Shopping list

Analysis and Design Document

Student:Tuturuga Nicolae

**Group:30238**

Revision History

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

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

*[Present the project specification]*

# Description

Project name is shopping list, the main idea of these project is to create a shareable shopping list for more users. An user will be able to create an account and create a group, or join a group for example a family group. In that group he will be able to post products to buy for example milk, or mark a product as bought and also he will be able to add the price he paid for it. An user will also have the possibility to add some requirements like the producer of milk for example Zuzu, and he will be able to indicate a shop to buy from, this constraints will be optional.

# Elaboration – Iteration 1.1

# Domain Model

*[Define the domain model and create the conceptual class diagrams]*

An user will be able to create a group, and he will be able to join many groups, not only one.

A group will be created from 0 or more members, an user will be able to create one, and other users will be able to join with a group id.

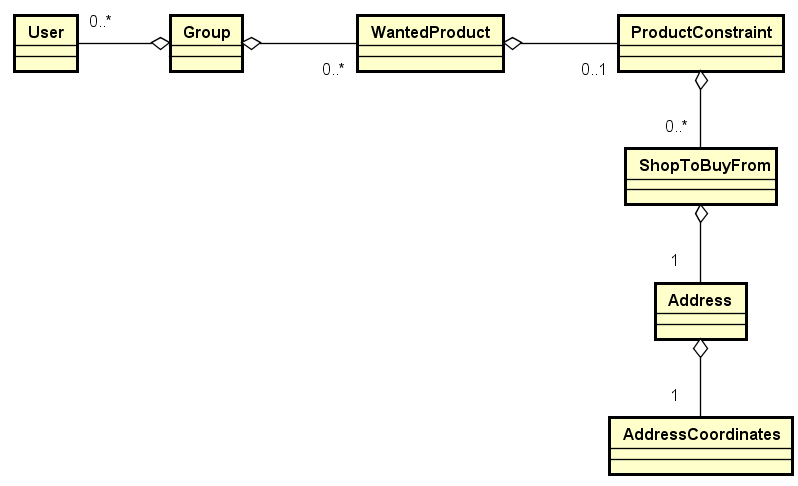
Wanted product represents the product a user wants to buy, users can add one or more products.

Product constraint its an optional field, it can contain the product producer, the shop to buy from and other constraints like.

Shop to buy from represents the shop selected by the user from which other members should buy wanted product.

Address represents the address shop.

Address coordinates represents shop’s address coordinates with latitude and logitude.



# Architectural Design

## Conceptual Architecture

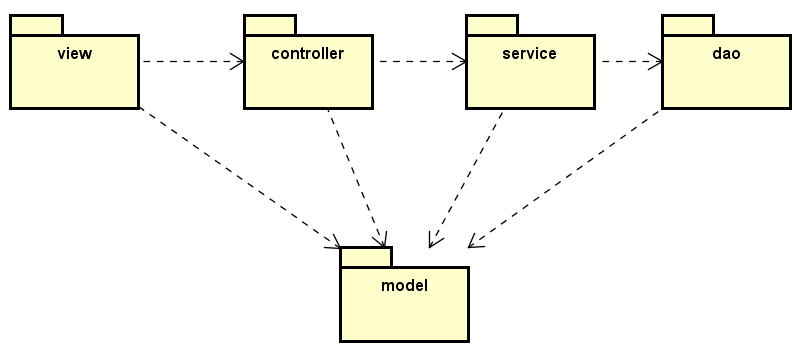
*[Define the system’s conceptual architecture; use an architectural style and pattern - highlight its use and motivate your choice.]*

The architecture I choose to use is a Model-View-Controller architecture. The reasons I choose this architecture is because of advantages:

* MVC enables logical grouping of related actions on a controller together. The views for a specific model are also grouped together.
* Low coupling among models, views or controllers.
* Separation of responsibilities makes future modification or development easier.
* Models can have multiple views .

