Best Furniture Deals

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 layered architecture, another one consist of using a factory method to build and apply discounts on existing products which change the order quantity and overall price accordingly. 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>*

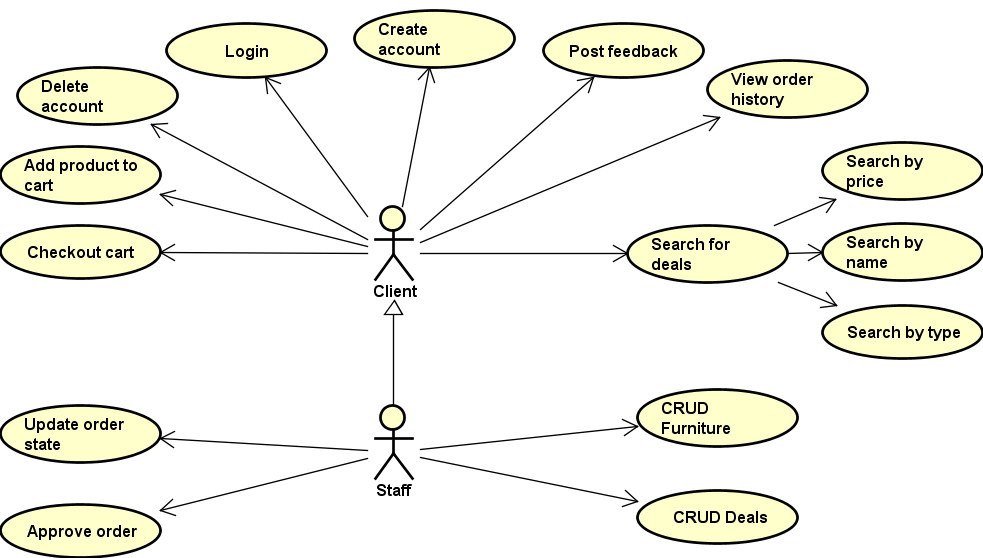
*Level: <one of: summary level, user-goal level, sub-function>*

*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>*

*]*



3. System Architectural Design

**3.1 Architectural Pattern Description**

*[Describe briefly the used architectural patterns.]*

Layered architecture is one of the design I used, it is the most common architecture pattern, also known as the n-tier architecture pattern. Components within the layered architecture pattern are organized into horizontal layers, each layer performing a specific role within application (presentation layer, business logic layer and database access layer) and each layer communicates with de layer after and before it.

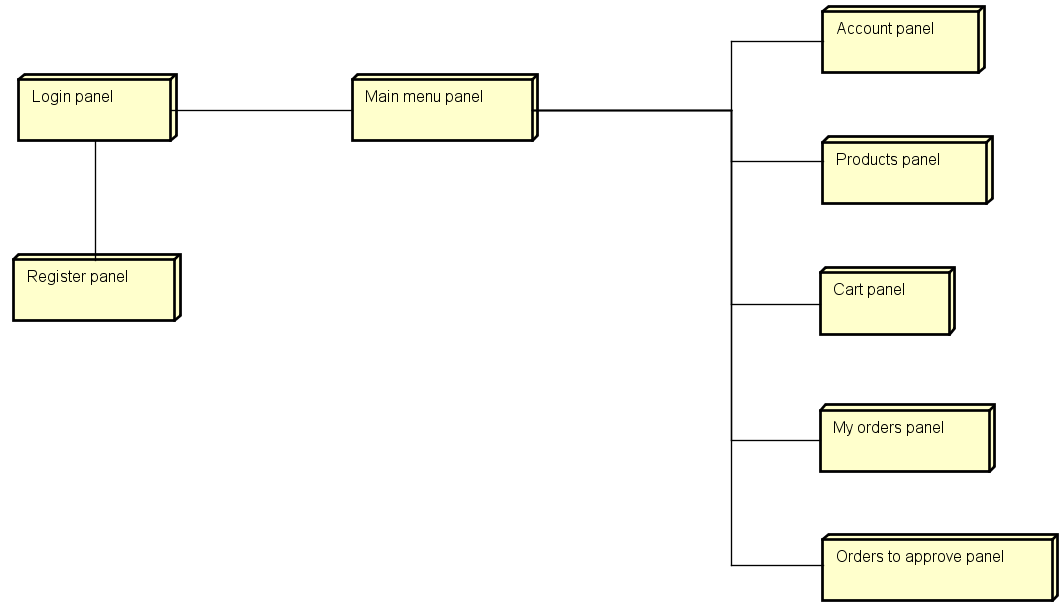
Table Data Gateway is another architectural pattern I used, it consists in using an object which acts as a gateway to a database table. The idea is to separate the responsibility of fetching items from a database from the actual usages of those objects. Users of the gateway are then insulated from changes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

Transaction Script is another design pattern I used, this means that business logic is organized in procedures where each procedure handles a single request from the presentation.

