**CyberBook – dokumentacja projektu wykonywanego w ramach zajęć:   
Przetwarzanie danych w chmurach obliczeniowych**

Mikołaj Marchewa

Wydział Fizyki i informatyki Stosowanej

Akademia Górniczo-Hutnicza im. Stanisława Staszica w Krakowie

15 grudnia 2021

1. **CyberBook, czyli książkowa platforma społecznościowa**

Głównym celem projektu było zrozumienie zasady działania, mechanizmów, przypadków użycia grafowej bazy danych *neo4j.* W efekcie powstała platforma społecznościowa zrzeszająca miłośników książek i pozwalająca na dzielenie się opiniami na ich temat, a także na obserwację aktywności innych użytkowników portalu.

Interfejs graficzny został zahostowany pod adresem:

**https://happybutter.github.io/Cyber-book-frontend/#/**

Rest API dostępne jest pod adresem:  **https://young-gorge-53186.herokuapp.com/**

Przykładowe konto użytkownika:

**Login: admin@admin.com**

**Hasło: admin**

1. **Funkcjonalności**

Po stronie użytkownika portalu:

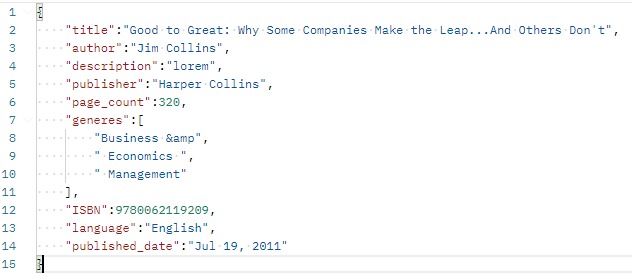
* rejestracja w serwisie – tworzenie nowych kont,
* logowanie do już istniejącego konta,
* przeglądanie najlepiej ocenianych książek,
* przeglądanie recenzji książek,
* wyszukiwanie książki po tytule (lub jego fragmencie),
* obserwacja innych użytkowników portalu,
* wyszukiwanie innych użytkowników portalu (po imieniu/nazwisku lub ich fragmencie)

Po stronie administracji:

* dla administratora powstał specjalny endpoint dostępny pod adresem:

https://young-gorge-53186.herokuapp.com/book/

dzięki któremu z wykorzystaniem zapytania typu POST, administrator może dodać do forum nową książkę. Automatycznie utworzą się (lub nie – jeśli istnieją) węzły będące gatunkami literackimi oraz węzeł autora książki, a także relacje wiążące nowe węzły ze sobą. Struktura ciała zapytania na podstawie przykładowej pozycji:



1. **Projekt logiczny**

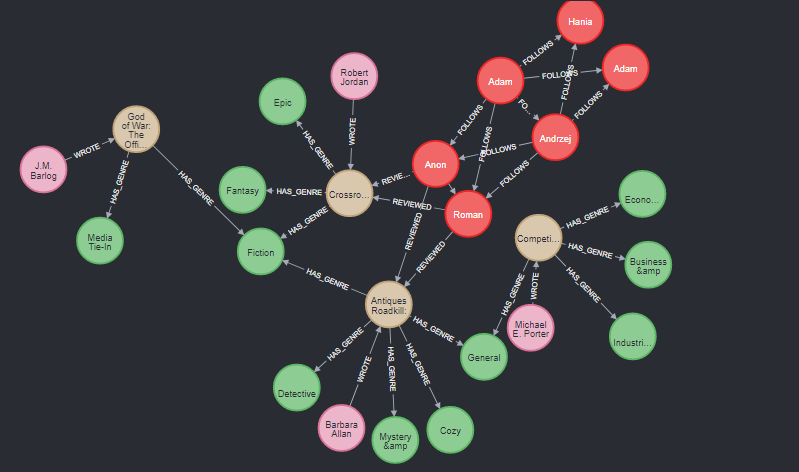
Węzły:

* User - użytkownik serwisu,
  + id,
  + firstName
  + lastName
  + email (pole unikalne)
  + password
* Book – książka,
  + ISBN (pole unikalne),
  + tilte,
  + publisher,
  + description,
  + pageCount,
  + published\_date,
  + language,
  + rate,
  + voteCount
* Author – autor książki,
  + name
* Genre – gatunek literacki,
  + Name

Relacje międzywęzłowe:

