Department of Computer Science
Technical University of Cluj-Napoca

Software Design

Assignnments

Name: Burlea Maria-Cătălina
Group: 30433
burleamariacatalina@gmail.com

Contents

Chapter 1

Objective

The main objective of this assignment is to design and implement an application that features at least two tables in a database, structured in a one-to-many relationship.
The assignment will primarily focus on backend development, using as programming language Java.

The key functionalities of the application will include CRUD (Create, Read, Update, Delete) operations for both tables, allowing for seamless manipulation of data.

For this assignment, I have chosen to design and implement an Art Gallery Management System using Java Spring Framework alongside MySQL for data persistence and management.

The backend structure will be organized into appropriate packages to maintain code modularity and clarity. Since it is a Spring Boot application, the project packages are separated based on their responsibilities (e.g., entities, services, controllers).

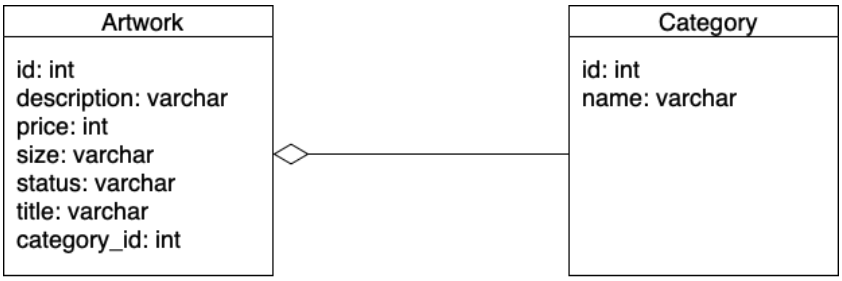
The functionalities will be demonstrated using Postman, showcasing the robustness and reliability of the implemented backend system.

In addition, an authentication mechanism using bearer token has been implemented. As a result, a preliminary step involves initiating a GET request for login, where the user must provide their email and password. Upon successful authentication, a token is generated. Subsequently, if the user is an admin of the application, she/he gain access to a distinct set of actions. Additionally, functionalities such as Sign Up, Change Password, Update, and Get All Users are facilitated.

Chapter 2

Database Summary

The database comprises three primary tables: User, Artwork, and Category. Within this schema, each Category can be associated with multiple Artwork entries.
This relational structure allows for a flexible organization where Artworks can be grouped into distinct categories, facilitating efficient management and retrieval of art pieces based on their classifications.



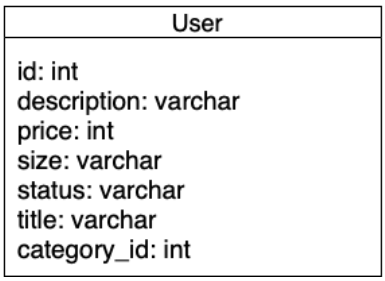


Figure 2.1: Database Schema

Chapter 3

Division into packages

1. **Utils**:
    - It contains an utility class that provide common functionlity across the application.
    - It contains a static method getResponseEntity that generates a ResponseEntity with a custom response message and HTTP status.
    - The class is made final to prevent inheritance, and the constructor is private to prevent instantiation.
2. **Entity**:
    - This package contains entity classes that represent the structure of my database tables, such as User, Artwork, Category.
3. **Service**:
    - This package holds service interfaces that define the business logic operations for the following entities: UserService, ArtworkService, and CategoryService.
4. **ServiceImpl**:
    - Contains the implementations of the service interfaces defined in the service package, such as UserServiceImplementation, ArtworkServiceImplementation, and CategoryServiceImplementation.
5. **Repository**:
    - Holds repository interfaces responsible for database operations, such as UserRepository, ArtworkRepository, and CategoryRepository.
    - The interface serve as bridges between the application's business logic and the database. These interfaces extends the JpaRepository interface provided by **Spring Data JPA**, which offers powerful features for implementing database operations.
6. **Controller**:
    - It contains the following controller interfaces: UserController, ArtworkController, and CategoryController.
7. **ControllerImpl**:
    - Contains the implementations of controller interfaces defined in the controller package, such as UserControllerImplementation, ArtworkControllerImplementation, and CategoryControllerImplementation.

