Student: Luminita Marghescu
Group: 30234

Table of Contents

1. Requirements Analysis

1.1 Assignment Specification

The C# application implements an order manager of a furniture manufacturer. It has two types of users (a regular user represented by the order manager and an administrator user) which have to provide a username and a password in order to use the application.

1.2 Functional Requirements

The regular user can perform the following operations:

- Add/update/view order information (customer, shipping address, identification number, delivery date, status.).
- Create/update/delete/view product information (title, description, color, size, price, stock etc).
- Add products to order and update order value and stock accordingly.

The administrator user can perform the following operations:

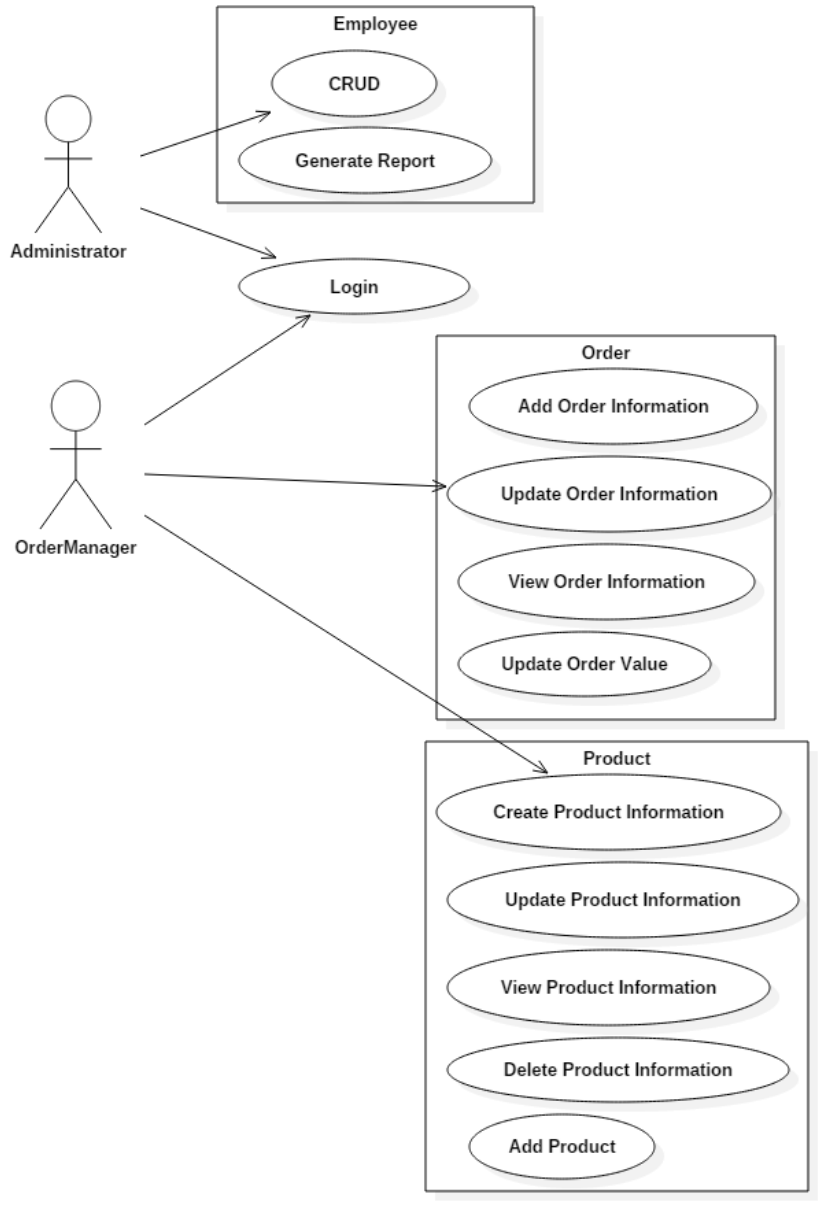
- CRUD on employees' information (Create, Read, Update and Delete operations).
- Generate reports for a particular period containing the activities performed by an employee.

1.3 Non-functional Requirements

- Accessibility
- Availability
- Data back-up
- Efficiency
- Price
- Privacy
- Portability
- Response Time
- Safety
- Security
- Testability

2. Use-Case Model

Use case: <Add Product>
Level: <sub-function>
Primary actor: <Order Manager>
Main success scenario: <Login as Order Manager, Add Product>
Extensions: <Scenario of failure: Login as Administrator>



3. System Architectural Design

3.1 Architectural Pattern Description

Three-Tier architecture is an architectural pattern in which the user interface("Presentation Layer"), functional process logic ("Business Layer"), computer data storage ("Data Access Layer") are developed and maintained as independent modules.

