Assignment 1

Analysis and Design Document

Student: Marian-Florin Salvan

**Group: 30238**

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

As inputs we have all kinds of information about the clients or employees. These inputs are used in order to obtain our outputs, like: show the client/employee information in a processed way. These information is saved in the data base and their procession takes place in the business logic part of the system.

# Non-functional Requirements

The criteria used to judge the operations made by this system are the following:

* Separate implementation from interface
* User friendly interface
* Layered application(each layer with it’s own functionality that bound together make the system work)

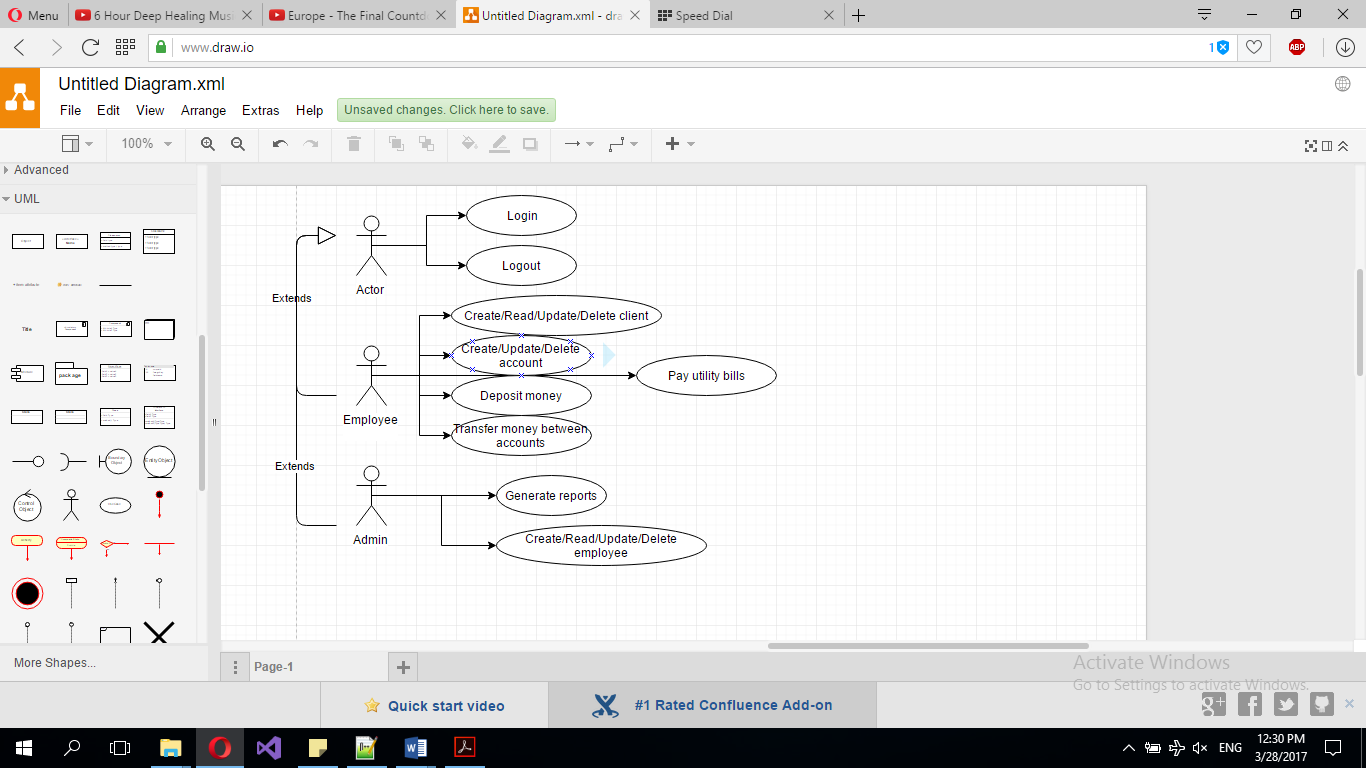
2. Use-Case Model

*Level:* user-goal level

*Primary actor:* employee who wants to sign in

*Main success scenario:* user successfully logs into account

