Student: Bánhidi Zoltán

**Group:** 30234

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

The assignment consists of designing and implementing a program which manages a furniture selling company. It must store the related data (product information, orders, users and logging data) in a database. The application should have two types of users:

* regular users who can manage products and orders
* administrators who can manage the users and can review their activity

Both users must log in using a username and a secured password.

# Functional Requirements

The regular users can do the following operations:

* CRUD on products
* apply filters for products
* CRUD on their orders
* CRUD on their orders shopping cart
* calculate their orders total price
* manage their account (change username and/or password)

The administrators could do the following operations:

* CRUD on users data (admins and regular users)
* review the regular users activity
* filter the regular users activity
* delete regular users activity
* manage their account (change username and/or password)

# Non-functional Requirements

* *Availability*: the system will be available as long as is connected to the database
* *Usability*: the application is easy to use, with user-friendly graphical interface
* *Security*: the account of a user can be accessed only by providing the username and the password

2. Use-Case Model

Use case: add new product

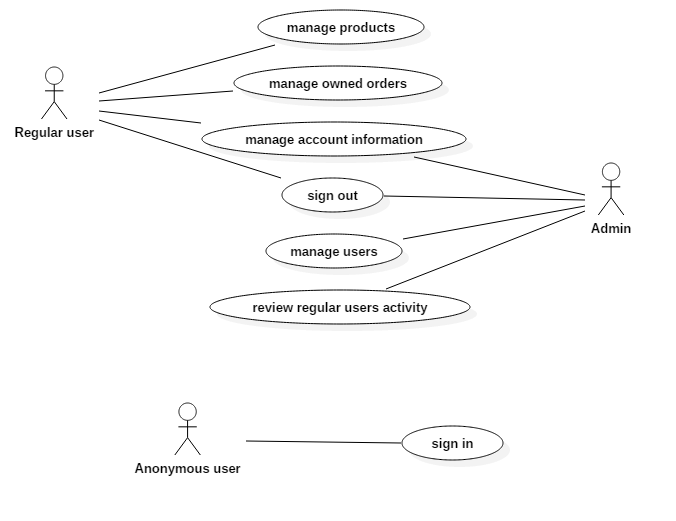
Level: user-goal level

Primary actor: regular user

Main success scenario:

* **The regular user logs in**
* **Navigates to the “Products” tab**
* **Clicks the “Add new product” button**
* **Introduces valid product information**
* **Clicks the “Add new product” button**

Extensions: The addition fails if a product with similar name already exists or invalid data is given.



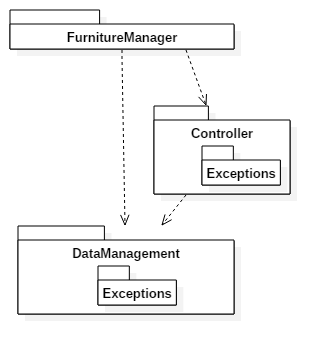
3. System Architectural Design

**3.1 Architectural Pattern Description**

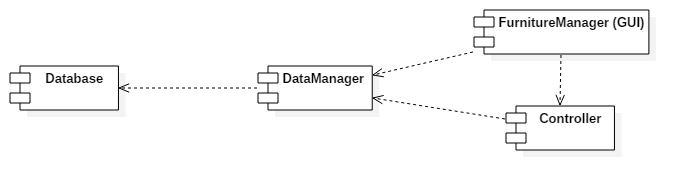
The system is designed using the Layers Architectural Pattern. Layered architecture focuses on grouping related functionality within an application into distinct layers that are stacked vertically on top of each other. Functionality within each layer is related by a common role or responsibility. Upper layers such as the user interface layer send commands to lower layers, such as the business and data layers. However the lower layers can not access the functionalities of the upper layers. Every layer (except the first) is an abstraction of the under layer.

