UNIVERSITATEA TEHNICĂ
DIN CLUJ-NAPOCA

Clothing Brand
E-Commerce Website

Name: Sarkozi Lorand
Group: 30234

Table of Contents

Deliverable 1

Project Specification

Create a Website for my clothing brand which will allow customers to put clothing products in their cart, view their shopping cart and their shopping history, and make orders.

Functional Requirements

1. **User Authentication:**
   - Users should be able to create accounts or log in using email/password.
2. **Product Browsing:**
   - Users can browse through different categories of clothing products (e.g., t-shirts, hoodies) with images, descriptions, and prices displayed.
3. **Product Search:**
   - Users can search for specific products using keywords or filters like size, color, etc.
4. **Shopping Cart Management:**
   - Users can add/remove items from their shopping cart.
   - The system should calculate the total cost of items in the cart.
5. **Checkout Process:**
   - Users can proceed to checkout securely.
   - Users should provide shipping and billing information.
   - Integration with payment gateways for secure transactions.
6. **Order Management:**
   - Users and administrators can view order history and status.
   - Administrators can update order status (e.g., processing, shipped).
7. **Content Management:**
   - Administrators can add, edit, or delete product listings.
   - Administrators can update website content (e.g., About Us page, Contact Us page).
8. **User Communication:**
   - Users can contact customer support via email or contact form.

Use Case Model 1

Use Cases Identification

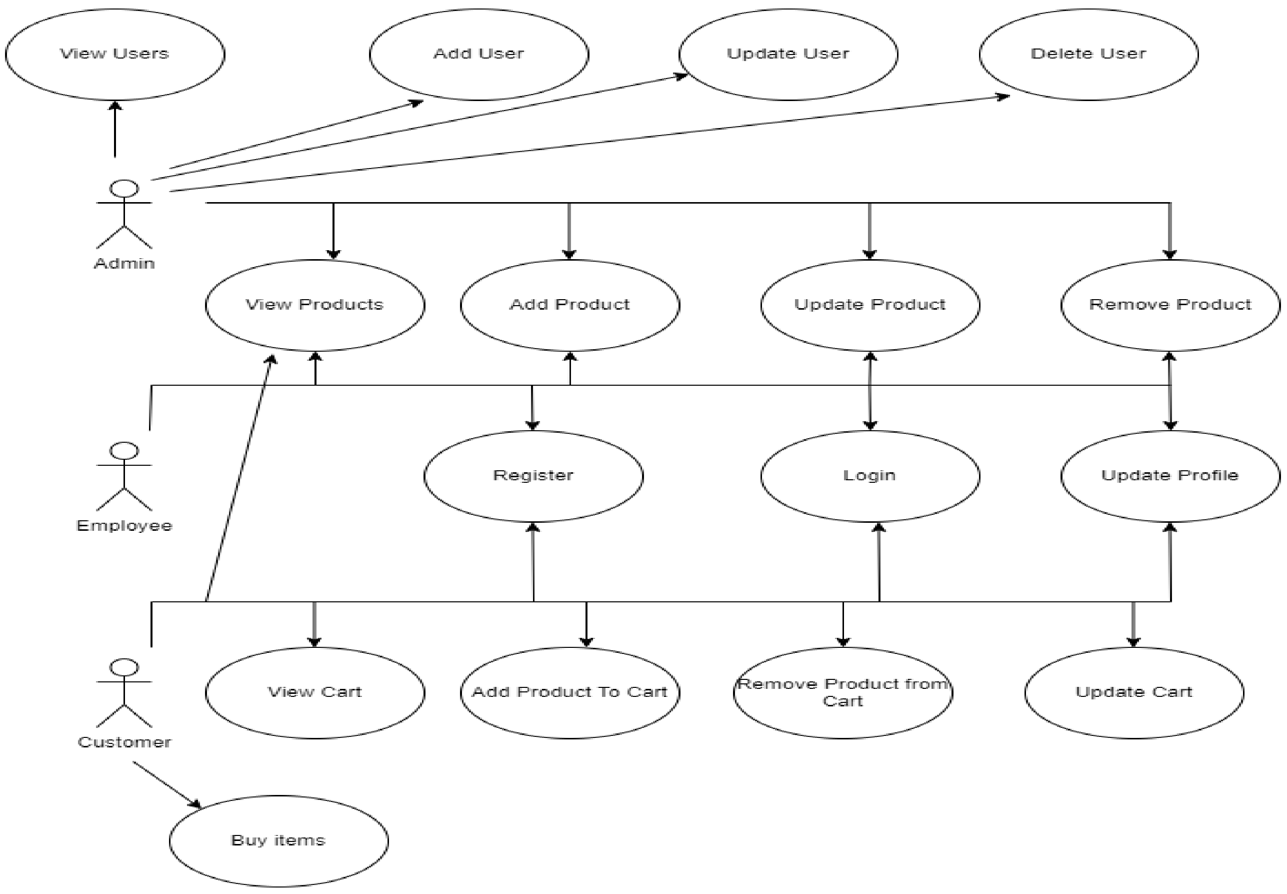
***Use-Case**: Browse Products*
***Level**: User*
***Primary Actor**: Customer*
***Main success scenario**:*

