
# UNIVERSITÀ DI PISA

## Large-scale and multi-structured database

ACADEMIC YEAR 2022-23

Maria Fabijan

## Table of Contents

# 1. INTRODUCTION

The YourBook is the web-application that gathers reviews of books and share this with different users. It is specialty useful for users whose are looking for the new book that will be interesting for them.

The application's main functionalities are collecting, organizing and presenting to users information about 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 the same functionalities as the regular user but also has additional functionalities available only to an administrator of the website.

## 2.2. Functional and Non-Functional Requirements

1. **Functional requirements.**
    1. **Unregistered user**
        - The system must allow unregistered user to browse books.
        - The system must allow unregistered user to display the information of selected book. This information must include product name, rating, product description, reviews, categories.
        - The system must allow an unregistered user to become a user by registering his/her full name, password, email address, password.
        - The system must allow a registered to login as user .

    2. **Registered user**
        - The system must allow a user change his/her personal details.
        - The system must allow a user to browse book.
        - The system must allow a user to find book.
        - The system must allow a user to display the information of selected book. This information must include book name, rating, book description, reviews, categories.
        - The system must allow a user to add book to his/her wish list while he/she is viewing the book.
        - The system must allow a user to delete book to his/her wish list while he/she is viewing the book.
        - The system must allow a user to browse books.
        - The system must allow a user to find review.
        - The system must allow a user to view a review.

- The system must allow a user to add a review.
- The system must allow a user to delete a review.
- The system must allow a user to browse users.
- The system must allow a user to find user.
- The system must allow a user to view user.
- The system must allow a user to follow the user.
- The system must allow a user to unfollow the users.
- The system must allow a user to browse users.
- The system must allow a user to browse user's reviews.
- The system must allow a user to find user's review.
- The system must allow a user to browse users.
- The system must allow a user to find top followed users.
- The system must allow a user to find most liked book.
- The system must allow a user to browse users.

3. **Company manager**
    - The system must allow a manager to view Analytics
    - The system must allow a manager to show the highest active users
    - The system must allow a manager to show the most searched books.

2. **Non-funcional requiements.**
    - The system must be a responsive website application.
    - The system must encrypt user's passwords.
    - The system will be developed by using Java.
    - The system will have low latency in accessing the database.
    - The system will  be available 24/7.
    - The system will be tolerance to data loss.