**3.2 Diagrams**

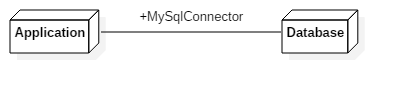
Package diagram:



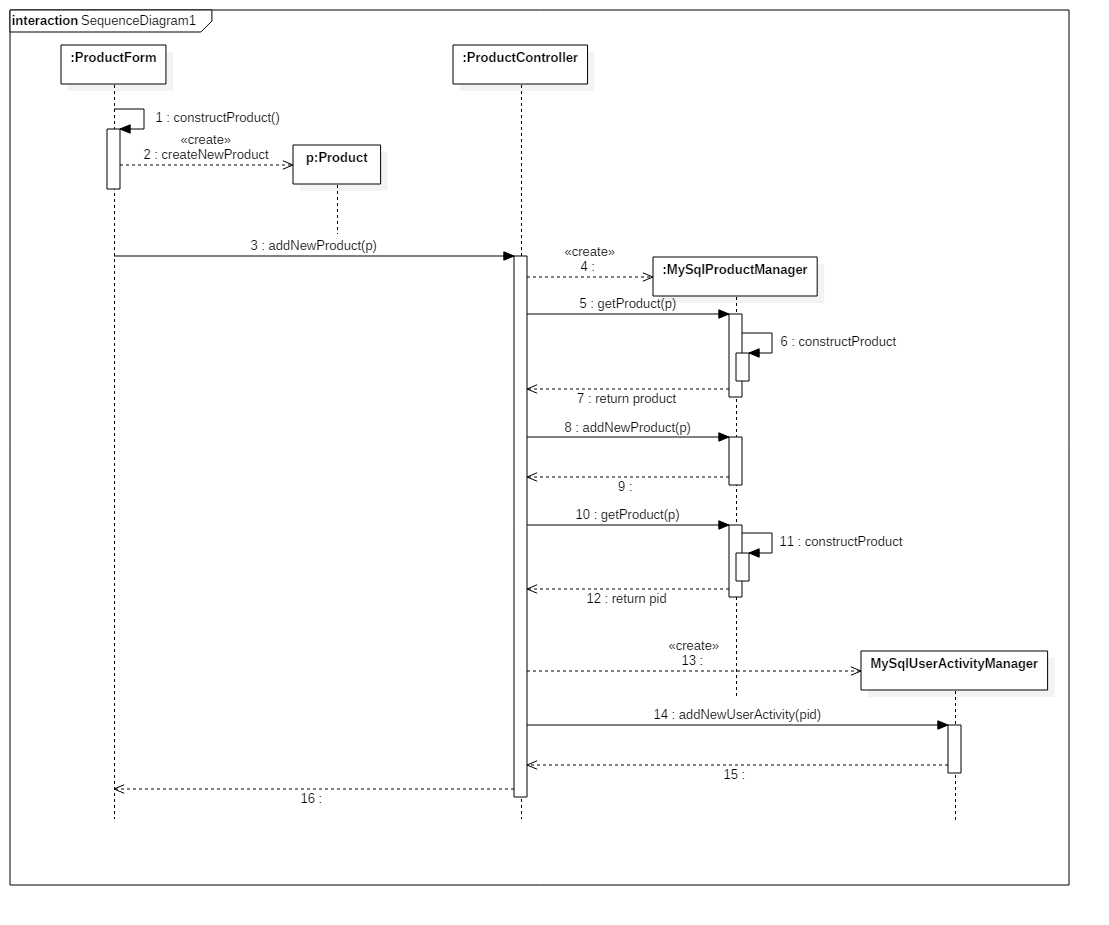
Component diagram:



Deployment diagram:



4. UML Sequence Diagrams



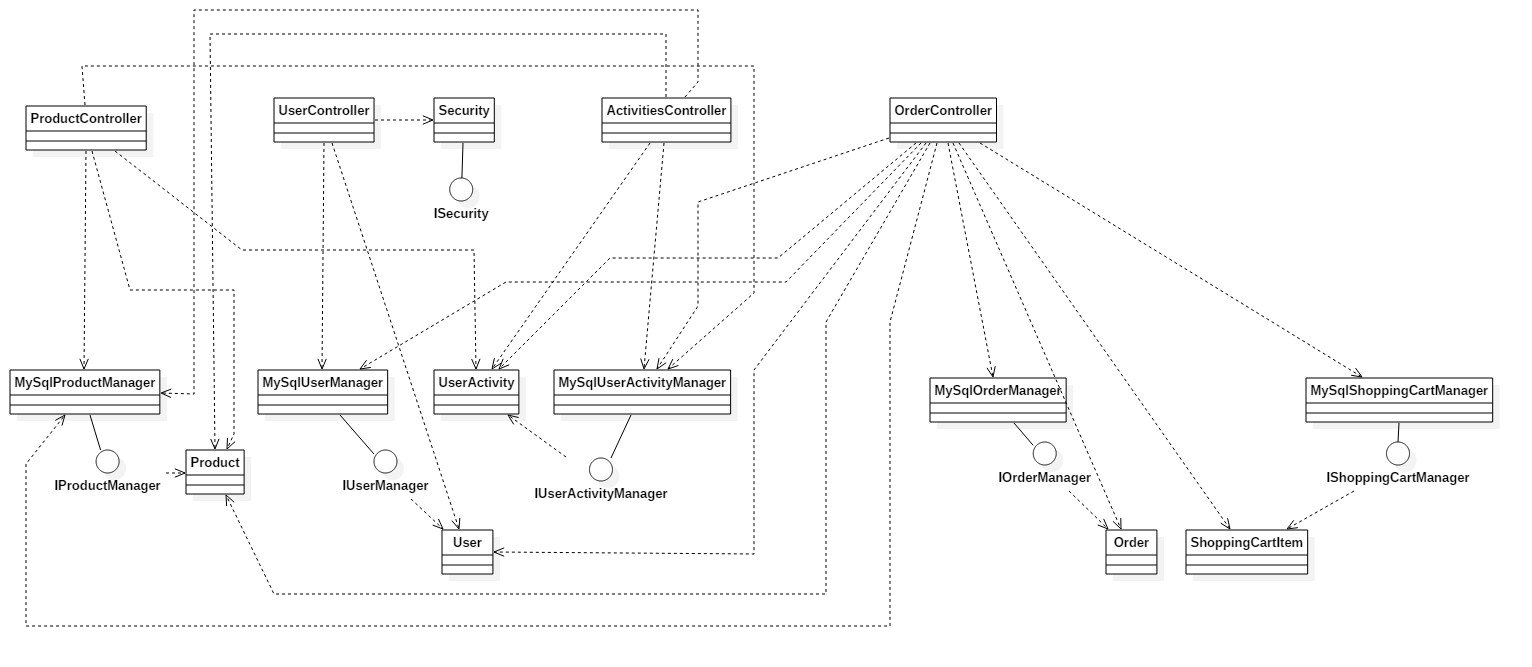
5. Class Design

**5.1 Design Patterns Description**

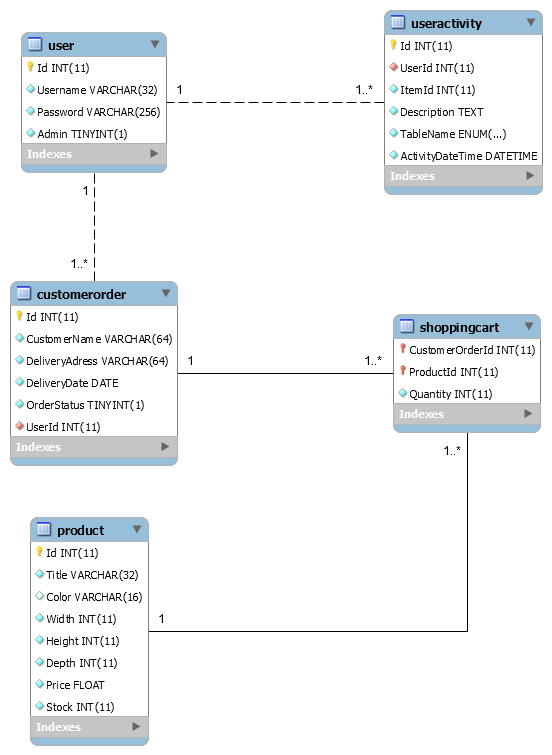
I used the Model-View-Controller design pattern to build the application. The model is constructed by the *DataManagement* package, the view by the *FurnitureManager* package and the controller consists of the *Controller* package. This patterns basic idea is to separate the data representation (model), the business logic (controller) and the presentation (view) from each other. In this manner every modification is easier, and the presentation component can be changed if it is necessary.

**5.2 UML Class Diagram**

For simplicity, in the class diagram are not represented the GUI components.



6. Data Model



7. System Testing

Unit test were written for database manager classes, and was used with a clone Database.

8. Bibliography

* Mihaela Dînșoreanu, Software Design