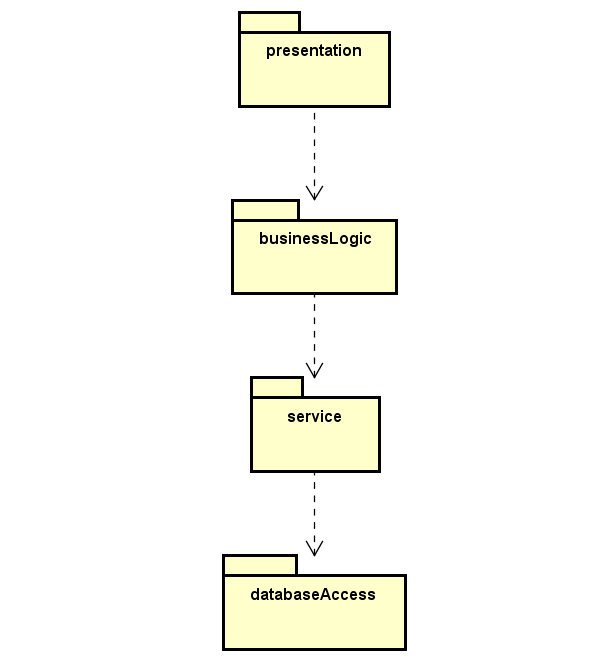
**3.2 Diagrams**

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

**Software deployment diagram**

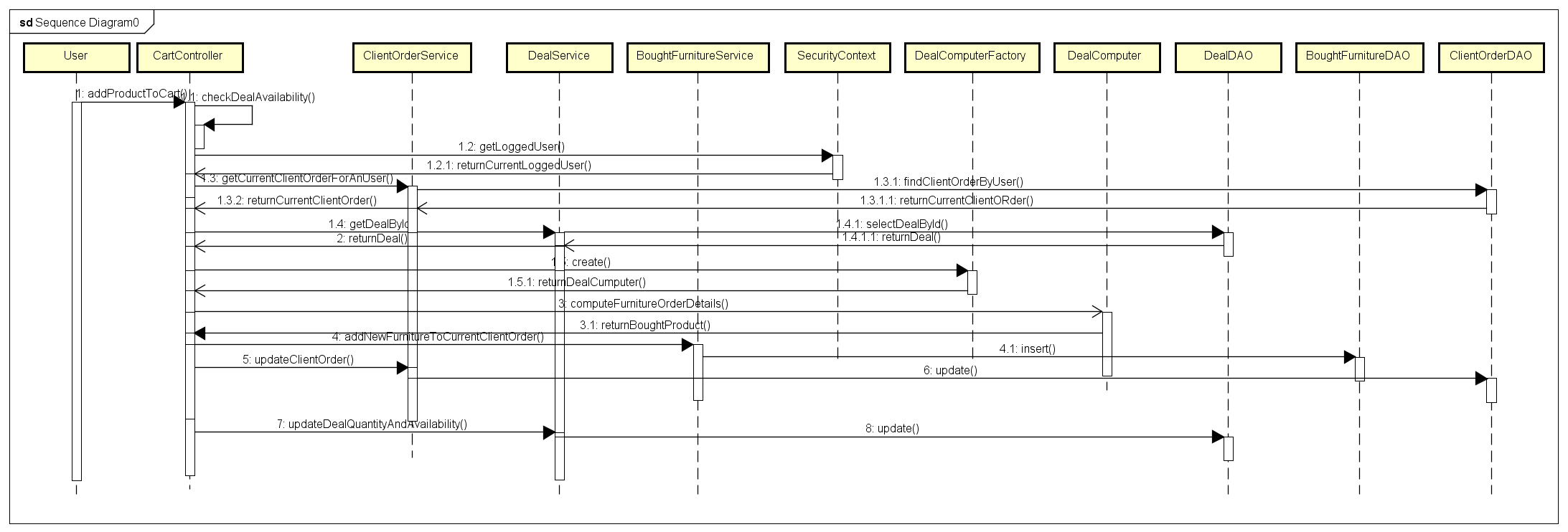


**Package diagram**



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.]*

Layered architecture is represented by the split of the application in layers: database -> service -> businessLogic -> presentation, and each one uses method only from one under it.

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

Transaction Script is exemplified by the use of the classes from business logic, like CartTS, UserTS, which are responsible to wrap all the logic of that specific method and let the user to call only that method from gui.

