



UNIVERSITY OF PISA

Large Scale and Multi-Structured Databases

Year 2019/20

BookRater

A proposed extension to BibliOS, powered by Neo4j

GERARDO ALVARO

MARCO BONGIOVANNI

RICCARDO POLINI

GIULIO SILVESTRI

INDEX

1. INTRODUCTION.....	1
2. REQUIREMENTS ANALYSIS.....	2
2.1. Application Actors.....	2
2.2. Functional and Non-Functional Requirements.....	2
3. UML DIAGRAMS.....	3
3.1. Use-Case Diagram.....	3
3.2. Class Diagram.....	4
4. DATABASE ORGANIZATION.....	5
4.1. Graph Model.....	5
5. SOFTWARE ARCHITECTURE.....	6
5.1. Repository Structure.....	6
6. INSTRUCTION MANUAL.....	7
7. ON-GRAPH OPERATIONS.....	15
8. CONCLUSIONS.....	16

1. INTRODUCTION

BookRater will be a module proposed as an extension of *BibliOS*, our application for Task 1, which will make use of a graph database in order to implement a sort of social portal for book reviews.

For practical reasons, it will be presented as a stand-alone application in order to highlight the new functionalities.

Users will browse a list of unread books and “mark-as-read” the ones they wish. After doing so, they will have to rate the given book with 1-5 stars.

Users will also be able to apply “tags” to specific books. Tags are metadata strings used to identify in some way the given book.

A Suggestions page is available, in which a user can browse a list of recommended books. The list is computed by taking into account the user’s preferences and tags in common between books.

If a book is desired, a user can add it to his personal wish-list, so that he can read it in the future.

Admins have the power of adding new books to the Catalogue and removing existing books.

2. REQUIREMENTS ANALYSIS

2.1. Application Actors

The actors of the application are the *User* and the *Admin*.

The first one is the main user of the application.

The second, has additional responsibilities available only to an administrator of the application.

2.2. Functional and Non-Functional Requirements

The login phase of Users/Admins was managed by JPA in the previous Task and will stay the same. It will not be marked on these requirements, nor on the use-case diagram.

The *functional* requirements of this application, divided with respect to the two actors, are as follows:

- A User can browse the catalogue of books in the library.
- A User can “mark-as-read” a book and rate it.
- A User can view a *book suggestions* page, based on user ratings & wish list.
- A User can add a *tag* to a book. A tag is metadata that identifies in some way the books.
- A User can add books to a personal *wish-list*, used by the application to suggest new books.
- The Admin can add/remove books.
- The Admin can browse the catalogue.
- The Admin can browse the tags & statistics of a specific book.

The *non-functional* requirements of the application are:

- The application’s interface must be user-friendly.
- The application must have a low response time.
- The application will store information in a Graph Database (Neo4j).
- The application must guarantee data consistency.
- The application must be reliable: no system crashes, exceptions are handled etc.

3. UML DIAGRAMS

3.1. Use-Case Diagram

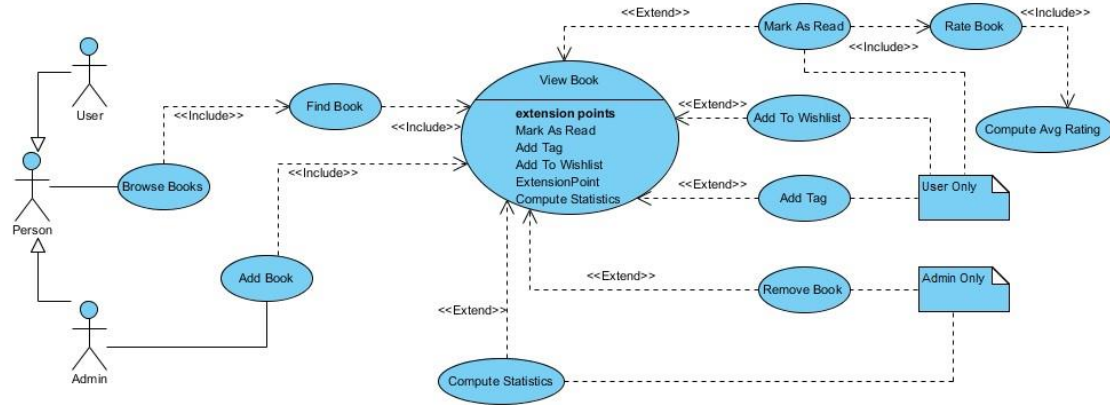


Figure 1: Use-Case Diagram

In the above figure is reported the Use-Case diagram in which we can see: the actors of the application, *Admin* and *User*; their respective action lists and some notes to specify when an action is available only to the admin or only to the user (if it is not already obvious from the diagram).

