<Assignment 2>

Analysis and Design Document

Student: Campean Bogdan Mihai

**Group: 30231**

Table of Content

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 Chira2018

# 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:

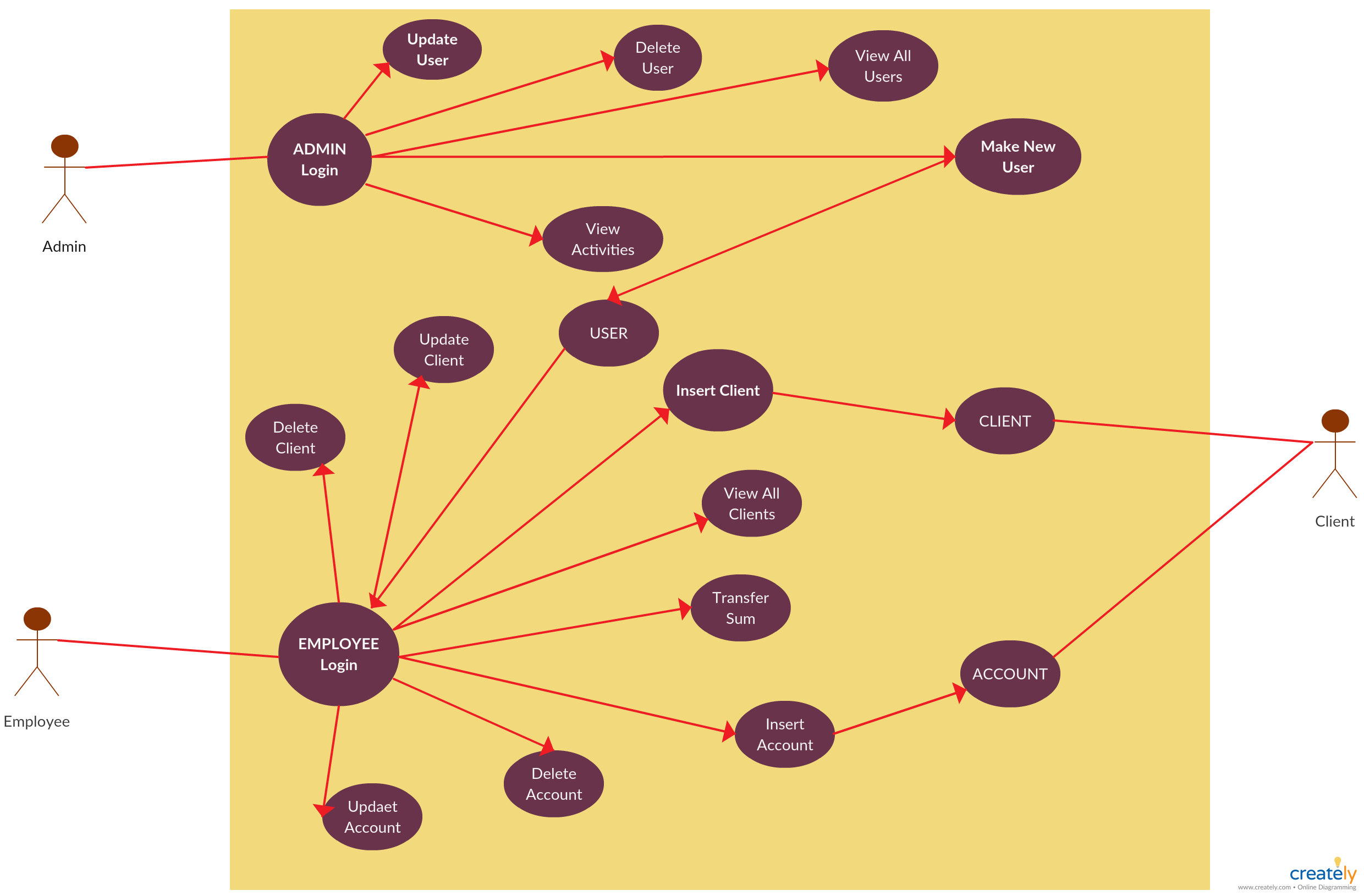
- CRUD on employees’ information.

- Generate reports for a particular period containing the activities performed by an employee.

# Non-functional Requirements

Pentru a folosi aplicatia, fiecare utilizator (administrator/angajat) trebuie sa se autentifice in aplicatie cu username-ul si parola proprie. In urmatorul pas se va deschide fereastra corespunzatoare titlului pe care il are acel username: administrator sau angajat.

2. Use-Case Model



*Use case: Crearea unui user nou pentru angajat.*

*Level: user-goal level*

*Primary actor: Admin*

*Main success scenario:*

1. Administratotul introduce un nume de utilizator si o parola in campurile destinate si va bifa sau va lasa nebifata casuta in care se specifica daca se creeaza un administrator sau un angajat.
2. Dupa inserarea acestor date, administratorul apasa pe butonul “Save User” si va aparea un mesaj de success in cazul in care utilizatorul a fost salvat cu success. In caz contrar, atunci cand campurile nu sunt valide, se va afisa un mesaj de eroare in care se specifica eroarea.
3. Angajatul face acum parte din baza de date si va putea utiliza aplicatia cu numele si parola sa.

