Assignment 1

Analysis and Design Document

Student: Modreanu Eduard

**Group: 30431**

Table of Contents

1. Requirements Analysis

1.1 Assignment Specification

1.2 Functional Requirements

1.3 Non-functional Requirements

2. Use-Case Model

3. System Architectural Design

4. UML Sequence Diagrams

5. Class Design

6. Data Model

7. System Testing

8. Bibliography

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 data will be stored in a database. Use the Layers architectural pattern to organize your application.

# Functional Requirements

* Identify user, according to the username and password provided, either as a regular user or as an administrator, and grant appropriate privileges
* Allow regular user to add/update/view client information
* Allow regular user to create/update/delete/view client account
* Allow regular user to transfer money between accounts
* Allow regular user to process utility bills
* Allow administrator user to create/update/delete/view employee information
* Allow administrator user to generate reports on employee activity for a particular period of time

# Non-functional Requirements

* Use the Layers architectural pattern to organize your application.
* Verify inputs
* No negative inputs for money operations

2. Use-Case Model

Assign1UseCase.png

Add Client: adds a new client into the list of clients

Level: user-goal level

Primary Actor: Regular User

Main success scenario:

* Successfully log in as regular user by providing a username and a password
* Insert required client data
* Press the “Add” button
* The new client is inserted in the database, and is shown on screen

Extension:

* The user did not provide the correct username or password, thus was not logged into the system

3. System Architectural Design

**3.1 Architectural Pattern Description**

The architectural pattern chosen for this application is the Layers architectural pattern. This pattern splits the application into three main layers:

* Presentation Layer, in which we have the elements visible to the user, and the parts with which the user can interact
* Business Logic 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, depending on commands received from the presentation
* Data Source Layer, which usually handles the communication with databases.

