Supermarket Manager

Analysis and Design Document

Student: Marin Andreea

**Group: 30431**

Revision History

|  |  |  |  |
| --- | --- | --- | --- |
| **Date** | **Version** | **Description** | **Author** |
| 10/04/23 | 1.0 | First version | Marin Andreea |
| 02/05/23 | 2.0 | Second version | Marin Andreea |
|  |  |  |  |
|  |  |  |  |

Table of Contents

I. Project Specification 4

II. Elaboration – Iteration 1.1 4

1. Domain Model 4

2. Architectural Design 4

2.1 Conceptual Architecture 4

2.2 Package Design 4

2.3 Component and Deployment Diagrams 4

III. Elaboration – Iteration 1.2 4

1. Design Model 4

1.1 Dynamic Behavior 4

1.2 Class Design 4

2. Data Model 4

3. Unit Testing 4

IV. Elaboration – Iteration 2 4

1. Architectural Design Refinement 4

2. Design Model Refinement 4

V. Construction and Transition 5

1. System Testing 5

2. Future improvements 5

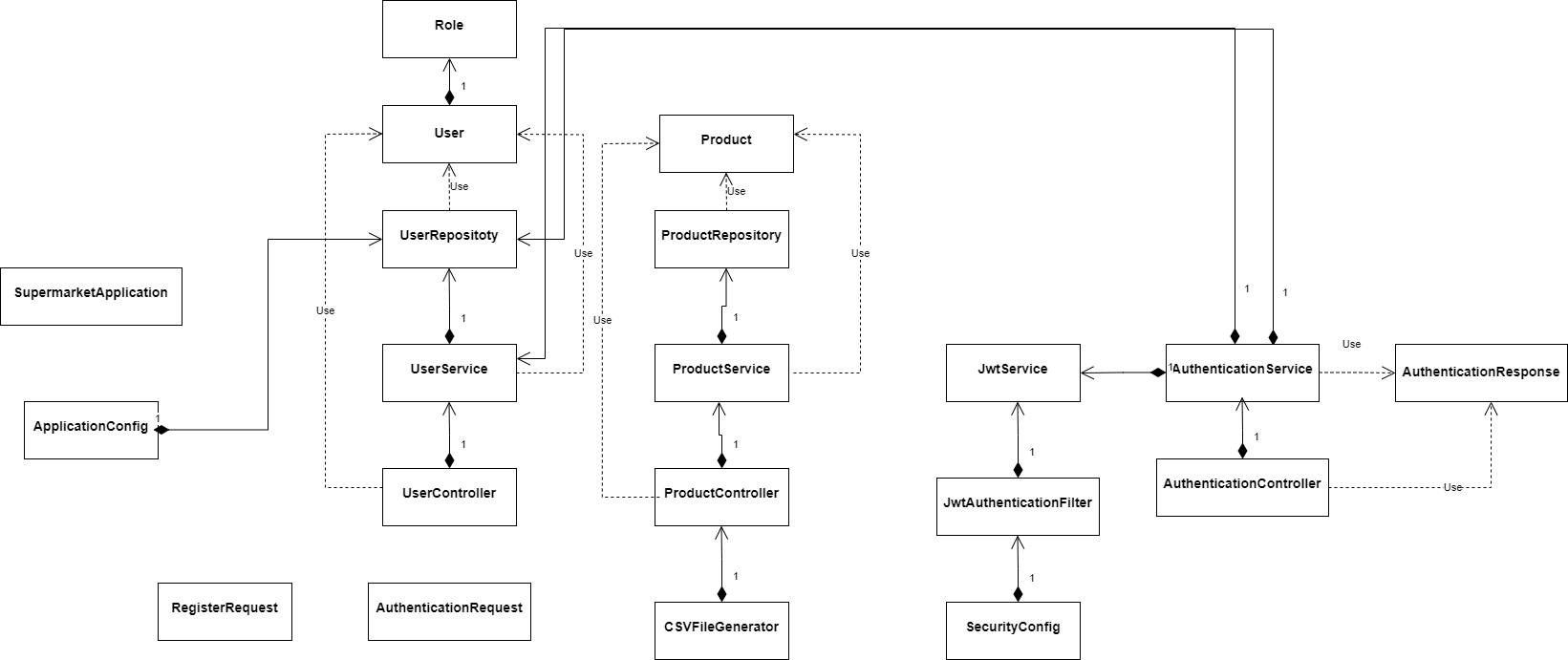
VI. Bibliography 5

# Project Specification

The goal of this project is to design and implement a client-server application for the employees of a supermarket. The application will have two types of users (a regular user represented by the cashier, and an administrator user). Both of them have to provide a username and a password in order to use the application.

# Elaboration – Iteration 1.1

# Domain Model



The conceptual class diagram of the project presents the classes contained by my project and illustrates the relationships between them. “Role” is an enumeration class that contains two strings: ADMIN and CASHIER and it is used to make the difference between the users of the application. UserRepository and ProductRepository are interfaces that extend **CrudRepository** and are used for implementing CRUD operations.

The authentication is done using jwt tokens. When the user logs in the application a token is generated and then is used for authorization.

