Banking Application

Analysis and Design Document

Student: Angela-Paula Modringa

**Group:30432**

Revision History

|  |  |  |  |
| --- | --- | --- | --- |
| **Date** | **Version** | **Description** | **Author** |
| 04.04.2021 | 1.0 | Analysis and design | Angela-Paula Modringa |
| 06.06.2021 | 1.1 | Analysis and design | Angela-Paula Modringa |
|  |  |  |  |
|  |  |  |  |

Table of Contents

I. Project Specification 4

II. Elaboration – Iteration 1.1 4

1. Domain Model 4

2. Architectural Design 4

2.1 Conceptual Architecture 4

2.2 Package Design 4

2.3 Component and Deployment Diagrams 4

III. Elaboration – Iteration 1.2 4

1. Design Model 4

1.1 Dynamic Behavior 4

1.2 Class Design 4

2. Data Model 4

3. Unit Testing 4

IV. Elaboration – Iteration 2 4

1. Architectural Design Refinement 4

2. Design Model Refinement 4

V. Construction and Transition 5

1. System Testing 5

2. Future improvements 5

VI. Bibliography 5

# Project Specification

The object of this project is to design and implement a banking application, used by clients that want to manage their bank account easily, without having to go to the bank.

# Elaboration – Iteration 1.1

# Domain Model

*O imagine care conține text, exterior, placă

Descriere generată automat*

# Architectural Design

## Conceptual Architecture

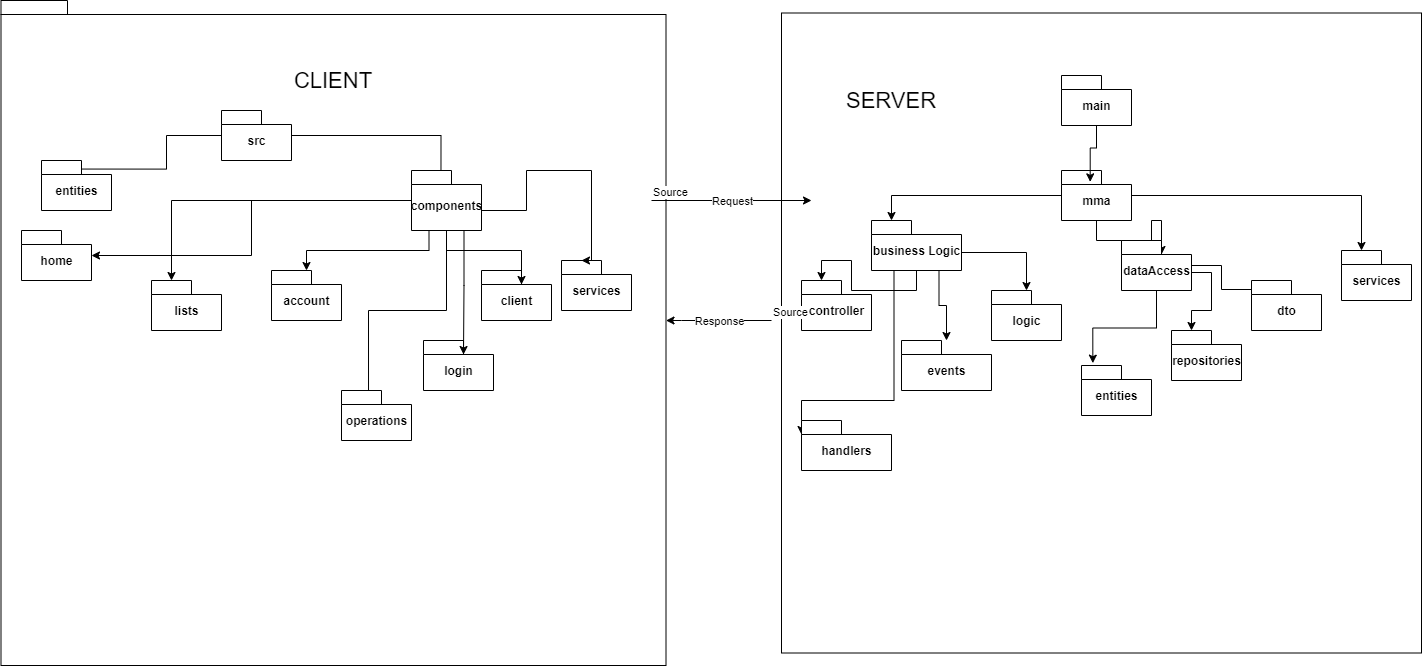
The architectural pattern used for this project is **CQRS architecture,** over the classic