3.2. Class Analysis Diagram

In *Figure 2* are reported the main entities of the application and the relationships among them.

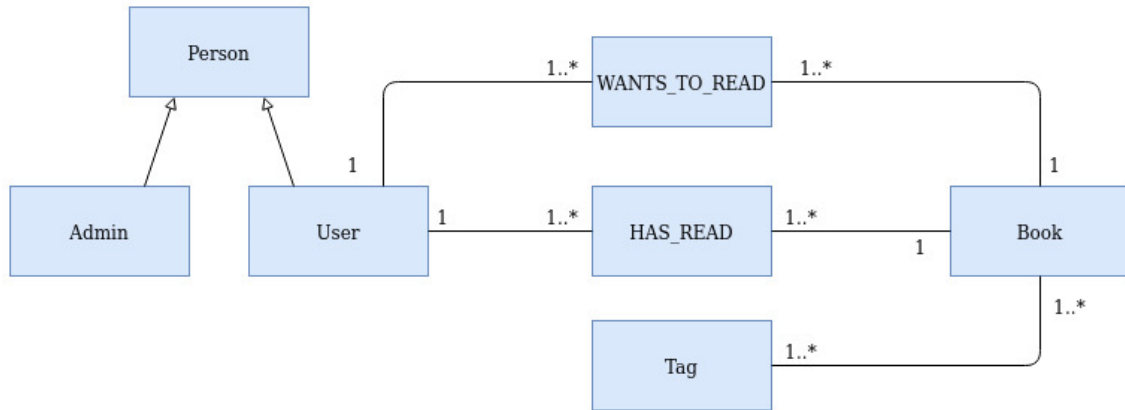


Figure 2: Class Diagram

User and Admin are generalized into Person.

Each User can add one or more Books to his *Wish-List*, and each Book can be added to one or more *Wish-List*.

Each User may have read one or more Books, for which he provides a Rating, and each Book obviously can be read by one or more Users. When a User *rates* a book, it is automatically removed from the wish list, if it was there.

Each Book may be tagged with one or more *Tag*, and each Tag may identify one or more Books.

4. DATABASE ORGANIZATION

For this task we decided to use a new book dataset that we found online. We chose to do this as the dataset of Task1 was of our making and it was very simple and small. With the new dataset, instead, we were able to see the functioning of our graph DB on a large number of nodes and relationships.

4.1. Graph Model

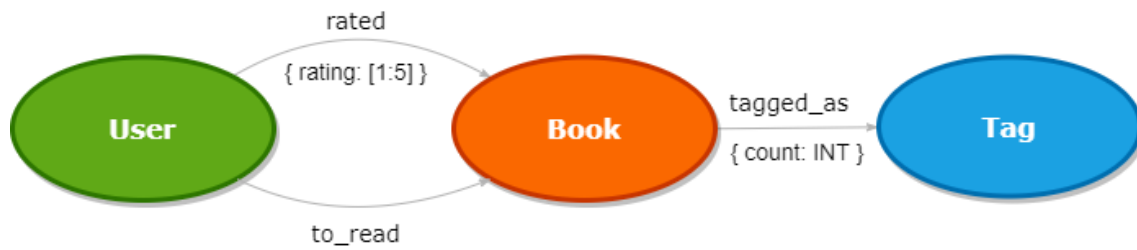


Figure 3: Graph Model

In the above figure is reported the graph representation of our application.

Entities of the application (User, Book, Tag) are specified by a vertex; relations between two entities are represented through directed edges.

Between entities “User” and “Book” there are two different type of relation:

- Users rate books with a score of 1 to 5.
- Users can add books to the list of books he wants to read.

Between entities “Book” and “Tag” there is just one relationship:

- Books can be tagged with a specific tag value (if a book is already tagged with that value its counter is incremented otherwise a new counter for that tag value is created and initialized to 1).

5. SOFTWARE ARCHITECTURE

As *BibliOS*, also *BookRater* has a Client-Side and a Server-side.

The Client-Side is essentially the same as *BibliOS*, it is composed of the GUI implemented in *javafx* which interact with both user and admin, and by the various Interface Controllers.

The Server-side is composed by:

- *Hibernate Entity Manager* which handles the relational DB (containing user information).
- *Neo4j Java Driver* which handles the graph DB (containing book information and relationships with both users and tags).

Please note that since *BookRater* is just an extension of *BibliOS* we did not handle the registration of new users (already handled in *BibliOS*) however, to achieve consistency between the two databases when a user is registered, both JPA and neo4j functions must be called in order to add a new row in the relational database and a new node in the graph one, with matching IDs.

