Online Medieval Weapon and Armour Store
Analysis and Design Document

Nicolae-Florian Onica
30432

Revision History

Date	Version	Description	Author
4.04.2018	1.0	Domain Model, Architectural Design, Component and Deployment diagrams	Nicolae-Florian Onica
21.05.2018	1.1	Revamp of Architecture, Domain Model, Design Patterns and Class Diagram	Nicolae-Florian Onica

Table of Contents

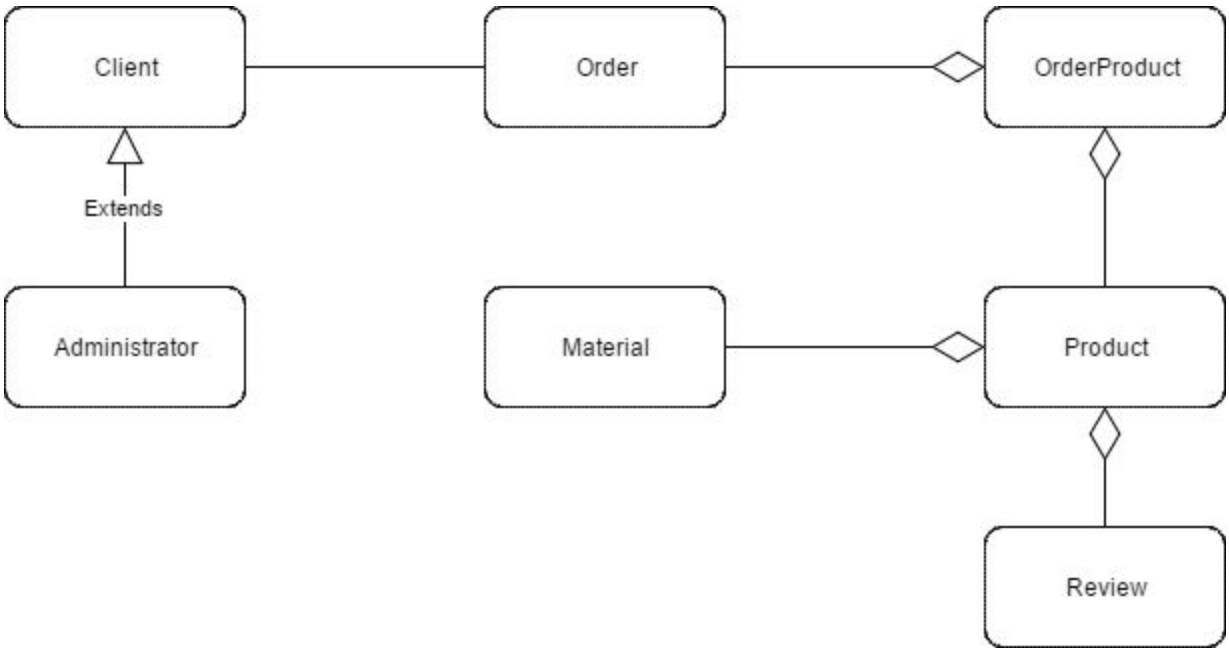
I. Project Specification

This project is an online medieval weapon and armor store that has listings of various weapons along with their features. The project allows users to buy weapons, armor or parts of them online. They can check stats, read reviews and build their own custom weapon.

