<FastPay>

Analysis and Design Document

Student: Jurj Alexandru

**Group:30233**

Revision History

|  |  |  |  |
| --- | --- | --- | --- |
| **Date** | **Version** | **Description** | **Author** |
| <22/04/2019> | <1.0> | Iteration 1.1 | Jurj Alexandru |
| <08/05/2019> | <1.1> | Iteration 1.2 | Jurj Alexandru |
| <30/05/2019> | <1.2> | Iteration 1.3 | Jurj Alexandru |
|  |  |  |  |

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

*[Present the project specification]*

# Elaboration – Iteration 1.1

# Domain Model

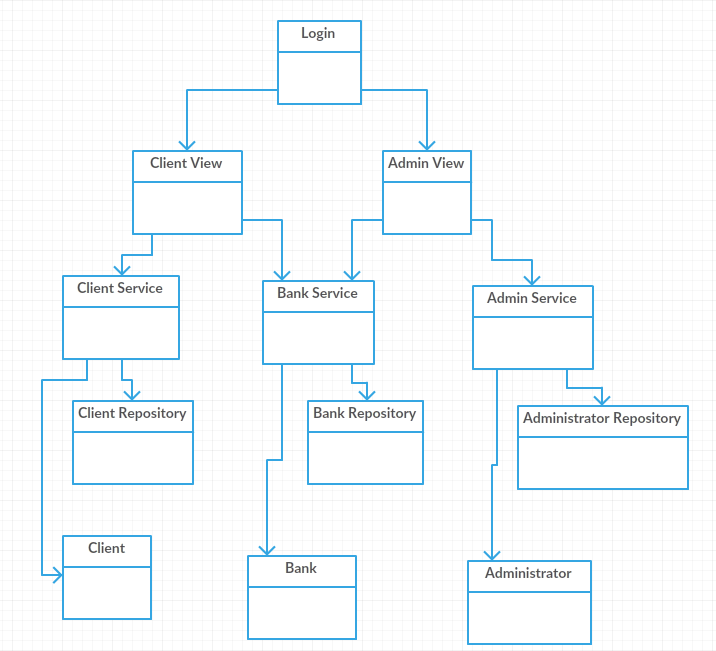
Domain model:

-Client: any user that has a valid account

-Admin: user that can add or delete matches

-Bank: contains information about the bank

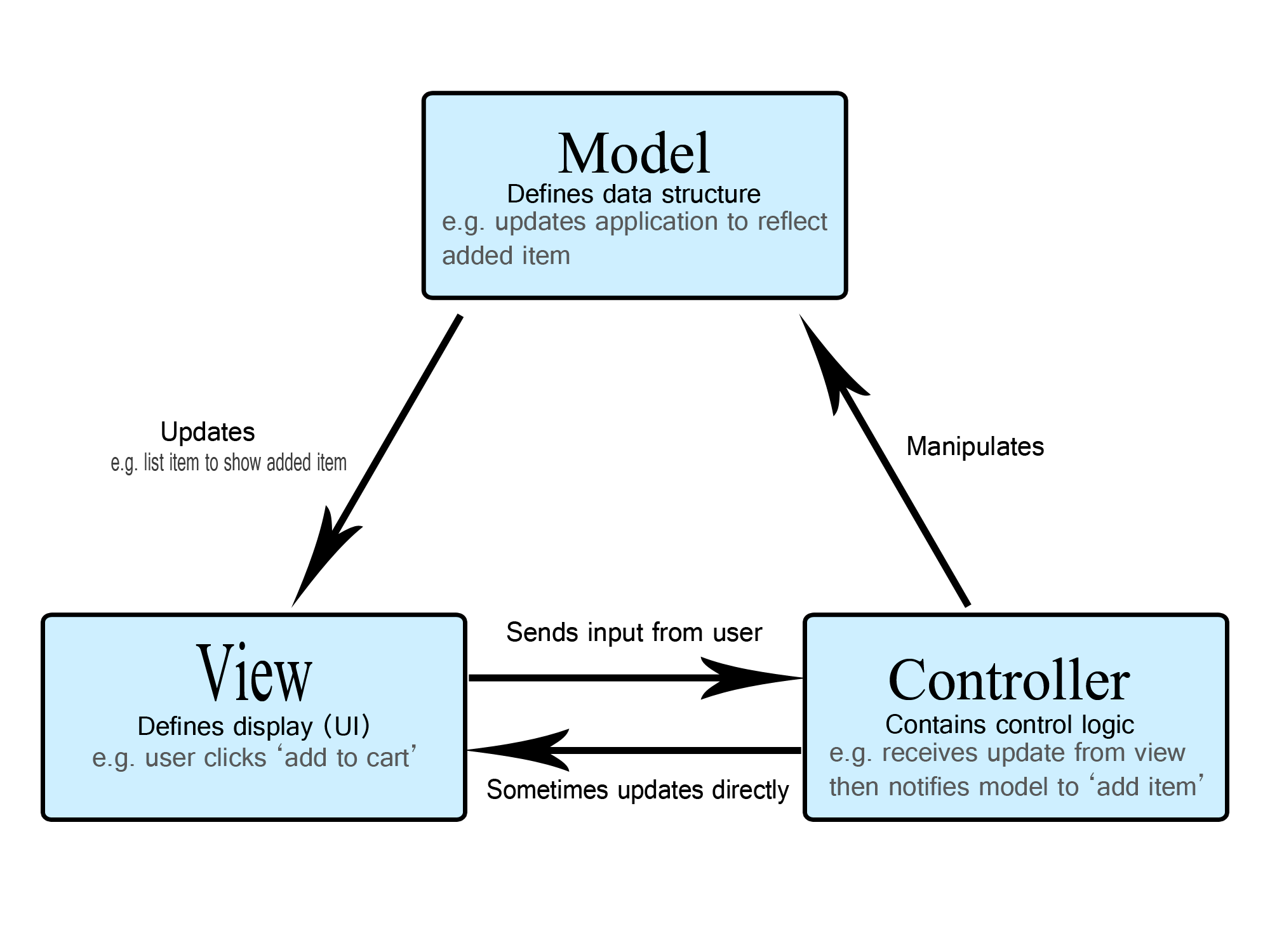
Class Diagram:



# Architectural Design

## Conceptual Architecture

**Model–View–Controller** (usually known as MVC) is an architectural pattern commonly used for developing user interfaces that divides an application into three interconnected parts. This is done to separate internal representations of information from the ways information is presented to and accepted from the user. The MVC design pattern decouples these major components allowing for efficient code reuse and parallel development.



**Model**

The central component of the pattern. It is the application's dynamic data structure, independent of the user interface. It directly manages the data, logic and rules of the application.

**View**

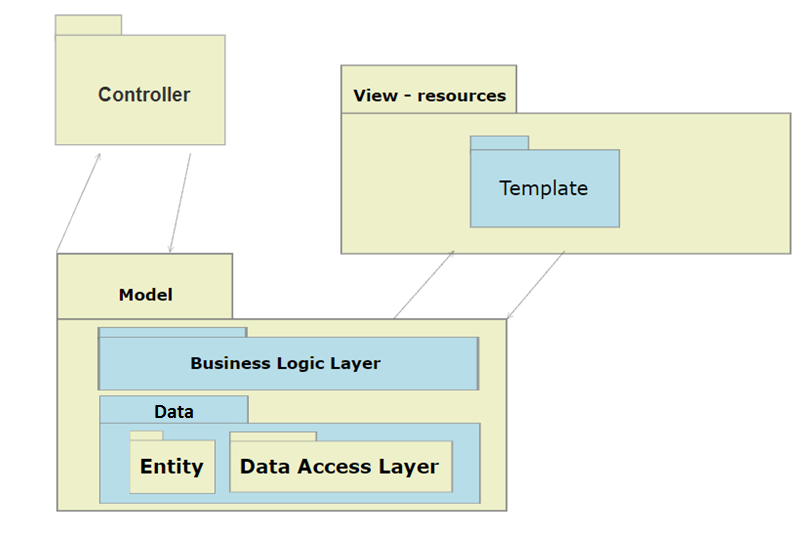
Any representation of information such as a chart, diagram or table. Multiple views of the same information are possible, such as a bar chart for management and a tabular view for accountants.

**Controller**

Accepts input and converts it to commands for the model or view.

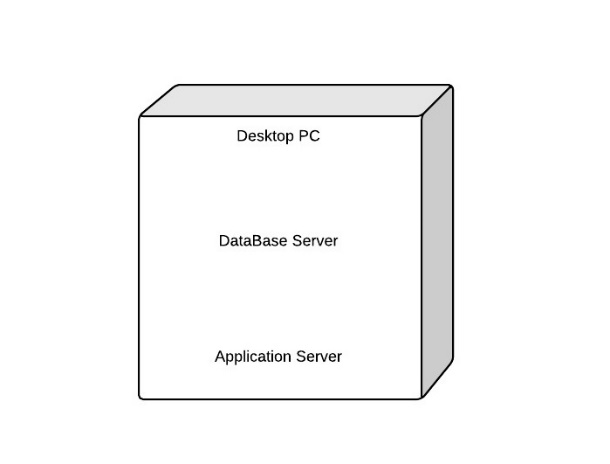
## Package Design

Package Diagram:



## Component and Deployment Diagrams

Deployment Diagram



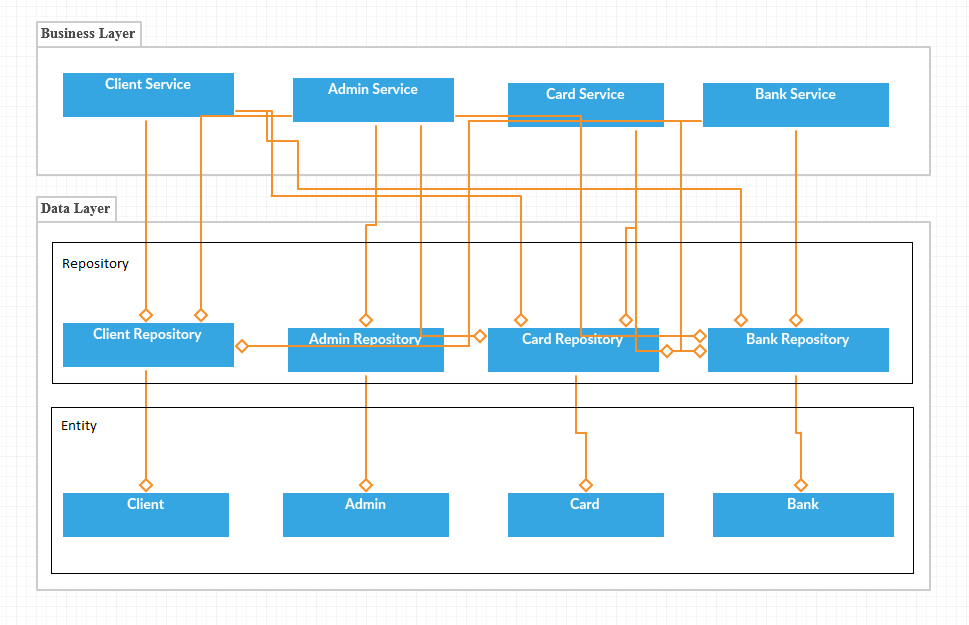
# Elaboration – Iteration 1.2

# Design Model

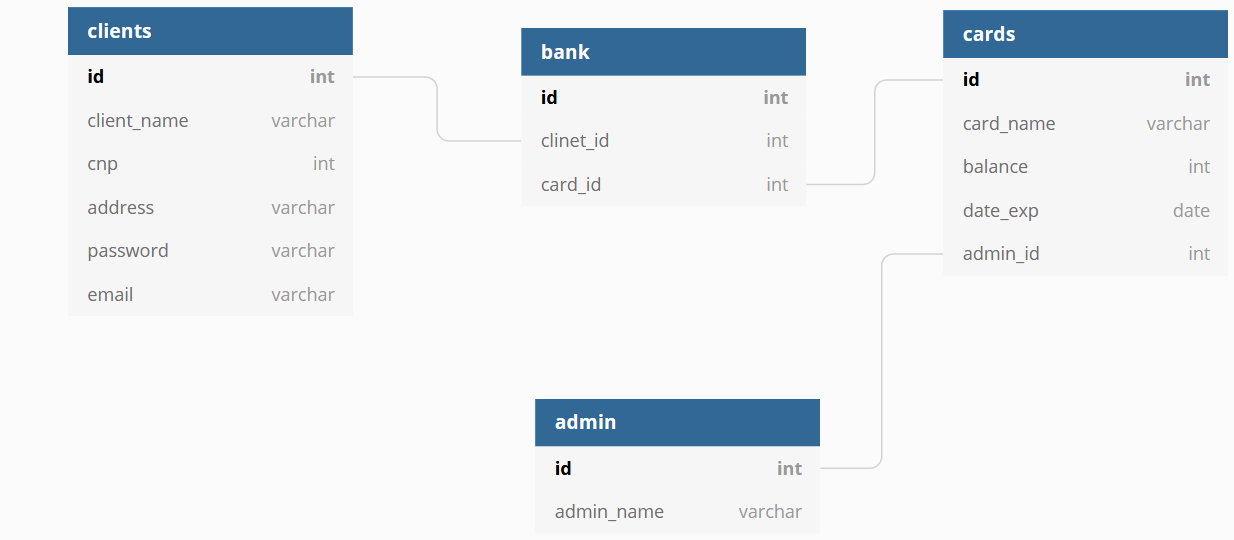
## Dynamic Behavior

## 

## Class Design



# Data Model

**

# Unit Testing

A unit test is a piece of code written by a developer that executes a specific functionality in the code to be tested and asserts a certain behavior or state.

The percentage of code which is tested by unit tests is typically called test coverage.

A unit test targets a small unit of code, e.g., a method or a class. External dependencies should be removed from unit tests, e.g., by replacing the dependency with a test implementation or a (mock) object created by a test framework.

Unit tests are not suitable for testing complex user interface or component interaction. For this, you should develop integration tests

# Elaboration – Iteration 2

# Architectural Design Refinement

No architectural design changes were made.

# Design Model Refinement

I added a Money Class which is used by a money convertor.

# Construction and Transition

# System Testing

*[Describe how you applied integration testing and present the associated test case scenarios.]*

# Future improvements

As future improvements, I could add some extra operations for the users, both clients and admin. Also, the view can be improved, in order to be more friendly with the user.

# Bibliography

[1] <https://jsoup.org/>

[2]<https://www.w3schools.com/html/>

[3]<https://www.tutorialspoint.com/spring_boot/index.htm>

[4]<https://stackoverflow.com/questions/tagged/spring>