- *Customer accesses the website.*
- *Customer navigates to the "Products" section.*
- *Customer selects a category or uses search functionality.*
- *Customer views product listings with details.*
- *Customer adds desired products to the shopping cart.*

Extensions:

- *If the website is experiencing technical issues, display an error message and prompt the customer to try again later.*

UML Use Case Diagrams



Supplementary Specification

Non-functional Requirements

- *Performance: The website must load within 3 seconds to ensure a seamless user experience and reduce bounce rates.*

- *Security: Implement HTTPS encryption and secure payment gateways to protect user data and transactions, ensuring trust and compliance with regulations.*

- *Scalability: The website infrastructure must be scalable to accommodate increasing traffic and product catalog size, supporting business growth without performance degradation.*

- *Reliability: The website should have an uptime of at least 99.9% to ensure continuous availability, minimizing downtime and maximizing customer satisfaction.*

Design Constraints

- *Technology Stack: The project is constrained to be built using Java, specifically leveraging the Spring Boot framework for backend development. All server-side logic, RESTful API endpoints, and data handling must be implemented in Java using Spring Boot.*

- *Thymeleaf Templating Engine: The project utilizes Thymeleaf as the templating engine for server-side rendering of HTML views. All frontend templates and dynamic content rendering must be implemented using Thymeleaf syntax within the Java Spring Boot application.*

- *Model-View-Controller (MVC) Architecture: The project follows the MVC architectural pattern, with Spring Boot serving as the backend controller to handle HTTP requests and responses. Thymeleaf templates are used for rendering views (HTML) based on data provided by the controller.*

- *Dependency Injection: Spring Boot's dependency injection mechanism must be utilized for managing dependencies and wiring components within the application, promoting loose coupling and enhancing maintainability.*

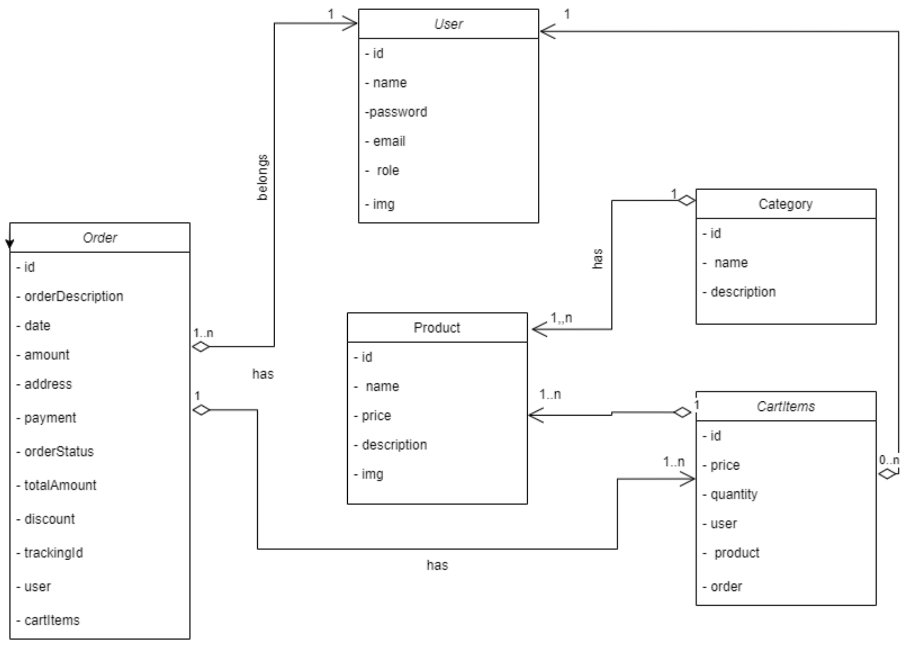
- *RESTful API Design: If applicable, any client-server communication or external integrations should follow RESTful API design principles, leveraging Spring Boot's capabilities for creating and consuming RESTful services.*