II. Elaboration – Iteration 1.1

1. Domain Model

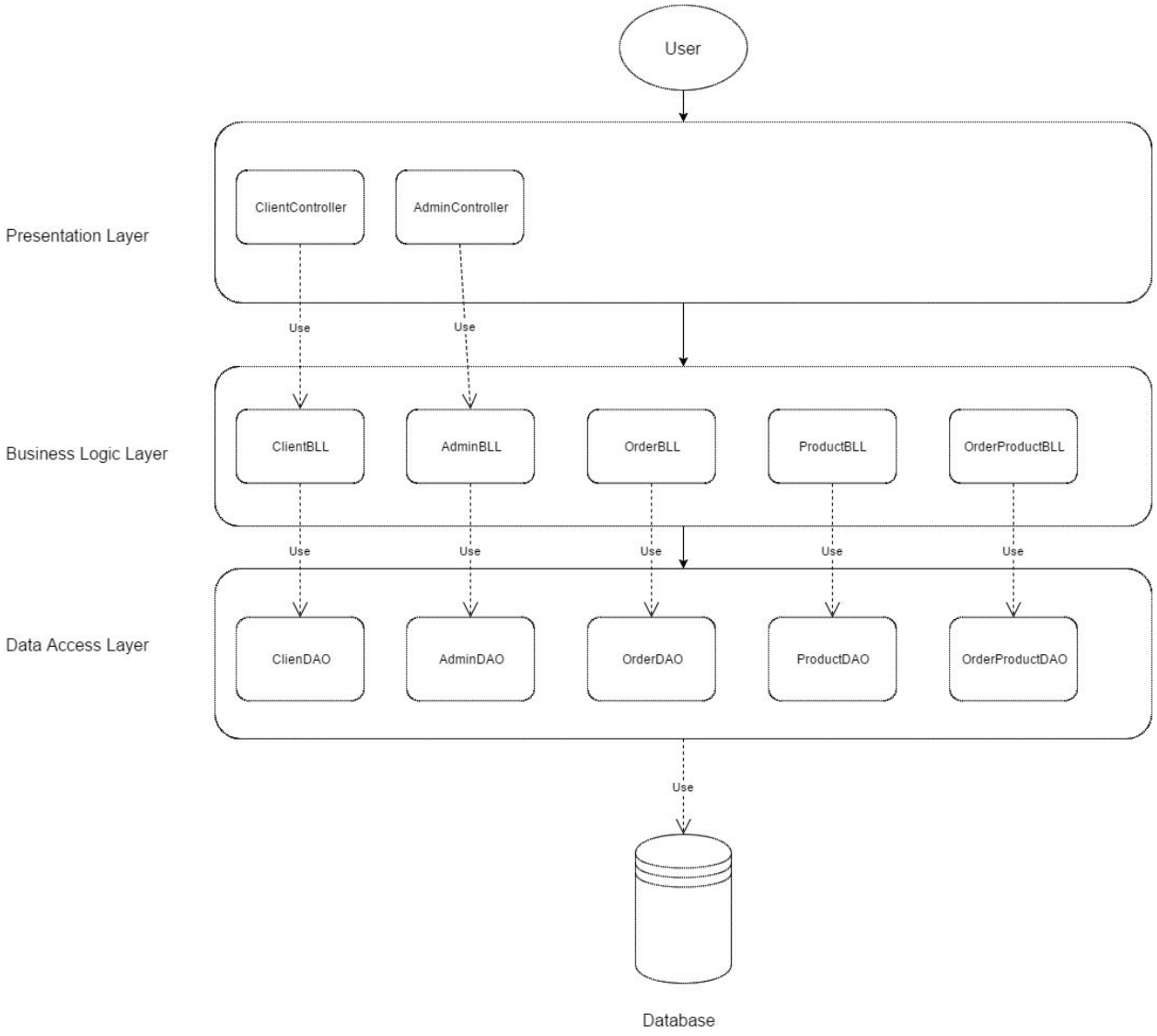
The domain model has 2 actors, the client and the administrator. The user places the order containing the products which he wants or create a customized item which will be added to the order. The administrator performs CRUD operations on products and checks user reviews.



