Desk employees

Analysis and Design Document

Student: Iulia Ungur

**Group: 30235**

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. Bibliography 3

1. Requirements Analysis

# Assignment Specification

The application was designed to help, the front desk employees of a bank, handle the clients and their accounts in a user friendly way. They are able to fulfill the clients’ requests in managing their accounts and transfer money between them as well as process bills that the clients come to pay.

# Functional Requirements

Features included on the employees’ side are:

* adding, updating and viewing a clients’ information – which are his name, identity card number, personal numerical code, address;
* creating, updating, deleting and viewing a client account – accounts are either savings account or checked and contain the following information: identification number, type, amount of money and date of creation
* transferring money between accounts – selecting from a list the desired accounts and the amount of money to be transferred
* process utilities bills – the company being processed as another regular client with its’ own account and here the client has the option of paying cash

Features included on the administrators’ side are:

* creating, viewing, updating and deleting an employees’ information
* generating reports that are shown in a list with activities of an employee

# Non-functional Requirements

Availability: the system is available as long as there is an internet connection and a data base connection

Performance: the system is intuitive for the users to work with, so any fair amount of work can be performed in a short period of time

Security: the system is secure as long as the database is secured. Any unauthorized access to the db can put at harm the system

Testability: due to the fact that the system is organized on layers, every component can be tested individually

Usability: as mentioned, the system is intuitive and each type of user has its display out of the login page, which makes the flow of work much easier

2. Use-Case Model

Use case: EmployeeActivity

Level: summary level

Primary actor: Employee

Main success scenario: Employee logs in and handles the client information on the corresponding tab

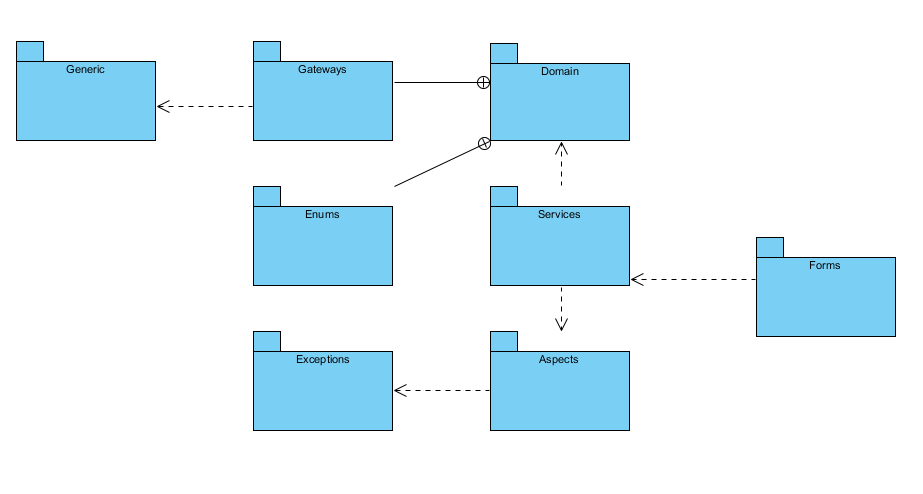
Extensions: Employee can do multiple actions on the same account