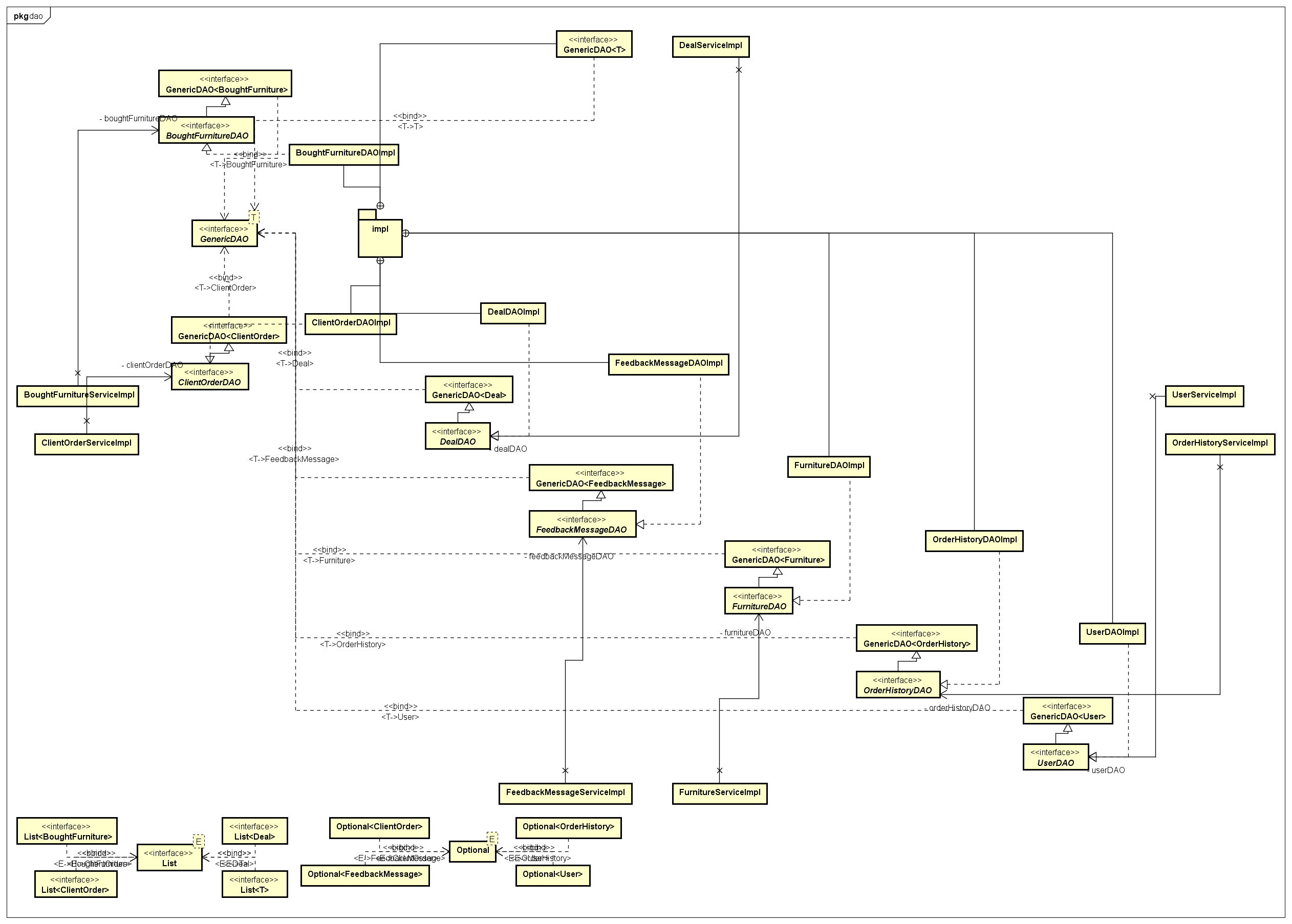
Factory method this is a creational pattern that uses factory methods to deal with the problem of creating objects without having to specify the exact class of the object that will be created.

Singleton is used by without the need of implementing it because of usage of spring framework (spring beans are by default singleton).

