Bank Application

Student: Deac Dan

**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:

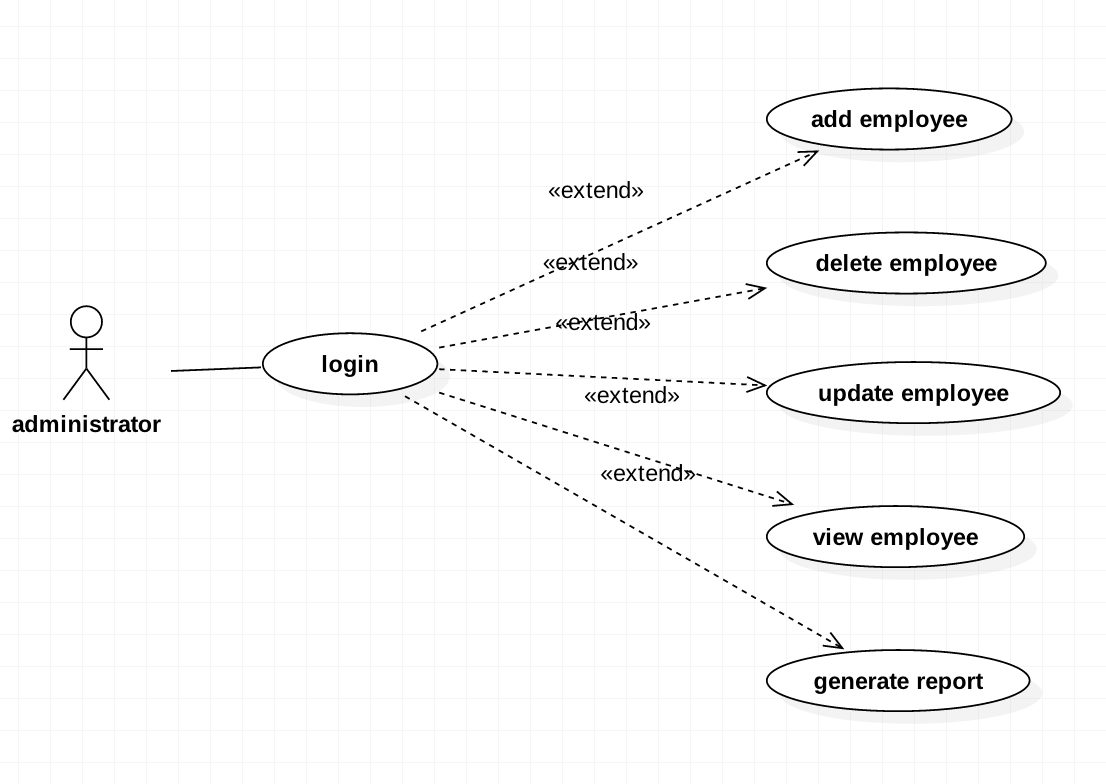
* CRUD on employees’ information.
* Generate reports for a particular period containing the activities performed by an employee.