**Client-Server architecture**. CQRS stands for Command **Q**uery **R**esponsibility **S**egregation. Systems built with the CQRS principle distinguish between data models used for Commands (write operations) and Queries (read operations). The command model is used to perform write/update operations efficiently while the query model is used for supporting the various read patterns effectively. The data between the two models is kept in sync by propagating the changes in the command model to the read model via domain events or any other mechanisms.

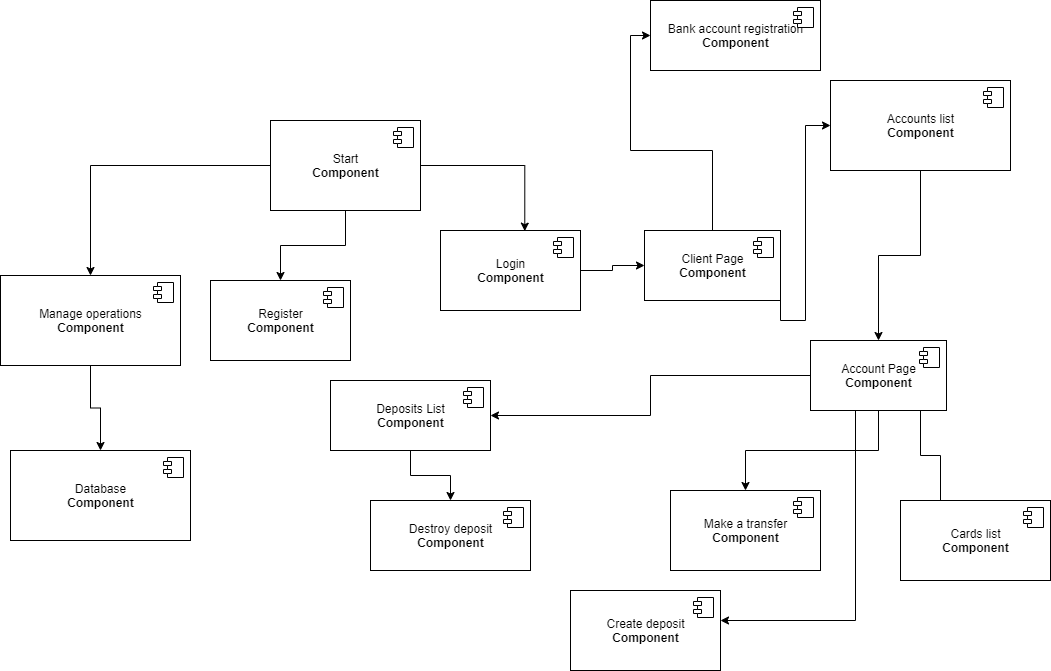
For the server side the CQRS pattern was used to modify the way that operations are transmitted and performed.

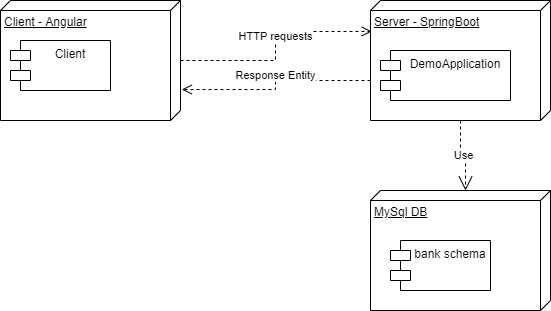
The client-server architecture works under the following principle: it partitions tasks between the providers of a service, called servers, and service requesters, called clients. Clients usually initiate communication sessions with servers, which await incoming requests. The device a client uses to ask for service from the server is the mediator between the service requester and the service provider and knows how to speak with both of them.