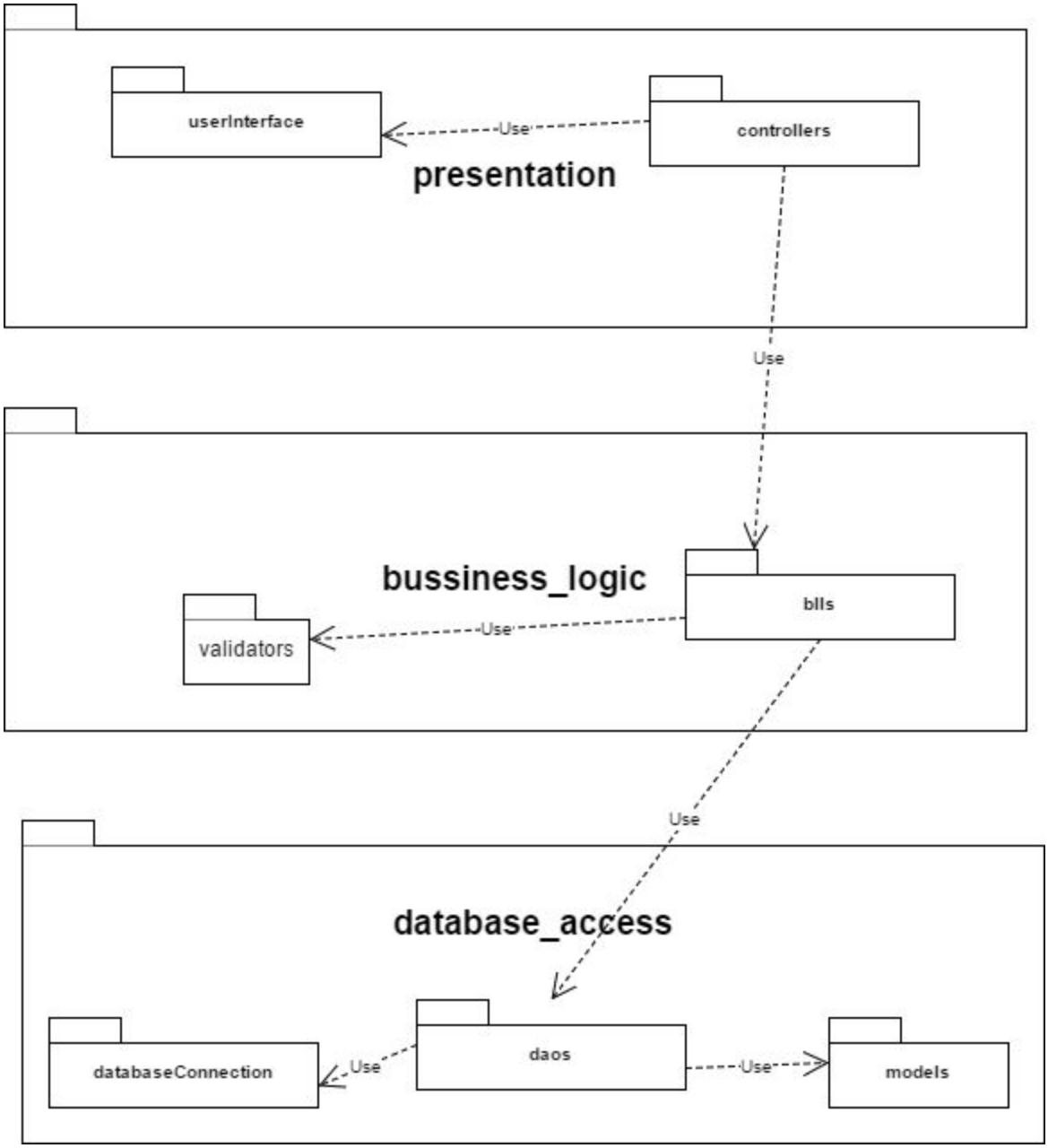
2. Architectural Design

3. Conceptual Architecture

The architectural pattern used for this application is the layered architecture pattern. The components within this pattern are organized into horizontal layers, each layer performing a specific role within the application (e.g., presentation logic or business logic). In this case three layers will be used: presentation, business and database layer.