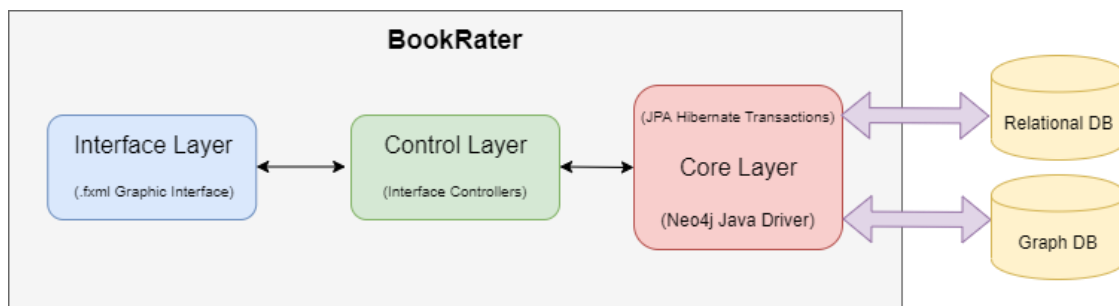


Figure 4: Software architecture of *BookRater*

5.1. Repository Structure

BookRater is a *Maven* project. The project repository is organized as follows:

- **./** : contains the *makefile* and the *POM* file used to generate the maven dependencies and build the project.
- **src/main/java**: contains all the source files of the application.
 - **/controller**: contains the controller classes.
 - **/models**: contains the Java classes that reflect the entities of the application.
- **src/main/resources**: contains the FXML files that describe the appearance of the various interfaces.

6. INSTRUCTION MANUAL

Login Interface

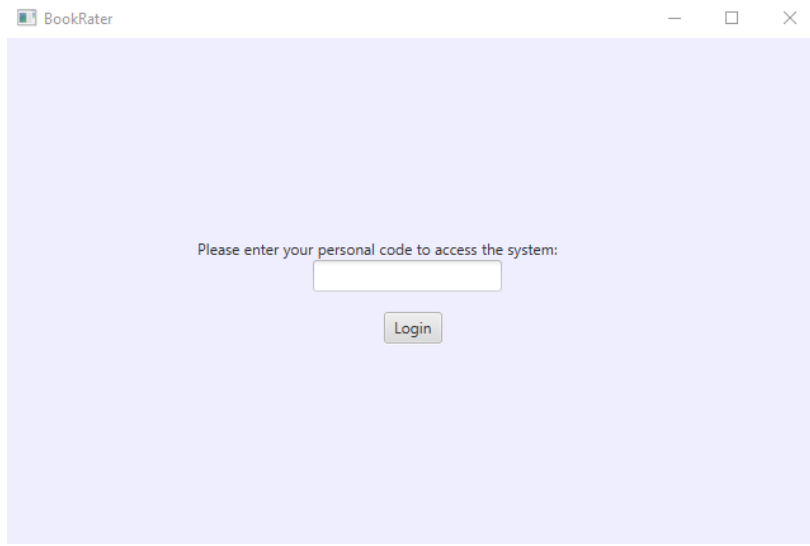


Figure 5: Login interface in BookRater

The login screen of *BookRater* contains:

- **Login field** where both user and admin can insert his personal code to access the main functionalities of *BookRater*.
- **Login button** which redirects the person to the appropriate interface, checking if he is a user or an admin.

Main User Interface

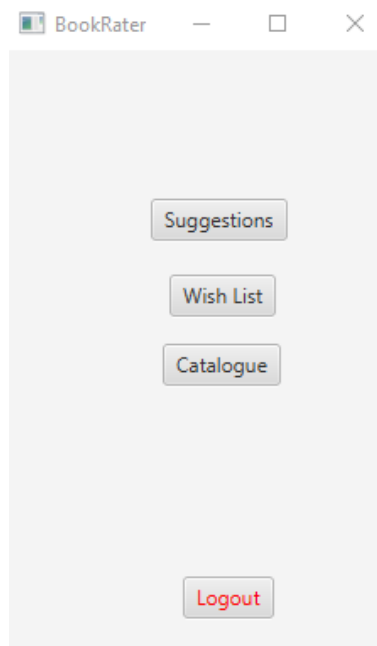


Figure 6: Main user interface

The main user interface contains:

- **Suggestions button** where users can consult their personal list of suggested books provided by *BookRater*, according to their tastes.
- **Wish List button** to browse books that the user wants to read.
- **Catalogue button** to browse the entire books collection of *BookRater*.
- **Logout button** used to return in the login screen.