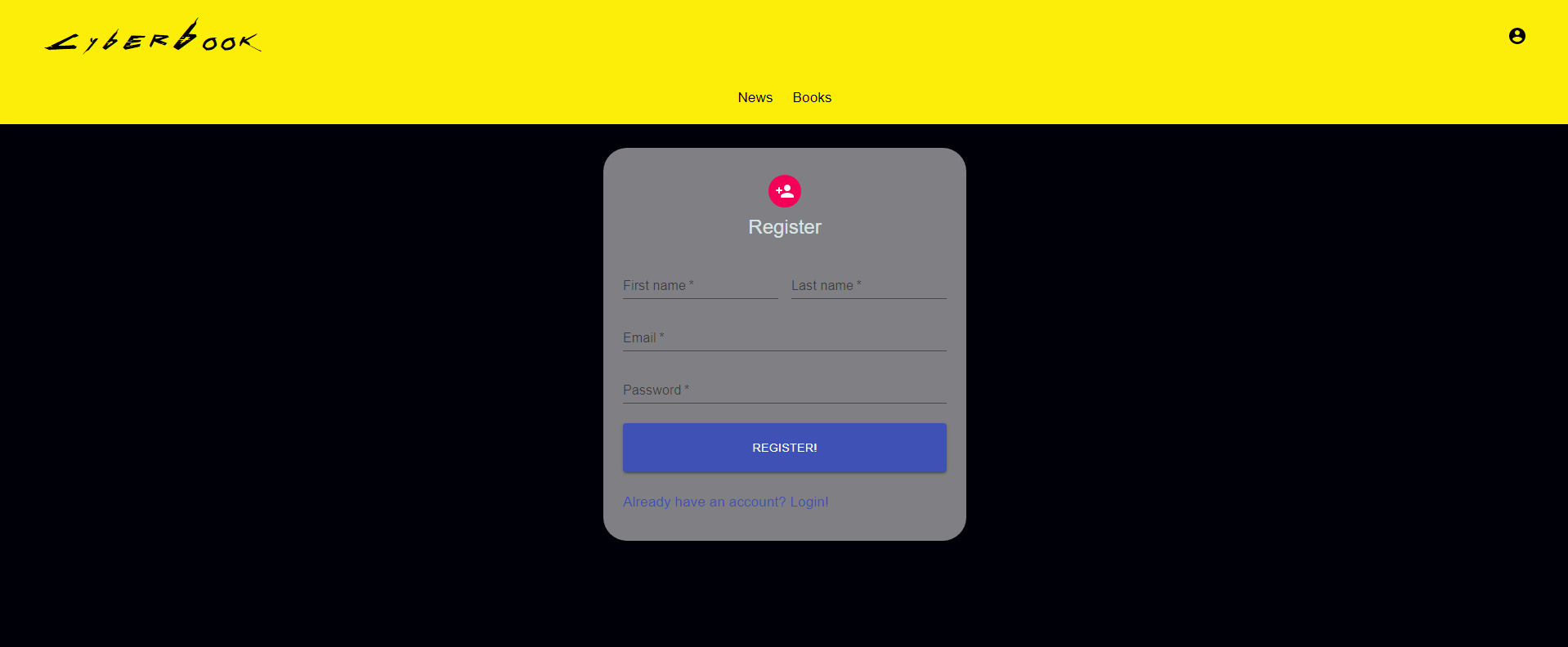
* Book-[:HAS\_GENRE]->Genre
* Author-[:WROTE]->Book
* User-[:FOLLOWS]->User
* User-[:REVIEWED->Book

Baza została wypełniona wybranymi pozycjami ze zbioru książek dostępnego pod adresem: *https://www.kaggle.com/bilalyussef/google-books-dataset*



1. **Projekt funkcjonalny**

* Po wejściu na portal znajdujący się pod adresem:   
  *https://happybutter.github.io/Cyber-book-frontend/#/*  
  użytkownik serwisu otrzymuje możliwość utworzenia nowego konta:



Co realizuje kwerenda:

