Assignment 1

Analysis and Design Document

Student: Nimigean Mihnea

**Group: 30237**

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.

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.

# Functional Requirements

The system should permit regular users to add new clients and update or view client information for existing clients. Regular users also have permission to execute CRUD statements over the client accounts. Transfer between accounts and processing of utilities bills are also an operation permitted to the regular user. The administrator of the application is allowed to execute CRUD statements over employees and generate reports about a specific employee.

# Non-functional Requirements

The non-functional requirements of the system include the usage of a layered architectural pattern and ensuring all the data is being verified before being introduced in the database.

2. Use-Case Model

Use case: ***Add new account.***

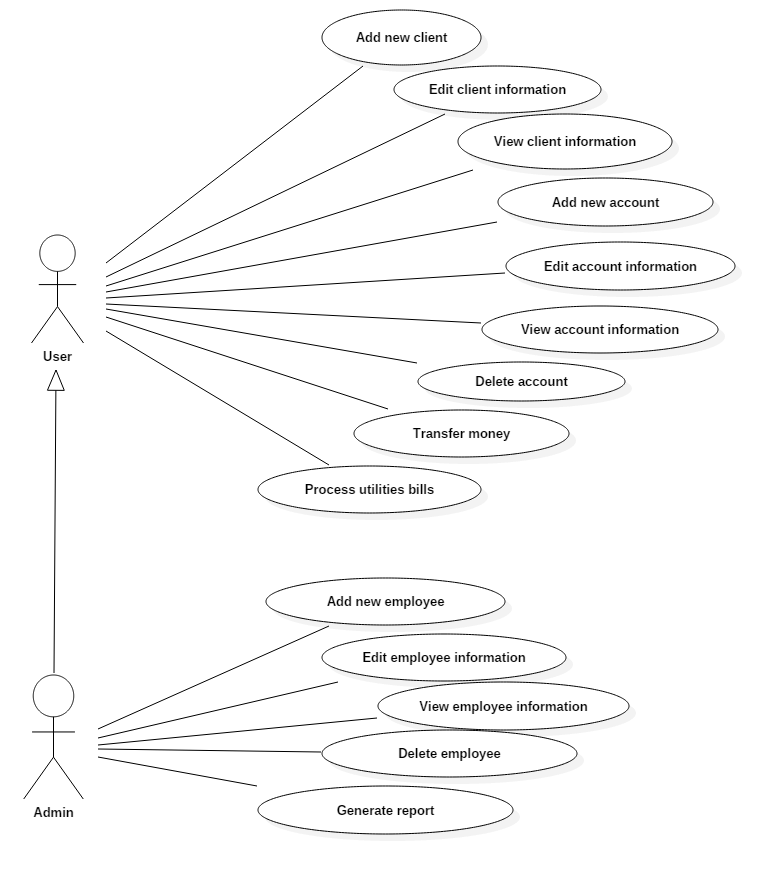
Level: user-goal level

Primary actor: Regular employee (User)

Main success scenario:

|  |  |
| --- | --- |
| Action | System Response |
| 1. Access the web application | The login page is displayed |
| 1. Log in | The user is directed to the home page |
| 1. Click on Add and select Account | Display the form to be completed |
| 1. Enter the account type (debit/credit), account balance, and the client’s CNP. | The system checks if the client exists and if it’s possible to add the new account. If yes, the operation redirects to /home |

Extensions: The client doesn’t exist, so the account can’t be added.