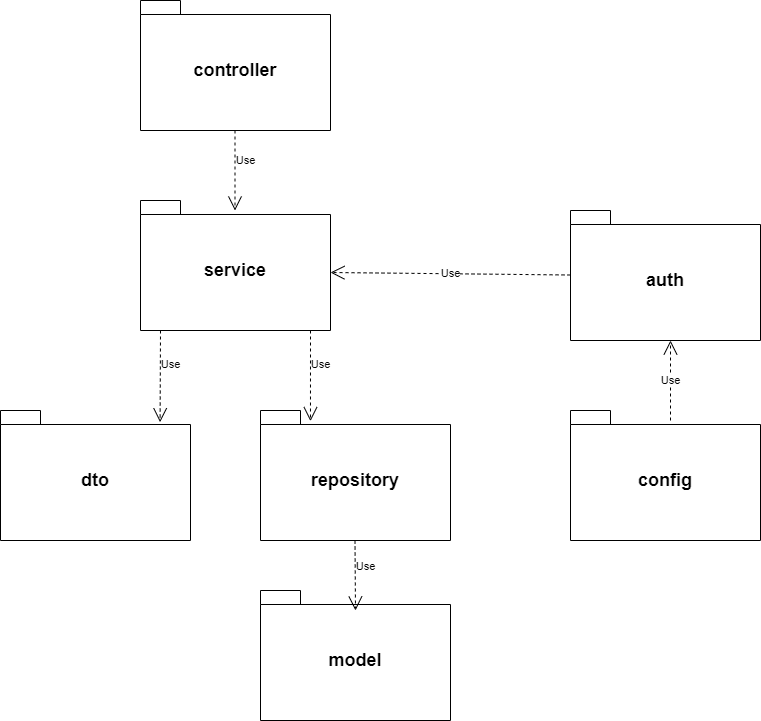
# Architectural Design

## Conceptual Architecture

The architecture used for this project is a layered one, as it can be deduced from the class diagram. Layered architecture is said to be the most common and widely used architectural framework in software development. It is also known as an n-tier architecture and describes an architectural pattern composed of several separate horizontal layers (in my case model, controller, repository, service) that function together as a single unit of software. I chose this architectural style because layers are only connected to the layers directly below them which means that layers can be modified and the change won’t affect other layers.

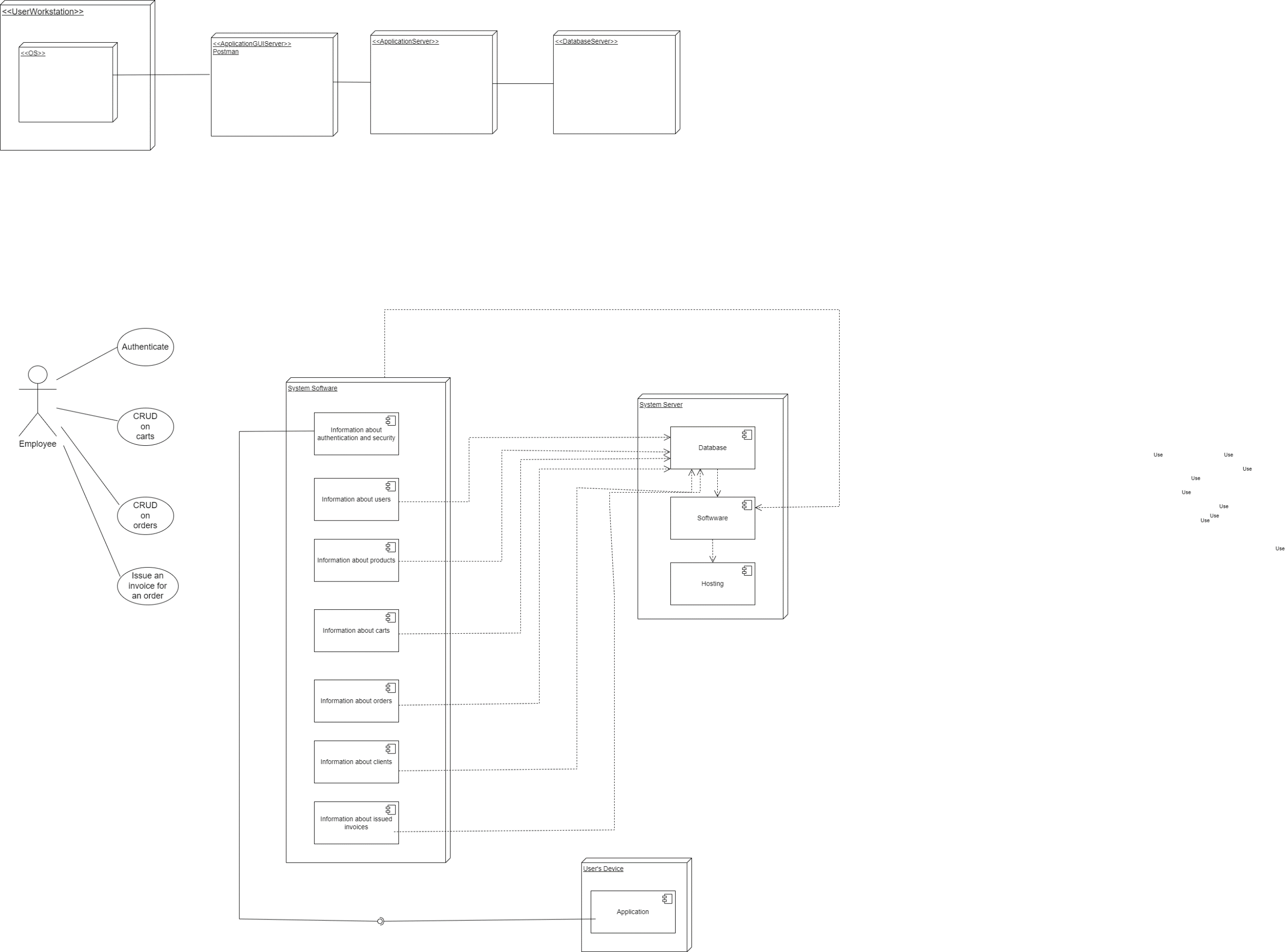
## Package Design

The package diagram also highlights the use of a layered architecture.

