<BankingService>

Analysis and Design Document

Student:Boian Maria

**Group:30433**

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

[Application description]

Use JAVA API to design and implement the API part of a web 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

*[Present the 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 (Pay) utilities bills (by inserting bill information).

The administrator user can perform the following operations:

- CRUD on employees’ information.

- Generate two types of reports files (one in pdf and one in csv format) for a period containing the activities performed by an employee.

# Non-functional Requirements

*[Discuss the non-functional requirements for the system]*

The system is secured with a log in text box that expects two types of users: ADMIN orUSER. After the user has login in , it can perform CRUD operations either on clients and account, or the employees data.

2. Use-Case Model

*[Create the use-case diagrams and provide one use-case description (according to the format below).*

*Use-Case description format:*

*Use case: <use case goal>*

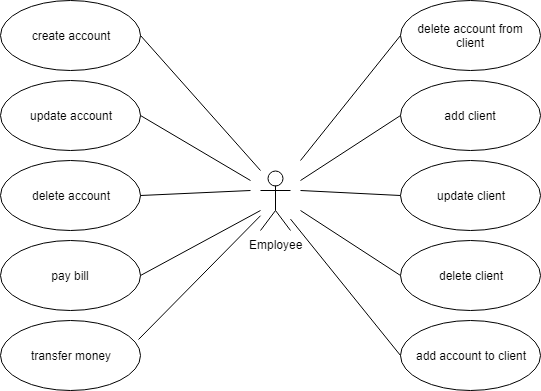
*Level: <one of: summary level, user-goal level, sub-function>*

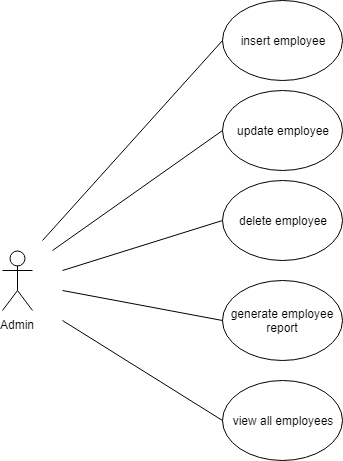
*Primary actor: <a role name for the actor who initiates the use case>*

*Main success scenario: <the steps of the main success scenario from trigger to goal delivery>*

*Extensions: <alternate scenarios of success or failure>*

*]*





*Use case: <Pay bill>*

*Level: < user-goal level >*

*Primary actor: <Employee>*

*Main success scenario: <Having the bill paid represents the success scenario.In order to get to this, employee must pay the amount on the bill with the client’s account. >*

*Extensions: <A failure scenario would be an empty account. >*

3. System Architectural Design

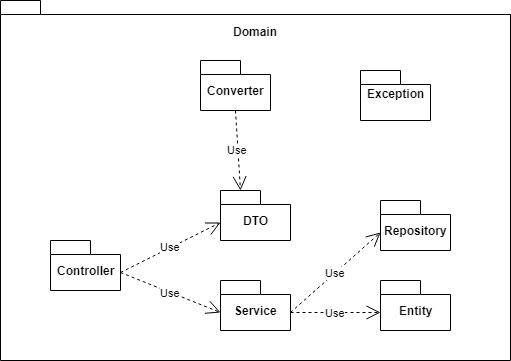
**3.1 Architectural Pattern Description**

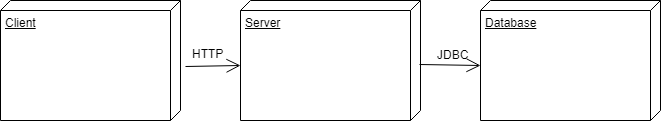
*[Describe briefly the used architectural patterns.]*

The main idea behind Layered Architecture is a separation of concerns –avoiding mixing domain or database code with the UI stuff, etc. The actual idea of separating a project into layers suggests that this separation of concerns should be achieved by source code organization.The presentation layer consists of the controllers and swagger, the business layer is responsible with CRUD operations implementation and the data layer contains entities and repositories which manage the connection with the database.There are also two mini-layers , one which consist of DTO and one with Converters.

**3.2 Diagrams**

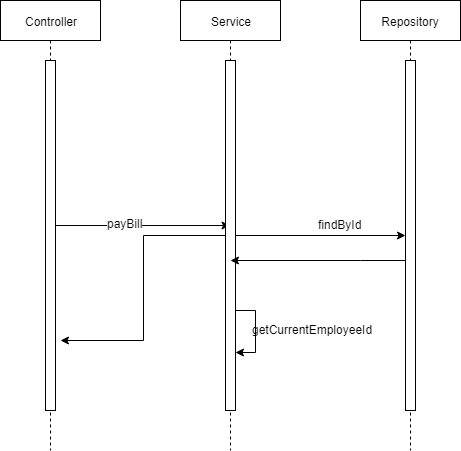
*[Create the system’s conceptual architecture; use architectural patterns and describe how they are applied. Create package, component and deployment diagrams]*

