<Online Grocery Shopping >-

Analysis and Design Document

Student: Antonescu Maria-Cristina

**Group: 30431**

Revision History

|  |  |  |  |
| --- | --- | --- | --- |
| **Date** | **Version** | **Description** | **Author** |
| <dd/mmm/yy> | <x.x> | <details> | <name> |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

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 scope of the project is to design and develop a web application that will enable customers to purchase groceries online and have them delivered to their doorstep. The admin will have access to the backend of the application, where they can perform CRUD (Create, Read, Update, Delete) operations on the products and manage orders.*

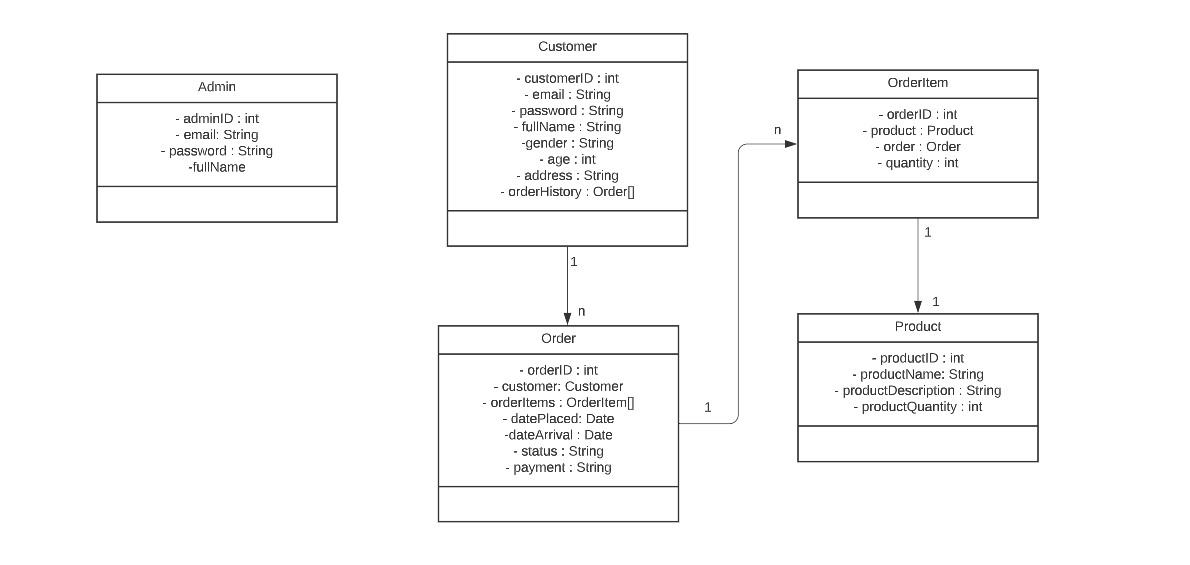
# Elaboration – Iteration 1.1

# Domain Model

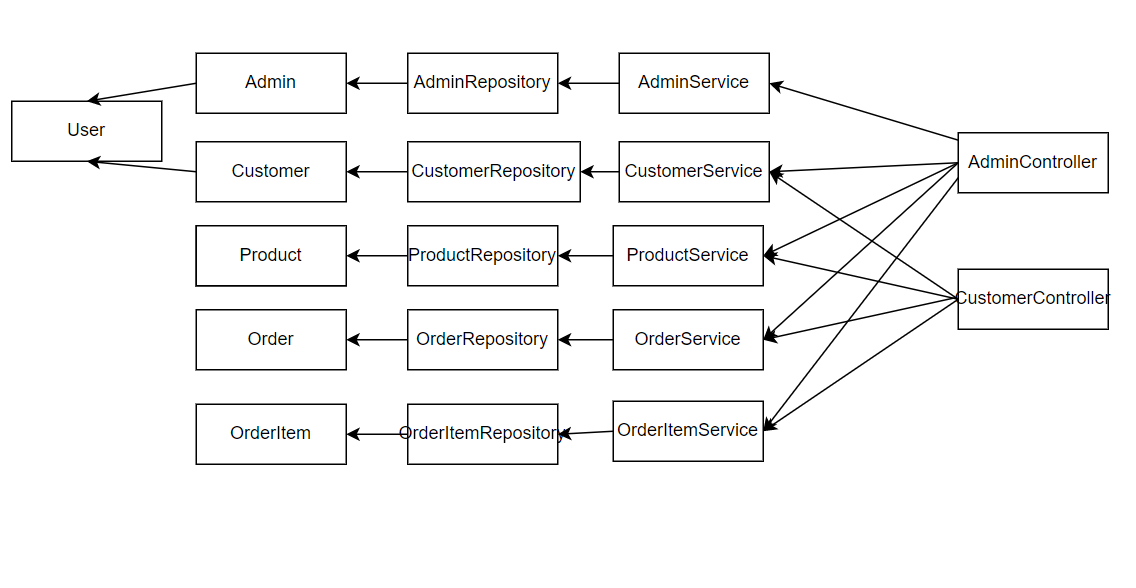
*The main entities in the system will be the following:*

