UNIVERSITATEA TEHNICĂ DIN CLUJ-NAPOCA

Perfume webstore

Proiectare Software

Nume: Muresan Ioana Danina
Grupa: 30236

Facultatea de Automatica
si Calculatoare

28 Martie 2023

Cuprins

1 Deliverable 1

1.1 Project Specification

The objective of this project is to develop a fully functional online perfume store using the Spring Framework. The store will allow customers to browse and purchase different perfumes online.

User Roles: The online store will have two types of users:

- **Customer:** can browse and purchase perfumes online and view and add a review.
- **Administrator:** can manage the product catalog, manage orders.

Features

- Customers can search for the product according to the specified criteria.
- Customers can add and delete products from the shopping cart.
- Customers can order the products in the shopping cart.
- Customers can change their password and view their orders.
- Admin can add or modify a product.
- Admin can change the data of any user.
- Admin can view orders of all users.

1.2 Functional Requirements

1. **User registration and login**: The system should allow users to register and create an account to browse and purchase products. Users should be able to log in to their account using their email and password.
2. **Product Catalog**: The system should display a catalog of perfumes with their details like name, brand, price, fragrance notes, size, and image. Users should be able to search and filter products by their attributes.
3. **Product Detail Page**: The system should display detailed information about each product, including product images, descriptions, reviews, ratings, and related products.
4. **Shopping Cart**: The system should allow users to add products to their cart and proceed to checkout or continue shopping. Users should be able to edit and remove items from their cart.
5. **Order Management**: The system should allow users to track their orders, view order history, and manage their shipping and billing addresses.
6. **Review**: The system should allow users to provide reviews about the products. The store owner should be able to manage and moderate the reviews.
7. **Mobile-Friendly**: The system should be mobile-friendly to allow users to browse and shop on their mobile devices.

1.3 Use Case Model 1

1.3.1 Use Cases Identification

- Browse Products: The user wants to browse the catalog of perfumes and filter the products based on various attributes like brand, price, and fragrance notes.
- Search Products: The user wants to search for a specific perfume by name or keyword.
- Add to Cart: The user wants to add a product to the shopping cart and proceed to checkout.

- Edit Cart: The user wants to edit the items in the shopping cart, including changing the quantity or removing items.
- View Order History: The user wants to view the list of orders placed and their status.
- Provide Review: The user wants to provide review about the product.
- Manage inventory: The admin can use the store's inventory management system to add, remove or update product information, such as product name, description, price, and quantity.
- Manage order : The admin