*Extensions:* user enters wrong username or password, data-base error

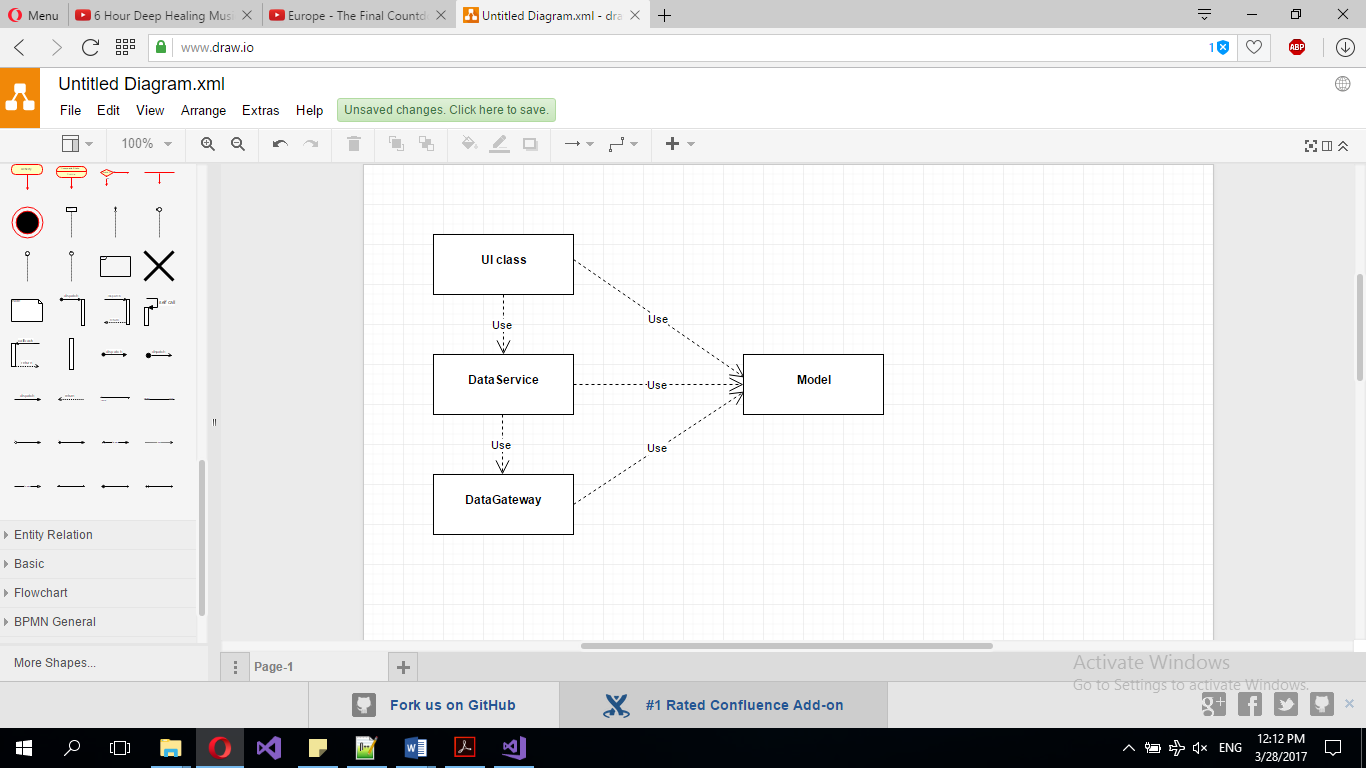


3. System Architectural Design

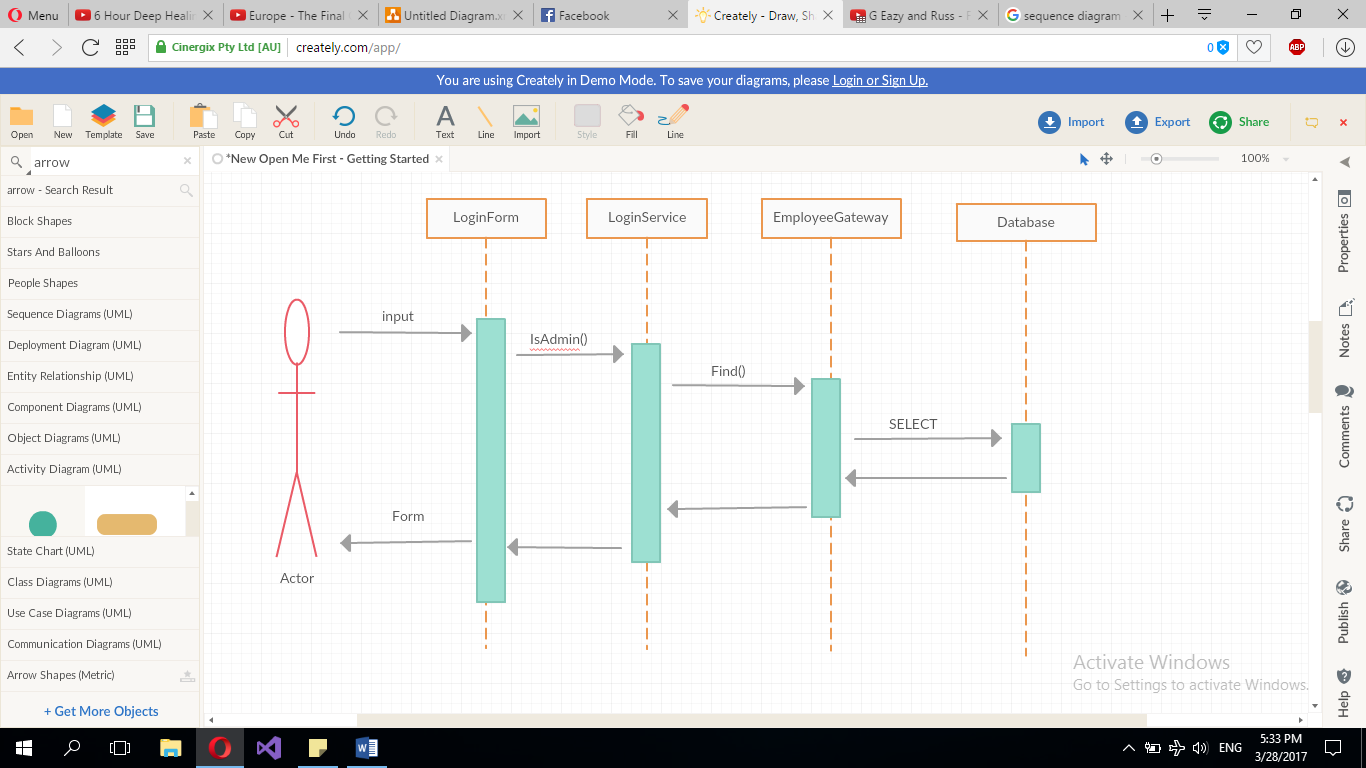
**3.1 Architectural Pattern Description**

The system uses the layered architectural pattern. This means that this system is split in 3 layers(presentation layer, business logic layer, data access layer and data base layer). Each layer can interact only with the layer beneath, but all of them can interact with a model.