* *Admin – a type of user, which can log-in the app, perform CRUD operations Customers, Products and Order;*
* *Customer – can register, log-in the app, can place an Order, view products;*
* *Product – described by the unique code, name, description and current quantity;*
* *Order – described by customerID, other details such as the date it has been placed, the date it should be delivered, address, the order items and the payment details*
* *OrderItems – entity which manages the relationship between Product and Order, it has productid, quantity, orderID.*

*The domain model:*

**

*The conceptual class diagram:*

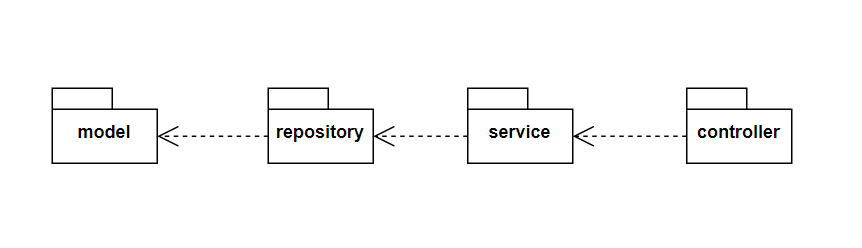
**

# Architectural Design

## Conceptual Architecture

1. *Presentation layer: This layer is responsible for handling user requests and responses. It consists ina controller that processes user input and returns the appropriate response.*
2. *Business layer: This layer is responsible for handling business logic and performing data manipulation. It acts as a bridge between the presentation layer and the data access layer.*
3. *Persistence layer: This layer is responsible for accessing data from a database or other data storage system. It consists of data access objects (DAOs) that perform CRUD (Create, Read, Update, Delete) operations on the data.*

## Package Design

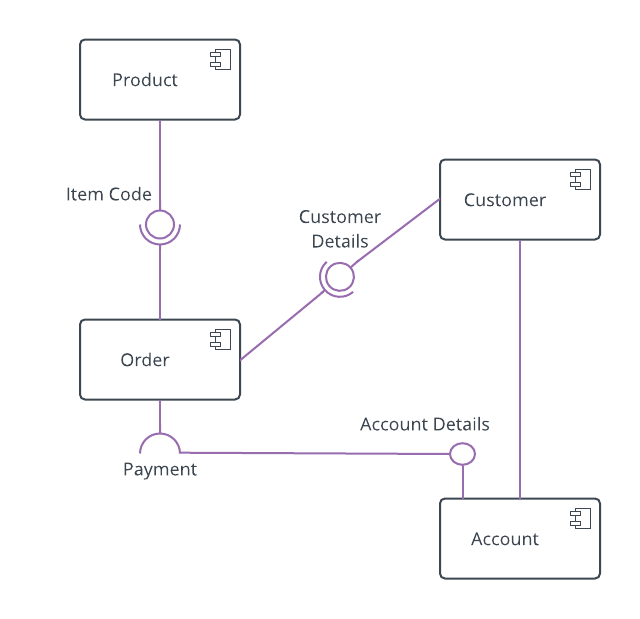
**

*Package description:*

* *Model – the package which contains the entity definitions*
* *Repository – the package which contains the JpaRepositories*
* *Service – the package which contains the Service classes*
* *Controller – the controller for the 2 types of users: Admin and Customer*