## Package Design

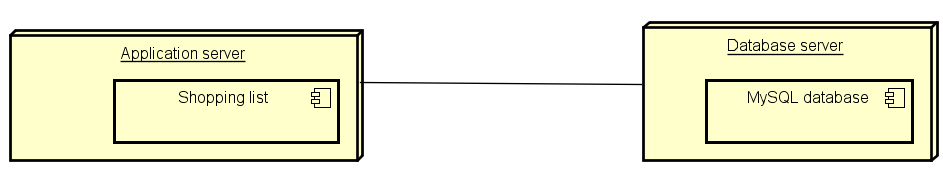
*[Create a package diagram]*



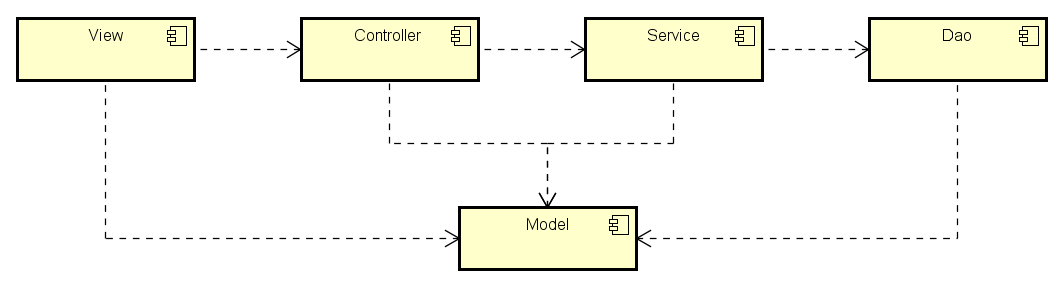
## Component and Deployment Diagrams

*[Create