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:

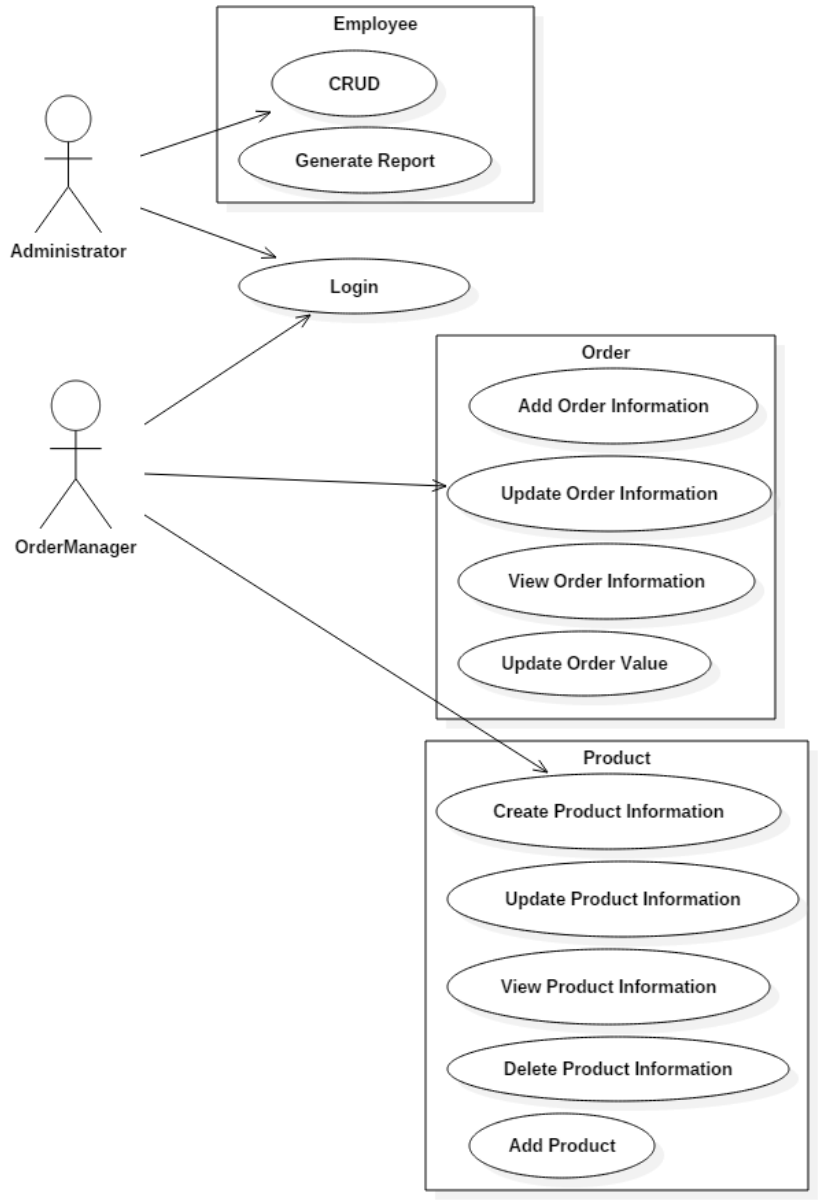
- CRUD on employees' information (Create, Read, Update and Delete operations).
- Generate reports for a particular period containing the activities performed by an employee.

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

This project implements two different architectural patterns: MVP (Model-View-Presenter) and Factory Method for generating the reports, writing the report either in a simple text file or a xml format, depending on a system setting.

MVP is an user-interface architectural pattern engineered to improve the separation of concerns in presentation logic:

- the model is an interface defining the data to be displayed or otherwise acted upon in the user interface.
- The presenter acts upon the model and the view. It retrieves data from repositories (the model), and formats it for display in the view.
- The view is a passive interface that displays data (the model) and routes user commands (events) to the presenter to act upon that data.

Factory method pattern 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. This is done by creating objects by calling a factory method—either specified in an interface and implemented by child classes, or implemented in a base class and optionally overridden by derived classes—rather than by calling a constructor.

