Bank Application

Analysis and Design Document

Student: Gliga-Hambet Bogdan Andrei

**Group:30431**

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

The designed application is designed for the front desk employees of a bank. It is designed to allow different types of users manage their accounts, perform actions on their accounts and transfer funds between them.

# 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

The application is a web-app which can be accessed via the browser, as long as the user has an active interned connection. It must have persistent storage in which the data is stored such that it is not lost.

2. Use-Case Model

C:\Users\GligaBogdan\Downloads\Untitled Diagram.png

Use case: Generate Activity Reports

Level: User-goal level

Primary actor: Application Administrator

Main success scenario: Login with user having permission rights => View All Employees List => Select desired employee => Click “Generate activity report” button

Extensions: Failure in case of corrupted data in the Database

3. System Architectural Design

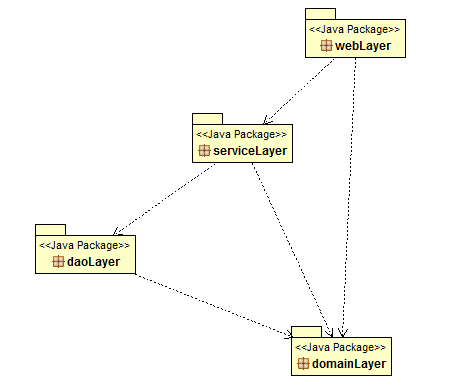
**3.1 Architectural Pattern Description**

The architectural pattern used in this application is the “Layers Pattern”. A multilayered software architecture is a [software architecture](https://www.wikiwand.com/en/Software_architecture) that uses many [layers](https://www.wikiwand.com/en/Layer_(object-oriented_design)) for allocating the different responsibilities of a software product.

**3.2 Diagrams**

More specific, the layers used in this application were the: presentation-layer ( consisting of JSPs), web-layer (containing all the Spring Controllers), service-layer (containing the Spring Services handling the business logic), the dao-layer (containing DAOs for database interaction) and the domain-layer ( being consisted of simple POJOs)

As it can be noticed from the package diagram, layers only talk to the layer below them in order to keep a clear separation of functionality in the application



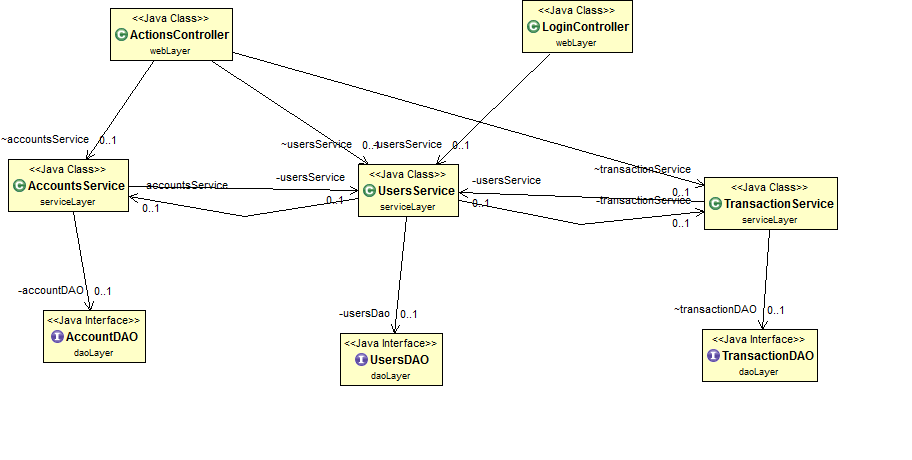
4. UML Sequence Diagrams

This is the sequence diagram for generating a report by the administrator for seing all the transactions made from a specific user’s bank account

C:\Users\GligaBogdan\Downloads\Untitled Diagram (1).png

5. Class Design

**5.1 UML Class Diagram**



As it can be seen from the UML class diagram as well, the layers architectural pattern is very obvious. The ActionsController and the LoginController only interact with the services, namely AccountsService, UsersService and TransactionService, which in turn interact tith the AccountDAO, UsersDAO and TransactionDAO

6. Data Model

**6.1 User**

This is used to model a real time user of the application. It has different fields on it such as the name, username, password, etc. It is the main model associated with each application user and it is by its attributes that the level of operations he can perform is decided.

**6.2 Account**

The account class represents a real time bank account. Each account has a unique id number and a certain amount of cash in it. The amount of cash cannot be negative and each user can have multiple bank accounts. The date and time when the account was created is also stored.

**6.3 Transaction**

As each user can transfer money between his account or to another user’s account, all this flows of money is tracked using the Transaction class. In it data about that specific money transaction is stored such as a unique id, the date when it was performed, between which users and for which amount of money. The administrator will be able to see all of these transactions.

7. System Testing

The application has been tested using JUnit tests, as they provide an easy, robust and reliable method to test your application. The main features that have been testes are the classes in the DAO Layer, as they handle the database operations and they are the most prone to fail.

A Test Suite has been designed for each of the 3 DAO classes, namely: AccountDAOTests, TransactionDAOTests and UsersDAOTests. Each one of these has methods which independently test CRUD operations which are performed by the application. Before each test, the database is cleaned and populated with validation data and after each test the database is cleaned again in order to leave it in a stable state and not affect other Junit tests

8. Bibliography

<http://docs.spring.io/spring/docs/3.0.x/spring-framework-reference/htmlsingle/spring-framework-reference.html>

<http://hibernate.org/orm/documentation/5.1/>

<https://tiles.apache.org/framework/tutorial/configuration.html>