Catalogue Interface



BookRater

Welcome, Alessio Bongio

[Logout](#) [Back to Menu](#) [View Rated](#)

[Previous](#) [Next](#) Page 25 of 667

ID	Title	Author	Avg_Rating
1027	The Blood of Olympus	Rick Riordan	4.41
1265	Maus II : And Here My Troubles Began	Art Spiegelman	4.41
1363	Proven Guilty	Jim Butcher	4.41
1394	White Night	Jim Butcher	4.41
1513	Lover Avenged, part one	J.R. Ward	4.41
1836	The Kissing Hand	Audrey Penn, Ruth E. Harper, Nancy M. Leak	4.41
1890	The Indigo Spell	Richelle Mead	4.41
2176	The Cat in the Hat and Other Dr. Seuss Favorites	Dr. Seuss, Various	4.41
2205	קיצור תולדות האנושות	Yuval Noah Harari	4.41
2331	November 9	Colleen Hoover	4.41
2840	Homegoing	Yaa Gyasi	4.41
3679	Scott Pilgrim, Volume 6: Scott Pilgrim's Finest Hour	Bryan Lee O'Malley	4.41
3852	Deity	Jennifer L. Armentrout	4.41
4150	How the Light Gets In	Louise Penny	4.41
4344	The Day the Crayons Quit	Drew Daywalt, Oliver Jeffers	4.41

[Mark as Read](#) [Add to Wish List](#) [Add Tag](#)

Figure 7: User catalogue (non-Read books) in BookRater



BookRater

Welcome, Alessio Bongio

[Logout](#) [Back to Menu](#) [View non-Read](#)

[Previous](#) [Next](#) Page 1 of 1

ID	Title	Author	My Rating
135	A Storm of Swords	George R.R. Martin	4.0
9076	Preach My Gospel (A Guide to Missionary Service)	The Church of Jesus Christ of Latter-day Saints	1.0
964	The Hobbit and The Lord of the Rings	J.R.R. Tolkien	5.0
3753	Harry Potter Collection (Harry Potter, #1-6)	J.K. Rowling	1.0
6351	Holy Cow: An Indian Adventure	Sarah Macdonald	4.0

[Change Rating](#) [Add to Wish List](#) [Add Tag](#)

Figure 8: User catalogue (Read books) in BookRater

In these two interfaces the user of *BookRater* will have the possibility to browse both read and non-read books and will be able to perform the following actions:

- **Logout** button used to return in the login screen.
- **Back to Menu** button used to return in the main user interface.
- **Books table** which contains all the books in the catalogue (ID, Title, Author, Avg Rating/My Rating).
- **View Rated** to switch to already read and rated books.
- **View non-Read** to switch to non-Read books.
- **Next/Previous Page**: buttons which scroll the table pages containing the books information.
- **Mark as read**: to specify that the selected book has already been read.
- **Add to Wish List**: to add the selected book to the Wish list.
- **Change Rating**: to modify the rating given to a read book.
- **Insert Metadata Tag Field**: where user can insert a tag for a selected book.
- **Add Tag**: to confirm the insertion of the tag.