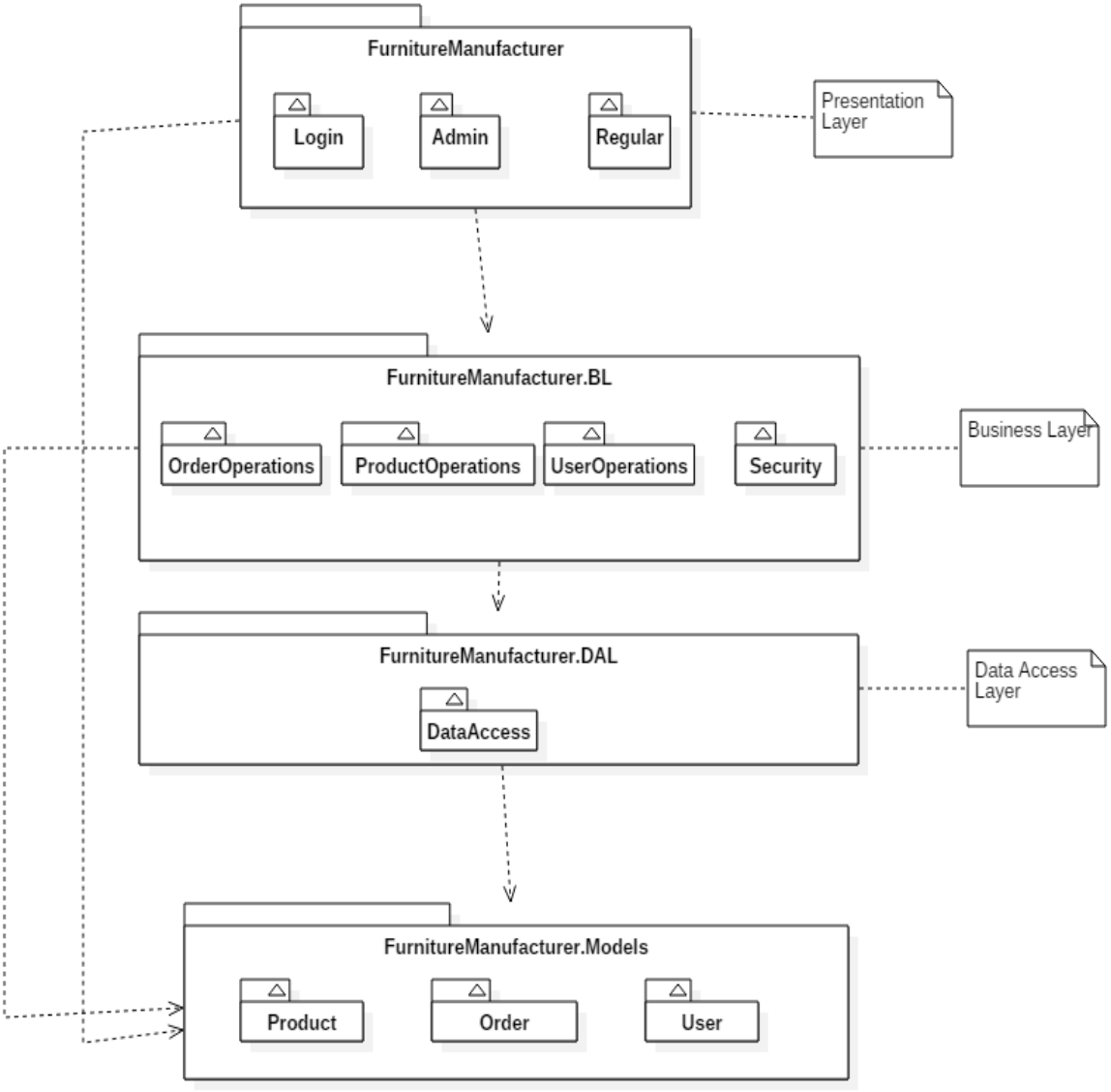
Apart from the usual advantages of modular software with well-defined interfaces, the three-tier architecture is intended to allow any of the three tiers to be upgraded or replaced independently in response to changes in requirements or technology. For example, a change in the presentation tier would only affect the user interface code.