## Component and Deployment Diagram

*Component diagram*

**

*Deployment diagram*

# 

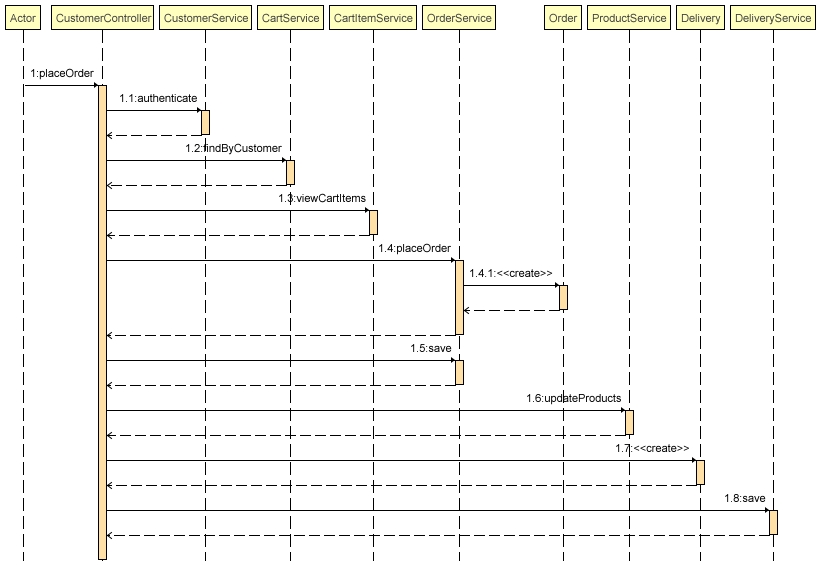
# Elaboration – Iteration 1.2

# Design Model

## Dynamic Behavior

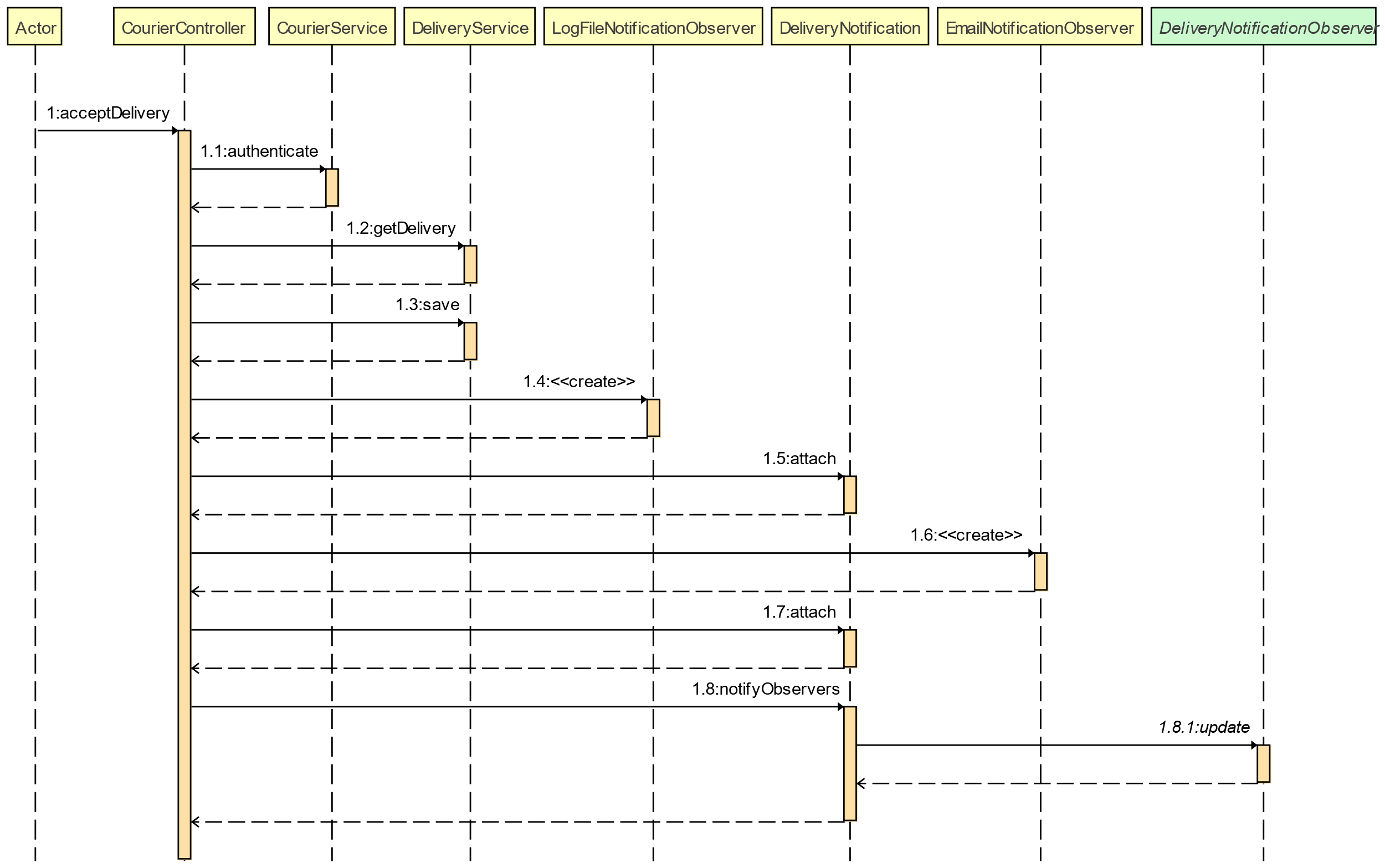
*Sequence Diagrams*

*I will present a sequence diagram for the place order use case, because it is the most relevant scenario so far for this project.*