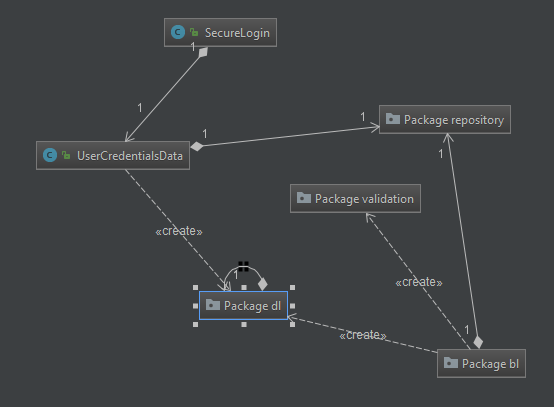


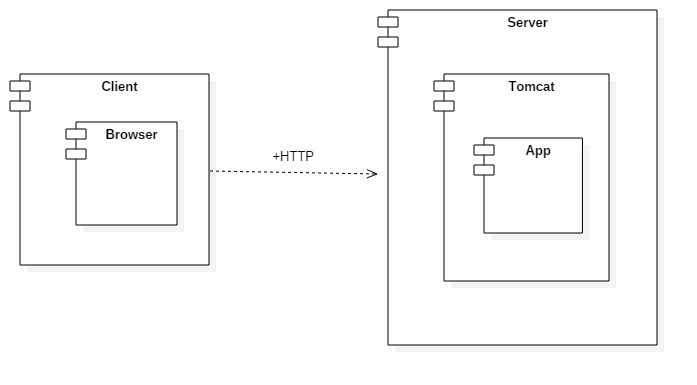
3. System Architectural Design

**3.1 Architectural Pattern Description**

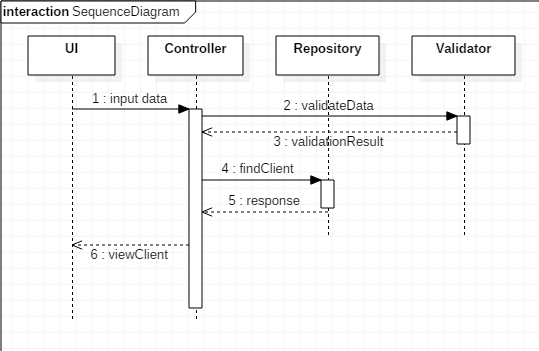
The system uses the layers architectural design pattern. It divides the application in three layers: the presentation layer, which is used to handle the front end of the application, the business layer which handles all the logic of the system (input handling, validations) and the data layer which handles the connection and access to the database.

