Bank Manager

Analysis and Design Document

Student: Emiliana Popa

**Group: 30233**

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

Use JAVA/C# API to design and implement an application for the front desk employees of a bank. The application should have two types of users (a regular user represented by the front desk employee and an administrator user) which have to provide a username and a password in order to use the application.

# Functional Requirements

The regular user can perform the following operations:

* Add/update/view client information (name, identity card number, personal numerical code, address, etc.).
* Create/update/delete/view client account (account information: identification number, type, amount of money, date of creation).
* Transfer money between accounts.
* Process utilities bills.

The administrator user can perform the following operations:

* CRUD on employees’ information.
* Generate reports for a particular period containing the activities performed by an employee.

# Non-functional Requirements

The data will be stored in a database. Use the Layers architectural pattern to organize your application. Use a domain logic pattern (transaction script or domain model) / a data source hybrid pattern (table module, active record) and a data source pure pattern (table data gateway, row data gateway, data mapper) most suitable for the application

All the inputs of the application will be validated against invalid data before submitting the data and saving it in the database.

2. Use-Case Model

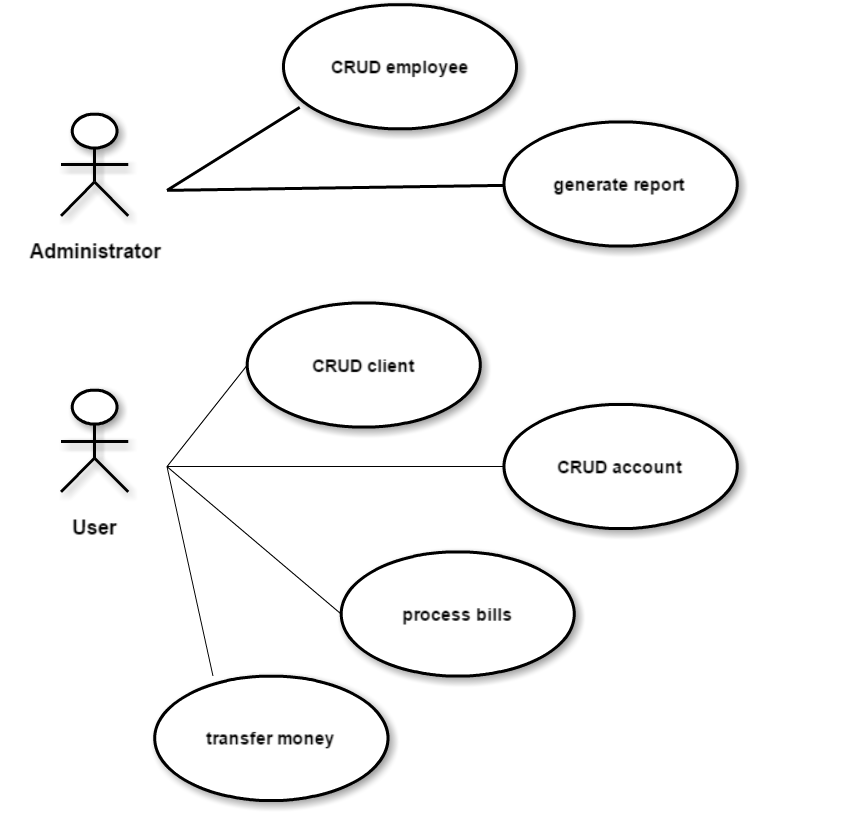
*Use case:* log in

*Level:* user-goal level

*Primary actor:* Regular user

*Main success scenario:* provide username and password and press the log in button in which case the actor will see a menu of the rest of actions he can perform

*Extensions:* the actor doesn’t have an account so he has to ask the administrator to create one

