Feyzan Colak, Flavio Messina, Noemi Cherchi
Large-scale and multi-structured databases project
2023-2024

Contents

Introduction

MangaVerse is a web application project developed for the Large-scale and multi-structured databases course of the University of Pisa. This web application aims to provide users with a comprehensive platform to explore, search, and interact with a vast collection of manga and anime. Unregistered users can browse the manga and anime, access basic details about each media content and utilize search and filtering options. Users can register and with registering, user can personalize their profiles and engage in a community with. The platform also offers a set of features for registered users, such as liking media contents, following other users, adding reviews and ratings. Additionally, users receive personalized suggestions based on their preferences and current trends. The site manager has access to detailed analytics about media contents and user activities.

Beside the user roles, the web application also has managerial roles. MangaVerse provides an analytics dashboards to track media contents and user activities also managing media contents and user accounts. These features allow manager to add, update, or remove manga and anime entries and monitor trends in user interactions and popular genres.

Through its comprehensive set of features, MangaVerse aims to provide a community of manga and anime enthusiasts with a platform to explore, share, and engage with their favorite content. This platform enhances the user experience and facilitates deep engagement with both the content and the community.

Analisys

Actors

- **Unregistered User**: A visitor who has not created an account on the platform. They can browse media contents, search and filter, view media content details, and register/login.
- **Registered User**: A user who has created an account on the platform. They can perform all the actions of an unregistered user, as well as logout, manage their profile, explore other user profiles, interact with media contents/users, review media contents, and receive advanced recommendations.
- **Manager**: A registered user with administrative features. They can access an analytics dashboard, manage user accounts, content entries, and monitor trends.

Requirements

Unregistered User:

- Browse Media Contents:
  - View a list of available manga and anime on the home page.
  - Access basic details about each media content without logging in.
- Search and Filter:
  - Use the search bar to find specific manga or anime by title.
  - Utilize basic filtering options to refine the media content list.
- View Media Content Details:
  - Click on a media content to view detailed information, including synopsis and genre.
- Register/Login:
  - Access a registration page to create a new account.
  - Use valid credentials to log into the account.
- Explore Features:
  - Access information about the features available to registered users.

Registered User:

- Browse Media Contents:
  - View a list of available manga and anime on the home page.
  - Access basic details about each media content without logging in.
- Search and Filter:
  - Use the search bar to find specific media content by title.
  - Utilize basic filtering options to refine the media content list.
- View Media Content Details:
  - Click on a media content to view detailed information, including synopsis and genre.
- Logout:

  - Ends the user's session.
- Profile Management:
  - Edit and update personal information (e.g., profile picture, bio).
  - Change account password.
- Explore Other User Profiles:
  - View profiles of other registered users.
  - See their liked manga, anime and reviews.
- Interact with Media Contents/Users:
  - Like or dislike manga and anime to indicate preferences.
  - Follow/unfollow other users.
- Review Media Contents:
  - Add reviews and ratings to manga and anime.
  - View and edit own reviews.
- Advanced Recommendations:
  - Receive more refined media content suggestions based on detailed user interactions.
  - Receive users suggestions based on common interests.

Manager(Registered User with Administrative Features):

- Analytics Dashboard:
  - Access a comprehensive analytics dashboard with data on user engagement and media contents trends.
- User Management:
  - View and manage user accounts, including account activation and deactivation.
- Content Management:
  - Manage media content entries, including adding new manga and anime, updating information, and removing entries if necessary.
- Monitor Trends:
  - Monitor trends in user interactions, popular genres, and trending manga.

Non Functional Requirements

Performance

- Response Time: The system should have low latency, with pages loading within an acceptable timeframe.
- Scalability: The system should be able to handle an increasing number of users and data without significant degradation in performance.
- Concurrency: The application should support multiple users simultaneously without performance bottlenecks. For very high traffic scenarios, acceptable delays may be introduced.

Security

- Data Encryption: All user data, including passwords, should be securely encrypted during transmission and storage.

User Interface

- Responsiveness: The user interface should be responsive, providing a consistent and seamless experience across various devices and screen sizes.
- Intuitiveness: The interface should be user-friendly, with clear navigation and easily understandable features.