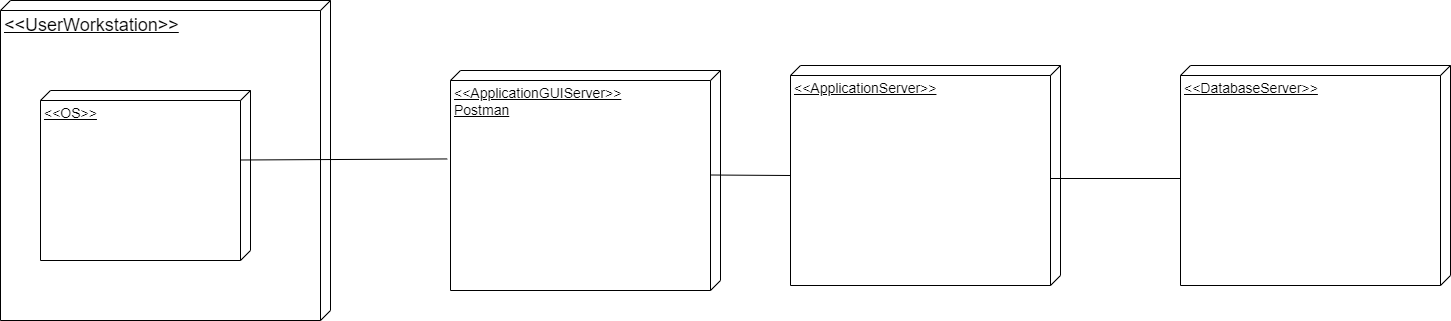


## Component and Deployment Diagrams

Component Diagram



Deployment Diagram

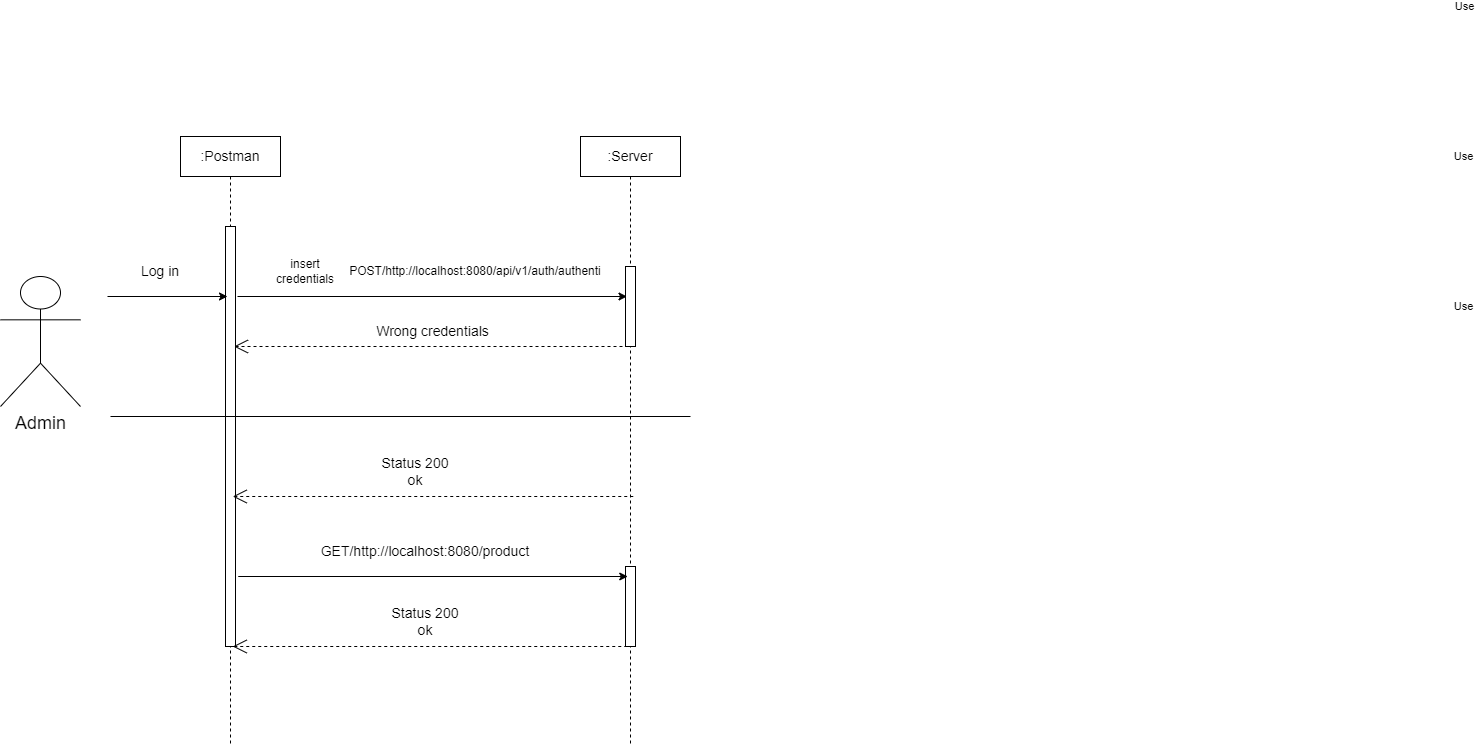


