Dokumentacja projektu “Bookbase”

Bazy danych 2019/2020

Autorzy:

Wojciech Ankus, Kacper Rosiak, Tomasz Kozyra

Funkcjonalności aplikacji:

Aplikacja służy do przechowywania **informacji o książkach i autorach**, oraz **wystawiania opinii**. Po zalogowaniu użytkownik może wyświetlić listę książek, oraz autorów. Klikając na książkę/autora wyświetlają się szczegółowe informacje. W przypadku książek, użytkownik ma możliwość dodania oceny przy użyciu formularza.

Zalogowany użytkownik ma ponadto możliwość **dodawania** książki / autora / kategorii, oraz **edycji** już istniejących książek / autorów..

Aplikacja umożliwia **sortowanie zawartości**:

- Alfabetycznie
- Po kategorii
- Po ocenie
- Po dacie

Wykorzystane technologie:

Backend:

- Java
- Spring (+ Spring Boot, Spring Security)

Frontend:

- Vaadin

Baza danych:

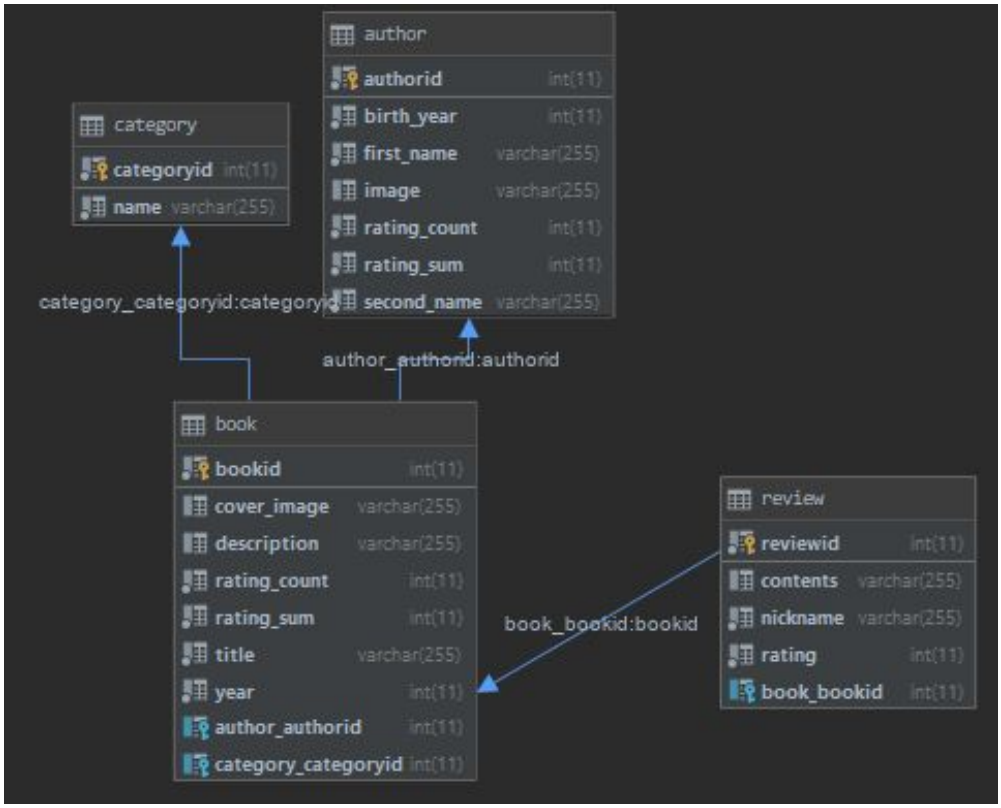
- Spring Data JPA (Hibernate)
- MySql

Uruchamianie:

Demo aplikacji działa pod adresem:
http://bookbase-env.eba-cibi9f5f.eu-central-1.elasticbeanstalk.com/

W celu uzyskania dostępu do aplikacji **należy się zalogować**.