UML use case diagram

UML class analysis

UML class diagram

Design

Document Database

For the document database, we will use MongoDB. MongoDB is a NoSQL database that stores data in flexible, JSON-like documents. It is a popular choice for applications that require flexibility and scalability. MongoDB is a document database, which means it stores data in JSON-like documents. These documents are flexible, meaning they can have different fields and structures. This makes MongoDB a good choice for applications that require flexibility in their data model. MongoDB is also a scalable database, meaning it can handle large amounts of data and traffic. It is designed to scale out, meaning you can add more servers to handle more traffic. This makes MongoDB a good choice for applications that need to scale quickly.

Collections The database will have the following collections:

- Anime: This collection will store information about anime, such as titles, tags, and synopsis.
- Manga: This collection will store information about manga, such as titles, genres, and authors.
- Reviews: This collection will store user ratings and comments for media content.
- Users: This collection will store user data, such as usernames, passwords, email addresses, gender and location.

MongoDB document example

Anime:


```
{
  "_id": "65789bb52f5d29465d0abcfb",
  "title": "0",
  "type": "SPECIAL",
  "episodes": 1,
  "status": "FINISHED",
  "picture": "https://cdn.myanimelist.net/images/anime/12/81160.jpg",
  "tags": [
    "drama",
    "female protagonist",
    "indefinite",
    "music",
    "present"
  ],
  "producers": "Sony Music Entertainment",
  "studios": "Minakata Laboratory",
  "synopsis": "This music video tells how a shy girl with a secret love and
     curiosity...",
  "latest_reviews": [
    {
      "id": "657b301306c134f18884924c",
      "date": "2023-10-03T22:00:00.000+00:00",
      "rating": 4,
      "user": {
        "id": "6577877ce68376234760745c",
        "username": "Tolstij_Trofim",
        "picture": "https://thypix.com/wp-content/uploads/2021/10/manga-
           profile-picture-10..."
      }
    },
  ],
  "anime_season": {
    "season": "FALL",
    "year": 2013
  },
  "average_rating": 6.7,
  "avg_rating_last_update": true,
  "likes": 4
}
```

Manga:


```
{
  "_id": "657ac61bb34f5514b91ea223",
  "title": "Berserk",
  "type": "MANGA",
  "status": "ONGOING",
  "genres": [
    "Action",
    "Adventure",
    "Award Winning",
    "Drama",
    "Fantasy",
    "Horror",
    "Supernatural"
  ],
  "themes": [
    "Gore",
    "Military",
    "Mythology",
    "Psychological"
  ],
  "demographics": [
    "SEINEN"
  ],
  "authors": [
    {
      "id": 1868,
      "role": "Story & Art",
      "name": "Kentarou Miura"
    },
    {
      "serializations": "Young Animal"
    }
  ],
  "synopsis": "Guts, a former mercenary now known as the \"Black Swordsman
     ,\" is out fo...",
  "title_english": "Berserk",
  "start_date": "1989-08-25T00:00:00.000+00:00",
  "picture": "https://cdn.myanimelist.net/images/manga/1/1578971.jpg",
  "average_rating": 3.33,
  "latest_reviews": [
    {
      "user": {
        "id": "6577877be683762347605ce7",
        "username": "calamity_razes",
        "picture": "https://imgbox.com/7MaTkBQR"
      },
      "date": "2012-12-15T00:00:00.000+00:00",
      "comment": "An insult to the art of manga; avoid at all costs.",
      "id": "657b302206c134f18886f5ef"
    },
  ],
  "anime_season": {
    "season": "FALL",
    "year": 2013
  },
  "average_rating": 6.7,
  "avg_rating_last_update": true,
  "likes": 4
}
```

Reviews:


```
{
  "_id": "657b300806c134f18882f2f1",
  "user": {
    "id": "6577877be68376234760596d",
    "username": "Dragon_Empress",
    "picture": "images/account-icon.png",
    "location": "Columbus, Georgia",
    "birthday": "1987-07-29T00:00:00.000+00:00",
    "rating": 7
  },
  "anime": {
    "id": "65789bbc2f5d29465d0b18b7",
    "title": "Slayers Revolution",
    "date": "2023-07-23T06:27:54.000+00:00",
    "comment": "Above-average quality in animation and soundtrack."
  }
}
```


Users:


```
{
  "_id": "6577877be683762347605859",
  "email": "xdavis@example.com",
  "password": "290
      cb38a679d5eb68d11b9ea1e21f48234eba6de19f95612dbcb70ce0c7e4e78",
  "description": "Liberating the mind from stress with the power of anime
      zen.",
  "picture": "https://thypix.com/wp-content/uploads/2021/10/manga-profile-
      picture-44",
  "username": "Xinil",
  "gender": "Male",
  "birthday": "1985-03-04T00:00:00.000+00:00",
  "location": "Libya",
  "joined_on": "2014-05-29T00:00:00.000+00:00",
  "app_rating": 5,
  "followed": 40,
  "followers": 29
}
```


Indexes

We created two indexes in the reviews collection to improve query performance. One for the users id and another for the anime and manga id. This will allow us to quickly retrieve reviews for a specific user or media content.

