Computer Parts Online Shop

Analysis and Design Document

Student: Cotet Eusebio Calin

**Group:30233**

Revision History

|  |  |  |  |
| --- | --- | --- | --- |
| **Date** | **Version** | **Description** | **Author** |
| <05/04/2017> | <1.0> |  | Cotet Eusebio Calin |
| <03/05/2017> |  |  | Cotet Eusebio Calin |
|  |  |  |  |
|  |  |  |  |

Table of Contents

I. Project Specification 4

II. Elaboration – Iteration 1.1 4

1. Domain Model 4

2. Architectural Design 4

2.1 Conceptual Architecture 4

2.2 Package Design 4

2.3 Component and Deployment Diagrams 4

III. Elaboration – Iteration 1.2 4

1. Design Model 4

1.1 Dynamic Behavior 4

1.2 Class Design 4

2. Data Model 4

3. Unit Testing 4

IV. Elaboration – Iteration 2 4

1. Architectural Design Refinement 4

2. Design Model Refinement 4

V. Construction and Transition 5

1. System Testing 5

2. Future improvements 5

VI. Bibliography 5

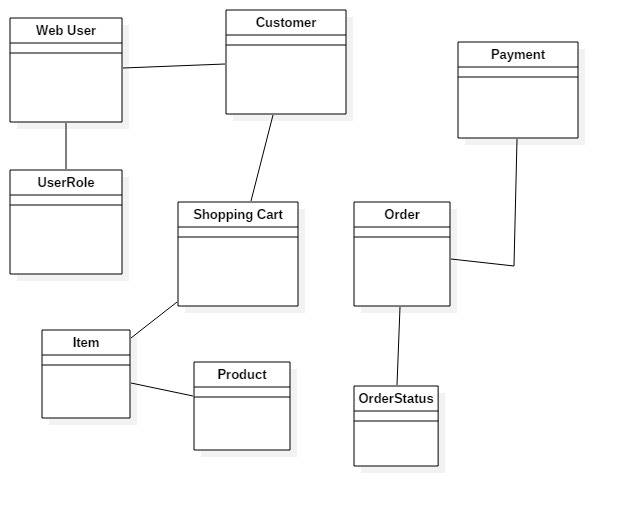
# Project Specification

# Elaboration – Iteration 1.1

The application represents an online computer parts shop. The application is a way for the guests to visualize, create an account, buy computer parts, review products and see reviews from other clients and also a way for the administrator to manage the online shop by modifying stocks and block users if needed.

# Domain Model

The domain model will be structured as shown in the conceptual class diagram given below:



# Architectural Design

## Conceptual Architecture

The application will be MVC pattern since I will use Spring web MVC in my application.

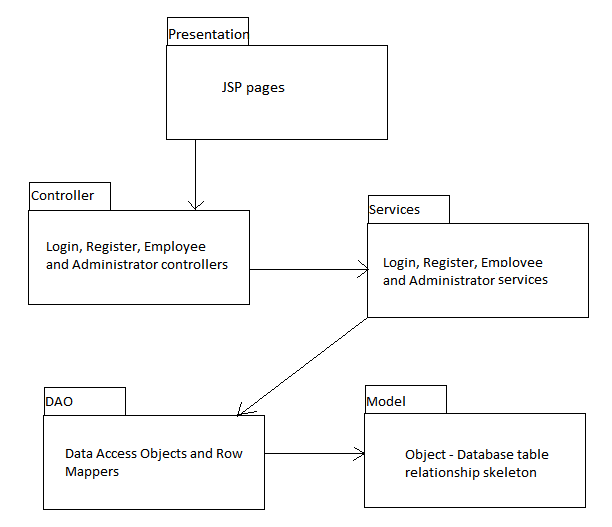
The Layers pattern will structure the application in four major modules:

* Presentation Layer – this layer will include everything that can be seen in a browser by the user
* Business Layer – it will include all the functionality the application needs (i.e. add to cart and place order actions)
* DAO Layer – this layer will be responsible with the database access and CRUD operations requested by the business layer
* DB Layer – this layer will consist of the database itself

The MVC pattern will have a similar structure. The View and Controller components will be similar to the Presentation and Controller layers described above.