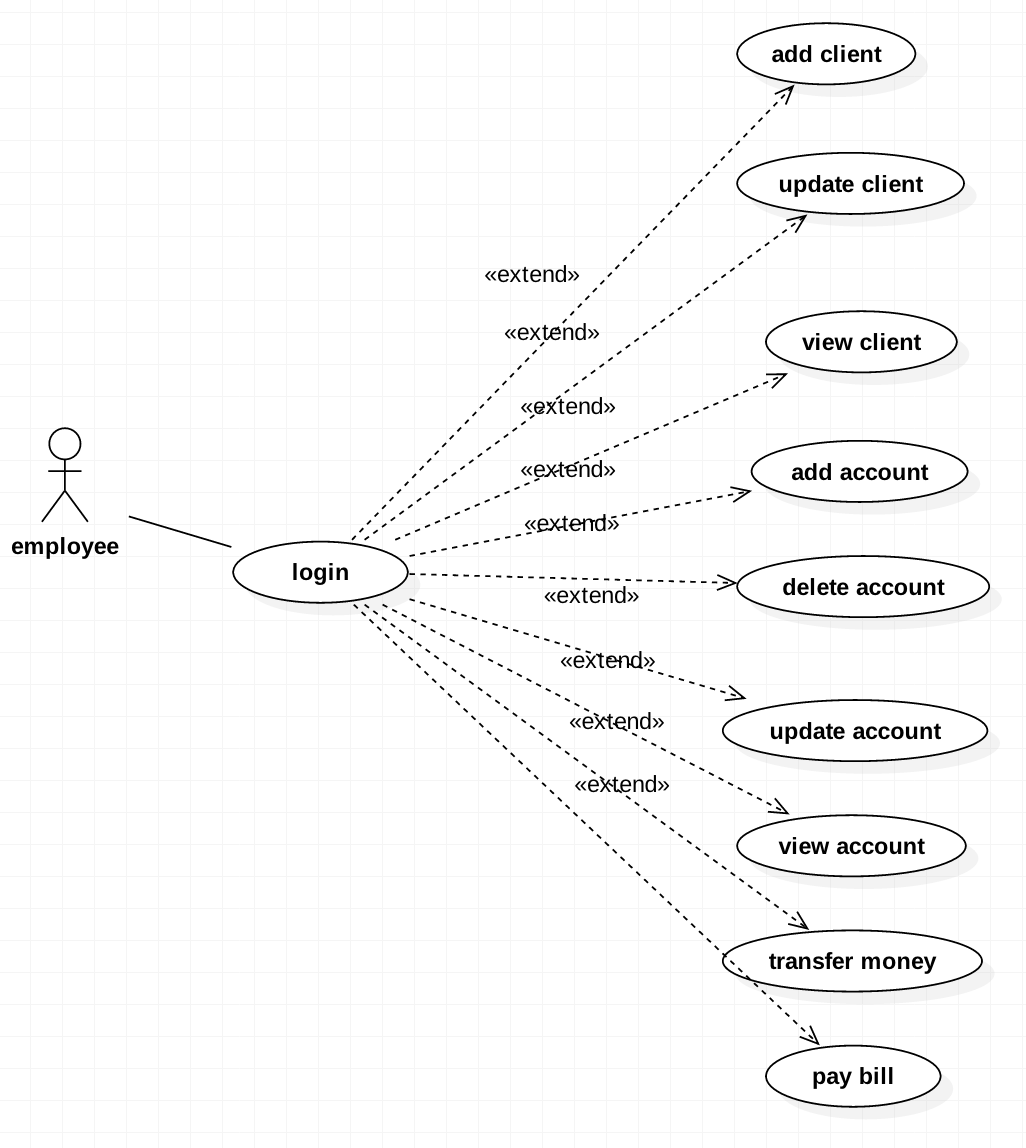
# Non-functional Requirements

My application respects the following non-functional requirements:

* Security: only the authorized users can use the app
* Privacy: only the administrator has access to the employee's personal information
* Usability: the application has a GUI so it's easy to use
* all the information is stored in a structured database
* all the inputs of the application are validated against invalid data before submitting the data and saving it in the database.

2. Use-Case Model





Use case: generate employee report

Level: user-goal level

Primary actor: administrator

Main success scenario:

* the administrator types his username and his password and presses LOGIN button
* will introduce the username of the employee for which he will generate the report
* will introduce the period of time between which the report will be generated
* will press the GENERATE button
* system will match the input username with an username that exists in the database and will return the report

Extensions: the username of the employee that the administrator typed does not exists in the database: a message will be showed

3. System Architectural Design

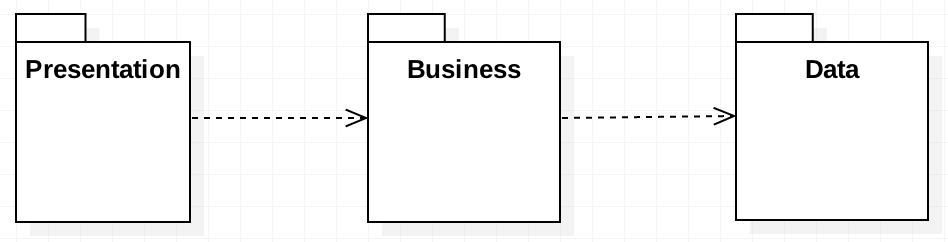
**3.1 Architectural Pattern Description**