Graph Database

For the graph database, we will use Neo4j. Neo4j is a graph database that stores data in nodes and relationships. It is a popular choice for applications that require complex relationships between data. Neo4j is a graph database, which means it stores data in nodes and relationships. Nodes represent entities, such as users or products, and relationships represent connections between nodes. This makes Neo4j a good choice for applications that require complex relationships between data. Neo4j is also a scalable database, meaning it can handle large amounts of data and traffic. It is designed to scale out, meaning you can add more servers to handle more traffic. This makes Neo4j a good choice for applications that need to scale quickly.

Nodes

The database will have the following nodes:

- User: This node will store information about users, such as id, usernames, and picture.

- Anime: This node will store information about anime, such as id, titles and picture.
- Manga: This node will store information about manga, such as id, titles and picture.

Relationships
The database will have the following relationships:

- LIKE: This relationship will connect users to anime and manga nodes. It will store the date when the user liked the media content.
- FOLLOW: This relationship will connect users to other users.

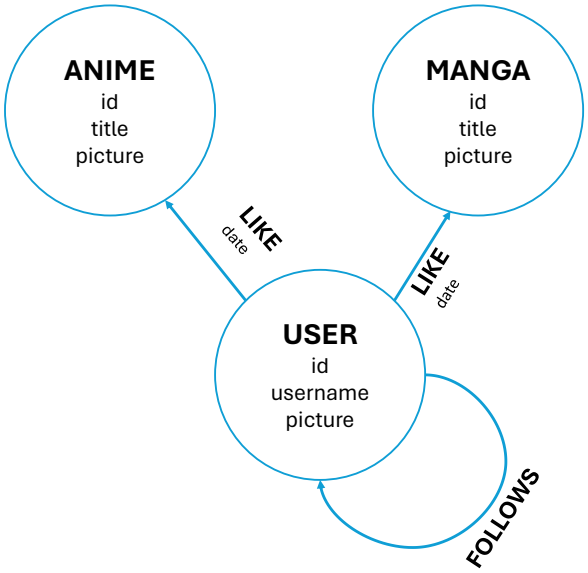

Figure 3.1: GraphDB

Implementation

Development Environment

To ensure efficient and successful Implementation of MangaVerse web application, choosing the appropriate development environment is one of the most important points of the project.

Programming Languages

- **Backend:** Java is the main programming language used in the project's backend development.
- **Frontend:** HTML, CSS, JavaScript are utilized for building user interface in the project.
- **Data Preprocessing:** Python is used in the project to conduct data preprocessing task with the help of its powerful libraries and ease of use features.

Database

- **Document Database:** MongoDB is used in the project to store and manage document-based data with the help of its flexibility and scalability features.
- **Graph Database:** Neo4j is used in the project to manage and query graph data and handle complex relationships and connections in the data efficiently.

Integrated Development Environment

Intellij IDEA is used as an primary IDE. It is powerful Java integrated development environment for developing software in an efficient way.

Version Control

Github is used to provide a collaborative development with its version control system.

Web Server

Apache Tomcat is used as a web server to provide reliable environment for deploying and running the java based web application.

Build Automation

Maven is used as a build automation tool. It is used to manage the project's build, reporting, and documentation from a central piece of information.

Testing

JUnit is used as a testing framework for Java code. It is used to write and run repeatable automated tests. This ensures the reliability and efficiency of the codebase throughout the development process.

Main Modules

- Configuration
- Controller
- DAO (Data Access Objects)
- DTO (Data Transfer Objects)

- Model
- Service
- Utils
- User Interface

Configuration

Configuration module contains a class named *AppServletContextListener* which is responsible for initializing and managing database connections for the web application. The configuration class implements ServletContextListener interface. *@WebListener* annotation is used to provide listening for application lifecycle events. This annotation contains two methods, which are *contextInitialized(ServletContextEvent sce)* and *contextDestroyed(ServletContextEvent sce)*. The first method is called when the web application is started and the second method is called when the web application is shut down.

