Tax Manager

Analysis and Design Document

Student: Sorin Urcan

**Group: 30442**

Revision History

|  |  |  |  |
| --- | --- | --- | --- |
| **Date** | **Version** | **Description** | **Author** |
| 06/Apr/17 | 1.0 | First commit | Sorin Urcan |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

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

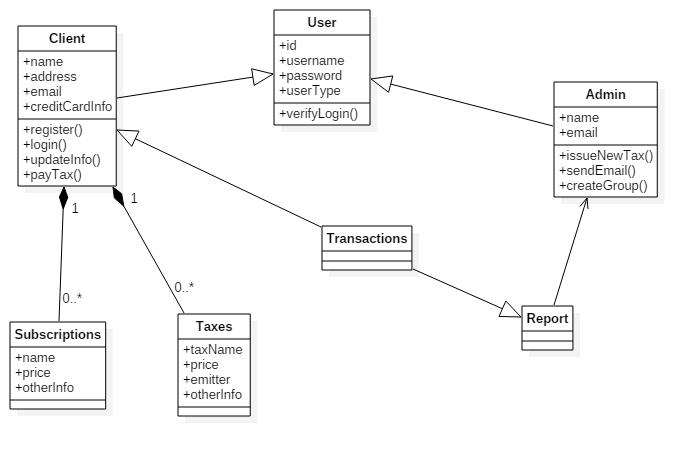
3. Tickets/Fines/Utilities/Subscriptions Payment Manager

Design and implement a client-server application for managing the payment of any kind. The application consists of two components which can be accessed using a username and a password. One component is used by a tax emiter who creates a request for a user to pay. The other component is used by the client to do the following operations: (1) add/update/delete the information (name, age, genre, bank card details, identity card details), (2) add/update/delete taxes, (3) create reports. When a tax emitter issues a tax to a client, the client is notified by email and/or account notification. All the information is stored in a database. Emitter can generate reports, create groups of clients, send mail to a group, generate taxes to a group, cancel a tax/subscription.