8. **Constants**:
    - Holds set of constants used throughout the application to maintain consistent messaging and error handling.
9. **JWT**:
    - This package contain classes related to security configurations, such as JWT authentication and authorization, as seen in SecurityConfiguration, JWTFilter, and JWTUtil.
10. **Wrapper**:
    - It contains wrapper classes that encapsulate entities or DTOs (Data Transfer Objects), such as UserWrapper and ArtworkWrapper.

Chapter 4

Diagrams

4.1 UML Class Diagram

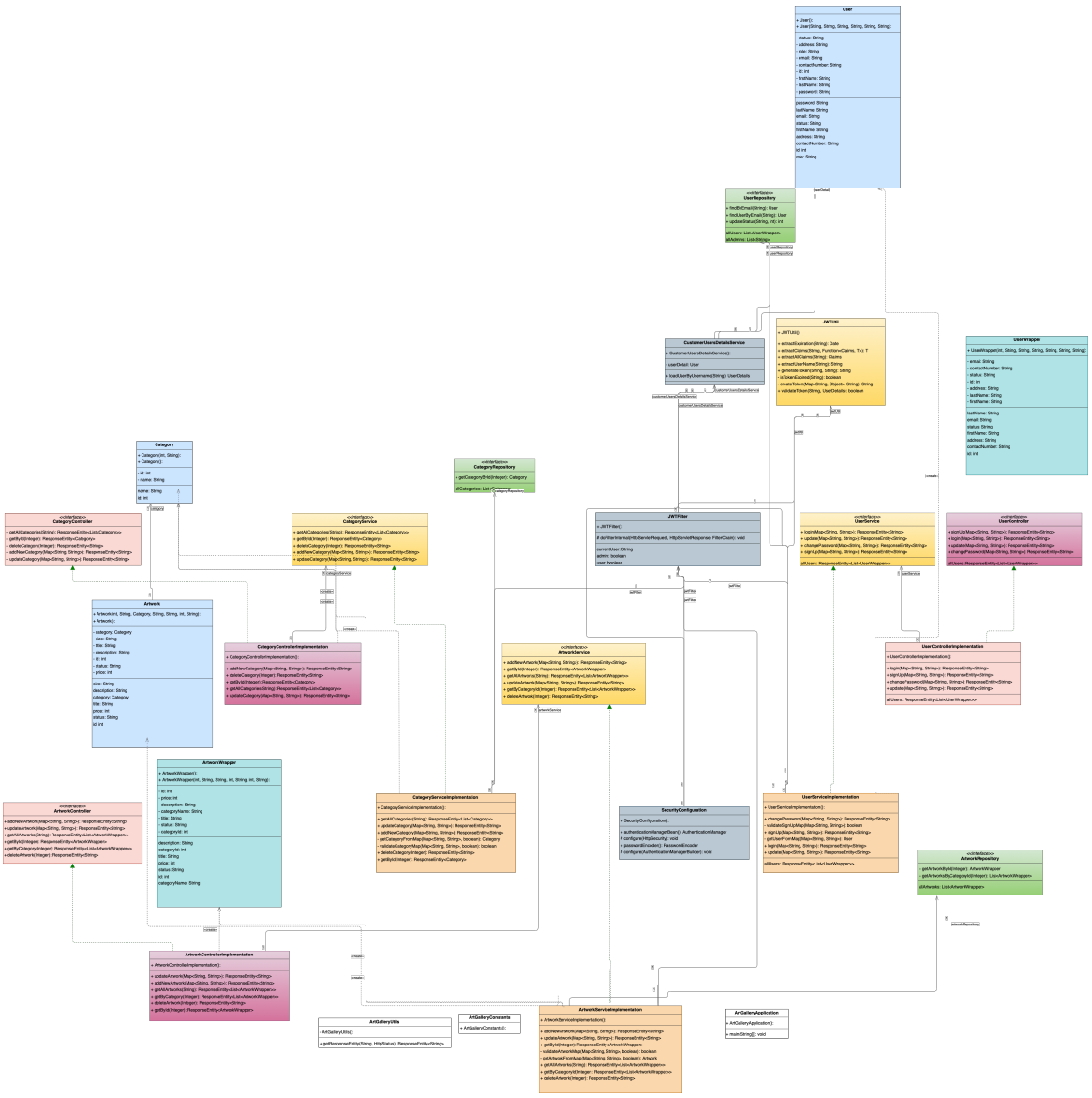


Figure 4.1: UML Class Diagram

4.2 Use Case Diagram

We have two types of users in our system: regular users and admins.