**

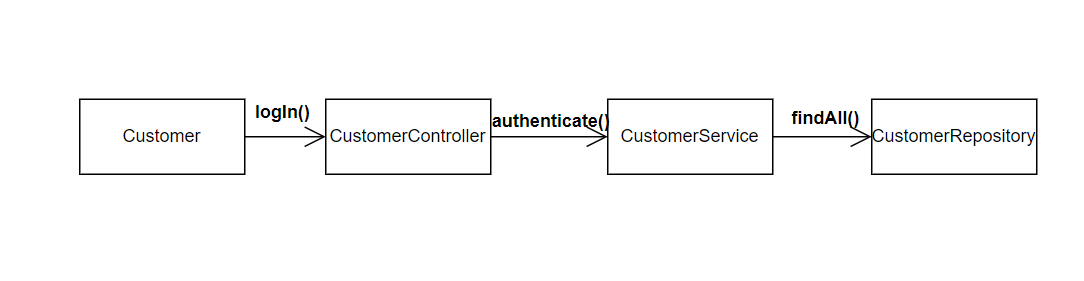
*When a customer tries to place an order, the cart which has been created by that customer is found, then the CartItems into that specific cart are found. Then, each CartItem is validated and an order and a delivery are created.*

*Another relevant scenario is the accept delivery use case.*

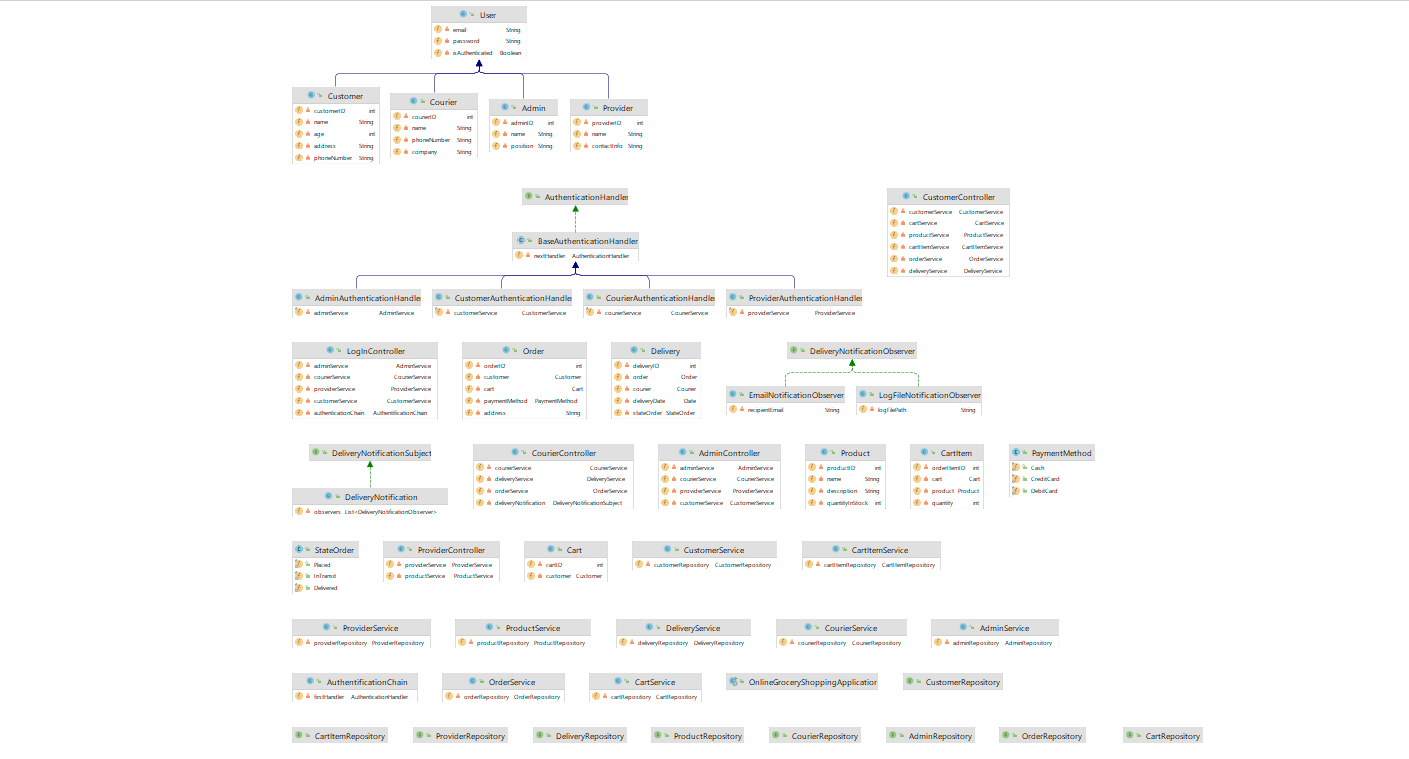
**

*When a courier accepts a delivery, the state of the delivery is changed and the observers are notified of this change and send an email/write into file.*

