Application Bank

Analysis and Design Document

Student: Mocan Ioana

**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

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
* Create/update/delete/view client account
* 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

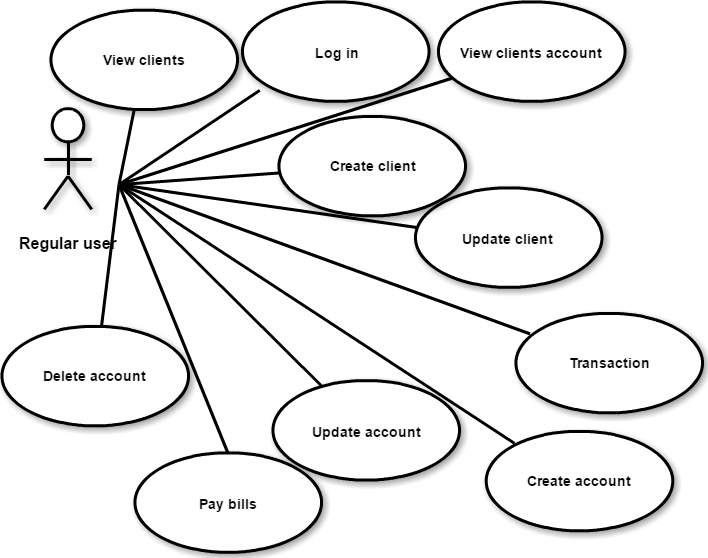
The application will have two types of users that can login on the application. The regular user will be asked what option he chooses meaning : client , account , transaction or pay bills .

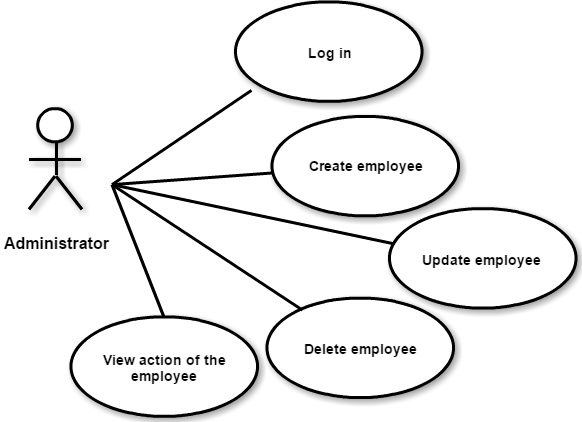
The first two option bring another set of option if the user want to create , update or view a client if not he can click on cancel and de first set comes back or if he want to create , update , delete or view the client accounts.

The administrator will be sent to another set of options like create , delete , update or view activities performed by an employee.

All this set will open a new window that will have the fields which has to complete in order to se information or to add to the data base information.

2. Use-Case Model





3. System Architectural Design

**3.1 Architectural Pattern Description**

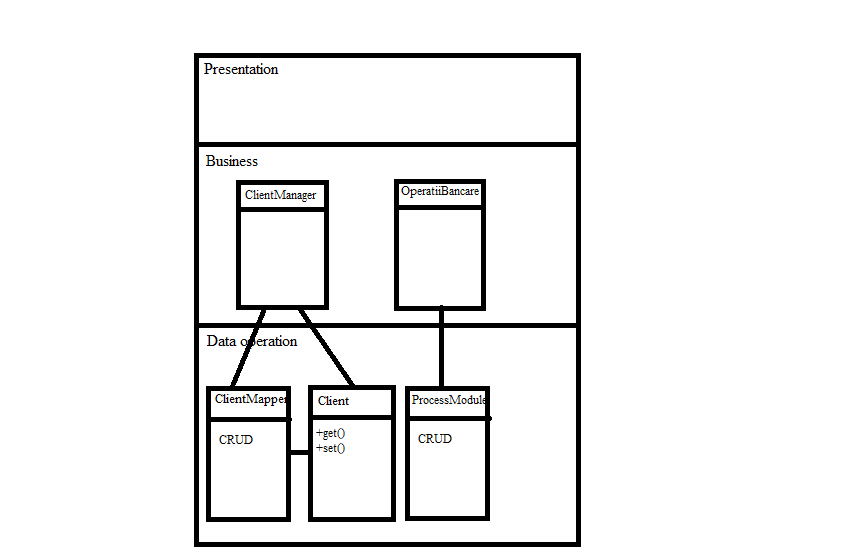
The **Layers** architectural pattern helps to structure applications that can be decomposed into groups of subtasks in which each group of subtasks is at a particular level of abstraction.

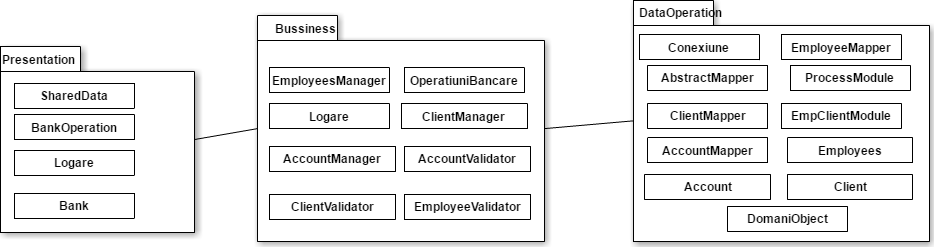
In the persistence layers will perform CRUD (Create, Read, Update, Delete) operations.

