

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……...….....……………………………………………… | | 4 |
|  | 3.1. Use-Case Diagram…..…………………………………………… | 4 |
|  | 3.2. Class Diagram…………………...……………………………….. | 5 |
| 4. DATABASE ORGANIZATION…................................................................. | | 6 |
|  | 4.1. Graph Model.……………………………………………………. | 6 |
| 5. SOFTWARE ARCHITECTURE……………………………………………. | | 8 |
|  | 5.1. Repository Structure…………………………………………….. | 8 |
| 6. INSTRUCTION MANUAL.………………………………………………... | | 9 |
| 7. ON-GRAPH OPERATIONS……….……………………………………….. | | 14 |
|  |  |  |
|  |  |  |
|  | |  |
| 9. CONCLUSIONS…………………………………………………………….. | | 19 |

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.

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

1. **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 the same functionalities as the regular user but also has additional responsibilities available only to an administrator of the website.

**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:

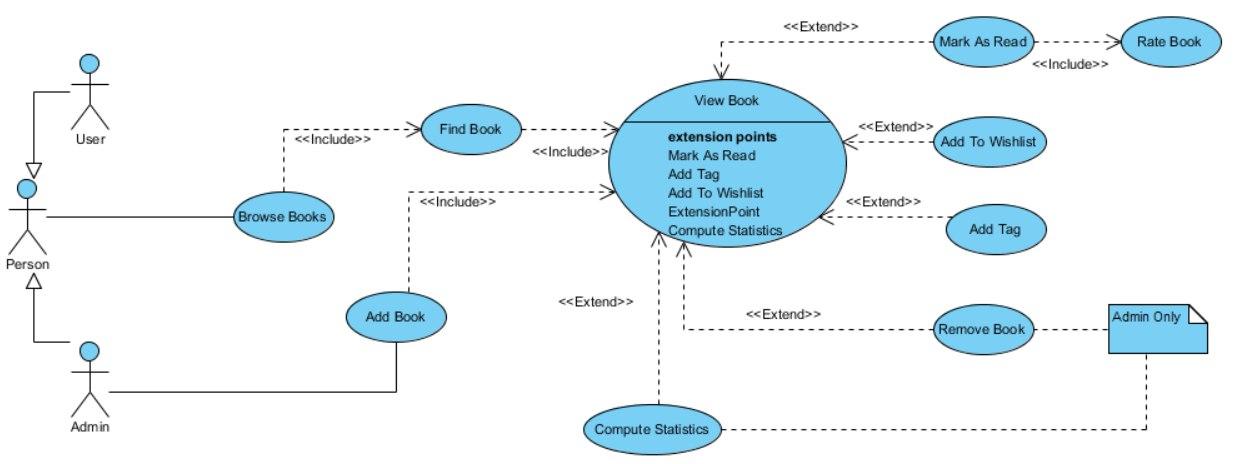
* A User can browse the catalogue of books in the library.
* A User can “mark-as-read” a book and rate it.
* A User can view a *book suggestions* page, based on user ratings & wish list.
* A User can add a *tag* to a book. A tag is metadata that identifies in some way the books.
* A User can search books by tag.
* 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.

1. **UML DIAGRAMS**

**3.1. Use-Case Diagram**

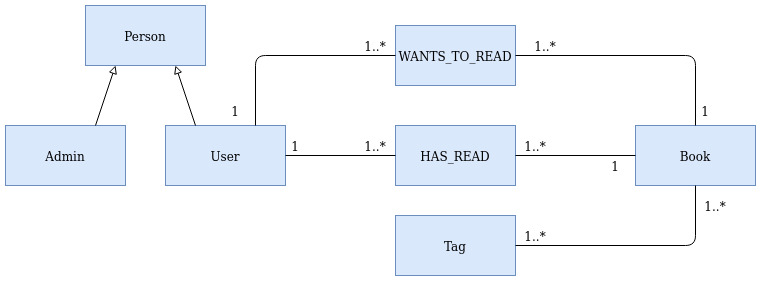
****

*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.

A User can add one or more Books to his *Wish-List*.

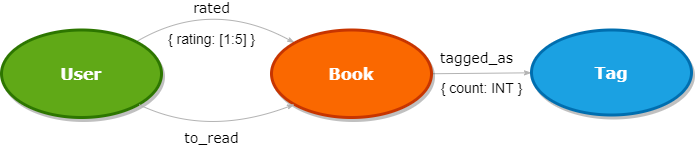
A User may have read one or more Books, for which he provides a Rating.

A User can add one or more metadata tags to a book.

1. **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 particular 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).

1. **SOFTWARE ARCHITECTURE**

*Figure 3: Software architecture of OrgaBet*

**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.

1. **INSTRUCTION MANUAL**

How to use *BookRater*

1. **ON-GRAPH OPERATIONS**

**TABELLA DA DOMAIN SPECIFIC QUERY IN GRAPH CENTRIC**

**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.