I choose Layered Design Pattern as an architectural pattern. The main feature of this pattern is that the data is structured into three layers: presentation layer, business layer and the data layer. The presentation layer contains the GUI and it only gets the input data from the GUI and passes to the business layer and gets the data from the business layer and displays it on the interface. The business layer is the link layer between presentation and data layer. It gets the data from the presentation layer, transforms and passes it to the data layer and vice versa. The data layer is the layer that is connected to the database.

**3.2 Diagrams**

Conceptual diagram:

Package diagram:



Each package is associated to a layer:

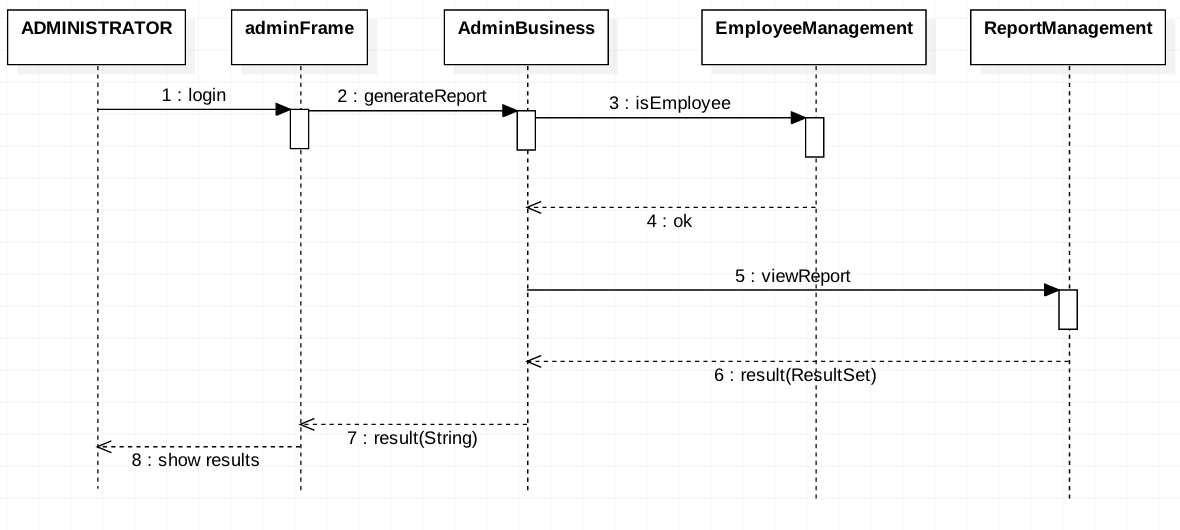
* Presentation package contains the GUI class
* Business package contains classes Account, Client, User, AdminBusiness and EmployeeBusiness
* Data package contains classes Connect, EmployeeManagement, AccountManagement, ClientManagement and ReportManagement

Deployment diagram:



4. UML Sequence Diagrams

Sequence diagram for generate report scenario:



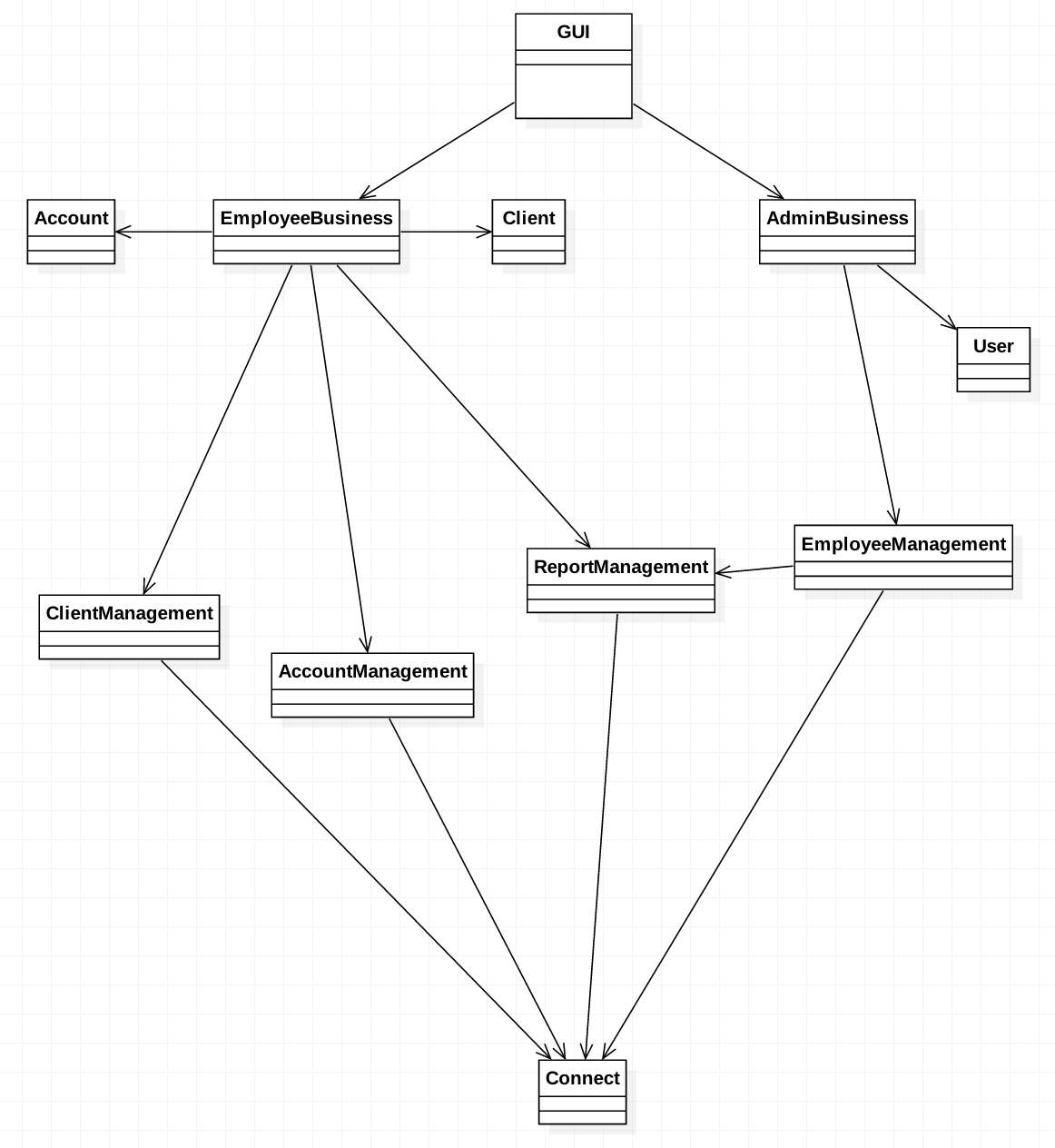
5. Class Design

**5.1 Design Patterns Description**

As a data source pure pattern I've chosen Table Data Gateway, so I've created a separate class for each table. This classes are taking the information from the business layer and are executing queries on the database. They return the result of the queries back to the business layer classes.

As a domain logic pattern I've chosen Transaction Script. So, the purpose of the classes that are in business layer is to take the information from the presentation layer, to transform and to send it forward to the data source layer. Here, all the inputs of the application are validated against invalid data before submitting the data and saving it in the database.

