Best Furniture Deals

Analysis and Design Document

Student: Tuturuga Nicolae

**Group: 30238**

Table of Contents

1. Requirements Analysis 3

1.1 Assignment Specification 3

1.2 Functional Requirements 3

1.3 Non-functional Requirements 3

2. Use-Case Model 3

3. System Architectural Design 3

4. UML Sequence Diagrams 3

5. Class Design 3

6. Data Model 3

7. System Testing 3

8. Bibliography 3

1. Requirements Analysis

# Assignment Specification

[Application description]

Application title is Best Furniture Deals which should be a deal search engine for furniture products. Users should be able to create accounts, login and search for deals, they also should be able to filter deals by price, name and type and add associated product to their cart. They should be able to pay only with cash and post a feedback when order is completed. Their orders should be validated and updated by stuff.

# Functional Requirements

*[Present the functional requirements]*

# Normal users should be able to:

* search for furniture
* create account
* login
* add product to cart and checkout
* provide feedback

Staff should be able to:

* validate orders
* update order state
* manage deals
* manage furniture

# Non-functional Requirements

*[Discuss the non-functional requirements for the system]*

First requirement is to implement the application and test it as we build it, another requirement is to use an OOP language (I choose Java). Next requirements are about design, first of them is to use a client server architecture, another one consist of implementing observer design pattern to notify user when an order state is updated. And another one is to validate all the inputs of the application.

2. Use-Case Model

*[Create the use-case diagrams and provide one use-case description (according to the format below).*

*Use-Case description format:*

*Use case: <use case goal>*

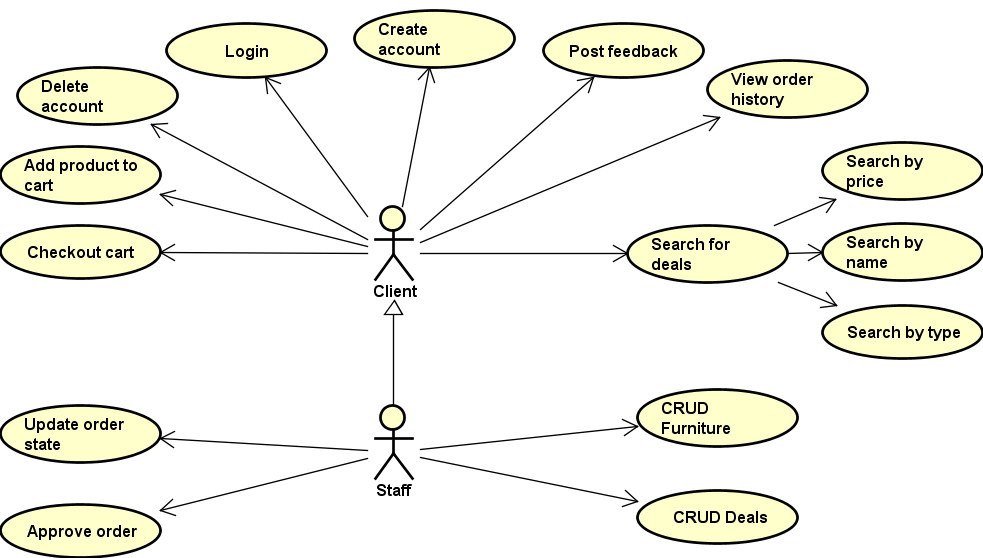
*Level: <one of: summary level, user-goal level, sub-function>*

*Primary actor: <a role name for the actor who initiates the use case>*

*Main success scenario: <the steps of the main success scenario from trigger to goal delivery>*

*Extensions: <alternate scenarios of success or failure>*

*]*



**Use case description:**

**Create account:**

**Use case goal:**

The purpose of create account is to register a new account into the the application, by saving the data into the database.

**Success:**

Primary actor is the client who wants to register. This use case begins when client wants to signup into the app.

1. Client requests for a register form.
2. Server redirects him to one.
3. Client inserts his data.
4. Server verifies inserted data to be valid.
5. If inserted data is correct user is inserted.

**Error:**

If inserted data of an user is invalid when he tries to login, he will be redirected to the same register form, but he will see an error message.