****

3. System Architectural Design

**3.1 Architectural Pattern Description**

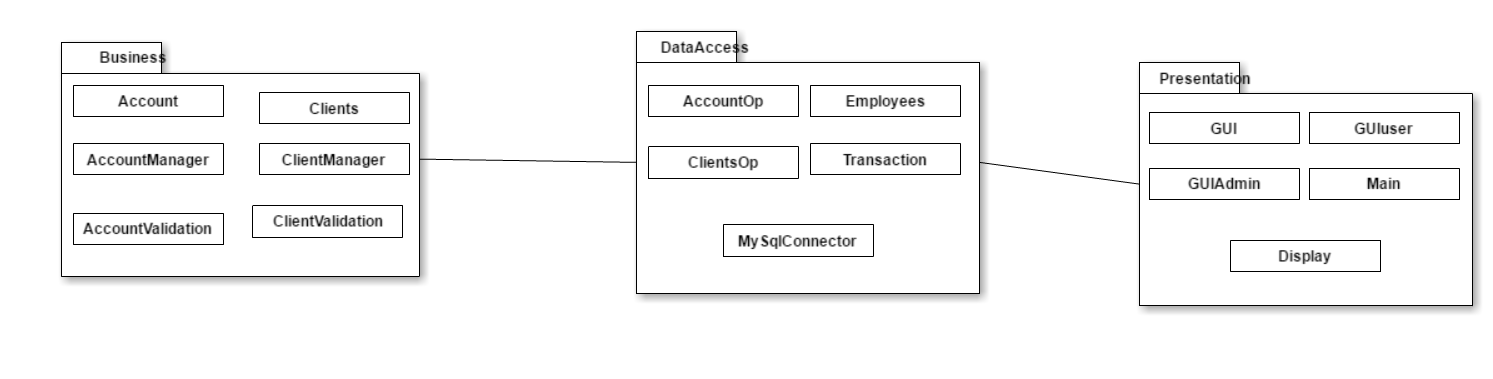
The **Layers** architectural pattern structures applications that can be decomposed into groups of subtasks in which each group of subtasks is at a particular level of abstraction.

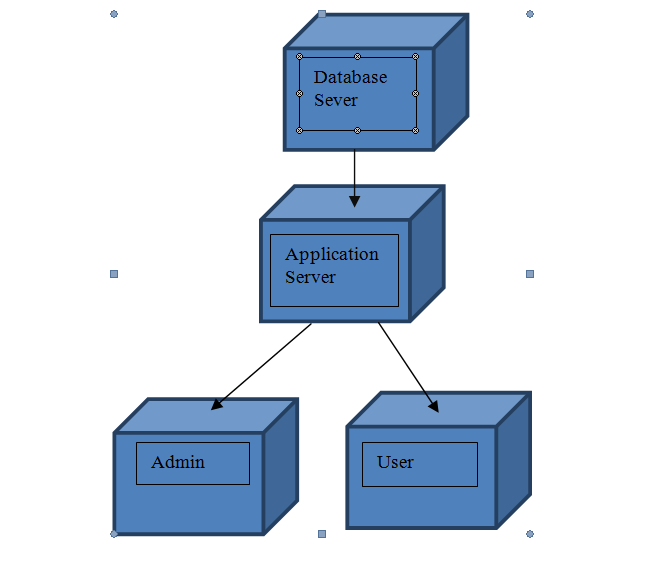
In the data access layers will perform CRUD (Create, Read, Update, Delete) operations.

Business logic or domain logic is the part of the program that encodes the real-world [business rules](https://en.wikipedia.org/wiki/Business_rule) that determine how data can be [created, stored, and changed](https://en.wikipedia.org/wiki/Create,_read,_update_and_delete).

The presentation layer is responsible for the delivery and formatting of information to the application layer for further processing or display.