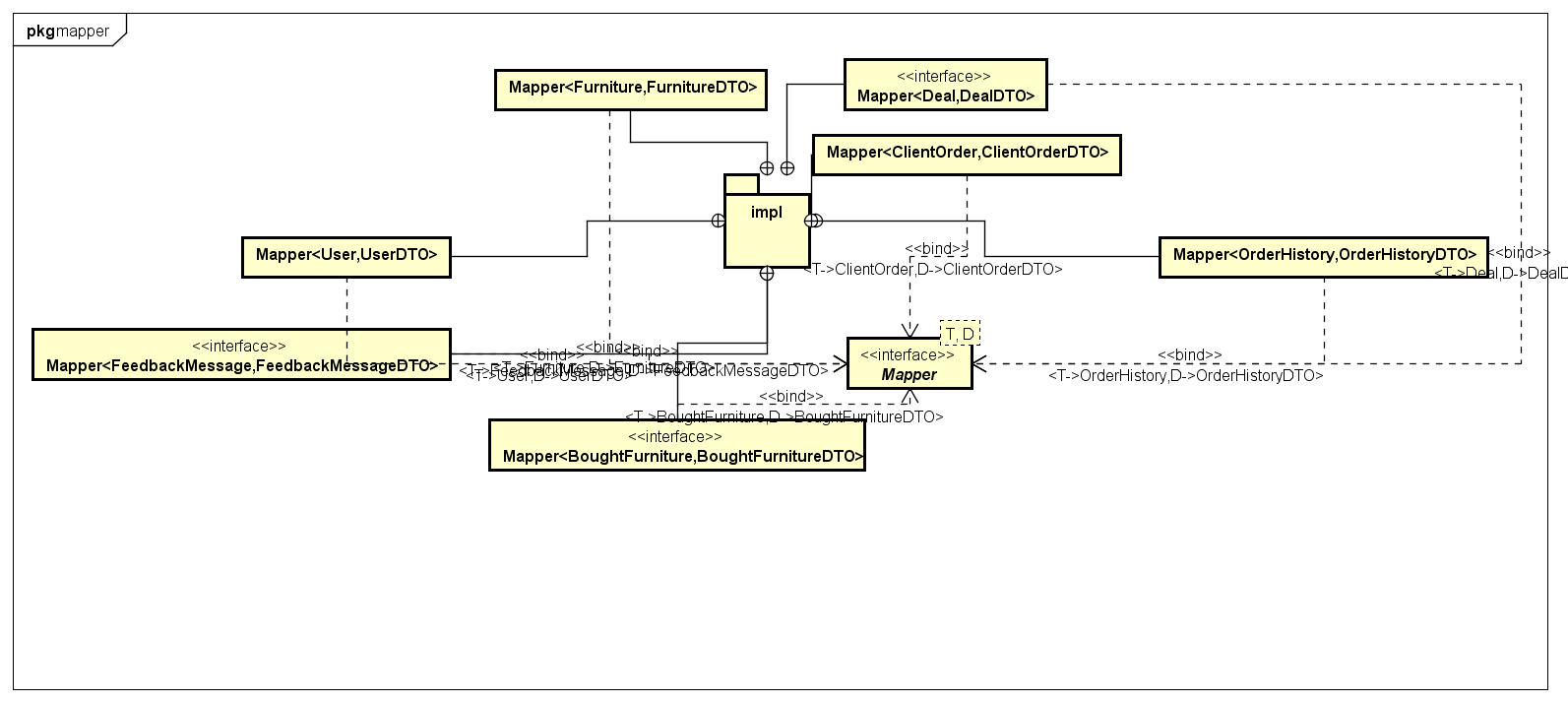
**5.2 UML Class Diagram**

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

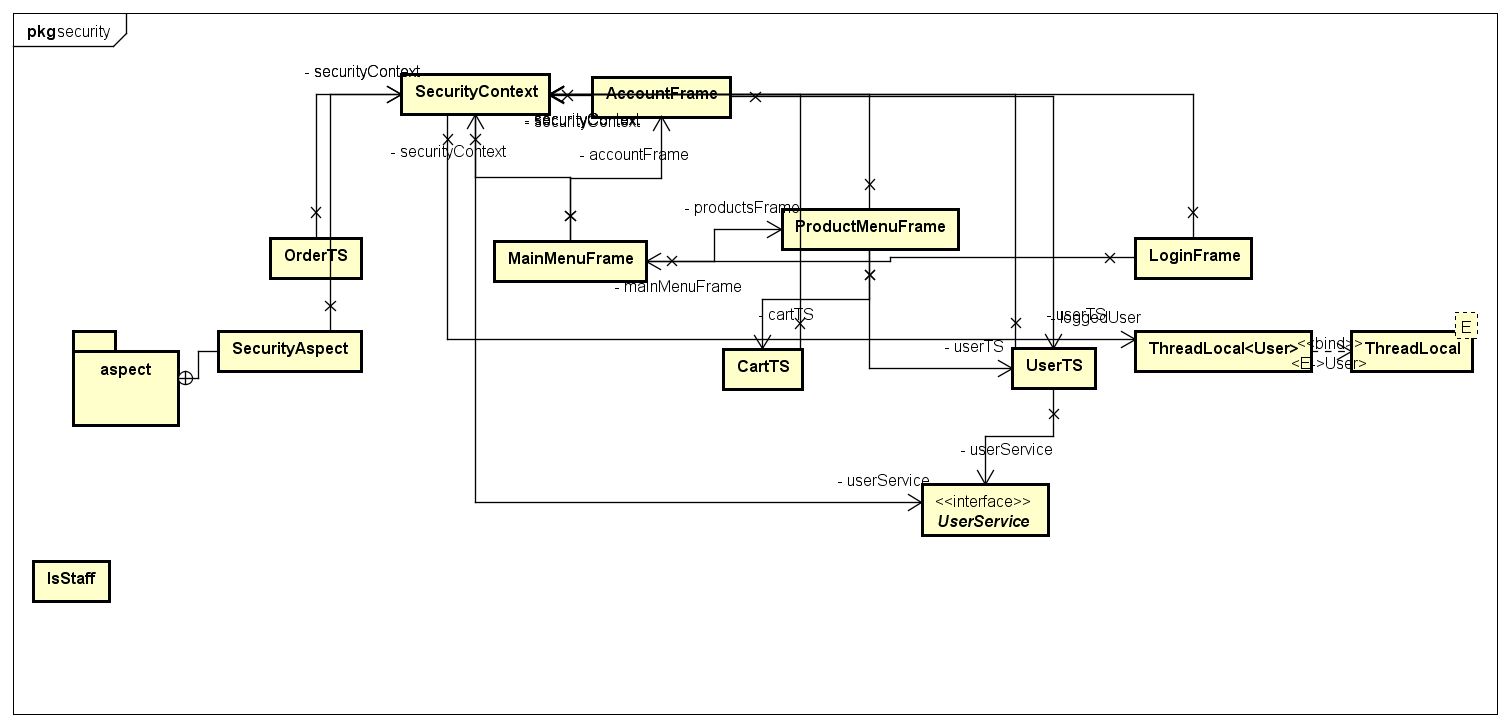
DAO package :