3. System Architectural Design

**3.1 Architectural Pattern Description**

*[Describe briefly the used architectural patterns.]*

**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.

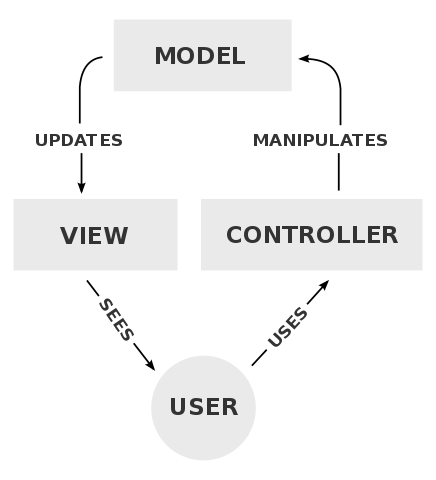
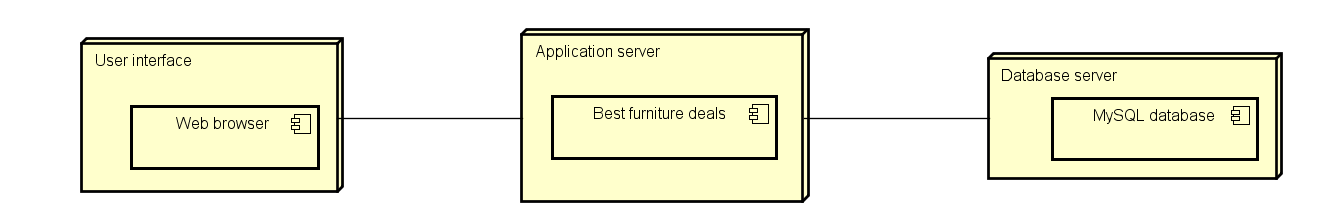


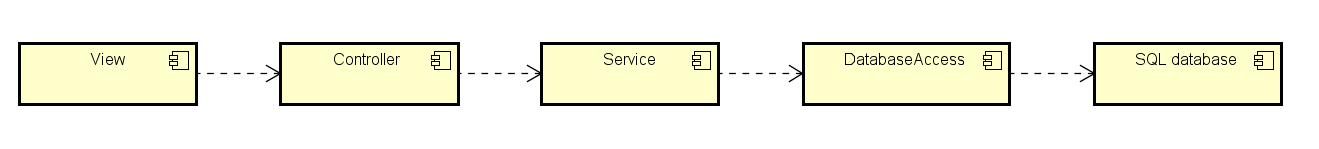
Table Data Gateway is another architectural pattern I used, it consists in using an object which acts as a gateway to a database table. The idea is to separate the responsibility of fetching items from a database from the actual usages of those objects. Users of the gateway are then insulated from changes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

**3.2 Diagrams**

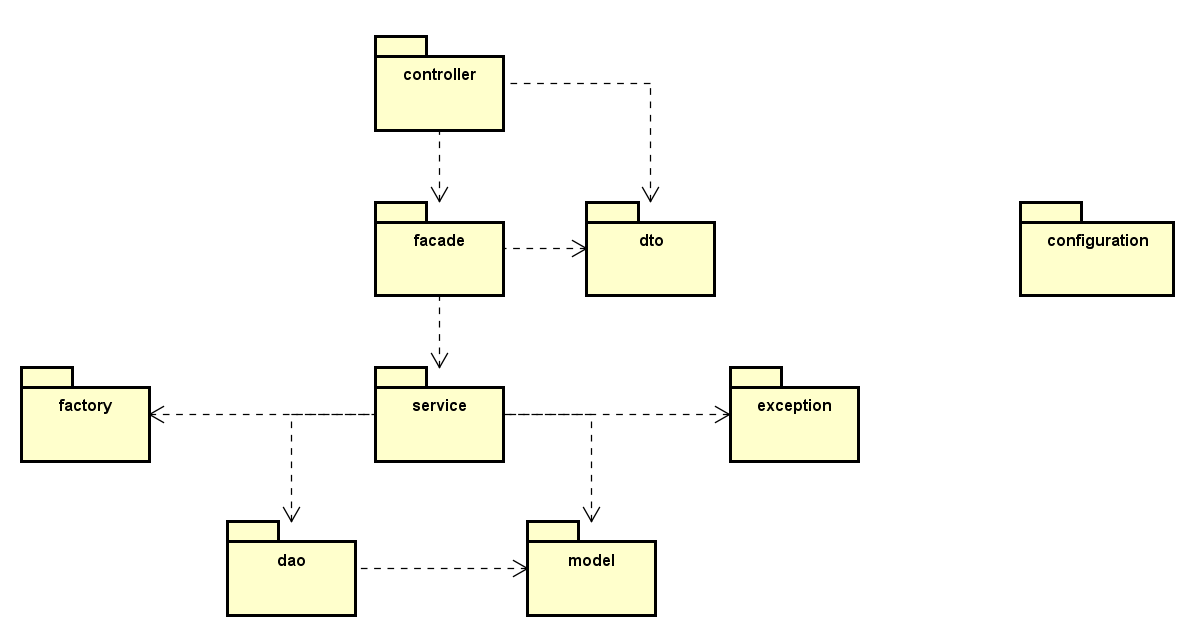
*[Create the system’s conceptual architecture; use architectural patterns and describe how they are applied. Create package, component and deployment diagrams]*