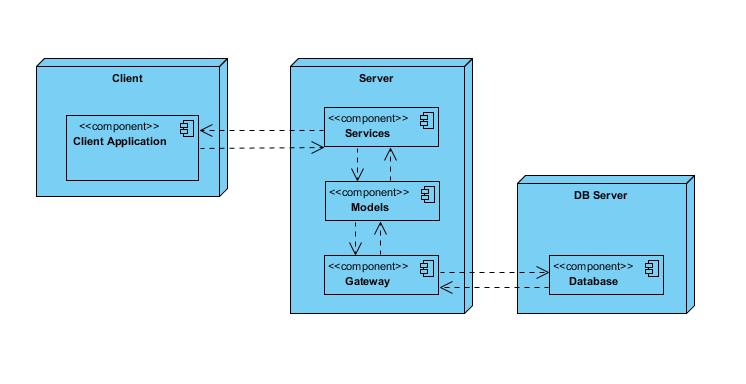
3. System Architectural Design

**3.1 Architectural Pattern Description**

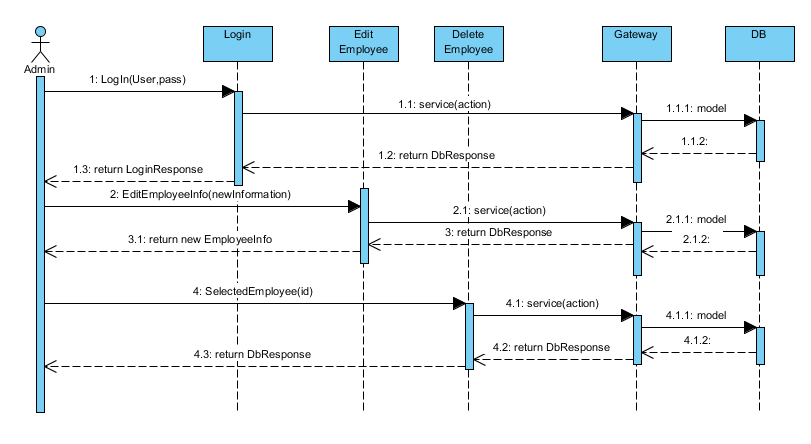
The pattern used on the architectural level is the Layers architectural pattern which partitions the concerns of the application into stacked groups. Meaning that the Layered architecture focuses on the grouping of related functionality within an application into distinct layers that are stacked vertically on top of each other. Functionality within each layer is related by a common role or responsibility. Communication between layers is explicit and loosely coupled. Layering the application appropriately helped to support a strong separation of concerns that, in turn, support flexibility and maintainability.

**3.2 Diagrams**





4. UML Sequence Diagrams



5. Class Design

**5.1 Design Patterns Description**

A domain model is generally implemented as an object model within a layer that uses a lower-level layer for persistence and "publishes" an API to a higher-level layer to gain access to the data and behavior of the model.

It is a system of abstractions that describes selected aspects of a sphere of knowledge or influence. The model can then be used to solve problems related to that domain. The domain model is a representation of meaningful real-world concepts pertinent to the domain that need to be modeled in software. The concepts include the data involved in the business and rules the business uses in relation to that data.