CREATE (u:User {

id: apoc.create.uuid(),

firstName: $firstName,

lastName: $lastName,

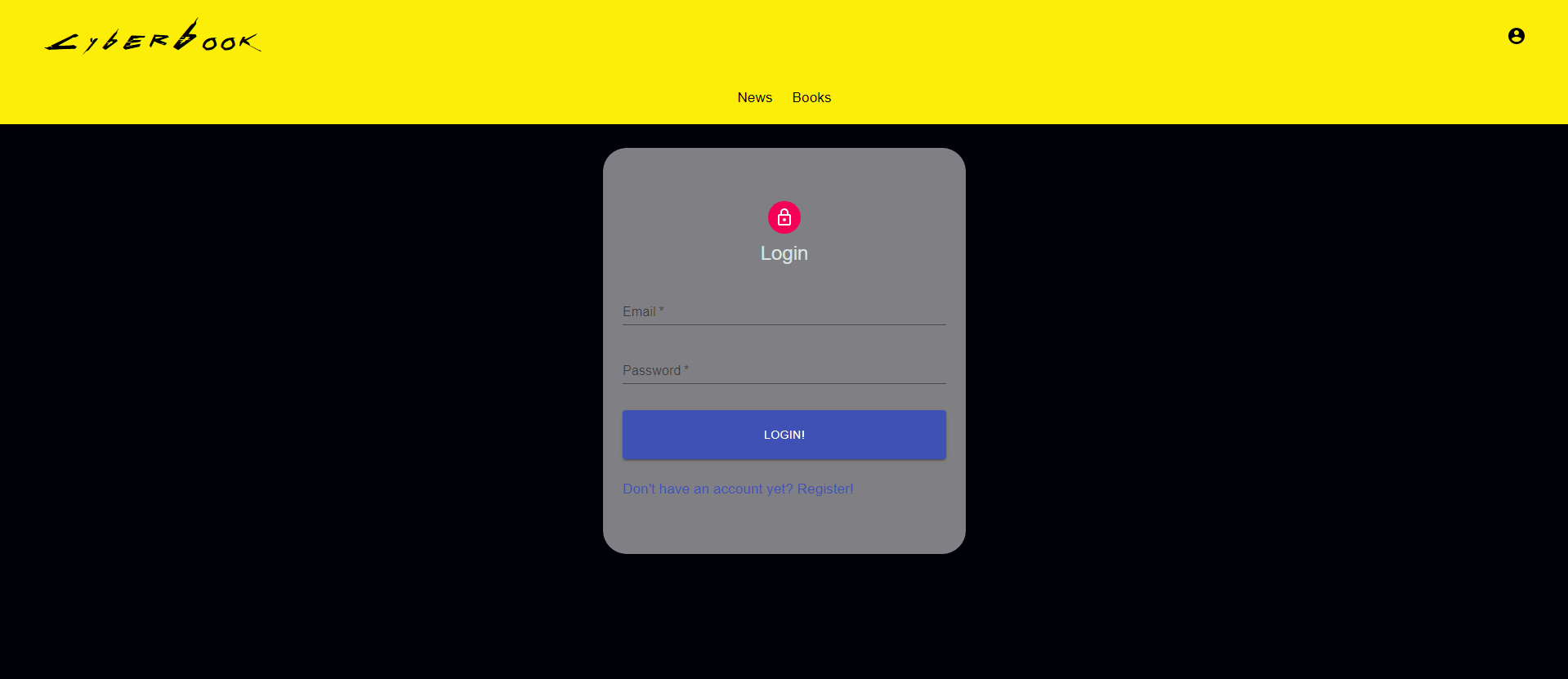
email: $email,

password: $password

})