# Elaboration – Iteration 1.2

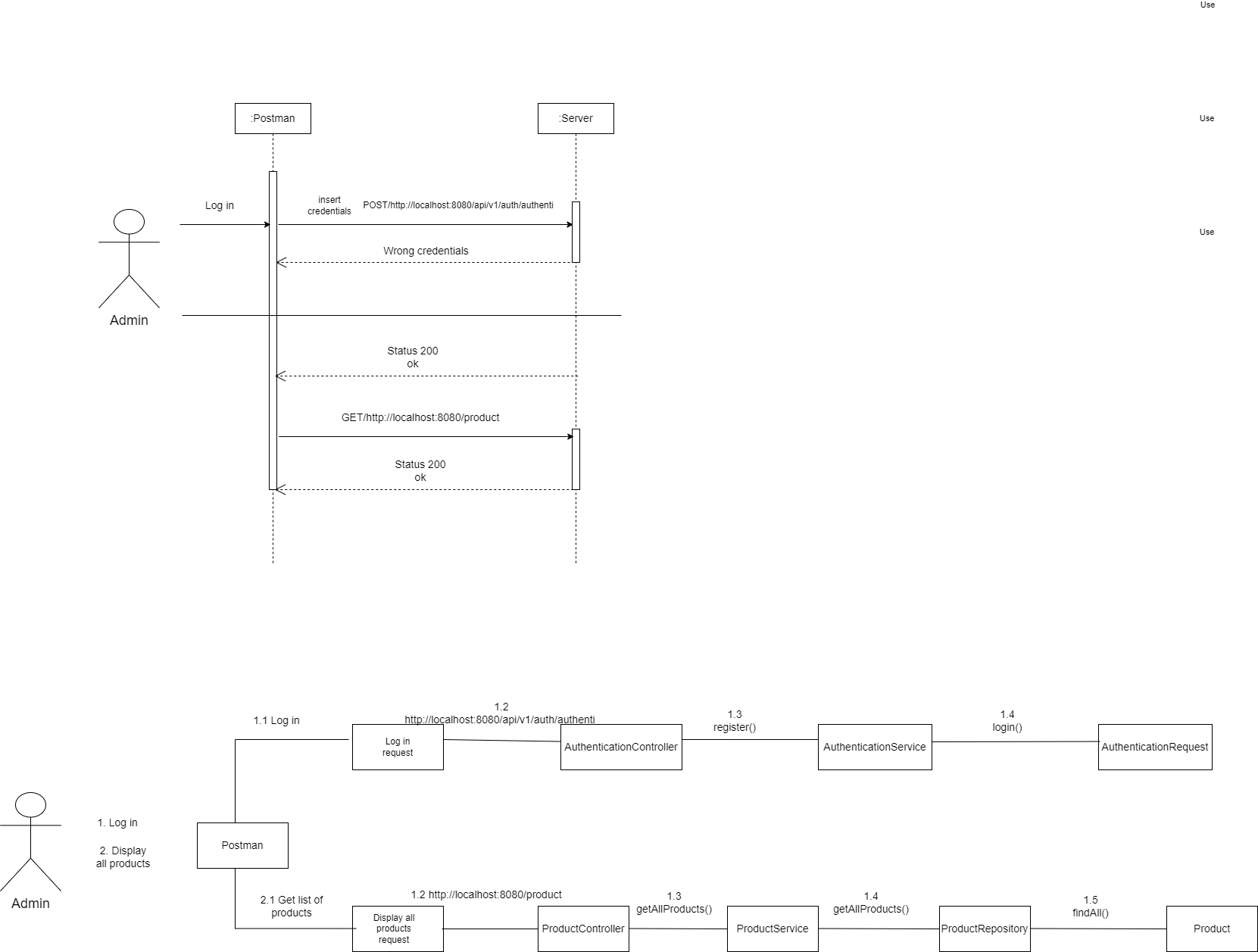
# Design Model

## Dynamic Behavior

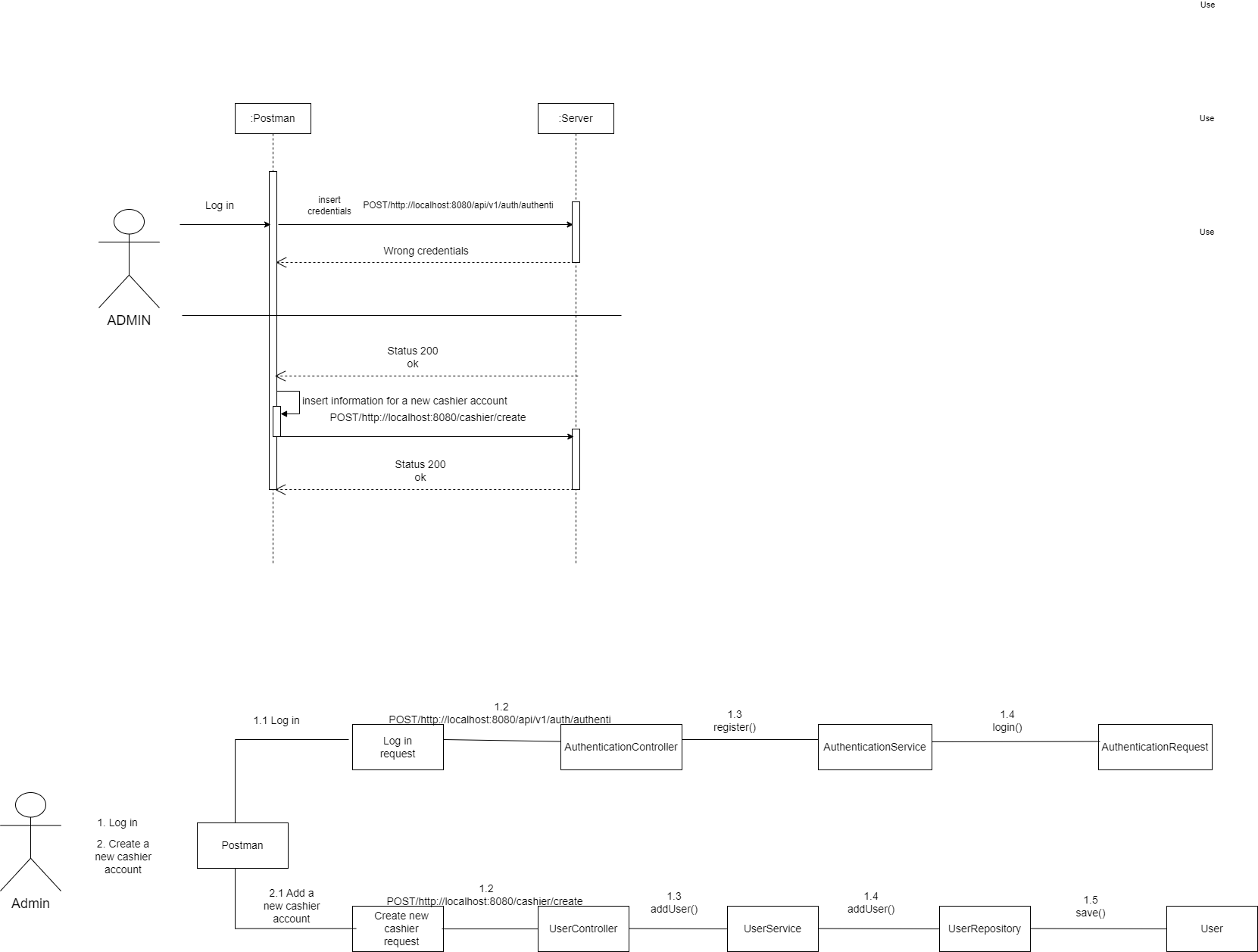
* View list of products
* Sequence diagram