*Communication Diagram*

**

## Class Design

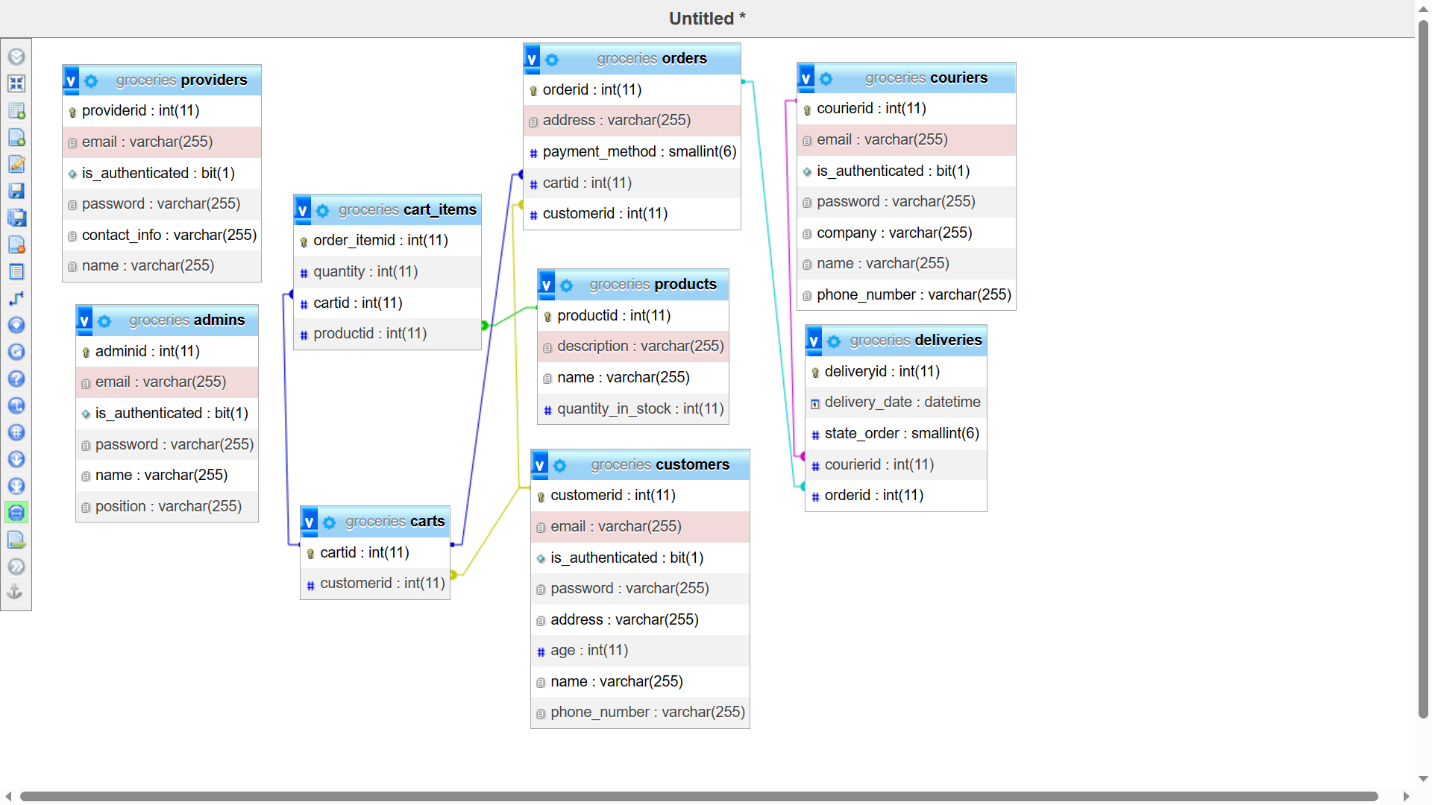
**

*So far, I have used 2 behavioral design patterns in this project:*

***1.Chain of responsibility Pattern*** *–I used this pattern to have a chain of handlers, where each handler is responsible for handling the login request for a specific type of user. There are handlers for admin, customer, provider, and courier. This is to make the login part of the project much easier to handle from the code perspective.*

***2.Observer Pattern*** *- I used the Observer pattern to send an email notification to the customer after the delivery has been picked up by a courier and to write into a file. The Observer pattern allows an object (the subject) to notify a list of other objects (the observers) of any changes to its state.*

# Data Model

**

*There are 5 entities in the system:*

*•* ***Admin*** *– is a type of User, has email, password and has permission to perform CRUD operations on all other types of users.*

*•* ***Customer*** *– is a type of User, has email, password, full name, address, phone number and age.*

*•* ***Provider*** *– is a type of User, has email, password, name and contact info.*

*•* ***Courier*** *– is a type of User, has email, password, name, company and phone number.*