RETURN u.id, u.firstName, u.lastName, u.email

* Lub możliwość zalogowania się przy użyciu wcześniej utworzonego konta:

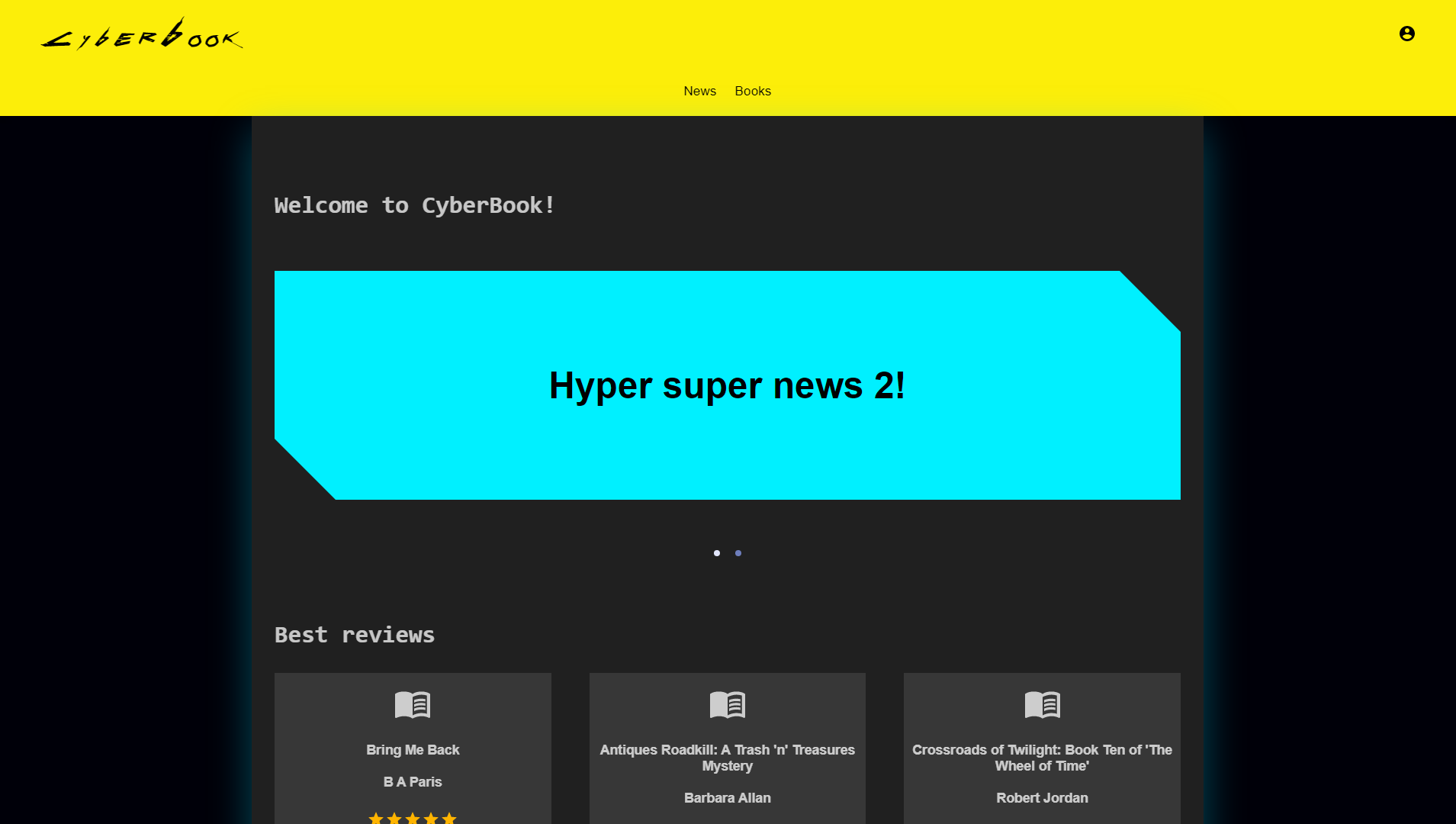
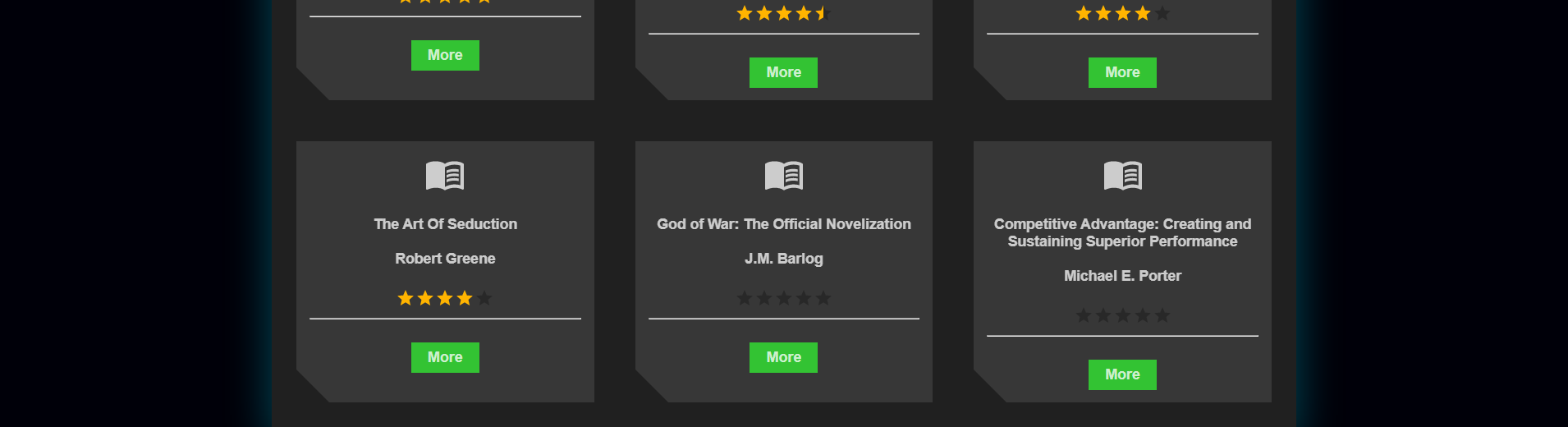