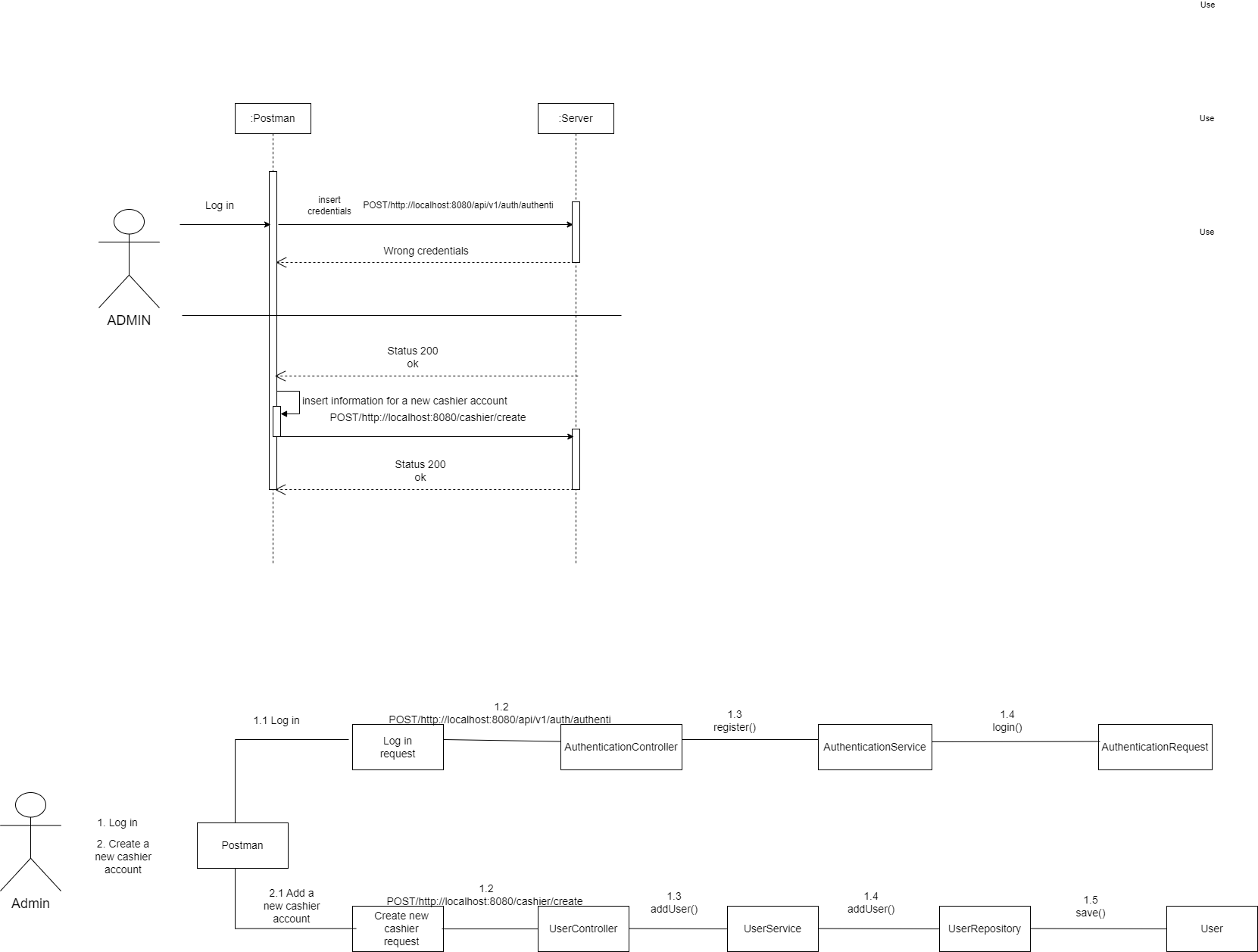
* Communication diagram