the component and deployment diagrams.]*

Deployment



Component

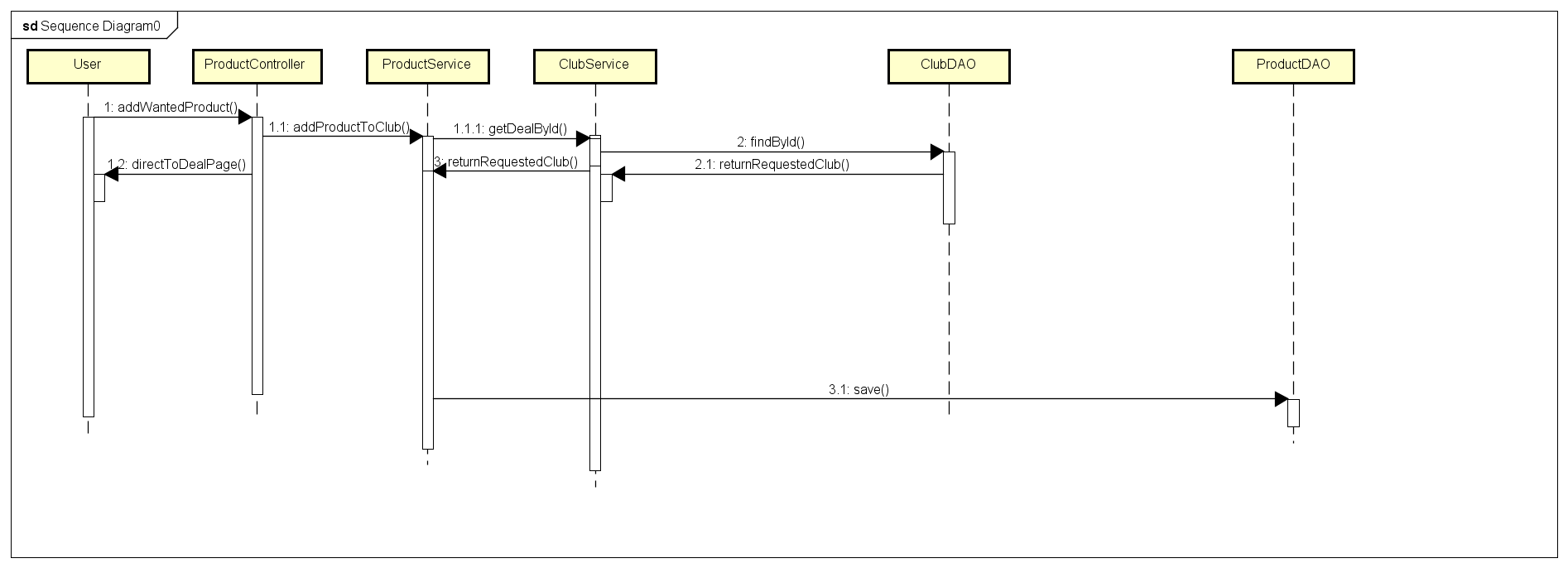


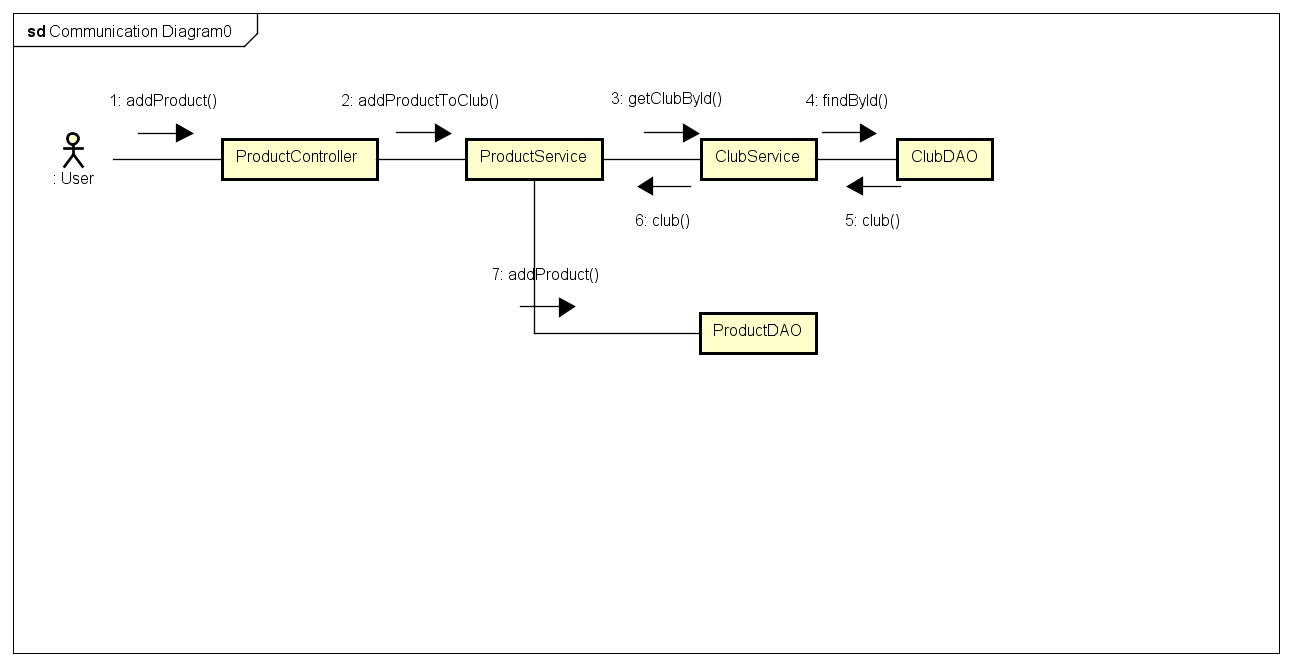
# Elaboration – Iteration 1.2