1.3.2 UML Use Case Diagrams

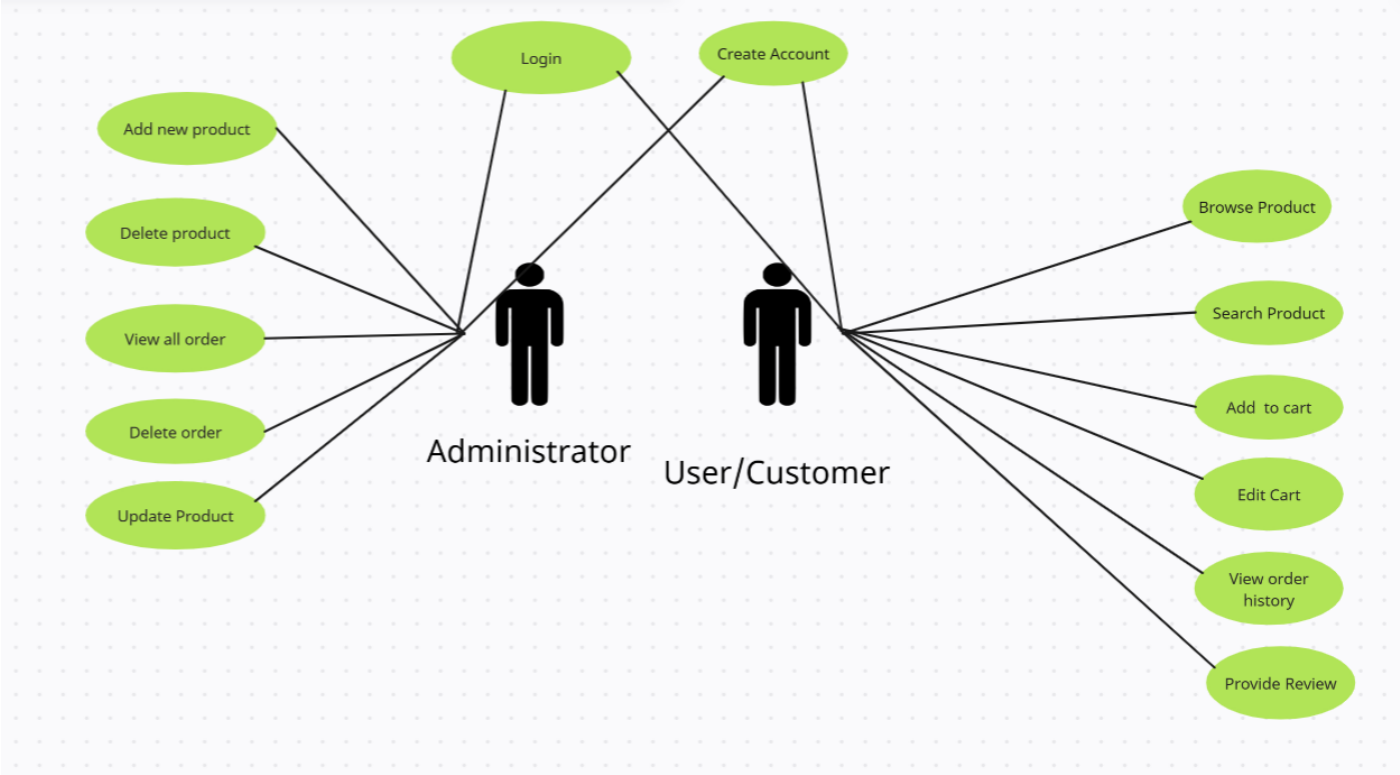

Figura 1: Use Case Diagrams

1.4 Supplementary Specification

1.4.1 Non-functional Requirements

Non-functional requirements are the aspects of a system that specify how well the system should perform rather than what the system should do. Here are some potential non-functional requirements for a perfume online store:

- Performance: The website should load quickly and respond to user input promptly to ensure a smooth shopping experience for customers.
- Availability: The website should be available 24/7, with minimal downtime or maintenance periods, to ensure customers can access the store whenever they want.
- Security: The website should be secure to protect customer data, such as personal information and payment details, from unauthorized access and cyber attacks.
- Reliability: The website should be reliable and stable, with minimal errors or bugs, to ensure customers can complete their purchases without disruption.

- Usability: The website should be easy to use and navigate, with a clear and intuitive user interface that meets the needs of customers.
- Compatibility: The website should be compatible with a range of devices and browsers, including mobile devices, to ensure customers can access the store from any device they choose.

Overall, these non-functional requirements are important to ensure the online perfume store provides a high-quality user experience, protects customer data, and maintains the store's reputation and profitability.

1.4.2 Design Constraints

The application is developed using Java programming language and Spring framework, which is integrated with a MySQL database. The data is fetched and tested using Postman. The application runs on localhost on port 8083. The user interface is created using Angular, ensuring a modern and user-friendly interface. The data security measures are implemented using Spring Boot, providing a secure and reliable platform for data management.

1.4.3 Glossary

Here are some terms that may be used in the context of an online perfume store:

- Inventory management system: A software system used to manage and track the store's inventory, including product information, quantity, and availability.
- Performance: The measure of how well the website functions, including speed, responsiveness, and stability.
- Reliability: The measure of how well the website performs, including the frequency and severity of errors or bugs.
- Usability: The measure of how easy it is for users to navigate and use the website, including the user interface and design.
- Compatibility: The measure of how well the website functions on different devices and browsers.
- Review: A review is an evaluation or assessment of something, such as a product, service, or performance. A review can be positive, negative, or neutral, and it typically involves an analysis of the strengths and weaknesses of the item being reviewed

2 Deliverable 2

2.1 Domain Model

A domain model is a conceptual representation of a system, describing the entities and their relationships within the system.

Entities:

- User: a person who interacts with the website and can purchase perfumes and leave reviews.
- Order: a record of a user's purchase, including selected perfumes and shipping information.
- OrderItem: a record of a specific perfume and quantity purchased in an order.
- Perfume: a product available for purchase on the website.
- Review: a user's feedback on a specific perfume.