Schemat bazy danych:



Opis tabel w bazie danych:

- **Book** - zawiera informacje o książkach i ich ocenach. Przechowywana jest suma ocen oraz ich ilość - pozwala to na łatwe wyliczanie średniej oceny dla książki. Posiada klucze obce do Autora oraz Kategorii.
- **Author** - zawiera informacje o autorach oraz ocenach książek napisanych przez autora. Przechowywana jest suma oraz ilość ocen dla wszystkich książek danego autora - pozwala to na łatwe wyliczanie średniej oceny autora gdy napisał on wiele książek.
- **Category** - zawiera informacje o kategoriach książek.
- **Review** - zawiera informacje o ocenach. Posiada klucz obcy do Książki w celu identyfikacji której książki dotyczy dana ocena.

Opis klas w aplikacji:

- **BookbaseApplication.java**

Klasa zawierająca maina aplikacji.

- **com.bookbase.backend**
  - **com.bookbase.backend.entity**

Pakiet zawiera klasy odpowiadające za strukturę bazy danych, a więc obiekty w niej przechowywane: autora, książkę, kategorię oraz opinię.

  - **com.bookbase.backend.repository**

Pakiet zawiera interfejsy Spring Data JPA wykorzystywane przez aplikację do wyciągania informacji z bazy danych na której pracuje aplikacja. Repozytoria udostępniają metody do obsługi bazy danych takie jak m. in. :

- ➢ findAll() - pobranie wszystkich książek / autorów / kategorii / ocen
- ➢ save(entity) - zapisanie rekordu w bazie danych
- ➢ delete(entity) - usunięcie rekordu z bazy danych

Można również w łatwy sposób zdefiniować w nich własne metody, które są tłumaczone przez aplikację na zapytania SQL.

  - **com.bookbase.backend.service**

Pakiet zawiera klasy serwisów, które służą do manipulowania danymi z bazy i zwracania ich do widoków (w widokach są przechowywane instancje serwisów). Serwisy są pewnego rodzaju pomostem pomiędzy repozytoriami a aplikacją.

- **com.bookbase.security**

Pakiet zawiera klasy odpowiedzialne za logowanie się do aplikacji, oraz blokadę adresów dopóki się nie zalogujemy.

- **com.bookbase.ui.views**

Pakiet zawiera wszystkie klasy związane z frontendem.

  - **com.bookbase.ui.views.authors**

Klasy związane z widokiem autorów.

AuthorsView.java - główny widok autorów, posiada obiekty klas:

- ➔ **AuthorDetails.java** - szczegóły dotyczące danego autora wysuwają się po kliknięciu w niego na liście

- ➔ **AuthorForm.java** - formularz do dodawania / edytowania autora, który wysuwa się w miejsce detali

- **com.bookbase.ui.views.books**

Pakiet zawiera klasy związane z widokiem książek.
BooksView.java - główny widok książek, posiada obiekty klas:

- ➔ **BookDetails.java** - szczegóły dotyczące danej książki wysuwają się po kliknięciu w nią na liście
- ➔ **BookForm.java** - formularz do dodawania / edytowania książki, który wysuwa się w miejsce detali
- ➔ **BookReview.java** - formularz służący do dodawania recenzji do danej książki, jest dostępny z poziomu detali książki
- ➔ **CategoryForm.java** - formularz służący do dodawania nowej kategorii książek

- **com.bookbase.ui.views.home**

Pakiet zawiera klasę **HomeView.java** odpowiadającą za widok “podstawowy”, czyli ten dostępny po zalogowaniu zanim przejdziemy do książek bądź autorów.

- **com.bookbase.ui.views.login**

Pakiet zawiera klasę **LoginView.java** odpowiadającą za widok przed zalogowaniem do aplikacji.