The Model layer will represent a skeleton for the structure of the application. A more precise example of the concept of the model layer would be a package that contains the classes the define the structure of a table from a database (a class that contains class attributes representing the table columns from the database, constructors and getter and setter methods for those attributes).

## Package Design



## 

## Component and Deployment Diagrams

Deployment diagram

# C:\Users\catal\AppData\Local\Microsoft\Windows\INetCache\Content.Word\project deployment diagram.png

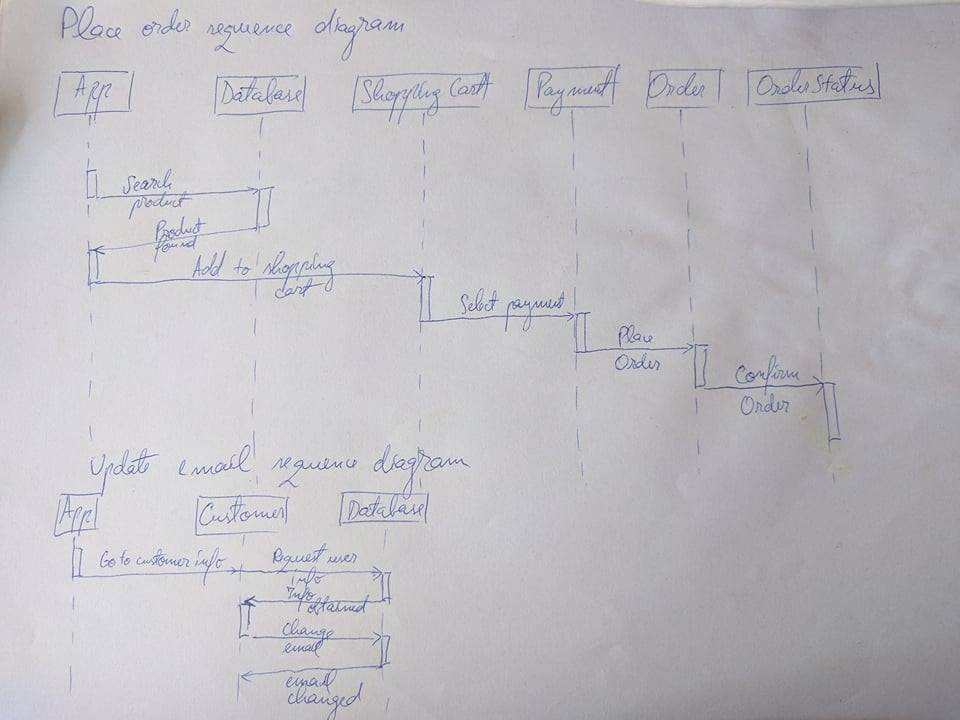
Component Diagram



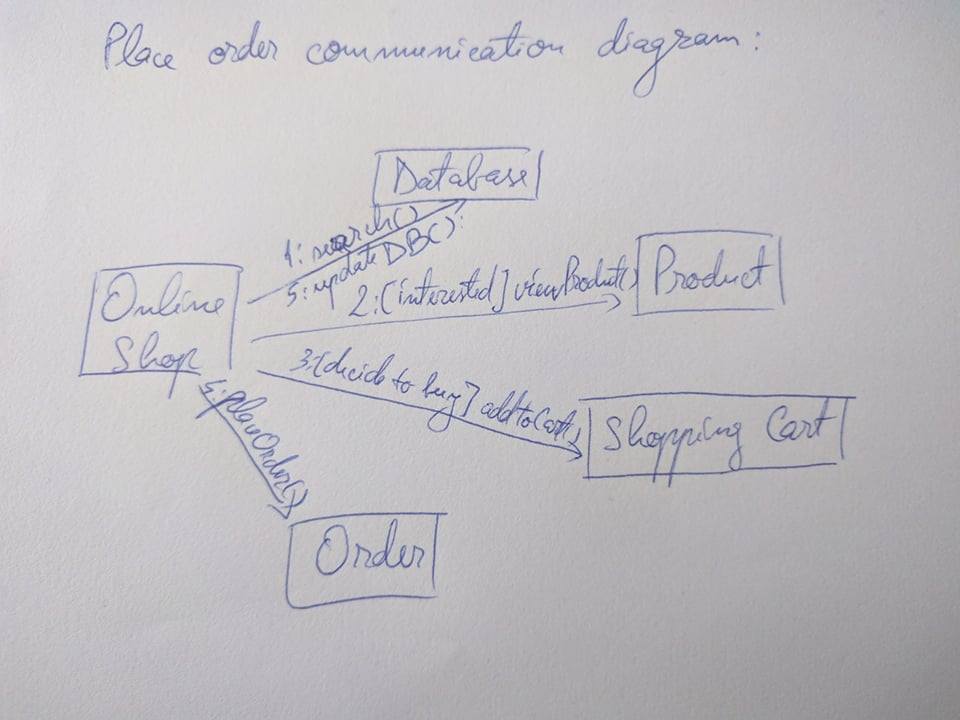
# Elaboration – Iteration 1.2

# Design Model

## Dynamic Behavior



## 

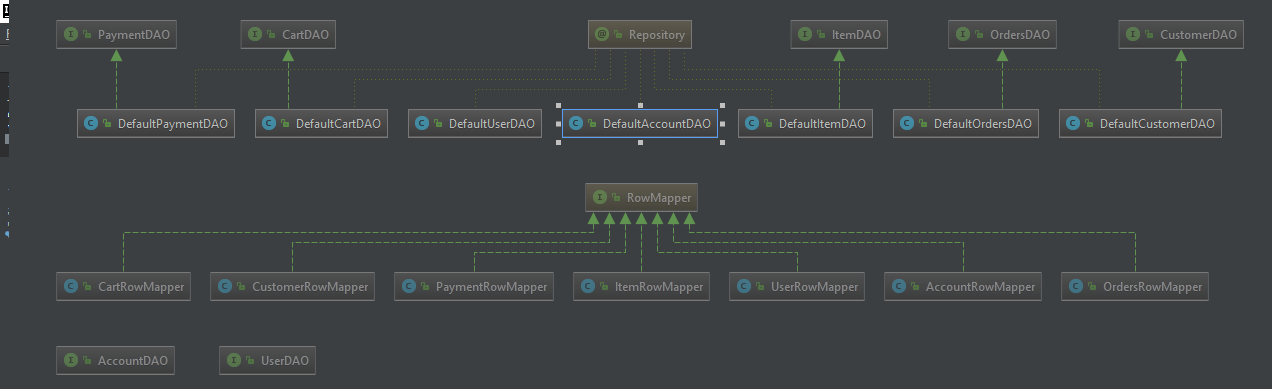
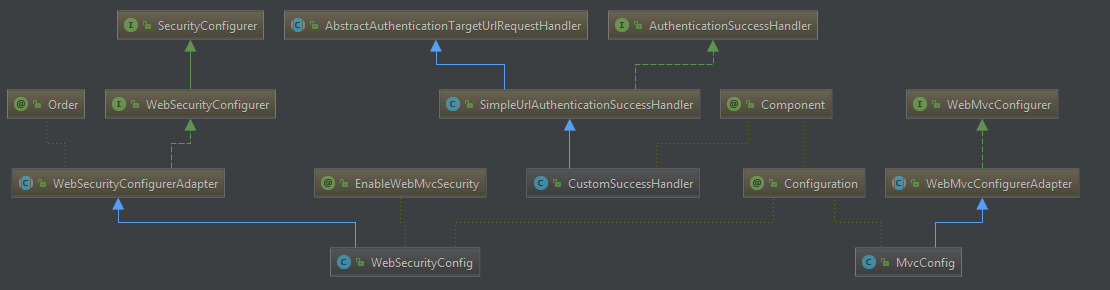
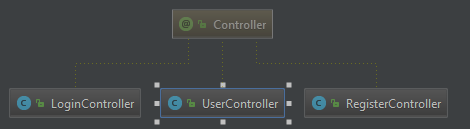
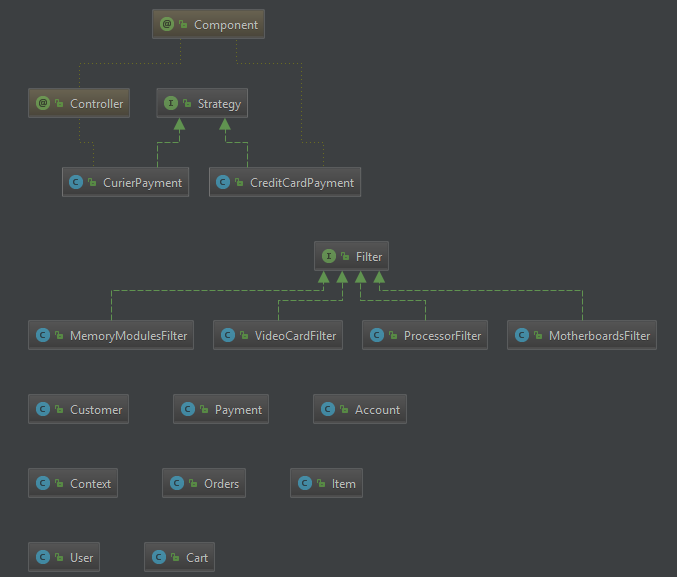