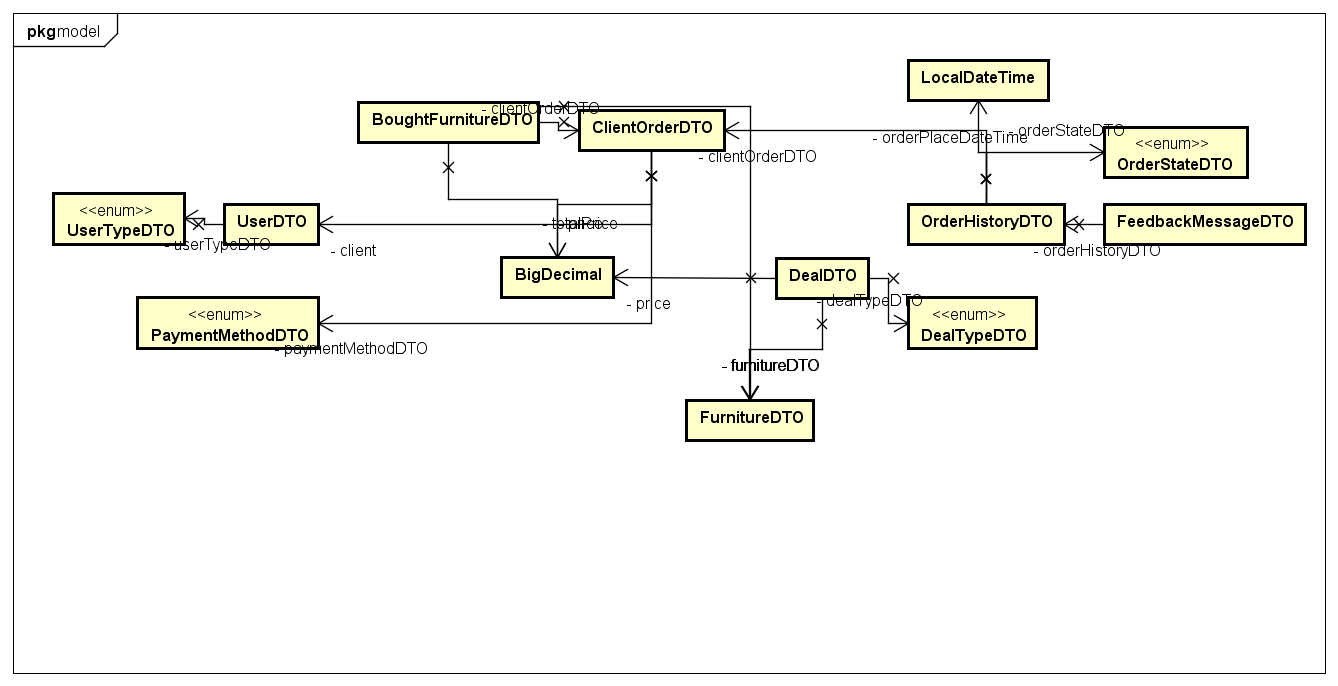
Mapper package :