Relationships:

- A User can place Orders.
- An Order is associated with a User and contains one or more OrderItems.
- An OrderItem is associated with an Order and a Perfume, and contains a quantity.
- A Perfume can have multiple Reviews.
- A Review is associated with a Perfume.

Class diagrams

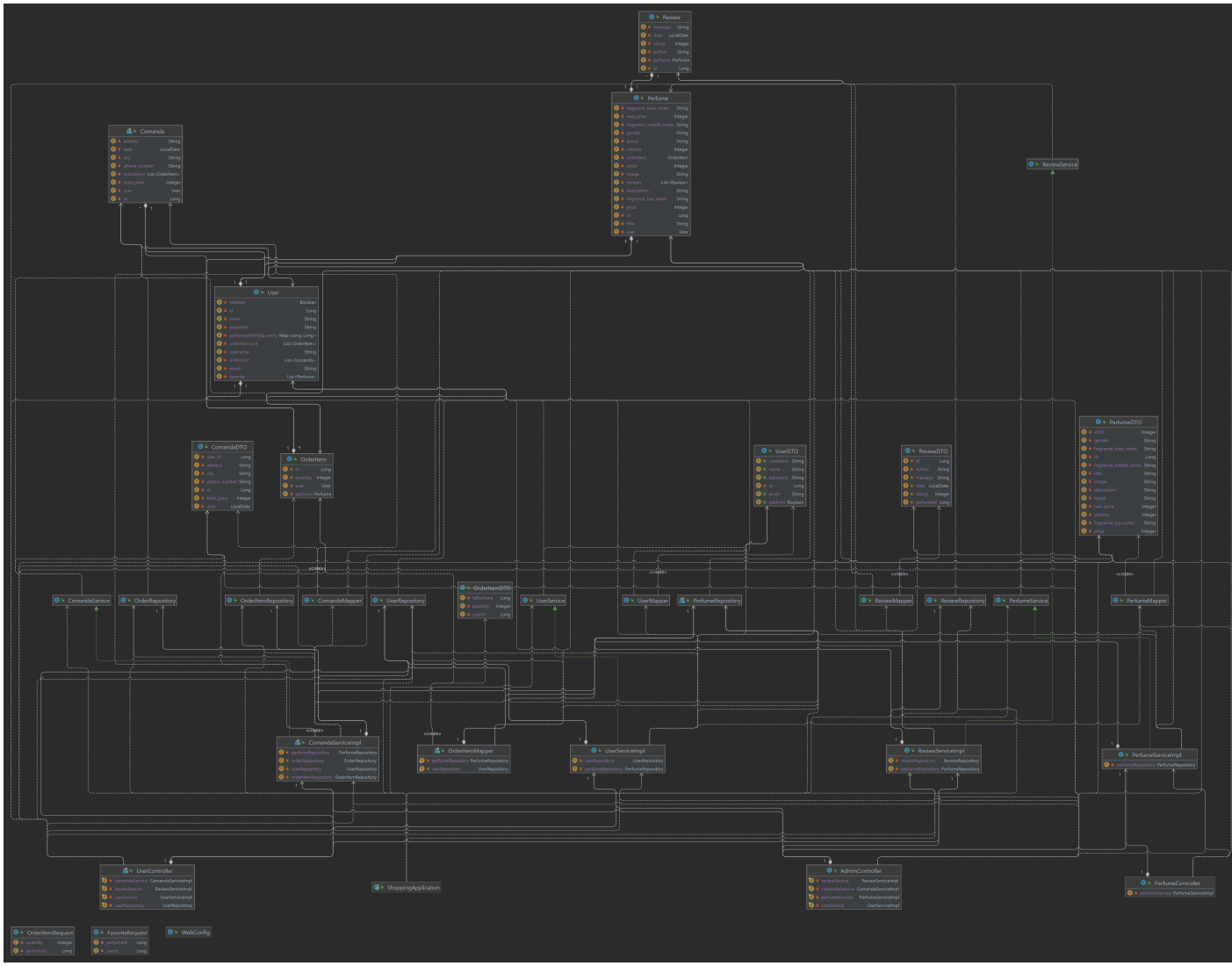

Figura 2: Class diagrams

2.2 Project Specification

2.2.1 Conceptual Architecture

Model-View-Controller (MVC) pattern

Spring is a popular Java framework that is widely used to develop web applications. The conceptual architecture used by Spring is based on the Model-View-Controller (MVC) design pattern. The MVC pattern separates the application into three main components: the model, the view, and the controller.