*Extensions: -*

3. System Architectural Design

**3.1 Architectural Pattern Description**

Pentru acest proiect am folosit sablonul arhitectural de business. Acesta este impartit pe 3 sau mai multe niveluri, insa pentru cerintele date au fost de ajuns doar 3 dintre ele: nivelul legaturii bazei de date (repository), nivelul logicii fiecarui apel (service) si nivelul interfetei de unde se introduce datele care urmeaza sa fie procesate (presentation). Din fiecare nivel se poate accesa doar un nivel mai jos: presentation > service > repository.

**3.2 Diagrams**

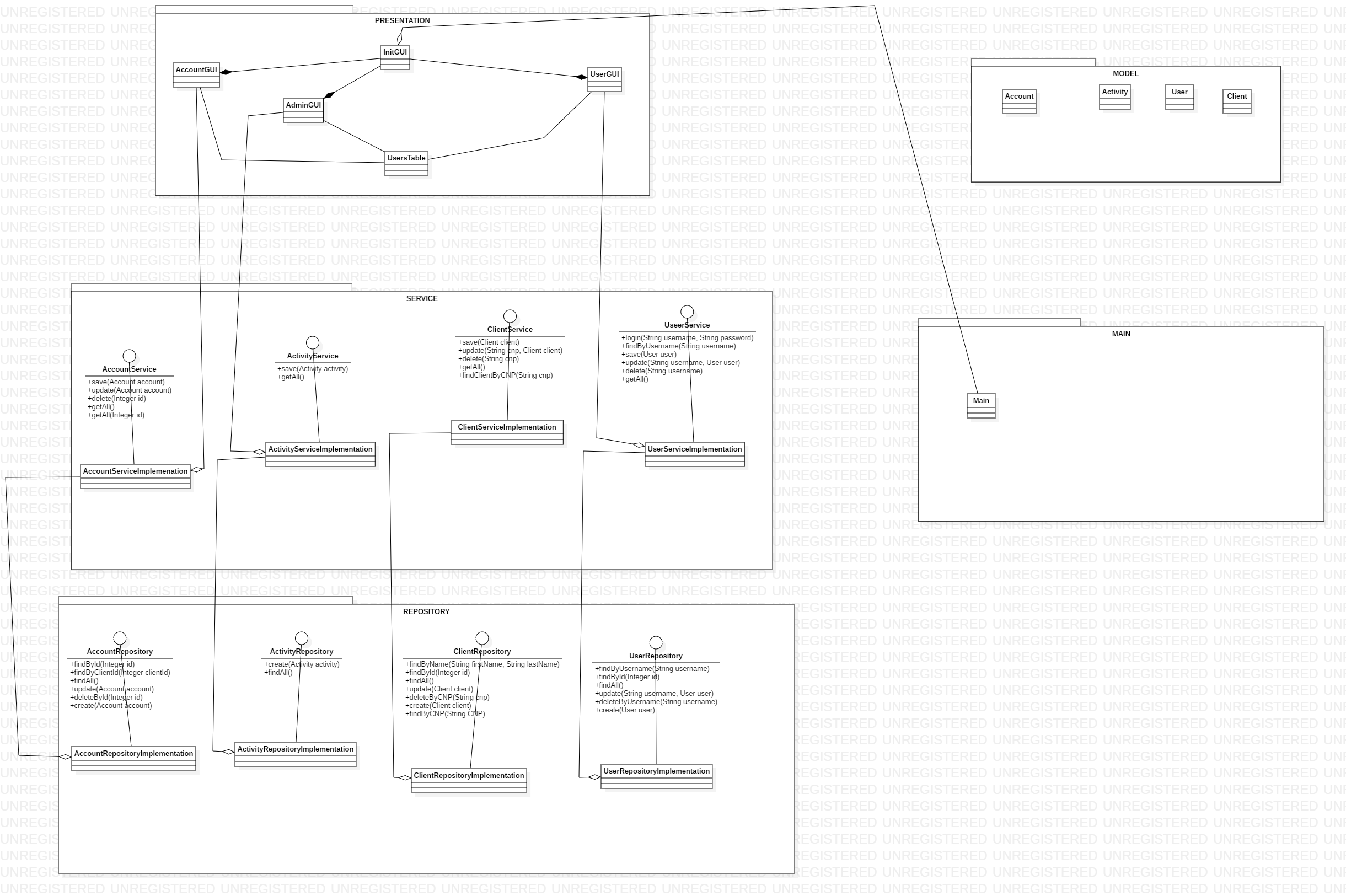
4. UML Sequence Diagrams

5. Class Design

**5.1 Design Patterns Description**

*-*

**5.2 UML Class Diagram**



6. Data Model

7. System Testing

Testarea se face cu Junit, testandu-se principalele functionalitati ale aplicatiei (operatiile CRUD). In clasa de testare se testeaza fiecare nivel ale sablonului arhitectural.

In plus, in fiecare metoda care se foloseste la nivelul Service, fiecare camp este vverificat daca datele introduse sunt corecte (ex. Numele si prenumele sa inceapa cu litere mari; CNP sa fie format doar din cifre, etc).

8. Bibliography