Database Connection Management: Database connection is provided with *openConnection()* and *closeConnection()* methods. They are both initialized for managing connection for MongoDB and Neo4j databases. Connections are managed with corresponding DAO classes which are BaseMongoDBDAO and BaseNeo4jDAO.

With using the configuration module for database connection, web application ensures robustness and reliability in its data access layer.

Controller

The controller modules plays a role as intermediary between the user requests and backend of the MangaVerse wab application as servlet classes. they receives the user requests, process them and returns with the corresponding response. Each controller class utilized a switch-case structure to determine the action requested and invokes the appropriate handler method accordingly.

Example code snippet from MediaContentServlet:

```
{
    protected void processRequest(HttpServletRequest request,
        HttpServletResponse response) throws ServletException, IOException {
        String action = request.getParameter("action");

        switch (action) {
            case "toggleLike" -> handleToggleLike(request, response);
            case "addReview" -> handleAddReview(request, response);
            case "deleteReview" -> handleDeleteReview(request, response);
            case "editReview" -> handleEditReview(request, response);
            case "getMediaContent" -> handleGetMediaContentById(request,
                response);
            case "getMediaContentByTitle" -> handleSearchMediaContentByTitle
                (request,response);
            case null, default -> handleLoadPage(request,response);
        }
    }
}
```

The controller module contains the following classes:

- Exception
  NotAuthorizedException: This exception is thrown when the user is not authorized to access the requested resource.
- AuthServlet
  The AuthServlet class handles the user authentication and authorization processes. It includes login, logout and sign up functions

- MainPageServlet
  The MainPageServlet class is responsible for handling the main page of the web application. It includes the main page of the web application and the search functionality. It provides request related to displaying main page and searching media contents.

- ManagerServlet
  The ManagerServlet class manages administrative requests in manager page. These request are primarily about manga, anime and user analytics such *averageRatingByMonth(), trendMediaContentByYear(), getBestCriteria()...*

- MediaContentServlet
  The MediaContentServlet class is responsible for managing request related with media contents. These requests include like, adding,deleting or editing reviews and retrieving media content details.

- ProfileServlet
  The ProfileServlet class is responsible for managing user profile related requests. These requests include updating user profile, following/unfollowing other users, getting user profile details such as liked anime and manga and user reviews.

- UserServlet
  The UserServlet class is responsible for managing user related requests and interactions. These requests include retrieving followers list, following list and user information.

DAO (Data Access Objects)

The DAO module includes the logic for accessing and managing data in the database and provides data retrieval, storage and manipulation. This module includes classes with CRUD (create, read, update, delete) operations and query executions. It provides a layer of abstraction between the database and the rest of the application and ensures the separation of concerns. The DAO module contains the following classes:

- Enums
  - DataRepositoryEnum

- Exceptions

- Interfaces
  - MediaContentDAO
  - ReviewDAO
  - UserDAO

- Mongo
  - AnimeDAOMongoImpl
  - BaseMongoDBDAO
  - MangaDAOMongoImpl
  - ReviewDAOMongoImpl
  - UserDAOMongoImpl

- Neo4j
  - AnimeDAONeo4jImpl
  - BaseNeo4jDAO
  - MangaDAONeo4jImpl
  - UserDAONeo4jImpl

- DAOLocator

Example code snippet from MangaDAOMongoImpl:

```
{
    //MongoDB queries
    //Best genres/themes/demographics/authors based on the average rating
    @Override
    public Map<String, Double> getBestCriteria (String criteria, boolean
        isArray, int page) throws DAOException {
        try  {
            MongoCollection<Document> mangaCollection = getCollection(
                COLLECTION_NAME);
            int pageOffset = (page-1)*Constants.PAGE_SIZE;

            List<Bson> pipeline;
            if (isArray) {
                pipeline = List.of(
                        match(and(exists(criteria), ne("average_rating",
                            null))),
                        unwind("$" + criteria),
                        group("$" + criteria, avg("criteria_average_rating",
                             "$average_rating")),
                        sort(descending("criteria_average_rating")),
                        skip(pageOffset),
                        limit(25)
                );
            } else {
                pipeline = List.of(
                        match(Filters.exists(criteria)),
                        group("$" + criteria, avg("criteria_average_rating",
                             "$average_rating")),
                        sort(new Document("criteria_average_rating", -1)),
                        skip(pageOffset),
                        limit(25)
                );
            }

            List <Document> document = mangaCollection.aggregate(pipeline).
                into(new ArrayList<>());
            Map<String, Double> bestCriteria = new LinkedHashMap<>();
            for (Document doc : document) {
                Double avgRating = doc.get("criteria_average_rating")
                    instanceof Integer?
                        doc.getInteger("criteria_average_rating").
                            doubleValue() :
                        doc.getDouble("criteria_average_rating");
                if (criteria.equals("authors")) {
                    bestCriteria.put(doc.get("_id", Document.class).
                        getString("name"), avgRating);
                } else {
                    bestCriteria.put(doc.get("_id").toString(), avgRating);
                }
            }

            return bestCriteria;

        } catch (Exception e) {
            throw new DAOException(DAOExceptionType.GENERIC_ERROR, e.
                getMessage());
        }
    }
}
```

