Online Health Shopping Portal
Analysis and Design Document
Student: Ana-Maria Nanes
Group:30432

Revision History

Date	Version	Description	Author
05/04/2018	1.0	Domain Model + Architectural Design	Ana-Maria Nanes
25/04/2018	2.0	Design Model + Data Model	Ana-Maria Nanes
21/05/2018	3.0	Final Documentation Version	Ana-Maria Nanes

Table of Contents

I. Project Specification

The application in an online paltform that helps the users in curing their disease by giving the list of fruits and herbs that the user should consume in order to get rid of its disease.

The main purpose of the application is to help the user to find the accurate products according to the introduced disease and to reduce as much as possible the search time.

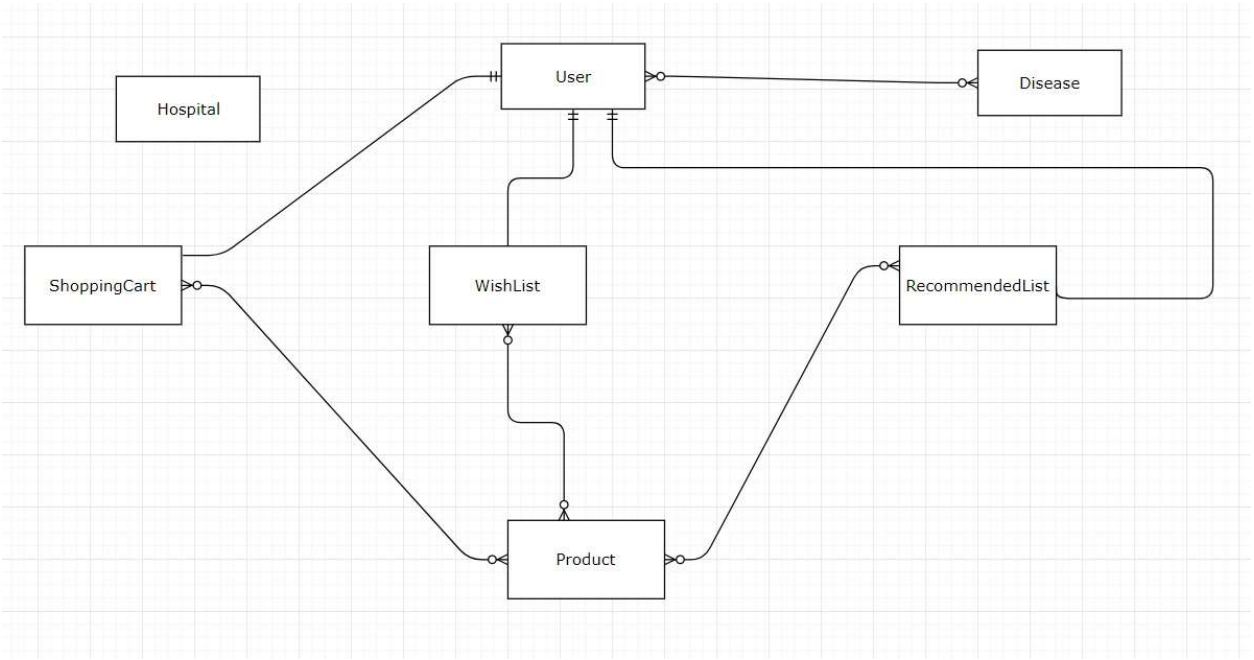
The system also allows the user to see the description of each herb or fruit in the system in order to read about the benefits of consuming it.

The user also has the possibility to add products to the chart and buy them. The user can choose the products or can select the option that recommends exactly the necessary herbs, fruits and vegetables.

II. Elaboration – Iteration 1.1

1. Domain Model

The domain model is the conceptual model of the domain that incorporates both behavior and data.