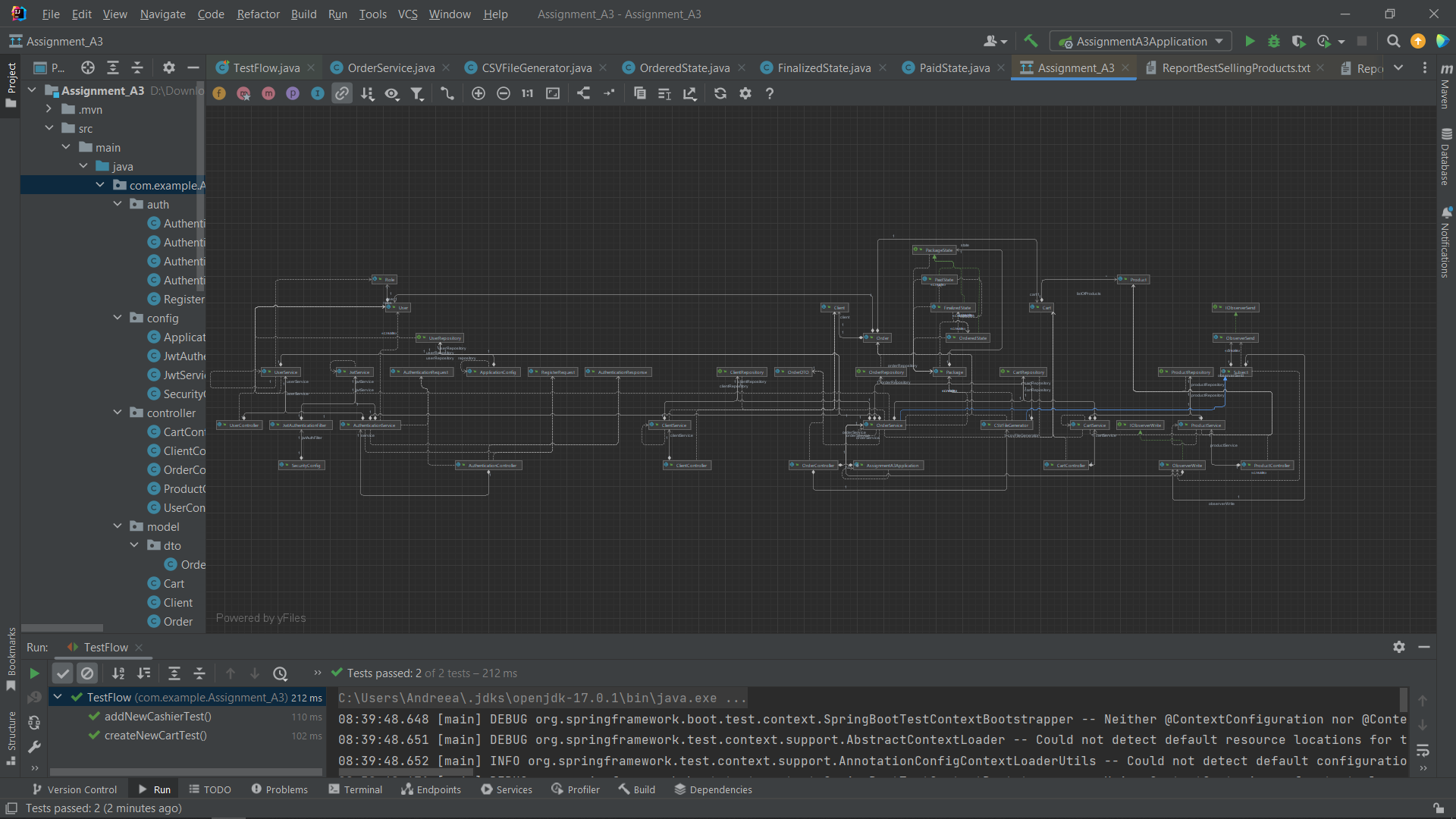
* Create a new cashier account
* Sequence diagram