*•* ***Product*** *– is an entity which describes a product, it has a name, description, price and total quantity in stock.*

*•* ***Cart*** *– is an entity which represents a customer's cart, it has a reference to the customer.*

***• CartItem*** *– is an entity which represents a product added to a cart, it has a reference to the cart, a reference to the product and a quantity.*

***• Order*** *– is an entity which represents a customer's order, it has a reference to the customer, a reference to the cart and a payment method and an address.*

***• Delivery*** *– is an entity which represents a delivery, it has a reference to the courier, a reference to the order and a status.*

*The relationships between the entities:*

* *CartItem – Cart --- ManyToOne*
* *CartItem – Product – ManyToOne*
* *Delivery – Courier – OneToOne*
* *Delivery – Order – OneToOne*
* *Cart – Customer – OneToOne*
* *Cart – Order – OneToOne*

# Unit Testing

*I have used JUnit to write 2 unit tests, which validate the log-in and the placeOrder methods from the CustomerController.*

**

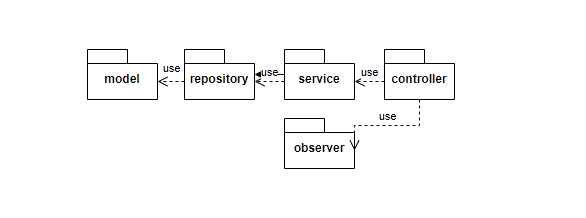
# Elaboration – Iteration 2

# Architectural Design Refinement

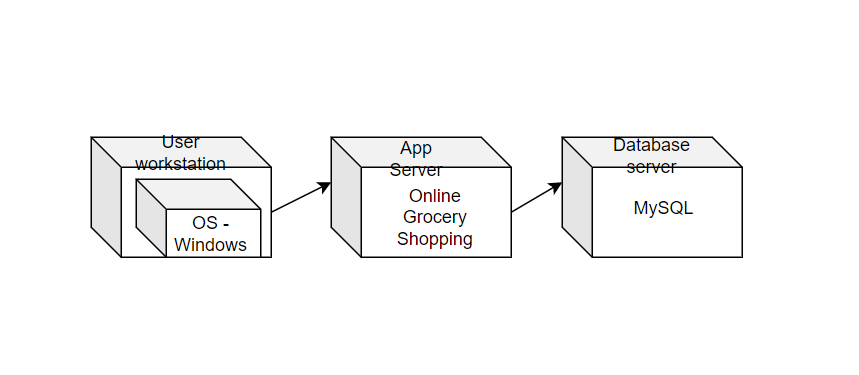
*I have the following packages:*

* ***Model –*** *includes all the entities in the system, I have added 2 additional entities to the original model: Delivery (which is meant for the Courier) and Cart (the customer can add products to the cart, update, delete items and then place the order);*
* ***Repository –*** *includes interfaces for database design;*
* ***Service*** *– includes methods which perform CRUD operations on entities from the database;*
* ***Controller –*** *controllers for the 4 types of users, which define the requests;*
* ***Observer –*** *includes all the classes to implement the observer design pattern.*

