Flower Shop

Analysis and Design Document

Student: Mateiu Bianca

**Group: 30434**

Revision History

|  |  |  |  |
| --- | --- | --- | --- |
| **Date** | **Version** | **Description** | **Author** |
| 16/03/202 | 1.0 | Project Deliverable 1: Project Analysis and Design (I. Project Specification, II. Elaboration Iteration, V. Construction and Transition) | Mateiu Bianca |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

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

The objective of this project is to design and implement a desktop application for a Flower Shop. The application should allow two types of users, a regular user and an administrator, each of them having different access rights. The application should also integrate a sign-in method, based on username and password.

Given that this project has academical purpose, it is required that we use the Model-View-Controller architectural pattern and the Factory design pattern.

The regular user can perform the following operations:

* + Search list of flowers/bouquets
  + Buy one item at a time with his money (only admin can set wallet amount)

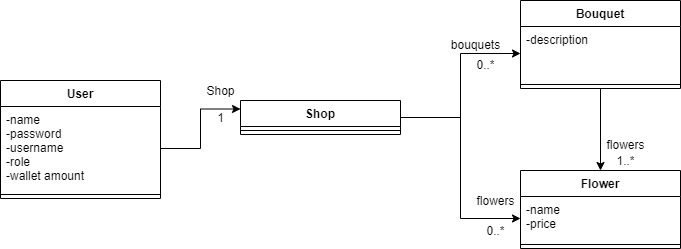
The administrator can perform the following operations:

* + CRUD on flowers/bouquets
  + CRUD on regular users' information
  + Generate two types of report files, one in pdf format and one in txt or html format, with the sold flowers

The data used in the application needs to be saved in a database, using an Object Relational Mapping framework.

# Elaboration – Iteration 1.1

# Domain Model



# Architectural Design

## Conceptual Architecture

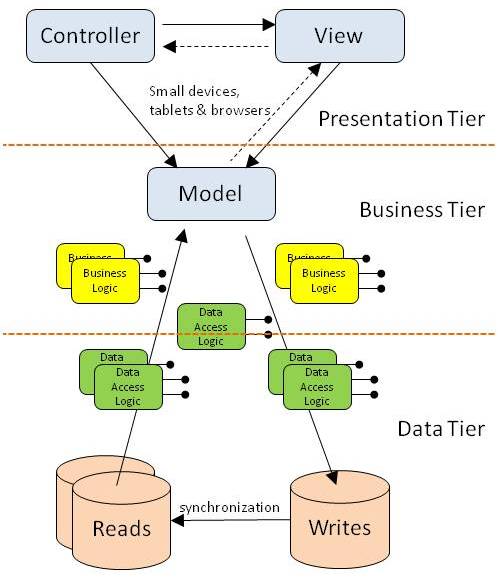
The application must have a Three Tier Architecture. The three-tier architecture separates the deployment of software components into three logical layers. These layers include:

* + Presentation Tier - responsible for rendering the User Interface
  + Business Tier (or Logic Tier) - responsible for processing the business rules or logic
  + Data Tier - responsible for interacting with the data storage system

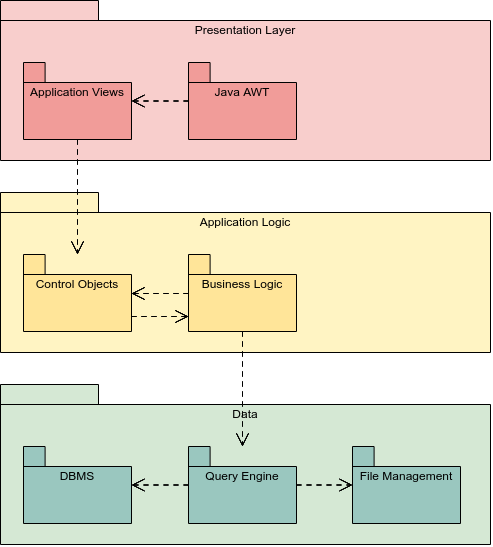
This separation is for a number of reasons which include maintainability, reuse and deployment. When software is modularized and separated into these three-tiers, the modules can be deployed to different server infrastructures for security, scalability and performance. And the deployment approach can change as the need requires. The modules are also easier to maintain and reuse when they are each built for a discrete purpose.