Glossary

[Present the noteworthy terms and their definition, format and validation rules if appropriate.]

Deliverable 2

Domain Model

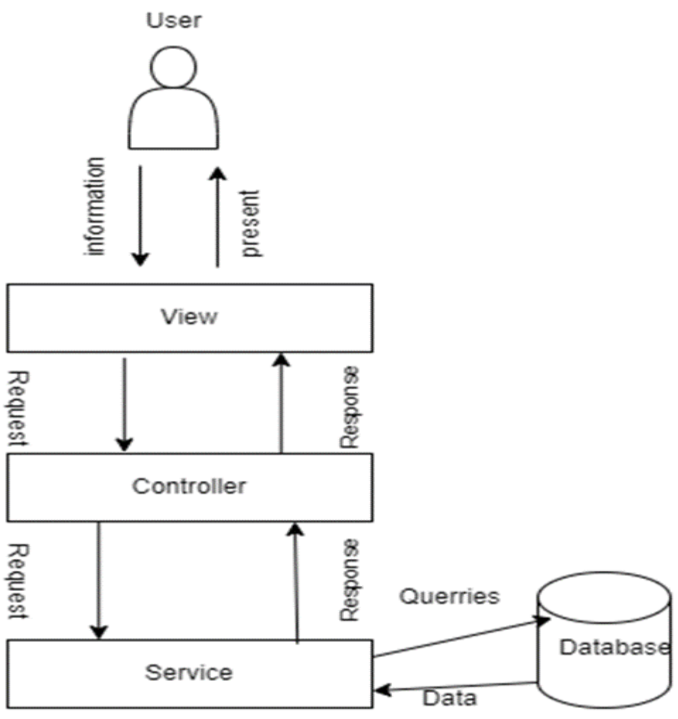


Domain Model Diagram

Architectural Design

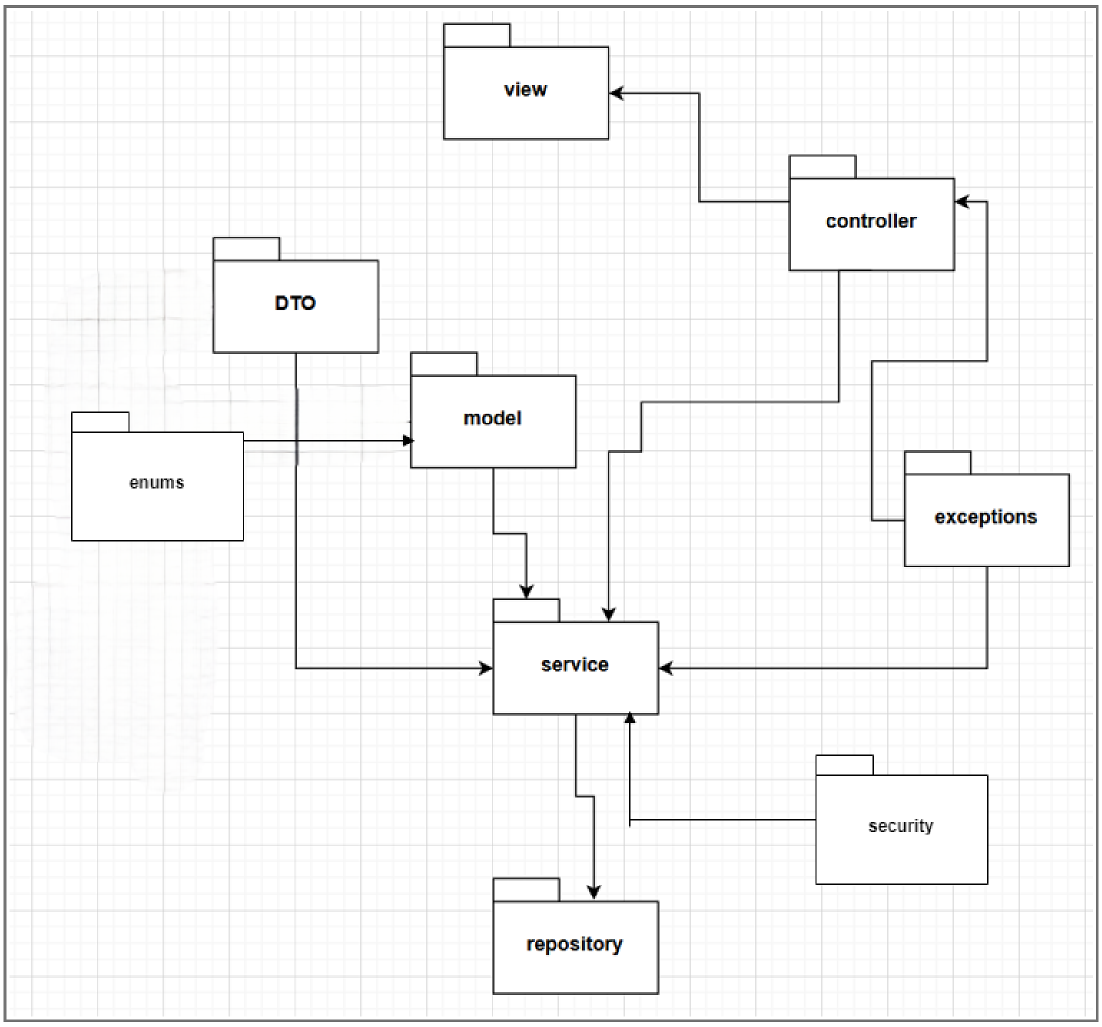
Conceptual Architecture

In my project I used as Conceptual Architecture the Layers Architecture which is represented in the following Diagram:



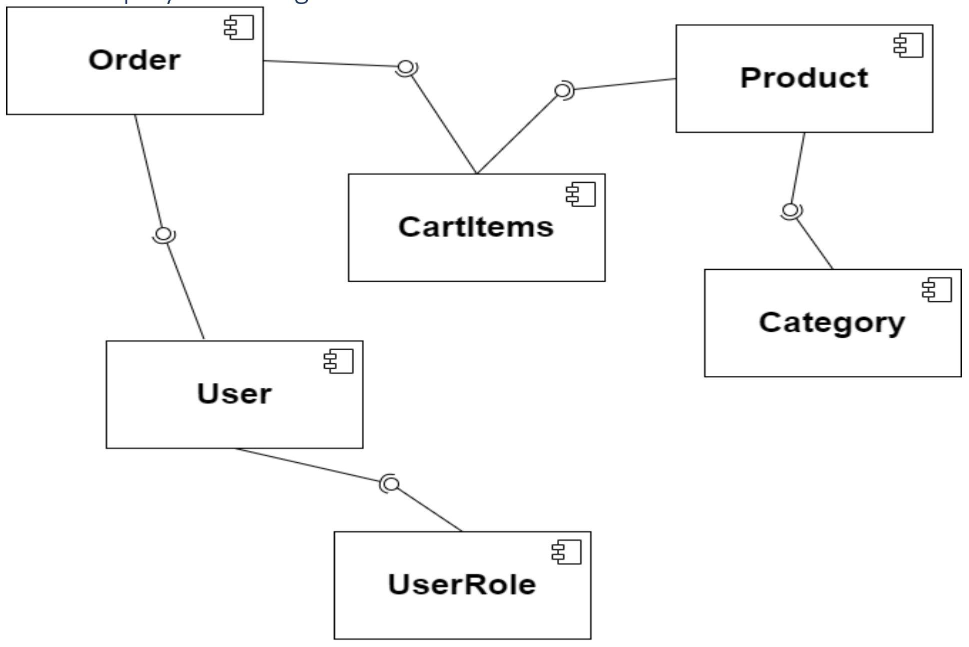
Layers Architecture Diagram

Package Design



Package Diagram

Component and Deployment Diagram



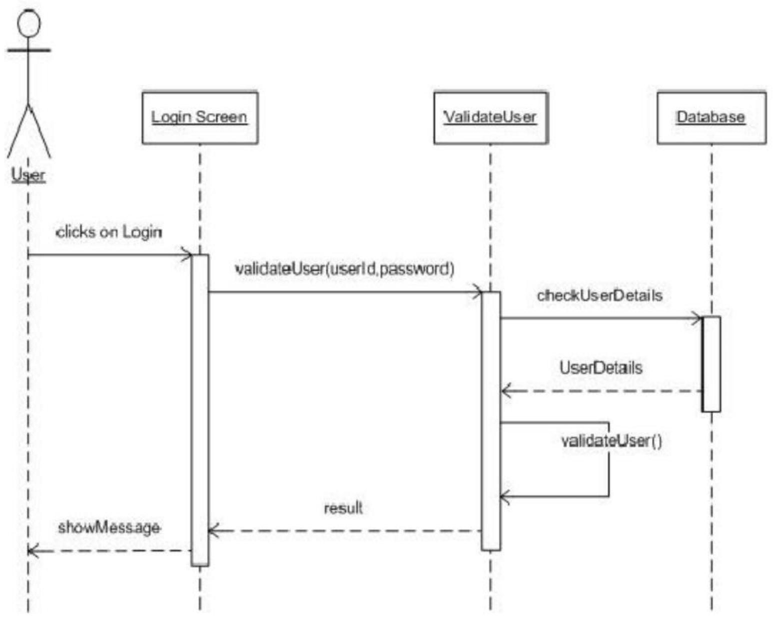
Component and Deployment Diagram

Deliverable 3

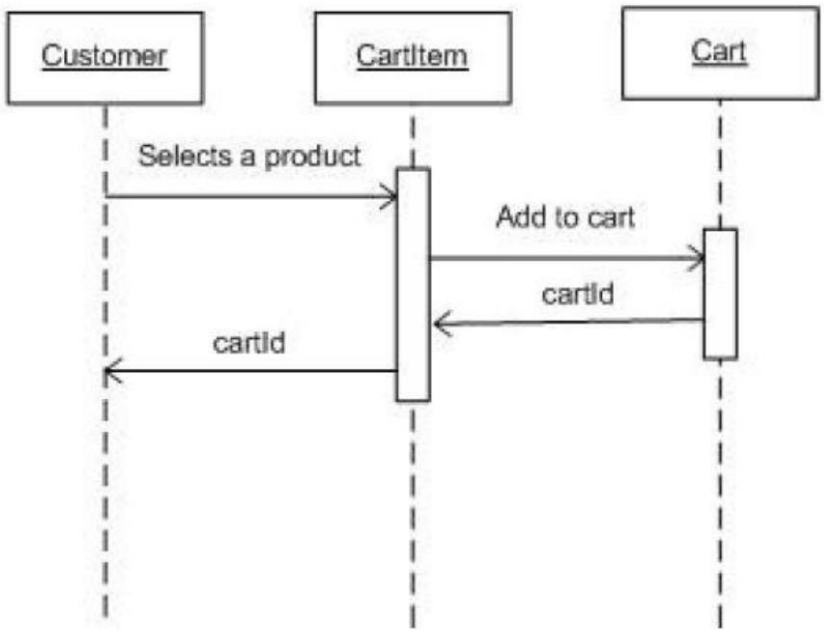
Design Model

Dynamic Behavior

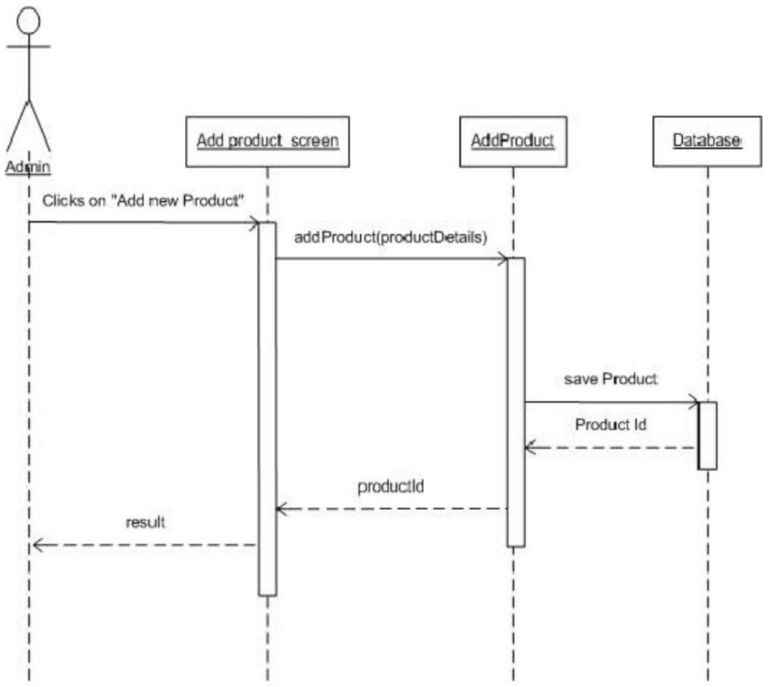
LogIn for User*: The login is made in the login screen that appears for any users that accesses the website. In the login screen the user