**Akademia Górniczo-Hutnicza im. Stanisława Staszica w Krakowie**

Wydział Fizyki i Informatyki Stosowanej

Dokumentacja projektu

Baza rekomendacji książek

Autor:

Rafał Walkowiak

22.12.2022

**1. Założenia projektu:**

Celem projektu jest opracowanie API oraz aplikacji webowej, która będzie wykorzystywała stworzone API. Aplikacja będzie rekomendowała użytkownikowi książki do przeczytania, na podstawie jego wcześniejszych lektur. Technologie, które będą wykorzystane do wykonania projektu:

* Node.js
* React.js
* Neo4j

**2. Zakres funkcjonalności**

Poprzez zestaw punktów końcowych (ang. endpoints) API pozwala na dodawanie, wyświetlanie, modyfikowanie oraz usuwanie danych z bazy.

**2.1. Projekt bazy grafowej**

Baza danych składa się z 4 węzłów, które przechowują określone dane w postaci właściwości:

* User – użytkownik w systemie, który może oceniać książki.
  + \_id – unikalnie id użytkownika
  + name – imię użytkownika
  + surname – nazwisko użytkownika
  + nickname – nickname użytkownika
  + email – email użytkownika
  + password – hasło użytkownika
* Book – książka w systemie, która może zostać oceniona przez użytkowników. Każda książka może mieć przypisanego autora oraz etykiety.
  + \_id – unikalne id książki
  + title – tytuł książki
  + isbn – międzynarodowy znormalizowany numer książki
  + year – rok wydania książki
* Author – autor w systemie, które może pisać książki.
  + \_id – unikalne id autora książek
  + name – imie autora
  + surname – nazwisko autora
  + age – wiek autora
* Tag – etykieta w systemie, którą można przypisać do książek.
  + \_id – unikalne id etykiety
  + name – nazwa etykiety

W bazie istnieją trzy typy relacji pomiędzy węzłami:

* RATED – Użytkownik ocenił książkę. Posiada właściwość:
  + score – ocena książki - wartość od 0 do 10

Diagram

Description automatically generated

* WROTE – Autor napisał książkę

Diagram

Description automatically generated

* TAGGED – Książka ma przypisaną etykietę

Box and whisker chart

Description automatically generated with low confidence

Połączone w ten sposób węzły umożliwiają wyszukiwanie rekomendacji nowych książek, na podstawie wcześniej przeczytanych. Przykładowe połączenie między wszystkimi węzłami dla określonej książki i użytkownika.

Diagram

Description automatically generated

Zapytanie wyszukujące rekomendowane książki dla danego użytkownika:

MATCH (u:User {\_id:'${id}'})-[r:RATED]->(b:Book)

WITH b, toInteger(r.score) AS score

ORDER BY score DESC

LIMIT 3

WITH DISTINCT b

MATCH (b)-[tt:TAGGED]->(t:Tag)

WITH DISTINCT t

MATCH (b:Book)-[:TAGGED]->(t)

WITH DISTINCT b

MATCH (u:User)-[r:RATED]->(b)

WITH b AS book, AVG(toInteger(r.score)) AS avgScore, COUNT(r) AS reviewCount

WHERE NOT (:User {\_id:'${id}'})-[:RATED]->(book)

RETURN book, avgScore, reviewCount

ORDER BY avgScore DESC

Zapytanie znajduje etykiety trzech najlepiej ocenionych przez użytkownika książek. Następnie znajduje wszystkie książki z danymi etykietami oraz sortuje je malejąco na podstawie średnich ocen użytkowników. Z otrzymanych w ten sposób książek usuwane są książki, które już zostały przeczytanie przez danego użytkownika. Oprócz średniej oceny danej książki przez użytkowników zapytanie zwraca również informację, ile razy dana książka została oceniona.

**2.2. Projekt API**

API stworzone w ramach projekty umożliwia wyszukiwanie, dodawania, modyfikowanie oraz usuwanie danych z grafowej bazy danych Neo4j. Do stworzenia API została wykorzystana technologia Node.js w wersji 18.12.1 oraz biblioteki:

* cors (2.8.5)
* dotenv (16.0.3)
* express (4.18.2)
* nanoid (3.0.0)
* neo4j-driver (5.3.0)
* nodemon (2.0.20)

Punkty końcowe dostępne w ramach API:

Zapytania GET:

* user/ - zwraca wszystkich użytkowników
* user/:id – zwraca użytkownika o podanym :id
* user/:id/reviews – zwraca wszystkie książki ocenione przez użytkownika o podanym :id
* user/:id/rated/:value – zwraca wszystkie książki ocenione przez użytkownika na ocenę powyżej wartości :value
* user/:id/recommendations – zwraca książki rekomendowane dla danego użytkownika
* book/ - zwraca wszystkie książki
* book/:id – zwraca książkę o podanym :id
* book/:id/tags – zwraca etykiety dla książki o podanym :id
* tag/ - zwraca wszystkie etykiety
* tag/:id – zwraca etykietę o podanym :id
* author/ - zwraca wszystkich autorów
* author/:id – zwraca autora o podanym adresie :id
* author/:id/books – zwraca książki napisane przez autora o podanym :id

Zapytania POST:

* user/ - dodaje użytkownika do bazy
* user/:idUser/rate – dodaje ocenę książki przez użytkownika o podanym :idUser
* book/ - dodaje książkę do bazy
* book/:bookId/tag/:tagId – dodaje etykietę o podanym :tagId dla książki o podanym :book:Id
* tag/ - dodaje etykietę do bazy
* author/ - dodaje autora do bazy
* author/:idAuthor/book/:idBook – oznacza książkę o podanym idBook jako napisaną przez autora o podanym :idAuthor

Zapytania PUT:

* user/:id – modyfikuje użytkownika o podanym :id
* book/:id – modyfikuje książkę o podanym :id
* tag/:id – modyfikuje etykietę o podanym :id
* author/:id – modyfikuje autora o podanym:id

Zapytania DELETE:

* user/:id – usuwa użytkownika o podanym :id
* user/:idUser/review/:idBook – usuwa recenzję użytkownika o podanym :idUser dla książki o podanym :idBook
* book/:id – usuwa książkę o podanym :id
* book/:idBook/tag/:idTag – usuwa etykietę o podanym :idTag dla książki o podanym :idBook
* tag/:id – usuwa etykietę o podanym :id
* author/:id – usuwa autora o podanym :id
* author/:idAuthor/book/:idBook – usuwa autora o podanym :idAuthor dla książki o podanym :idBook
* data/delete – usuwa wszystkie dane z bazy

**2.3. Projekt Aplikacji Webowej**

Aplikacja webowe służy głównie jako interfejs dostępu do grafowej bazy danych poprzez użycie API. Aplikacja posiada interfejs dla wyświetlania, dodawania oraz usuwania bazy z danych. Nie posiada interfejsu pozwalającego na modyfikację danych. Przy dodawaniu oraz wyświetlaniu danych aplikacja operuje na wartościach id obiektów w bazie.

**7. Bibliografia**

* <https://neo4j.com/developer-blog/discover-auradb-free-goodreads-books-recommendations/>