* Communication diagram

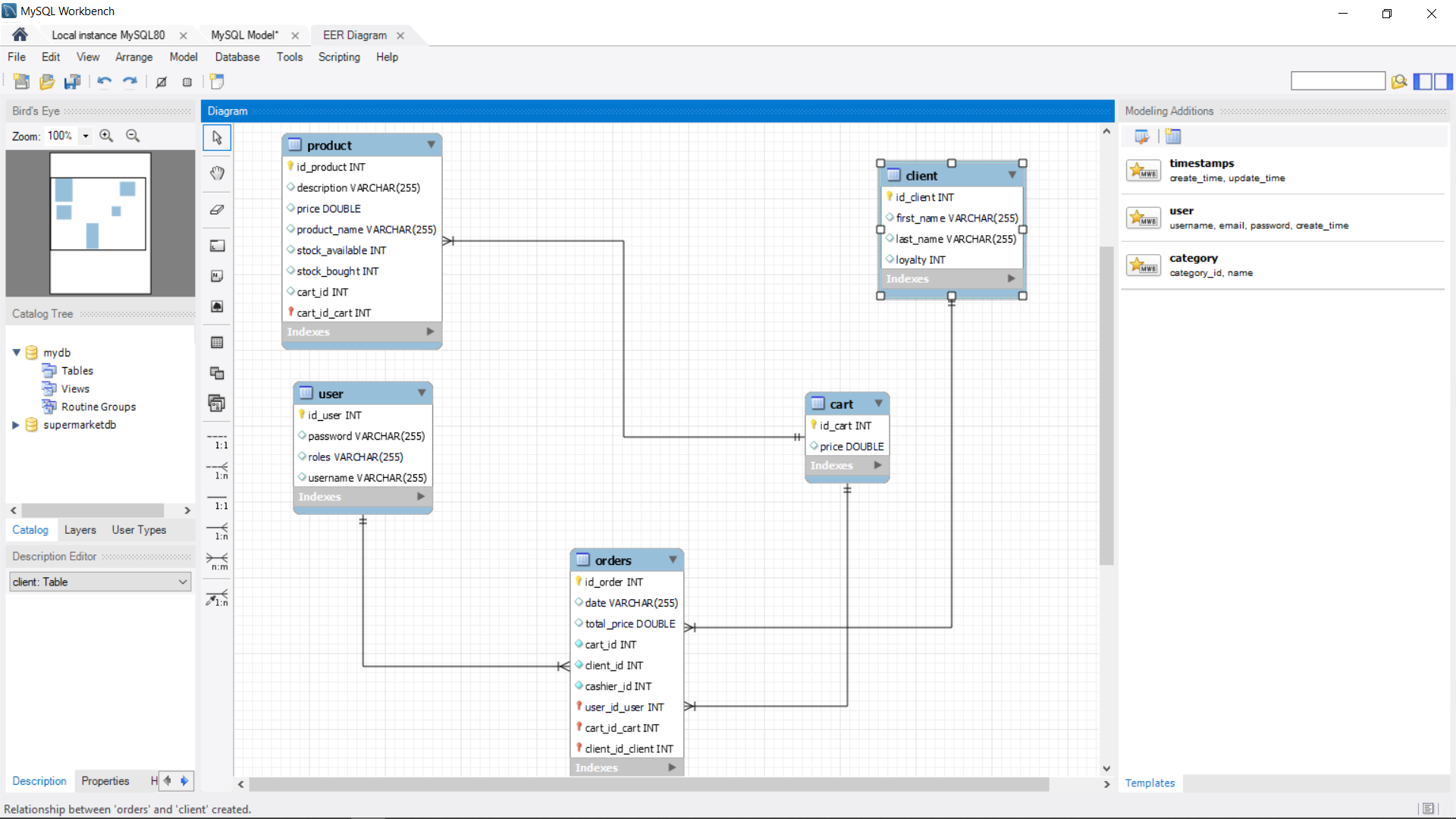


## Class Design



# Data Model

This is the database diagram of the system:



It can be seen that the system contains five data models:

* **User data model**: represents the table for the user entity. It consists of an id, username and password (which will be used for authentication) and a role(ADMIN or CASHIER)
* **Product data model**: represents the table for the product entity. It consists of an id, name of the product, description, price, available stock and stock bought so far(the available and bought stock will be updated each time a client buys the product)
* **Cart data model**: represents the table for the cart entity. It consists of an id, price and a list of products that will be bought. When a new product is added to the cart the available stock and bought stock of that product are updated and the price of the cart is updated as well.
* **Client data model**: represents the table for the client entity. It consists of an id, first name , last name and loyalty(which represents the number of orders done by the client)
* **Order data model**: represents the table for the order entity. It consists of an id, date, total price, cart id, cashier id and client id. A new order is created with an existing cart and the total price of the order will be the price of the cart. Here the client id is used for identifying the client associated with the order and increase his loyalty. When the loyalty variable reaches the value 10, the admin can activate a promotion of 50% to the total price of the order.

