

# **Assignment A1**

## **Analysis and Design Document**

**Student: Matei Cristina-Bianca**  
**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	5
4. UML Sequence Diagrams	7
5. Class Design	7
6. Data Model	8
7. System Testing	9
8. Bibliography	14

# 1. Requirements Analysis

## 1.1 Assignment Specification

The assignment requires the implementation of an application for the front desk employee of a bank. It should have two types of users (regular employee and administrator) that have to provide a username and a password in order to use the application.

## 1.2 Functional Requirements

The system should provide a user interface in order to be used by the users. These should have the possibility to choose from logging in as a regular employee or as an administrator. After logging in, they should choose from various operations:

The regular employee should have the possibility to choose from the following:

- Add client, View client information, Update address of a client
- Create/Delete/View client accounts
- Deposit money in an account
- Transfer money between accounts
- Process utilities bills

The administrator should choose from the following:

- Add/Delete employee, View employee information, Update password of an employee
- Generate reports for a particular period containing the activities performed by an employee

The users should provide the corresponding inputs for each operation, especially when they have to provide some integers or reals for the id or the amount of money. The system should provide text fields for every piece of information it needs to perform the operations in order to inform the users where they should write the specific attributes. Also, the system should notify them if an illegal input has been provided, to perform the chosen operation, to inform the users if the operation has finished with success or not and to generate the outputs for the “view” operations.

## 1.3 Non-functional Requirements

The system should have some important properties like maintainability, reusability and portability. The maintainability of this application is generated by a good organization of classes and methods that provide the functionality of the application. The reusability and portability are provided by the implementation of this application in Java, a common programming language on all platforms. It should also store the data in a database, use the Layers architectural pattern, use a domain logic pattern / a data source hybrid pattern and a data source pure pattern.

# 2. Use-Case Model

I will present the use-case description for updating the address of a client:

*Use case:* Update the address of a client

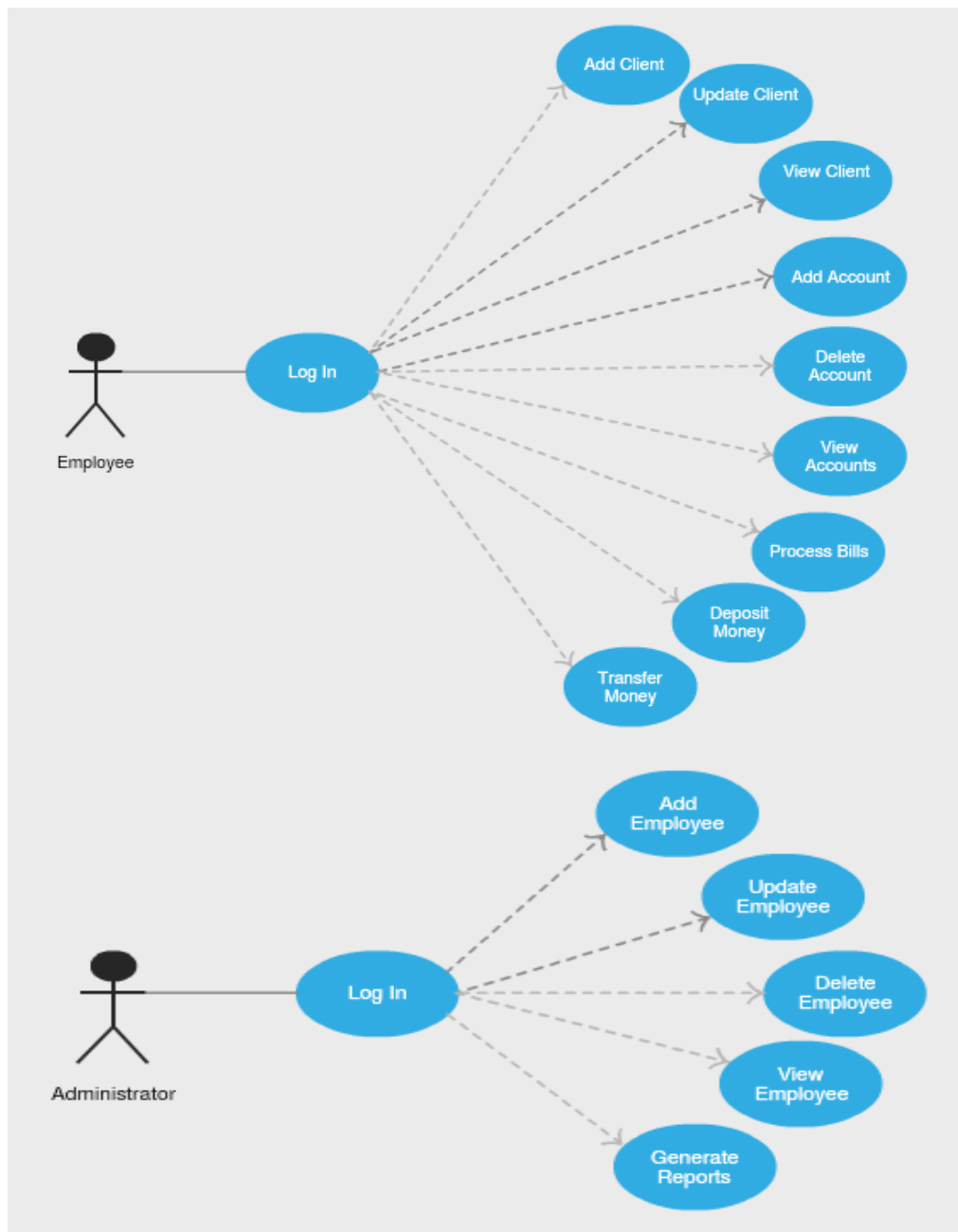
*Level:* User-goal level

*Primary actor:* Employee

*Main success scenario:* log in, introduce the name of the client and the new address, the client exists in the database, update the old address with the new one, success message

*Extensions:* introduce the name of the client and the new address, the client does not exist in the database, the update can't be done, warning message

The use-case diagrams for the application include the actors Employee and Administrator and they are described below:



## 3. System Architectural Design

### 3.1 Architectural Pattern Description

Layered Architecture 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 this application I used a Layered Architecture Pattern containing three layers:

- Presentation Layer
- Business Layer
- Data Source Layer

Presentation layer defines how to handle the interaction between the user and the software. The primary responsibilities of the presentation layer are to display information to the user and to interpret commands from the user into actions upon the business and data source layers.

Business layer involves calculations based on inputs and stored data, validation of any data that comes in from the presentation, and figuring out exactly what data source logic to dispatch. For this layer I have chosen the Table Module Pattern that organizes business logic with one class per table in the database and a single instance of a class contains the various procedures that will act on the data. I have made this choice because this application is almost entirely based on the data stored in the database and this matches perfectly with the Table Module Pattern which is very much based on table-oriented data.

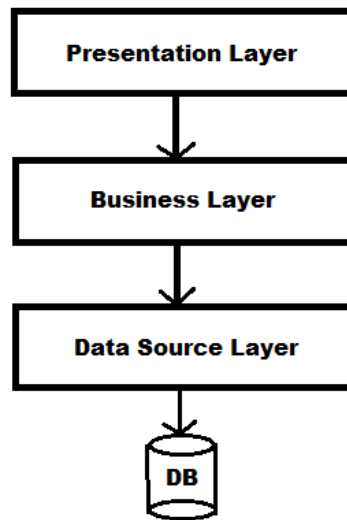
The role of the data source layer is to communicate with the various pieces of infrastructure that an application needs to do its job. Once I have chosen the business layer, I have to figure out how to connect it to my data sources in the data source layer. My decision is strongly based on my business layer choice, so I have chosen the Table Data Gateway Pattern because it best fits with Table Module. A Table Data Gateway holds all the SQL for accessing a single table or view: selects, inserts, updates, and deletes. Other code calls its methods for all interaction with the database.

### 3.2 Diagrams

The patterns described before are implemented in this application as follows:

- Table Module Pattern is implemented by creating a class for each table from database. We have five tables in database, so we also have five classes: Client, Account, Employee, Activity and Admin. Each of these classes defines the specific methods for each table.
- Table Data Gateway Pattern is implemented by creating a Gateway class for each class described before: ClientGateway, AccountGateway, EmployeeGateway, ActivityGateway and AdminGateway. Each of these contains all the SQL for accessing the database.

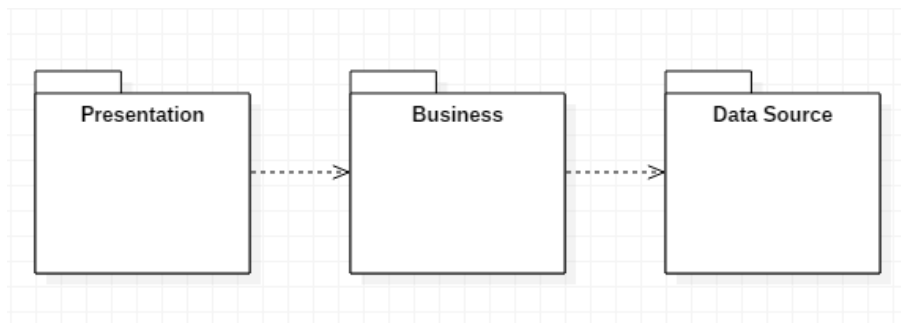
The Layered Architecture of this application that contains the three layers described in the section before is reproduced in the picture below.