## Package Design



## Component and Deployment Diagrams





# Elaboration – Iteration 1.2

# Design Model

## Dynamic Behavior

Create a deposit:

O imagine care conține text, muzică

Descriere generată automat

Destroy a deposit:

O imagine care conține text, muzică

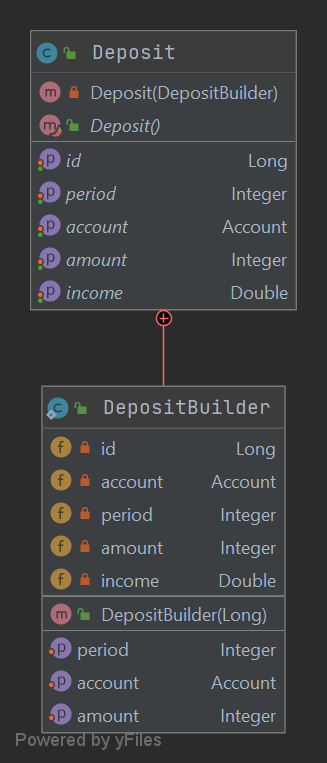
Descriere generată automat

## Class Design

O imagine care conține text

Descriere generată automat

I have used The Decorator design pattern to be able to find the age of a client without storing it in the database, because the age is constantly changing and it can be calculated from the date of birth.



I have used the Builder design pattern to create a Deposit class. It is used to assign a field step by step and return the complete object in the end. The process of constructing an object is generic, so we can create an object with any part of the fields we want, without needing specific constructors with those arguments.

The functionality is to create a deposit and to automatically calculate the income based on period and amount.

O imagine care conține text, captură de ecran, exterior, negru

Descriere generată automat

In the following UML I have included only a part of the classes for simplification. In the UML there are the commands and the events needed for some requests made by the client to take place, and the Mediator which deals with the logic of assigning an event its corresponding handler.

# Data Model

O imagine care conține text, negru, captură de ecran, placă

Descriere generată automat

# Unit Testing

For unit testing, I tested the CRUD (Create, Read, Update, Delete) operations that could be performed on the data models. An important aspect was validating the data that is introduced by the users. For example: if an user want to create a new account, the username must be verified in the database first and make sure it doesn’t already exists. If he wants to create a deposit, the deposited amount should be smaller or equal with the available amount from his account, etc.

I have verified that the color age is calculated correctly subtracting from today’s date, the date of birth of the user.

# Elaboration – Iteration 2

# Architectural Design Refinement

*[Refine the architectural design: conceptual architecture, package design (consider package design principles), component and deployment diagrams. Motivate the changes that have been made.]*

# Design Model Refinement

# Construction and Transition

# System Testing

For integration testing I was verifying that when a deposit is created for an account, the balance decrease, and when it is destroyed, the money is transferred automatically back to the account. I also verified the transfer operation, to make sure that the money is taken from an account and send to another (via id for own accounts or via iban for other’s account). If the account the user wants to send money to doesn’t exist, an error message is generated.

In the end, I introduced invalid input data to see if the error popups appear correctly to inform the user about what went wrong and why the action was not performed.

# Future improvements

For future improvement, the bill payment could be implemented, the interface could be styled more, a list with partner banks could be added. In that phase, the project could be extended easily, without breaking the present functionality.

# Bibliography

* <https://kislayverma.medium.com/architecture-pattern-cqrs-7a91e9050b0d>
* <https://openclassrooms.com/en/courses/6397806-design-your-software-architecture-using-industry-standard-patterns/6896156-client-server-architecture>
* <https://en.wikipedia.org/wiki/Builder_pattern#:~:text=The%20builder%20pattern%20is%20a,Gang%20of%20Four%20design%20patterns>.
* <https://www.tutorialspoint.com/design_pattern/decorator_pattern.htm>