**3.2 Diagrams**

**

******

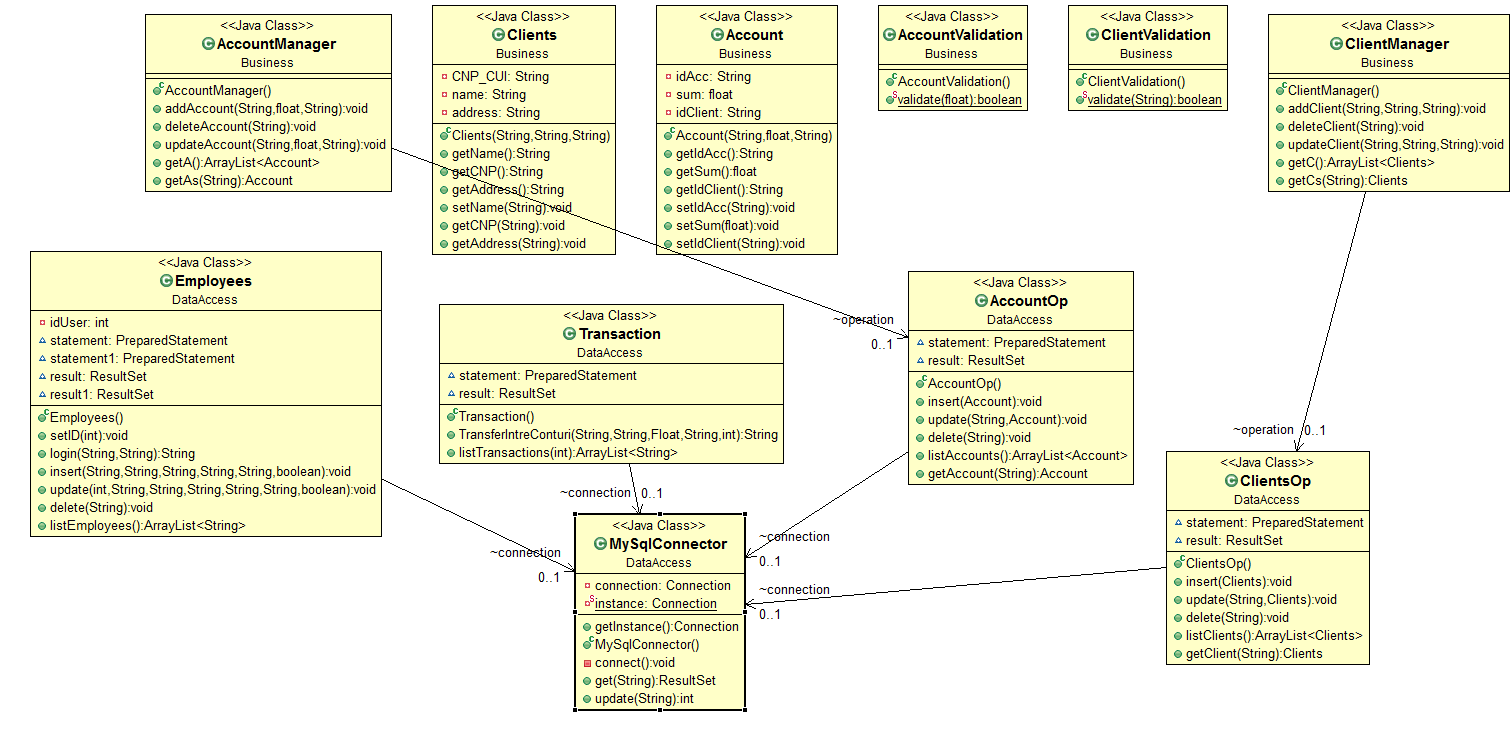
4. Class Design

**4.1 Design Patterns Description**

**Data Mapper** is a layer of Mappers that moves data between objects and a database while keeping them independent of each other and the mapper itself. It is a layer of software that separates the in-memory objects from the database. Its responsibility is to transfer data between the two and also to isolate them from each other. With Data Mapper the in-memory objects needn't know even that there's a database present; they need no SQL interface code, and certainly no knowledge of the database schema.

**Transaction Script** organizes all this logic primarily as a single procedure, making calls directly to the database or through a thin database wrapper. Each transaction will have its own Transaction Script, although common subtasks can be broken into subprocedures.

**4.2 UML Class Diagram**

**

5. Data Model

In my database I use four tables: clients, employees, account and transaction, a procedure which helps me transfer money from an account to another and two triggers which help put data in some tables automatically (For example when I add a client an account will be generated for him).

6. System Testing

I used two methods for testing the data introduced in the graphical interface by the user: one of them is checking whether the CNP has 13 digits or 8 for companies and the other method is checking if the sum is < 0 when the user wants to take money from the account.

7. Bibliography

Martin Fowler et. al, Patterns of Enterprise Application Architecture