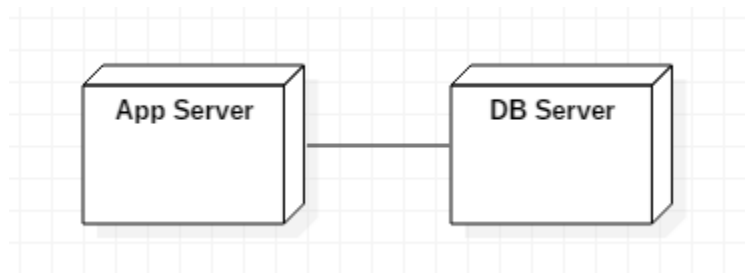
Having these three layers I separated the classes of the application in three packages with the names of the layers:

- Presentation package contains classes View and Controller
- Business package contains classes Client, Account, Employee, Activity and Admin
- Data Source package contains classes ClientGateway, AccountGateway, EmployeeGateway, ActivityGateway and AdminGateway

The package diagram of this application is described in the picture below.



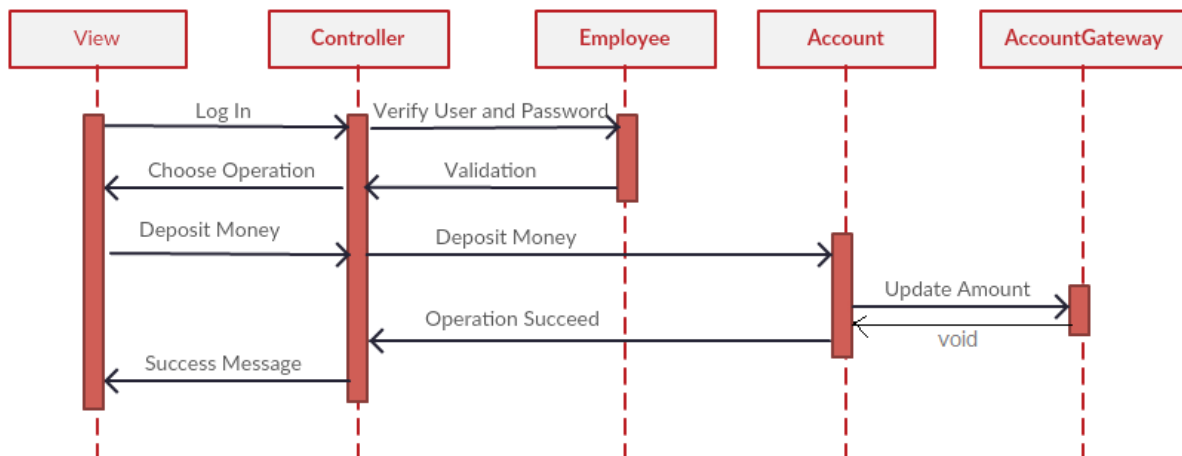
The deployment diagram is described in the following image.



## 4. UML Sequence Diagrams

A relevant scenario for this application is “Deposit money” that can be performed by the front desk employee only. The employee has to provide his username and password and then to complete the fields for the deposit operation with the account id and the amount of money to be stored.

The sequence diagram for this scenario is described by the picture below.



## 5. Class Design

### 5.1 Design Patterns Description

In this application I used the Model–View–Controller (MVC) design pattern. This is a software architectural pattern that divides a given application into three interconnected parts in order to separate internal representations of information from the ways that information is presented to and accepted from the user.