3.2 Diagrams

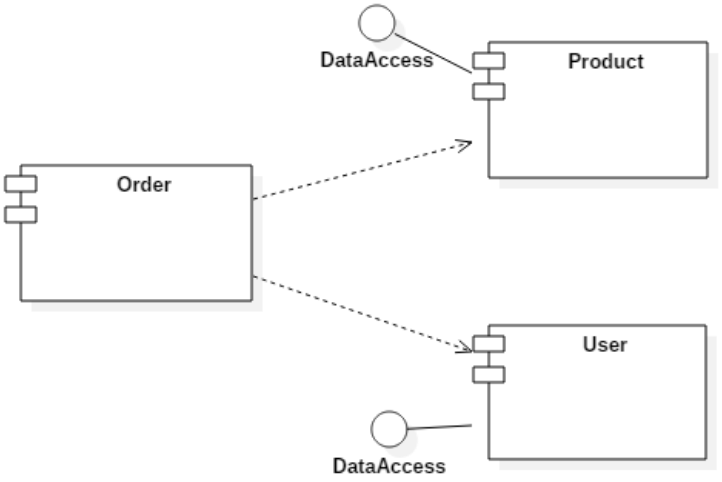
As I said before, I divided the project using Three-Tier Architecture:

- FurnitureManufacturer.Models = it contains 3 classes (Product, Order and User) which correspond to the tables from the DataBase
- FurnitureManufacturer.DAL = it contains only a class (DataAcces.cs) which represents the connection to the DataBase and the SQL Queries, for exemple Add/Update/Delete/Retrieve Product.
- FurnitureManufacturer.BL = it contains 4 classes (3 of them are the operations on Product, User, Order and Security is for password hashing).
- FurnitureManufacturer = it contains all the forms which implements the project functions