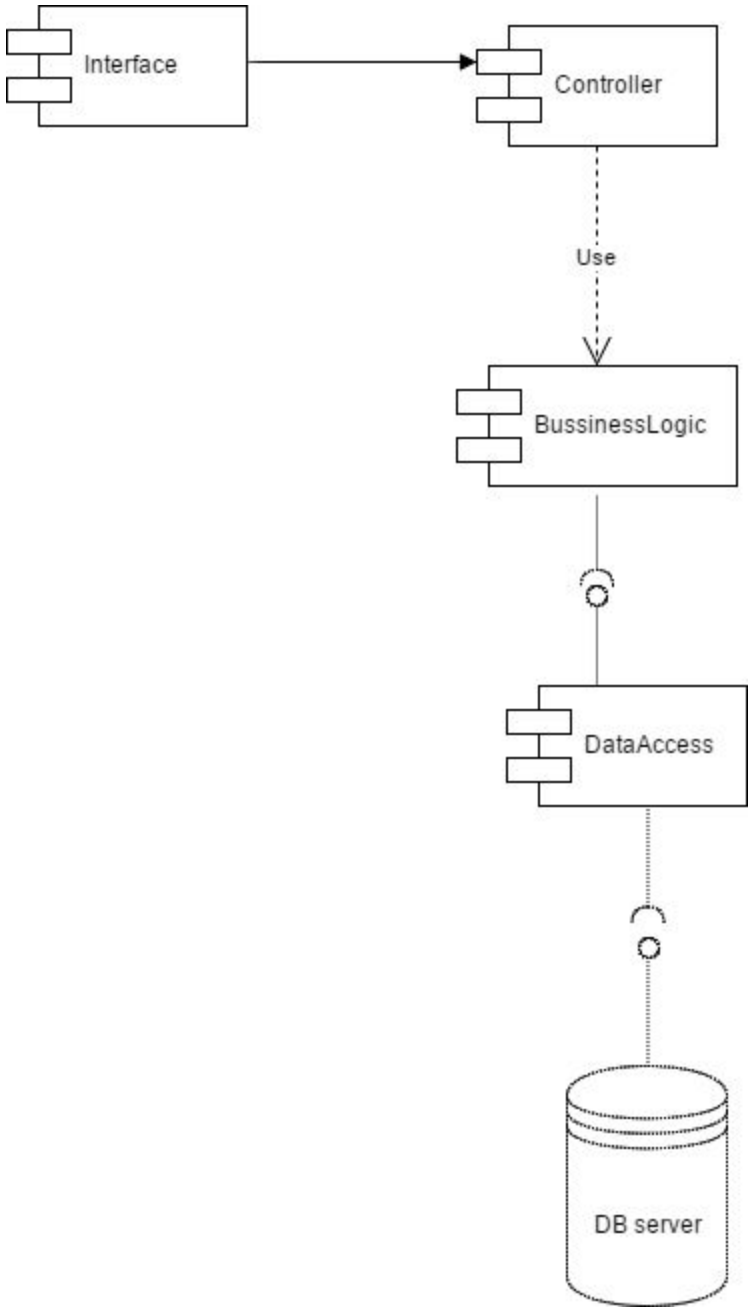

User
Presentation Layer
ClientController
AdminController
Use
Use
Business Logic Layer
ClientBLL
AdminBLL
OrderBLL
ProductBLL
OrderProductBLL
Use
Use
Use
Use
Use
Data Access Layer
ClienDAO
AdminDAO
OrderDAO
ProductDAO
OrderProductDAO
Use
Database