**

**Component :**

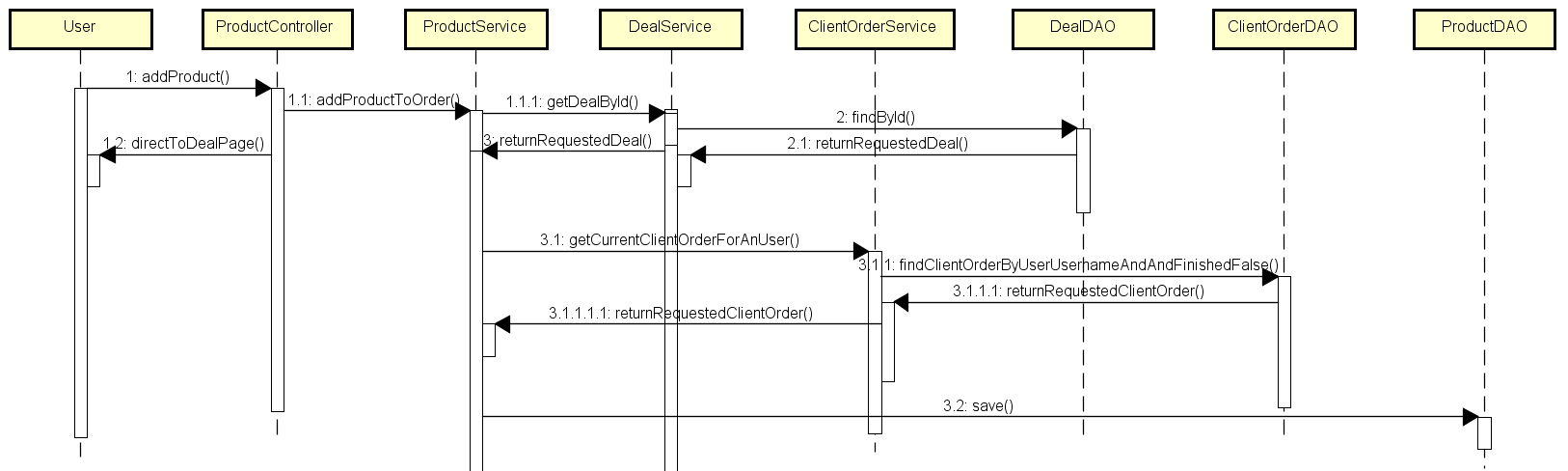
**

**Package diagram :**

****

4. UML Sequence Diagrams

*[Create a sequence diagram for a relevant scenario.]*



5. Class Design

**5.1 Design Patterns Description**

*[Describe briefly the used design patterns.]*

Client server architecture is represented by model view controller architecture.