- **MainLayout.java**

Klasa odpowiada za nawigację pomiędzy elementami aplikacji, czyli pasek nawigacyjny zawierający przyciski z odnośnikami do widoku głównego, autorów i książek

Przykładowe fragmenty kodu:

- W klasie **BookService.java**
    - Wyciągnięcie wszystkich książek z bazy

```
public List<Book> findAll(){
   return bookRepository.findAll();
}
```

    - Wyciągnięcie wszystkich książek z danej kategorii

```
public List<Book> findAll(Category category) {
   List<Book> result = new ArrayList<>();
   findAll().forEach(book -> {
       if (book.getCategory().equals(category))
           result.add(book);
   });
   return result;
}
```

    - Wyciągnięcie wszystkich książek danego autora

```
public List<Book> findAll(Author author) {
   List<Book> result = new ArrayList<>();
   findAll().forEach(book -> {
       if (book.getAuthor().equals(author))
           result.add(book);
   });
   return result;
}
```

- Wyciągnięcie wszystkich książek ze Stringiem “filter” w tytule

```
public List<Book> findAll(String filter) {
   List<Book> books = this.findAll();
   List<Book> result = new ArrayList<>();
   for (Book book : books)
       if(book.getTitle().toLowerCase().contains(filter.toLowerCase()))
           result.add(book);
   return result;
}
```

- Przykładowe dane wstawiane gdy baza jest pusta

```
@PostConstruct
public void populateTestData() {
   if (categoryRepository.count() == 0){
       categoryRepository.saveAll(
               Stream.of("Science Fiction", "Horror", "Nonfiction")
               .map(Category::new).collect(Collectors.toList()));
   }
   if (bookRepository.count() == 0) {
       Random r = new Random(0);
       List<Category> categories = categoryRepository.findAll();
       bookRepository.saveAll(
               Stream.of("The institute", "It", "Recursion", "Los Pollos Hermanos",
"Educated", "Bad  blood")
                       .map(name -> {
                               Book book = new Book();
                               book.setTitle(name);
                               book.setYear(r.nextInt()%20 + 2001);
                               return book;
                       })
                       .collect(Collectors.toList()));
   }
```

```
if (authorRepository.count() == 0) {
    List<Book> books = bookRepository.findAll();
    List<Category> categories = categoryRepository.findAll();
    List<Author> authors = new ArrayList<>();

    Author author1 = new Author("Stephen", "King", 1947);
    author1.setImage("https://i.imgur.com/yTMM4ay.jpg");
    books.get(0).setAuthor(author1);
    books.get(1).setAuthor(author1);
    books.get(0).setCategory(categoryRepository.findCategoryByName("Horror"));
    books.get(1).setCategory(categoryRepository.findCategoryByName("Horror"));
    books.get(0).setCoverImage("https://i.imgur.com/j6jkRgu.jpg");
    books.get(1).setCoverImage("https://i.imgur.com/ztTP2CG.jpg");
    authors.add(author1);

    Author author2 = new Author("Blake", "Crouch", 1954);

    books.get(2).setAuthor(author2);
    books.get(2).setCategory(categoryRepository.findCategoryByName("Science Fiction"));
    books.get(2).setCoverImage("https://i.imgur.com/mmpsHUM.jpg");
    authors.add(author2);

    Author author3 = new Author("Tara", "Westover", 1986);

    books.get(4).setAuthor(author3);
    books.get(4).setCategory(categoryRepository.findCategoryByName("Nonfiction"));
    books.get(4).setCoverImage("https://i.imgur.com/8dxTcI9.jpg");
    authors.add(author3);

    Author author4 = new Author("John", "Carreyrou", 1983);

    books.get(5).setAuthor(author4);
    books.get(5).setCategory(categoryRepository.findCategoryByName("Nonfiction"));
    books.get(5).setCoverImage("https://i.imgur.com/ouFpZM5.jpg");
    authors.add(author4);

    Author author5 = new Author("John", "Wick", 1954);
```

```
        authors.add(author5);
        books.get(3).setAuthor(author5);
        books.get(3).setCategory(categoryRepository.findCategoryByName("Science Fiction"));
        books.get(3).setCoverImage("https://i.imgur.com/64juBze.png");

        authorRepository.saveAll(authors);
        bookRepository.saveAll(books);
    }
}
```