**3.2 Diagrams**

**Package Diagram**



**Component Diagram**

**Deployment Diagram**



4. UML Sequence Diagrams



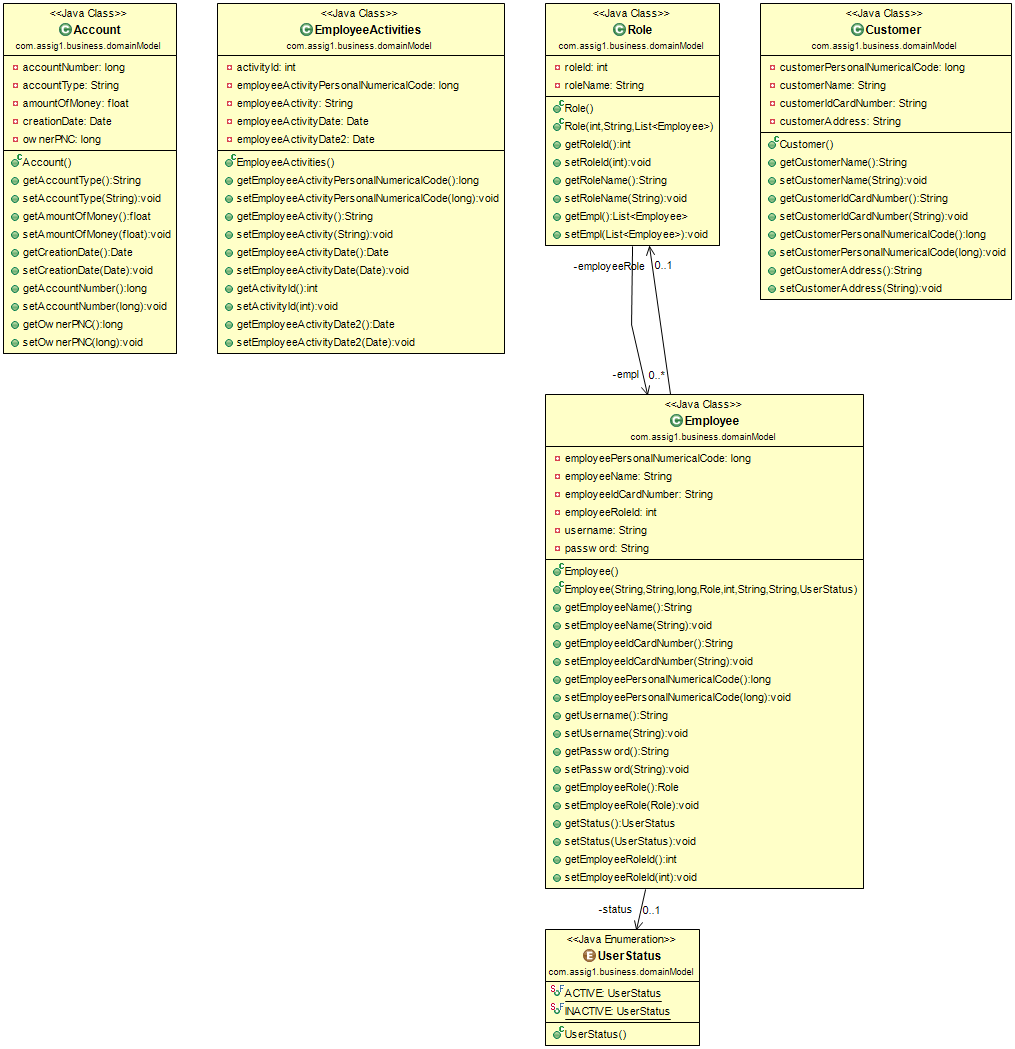
5. Class Design

**5.1 Design Patterns Description**

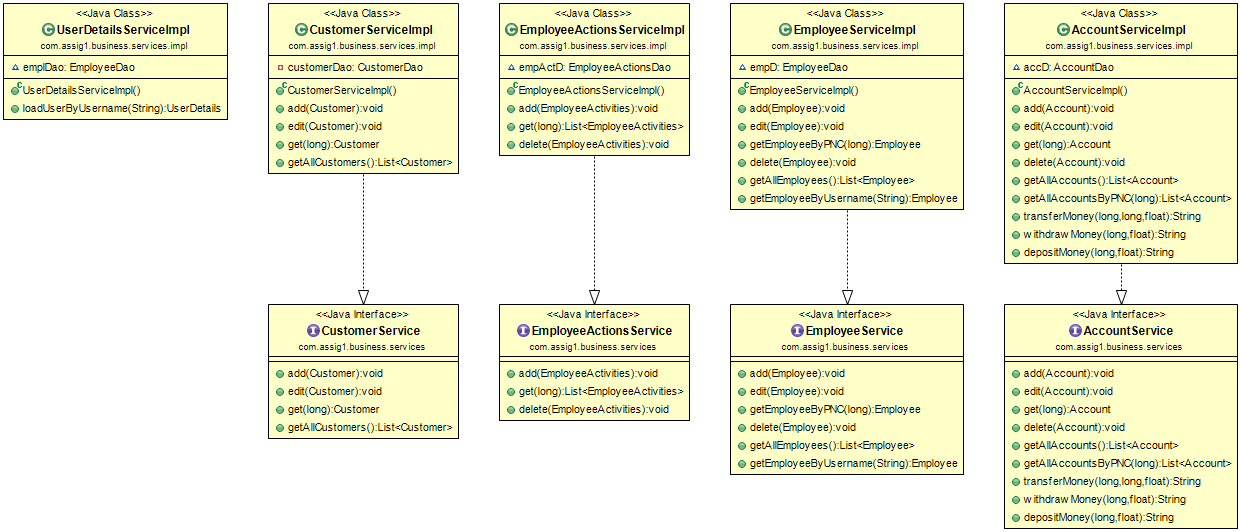
In this application, i used different design patterns for each of the layers specified above:

* Presentation Layer: for the presentation layer, the MVC design pattern was used, with the help of Spring MVC, and also Spring Security, for the login.
* Business Layer: for this layer, I used the domain model, in which are defined the entities needed, and also provided services which use the data source layer
* Data Source Layer: was implemented with Hibernate 5, using Data Access Objects to perform operations on the database