2. Architectural Design

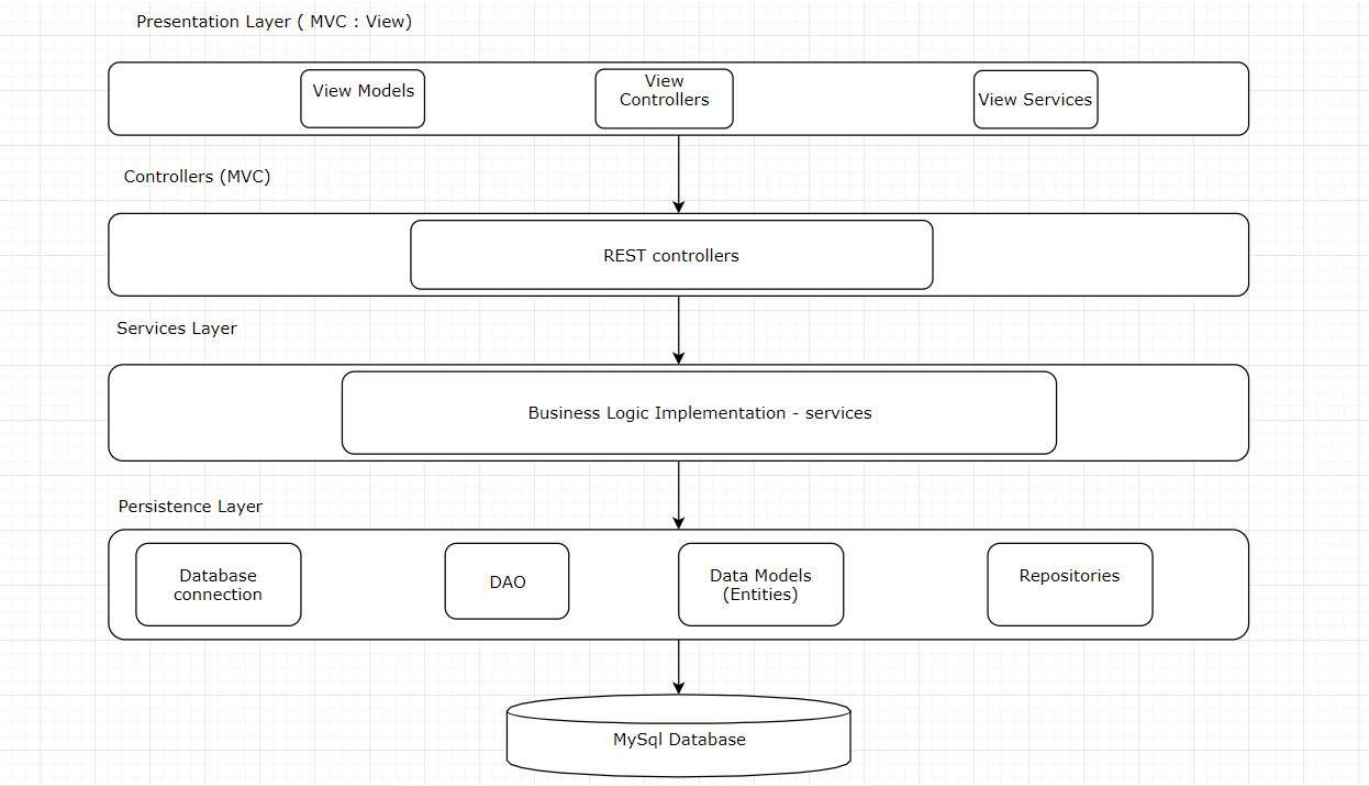
2.1 Conceptual Architecture

The architectural patters used are the following:

- Model-View-Controller Architectural Pattern
- Layers Architectural Pattern
- Client-Server Architecture

The used design patterns will be:

- Observer Design Pattern
- Builder
- Dependency Injection
- Inversion of Control



1. Layers Architectural Pattern

This pattern can be used in order to structure the application such that it can be decomposed into groups of subtasks. Each layer implements subtasks at a particular level of abstraction and each layer provides services to the next higher level. The following 4 layers will be used in order to offer the application the accurate structure:

- Presentation Layer
- Controller Layer
- Business Logic Layer
- Data Access Layer

These layers are modeled using different packages inside the application.

2. The Model-View-Controller Architectural Pattern

Model–view–controller (MVC) is an architectural pattern commonly used for developing user interfaces that divides an application into three interconnected parts. This is done to separate internal representations of information from the ways information is presented to and accepted from the user. The MVC design pattern decouples these major components allowing for efficient code reuse and parallel development.

- Model - Model represents an object or JAVA POJO carrying data. It can also have logic to update controller if its data changes.
- View - View represents the visualization of the data that model contains.
- Controller - Controller acts on both model and view. It controls the data flow into model object and updates the view whenever data changes. It keeps view and model separate.

3. The Client-Server Architecture

The client–server model is a distributed application structure that partitions tasks or workloads between the providers of a resource or service, called servers, and service requesters, called clients. A client does not share any of its resources, but requests a server's content or service function. Clients therefore initiate communication sessions with servers which await incoming requests.

The client–server model is a standard model for network applications. A server is a process that is continuously running and waiting to be contacted by a client process. A client process initiates contact with the server by connecting to it at a specified port.