Model: The model represents the data and business logic of the application. It is responsible for managing the application's data and providing access to it. In Spring, the model is typically implemented using Plain Old Java Objects (POJOs) that are annotated with the appropriate Spring annotations.

View: The view represents the user interface of the application. It is responsible for displaying the data to the user and receiving user input. In Spring, the view is typically implemented using HTML, CSS, and JavaScript.

Controller: The controller acts as an intermediary between the model and the view. It is responsible for handling user requests and updating the model and view accordingly. In Spring, the controller is typically implemented using Spring MVC, which provides a set of annotations and classes that make it easy to map requests to methods in the controller.

2.2.2 Package Design

Package diagrams

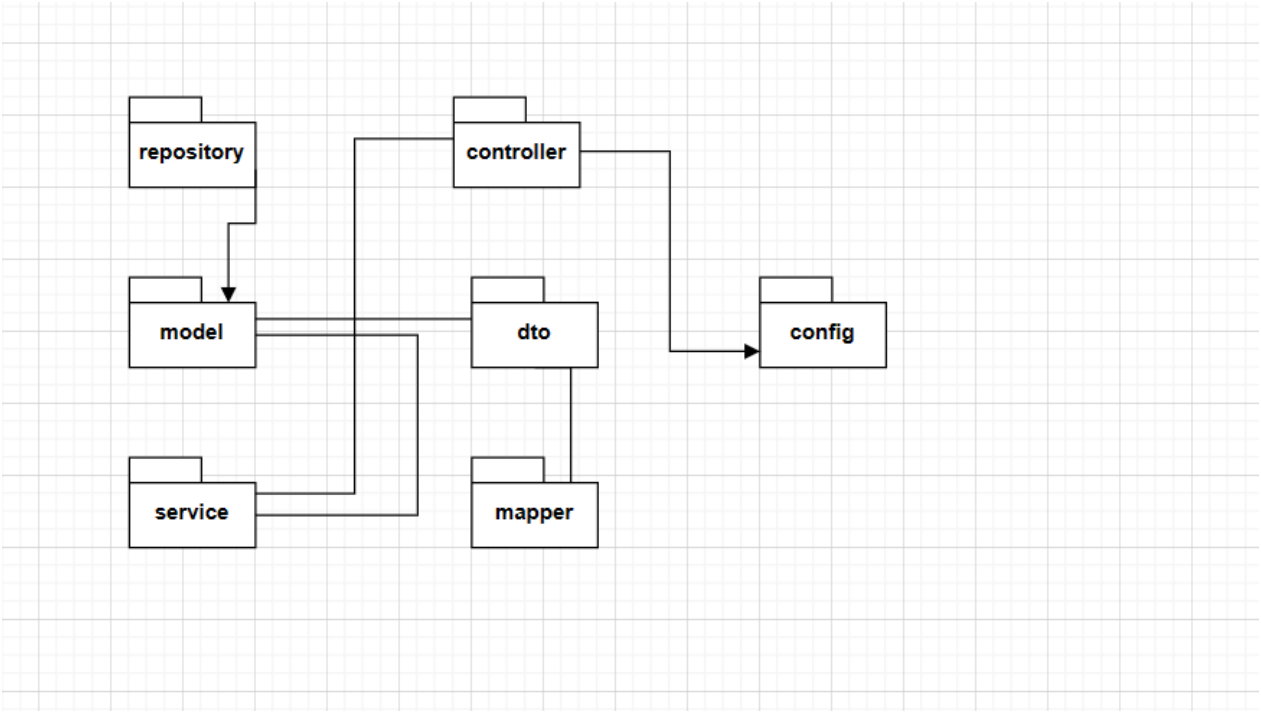

Figura 3: Package diagrams