MATCH (u:User)

WHERE u.email=$email AND u.password=$password

RETURN u.id, u.firstName, u.lastName, u.email

* Strona główna aplikacji – zakładka „News”:



Sekcja „Best reviews” wyświetla 6 najlepiej ocenianych książek pośród wszystkich dostępnych w bazie danych. Ich pobieranie realizuje kwerenda:

MATCH (a:Author)-[:WROTE]->(b:Book)

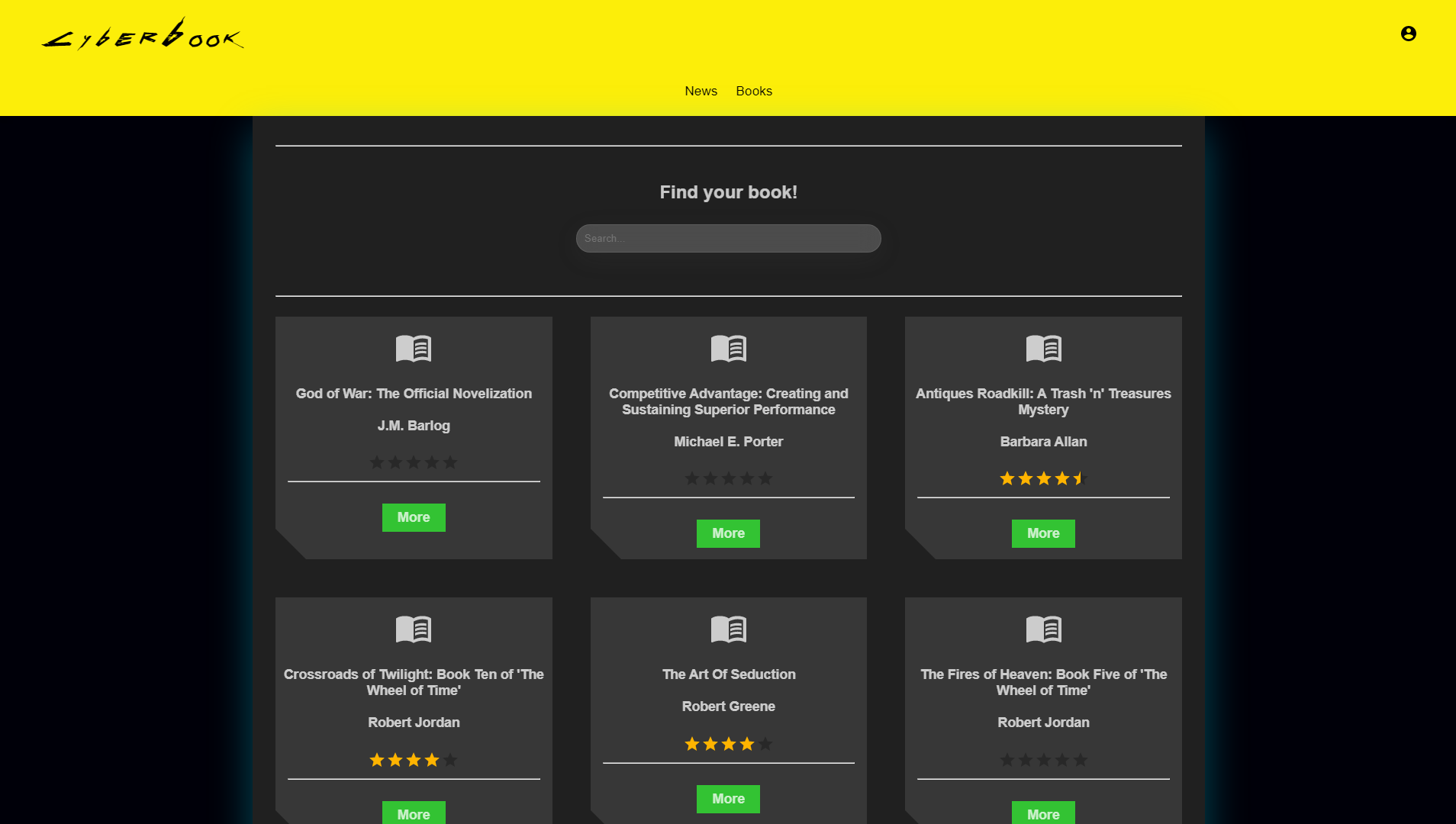
WITH [(b)-[:HAS\_GENRE]->(g) WHERE g:Genre | g.name] AS genres, b, a

ORDER BY b.rate DESC

LIMIT 6

RETURN collect({author: a.name, book: b, genres: genres})

* Zakładka „Books”:



Zakładka “Books” pozwala na przeszukiwanie zasobów bazy i odnalezienie interesującej użytkownika książki:

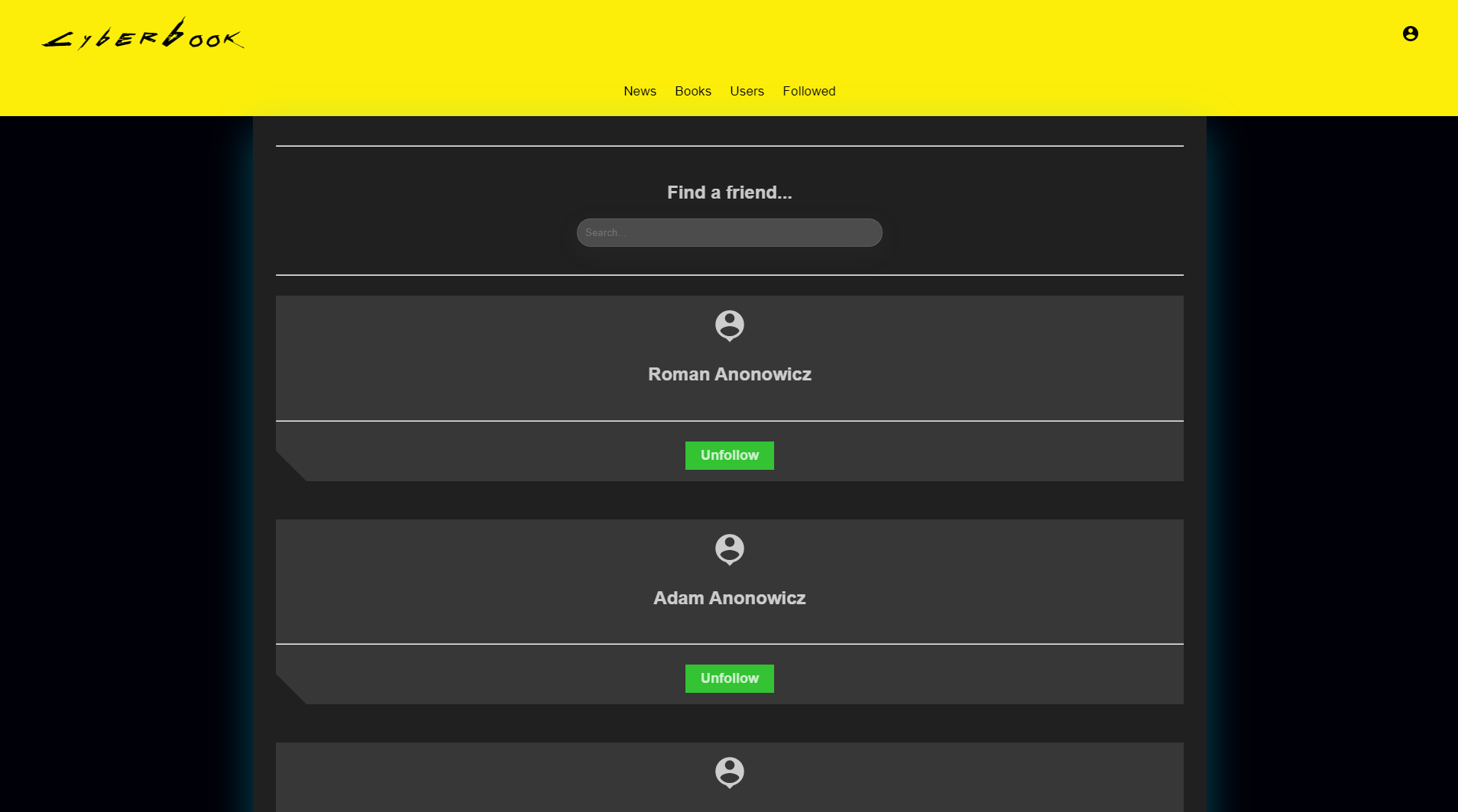
MATCH (a:Author)-[:WROTE]->(b:Book)