**3.2 Diagrams**



4. UML Sequence Diagrams



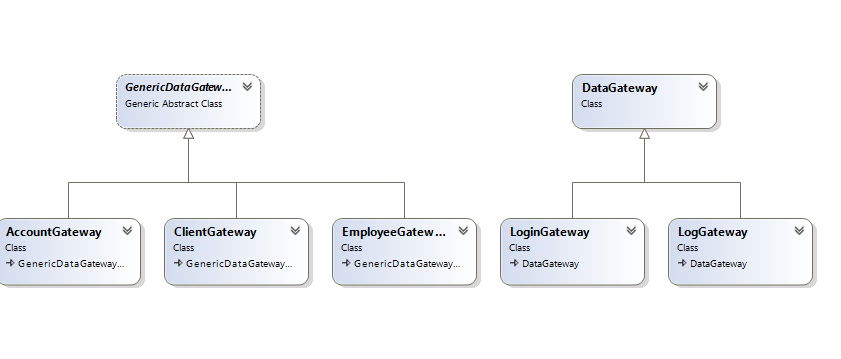
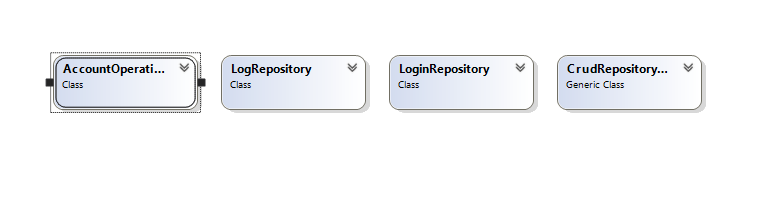
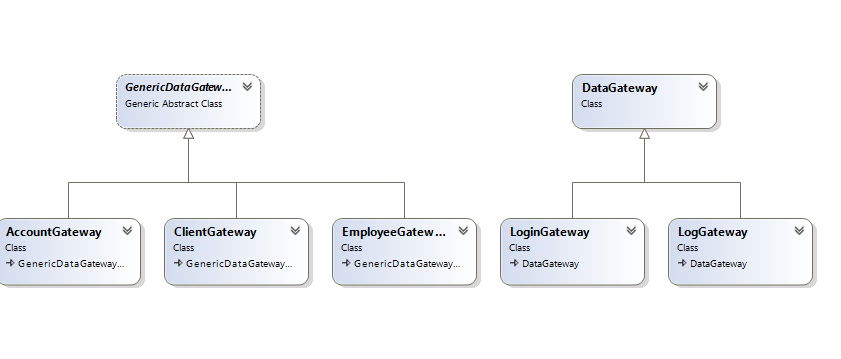
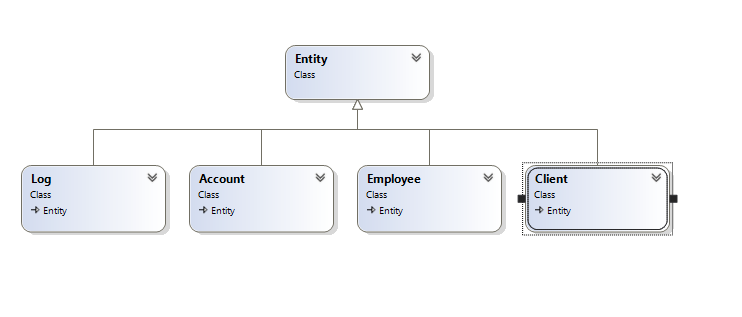
5. Class Design

**5.1 Design Patterns Description**

*Transaction Script –* used to organize all the logic of the application in small classes that group simple procedures which are making calls directly to a database or through a thin database wrapper.

*Table data gateway –* a table data gateway holds all the SQL for accessing a single table or view, making CRUD operations. Other code call its methods for all interaction with the database.

**5.2 UML Class Diagram**



6. Data Model

The data models used in the applications are 4 classes which are equivalent to the database tables:

* Employee: regular user or admin
* Client: holds all kind of information about a person who is a client of the bank
* Account: holds the balance, creation data and the ID of the client who holds it
* Log: model class used to describe the actions made by a regular employee

7. System Testing

The data input is tested when introduced in order to be valid.

Because this system is split in more layers the junit testing becomes very easy to implement (each layer can be tested separately).

8. Bibliography

<https://martinfowler.com>

<https://msdn.microsoft.com/en-us/library/54xbah2z(v=vs.110).aspx>

<https://msdn.microsoft.com/en-us/library/e80y5yhx(v=vs.110).aspx>