***Package diagram:***

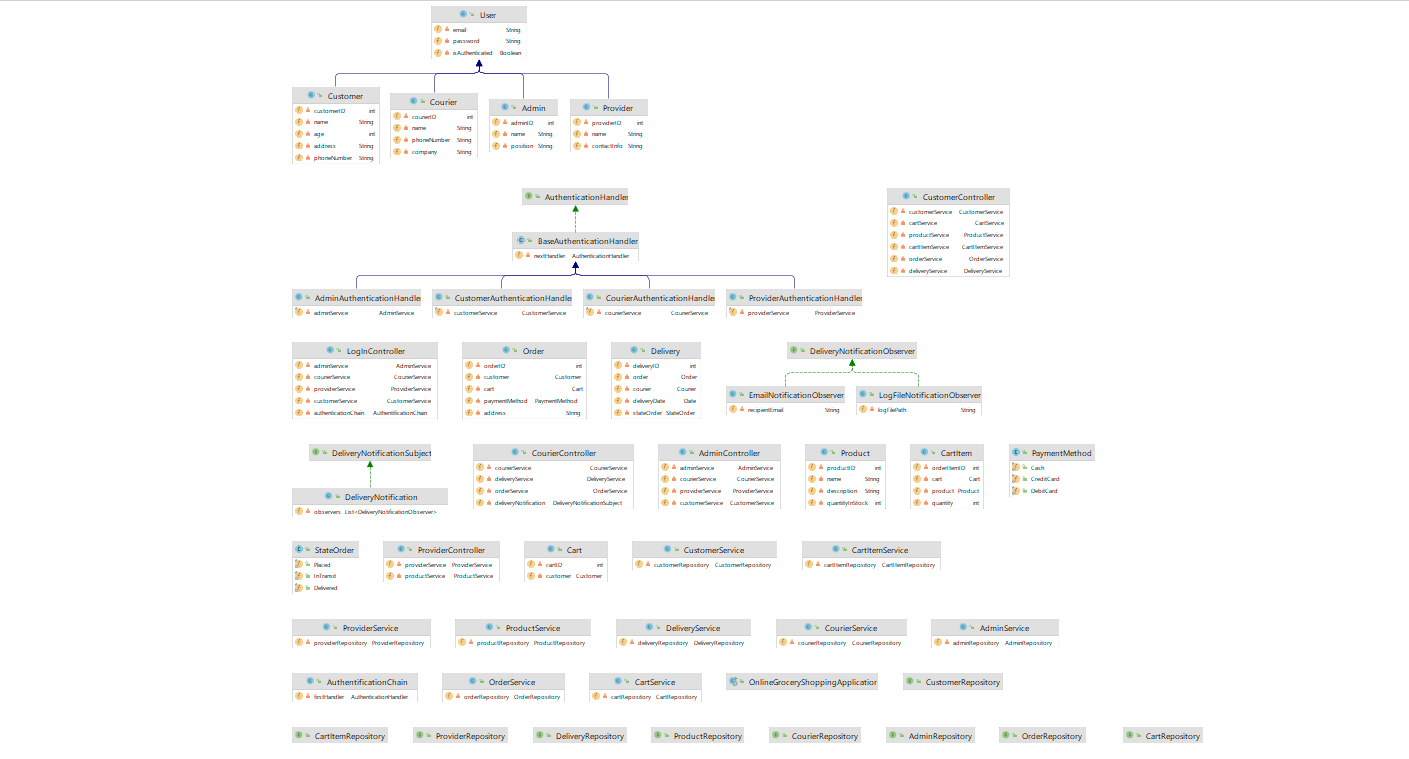
******

***Deployment diagram:***

******

# Design Model Refinement

*Final class diagram:*

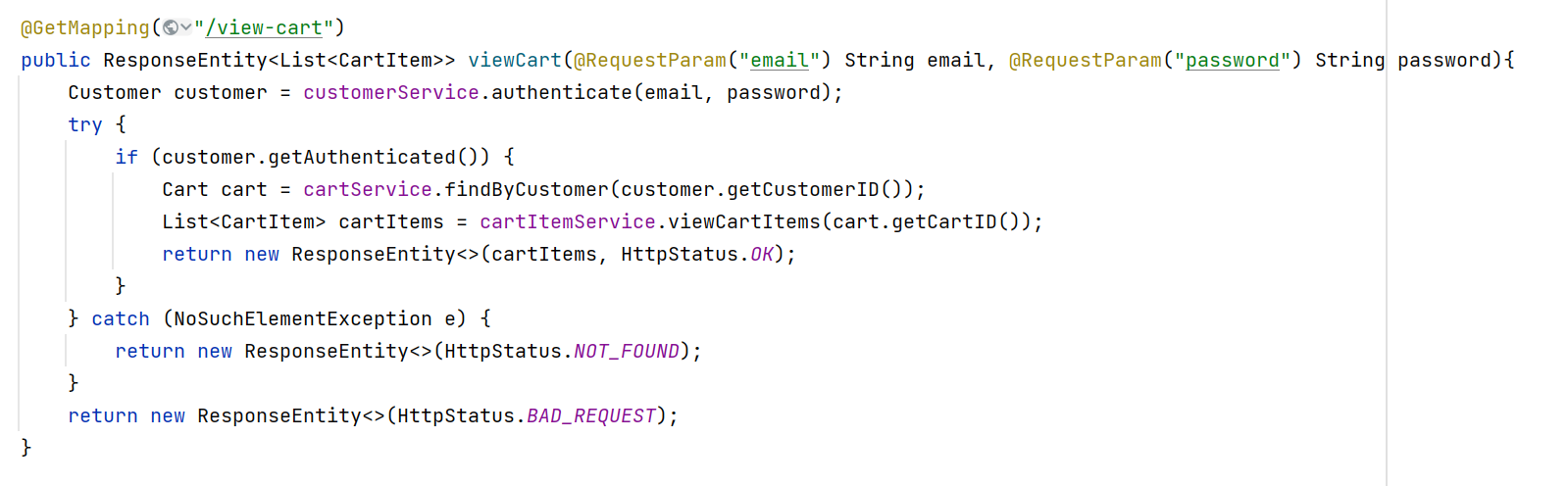
**

*I have detailed all the classes I have used at the previous section, when I described the packages.*