Wish List Interface

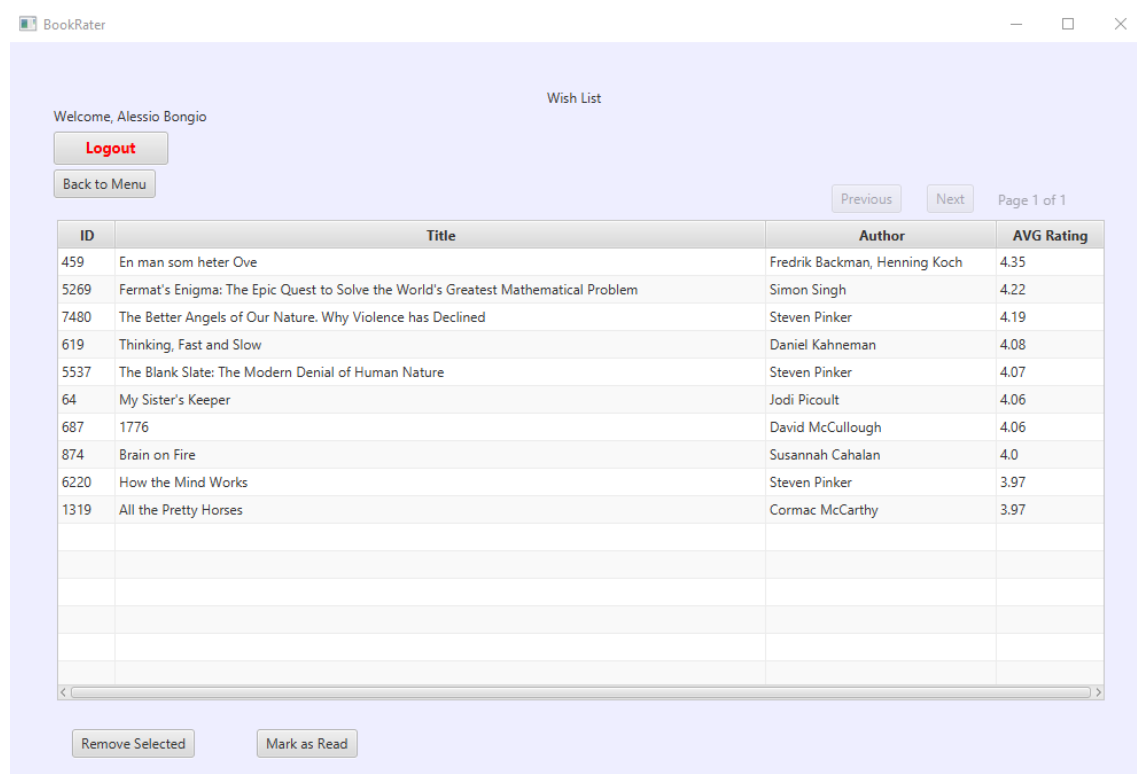


Figure 9: Personal user Wish list

The above panel allows users to browse through books they would like to read and contains:

- **Logout** button used to return in the login screen.
- **Back to Menu** button used to return in the main user interface.

- **Remove Selected:** to remove selected book from personal Wish List.
- **Mark as read:** to specify that the selected book has been read.

Suggestions Interface

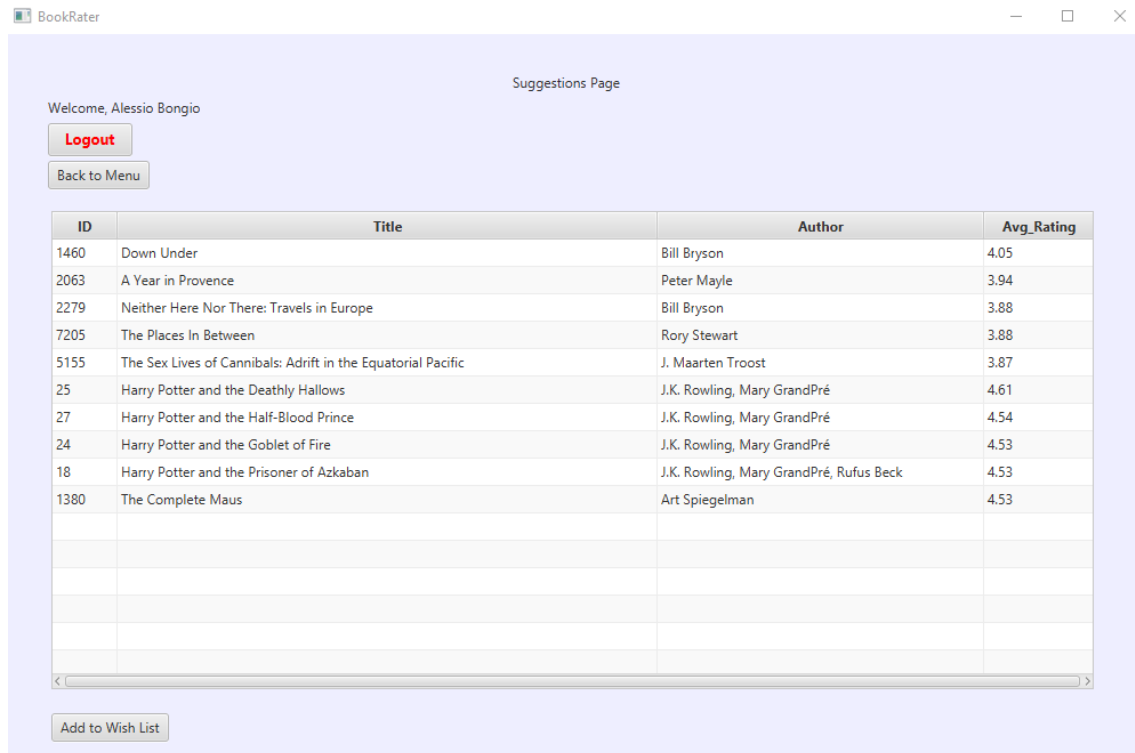


Figure 10: Personal user Suggestions Page

In the Suggestions Page user can browse through recommended books by *BookRater* based on users preferences and tags in common between books, it contains:

- **Logout** button used to return in the login screen.
- **Back to Menu** button used to return in the main user interface.
- **Add to Wish List** button used to add the selected book to the Wish list.