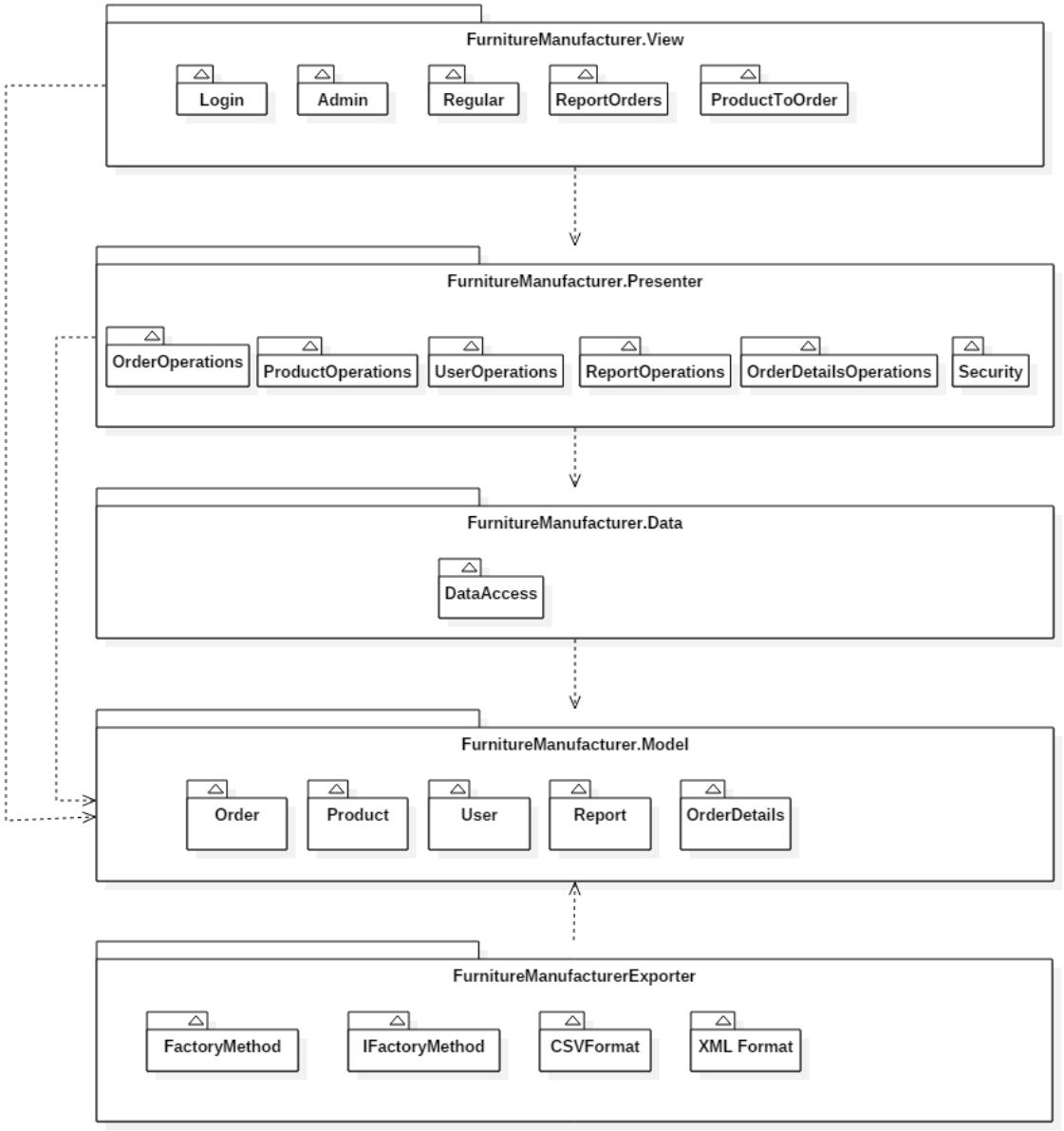
.

3.2 Diagrams

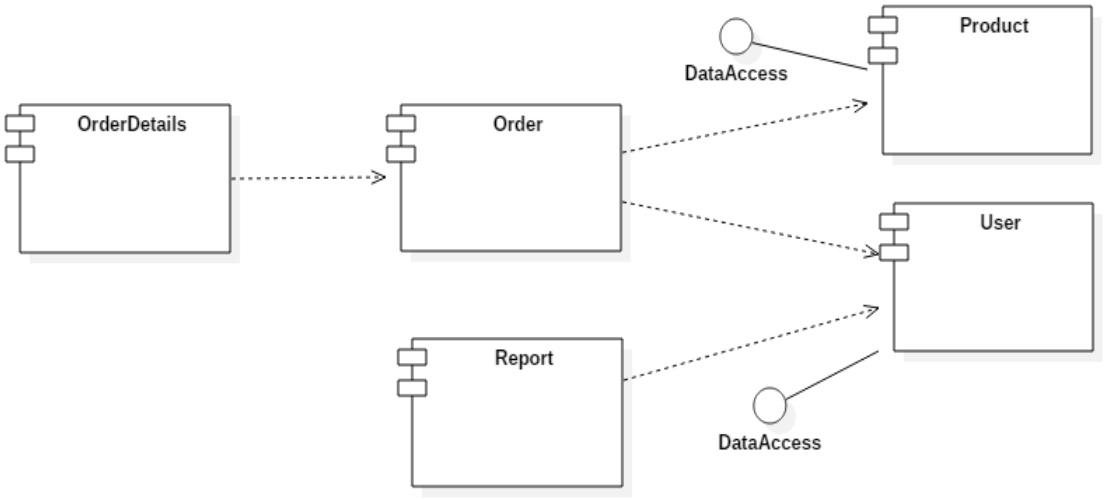
As I said before, I divided the project using Model-View-Presenter:

- FurnitureManufacturer.Model = it contains 5 classes (Product, Order, User, Report and OrderDetails) which correspond to the tables from the DataBase
- FurnitureManufacturer.View = it displays the model and routes all the commands to the presenter
- FurnitureManufacturer.Presenter = it contains 6 classes (5 of them are the operations on Product, User, Order, Report, OrderDetails and Security is for password hashing).
- FurnitureManufacturer.DataAccess = 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.Exporter = it contains the Factory Method classes
- FurnitureManufacturer = it contains the main function which runs the application

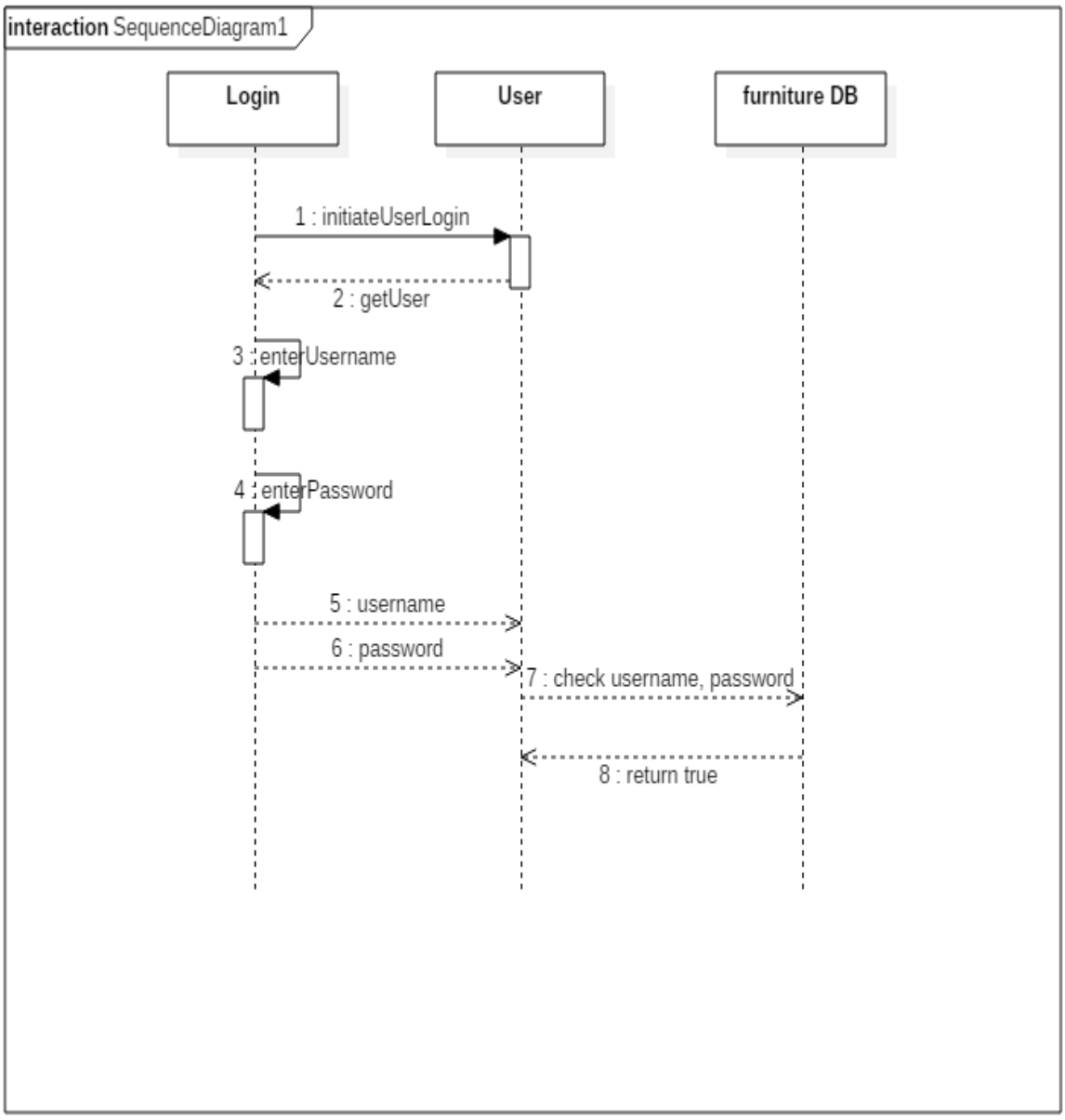
3.2.1 Package Diagram



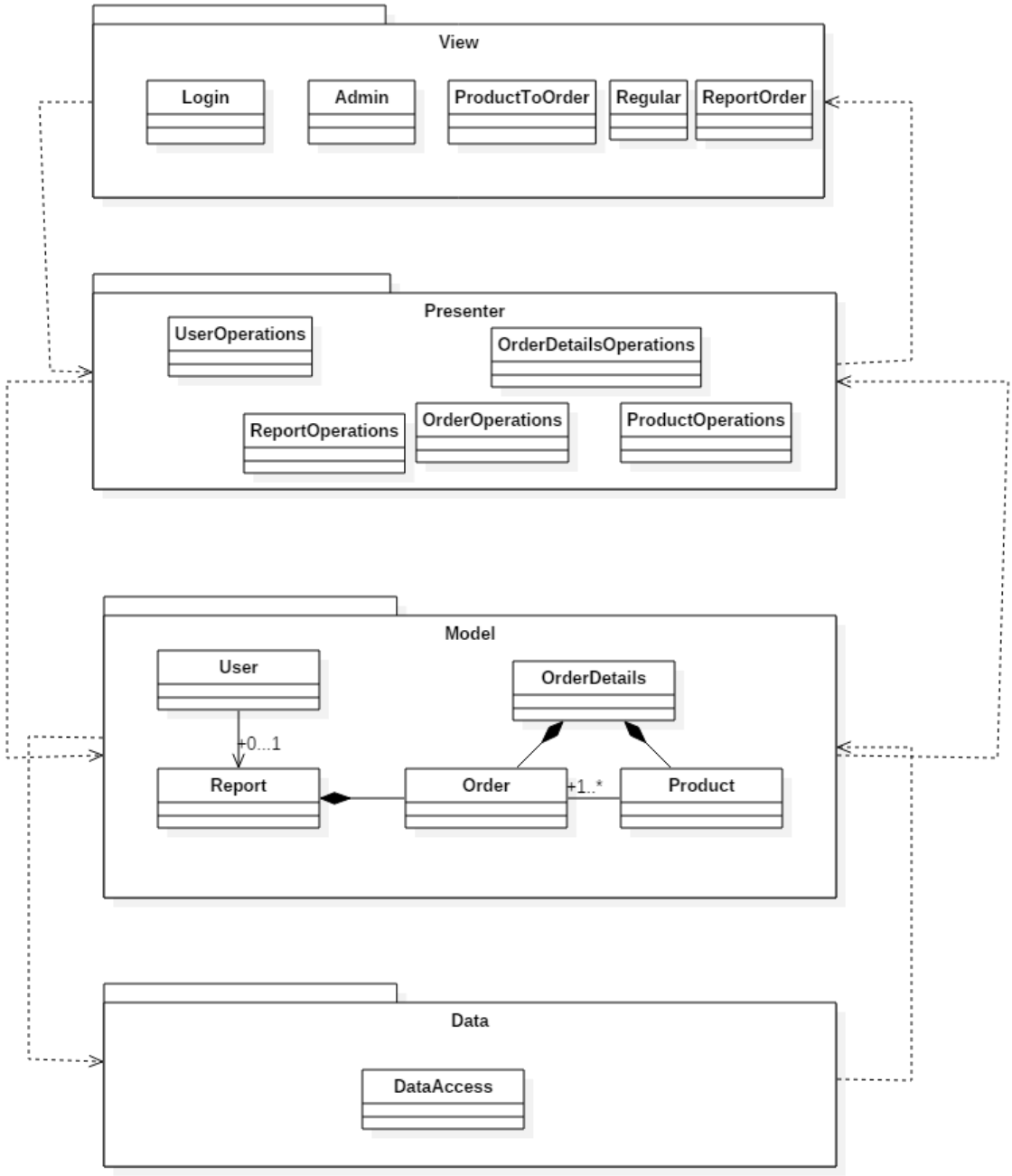
3.2.2 Component Diagram



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

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

idReport	Customer	addOp	updateOp	viewOp	date
1	cris	1	0	0	2017-04-27
3	Miiha	0	1	0	2017-05-03

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

idOrder	idProduct	PriceUnit	Quantity	Price
7	1	20	1	20
8	2	8	1	8

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.