## 3. UML DIAGRAMS

### 3.1. Use-Case Diagram

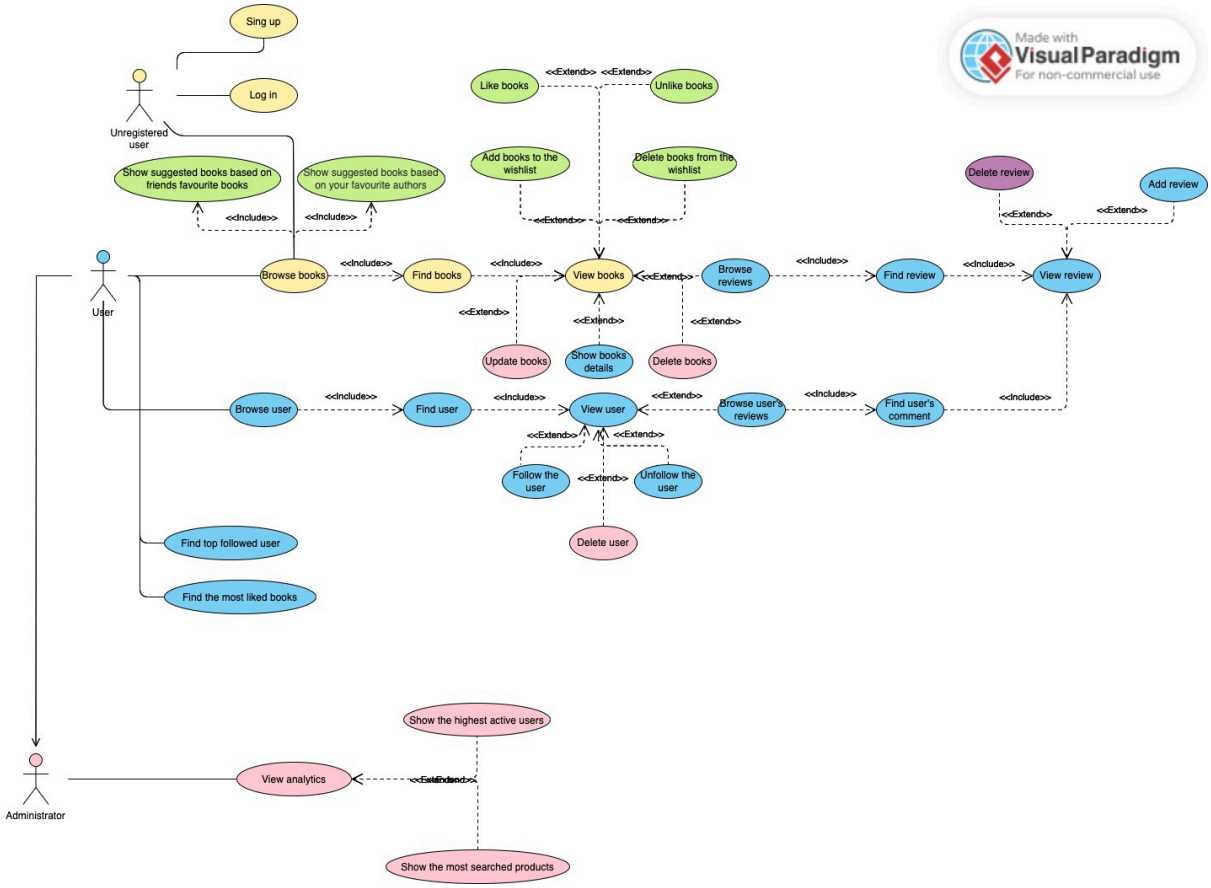



*Figure 1: Use-Case diagram*

In the above figure is reported the Use-Case diagram in which we can see: the actors of the application: Unregistered User, Admin and User.

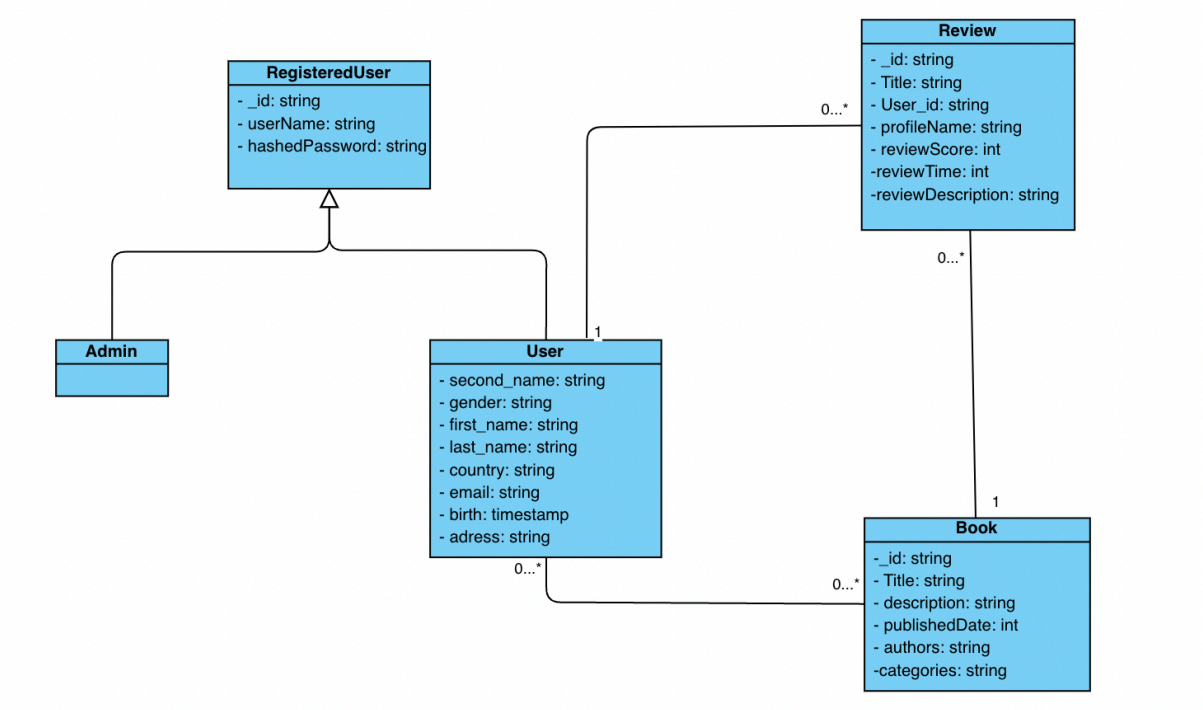
For simplicity of the diagram there are different colors for the actions that can be execute only by the specific actors. The specific colors define:

- yellow for Unregistered User
- blue for Registered User
- pink for Administrator
- green for Registered User, but not for Administrator
- purple is denoted constrain that Register User can delete only his/her review

Each action (excluding sing up) that an unregistered user can do, can be done also by registered
users and administrators.

An Administrator doesn't have a collection of books and reviews, also doesn't have a profile (just username and password), so can't have suggestions,

## 3.2. Class Analysis Diagram




## 4. DATA MODEL

The data model section includes a presentation of the document collections and the graph nodes stored in the database.

YourBook include two database.

First one is the document database, which is managed by MongoDB, in order to store the information about the users, the books and the reviews. This type of DB was chosen for its flexibility, performance capabilities and easy of use.

The second is the graph database for purpose management the social part, which users can add books to their favorites list and follow other users. With this type of database I deal with issue with was representation application domain as networks of connected entities, handle instances of entities that have relations to other instances of entities and require to rapidly traverse paths between entities. This graph database use Neo4j as DBMS.

### 4.1. Document DB

The DBMS include three collections:

- books
- users
- reviews

The reviews collection consists set of reviews documents. The users collection stores all user documents, which also contain reviews. The second collection stores all information related to the users and also the reviews.
The books collection include an embedded array reviews. By embedding the reviews within the book document, I can easily retrieve all of the reviews for a given book in a single query. This can be more efficient than storing the reviews in a separate collection and having to perform a separate query to retrieve them.

- Example of document in books collection:


```
{
  _id: ObjectId("640b7ed38fd520ac9cfdc19b"),
  Title: 'Alaska Sourdough',
  description: '"Sourdough is a magical food", as author Ruth Allman was fond of saying. There are folks in Alaska who claim the staff of life in their sourdough pots is more than 40 years old or date it to the time when Fairbanks was a mining town. Handwwritten to match the old-timers recipes, this book includes directions for several starters that can ripen in varying times, three days to one year. In this witty and useful last word on sourdough cookery, there are more than 95 recipes, loads of time-tested advice for the novice, and plenty of lore for sourdough fans. In this classic last word on sourdough cookery, there are recipes for Alaskan frontier staples (hotcakes to doughnuts) with time-tested advice and love.',
  authors: "['Ruth Allman']",
  publishedDate: '1976',
  categories: "['Cooking']",
  reviews: {
   Id: 'B000NKGYMK',
   Title: 'Alaska Sourdough',
   User_id: 'A3TIBVQGLAB2AM',
   profileName: 'William H. Akins Jr.',
   reviewScore: 5,
   reviewTime: 1243296000,
   reviewTitle: 'Alaska Sourdough',
   reviewDescription: `I received this book as a BD gift... I love it! I live in SE Alaska, a subsistance life style... been trying to learn how to "grow" a really good sourdough... this book has been really helpful and it has some really great recipes from the early 1900's and before! Now, if I can just figure out the quirks of baking and keeping a good sourdough "growing" on my salmon troller...If you love great sourdough, give this a try... "grow" your own using one of Ruth's recipes (included)... it is well worth the time and your guests will truly enjoy the meals you make from your sourdough jar...h. akinsnaukati, ak`
  }
 }
```