4. Package Design

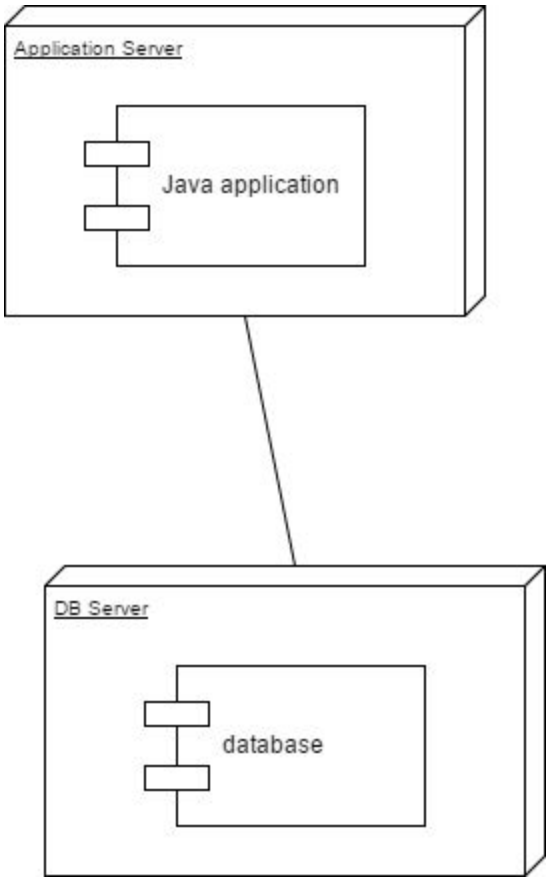


5. Component and Deployment Diagrams

Component diagram



Deployment diagram



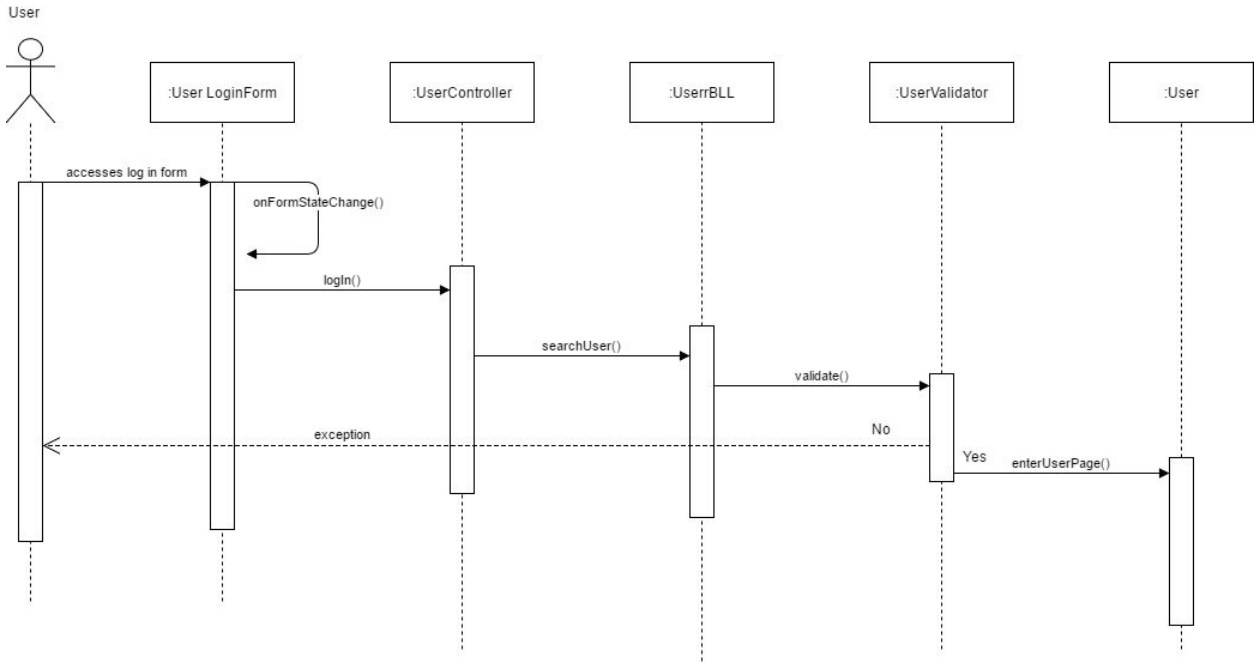
III. Elaboration – Iteration 1.2

1. Design Model

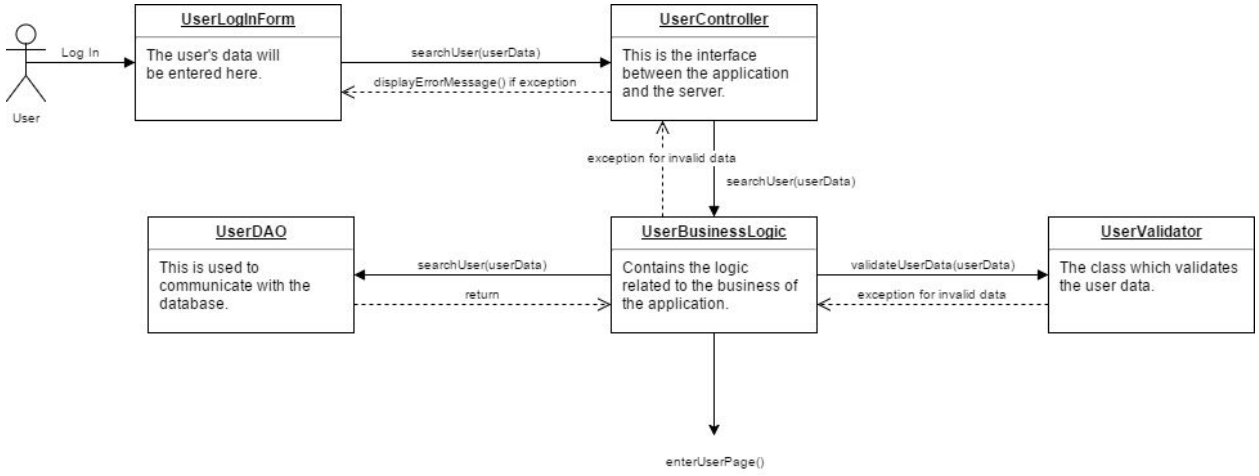
1.1 Dynamic Behavior

Login

Sequence diagram

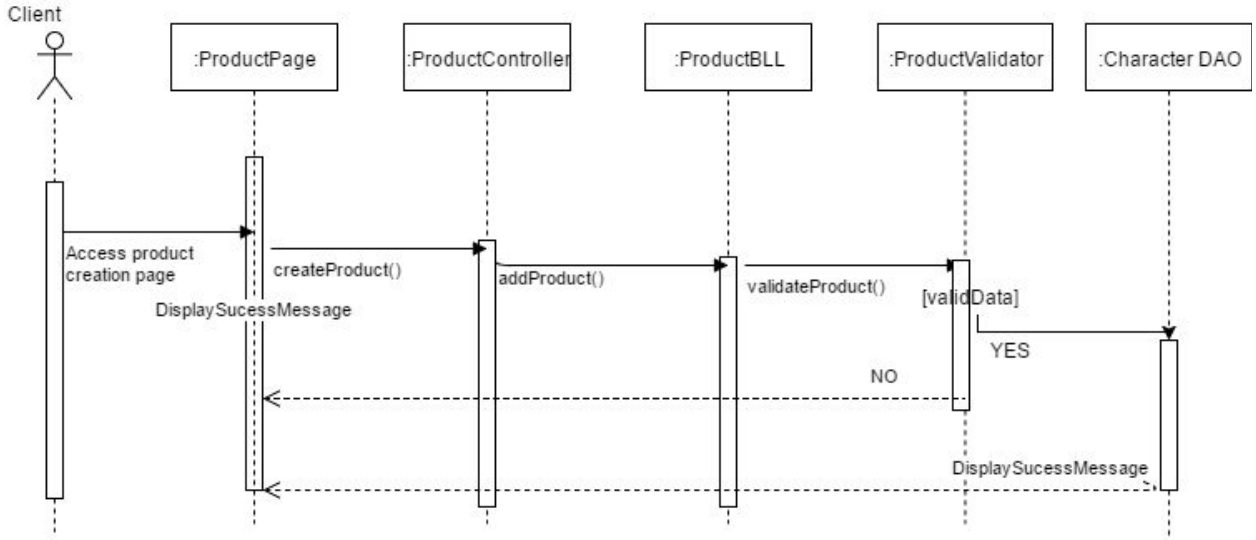


Communication diagram

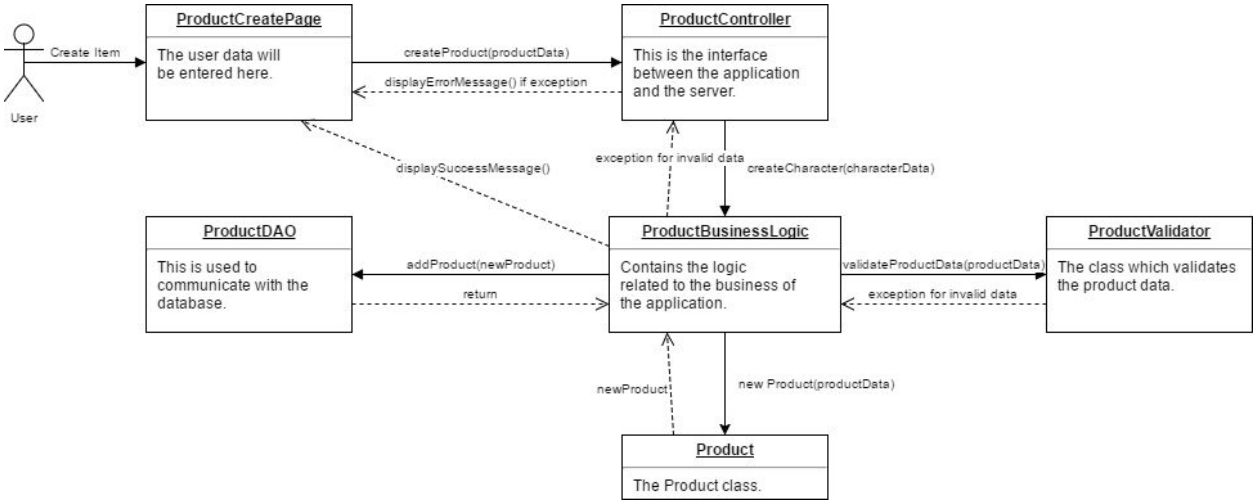