2.2.3 Component and Deployment Diagram

Component Diagram

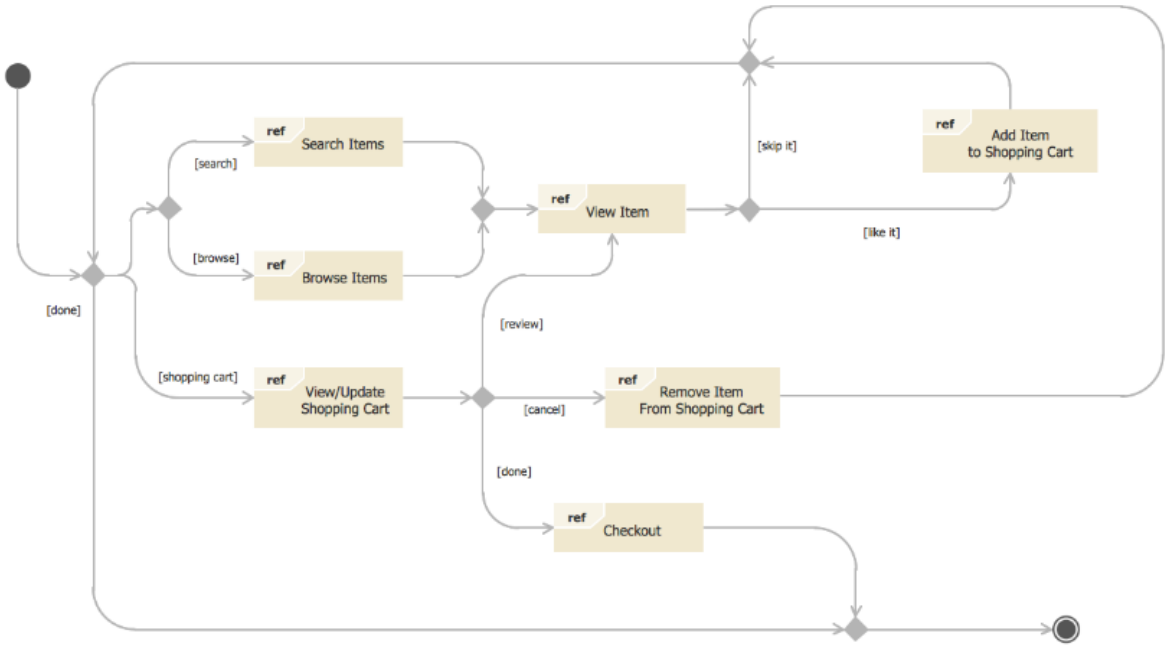

Figura 4: Component diagram

Deployment Diagram

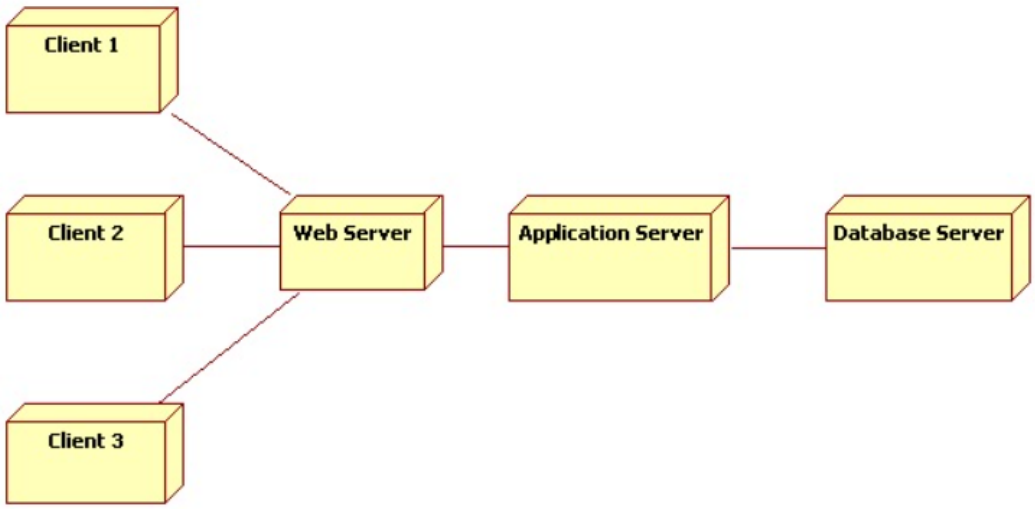

Figura 5: Deployment diagram

3 Deliverable 3

3.1 Design Model

3.1.1 Dynamic Behavior

Sequence Diagram

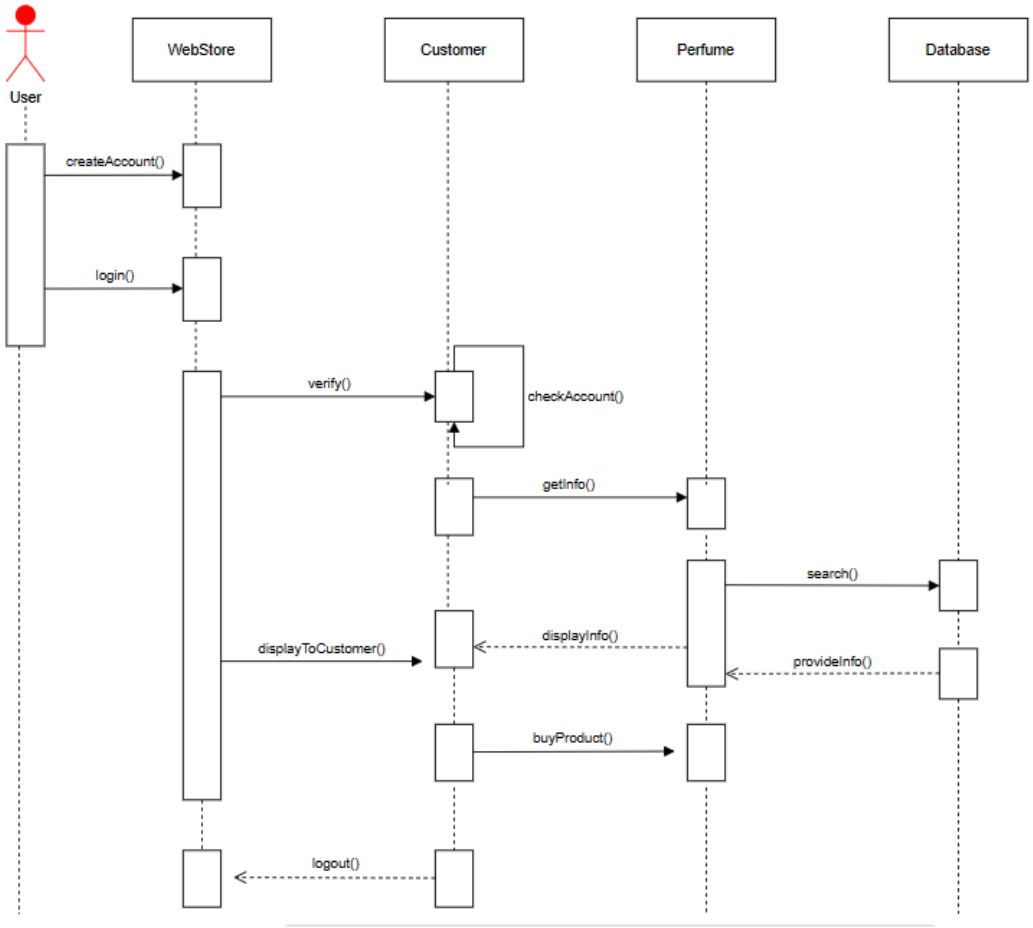

Figura 6: Sequence diagram

Communication Diagram

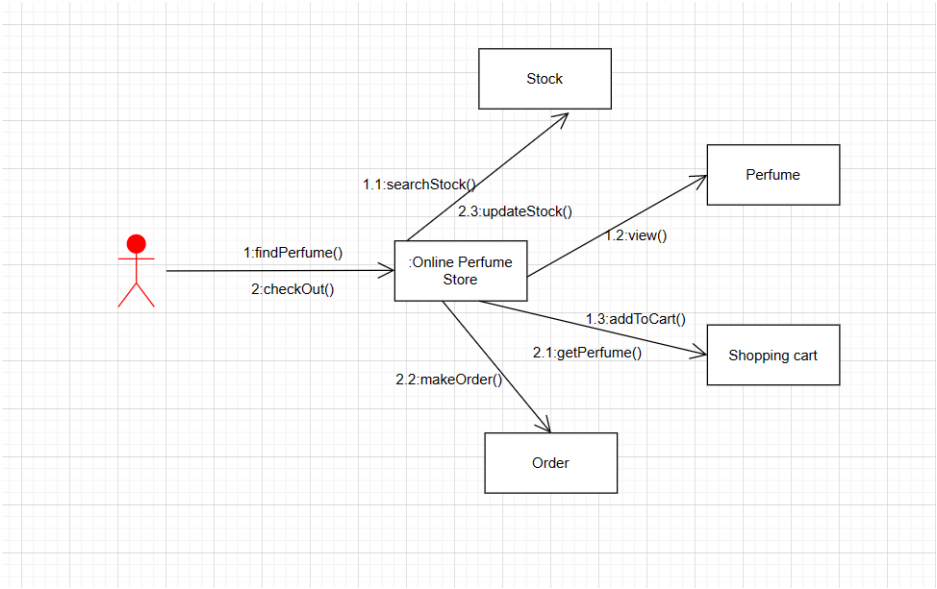

Figura 7: Communication diagram

3.1.2 Class Digram

3.2 Data Model

User:

Represents a user or customer of the web store. Contains attributes such as ID, name, username, password, email, isAdmin (indicating if the user is an administrator), loginDate (timestamp of the last login).

Has relationships:

One-to-many relationship with Perfume: Represents the favorite perfumes of the user.

One-to-many relationship with Comanda: Represents the orders placed by the user.

One-to-many relationship with OrderItem: Represents the order items associated with the user.

Many-to-many relationship with Perfume: Represents the perfumes added to the user's cart with their quantities.

Review:

Represents a review given by a user for a particular perfume. Contains attributes such as ID, author, message, date, rating.

Has a many-to-one relationship with Perfume: Represents the perfume for which the review is given.

Perfume:

Represents a perfume available in the web store. Contains attributes such as ID, title, gender, brand, fragrance top/middle/base notes, description, price, newprice (if applicable), volume, stock, image.

Has relationships:

One-to-many relationship with Review: Represents the reviews associated with the perfume.

One-to-one relationship with OrderItem: Represents the order item associated with the perfume.

Many-to-one relationship with User: Represents the user who added the perfume to their favorite list.

OrderItem:

Represents an item within an order.

Contains attributes such as ID, quantity.

Has a many-to-one relationship with Perfume: Represents the perfume associated with the order item.

Has a many-to-one relationship with User: Represents the user who placed the order.

Comanda (Order):

Represents an order placed by a user.

Contains attributes such as ID, totalprice, city, address, phonenumber, date, transport.

Has relationships:

Many-to-one relationship with User: Represents the user who placed the order.

One-to-many relationship with OrderItem: Represents the order items associated with the order.