# Design Model

## Dynamic Behavior

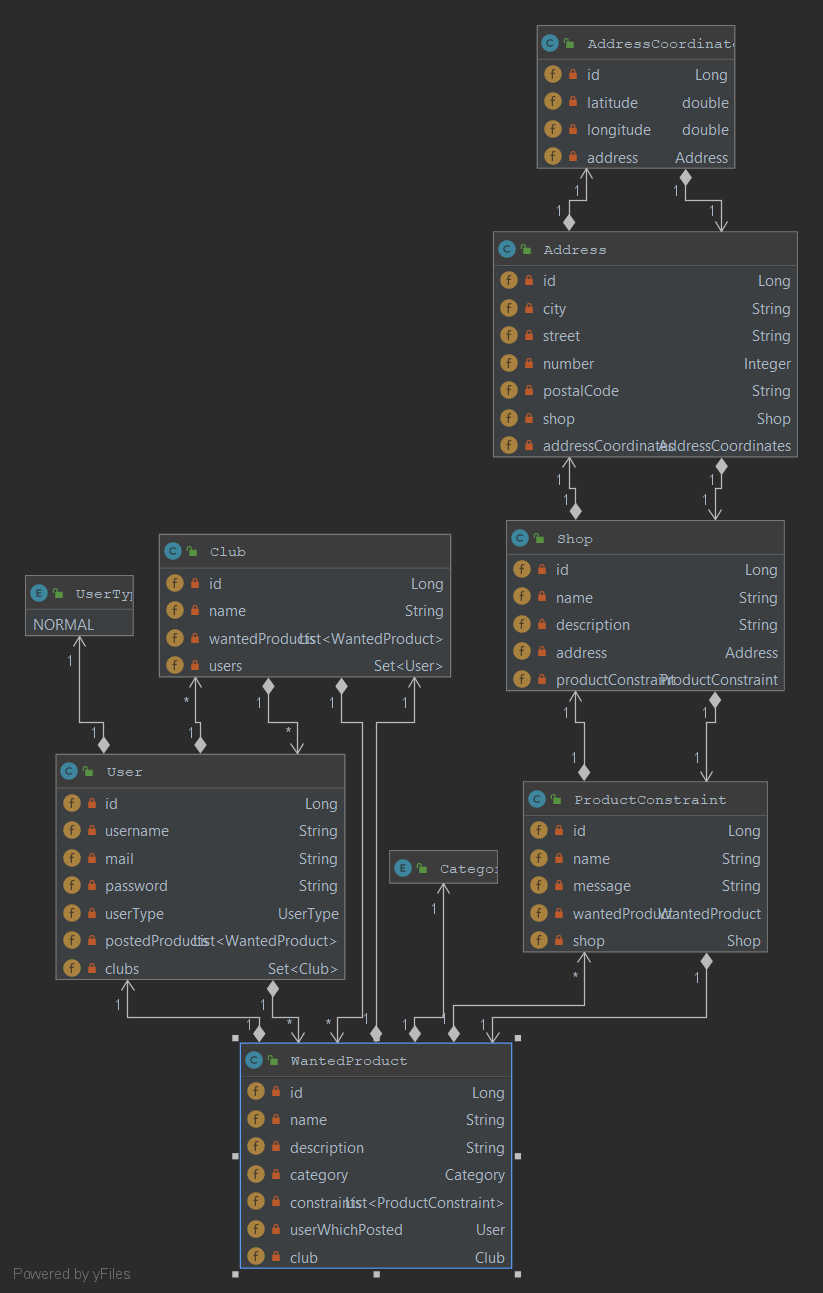
*[Create the interaction diagrams (1 sequence, 1 communication diagrams) for 2 relevant scenarios]*

