Library Manager to aplikacja do zarządzania zasobami biblioteki. Logika aplikacji bazuje na 4 encjach:

Origin: - zasób biblioteki, posiada wszelkie informacje typu: tytuł, autor, rok wydania, numer ISBN itd. Jest bazą dla książki (egzemplarza). Generalnie powinien być jeden Origin i wiele książek (czyli jego egzemplarzy, choć gdyby było kilka wydań/wersji to mogłoby być kilka Origin dla tego samego tytułu).

Book – egzemplarz danego Origin. Np. jest jeden Origin książki „Hobbit” i wiele książek (egzemplarzy), których część jest wypożyczona, cześć dostępna w bibliotece itp.

User – użytkownik biblioteki

Rental – wypożyczenie książki przez użytkownika. Jeden użytkownik może posiadać wiele wypożyczeń.

Pomiędzy encjami istnieją odpowiednie relacje one-to-many itd. Do zarządzania zasobami w bazie danych wykorzystano JPA/Hibernate.

**Back-end: podsumowanie:**

1. Co najmniej 20 End-pointów rest typu get, post put delete – TAK
2. Co najmniej 2 zewnętrzne API – jest jedno (Goodreads) do sprawdzania oceny książek (po kodzie isbn) – w kontrolerze Origin
3. Scheduler – TAK (wysyła mail co tydzień do wszystkich użytkowników, którzy mają wypożyczone książki)
4. Co najmniej 10 zapisów do bazy danych – TAK, requesty typu post, put, patch
5. Testy – TAK, pokrycie na poziomie 80% kodu + wszystkie metody kontrolerów.
6. 2 wzorce projektowe – jeden wzorzec (Singleton) w warstwie Front-end (serwisy)
7. Warstwa widoku – jest w podstawowym zakresie, zabrakło czasu na dodanie funkcjonalności, które są już w back-endzie (np. sprawdzanie ratingu w Goodreads, zliczanie ilości dostępnych książek wg. danego statusu itp.).

**Front-end: podsumowanie:**

1. Z braku czasu warstwa front-end funkcjonuje w ograniczonym zakresie.
2. Istnieją osobne zakładki dla każdej z 4 kluczowych encji (Origins, Books, Users, Rentals).
3. Zakładka Origins jest najbardziej rozbudowana, posiada opcje dodawania, edycji i kasowania Originów (formularz pojawia się po kliknięciu w rekord w tabeli lub przycisk ‘Add new’). Podobne funkcjonalności należałby dodać w pozostałych zakładkach (nie starczyło czasu) oraz specyficzne funkcjonalności jak wypożyczenie książki, zamknięcie wypożyczenia, sprawdzenie ratingu itd.

**Zapełnianie bazy przykładowymi rekordami: (poniżej przykładowe polecenia dla Postmana)**

INSERT INTO ORIGINS(id, author, isbn, published\_year, title)

VALUES(1, "J. R. R. Tolkien", "0345339681", 1937, "The Hobbit");

INSERT INTO ORIGINS(id, author, isbn, published\_year, title)

VALUES(2, "Antoine de Saint-Exupery", "0756751896", 1943, "The Little Prince");

INSERT INTO ORIGINS(id, author, isbn, published\_year, title)

VALUES(3, "Agatha Christie", "9780062073488", 1939, "And Then There Were None");

INSERT INTO ORIGINS(id, author, isbn, published\_year, title)

VALUES(4, "Dan Brown", "0385504209", 2003, "The Da Vinci Code");

INSERT INTO BOOKS (id, book\_status, origin\_id)

VALUES(5, 0, 1);

INSERT INTO BOOKS (id, book\_status, origin\_id)

VALUES(6, 1, 1);

INSERT INTO BOOKS (id, book\_status, origin\_id)

VALUES(7, 2, 2);

INSERT INTO BOOKS (id, book\_status, origin\_id)

VALUES(8, 0, 3);

INSERT INTO BOOKS (id, book\_status, origin\_id)

VALUES(9, 0, 3);

INSERT INTO BOOKS (id, book\_status, origin\_id)

VALUES(10, 1, 4);

# korzystając z postmana user creation\_date uzupełnia się automatycznie

INSERT INTO USERS (id, email, first\_name, last\_name, user\_creation\_date)

VALUES(11, "bobs@wp.pl", "Bob", "Smith", null);

INSERT INTO USERS (id, email, first\_name, last\_name, user\_creation\_date)

VALUES(12, "andys@gmail.com", "Andy", "White", null);

INSERT INTO USERS (id, email, first\_name, last\_name, user\_creation\_date)

VALUES(13, "sarahs@gmail.com", "Sarah", "Connor", null);

# korzystając z postmana (wypożyczenie / zwrot) daty uzupełniają się automatycznie

INSERT INTO RENTALS (id, active, rental\_date, return\_date, book\_id, user\_id)

VALUES(14, 1, null, null, 6, 11);

INSERT INTO RENTALS (id, active, rental\_date, return\_date, book\_id, user\_id)

VALUES(15, 1, null, null, 10, 13);

commit;

**POSTMAN:**

**Create origin**: http://localhost:8080/v1/origins

Body:

{

    "title": "The Hobbit",

    "author": "J. R. R. Tolkien",

    "publishedYear": "1937",

    "isbn": "0345339681"

}

**Create** **book**: <http://localhost:8080/v1/books/createBook?originId=1>

**Create** **user**: <http://localhost:8080/v1/users>

Body:

{

    "firstName": "Tom",

    "lastName": "Testowy",

    "email": "tomas@gmail.com"

}

**create Rental where userId = 3 & bookId = 2:** <http://localhost:8080/v1/rentals/createRental>

Body:

{

    "userId": "3",

    "bookId": "2"

}

**get Origin of origin ID** **= 1**: <http://localhost:8080/v1/origins/1>

**get Book where book ID = 2**: <http://localhost:8080/v1/books/2>

**get User with ID = 3**: <http://localhost:8080/v1/users/3>

**get Rental where rental ID = 4**: <http://localhost:8080/v1/rentals/4>

**set book status LOST where book ID = 2**: <http://localhost:8080/v1/books/2/lost>

**close Rental where rental ID = 4**: <http://localhost:8080/v1/rentals/close/4>

**get Books by Status and Title**: [http://localhost:8080/v1/books/{status}/{title}](http://localhost:8080/v1/books/%7bstatus%7d/%7btitle%7d)

**count Books By Status And Title**: [http://localhost:8080/v1/books/{status}/{title}/count](http://localhost:8080/v1/books/%7bstatus%7d/%7btitle%7d/count)

**get Goodreads rating**:

<https://www.goodreads.com/book/review_counts.json?key=abgn5L0yAfhLtm5lH2w7lw&isbns=0596009208>

key**:** abgn5L0yAfhLtm5lH2w7lw

isbns: 0596009208

**get all books:** <http://localhost:8080/v1/books>

**Są też inne typu delete, put itd. których tu już nie wymieniano.**