Rating Interface

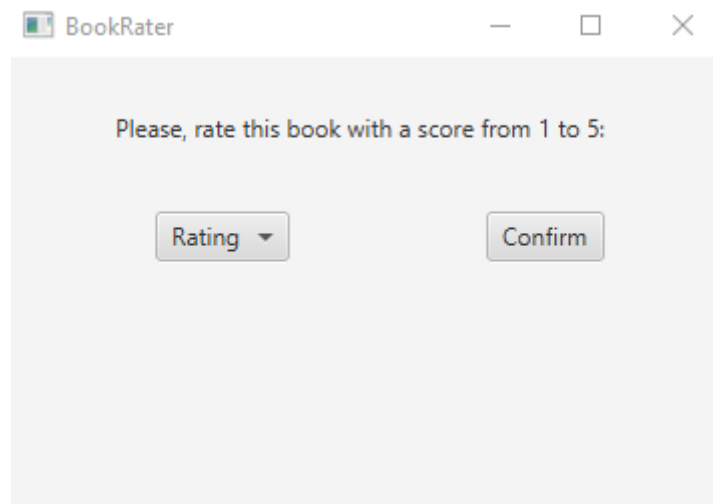


Figure 11: Users Rating Interface

Whenever a user marks a book as read he must express his evaluation of the book rating it with 1-5 stars, this panel contains:

- **Rating** drop down menu to evaluate selected book from 1 to 5.
- **Confirm** button used to confirm their choice.

Main Admin Interface

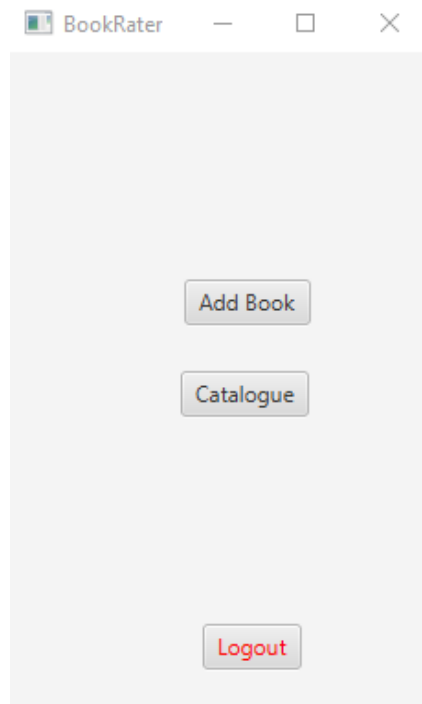


Figure 12: Main Admin Interface

The main admin interface contains:

- **Add Book** button which redirect admin to the interface to add a new book to the catalogue.
- **Catalogue** to browse the entire books collection of *BookRater*.
- **Logout** button used to return in the login screen.