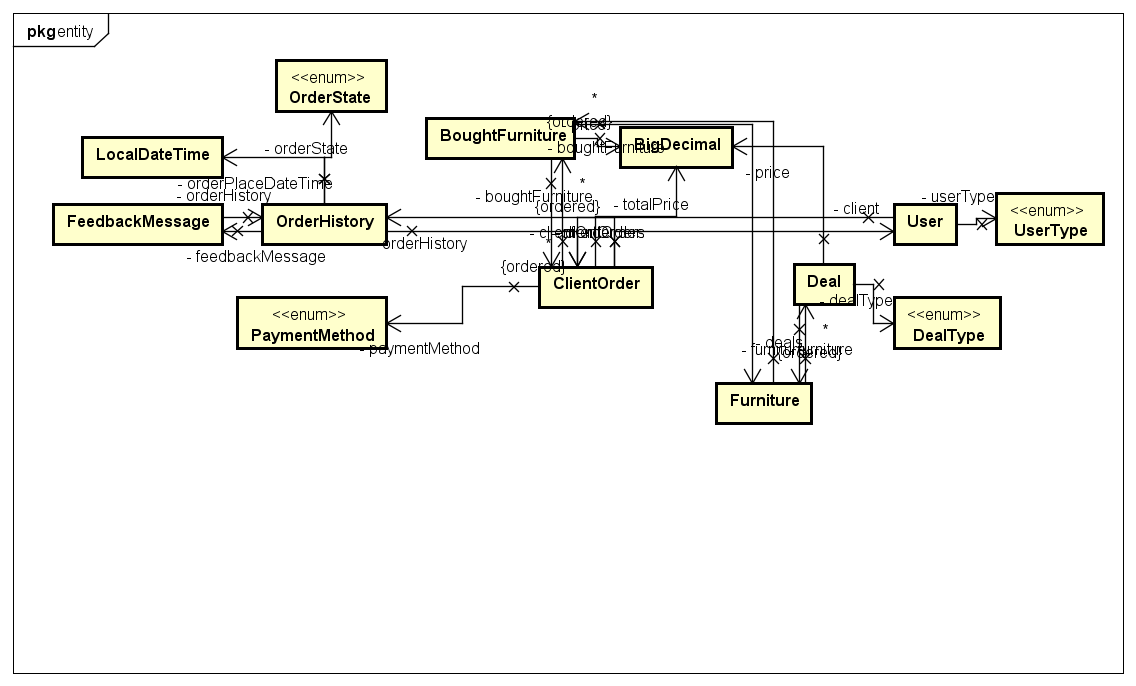
Security package :



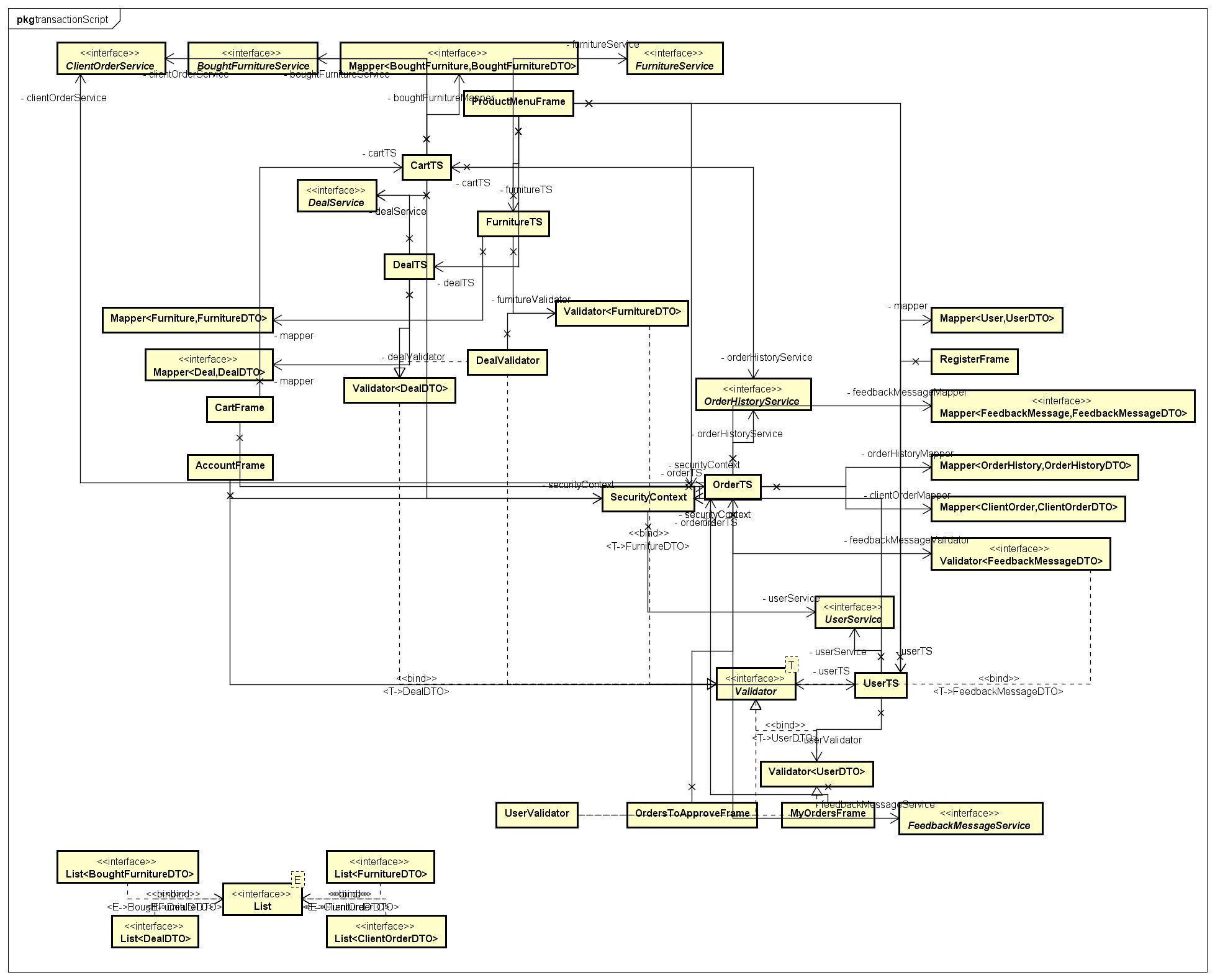
DTO/model package:



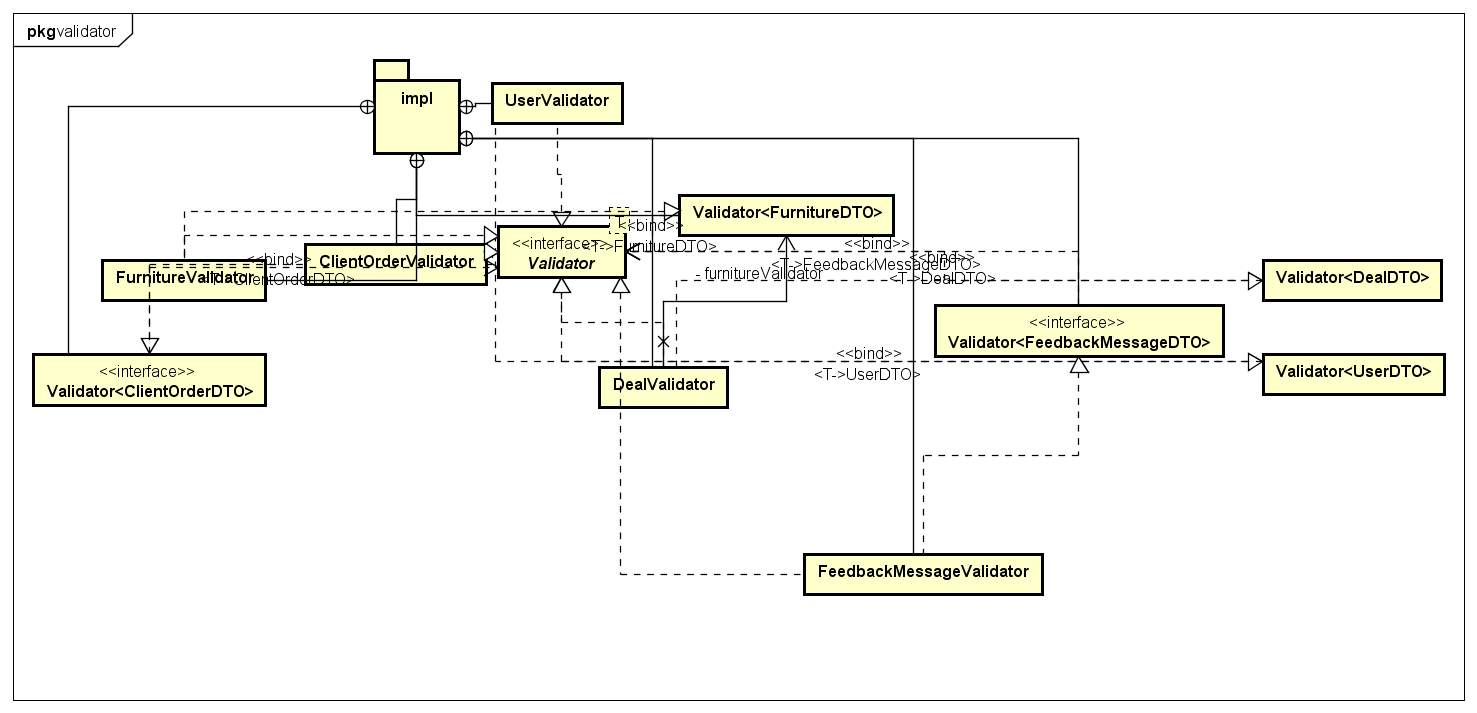
Entity package :