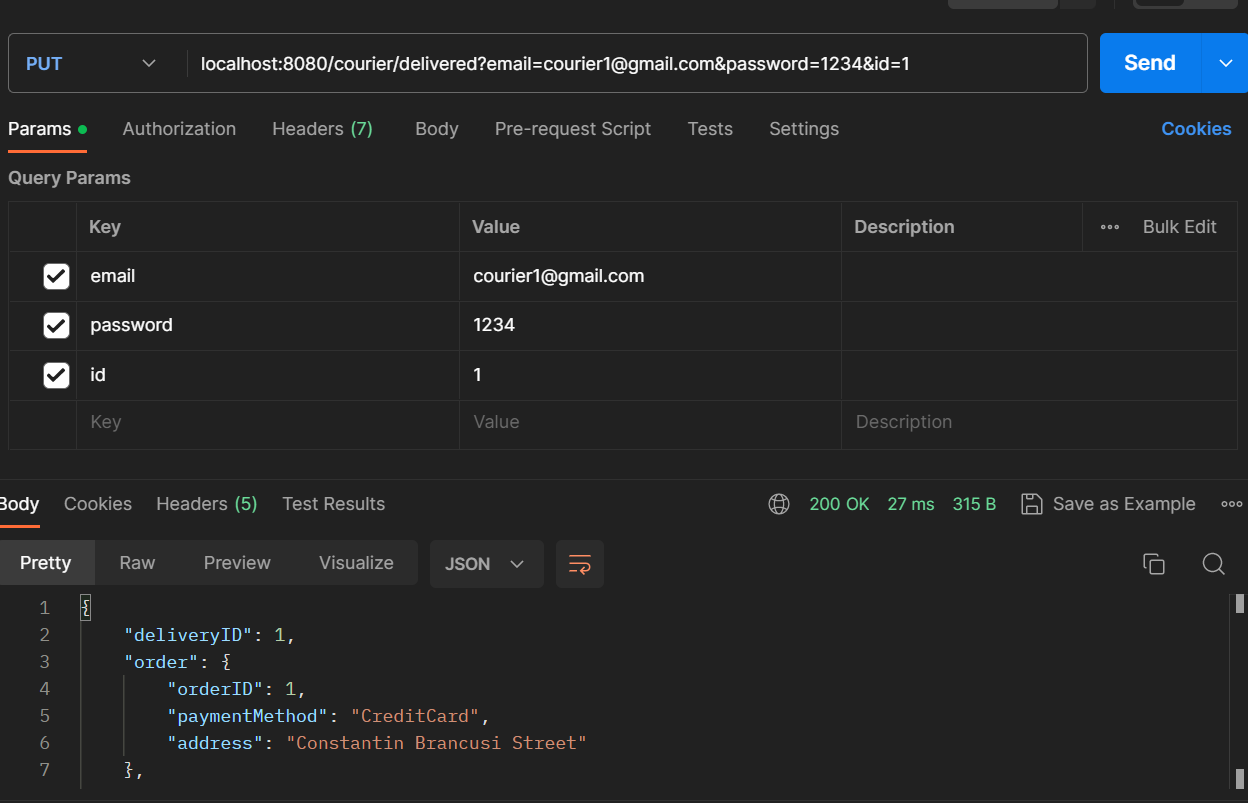
# Construction and Transition

# System Testing

*For this online grocery shopping system, I tested the requests in Postman. I returned a ResponseEntity to make sure that my info is update correctly. I also return HTTP codes to make testing much easier. Here is how a class from the controller looks:*

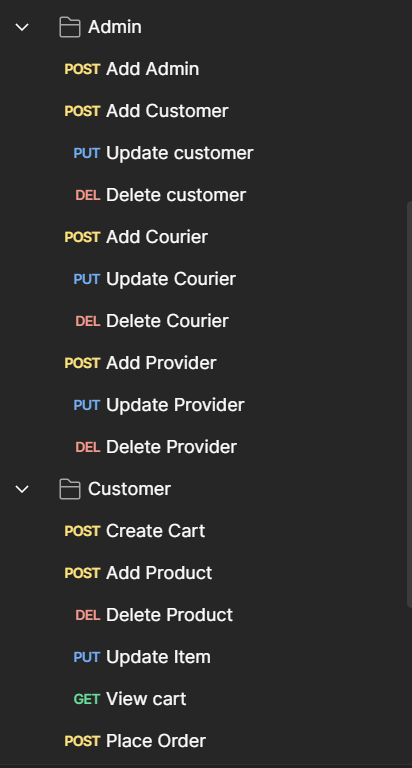
**

*This is an example of how testing a request using Postman looks like:*

**

*As it can be seen, the request was successful and it returned a Delivery Response Entity.*

*I have organized my requests according to the type of user they belong to:*

**

# Future improvements

*In the future, I want to add also a front-end for this application, in which users can actually visualize the products. Some features which could be added: view order history and track courier route for the customer, implement a rating system for couriers.*

# Bibliography