The client-server type of the scenario for the application:

1. The server process is started on a computer system. It initializes itself and then waits for a client process to contact it with a service request.

    - ✓ In the case of my application, the server-side is implemented in Java using Spring MVC in orde to access the data, to implement the services and to provide the application routes.

2. The client process, usually started on another system, is connected to the server's system over a network. The client process sends a request across the network to the server requesting service of some form, e.g. reading or writing a file on the server's system.

    - ✓ In the case of my application the client-side is implemented by an AngularJS application through which the request will be send.

3. When the server finishes processing the request, it waits for the next client request to arrive.

    - ✓ The client-side application can send as many request as desired, they will be received by the server and responses will be sent back with the requested data.

The communication protocol:

- ✓ In the case of the Online Health Shopping Portal the HTTP communication protocol will be used in order for the client side of the application to send requests and in order for the server side of the application to provide the responses with the required resources.

The Hypertext Transfer Protocol (HTTP) is an application protocol for distributed, collaborative, and hypermedia information systems. Hypertext is structured text that uses logical links (hyperlinks) between nodes containing text. HTTP is the protocol to exchange or transfer hypertext.

- ✓ Both the server and the client side will implement a certain routing mechanism. The router is the component that translates each incoming HTTP request to an action call (a public method in a controller class). Controllers will be found in the Java application and also in the AngularJS application.

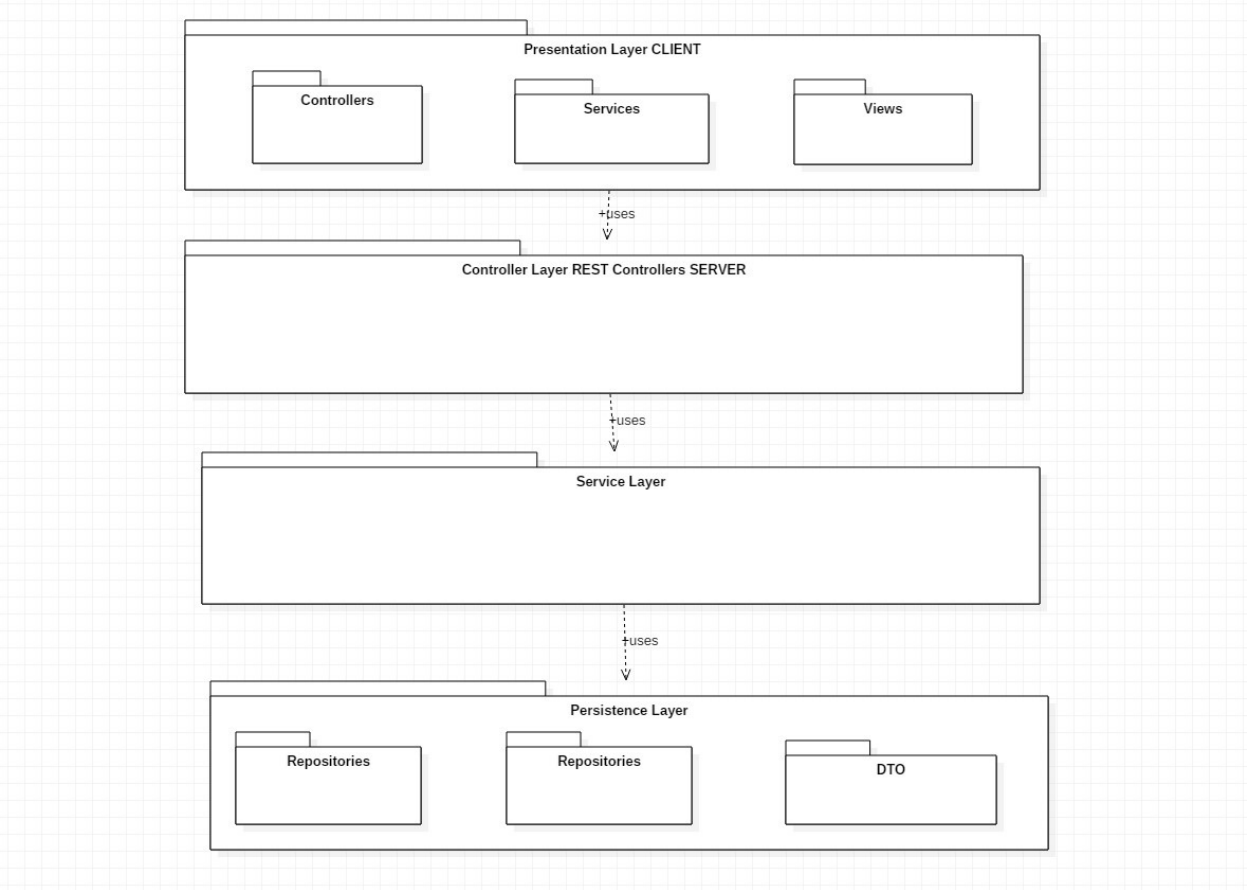
An HTTP request is seen as an event by the MVC framework. This event contains two major pieces of information:

- the request path (such as /clients/12, /photos/list), including the query string.
- the HTTP method.