**

**

4. UML Sequence Diagrams

*[Create a sequence diagram for a relevant scenario.]*

**

5. Class Design

**5.1 Design Patterns Description**

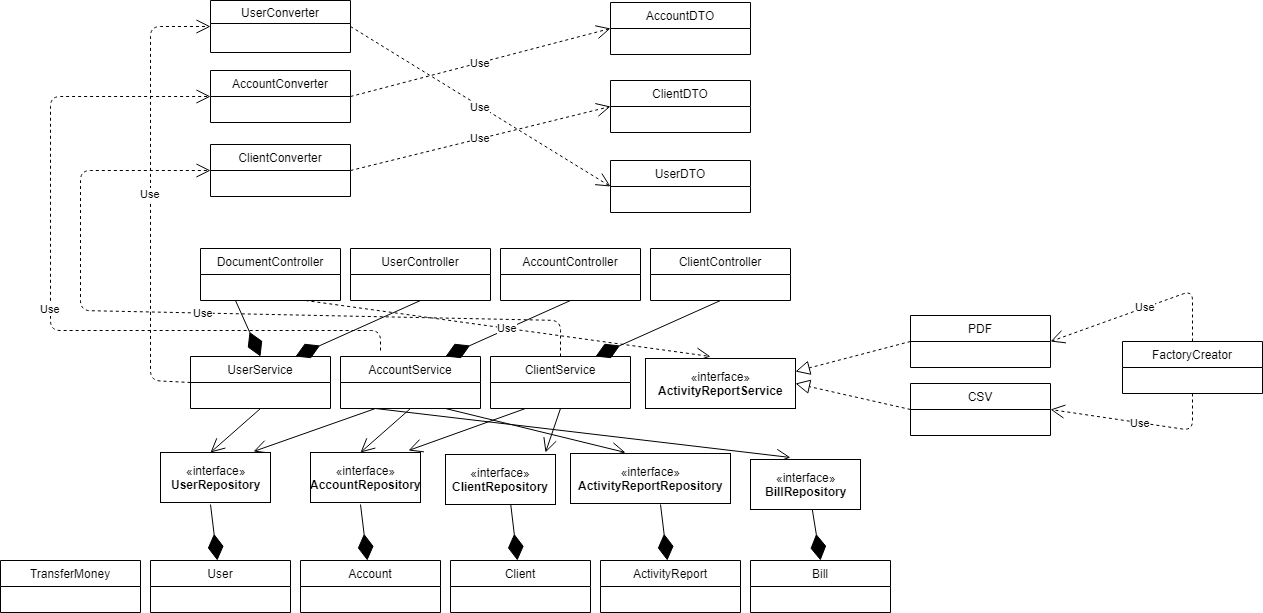
*[Describe briefly the used design patterns.]*

In short, a Factory is an object that can create objects without the use of a constructor**.** *In long,* a Factory is a function, method, or subroutine that can return objects that are considered to be ‘new’.

The idea is to use a static member-function (static factory method) which creates & returns instances, hiding the details of class modules from user.

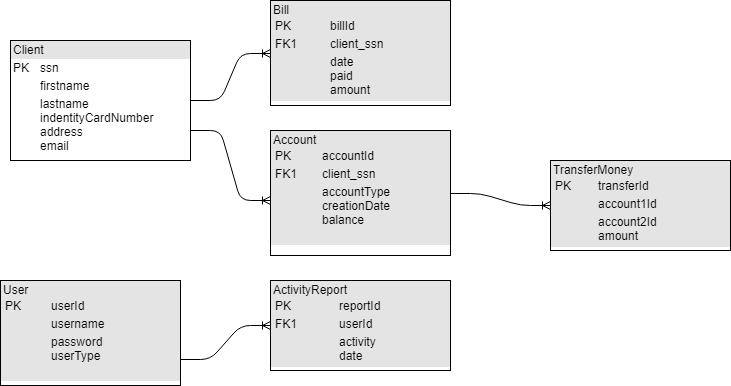
**5.2 UML Class Diagram**

*[Create the UML Class Diagram and highlight and motivate how the design patterns are used.]*



6. Data Model

*[Present the data models used in the system’s implementation.]*

**

7. System Testing

*[Present the used testing strategies (unit testing, integration testing, validation testing) and testing methods (data-flow, partitioning, boundary analysis, etc.).]*

8. Bibliography

<https://reflectoring.io/bean-validation-with-spring-boot/>

<https://developer.mozilla.org/en-US/docs/Web/HTTP/Basics_of_HTTP/MIME_types/Complete_list_of_MIME_types?fbclid=IwAR24fIoBiIrY25donBlHHaZPPAM4xNg51jbRZYeB6Gf0Egf1H1piBevwVFs>

<https://stackabuse.com/password-encoding-with-spring-security/>