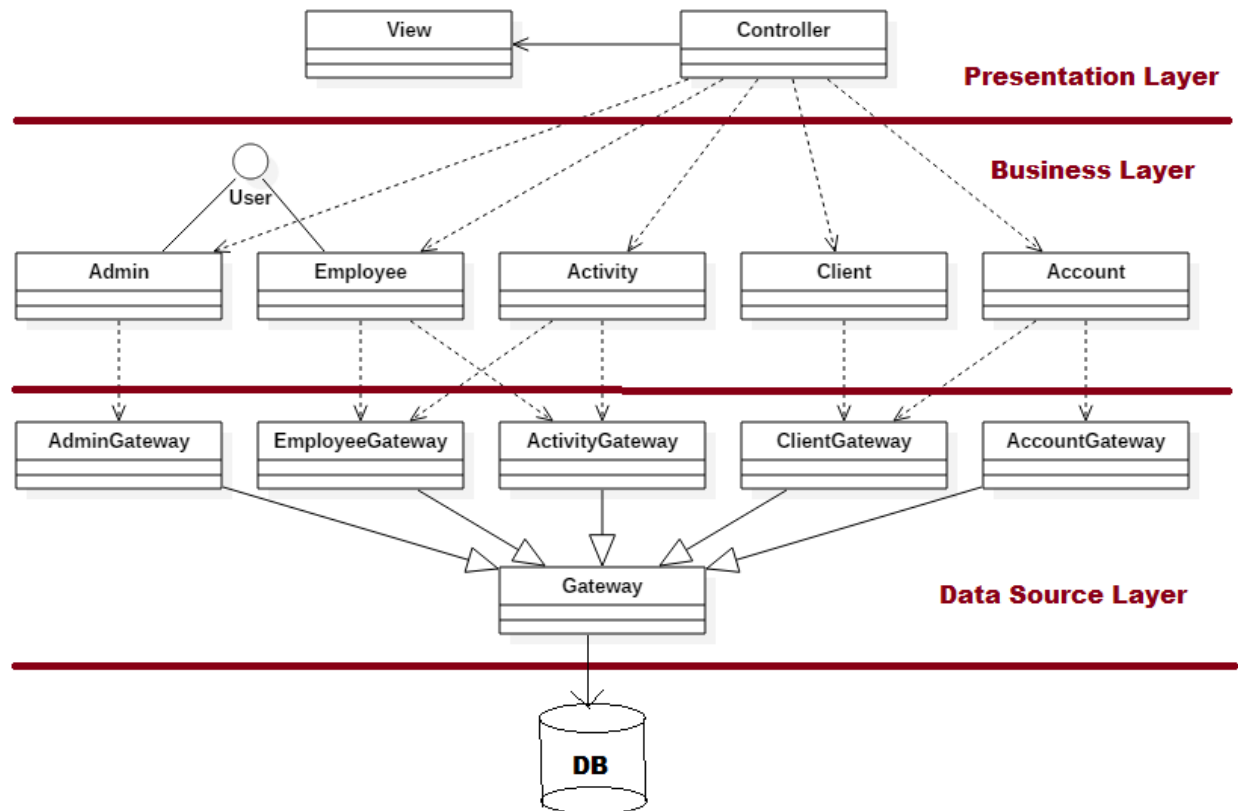
The model is the central component of the pattern. It expresses the application's behavior in terms of the problem domain, independent of the user interface. It directly manages the data, logic and rules of the application.

The view represents the user interface through which the system communicates with the user by input and output data. The controller accepts input and converts it to commands for the model or view.

### 5.2 UML Class Diagram

In our application the model is represented by all the classes from the business and data source layers. The view and the controller are both represented by only one class View and Controller respectively and they are situated in the presentation layer.

The UML class diagram of the application including the layers separation is described in the picture below.



Beside the classes that were mentioned in the sections before, View, Controller, Client, Account, Employee, Activity, Admin, ClientGateway, AccountGateway, EmployeeGateway, ActivityGateway and AdminGateway, there is an interface User implemented by the classes Admin and Employee by implementing the method logIn. There is also a class Gateway that includes all the information and the method required for database connection and it is inherited by all the classes from data source layer.

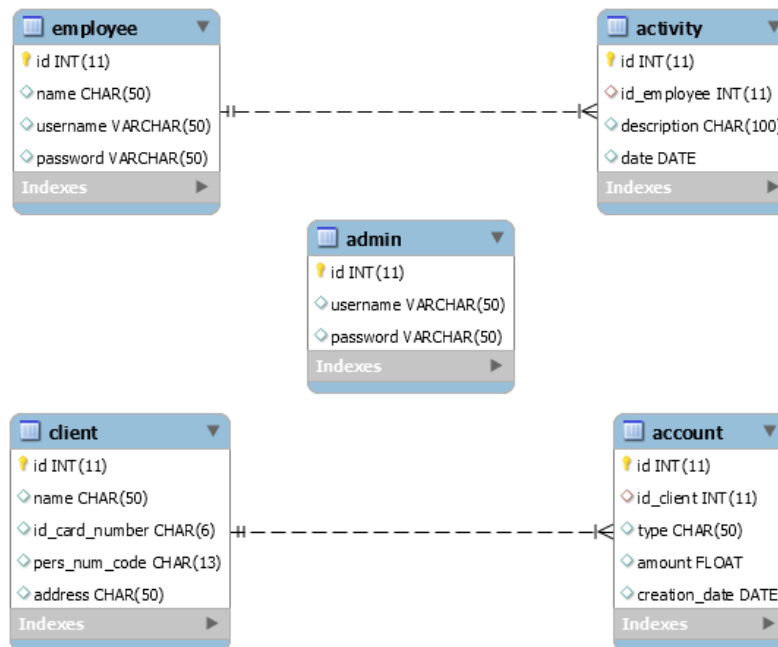
## 6. Data Model

As I mentioned before, the data is stored in a database. There are five table in the database containing information about the administrators, employees, clients, accounts and employees' activities as follows:

- Admin table includes: id, username and password
- Employee table includes: id, name, username and password
- Activity table includes: id, description, date and the foreign key that mentions the employee id as an employee can have multiple activities stored in the database
- Client table includes: id, name, address, identity card number and personal numerical code
- Account table includes: id, type, amount of money, date of creation and the foreign key that mentions the client id as a client can have multiple account

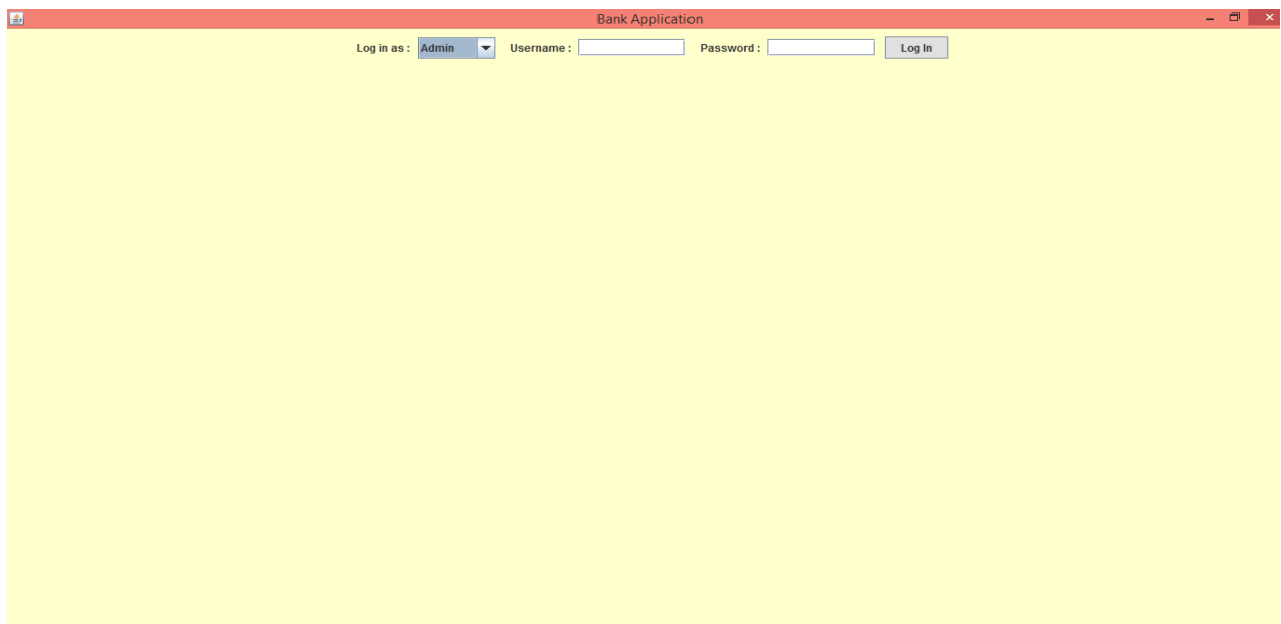
The database tables are described in the following picture.





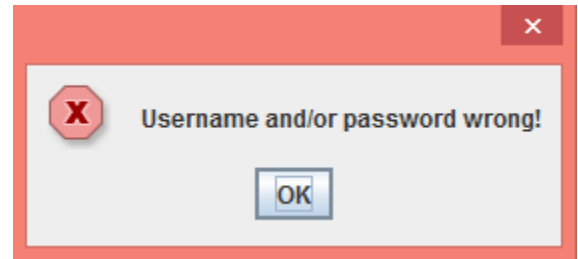
## 7. System Testing

When we open the application it will look like in the following image.