**5.2 UML Class Diagram**

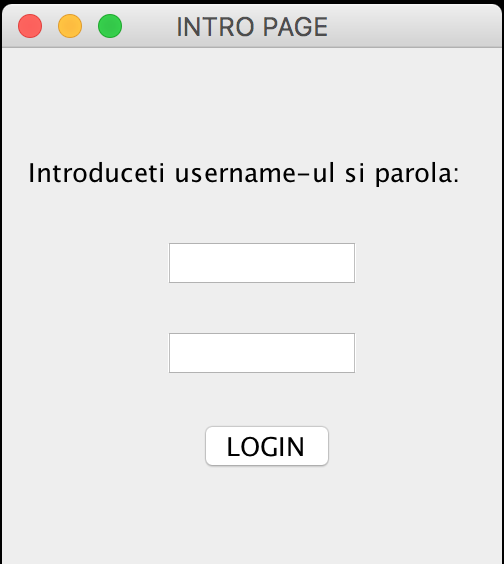


6. Data Model

In the database I have created four tables: clients, accounts, employees and report. Clients table contain information as id(PK), name, card number, personal code, address and e-mail. Accounts table contain information as id(PK), money, creation date, account type and client id(FK). The employees table contain information as username(PK) and password. The report table contain information as employee(FK), description and date.

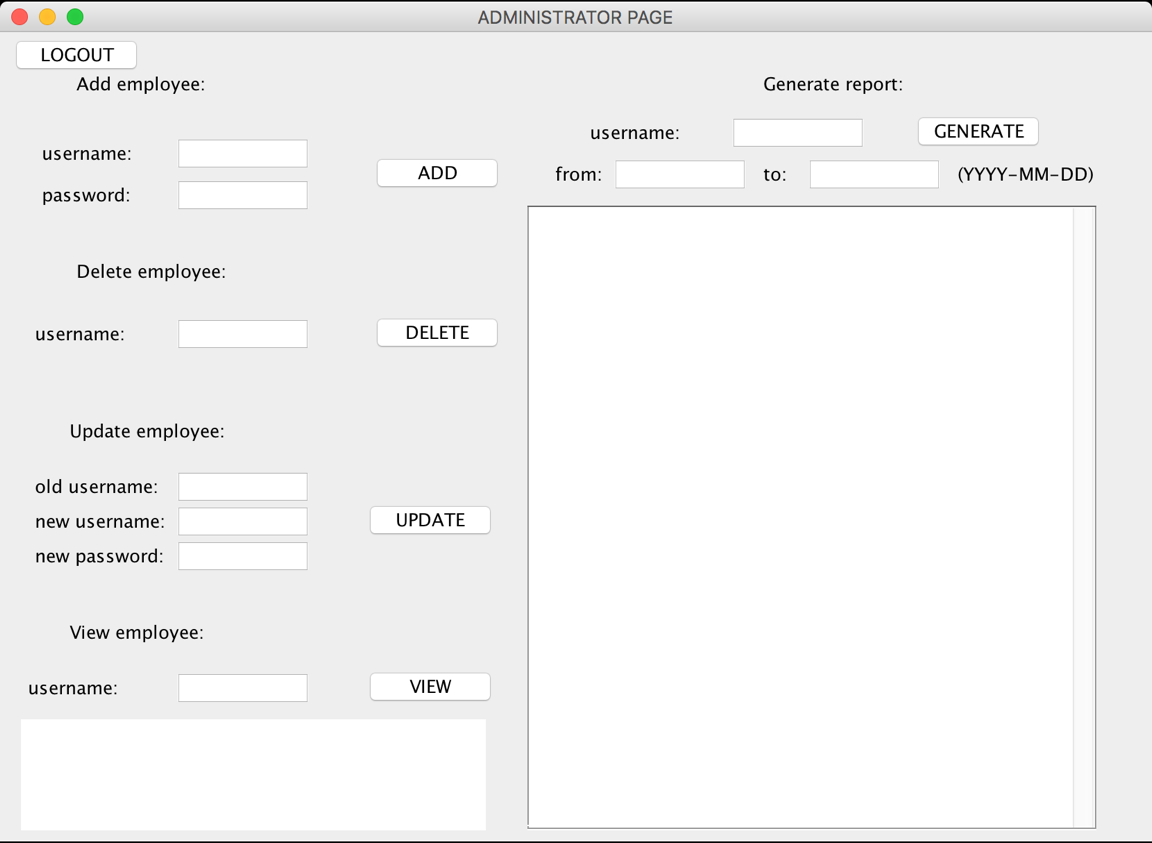
7. System Testing

After the user runs the .jar file:

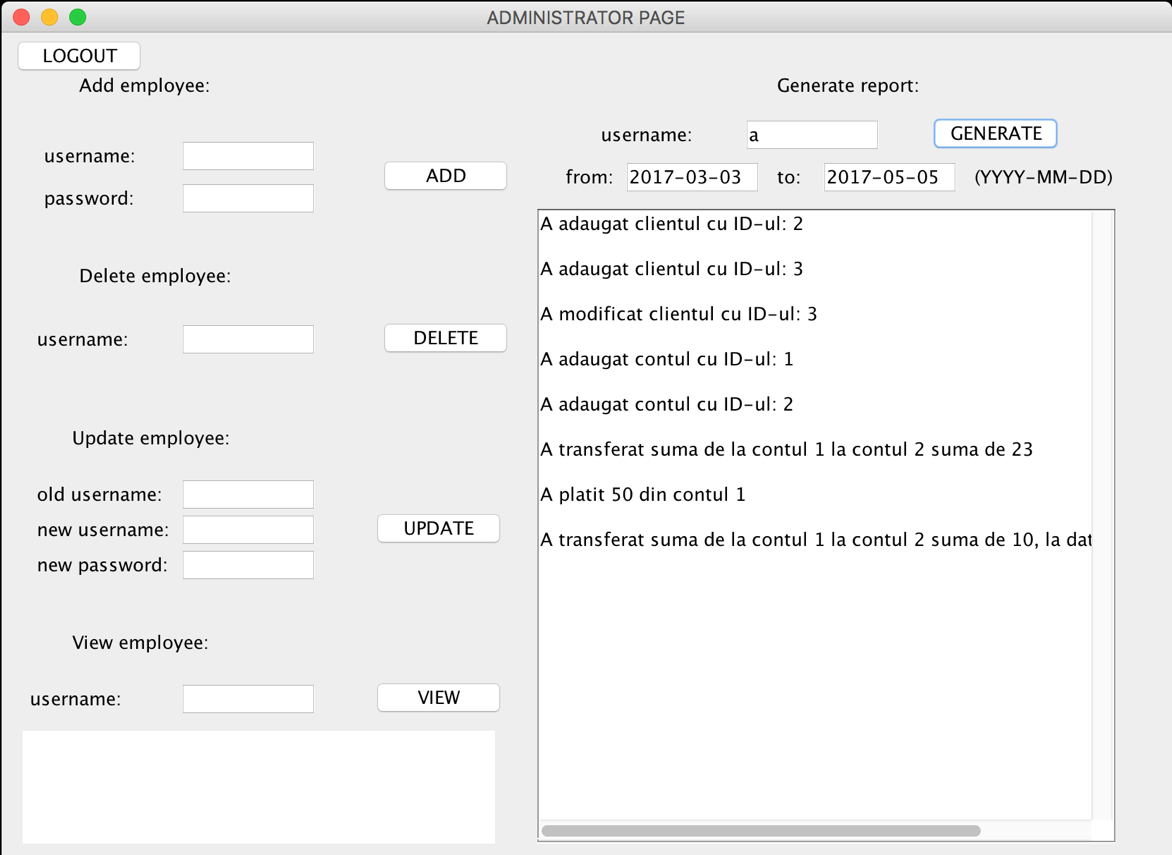


The admin or the employee introduces his username and his password. If the system matches the username and the password that were typed with the admin information or with an entry in the employees table from the database, it will show another window according to the type of user. Otherwise an error message will be showed.