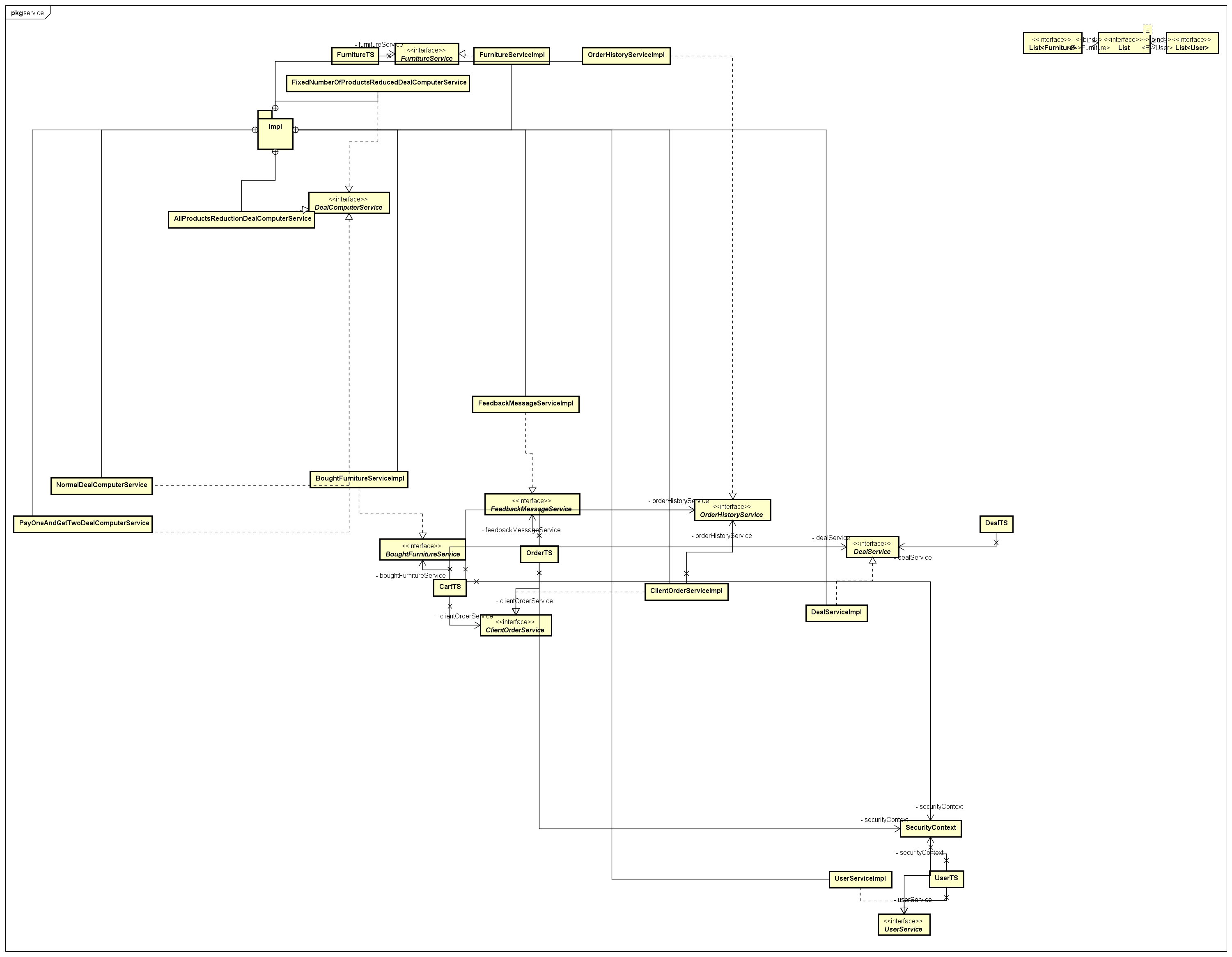
Transaction script package:



Validator package:



Service package:



6. Data Model

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

Is represented by entity subpackage of databaseAccess package, it consists of classes BoughtFurniture, 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>

<https://en.wikipedia.org/wiki/Factory_method_pattern>