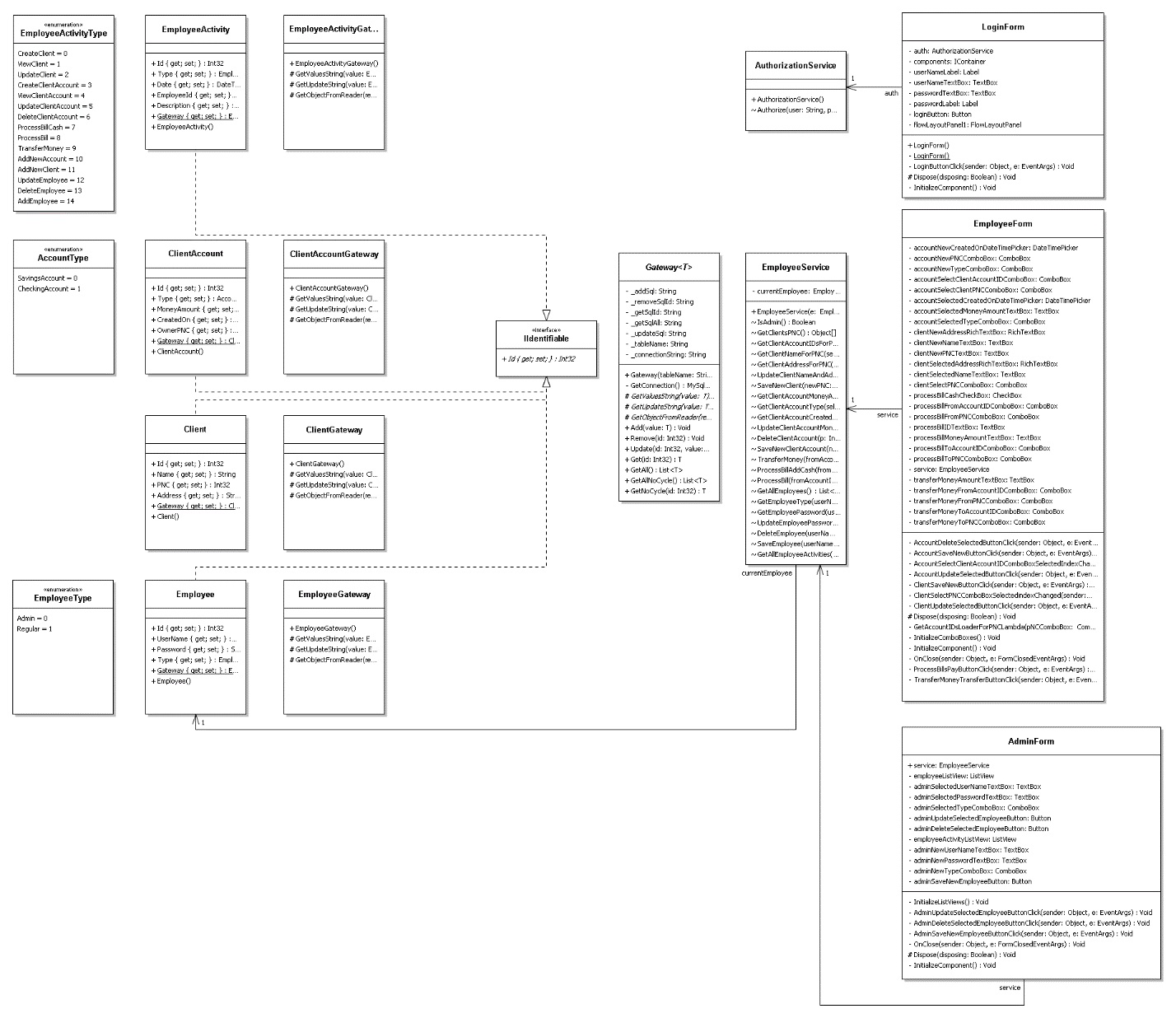
Table Data Gateway is a design pattern in which an object acts as a gateway to a database table. The idea is to separate the responsibility of fetching items from a database from the actual usages of those objects. Users of the gateway are then insulated from changes to the way objects are stored in the database.

The Gateway holds all the SQL for accessing a single table or view: selects, inserts, updates, and deletes. Other code calls its methods for all interaction with the database. A Table Data Gateway has one instance per table.

**5.2 UML Class Diagram**

The form controller calls the service to access the data. The service in order to retrieve the data from the database it calls the afferent gateway of the data model (a static instance of each table) from which it can tell the database to update according to the new data provided. Also the gateways are wrote in as minimal as it could be throughout an abstract class.

The implementation is based on low coupled classes that have minimal associations with each other except the basic instance.



6. Data Model

Regular Employee – handles the client information in terms of accounts, money and client and account information

Admin – another type of employee – handles the regular employee information and generates reports based on the employees activity

Client – holds its’ personal information and the accounts he possesses as well as the money amount for each account

Client Account – hold information about itself and the type of bank account checked or savings account

7. Bibliography

<https://msdn.microsoft.com/en-us/library/ee658117.aspx>

<https://en.wikipedia.org/wiki/Domain_model>

<https://en.wikipedia.org/wiki/Table_data_gateway>

<http://staff.cs.upt.ro/~ioana/arhit-engl/2015/DataAccessPattern.ppt>