**

**

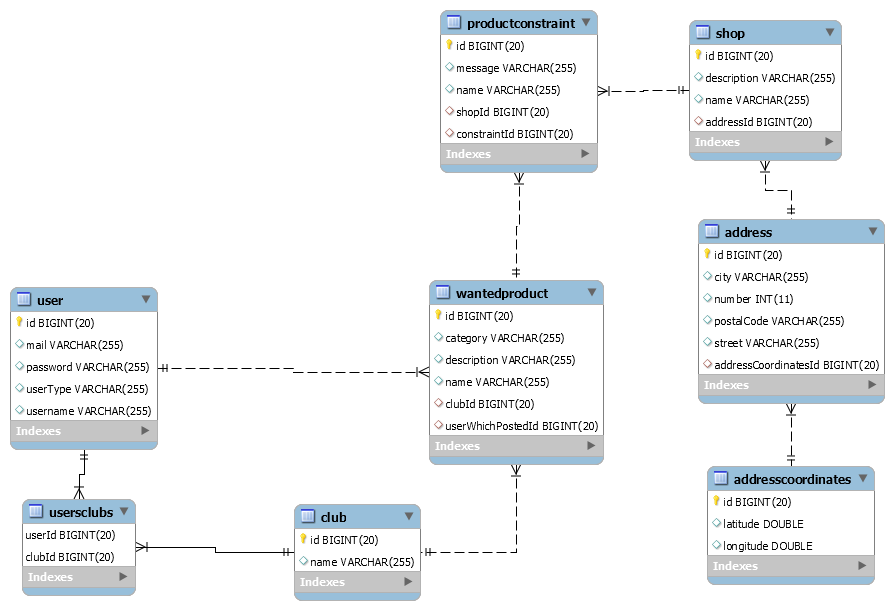
## Class Design

*[Create the UML class diagram; apply GoF patterns and motivate your choice]*

**

# Data Model

*[Create the data model for the system.]*

**

# Unit Testing

*[Present the used testing methods and the associated test case scenarios.]*

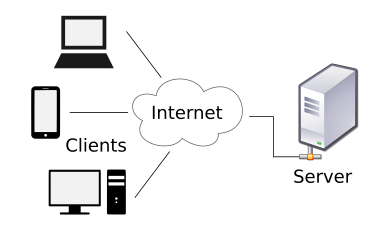
All data will be tested using unit tests. Tests will be created to test the comportment of the application for success cases and failure cases. For example of a success case is at a register when all user data is valid, and a failure case is when we try to get some objects (products, users etc.) that are not existent in database.

# Elaboration – Iteration 2

# Architectural Design Refinement

*[Refine the architectural design: conceptual architecture, package design (consider package design principles), component and deployment diagrams. Motivate the changes that have been made.]*

Client–server model is a [distributed application](https://en.wikipedia.org/wiki/Distributed_application) structure that partitions tasks or workloads between the providers of a resource or service, called [servers](https://en.wikipedia.org/wiki/Server_(computing)), and service requesters, called [clients](https://en.wikipedia.org/wiki/Client_(computing)).



**MVC** (model-view-controller) 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. MVC decouples these major components allowing efficient code reuse and parallel development. MVC components:

**Model:**

The central component of the pattern. It is the application’s dynamic data structure, independent of the user interface. It directly manages the data, logic and rules of the application.

**View:**

Any representation of information such as a chart, diagram or table. Multiple views of the same information are possible, such as a bar chart for management and a tabular view for accountants.

**Controller:**

Accepts input and converts it to commands for model and view.