- Example of document in users collection


```
{
  _id: ObjectId("6409cb9af24bbb1e548bdf35"),
  user_id: 'AVRN9E6X9I6KF',
  username: 'wilsoncolleen',
  second_name: 'Shaffer',
  hashed_password: 'a1827ef12a4cfb52c1f2c0506bde21871a50aa92c5ee2eb0b0471af46968e59e',
  gender: 'Female',
  first_name: 'Dominic',
  last_name: 'Mcdonald',
  country: 'Togo',
  email: 'sanchezrenee@example.com',
  birth: '1997-05-06',
  address: '988 Hayes Locks\nLesliefort, WY 76269'
 }
```


- Example of review in reviews collection


```
{
  _id: ObjectId("64187e15f6d9477aa8a605d8"),
  Id: 595344550,
  Title: 'Whispers of the Wicked Saints',
  User_id: 'A2YWUISTJ728T5',
  profileName: 'darby',
  reviewScore: 5,
  reviewTime: 1115856000,
  reviewTitle: 'A FIVE STAR BOOK',
```

reviewDescription: 'New review description'
}

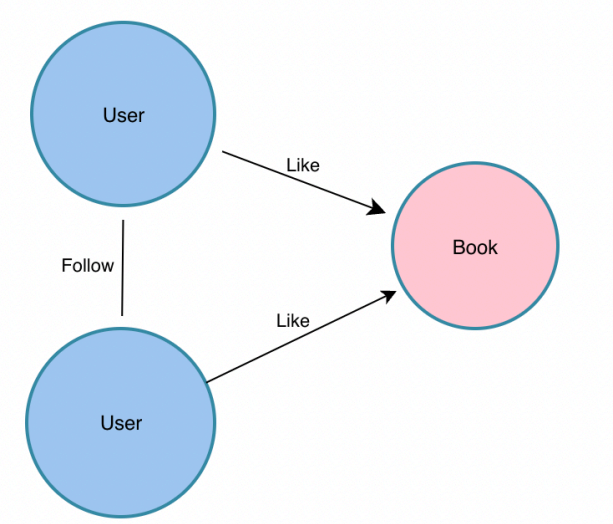
### 4.2. Graph DB

In the Graph DB I distinguished two entities:

- **User**
- **Books**

The relations between entities are:

- **User – User: Follow** (user can follow another user)
- **User – > Books: Like** (user can add the book to the his/her favorite list)




## 5. CRUD Operations

### 5.1 Mongo DB

#### *5.1.1 Create*

- Add book:

```
db.books.insertOne({
Title: 'The Great Gatsby',
 description: 'A novel by F. Scott Fitzgerald about the decadence and excess of the 1920s Jazz Age.',
 authors: ['F. Scott Fitzgerald'],
 publishedDate: '1925',
 categories: ['Fiction', 'Classic Literature'],
 reviews: {
  Id: '0743273567',
  Title: 'The Great Gatsby',
  User_id: 'A3OZDTEEAF8GS9',
  profileName: 'booklover23',
  reviewScore: 5,
  reviewTime: 1552656000,
  reviewTitle: 'A masterpiece of American literature',
```

reviewDescription: 'The Great Gatsby is a beautifully written novel that captures the spirit of the 1920s. Fitzgerald paints a vivid picture of the excess and glamour of the time, while also exploring deeper themes of love, loss, and the American Dream. The characters are complex and flawed, and their stories are both tragic and compelling. This is a book that everyone should read.'
}
});

#### *5.1.2 Read*

- Find user by name:
  db.user.find({name: “Maria”})

#### *5.1.3 Update*

Update review

- db.collection.updateOne(<filter>, <update>, <options>) ——>db.<collection name>.updateOne({name: “Maria”}, {$set {“gender”: “girls”}, $currentDate: { lastModified: true} } )

#### *5.1.4 Delete*

Delete review

- db.<collection name>.deleteMany({gender: “Female”})

### 5.2 Neo4J

#### *5.2.1 Create*

- Add user
  CREATE (u:User {
  country: "Zimbabwe",
  address: "123 Main Street
  Anytown, USA 12345",
  gender: "Female",
  birth: "1990-01-01",
  last_name: "Smith",
  second_name: "Rose",
  first_name: "Emily",
  email: "emily.smith@example.com",
  username: "emilysmith"
  })