We are also required to use the MVC design pattern. The MVC pattern separates different parts of rendering the User Interface (UI) into modules, so the code for rendering the actual interface is separated from the code that manages the data which is also separated from the code that handles the user events.

When the MVC and three-tier approaches are brought together the View and Controller are considered the presentation tier, and the Model exists in the business tier (and has access to many business and data tier modules). This can be observed in the following diagram:

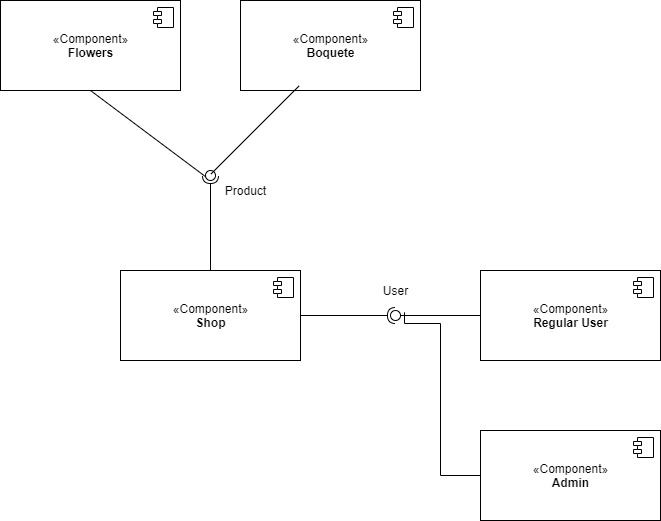


## Package Design



## Component and Deployment Diagrams

Component diagram:



Deployment diagram:

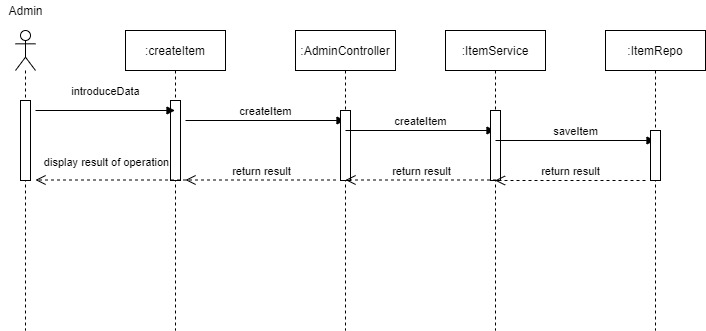


# Elaboration – Iteration 1.2

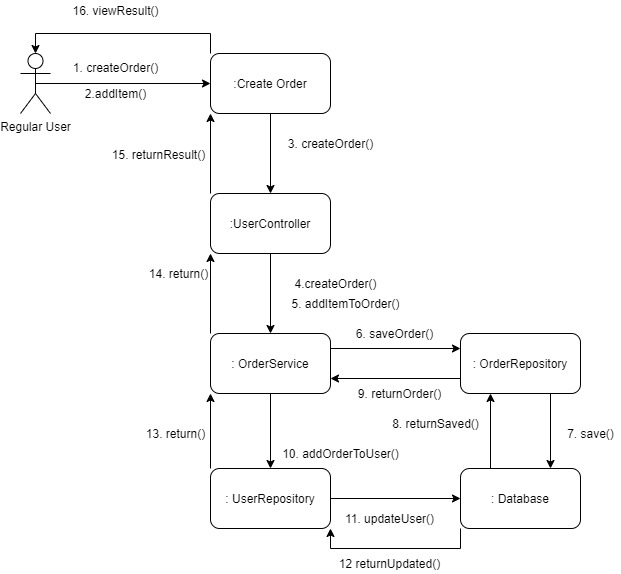
# Design Model

## Dynamic Behavior

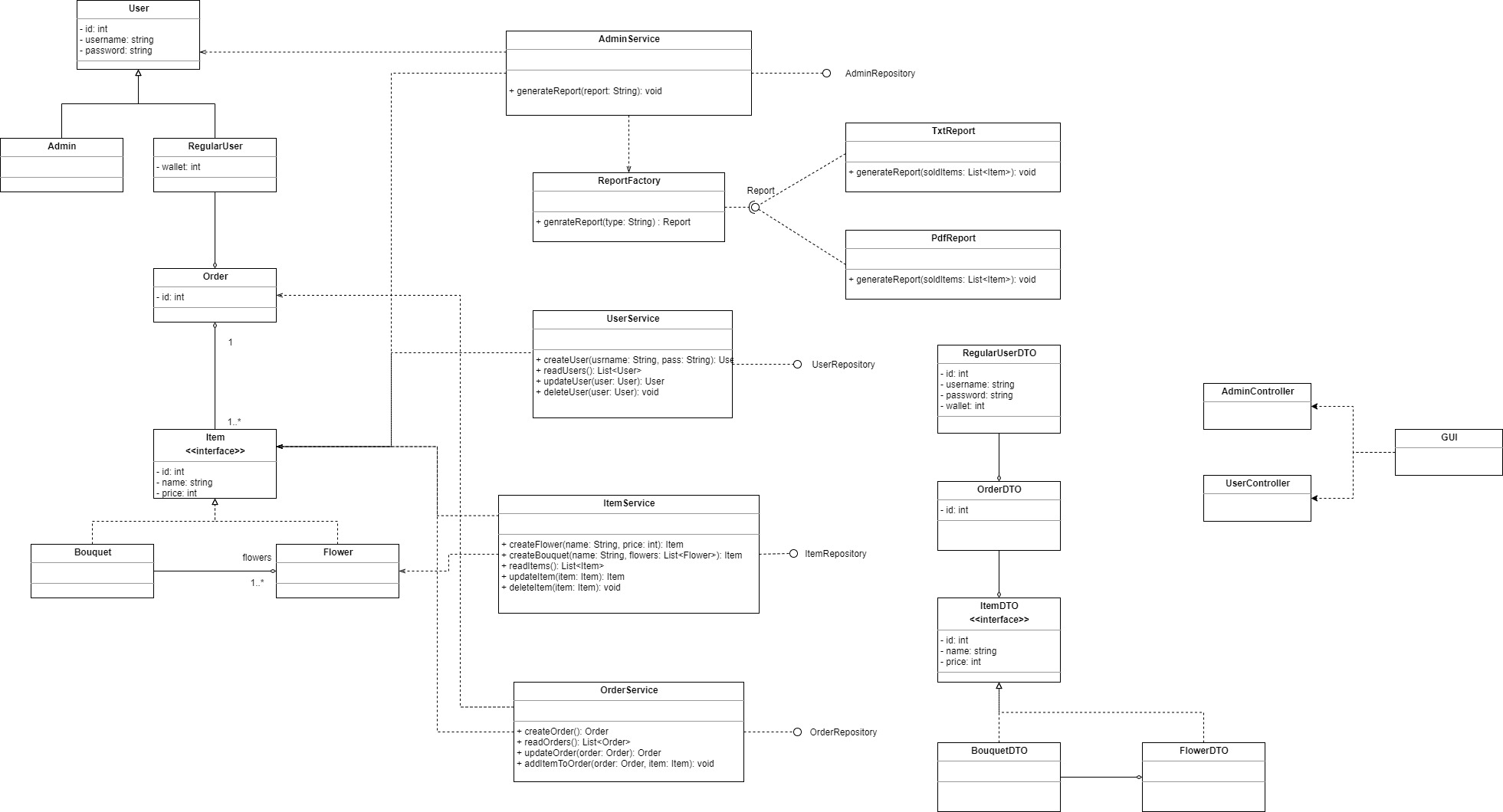
Sequence diagram for admin creating an item (flower or bouquet):