Regular users have the capability to Sign Up and Login. Once logged in, they can access details about artworks and categories based on their respective IDs, view all available categories and artworks, and change their password.

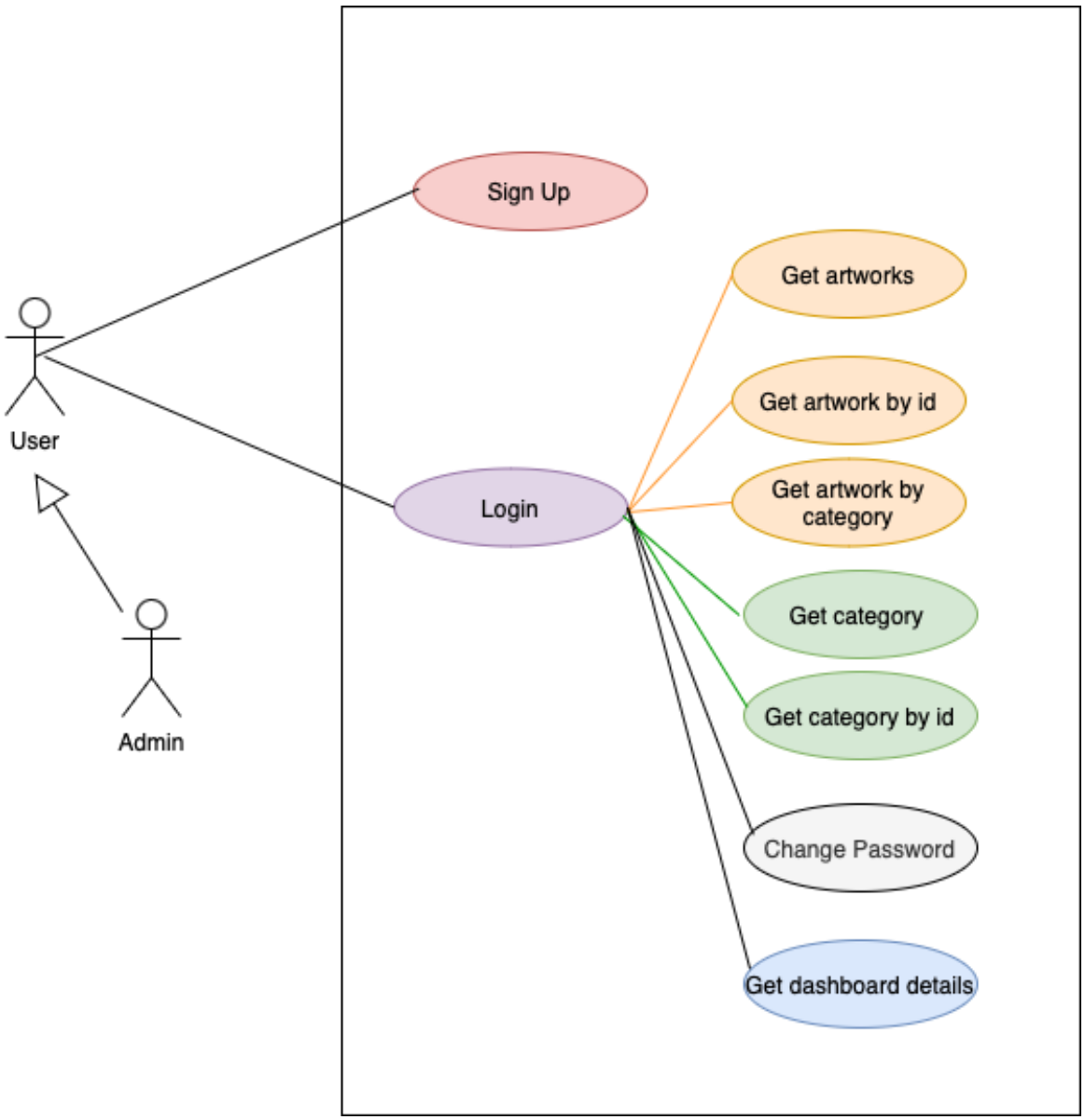


Figure 4.2: User use case diagram

On the other hand, admins, in addition to the functionalities available to regular users, have the authority to add, update, and delete artworks or categories. They can also view a list of all administrators' emails, access user information, and have the ability to update or delete user accounts after logging in.