Data Model

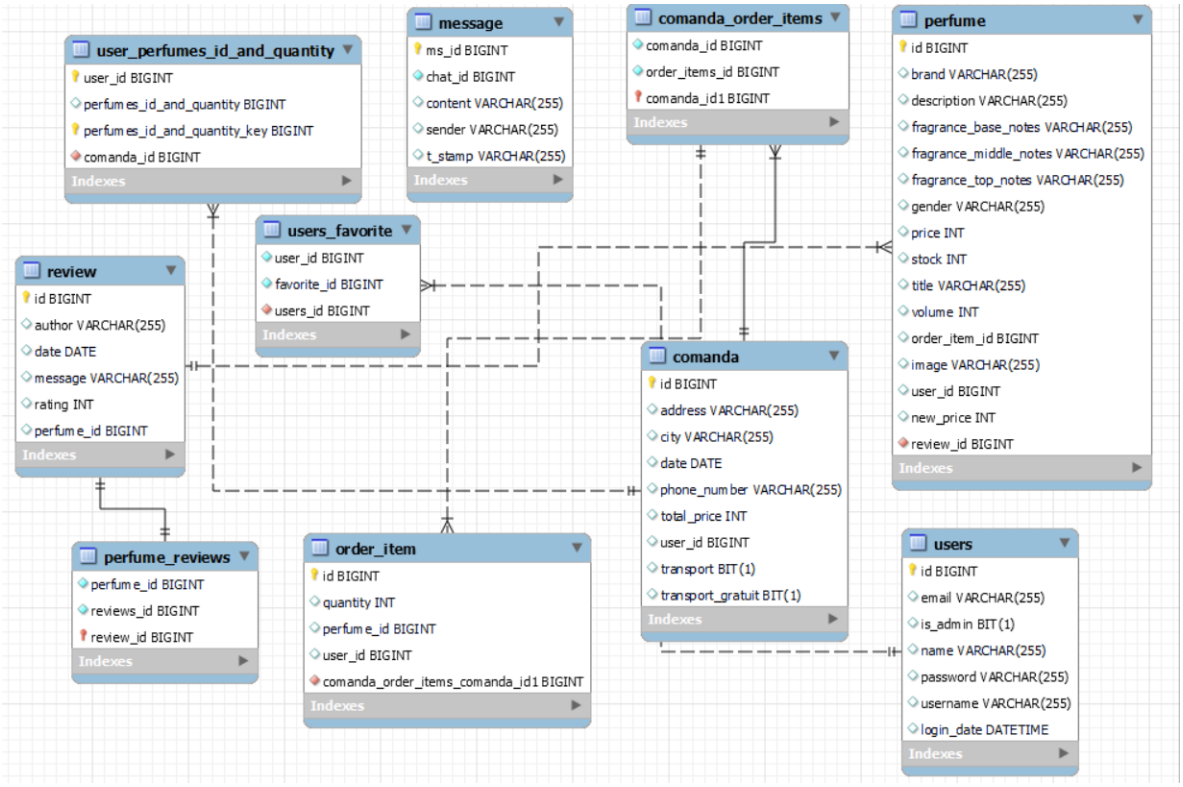

Figura 8: Data Model

4 System Testing

In the perfume webstore project, system testing played a crucial role in ensuring the reliability and functionality of the application. Mockito, a popular mocking framework, was employed to facilitate the testing of key components and interactions within the system.

The given example demonstrates a system test for the deletePerfume method in the PerfumeServiceImpl class.

```
@Test
public void testDeletePerfume() {
    // Arrange
    Long perfumeId = 1L;
    Perfume perfume = new Perfume();
    perfume.setId(perfumeId);
    PerfumeRepository repositoryMock = mock(PerfumeRepository.class);
    when(repositoryMock.findById(perfumeId)).thenReturn(Optional.of(perfume));
    PerfumeServiceImpl service = new PerfumeServiceImpl(repositoryMock);

    // Act
    String result = service.deletePerfume(perfumeId);

    // Assert
    assertEquals( expected: "Perfume deleted successfully", result);
}
```

Figura 9: DeletePerfume Test

The given example demonstrates a unit test for the getUserById method in the UserService class.

```java
@Test
void testGetUserById() {
    Long userId = 1L;
    User user = new User(userId, name: "John", password: "password", email: "john@example.com", isAdmin: false);
    when(userRepository.findById(userId)).thenReturn(Optional.of(user));

    Optional<User> result = userService.getUserById(userId);
    Assertions.assertTrue(result.isPresent());
    assertEquals(user, result.get());
}
```

Figura 10: GetUserById Test

5 Future Improvements

Future Improvements:

- **Enhanced User Experience:** To further enhance the user experience of the perfume webstore, additional features and improvements can be implemented. For instance, incorporating personalized recommendations based on user preferences, previous purchases, or trending perfumes can greatly enhance user engagement. Implementing a seamless and intuitive user interface with responsive design for mobile devices would also broaden the accessibility and convenience for users.
- **Integration with Social Media:** Integrating the perfume webstore with social media platforms can expand its reach and increase customer engagement. Adding social sharing options, allowing users to share their favorite perfumes or purchases with their social networks, can provide valuable word-of-mouth marketing.
- **Enhanced Security Measures:** Strengthening the security measures of the webstore is crucial to protect user data and ensure a secure online shopping environment. Implementing measures such as two-factor authentication, encryption of sensitive user information, and regular security audits can enhance the overall security posture of the application and instill trust in users.
- **Inventory Management and Order Tracking:** Implementing robust inventory management capabilities will enable efficient tracking of perfume stock levels, automated notifications for low stock items, and seamless order fulfillment. Additionally, integrating order tracking functionality, allowing users to track their orders in real-time, will provide transparency and enhance the overall customer experience.

6 Conclusion

In conclusion, the perfume webstore project successfully developed a user-friendly e-commerce platform using Angular and Spring technologies. The web application offers essential features such as user registration, product browsing, shopping cart management, and order processing. The integration of Mockito for system testing ensured the reliability of the application. Overall, the project demonstrates the capabilities of Angular and Spring in creating a functional and visually appealing perfume webstore.

7 Bibliography

https://spring.io/projects/spring-framework

https://www.baeldung.com/spring-security-registration-password-encoding-bcrypt

https://spring.io/guides/gs/messaging-stomp-websocket/
https://www.youtube.com/watchv=aCZmPgBHc88&ab_channel=JavaGuides
https://www.youtube.com/user/angularjs