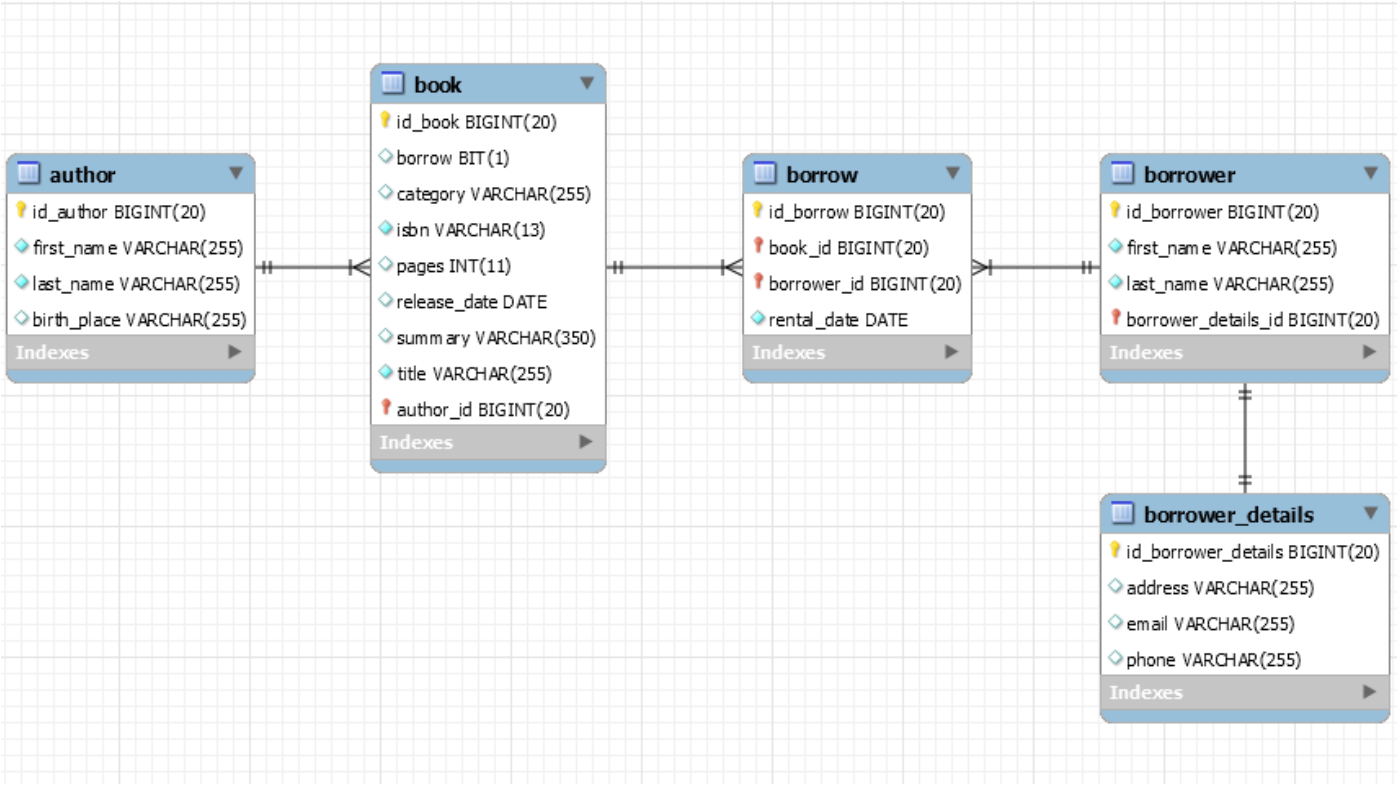
Projekt Biblioteka



Opis działania:

Na ekranie startowym wyświetlamy listę wszystkich książek w bibliotece. Każdy wiersz musi posiadać następujące wartości: *tytuł, imię i nazwisko autora, isbn, kategoria, do której należy, data wydania książki oraz w przypadku, gdy książka jest wypożyczona – nazwę wypożyczającego*.