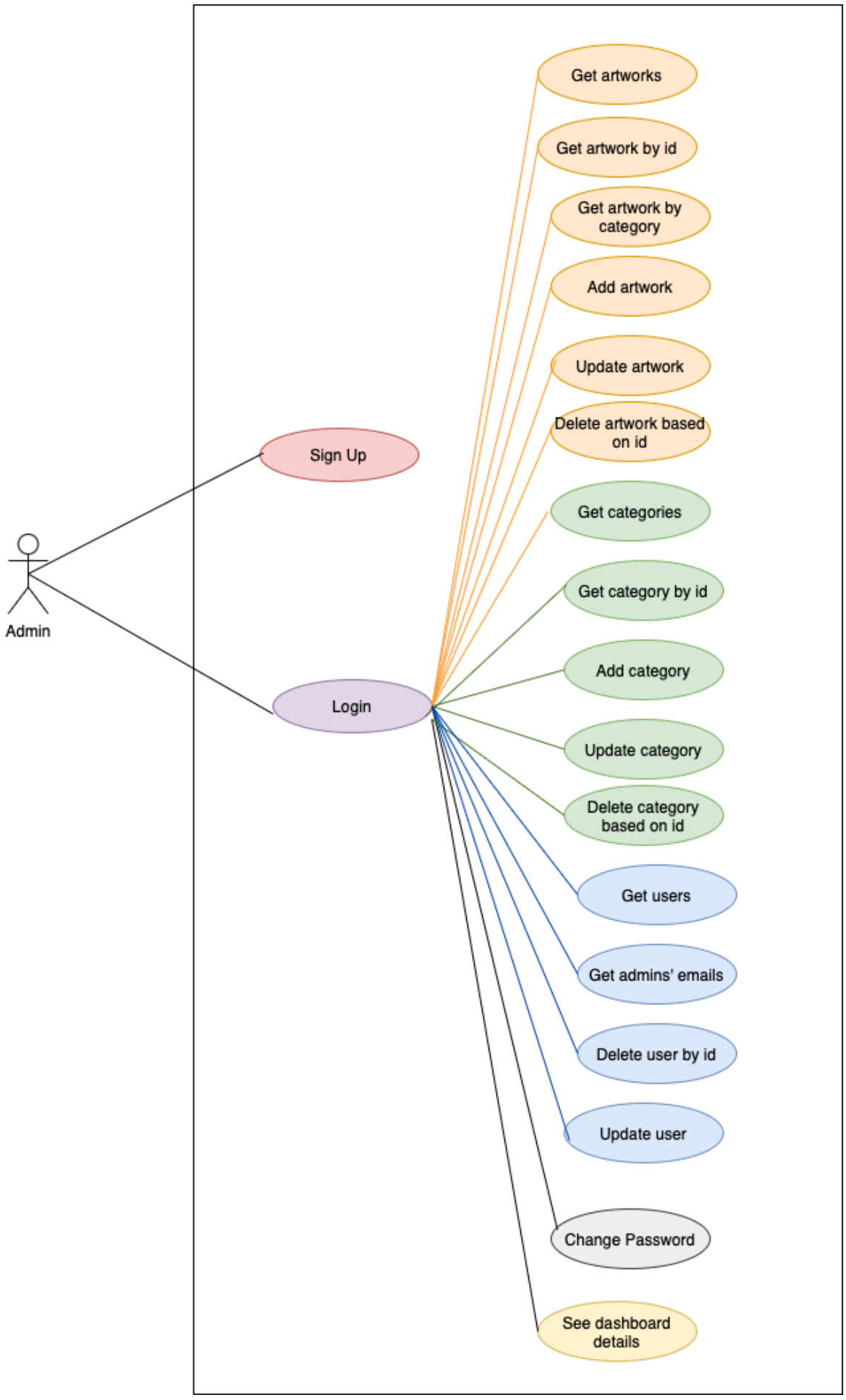


Figure 4.3: Admin use case diagram

Chapter 5

Bibliography

- Create a REST API in Spring Boot: https://medium.com/java-content-hub/creating-a-rest-a
- Building REST services with Spring: https://spring.io/guides/tutorials/rest
- Online Tutorial: https://www.youtube.com/watch?v=d1MfY7MpX4c&list=PLdRq0mbeEBmwdwZF3lWwCcWmD76GfEFVT pp=iAQB
- Validation in Spring Boot : https://reflectoring.io/bean-validation-with-spring-boot/
- Versioning Problems for Maven: https://chat.openai.com/