Bank Application

Student: Daiana Toader

**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 description format:*

*Use case: Transfer money between accounts*

*Level: user-goal level*

*Primary actor: regular user/admin*

*Main success scenario: The sum is successfully transferred between accounts*

*Extensions: The source account is a saving account/ the source account has an amount lower than the sum wanted to transfer/ the user enters characters instead of numbers.*

3. System Architectural Design

**3.1 Architectural Pattern Description**

For this project I choose to use Layered Architectural Pattern, because it separates very well the data access part from the business part and the presentation part where is the applications GUI.

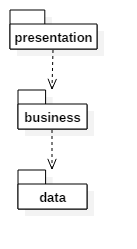
This architecture contains three layers:

* Data access: Here are the models from DB, the classes for Table Data Gateway and the Active Record classes, where I implement the CRUD operations.
* Business: Services that have methods that make specific operations and functionalities for Account, Client, User, Report, Transfer, Bill.
* Presentation: This layer contains the application’s GUI.

**3.2 Diagrams**

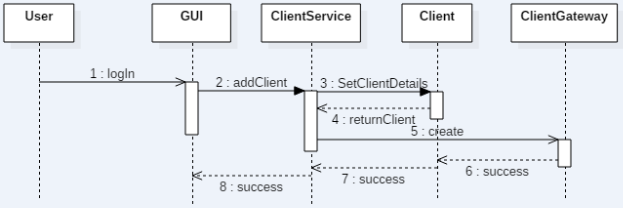
The application contains three packages that define the chosen architecture:

* data
* business
* presentation



4. UML Sequence Diagrams

Sequence diagram for adding a new user in the database.



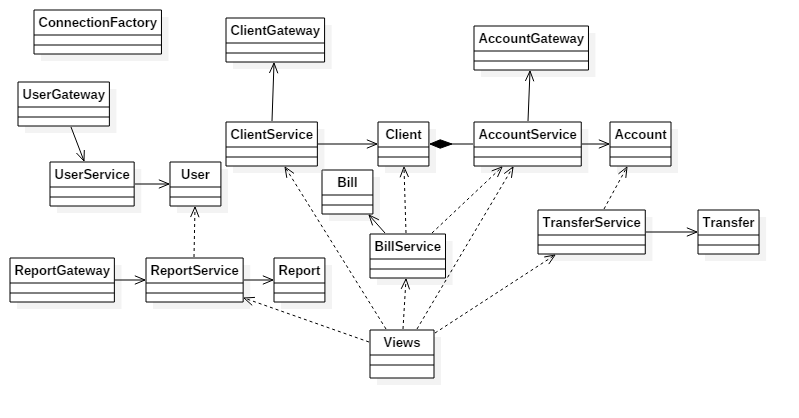
5. Class Design

**5.1 Design Patterns Description**

For this application I have chosen to use:

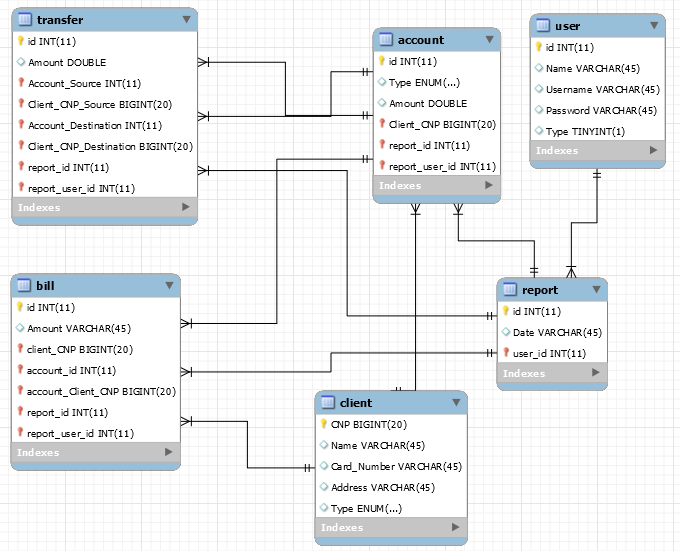
* Table Data Gateway: It holds all the SQL for accessing a single table or view: selects, inserts, updates, and deletes. In services are called its methods for all interaction with the database.
* Active Record: It has an object that carries both data and behavior. It uses the most obvious approach, putting data access logic in the domain object

**5.2 UML Class Diagram**



6. Data Model

I choose six entities for describe the functionality of my application: Client, User, Account, Transfer, Bill and Report. All the entities can be visualized in the next data model diagram:



7. System Testing

All the operations that a user and an admin wants to make are tested before they are send to the database. The users need to introduce a valid username and password so they can log in, a valid CNP for a Client and a valid sum for account creation, transferring moneys and paying utilities bills. The account source for transferring money and paying utilities bills needs to be a SPENDING one. If all of these requirements are not fulfilled than a warning message pops up and the user need to do the operation again.

8. Bibliography

* <https://www.martinfowler.com/eaaCatalog/activeRecord.html>
* <https://www.martinfowler.com/eaaCatalog/tableDataGateway.html>
* <http://stackoverflow.com/>