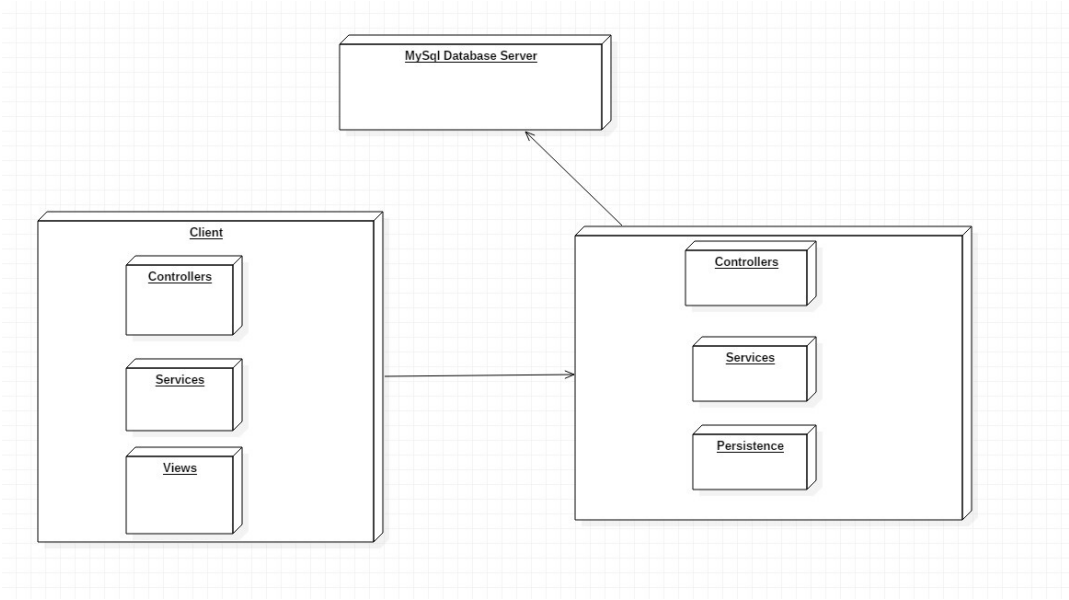
The HTTP method can be any of the valid methods supported by HTTP (GET, PATCH, POST, PUT, DELETE, HEAD, OPTIONS).

2.2 Package Design

The packages are used in odrer to model the layers.