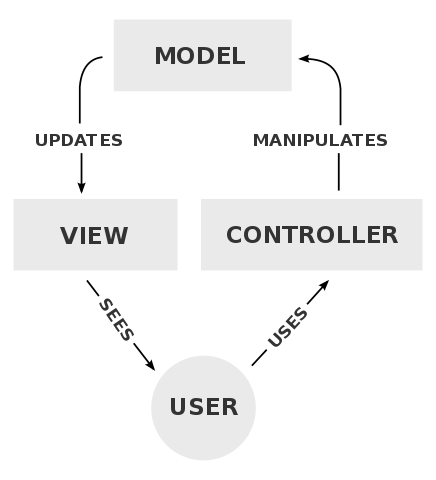
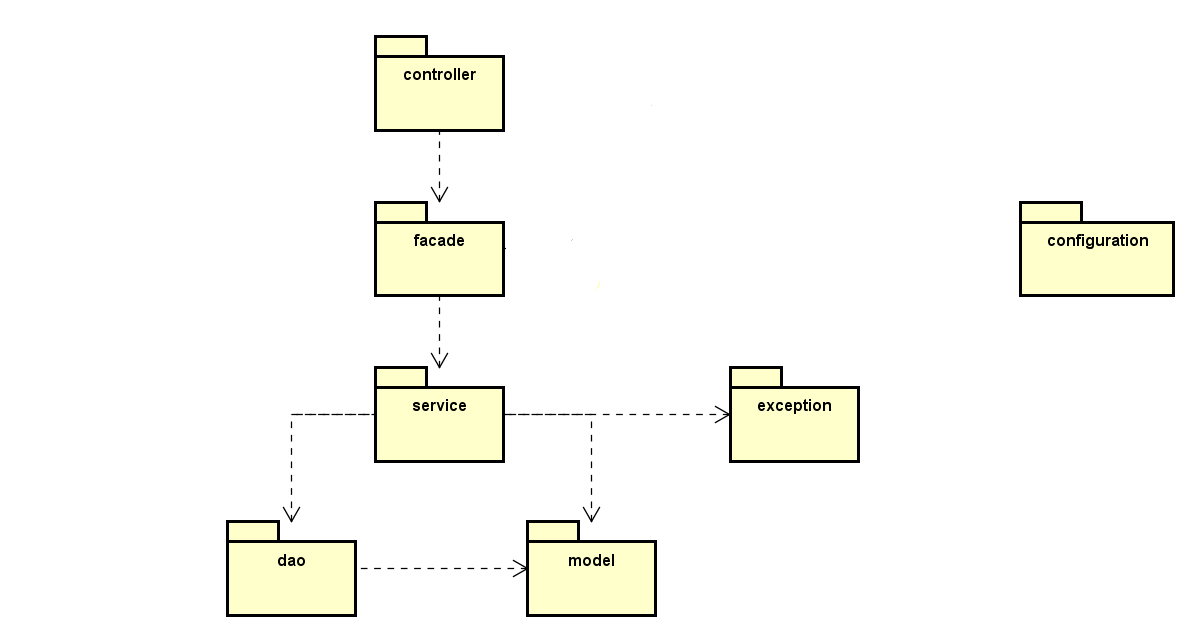
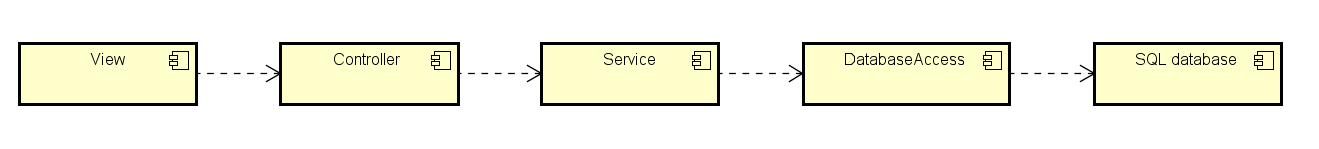


Table Data Gateway is another architectural pattern I used, it consists in using an object which acts as a gateway to a database table. The idea is to separate the responsibility of fetching items from a database from the actual usages of those objects. Users of the gateway are then insulated from changes to the way objects are stored in the database.