will put in hi email and password, and when clicking on the login button it s gonna check if the user email and passworsd is valid in the database, and if it s valid then with the help of the authorization it s gonna check for his role and based on the users role he will be authorized as an admin or customer. The users id, role, and an auto-generated token will be saved in the localStorage .*



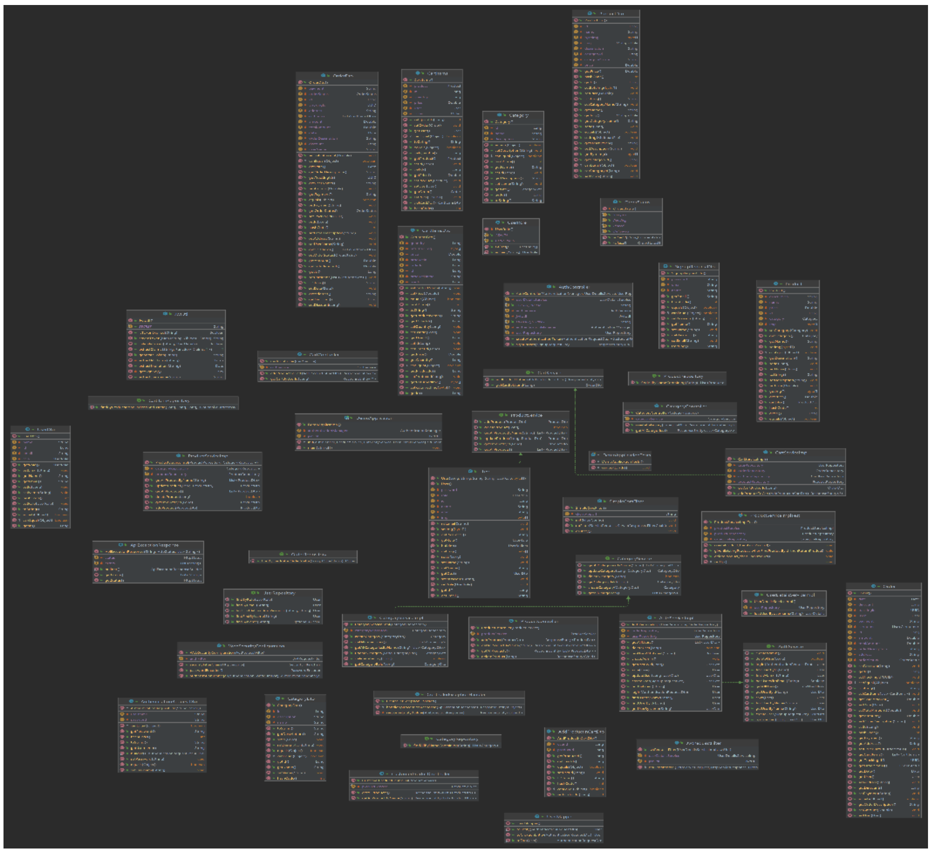
***Add Product to Cart** : The Add to cart function is available just for the customer, and it can be done while logging in as a customer. When you log in on the dashboard page the customer can see all the products available. Under every product there is a button for adding the product in the cart. When you click on it it selects the products with it s product id and it adds it to the logged In users cart. The user has a list of cartItems and whenever he clicks on a new item it will be added in the cartItems list.*