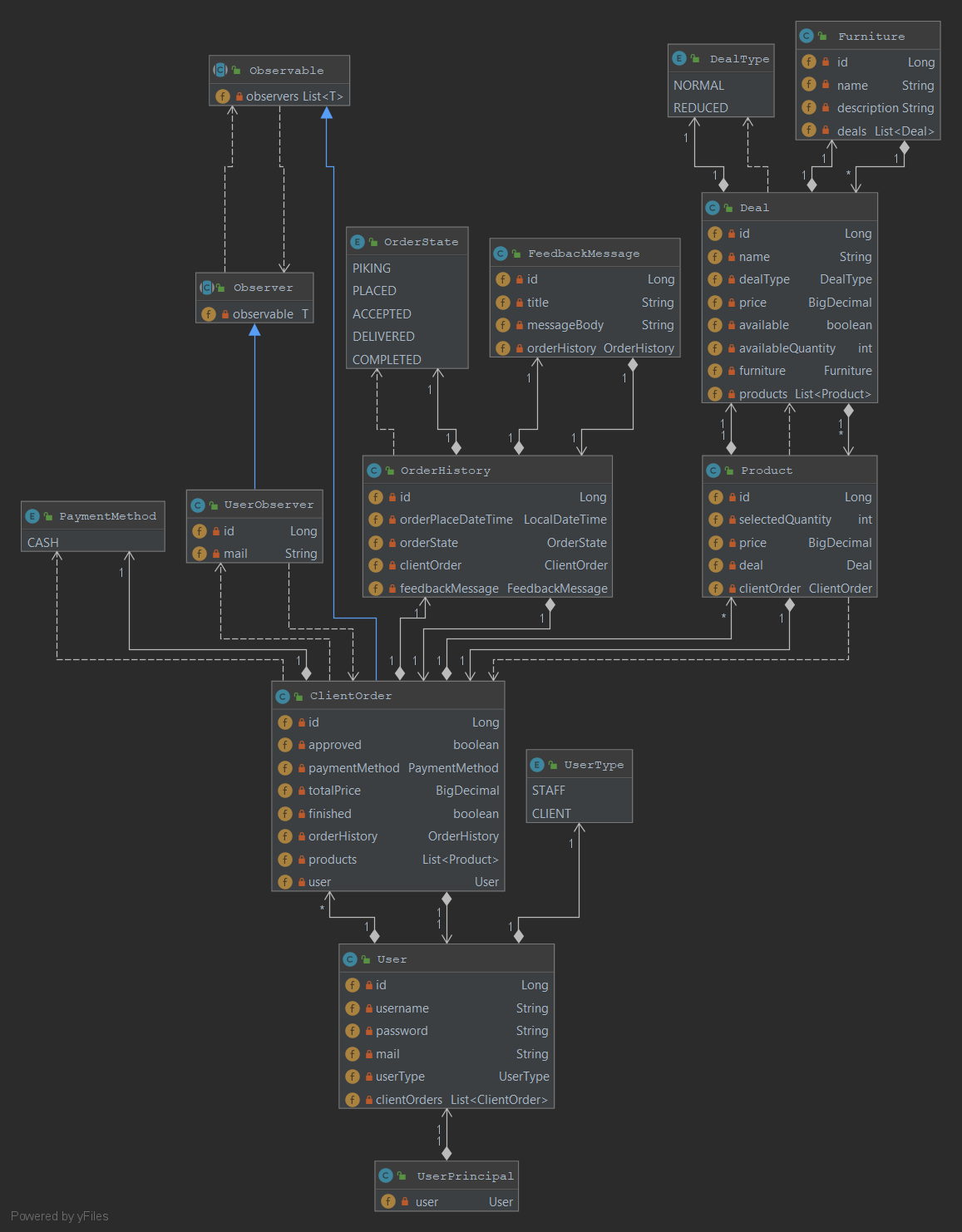
Table data gateway, usage of this pattern is exemplified by DAO classes, like UserDAO, DealDAO etc.

Observer design pattern is exemplified by the model abstract classes Observer and Observable and by ObserverService.

Facade design pattern is exemplified in facade package which is used to encapsulate the logic for converting the dto to a model entity class call wanted service, and convert obtained model if exists in a dto that controller can send to view.

**5.2 UML Class Diagram**

*[Create the UML Class Diagram and highlight and motivate how the design patterns are used.]*



6. Data Model

*[Present the data models used in the system’s implementation.*

Is represented by model package, it consists of classes Product, ClientOrder, Deal, FeedbackMessage, Furniture, OrderHistory, User, where each attribute of class corresponds with a column in the table.

7. System Testing

*[Present the used testing strategies (unit testing, integration testing, validation testing) and testing methods (data-flow, partitioning, boundary analysis, etc.).]*

To test my application, I wrote java tests using an in memory database with preinserted data by a SQL script I wrote and spring boot testing functionalities to load that data and context, I used integration test to test service and dao layers and integration tests for others, I also wrote tests that check corner cases. Another way I used to test my application was with the help of GUI I implemented.

8. Bibliography

<https://martinfowler.com/eaaCatalog/tableDataGateway.html>

<https://creately.com/blog/diagrams/deployment-diagram-tutorial/>

<https://www.oreilly.com/library/view/software-architecture-patterns/9781491971437/ch01.html>