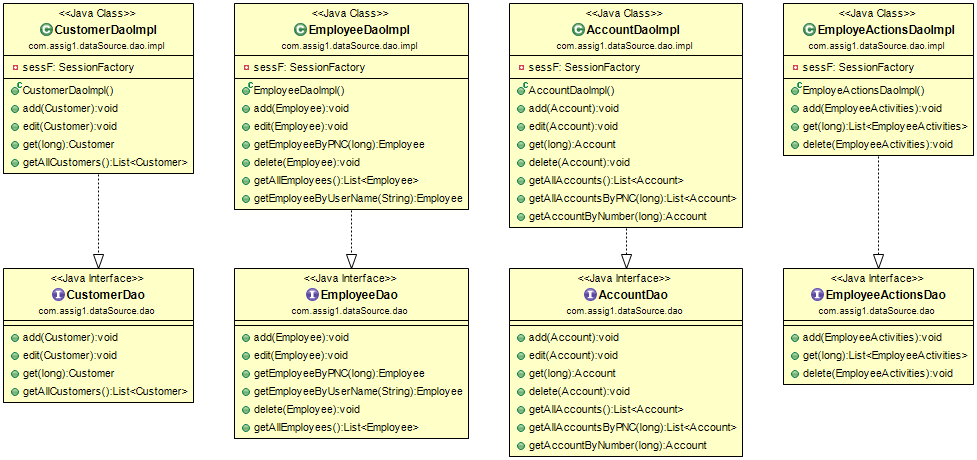
**5.2 UML Class Diagram**



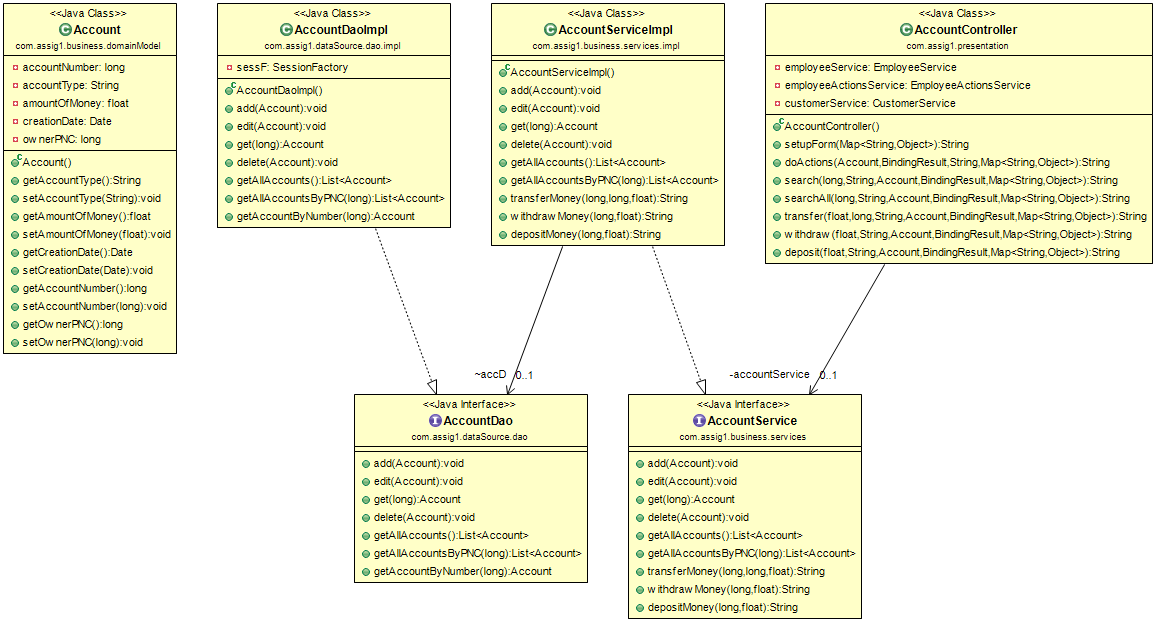
Entities Diagram



Service Diagram

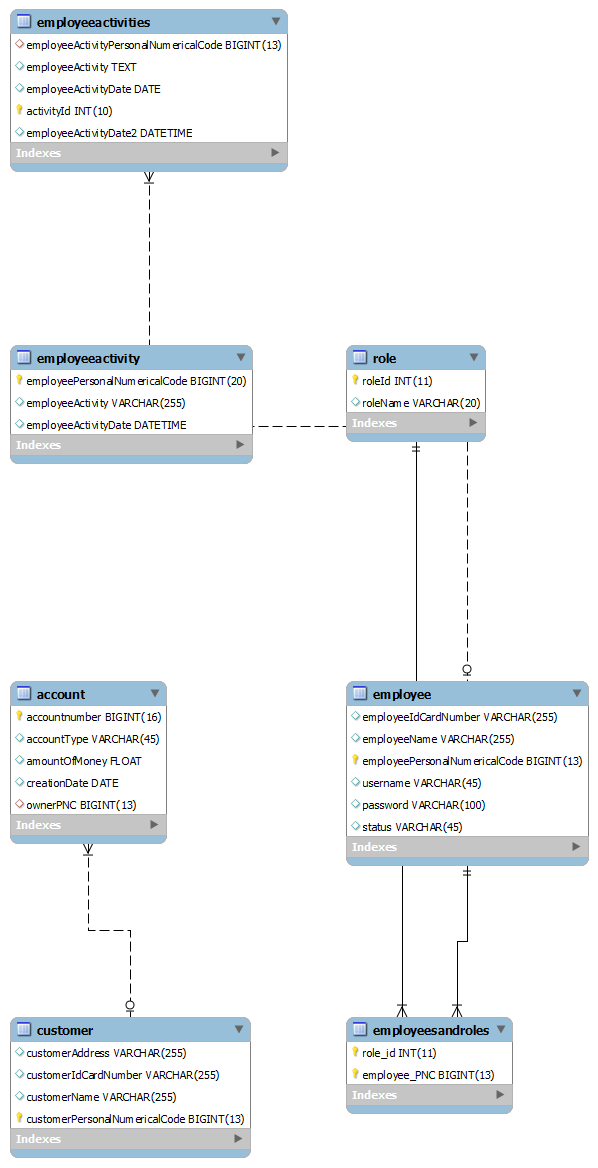


Dao Diagram



Linking between layers

6. Data Model



7. System Testing

Testing was done by hand during development

8. Bibliography

[www.stackoverflow.com](http://www.stackoverflow.com)

[www.tutorialspoint.com/](http://www.tutorialspoint.com/)

[www.javatpoint.com](http://www.javatpoint.com)