## Class Design

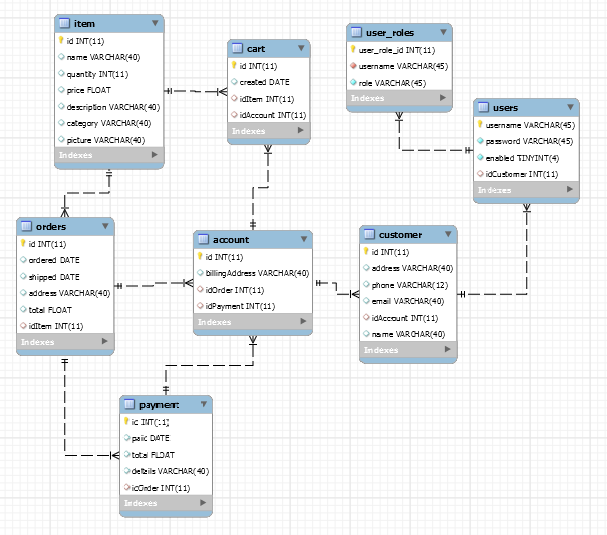
The design patterns I am going to use are the following:

* Filter pattern to easily filter the data retrieved from the database
* Strategy pattern to select the type of payment required to complete an order
* Table module to separate the object model from the actual DB queries
* DAO pattern to structure classes that build the objects with the data form the database
* Layered architectural pattern for the general structure of the

UML Class Diagram



# Data Model



# Unit Testing

For the main operations the system supports tests: delete, update, create, retrieve, etc. I manually tested the outcome of the operations, for example:

On login button click, form login view – this takes the user to admin view or user view depending on the introduced credentials.

On the login page, after completing the required field to log in into the account, the application takes the user to the main page, where he can start adding items to his shopping cart.

# Elaboration – Iteration 2

# Architectural Design Refinement

I have chosen the layered architectural pattern because it focuses on the grouping of related functionality within the application into distinct layers that are stacked vertically on top of each other. Communication between layers is explicit and loosely coupled.

I have chosen Filter pattern because it is a design pattern that enables developers to filter a set of objects using different criteria and chaining them in a decoupled. I use this pattern for filtering the items retrieved from the database, using their category as criteria.

Strategy pattern is used to select the payment method for finishing an order. In this pattern, a class behavior or its algorithm can be changed at run time. We create objects which represent various strategies and a context object whose behavior varies as per its strategy object.

# Construction and Transition

# System Testing

For this project I have not applied integration testing.

# Future improvements

This application could be improved by adding more user types, adding more features into the user interface like the possibility to create an account using an already existent social media account. This application could also have more payment methods, like PayPal or it could have even a Bitcoin payment method.

# Bibliography

* <https://www.javatpoint.com/spring-mvc-crud-example>
* <https://www.tutorialspoint.com/design_pattern/filter_pattern.htm>
* <https://www.tutorialspoint.com/design_pattern/strategy_pattern.htm>
* <https://docs.spring.io/docs/Spring-MVC-step-by-step/>
* <https://www.tutorialspoint.com/design_pattern/data_access_object_pattern.htm>
* <https://martinfowler.com/eaaCatalog/tableModule.html>
* <https://martinfowler.com/eaaCatalog/dataMapper.html>