**3.2 Diagrams**

**

**

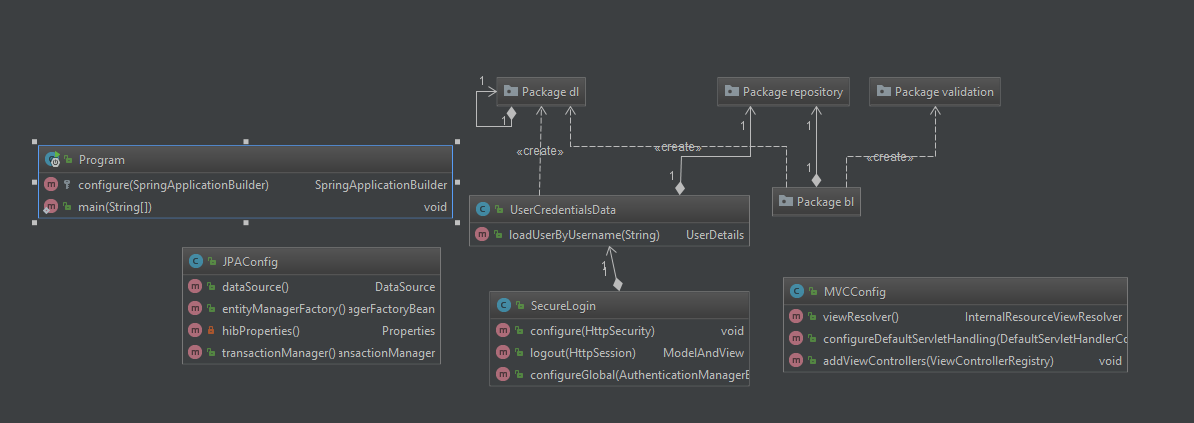
4. UML Sequence Diagrams

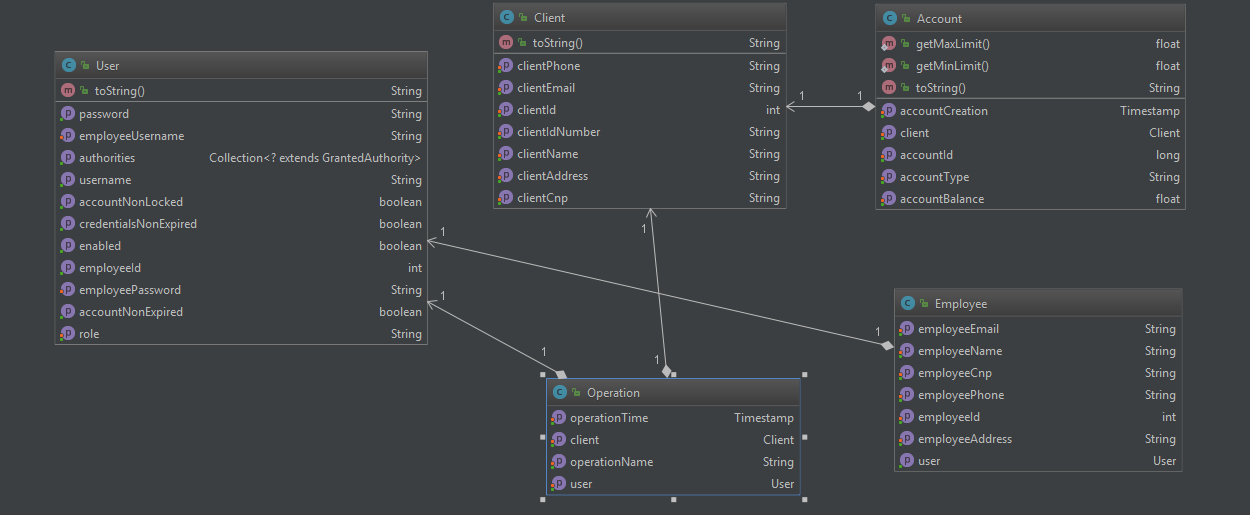


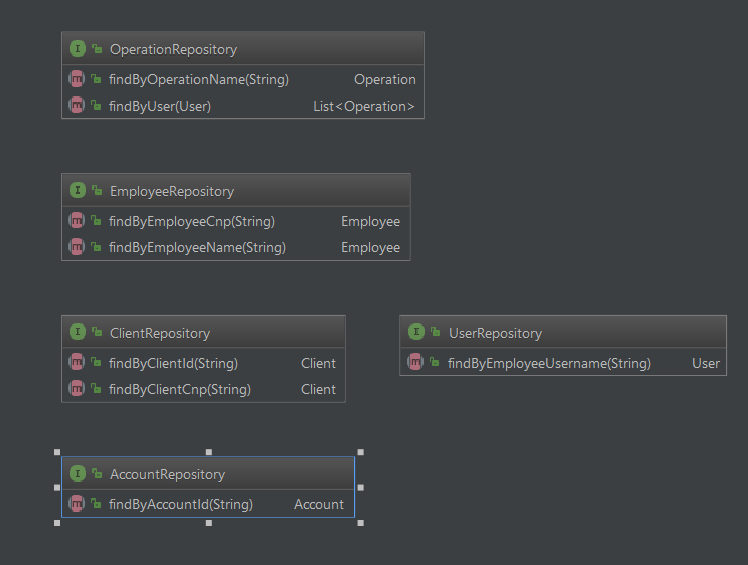
5. Class Design

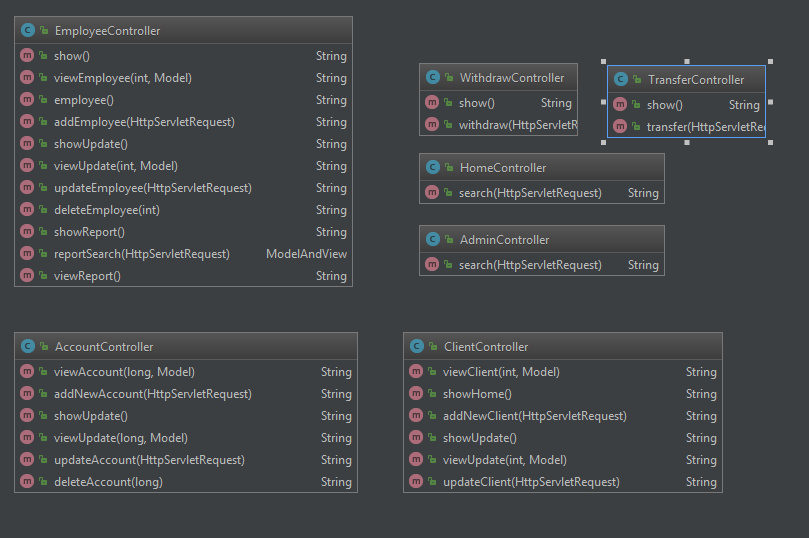
**5.1 Design Patterns Description**

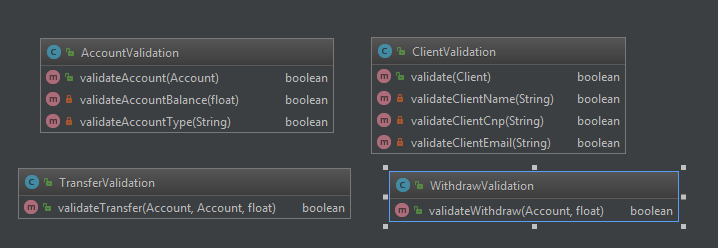
**5.2 UML Class Diagram**

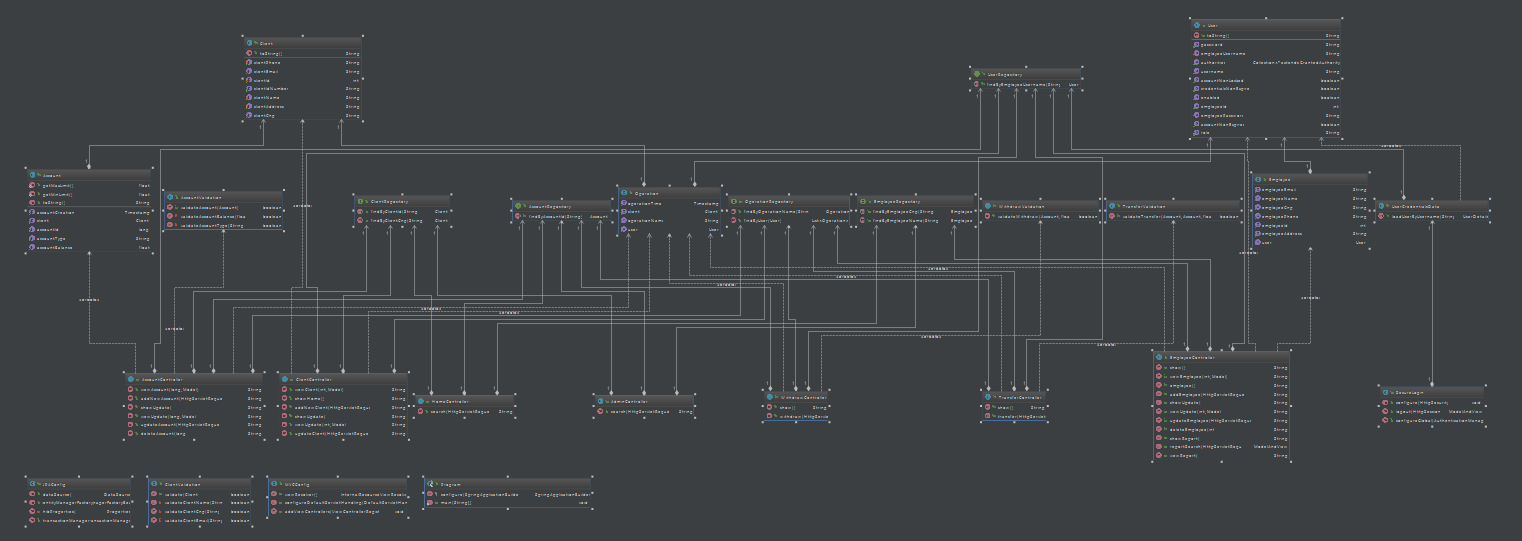
****











6. Data Model

The data model used for the application was derived from the problem specification. The classes created are the following: Client, Employee, User, Account, Operation. A client can have multiple accounts, while an account can have only one owner. There is an one to one relation between the Employee and User (a Employee must have only one User). A user can have one or two roles (User and/or Admin).

7. System Testing

The application was tested by providing input from the web interface. Due to the fact that mocking an Entity Manager used by Spring Data Repositories is very difficult, unit testing the operations could not be done without interfering with the database. The connection to the database was tested in the early phases of development.

8. Bibliography

1. Lectures M. Dansoreanu 2017 UTCN
2. Patterns of enterprise application architecture – Martin Fowler
3. Various web resources about Spring and JSP