2.3 Deployment Diagrams



III. Elaboration – Iteration 1.2

1. Design Model

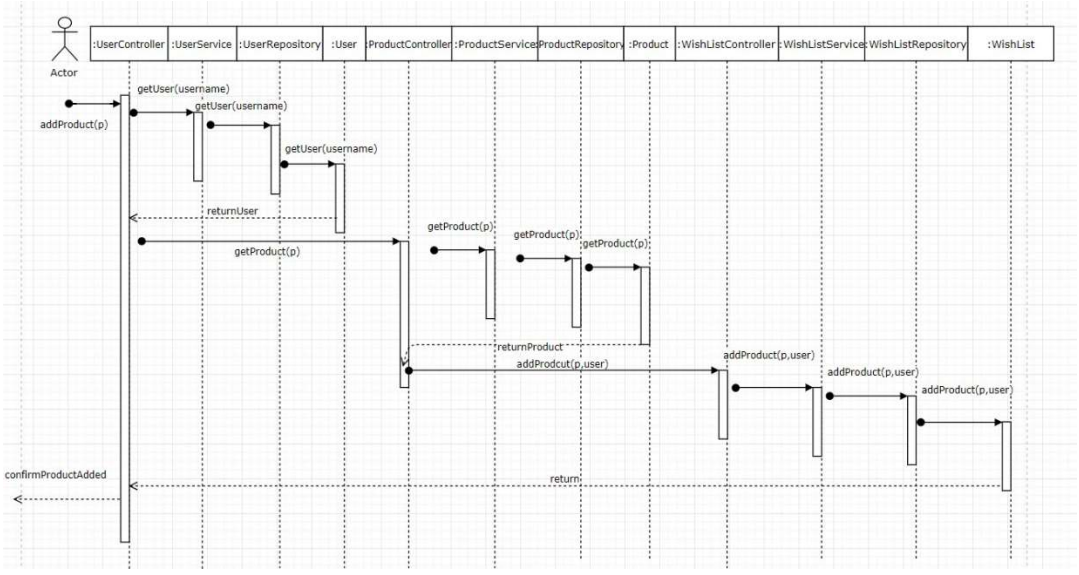
1.1 Dynamic Behavior

2 Relevant Scenarios are the following:

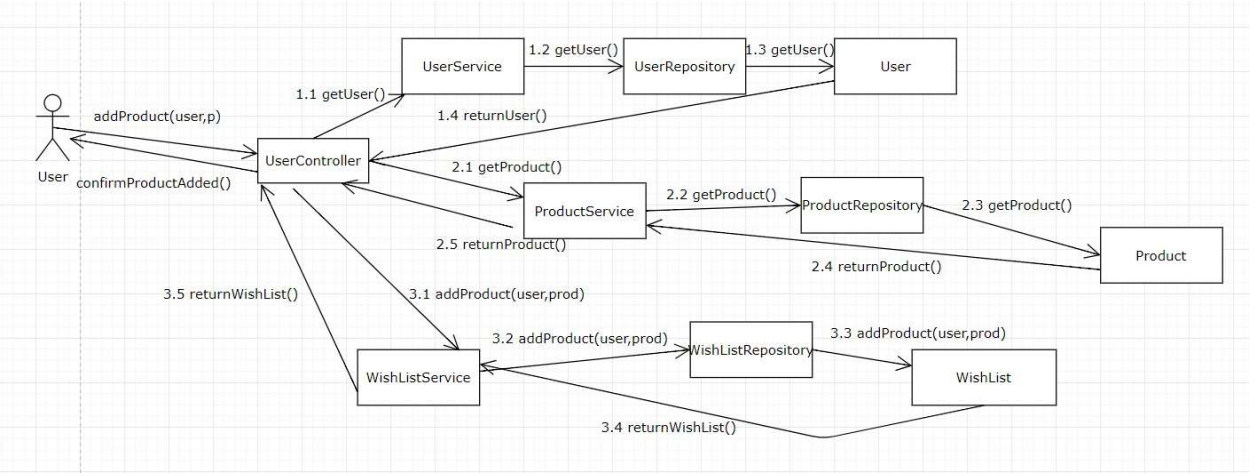
- The user wants to add a product to his/her wishing list.
- The user wants to obtain a recommended list that consists of products suitable for one of his/her sicknesses.

Use case 1: Add product to the wish list

The sequnce diagram is the following:

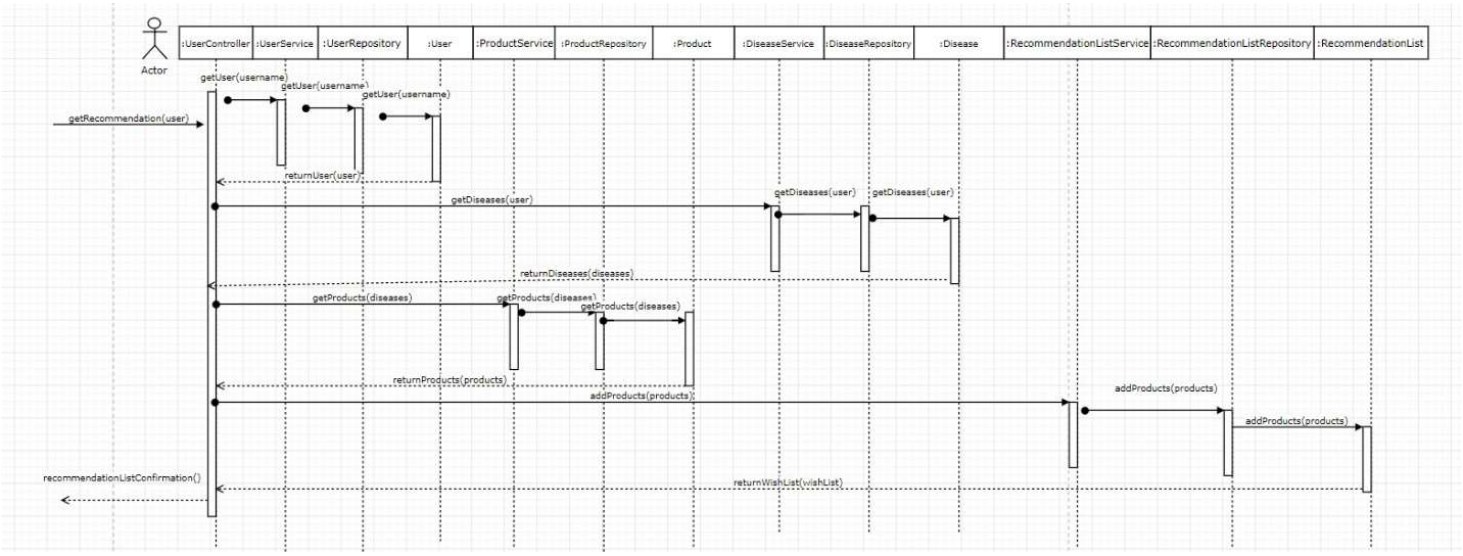


The communication diagram is the following:

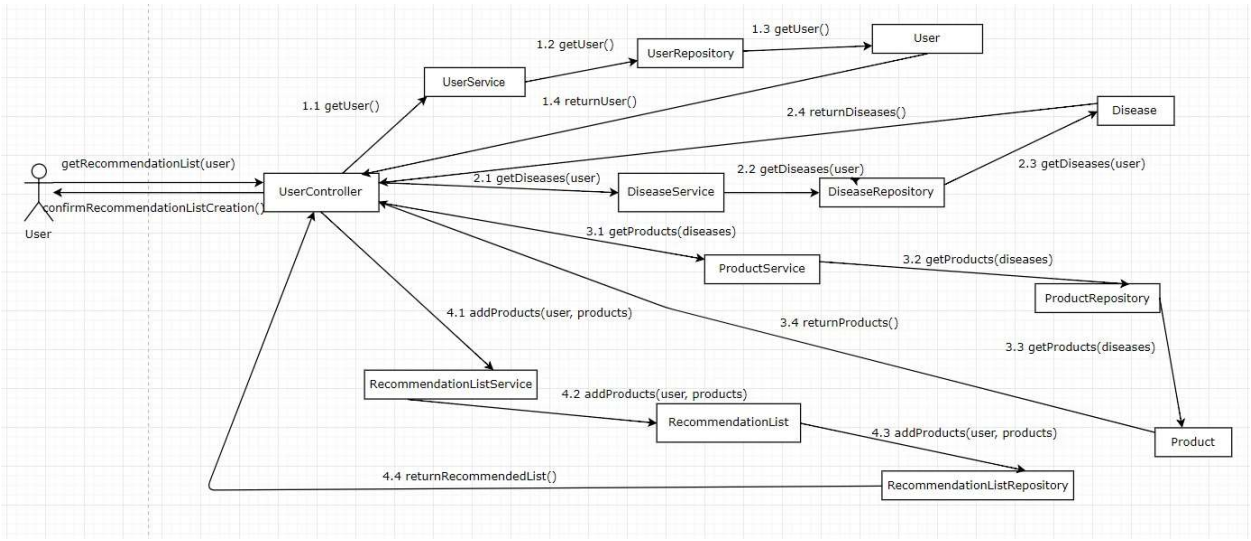


Use case 2: Get recommended List

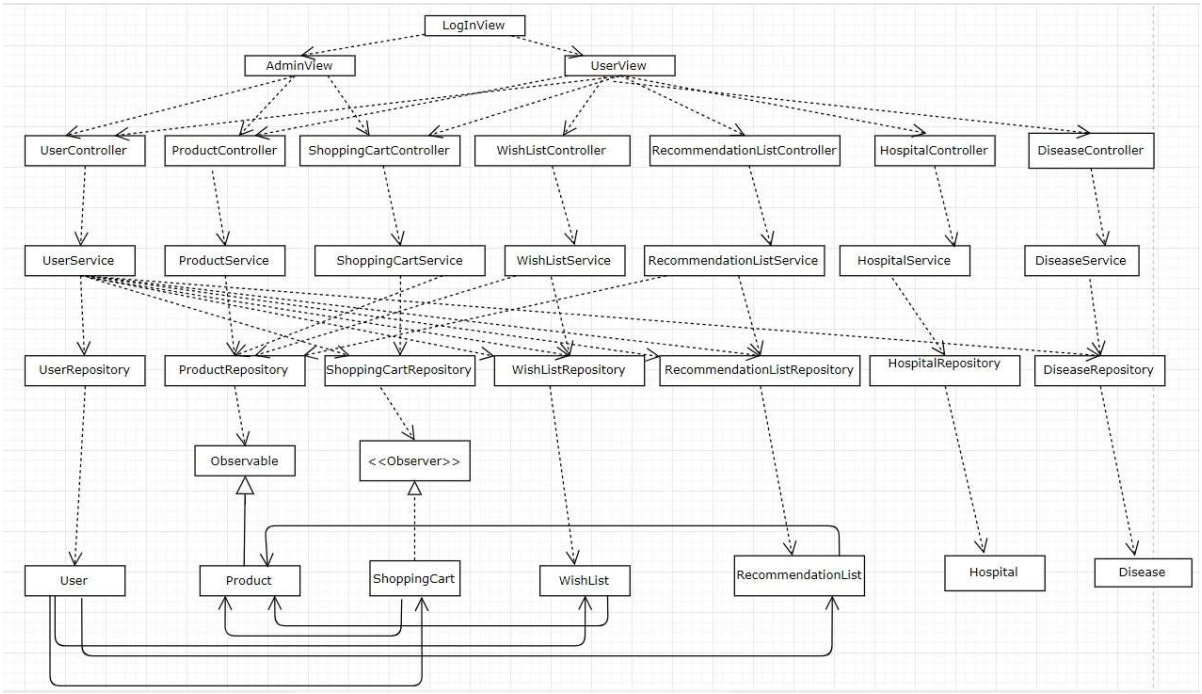
The sequence diagram is the following:



The communication diagram is the following:



1.2 Class Design



The used design patterns are : Inversion of Control, Dependency Injection, Observer Design Pattern and Builder Design Pattern.

The Observer Design Pattern:

It is used in order to let the patients know when a product, herb or fruit, that they were interested

in, is now on stock again. The class Patient will implement the Observer interface and the Fruit and Herb class will extend the Observable superclass.

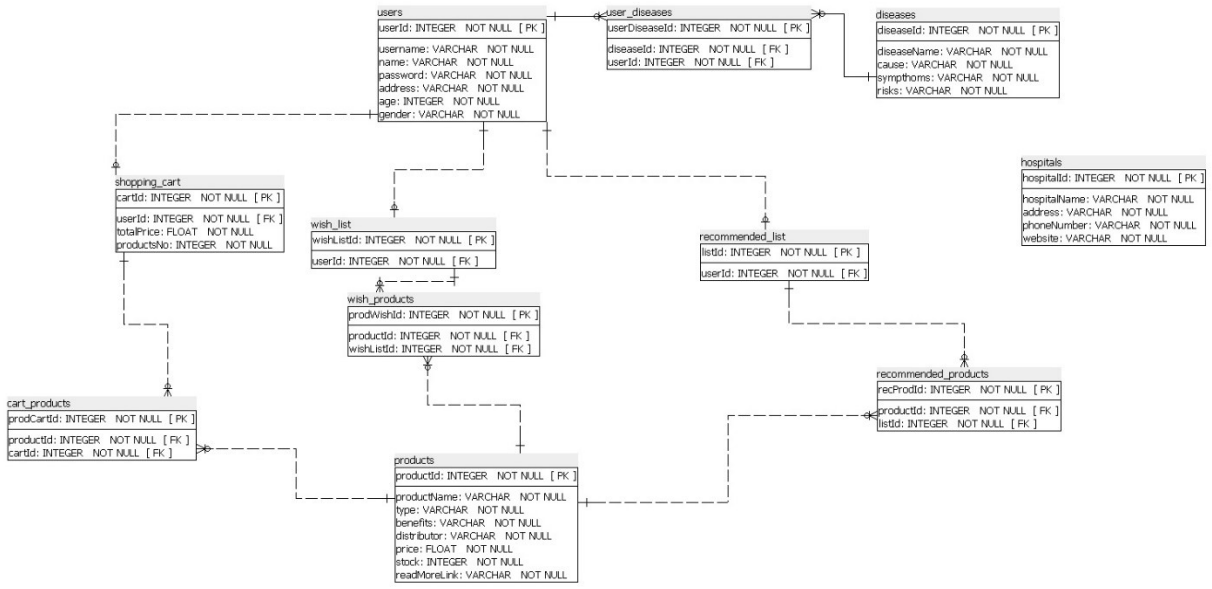
The Builder Design Pattern:

The Builder Design Pattern is used in order to set only certain fields of a class. For example, I choose to use it when creating the DTO classes. We have the main classes from the Entities package, from the Persistance Layer and we want to be able to create different views of the same entity – which is to set a value only on some of the fields of the class.