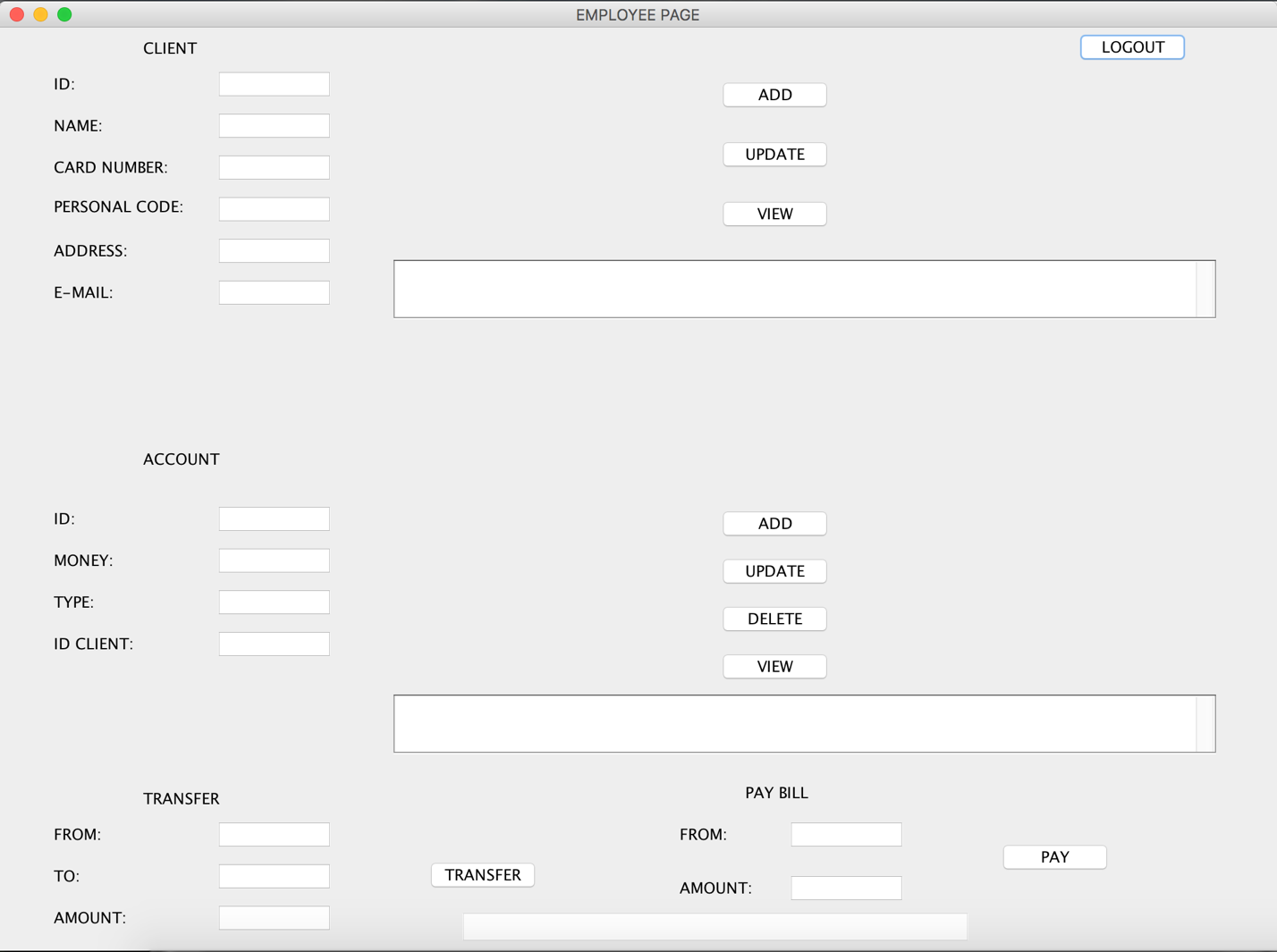
The administrator page:



On this window the administrator can perform various tasks like adding, deleting, updating and viewing an employee. The administrator introduces the necessary information for every operation and presses the associated button. Also, the administrator can generate reports for the employees by typing their username and the period of time.



The employee page:



Here, the employee can perform different operations on clients and accounts. He introduces the necessary information in text fields and presses the associated button for every operation. Also, he could transfer money between accounts and pay bills. The only condition that is imposed is that the "from" account needs to be a spending-type account. Also, if the money left on the account are less than the introduced amount an error message will be showed.

8. Bibliography

1. Patterns of Enterprise Application Architecture, By Martin Fowler, David Rice, Matthew Foemmel, Edward Hieatt, Robert Mee, Randy Stafford
2. Software Architecture Patterns, By Mark Richards
3. <https://www.codeproject.com/Articles/654670/Layered-Application-Design-Pattern>
4. <https://dev.mysql.com/doc/refman/5.7/en/tutorial.html>