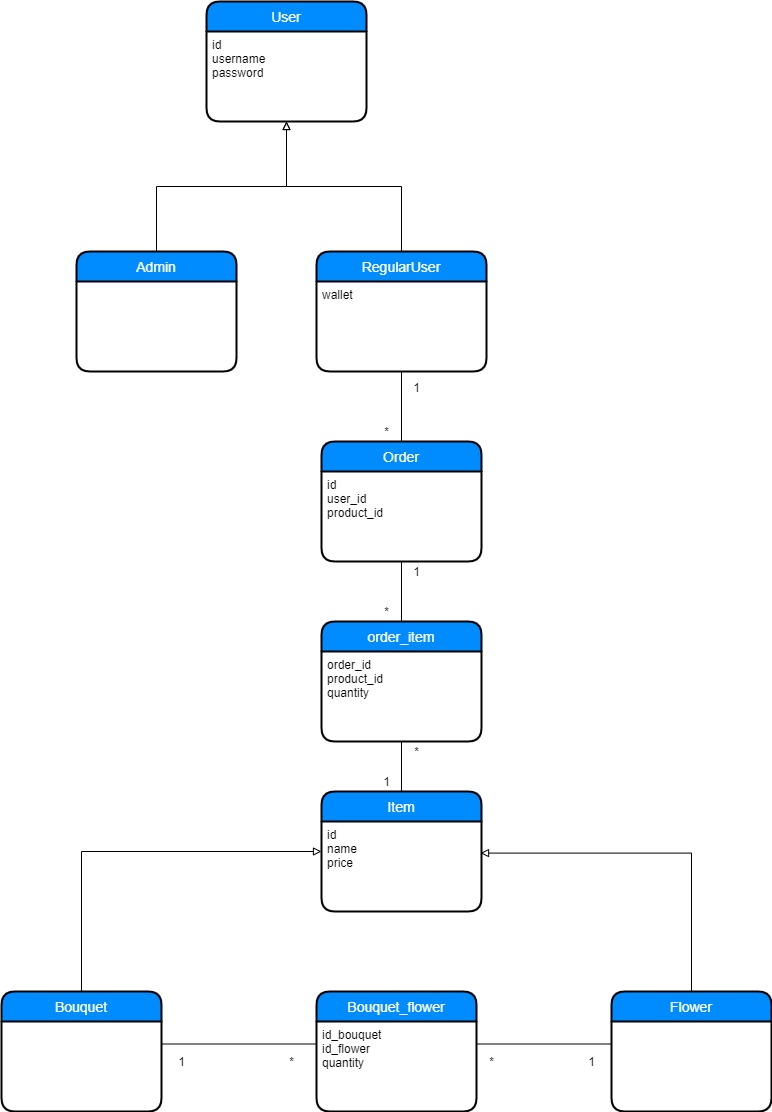
Communication diagram for regular user ordering an item:



## Class Design



# Data Model



# Unit Testing

I created Test Cases for Unit Testing for the methods in the classes from the business-layer. For this I used JUnit.

The main test cases include scenarios of Create, Read, Update and Delete items from my data model.

# Elaboration – Iteration 2

# Architectural Design Refinement

*[Refine the architectural design: conceptual architecture, package design (consider package design principles), component and deployment diagrams. Motivate the changes that have been made.]*

# Design Model Refinement

## *[Refine the UML class diagram by applying class design principles and GRASP; motivate your choices. Deliver the updated class diagrams.]*

# Construction and Transition

# System Testing

Integration testing is a level of software testing where individual units are combined and tested as a group. The purpose of this level of testing is to expose faults in the interaction between integrated units.

# Future improvements

Some future improvement would be to allow the regular user to set his wallet amount, because for this assignment, only the Admin can accomplish that. Also, the user may want to buy more than one item at a time, which is the current functioning of the application.

Another improvement would be to allow the user to build the bouquet however he wants, by allowing him to choose the flowers.

# Bibliography