WHERE toLower(b.title) CONTAINS toLower($title)

WITH [(b)-[:HAS\_GENRE]->(g) WHERE g:Genre | g.name] AS genres, b, a

RETURN collect({author: a.name, book: b, genres: genres})

* Dla zalogowanych – zakładka „Users”:



Dzięki wyszukiwarce użytkownik może odnaleźć swoich znajomych i zaobserwować bądź przestać obserwować wystawiane przez nich recenzje. Wyszukiwanie realizuje następująca operacja:

MATCH (u:User)

WHERE toLower(u.firstName + u.lastName) CONTAINS trim(toLower($name))

OR toLower(u.lastName + u.firstName) CONTAINS trim(toLower($name))

WITH u

UNWIND u.id as ID

MATCH (u1:User {id: $userId}), (u2:User {id: ID})

WITH exists((u1)-[:FOLLOWS]->(u2)) as isFollowing, u2

WHERE NOT u2.id=$userId

RETURN collect({

isFollowed: isFollowing,

firstName: u2.firstName,

lastName: u2.lastName,

id: u2.id })

Obserwacja użytkownika polega na dodaniu nowej relacji *FOLLOWS:*

MATCH (a:User {id: $userId})

MATCH (b:User {id: $userToFollowId})

MERGE (a)-[r:FOLLOWS]->(b)