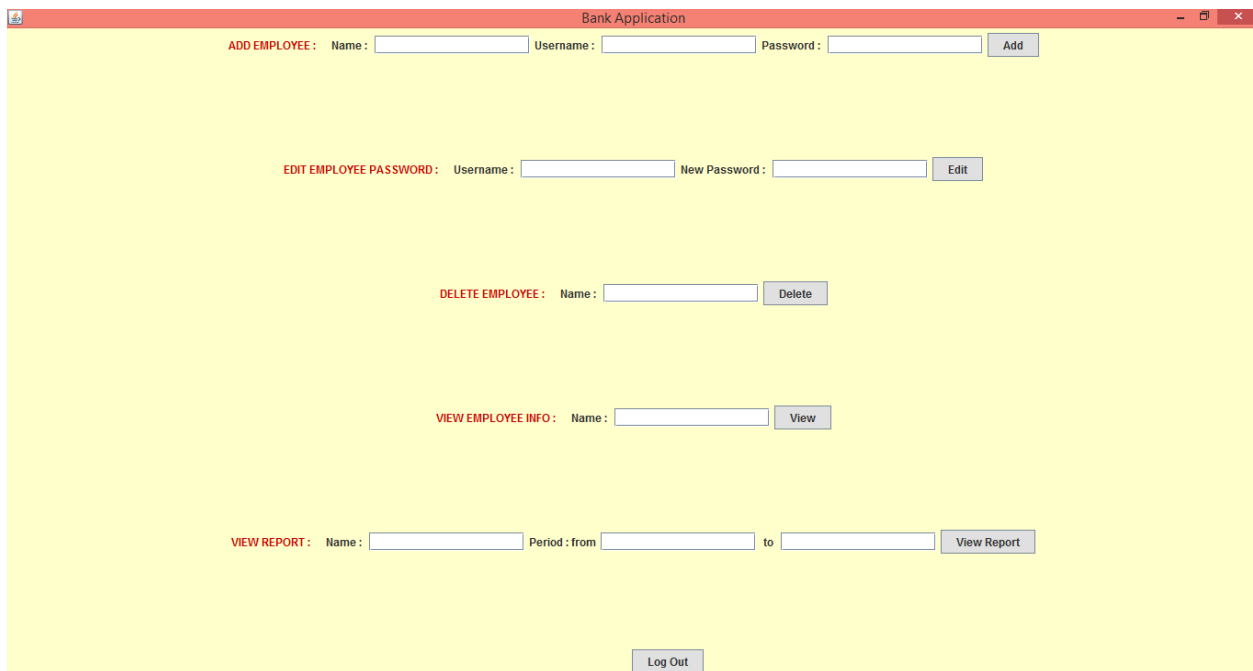


You can choose from the combo box to log in as an admin or as an employee. The admin that we will work with has the username “**admin1**” and the password “**1234**” and the employee has the username “**radu\_paul**” and the password “**1234**”. If you press the button before

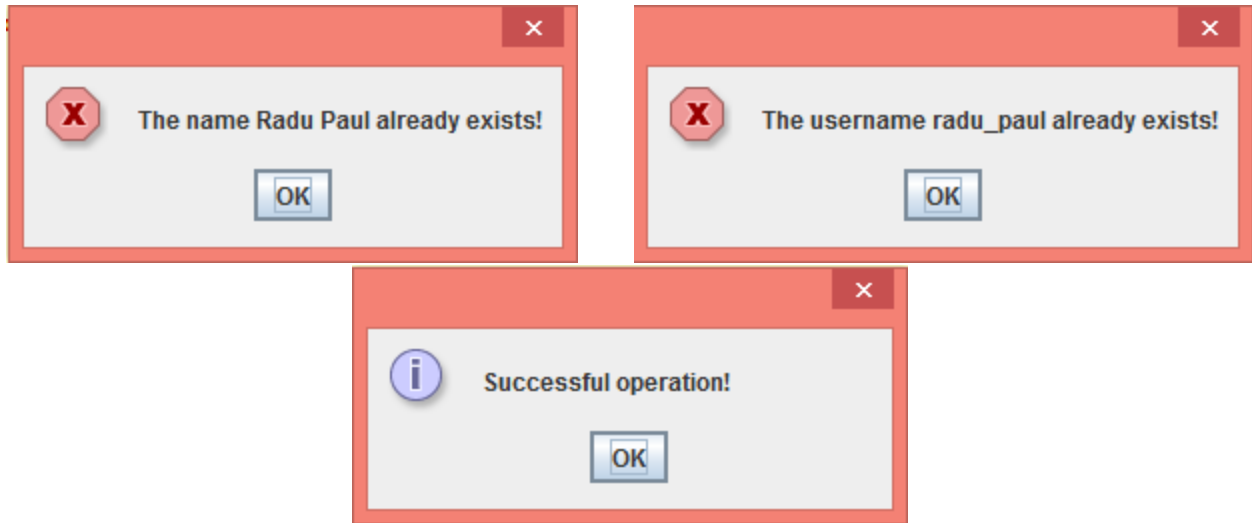
completing the fields or if the username or the password are not correct the will appear error messages like in the pictures below.