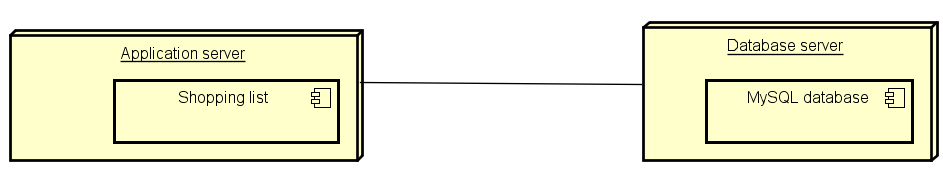
**Package design**

****

**Component**

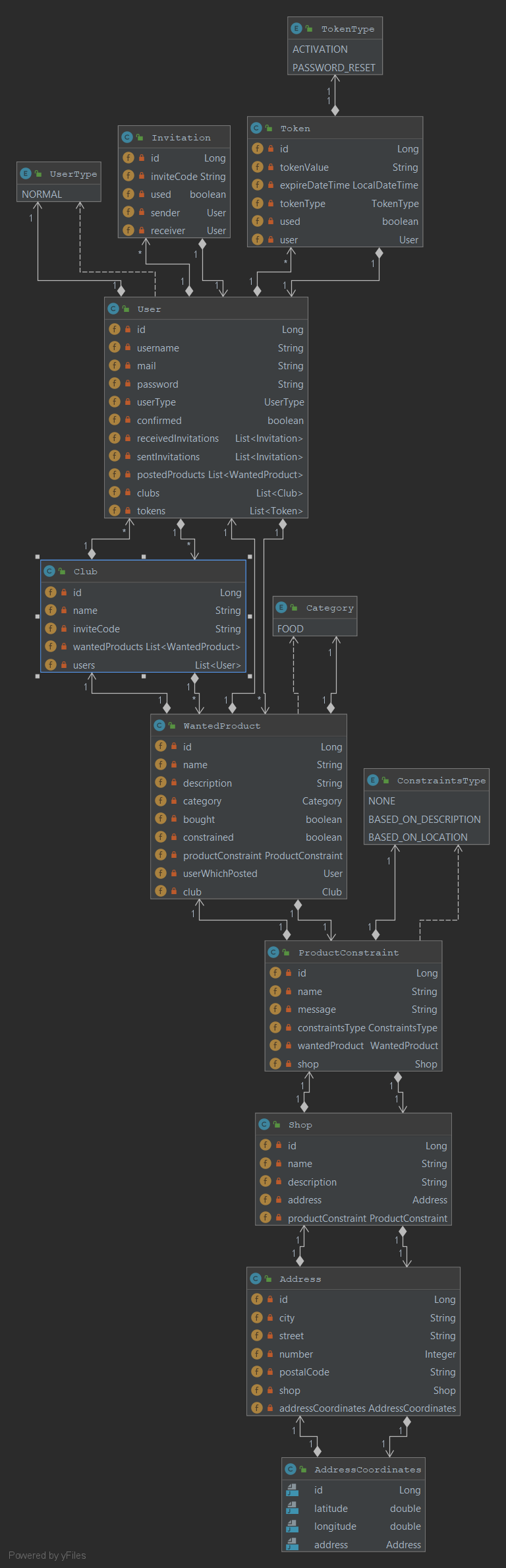
****

**Deployment**



# Design Model Refinement

## *[Refine the UML class diagram by applying class design principles and GRASP; motivate your choices. Deliver the updated class diagrams.]*

****

# Construction and Transition

# System Testing

*[Describe how you applied integration testing and present the associated test case scenarios.]*

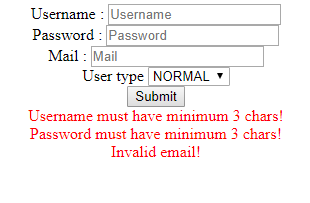
The system was be tested using unit tests and integration test, I used spring boot test framework to load data in an in-memory database, and test all possible scenarios when user inserts invalid data and when user insert valid data. I will also use mocks for tests to see if expected methods are called.

Example of successful use case:

Users select wanted data for a new account exp: username = nicu, [mail = nicu@yahoo.com](mailto:mail%20=%20nicu@yahoo.com), password = 12345 and presses the button which send the request to application, because it is correct he will be redirected to login page.

Example of use case that leads to fail:

Users select wanted data for a new account exp: username = 1, mail = nicu, password = 0 and presses the button which send the request to application, because data is incorrect he will be redirected to register page where will also appear error messages.



# Future improvements

*[Present future improvements for the system]*

A future improvement will be to add chat for a group.

# Bibliography

[https://en.wikipedia.org](https://en.wikipedia.org/)