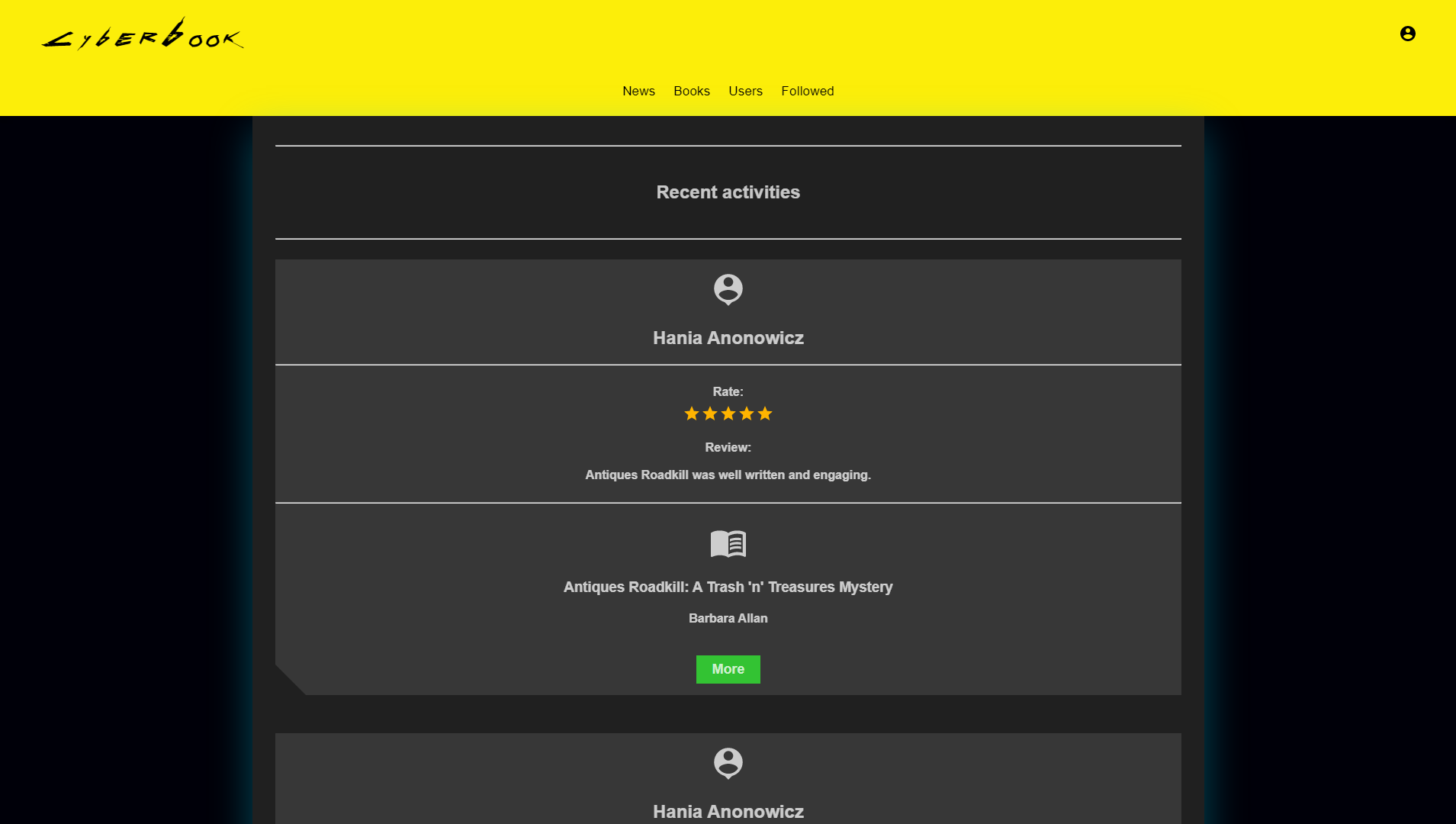
RETURN r

Zakończenie obserwacji, to usunięcie istniejącej krawędzi grafu:

MATCH (a:User {id: $userId})-[r:FOLLOWS]->(b:User {id: $userToUnfollowId})

DELETE r

* Dla zalogowanych – zakładka „Followed”:



W sekcji „Followed” użytkownik znajdzie posortowane chronologicznie (począwszy od najnowszych) recenzje obserwowanych użytkowników. Wyszukanie tych rekordów polega na wykonaniu poniższej kwerendy:

MATCH (u:User {id: $userId})-[:FOLLOWS]->(u2:User)-[r:REVIEWED]->(b:Book)<-[:WROTE]-(a:Author)

WITH collect({firstName: u2.firstName, lastName: u2.lastName, authorId: u2.id}) as author,