Example code snippet from UserDAONeo4jImpl:

```
{
    /**
    * Retrieves a list of users following a specific user from the Neo4j
        database.
    *
    * @param userId The ID of the user whose followers are to be retrieved.
    * @param loggedUserId The ID of the user requesting the list of
        followers.
    * @return A list of RegisteredUserDTO objects representing the followers
        of the specified user.
    * @throws DAOException If an error occurs while retrieving the followers
        list.
    */
   @Override
   public List<UserSummaryDTO> getFirstNFollowers(String userId, String
       loggedUserId) throws DAOException {
       try (Session session = getSession()) {
           StringBuilder queryBuilder = new StringBuilder("MATCH (follower:
               User)-[:FOLLOWS]->(:User {id: $userId}) ");
           if (loggedUserId != null) {
               queryBuilder.append("WHERE follower.id <> $loggedUserId ");
           }
           queryBuilder.append("RETURN follower AS user ");
           queryBuilder.append("ORDER BY follower.username ");
           queryBuilder.append("LIMIT 10");
           String query = queryBuilder.toString();

           Map<String, Object> params = new HashMap<>();
           params.put("userId", userId);
           if (loggedUserId != null) {
               params.put("loggedUserId", loggedUserId);
           }

           List<Record> records = session.executeRead(
                   tx -> tx.run(query, params).list()
           );

           return records.isEmpty() ? null : records.stream()
                   .map(this::recordToUserSummaryDTO)
                   .toList();

       } catch (Neo4jException e) {
           throw new DAOException(DAOExceptionType.DATABASE_ERROR, e.
               getMessage());

       } catch (Exception e) {
           throw new DAOException(DAOExceptionType.GENERIC_ERROR, e.
               getMessage());
       }
   }
}
```

DTO (Data Transfer Objects)

The DTO modules are the intermediary class between presentation layer and the DAO module in the web application. They transfer data structures between different layers and components of the application in a more standardized way.

Model

- Enums
- Media Content
  - Anime
  - Manga
  - Manga Author
  - Media Content
- Registered User
  - Mangager
  - Registered User
  - User
- Review

Service

Service module has also important role in the web application. The classes in the service module are responsible for containing the business logic and maintaining interaction between the DAO classes and the presentation layer. It handles complex operations with guarantying that the application's core functionalities are executed correctly. Some of the services that are provided in the service module are: *UserService, MediaContentService, ReviewService, TaskManager, ExecuterTaskService.* The package structure of Service module is as follows:

- enums
  - ExecuterTaskService
- exceptions
  - enums
  —— BusinessExceptionType
  - BusinessException
- impl
  - asinc media tasks
  —— CreateMediaTask
  —— DeleteMediaTask
  —— UpdateAverageRatingTask
  —— UpdateMediaRedundancyTask
  —— UpdateMediaTask
  —— UpdateNumberofLikesTask
  - asinc review tasks
  —— RemoveDeletedMediaReviewsTask
  —— RemoveDeletedUserReviewsTask
  —— UpdateReviewRedundancyTask
  - asinc user tasks
  —— CreateUserTask
  —— DeleteUserTask
  —— UpdateNumberOfFollowedTask
  —— UpdateNumberOfFollowersTask
  —- UpdateUserTask
  - AperiodicExecutorTaskServiceImpl
  - ErrorTaskManager
  - MediaContentServiceImpl

- PeriodicExecutorTaskServiceImpl
- ReviewServiceImpl
- UserServiceImpl
- interfaces
—— ExecuterTaskService
—— MediaContentService
—— ReviewService
—— Task
—— TaskManager
—— UserService

- ServiceLocator

Adopted Patterns and Techniques

Conclusion

Conclusion 1

Conclusion 2

Testing

Testing 1

Testing 2