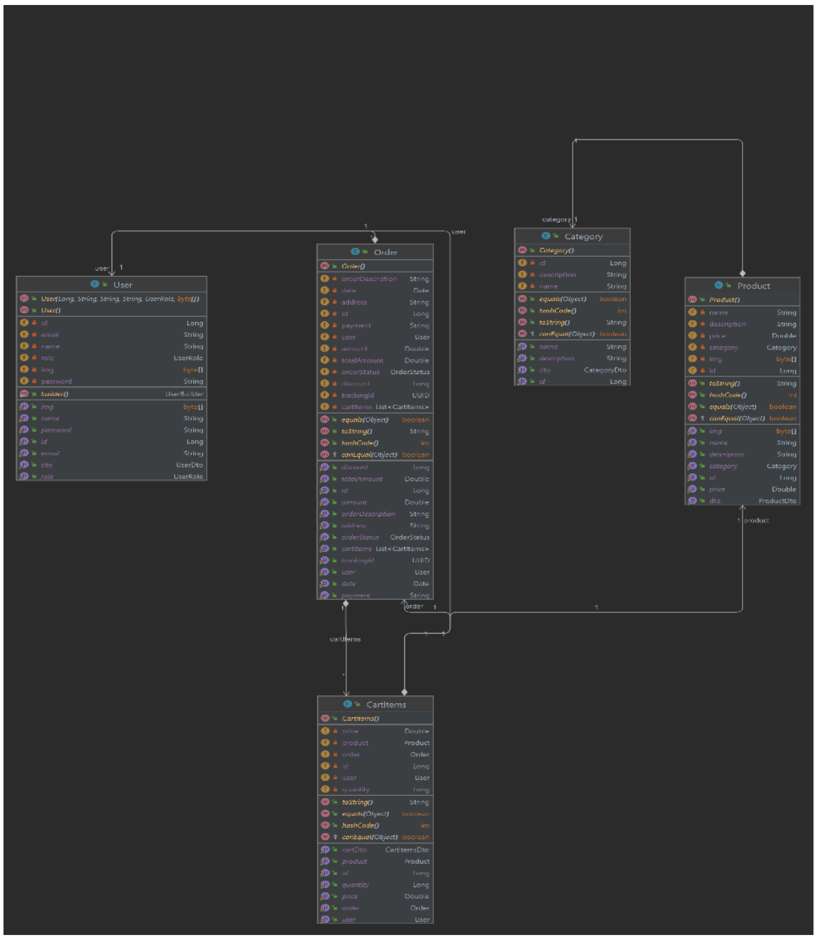
***Create new Product:** This function can be done just by an admin, and it appears just on the admin screen. You access it by clicking on the product page while logged in with an admin account. You can there select the products image, category, price, name, description.. When you click on Add Product button the product will be created and saved in the database, if everything goes well. If not an error message will be shown.*



Class Diagram



Data Model



System Testing

- ***JUnit Testing** : Implemenemented methods to show the services correctness. I created positive and negative testst also to see their behavior and correctness.*
- ***Manual Testing from Interface** : After creating the frontend I checked every methods functionality through the frontend . If I didn't use a method in the frontend I tested if it's working with the debugger.*

Future Improvements

- ***Coupons for order discounts.***
- ***Place order and delivery status for the order.***
- ***Analythics for the orders.***
- ***Review for orders and comments for them.***
- ***Finish the cart with adding more products and removing them if we don't want them***

Conclusion

Bibliography