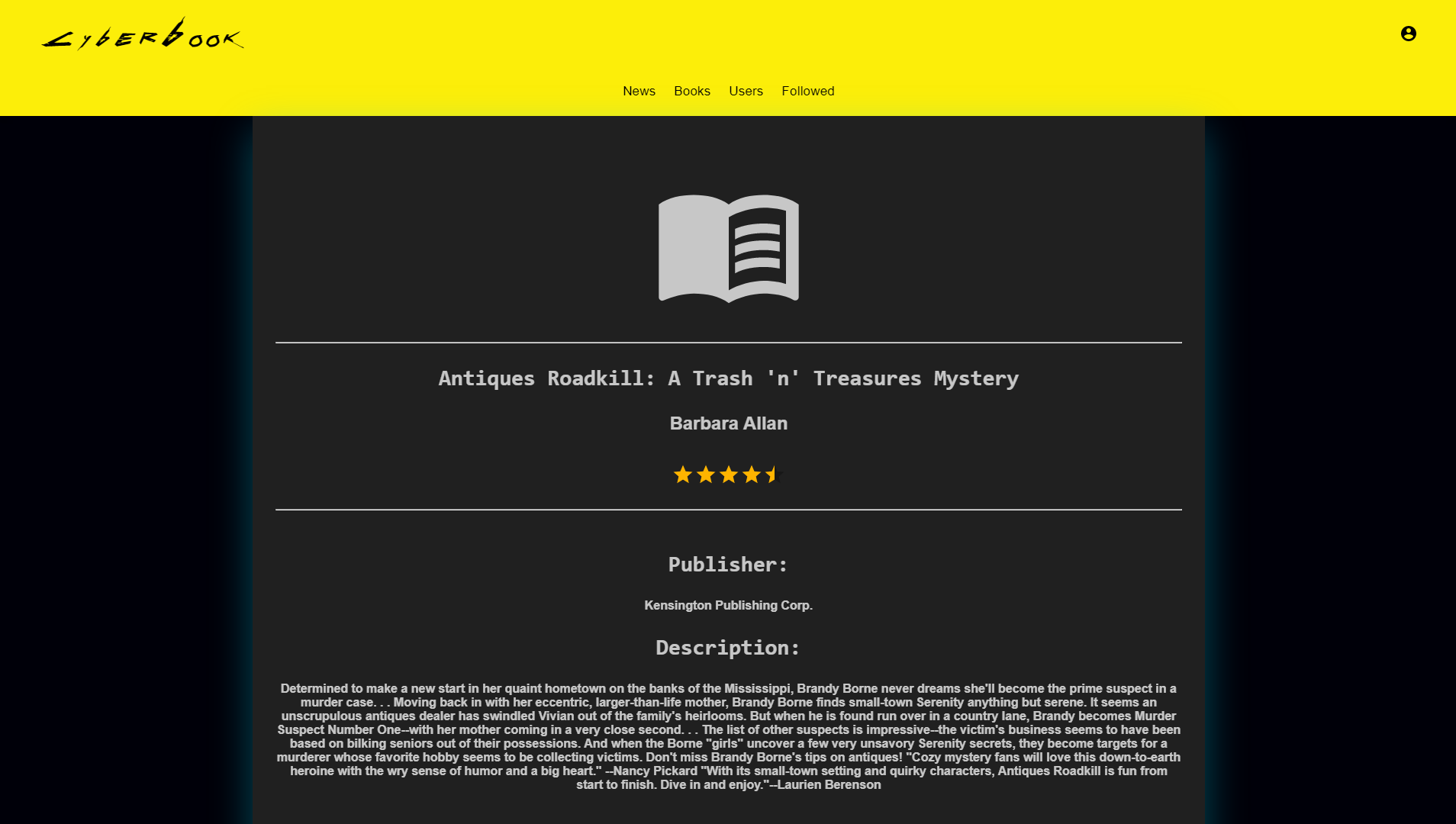
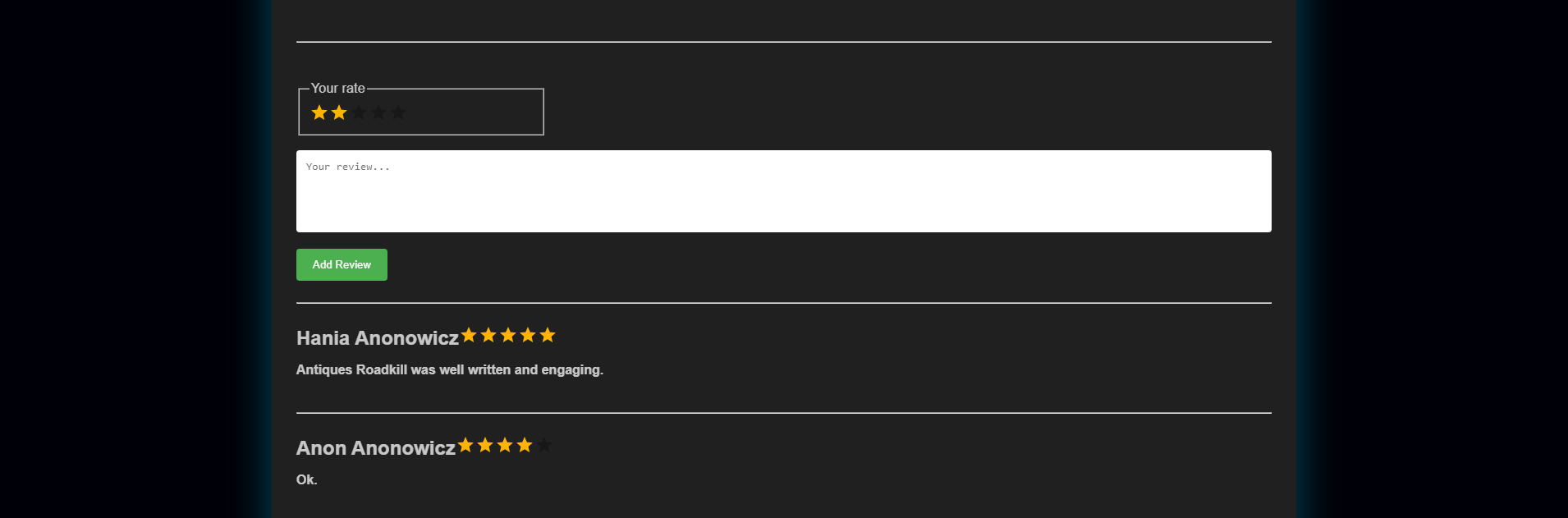
collect({ISBN: b.ISBN, rate: b.rate, title: b.title, author: a.name}) as book, r

RETURN collect({ author: author,

book: book, rate: r.rate,

review: r.review, creationDate: r.creationDate })

* Szczegóły wybranej książki:



Wyświetlenie szczegółów danej książki pozwala na przeczytanie najnowszych opinii, a także napisanie własnej recenzji. W efekcie dodania nowego komentarza ocena średnia książki zostanie ponownie przeliczona. Operacja ta jest zamknięta w obrębie jednej transakcji:

* 1. Dodanie recenzji.

MATCH (a:User {id: $userId})

MATCH (b:Book {ISBN: $isbn})

MERGE (a)-[r:REVIEWED {rate: $rate, review: $review, creationDate: $creationDate}]->(b)

RETURN r

* 1. Przeliczenie wartości średniej oceny

MATCH (b:Book {ISBN: $isbn})<-[r:REVIEWED]-()

WITH avg(r.rate) AS averageRate, count(\*) as countReviews, b

SET b.rate = averageRate, b.voteCount = countReviews

RETURN averageRate, countReviews

1. **Bibliografia**

* <https://neo4j.com/>
* <https://www.npmjs.com/package/neo4j-driver>
* <https://beta.reactjs.org/>
* <https://reactrouter.com/>
* <https://www.kaggle.com/bilalyussef/google-books-dataset>
* <https://pages.github.com/>