Create item

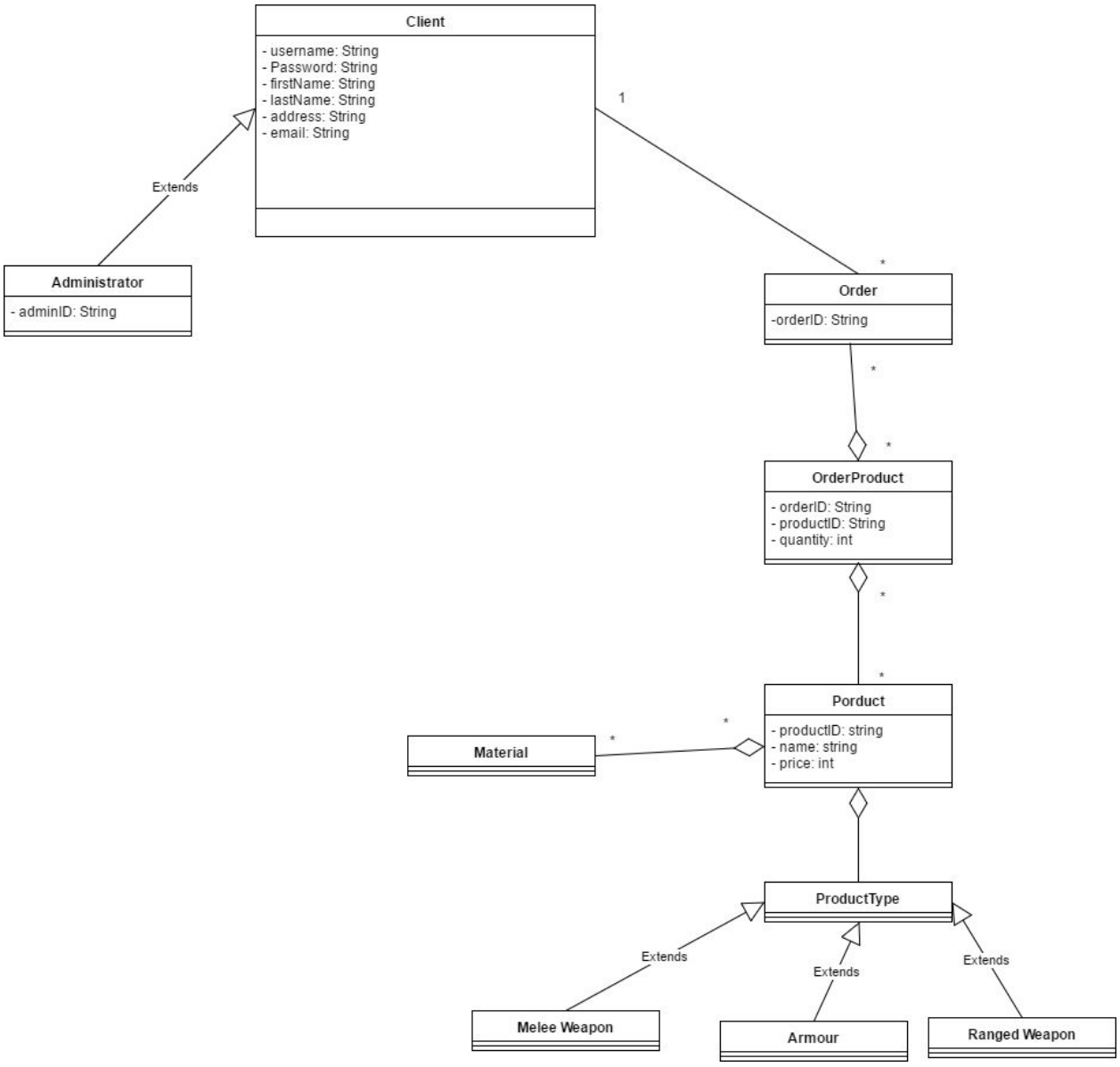
Sequence diagram



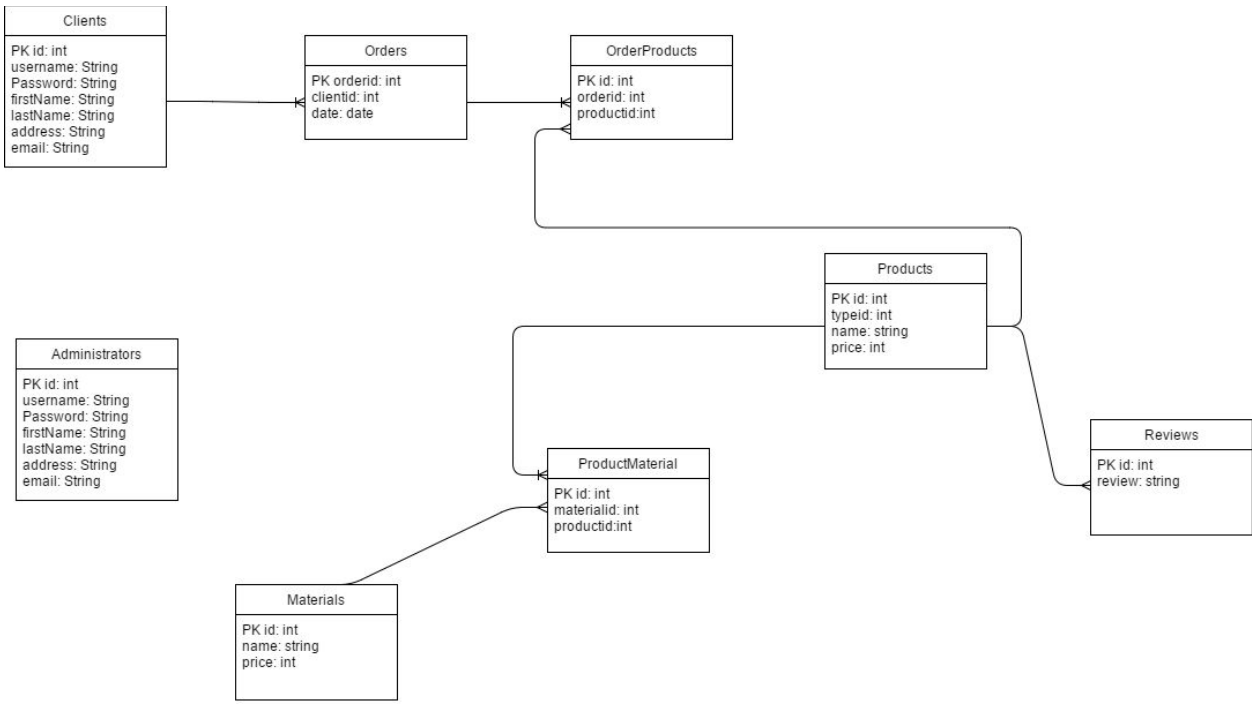
Communication diagram



1.2 Class Design



2. Data Model



3. Unit Testing

The testing strategy used is unit testing which implies separating the code into unit and testing each unit to see if they meet the required functionality.

Test case scenarios:

- CRUD operations on products
- User login/logout
- Creating an order

IV. Elaboration – Iteration 2

1. Architectural Design Refinement

The architectural pattern used for this application is the layered architecture pattern. The components within this pattern are organized into horizontal layers, each layer performing a specific role within the application (e.g., presentation logic or business logic). In this case three layers will be used: presentation, business and database layer.