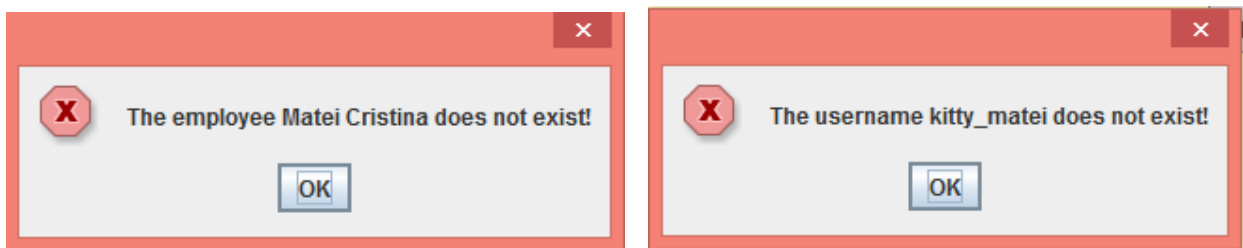
Firstly, we will log in as an admin. We choose the “Admin” field from the combo box, we write the corresponding username and password and then we press the button. There will appear the fields corresponding to the operations that an admin can perform like in the following image.

A screenshot of a web application titled "Bank Application". The interface has a yellow background and a red header bar. It contains several sections for user operations: "ADD EMPLOYEE:" with fields for Name, Username, and Password, and an "Add" button; "EDIT EMPLOYEE PASSWORD:" with fields for Username and New Password, and an "Edit" button; "DELETE EMPLOYEE:" with a Name field and a "Delete" button; "VIEW EMPLOYEE INFO:" with a Name field and a "View" button; and "VIEW REPORT:" with a Name field, a "Period: from" field, a "to" field, and a "View Report" button. At the bottom center is a "Log Out" button.

Exactly like at the log in operation, for all the other operations if we do not complete all the fields we will get an error message. For adding a new employee we have to provide all the information required. If the name or username introduces already exists in the database there will appear an error message and if not, we will get a message that informs us that the operation was successful.



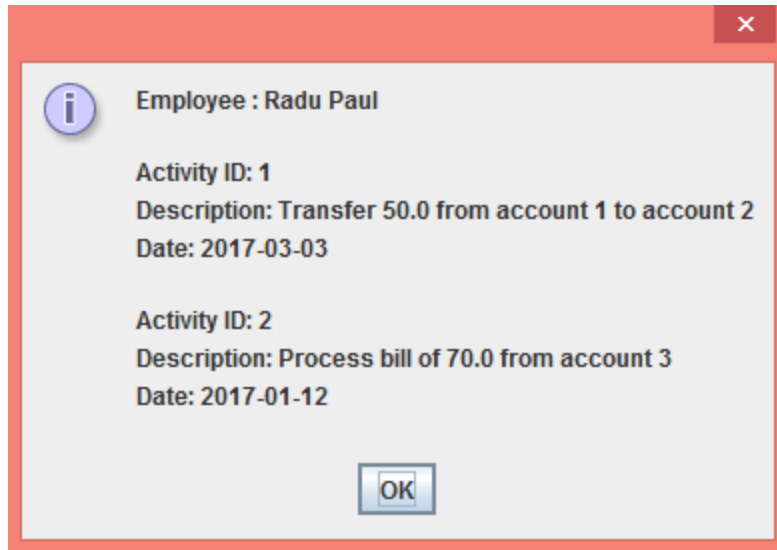
For all operations except the “view” ones, if all the information provided is alright there will appear this “Successful operation!” message. For the next operations existing for admin, edit, delete or view, we will get error messages if the name or username does not exist in the database.



For the “VIEW EMPLOYEE INFO” operation we will receive the information about a particular employee if he exists in the database. For the employee with the name “Radu Paul” we will get:



For the “VIEW REPORT” operation we will receive all the information about the activity of a particular employee in the specified period of time. Dates have to be introduced like “yyyy-mm-dd” or else we will not get the information. For the employee “Radu Paul” and the period from “2017-01-01” to “2017-03-29” we will get:



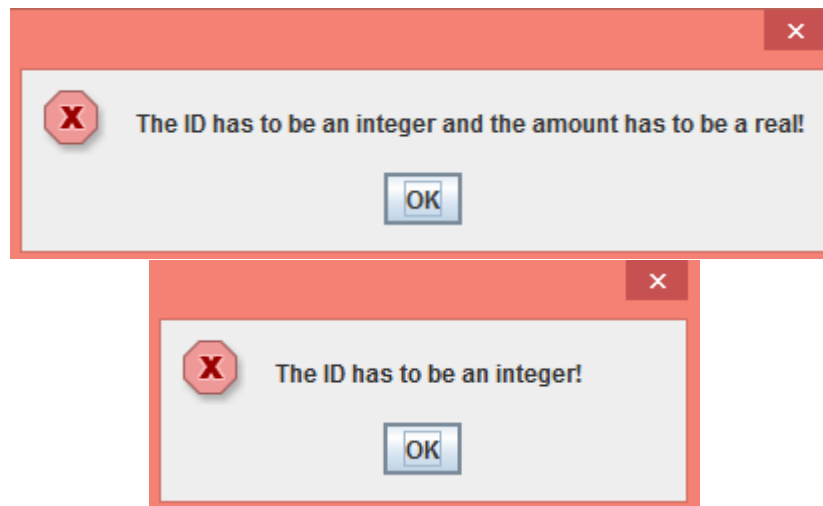
We can log out by pressing “Log Out” button and we will get back to the start page. Now we will select the “Employee” field, provide the corresponding username and password and there will appear the fields corresponding to the operations that an employee can perform like in the following image.

Exactly like at the operations of admin, there will appear an error message if the fields corresponding to the chosen operation are not completed and a “Successful operation!” message if the information provided is alright. Also, there will appear an error message for adding a new client if there already exists the client with the name provided and for the other operations if the names or ids provided do not exists in the database. The following pictures describe the error messages for adding a new client with the name “Matei Cristina” that already exists in the

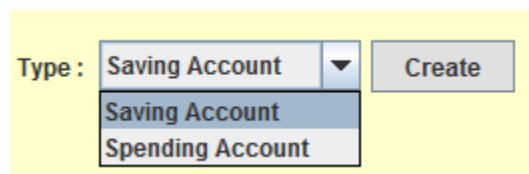
datadase, for deleting the account with the id 12 that does not exists and for editing the address of the client “Deac Dan” that does not exist.



For the operation of deposit, transfer money and process bills where we have to introduce account ids and amount of money we have to introduce integers and reals respectively or else we will receive some error messages.

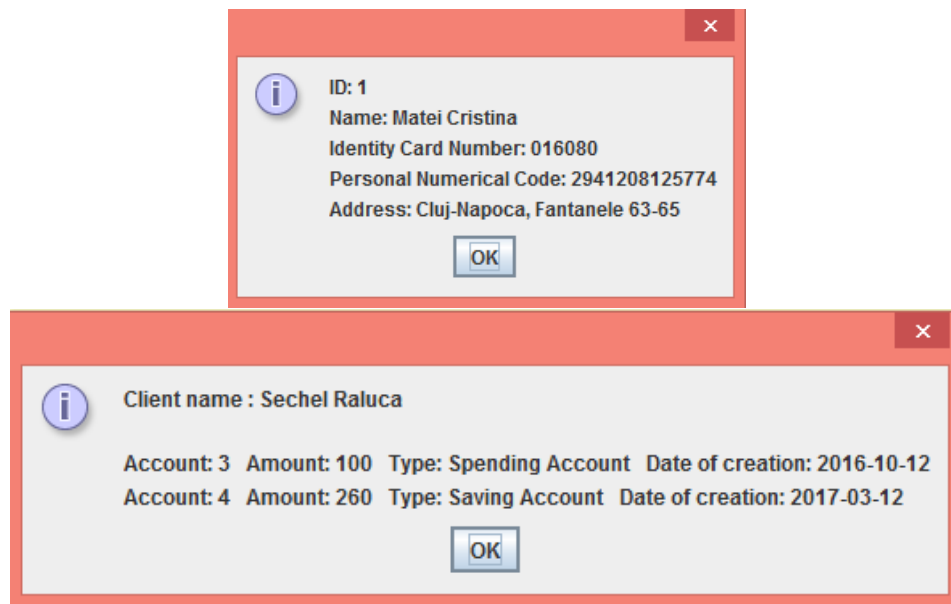


For creating a new account we have to provide the name of the client and to choose from the combo box the field “Saving Account” or “Spending Account”.



The operation “VIEW CLIENT INFO” show us all the information about a particular

client and the operation “VIEW ACCOUNTS” show us all the accounts of a client with all their information. The images below describe the information of the client “Matei Cristina” and the account of the client “Sechel Raluca”.



## 8. Bibliography

- Martin Fowler et. al, Patterns of Enterprise Application Architecture, Addison Wesley, 2003
- Mark Richards, Software Architecture Patterns, O'Reilly, 2015