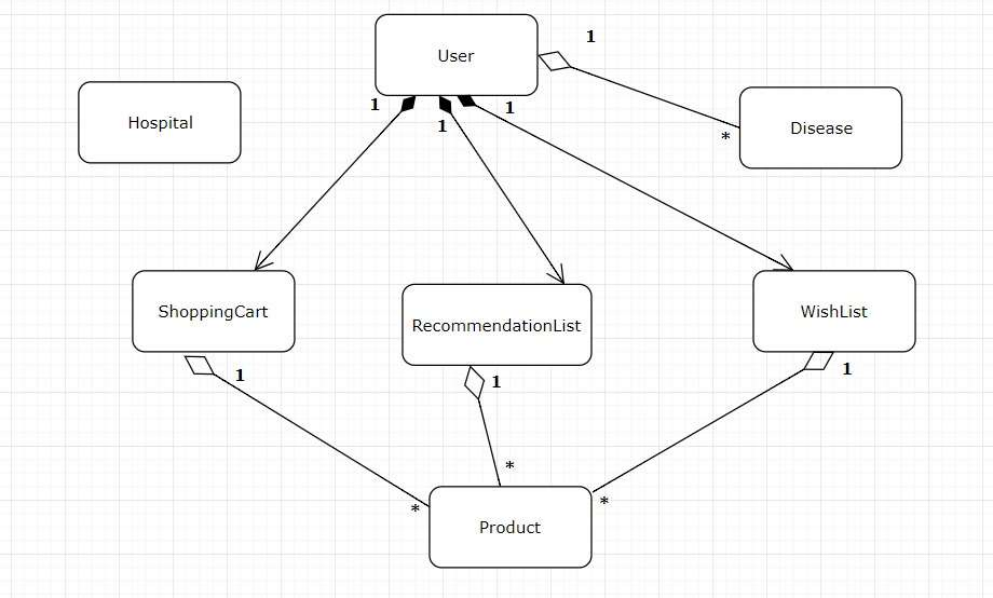
2. Data Model

The persistance data will be stored in a database. It will be modeled using a Relational Database Model. The database will be generated using an ORM. The used Database Management System is MySql.

The Relational Database Diagram of the System



The Data Model Representation



3. Unit Testing

I will use Mokito to test my application. Mocking is primarily used in unit testing. An object under test may have dependencies on other (complex) objects. To isolate the behaviour of the object we want to test we replace the other objects by mocks that simulate the behavior of the real objects. In short, mocking is creating objects that simulate the behaviour of real objects.

I will write small tests in order to mainly test the functions implemented in the service layer.

IV. Construction and Transition

1. System Testing

The system has been tested using the client-side application. The user, which can be an admin or a regular user interacts directly with the application through the friendly user interface implemented using the AngularJS framework and Bootstrap.

The following usual and particular cases have been tested:

- The user log in, the username and password validation.
- The distinction between an user account and an admin account.
- The creation of a new user account and user account validation.
- The system response when the provided data for an user account is not valid.
- The creation of the wish list, recommendation list and shopping cart when a new user account is created.
- The placing of an order containing the products in the shopping cart.
- The operations performed on the shopping cart – add new products , remove products, increase or decrease the quantity of any product.
- The update of the total cost of a certain order and the update of the number of products in the shopping cart.

- The operations the admin can perform on the data stored in the database.
- The update of product stock after an order is placed and the stock must be updated.
- The failure to add to the shopping cart a greater quantity of a product than the one that exists.
- The failure to add to the shopping cart a product that is not in the stock.
- The action of deleting a user account.

The system has managed to pass all the tests and the behaviour was the expected one.

2. Future improvements

Future improvements could be the following ones:

- The user can get a recommendation of a suitable hospital based on his/her diseases.
- The user can get notified when a product in his/her wish list gets avaialble.
- The user can make an appointment at a desired hospital.
- The user can move the products from the wish list directly in the shopping cart.
- The user can move the products from the recommendation list directly in the shopping cart.
- The navigation between the views of the application can be made easier.
- The application security can be implemented using Spring Security.
- The user log in part of the client-side application is implemented using cookies.

V. Bibliography

http://www.mkyong.com/tutorials/spring-boot-tutorials/
https://www.w3schools.com/angular/
https://www.javaworld.com/article/2077674/java-web-development/a-standardized-object-relational-mapping-mechanism-for-the-java-platform.html
https://www.tutorialspoint.com/design_pattern/observer_pattern.htm
https://zeroturnaround.com/rebellabs/spring-framework-annotations-cheat-sheet/
https://spring.io/guides/tutorials/bookmarks/