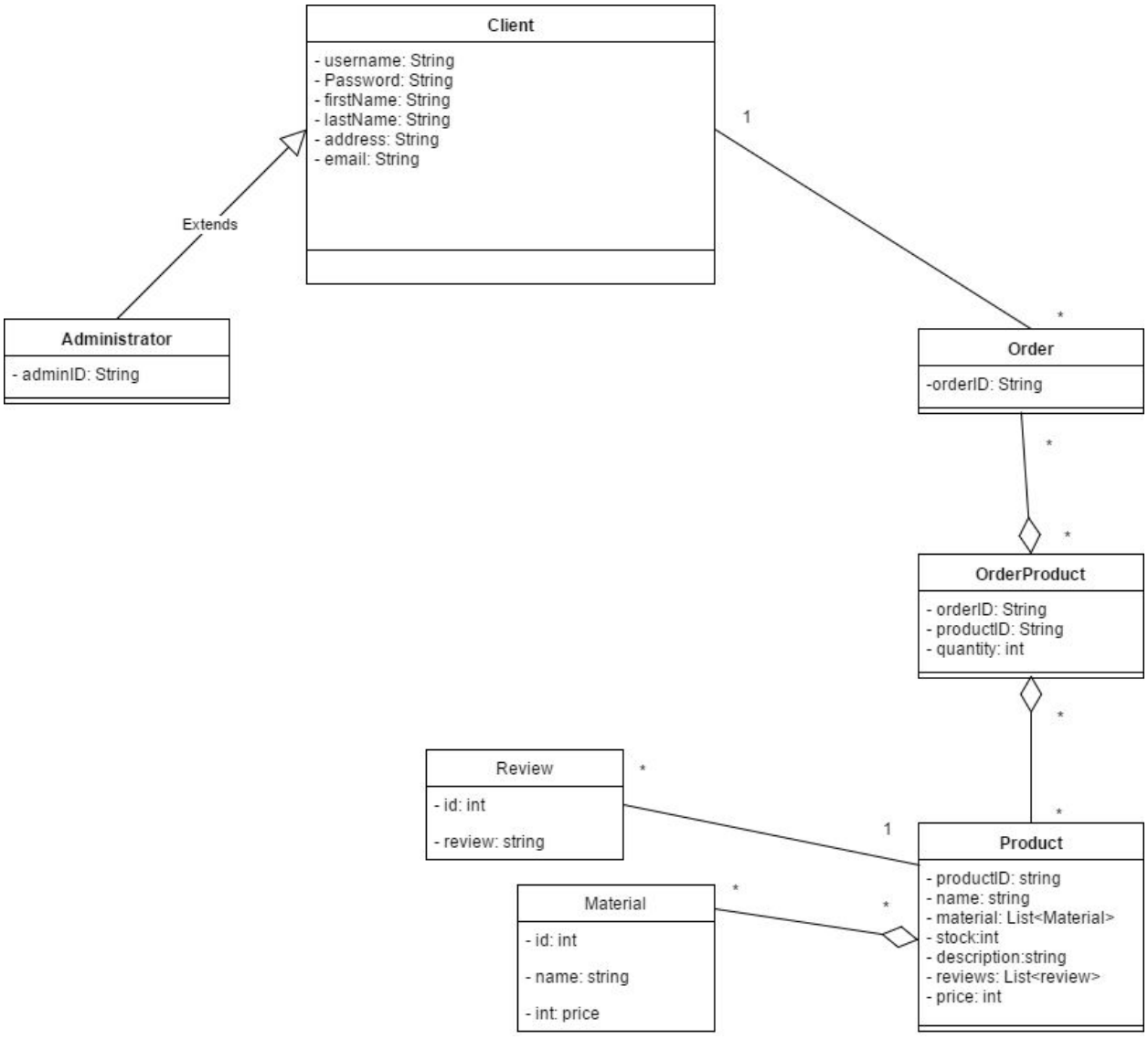
MVC pattern

This architecture has the following components: the model, which consists of the classes which are used to store and manipulate state, typically in a database of some kind, the view, which contains the user interface bits necessary to render, the model to the user, and the controller, which is the brains of the application. The controller decides what the user's input was, how the model needs to change as a result of that input, and which resulting view should be used.

Design patterns

One design pattern used in this application is the Observer pattern. It has two main components, an Observable class(in our case the article) which will be the subject of observation for the Observer interface(in this case the admin which should view reviews flagged), which has an update method which is being called anytime the Observable updates.
Builder pattern builds a complex object using simple objects and using a step by step approach. It is used here for the create custom item use-case implementation.
Factory pattern is used to create an object without exposing the creation logic to the client and refer to newly created object using a common interface. Here it is used to generate admin accounts.

2. Design Model Refinement



V. Construction and Transition

1. System Testing

The testing strategy used is unit testing which implies separating the code into unit and testing each unit to see if they meet the required functionality.
Test case scenarios:

2. CRUD operations on products
3. User login/logout
4. Creating an order

5. Future improvements

VI. Bibliography