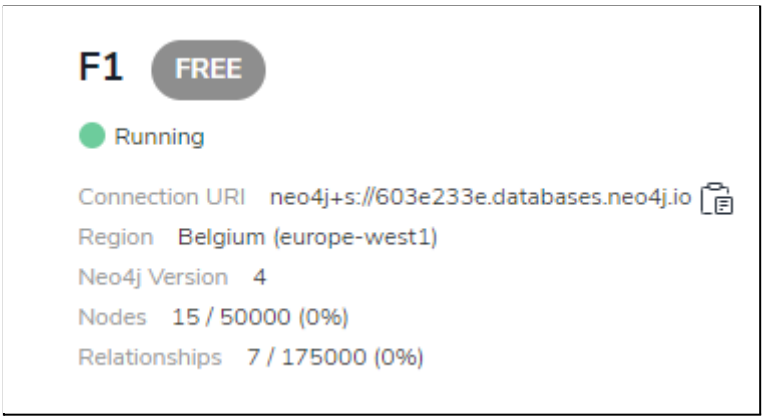
5. Baza danych

Baza danych została stworzona w portalu **Neo4j Aura**. Pracuje na 3 węzłach: **Racetrack, Driver** oraz **Team**. W bazie istnieją relacje przedstawione na poniższym obrazku.



Rys. 5 - Schemat przedstawiający połączenia w grafowej bazie danych

W bazie dodałem pewne początkowe dane - nowe można dodawać poprzez **API** lub też poprzez udostępniony **interfejs**.



Rys. 6 - Informacje na temat bazy stworzonej na portalu Neo4j Aura

6. Wdrożenie

Projekt został wdrożony przy pomocy dwóch technologii - serwer został opublikowany poprzez platformę **Heroku** (przepraszam, lecz miałem dość mało czasu, a z Heroku jest to całkiem szybkie)**,** natomiast aplikacja klienta została udostępniona przez platformę **Netlify** i jest dostępna pod tym linkiem: **https://f1-portal-tg.netlify.app/**

WAŻNE! - Jak wiadomo Heroku idealne nie jest i czasem trzeba chwilkę poczekać na wybudzenie API, tak więc proszę dać mu chwilkę na włączenie i wtedy odświeżyć stronę. Z góry dziękuję!

7. Podsumowanie

Choć jest duża szansa, że baza danych z przedstawionymi założeniami byłaby lepsza, gdyby była stworzona w innej technologii bazodanowej niż bazy grafowe, to myślę, że praca z aplikacją była dla mnie świetnym wprowadzeniem do bazy grafowej **Neo4j** i języka **Cypher**.

8. Literatura

Do stworzenia projektu wykorzystałem głównie **oficjalną dokumentację Neo4j**, oraz pomniejsze strony służące jako poradniki do korzystania z języka **Cypher**:

1. https://neo4j.com/docs/
2. https://neo4j.com/docs/cypher-manual/current/
3. https://expressjs.com/

9. Kod źródłowy

Kod źródłowy dostępny na platformie GitHub pod linkiem:
https://github.com/nerooc/neo4j-poc