# Unit Testing

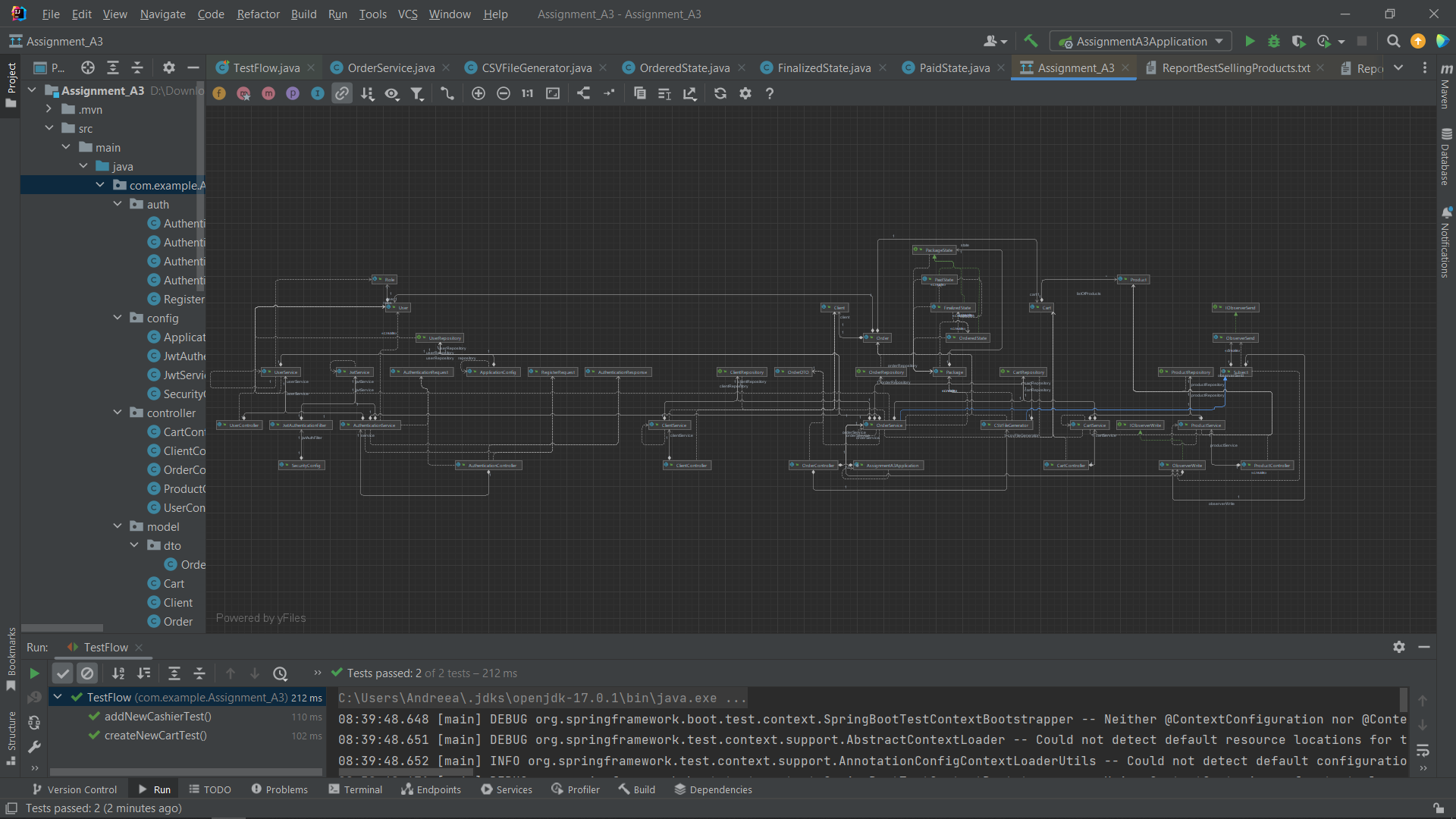
For testing the system I used Mock. The application provides two tests:

* Test for creating a new cashier. A new user is created. It is saved in the database using the repository for the User and at the end it verifies if the string returned after creating the user is equal to “A new user with id 4 was created!”
* Test for creating a new cart and adding products to the cart. Two new products are created, a new cart is created containing the two products and at the end the test verifies if the string returned after creating the cart is equal to “A new cart with id 3 was created successfully!”

# Elaboration – Iteration 2

# Architectural Design Refinement

# Design Model Refinement



# Construction and Transition

# System Testing

For testing the system I used Postman. The testing process involves the following steps:

* Authenticate as admin
* Display all cashiers
* Create a new cashier
* Display all products
* Create new products
* Authenticate as cashier
* Create a new cart
* Add products to the cart
* Create a new order
* Authenticate as admin
* Find all clients that benefit from promotion
* Authenticate as cashier
* Activate the promotion for clients that benefit from it
* Finalize the order
* Issue an invoice
* Authenticate as admin
* Issue reports on best selling products, products out of stock and cashiers’ activity

# Future improvements

As future improvements the system could:

* Activate promotions on best selling products (example: 2 products at price of one)
* Allow clients to register on their own and use a username and password to authenticate
* Activate promotions for clients that bring new clients to the supermarket(clients send a description and a unique token via email to other clients that have not registered yet and if they use the sent token to register than the client that sent them the token will benefit a promotion to his next order)

# Bibliography

* <https://www.youtube.com/playlist?list=PLqq-6Pq4lTTbx8p2oCgcAQGQyqN8XeA1x>
* <https://www.youtube.com/watch?v=KxqlJblhzfI>
* <https://www.youtube.com/watch?v=9SGDpanrc8U>
* <https://www.baeldung.com/spring-boot-data-sql-and-schema-sql>
* <https://www.baeldung.com/java-state-design-pattern>