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**

****

****

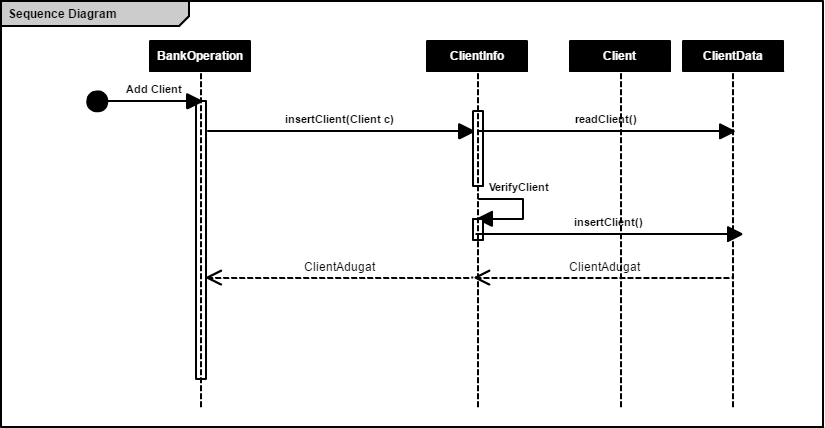
Database Sever

Application Server

Admin

User

4. UML Sequence Diagrams



5. Class Design

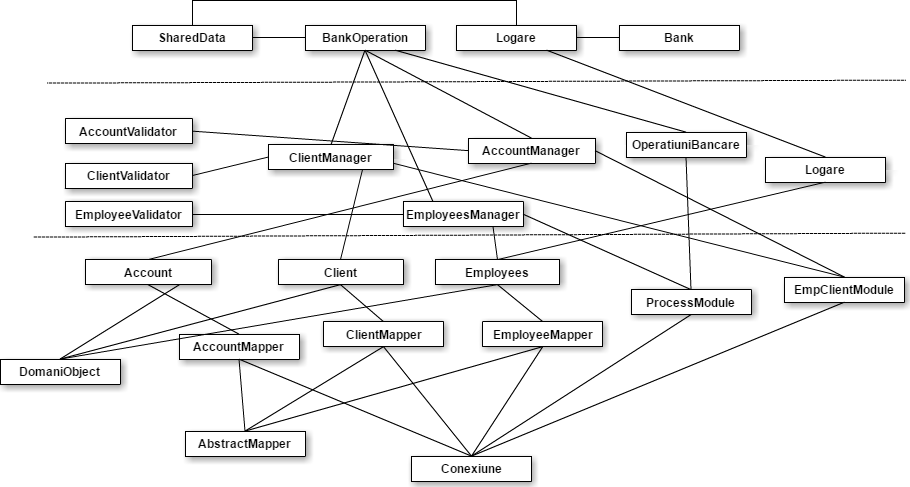
**5.1 Design Patterns Description**

A **Data Mapper** is a [Data Access Layer](https://en.wikipedia.org/wiki/Data_access_layer) that performs bidirectional transfer of data between a persistent data store (often a [relational database](https://en.wikipedia.org/wiki/Relational_database)) and an in-memory data representation (the domain layer). The goal of the pattern is to keep the in-memory representation and the persistent data store independent of each other and the data mapper itself. The layer is composed of one or more mappers (or [Data Access Objects](https://en.wikipedia.org/wiki/Data_Access_Object)), performing the data transfer.

A **Table Module** organizes domain logic with one class per table in the data-base, and a single instance of a class contains the various procedures that will act on the data.

A **Transaction Script** organizes all this logic primarily as a single procedure, making calls directly to the database or through a thin database wrapper.

**5.2 UML Class Diagram**



The design patterns are used by implementing for every table in the data base CRUD and next a mapper is made so the data extracted from the database can be processed and the used for resolving the request form the upper layer meaning the presentation layer. Another pattern used is table module that is a hybrid pattern meaning can be both in the domain and in the data process layer.

6. Data Model

In the database I have 6 tables: Employees , Client , Account , TypeA (type of the account), Process , EmpClient (connection between Employee and Client). Table Employees contains : ID\_Employee (primary key) , UserNume (unique) , Password , UserType . Table Client contains : CNP (primary key), Nume , IDCardNumber , Adress . Table Account contains : IdNumberAcc (primary key) , AmmountOfMoney , DataCreation ,CNP (foreign key) , IdType (foreign key) . Table TypeA contains: IdType (primary key) , TypeA . Table Process contains: ID\_Employee (primary key), IdNumberAcc (foreign key), AmmountOfMoney , DataOperation ,Op . Table EmpClient contains: CNP (foreign key),Operatia ,DataOp , ID\_Employee (foreign key) .

7. System Testing

I designed a couple of tests such that all the inputs of the application (JTextFields) for the classes form business part which are processing data that will be next inserte , update , delete , read . If a is required a number and the user will introduce a string, it will output a JOptionPane with an appropriate message like “the value is not ok” or is not completed correct. I also verify in the login part if the username is ok and the password if they match with the ones in the data base and I next send the employer id and the employer type( employee or admin) to the next window so I can make the difference between the action a employee can do and the admin.

8. Bibliography

1.Martin Fowler et. al, Patterns of Enterprise Application Architecture

2.<https://stackoverflow.com/>

3. <https://martinfowler.com/eaaCatalog>

4. <https://www.tutorialspoint.com/>

5. <https://en.wikipedia.org/wiki>

6. <http://richard.jp.leguen.ca/tutoring/soen343-f2010/tutorials/implementing-active-record/>