#### *5.2.2 Read*

- Find user by username
  MATCH (u:User {username:”emilysmith”}) RETURN u

#### *5.2.3 Update*

- Update user address
  MATCH (u:User {username:"emilysmith"}) SET u.adsress = "Topolowa 2, Kamieniec Wrocławski"

#### *5.2.4 Delete*

- Delete user (by username)

MATCH (u:User{username:"emilysmith"}) DELETE u

## 6. Queries

### 6.1 Mongo aggregation

- Display 10 *top countries with the highest number of users (admin)*

```
db.users.aggregate( [{ $group: { _id:"$country", numUsers: { $sum: 1 }}}, { $sort: { numUsers: -1 }}, { $limit: 10}, { $project: { _id: 0, country: "$_id", users: "$numUsers" }}])
```

- *find top rated books*

```
db.books.aggregate([
 { $unwind: "$reviews" },

 { $group: {
   _id: "$_id",
   title: { $first: "$Title" },
   avgScore: { $avg: "$reviews.reviewScore" }
 }},

 { $sort: { avgScore: -1 } },


 { $limit: 10 },

 { $project: {
   _id: 0,
   title: 1,
   score: "$avgScore"
 }}
]);
```

- *find most active users (admin)*

```
db.books.aggregate( [{
$group: {
      _id: "$reviews.User_id",
       count: { $sum: 1 }
    }
  },
  {
    $sort: { count: -1 }
  },
  {
    $limit: 5
  }
 ])
```

- *find top rated authors*

```
db.books.aggregate([
 { $unwind: "$authors" },
```

```
  { $group: {
      _id: "$authors",
      avg_review_score: { $avg: "$reviews.reviewScore" }
  }},

  { $sort: { avg_review_score: -1 } },

  { $limit: 5 }
])
```

## 6.2 Neo4J Queries

- Display top 3 most followed users

```
MATCH (u:User)-[:FOLLOW]->(u2:User) WITH u2, count(*) as followsCount
ORDER BY followsCount DESC LIMIT 3 RETURN u2.username AS username,
followsCount
```

- Display suggested users based on their favourite authors

```
MATCH (u:User)-[:LIKE]->(b:Book)<-[:LIKE]-(su:User)
WHERE u.username = 'aspencer' AND su <> u
WITH su, COLLECT(DISTINCT b.authors) AS authors
UNWIND authors AS author
WITH su, author, COUNT(*) AS bookCount
ORDER BY bookCount DESC
RETURN su.username AS suggestedUser, COLLECT(DISTINCT author)[..3] AS
commonAuthors
```

- Display books based on friend's favourite books

```
MATCH (me:User {username: 'ashleyreed'})-[:FOLLOW]->(friend:User)-[:LIKE]-
>(book:Book)
WHERE NOT (me)-[:LIKE]->(book)
RETURN DISTINCT book.title, book.authors, count(friend) AS numFriends
ORDER BY numFriends DESC
LIMIT 10
```

- Display most liked authors

```
MATCH (user:User)-[:LIKE]->(book:Book)
UNWIND split(book.authors, ',') AS author
WITH author, count(DISTINCT user) AS numLikes
ORDER BY numLikes DESC
RETURN author, numLikes
LIMIT 10
```

## 7. Indexes

Indexes in a document database can significantly improve query performance by reducing the number of documents that need to be scanned. Here are the indexes used in the experiment and their results:

| # | Collection | Index name | Index type | unique | Attribute |
|---|---|---|---|---|---|
| 1. | books | bookName | single | unique | title |
| 2. | books | publicationDate | single | not unique | publishedDate |
| 3. | reviews | Title | single | not unique | Title |
| 4. | users | username | single | unique | ussername |

1. 'Eyewitness Travel Guide to Europe'

- executionStats: {
  executionSuccess: true,
  nReturned: 5,
  executionTimeMillis: 100,
  totalKeysExamined: 0,
  totalDocsExamined: 7099,
  executionStages: {
   stage: 'COLLSCAN',
   filter: { Title: { '$eq': 'Eyewitness Travel Guide to Europe' } },