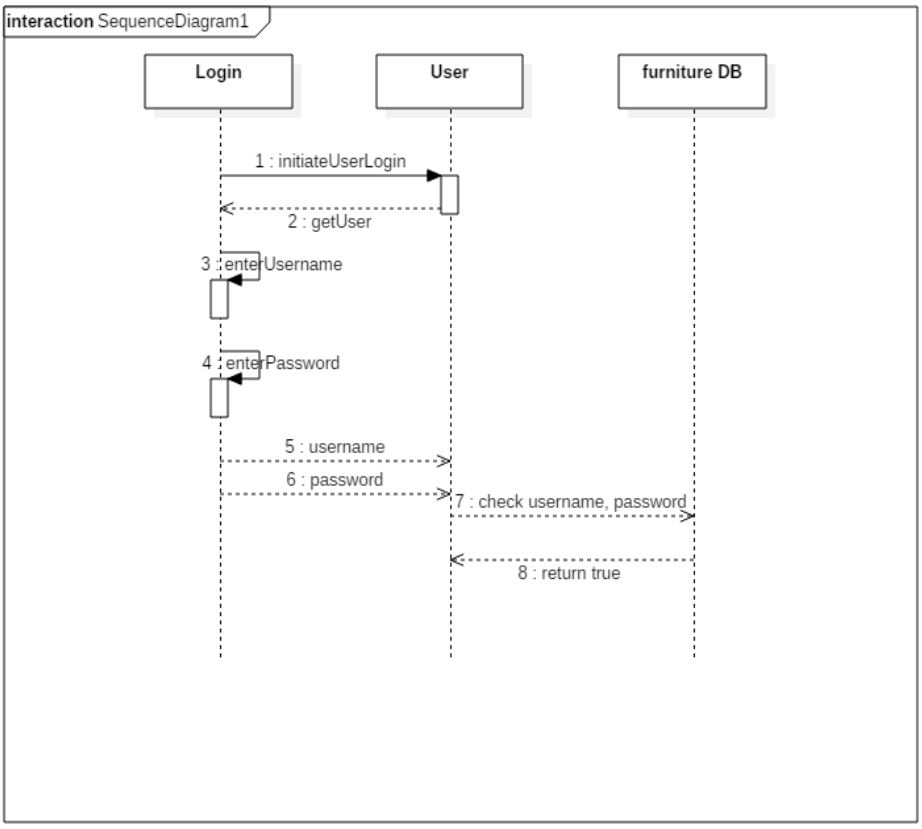
3.2.1 Package Diagram



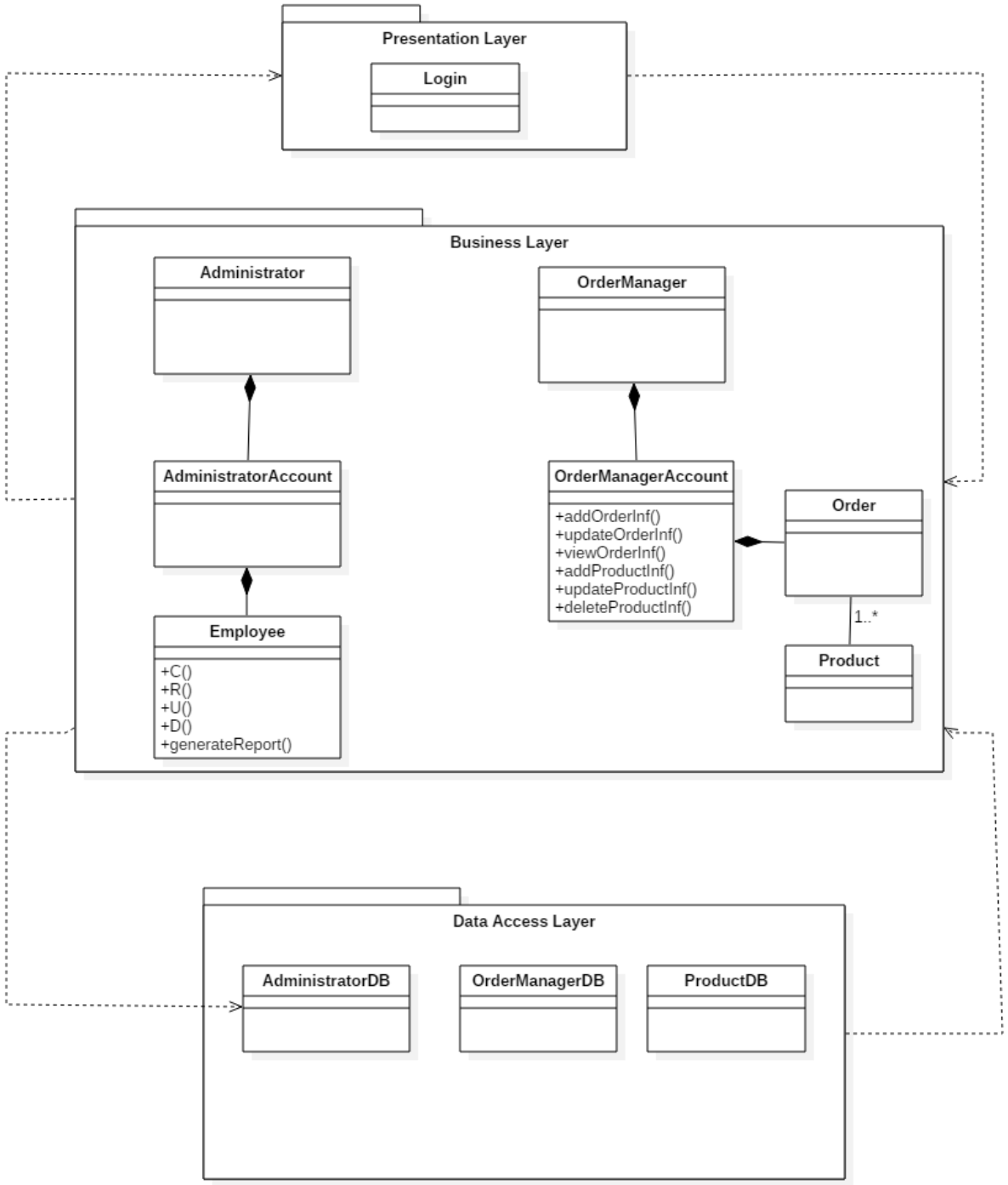
3.2.2 Component Diagram



4. UML Sequence Diagrams



5. Class Design



6. Data Model

I created this DB for using the application:

Database: furniture, Table: user, Purpose: Dumping data

Username	Password	Firstname	Lastname	Admin
bogdan	B06F04511800980D710735D6A9B070190A8FCF5F	Bogdan	Puscasu	1
anca	B06F04511800980D710735D6A9B070190A8FCF5F	anca	iuga	1
zflaviu	83592796BC17705662DC9A750C8B6D0A4FD93396	Flaviu	Zapca	1
flaviu.zapca	83592796BC17705662DC9A750C8B6D0A4FD93396	Flaviu	Zapca	0

Database: furniture, Table: order, Purpose: Dumping data

Customer	ShippingAddress	IdentificationNumber	DeliveryDate	Status
anca	Str. Observatorului	22	2017-03-08 00:00:00	pending
a	aa	1	2017-03-30 18:13:05	aaa

Database: furniture, Table: product, Purpose: Dumping data

Title	Description	Color	Size	Price	Stock
masa	camera de seara	a	1	16	2
a	aa	aaa	1	1	1
scaun	camera de seara	aa	1	1	1

7. System Testing

I used an unit test for the method called "AddProduct" (I tested only one individual unit of source code) to see whether it fits for use. The test returned true, so I implemented it correctly. This is a Dataflow test because the user follows the value of variables and the points at which these values are used.

8. Bibliography

1. https://en.wikipedia.org/wiki/Multitier_architecture