Każdy rekord posiada *input* typu *radio*, który umożliwia wykonanie poniższych akcji na rekordzie:

- Dodanie książki,
- Edycja wybranej książki
- Usunięcie wybranej książki
- Pokazanie szczegółów książki

Pod tabelą będą znajdować się przyciski z akcjami – *Add, Edit , Delete, Show*

Ekran dodawania książki zawiera formularz, odpowiednie inputy są walidowane. Zawiera 2 przyciski: *Add* i *Cancel*.

Ekran edycji książki analogicznie do ekranu dodawania książki. Wyświetla w poszczególnych inputach obecne informacje o książce.

Ekran szczegółów książki wyświetla szczegóły wybranej książki wraz z listą użytkowników. Zawiera możliwość wypożyczenia książki: wybranie użytkownika + wciśnięcie przycisku *Borrow* -> dana książka zostaje przydzielona do wypożyczającego. Przycisk *Borrow* staje się nie widoczny i na jego miejsce pojawia się przycisk *Return* umożliwiający zwrócenie książki. Na powyższym ekranie zawarty jest również buton *Back* umożliwiający powrót do strony głównej.

Należy utworzyć nowy projekt Maven z 4-ma modułami:

1. *domain* – moduł zawierający modele, własne wyjątki, dtosy,
2. *persistence* – moduł odpowiedzialny za odczyt i zapis do/z bazy danych
3. *service* – moduł wraz z logiką biznesową

4. *web* – moduł odpowiedzialny za widoki i pobranie/wysłanie danych

Należy utworzyć encje mapowane na tabele ze zdjęcia

Należy utworzyć odpowiednie obiekty DTO (Data Transfer Object)

Moduł persistence:

Każdy model zawiera osobną klasę w *persistence*, w której znajdują się metody CRUD (create, read, update, delete) z bazy danych. Tworzymy tylko te metody które potrzebujemy.

Do łączenia się z bazą danych używamy frameworka *Hibernate* (Tworzymy instancję obiektu *SessionFactory*)

Konwencja nazewnictwa:

Nazwa modelu + „Repository" lub „Dao" np.: BookRepository, BookDao

Do powyższych klas należy utworzyć interfejsy z przedrostkiem „I" np.: IBookRepository.

Moduł service:

Moduł będzie łączył się z modułem *persistence* w celu pobrania lub zapisu danych oraz będzie zawierał dodatkową logikę:

1. Uniemożliwienie usunięcia wypożyczonej książki
2. Uniemożliwienie wypożyczenia książki, która w danym momencie już jest wypożyczona
3. Obsługa wyjątków (można je obsłużyć też wyżej np.: w kontrolerach)

Konwencja nazewnictwa:

Nazwa modelu + „Service np.: BookService

Do powyższych klas należy utworzyć interfejsy z przedrostkiem „I" np.: IBookService.

Moduł web:

Moduł łączy się z modułem *service*.

Zawiera *Serwlety* oraz pliki *jsp*.

Walidacje:

Walidacje formularzy realizujemy za pomocą **Hibernate Validator** (należy dodać odpowiednią zależność w *pom.xml*).

Logi:

Do logowania błędów używamy **logback** (należy dodać odpowiednią zależność w *pom.xml*)

Logi należy wypisywać na standardowe wyjście, dodatkowo można zapisywać do pliku.

Należy pamiętać o prawidłowym ustawianiu poziomów (dla błędów – *error*, do śledzenia wykona n w kodzie – *info*).

Mocki:

Dane o czytelnikach i autorach należy zmokować. Mocki (zwykłe inserty do poszczególnych tabel) umieszczamy w skrypcie o nazwie *import.sql*.

Testy (extra):

Do utworzonej aplikacji można napisać testy wykorzystując **Junit /TestNG, Mockito, AsserJ, Harmcrest**, w razie potrzeb **PowerMockito:**

- Aplikacja powinna zawierać przynajmniej jeden test parametryzowany.
- Należy przetestować wyjątki.
- Do testów stosujemy szkielet: *given, when, then* lub *arrange act assert*