- With Index (test> db.books.createIndex( { Title: 1}))

executionStats: {
  executionSuccess: true,
  nReturned: 5,
  executionTimeMillis: 4,
  totalKeysExamined: 5,
  totalDocsExamined: 5,
  executionStages: {

2. db.books.find({ publishedDate: '2016-01-05'}).explain("executionStats")

   - executionStats: {
   executionSuccess: true,
   nReturned: 5,
   executionTimeMillis: 14,
   totalKeysExamined: 0,
   totalDocsExamined: 7099,
   executionStages: {

- executionStats: {
  executionSuccess: true,
  nReturned: 5,
  executionTimeMillis: 7,

totalKeysExamined: 5,
totalDocsExamined: 5,
executionStages: {

3. db.reviews.find({Title: 'Dramatica for Screenwriters'}).explain("executionStats")
   - executionStats: {
     executionSuccess: true,
     nReturned: 20,
     executionTimeMillis: 57,
     totalKeysExamined: 0,
     totalDocsExamined: 64216,
     executionStages: {
   - executionStats: {
     executionSuccess: true,
     nReturned: 20,
     executionTimeMillis: 8,
     totalKeysExamined: 20,
     totalDocsExamined: 20,
     executionStages: {

4. db.users.find({username:'walkerbrittany'}).explain("executionStats")
   - executionStats: {
     executionSuccess: true,
     nReturned: 3,
     executionTimeMillis: 445,
     totalKeysExamined: 0,
     totalDocsExamined: 102105,
     executionStages: {

   - executionStats: {
     executionSuccess: true,
     nReturned: 3,
     executionTimeMillis: 5,
     totalKeysExamined: 3,
     totalDocsExamined: 3,
     executionStages: {

Experimentally, we can see that using indexes can significantly improve query performance. In each query, the first execution without an index resulted in a longer execution time and a higher number of documents examined. However, the second execution with an index resulted in a much shorter execution time and a lower number of documents examined. This demonstrates the value of using indexes to optimize document database queries.

## 8. Summary

Overall, the YourBook project is a web application designed to gather book reviews and share them with users. The application allows unregistered users to browse books and view information about selected books, including the product name, rating, description,

reviews, and categories. Unregistered users can also register to become users by providing their full name, email address, and password.

Registered users have additional functionalities, including the ability to change their personal details, browse books, find books, add books to their wish list, delete books from their wish list, browse reviews, view reviews, add reviews, delete reviews, browse users, find users, view user profiles, follow users, unfollow users, browse other users' reviews, find specific reviews by users, find top followed users, find the most liked book, and browse other users.

The administrator, referred to as the company manager, has access to analytics and can view the highest active users and the most searched books.

The application has both functional and non-functional requirements. Functional requirements outline the specific actions that each type of user can perform, while non-functional requirements describe the desired characteristics of the system, such as responsiveness, password encryption, development language (Java), low latency, 24/7 availability, and tolerance to data loss.

The system utilizes two databases: a document database managed by MongoDB and a graph database managed by Neo4j. The document database stores collections for books, users, and reviews, while the graph database handles the social aspect of the application, allowing users to add books to their favorites list and follow other users.

The CRUD operations for the document database (MongoDB) include create, read, update, and delete operations for books, users, and reviews. Examples of these operations are provided in the text.

Similarly, the CRUD operations for the graph database (Neo4j) include create, read, update, and delete operations for users. Examples of these operations are also provided in the text.
The data model consists of document collections and graph nodes stored in the respective databases. The document database includes collections for books, users, and reviews, with embedded reviews within the book documents for efficient retrieval. The graph database includes entities for users and books, with relationships such as follow and like.
The queries section provides examples of MongoDB aggregation queries for finding top countries with the highest number of users, top-rated books, most active users, and top-rated authors. Similarly, examples of Neo4j queries are given for displaying the most followed users, suggesting users based on favorite authors, displaying books based on friends' favorite books, and finding the most liked authors.
Overall, the YourBook project aims to provide a user-friendly web application for book enthusiasts to explore and share book reviews, connect with other users, and discover new books based on their interests.