# Elaboration – Iteration 1.1

# Domain Model

*[Define the domain model and create the conceptual class diagrams]*

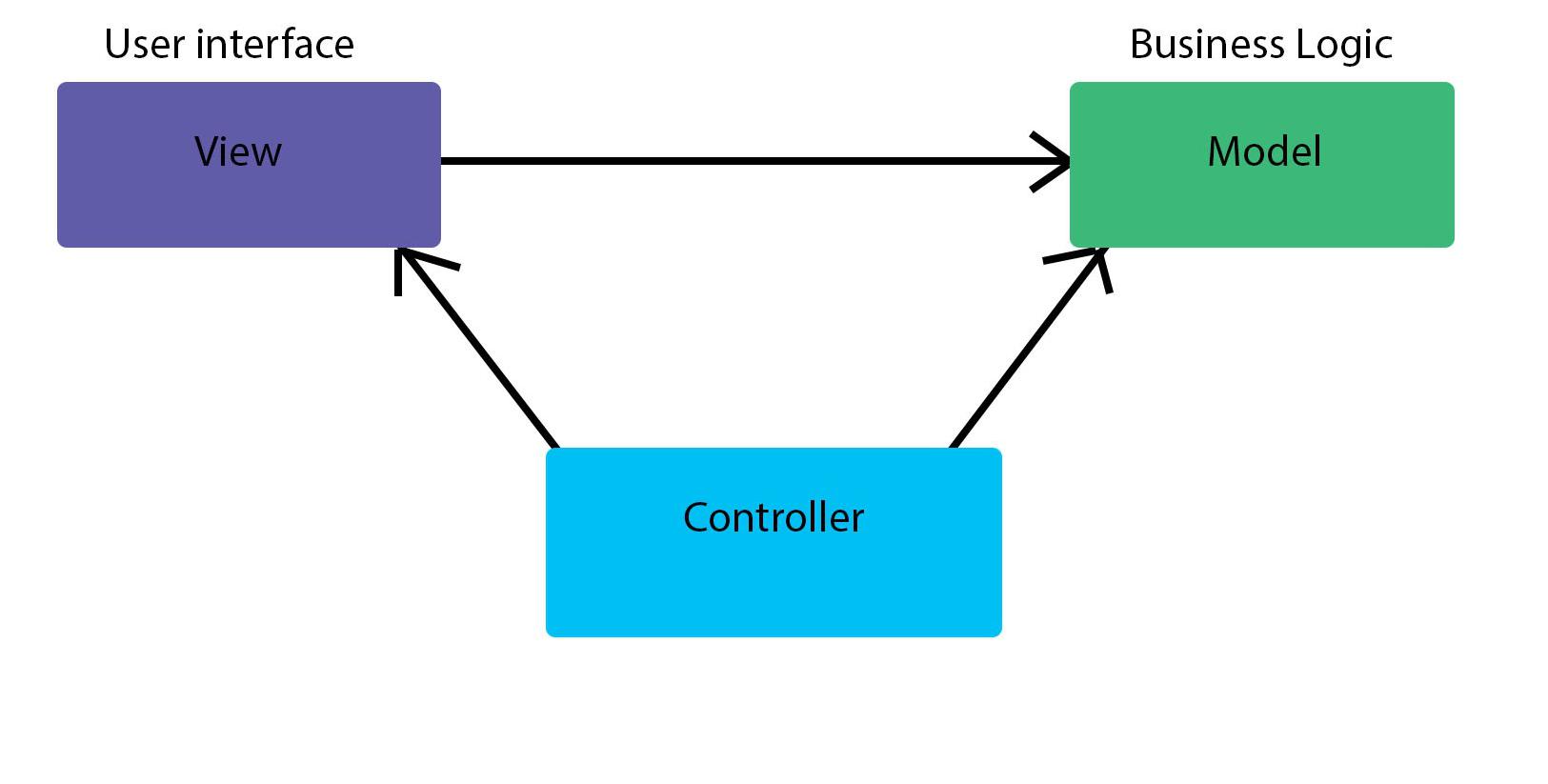


# Architectural Design

## Conceptual Architecture

The chosen architectural pattern is the model-view-controller. It fits the requirements very well, since it divides the software into three main parts, the model, the view and the controller. For our system, the model is represented by the User, Client, Admin, Tax the view is represented by the user interface and the controller is the domain logic of the system.

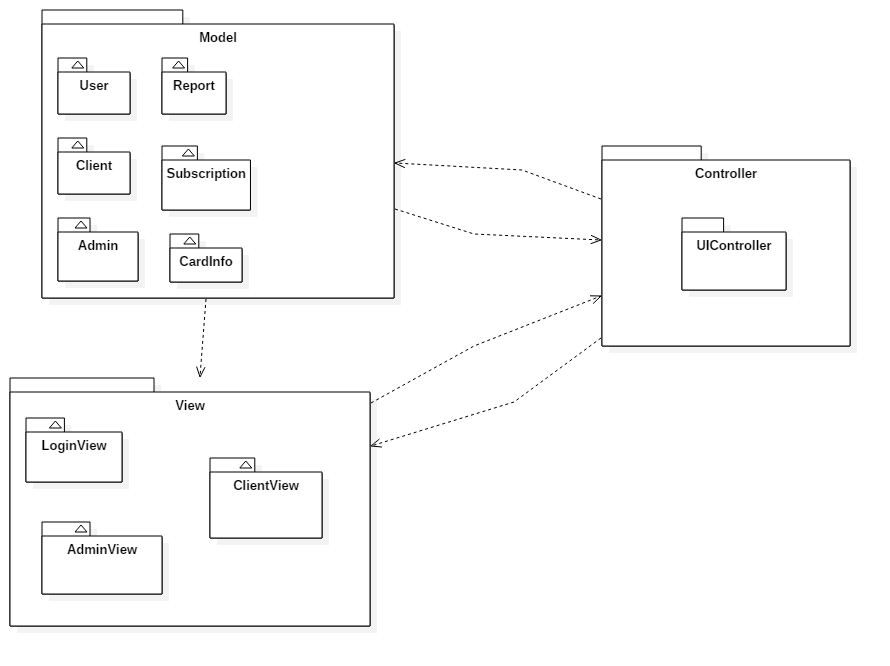
These three components interact with each other. The controller can send commands to the model to change it’s state, or send commands to the view in order to update it’s presentation of the model. The view requests information from the model through the controller. By interacting with the view, the user sends commands through the controller, which can modify the model or the view.



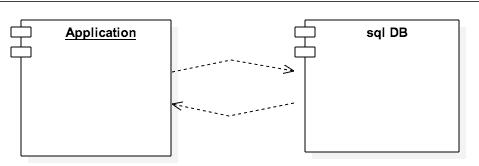
The architectural style is client-server based. This facilitates storage of large amounts of data, data which can be accessed remotely.

The Singleton pattern restricts the instantiation of a class to a single object. In our implementation, it is useful in instantiating only once the Database connection.

## Package Design



## Component and Deployment Diagrams



# Elaboration – Iteration 1.2

# Design Model

## Dynamic Behavior

*[Create the interaction diagrams (1 sequence, 1 communication diagrams) for 2 relevant scenarios]*

## Class Design

*[Create the UML class diagram; apply GoF patterns and motivate your choice]*

# Data Model

*[Create the data model for the system.]*

# Unit Testing

*[Present the used testing methods and the associated test case scenarios.]*

# 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

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

# Future improvements

*[Present future improvements for the system]*

# Bibliography