Add Book Interface

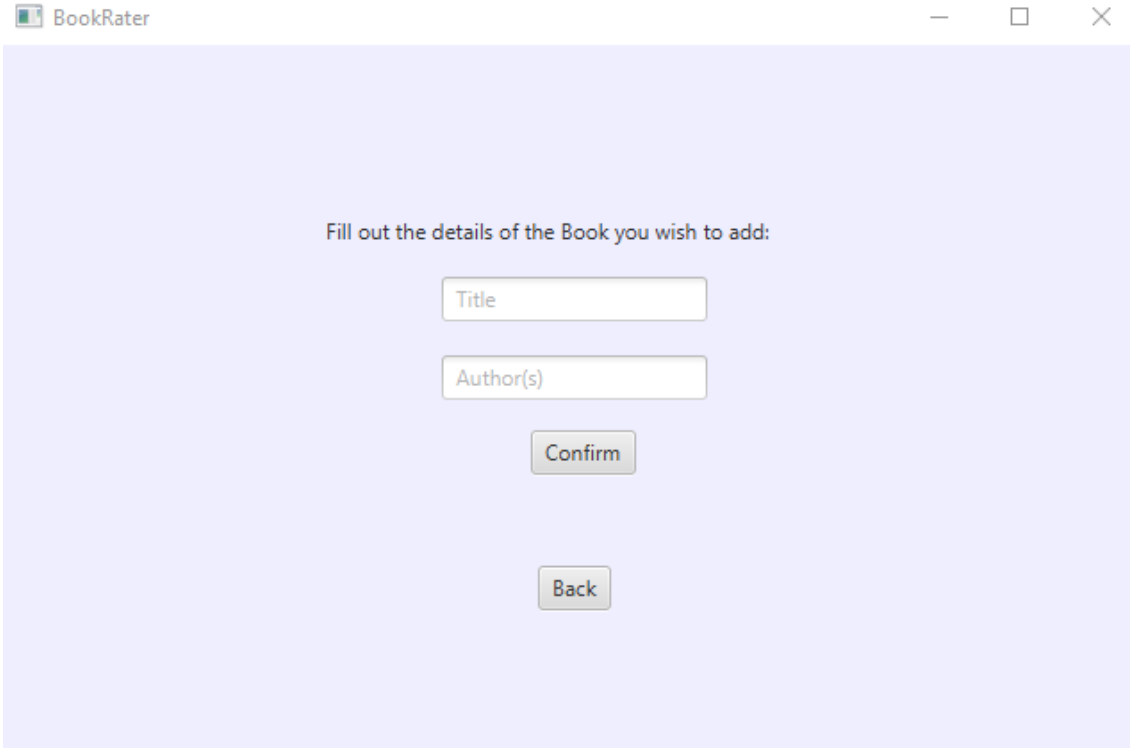
The image shows a screenshot of a web application window titled "BookRater". The window has a light blue background. In the center, there is a text prompt: "Fill out the details of the Book you wish to add:". Below this prompt are two text input fields. The first field is labeled "Title" and the second field is labeled "Author(s)". Below the input fields are two buttons: "Confirm" and "Back". The "Confirm" button is positioned above the "Back" button. The window also features standard window control buttons (minimize, maximize, close) in the top right corner.

Figure 13: Add Book Interface

The above panel allows admins to add a new book filling out its details, it contains:

- **Title** field to insert Book's title.
- **Author(s)** field to insert Book's author(s).
- **Confirm** button used to confirm their choice.
- **Back** button used to return in the Main Admin interface.

Admin Catalogue Interface

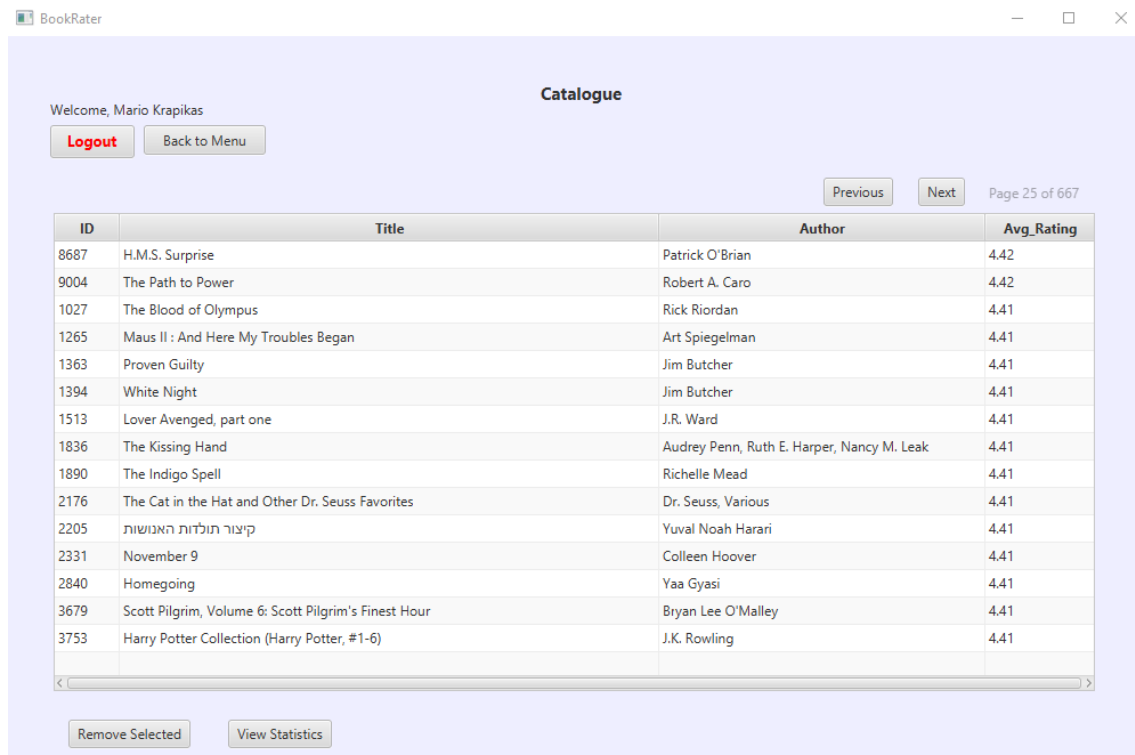


Figure 14: Admin Catalogue Interface

Admin Catalogue interface allows the following actions:

- **Logout** button used to return in the login screen.
- **Back to Menu** button used to return in the main user interface.
- **Books table** which contains all the books in the catalogue (ID, Title, Author, Avg Rating).
- **Next/Previous Page** buttons which scroll the table pages containing the books information.
- **Remove Selected** to remove the selected book from the catalogue.
- **View Statistics** to redirect the admin to the panel in which he can consult statistics of selected book.

Statistics Interface

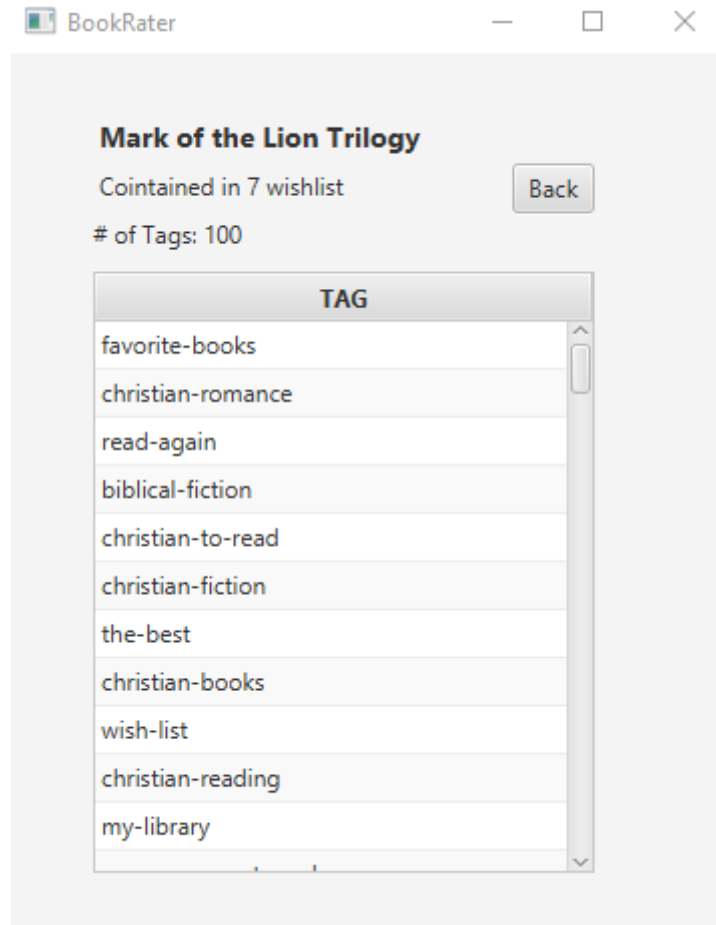


Figure 15: Book Statistics Interface

In the above panel admin can consult statistics of a selected book, especially he can see:

- How many Wish lists contain that book.
- All tags for that book and how many they are.

And, finally, he can return to the Main Admin interface with **Back** button.

7. ON-GRAPH OPERATIONS

Here is a selection of specific on-graph queries, extracted from our application. We decided to put into comparison the difference between the *domain-specific* query (that is, from the application's point of view), the *graph-centric* version and the actual implementation in cypher query language.

Domain-Specific Query	Graph-Centric Query	Cypher Query
What is the the average rating for a book?	Considering a Book-labeled vertex, what is the average weight of inbound edges labeled as RATING?	<pre>MATCH ()-[r:RATED]->(b:Book {book_id:\""+bookId+"\"}) WITH b, AVG(r.rating) AS new_rating</pre>
How many users added a book to their wish-list?	How many inbound edges labeled as TO_READ are incident to a considered Book vertex?	<pre>MATCH (:User)-[r:TO_READ]->(b:Book) WHERE b.book_id = '\" + bookId + '\" RETURN count(r) as totWish</pre>
How many books have a specific tag?	How many inbound edges labeled as TAGGED_AS are incident to a considered Tag vertex?	<pre>MATCH (b:Book)-[r:TAGGED_AS]->(t:Tag) WHERE b.book_id = '\" + bookId + '\" RETURN count(r) as totTags</pre>

8. CONCLUSIONS

The proposed application is only provided with the main features requested for this task however, it would be possible to implement other functionalities that a realistic context would otherwise require.