Bank Account Management App

Software Design Project

Avram Alexandru-Andrei

Group 30432

1. Introduction

The aim of this project is to develop a web application that mirrors the functionalities found in traditional banking apps, while also incorporating features reminiscent of Revolut. The main objective is to create a user-friendly app that provides users with essential banking services such as account management, transactions, and payments.

2. Tech Stack

The chosen tech stack comprises Java Spring Boot for the backend and Vue 3 with Vuetify and Vuex plugins for the frontend. The database will be managed using PostgreSQL.

3. Software Architecture

The main approach is a layered architecture, based on the MVC (Model-View-Controller) model. Its 3 main layers are:

- Data Access Layer:
  - Comprised of the Repository and DAO files
  - Manages interactions with the database or data storage
  - Handles tasks related to data retrieval, storage, and manipulation
- Business Layer:
  - Contains the core logic, computations and rules of the application
  - Enforces data validation and handles data processing
  - Implements functionalities
  - Contains Service classes and interfaces
- Presentation Layer:
  - Responsible for presenting information to the user and handling user interactions
  - Includes components such as user interfaces, views, and controllers
  - Displays data retrieved from the business layer and interacts with users to collect input or trigger actions
  - Handles API routing

Aditionally, there are other classes (such as DTOs) in between these layers, which have the main functionality of transfering and/or abstracting data.

4. Functional Requirements

The application must allow users to:

- Create an account by filling in their personal information (email address, first and last name, and a created password)
- Add a profile picture
- Log in using their previously created account (email and password combination)

- Add money to their account
- Complete utility payments (pay bills)
- Generate account statement

- Send and receive money to and from other users who are registered

- View all of the cards related to their account
- View card information, after providing the account password
- Create/Delete cards

5. Non-functional requirements

The app should also allow users to:

- Track their spendings (per last week, month, year – graph format)
- View latest transaction

- Add other users as “trusted” contacts

- Have a disaposable card (Card number and security code get scrambled after each transaction)
- Freeze their cards

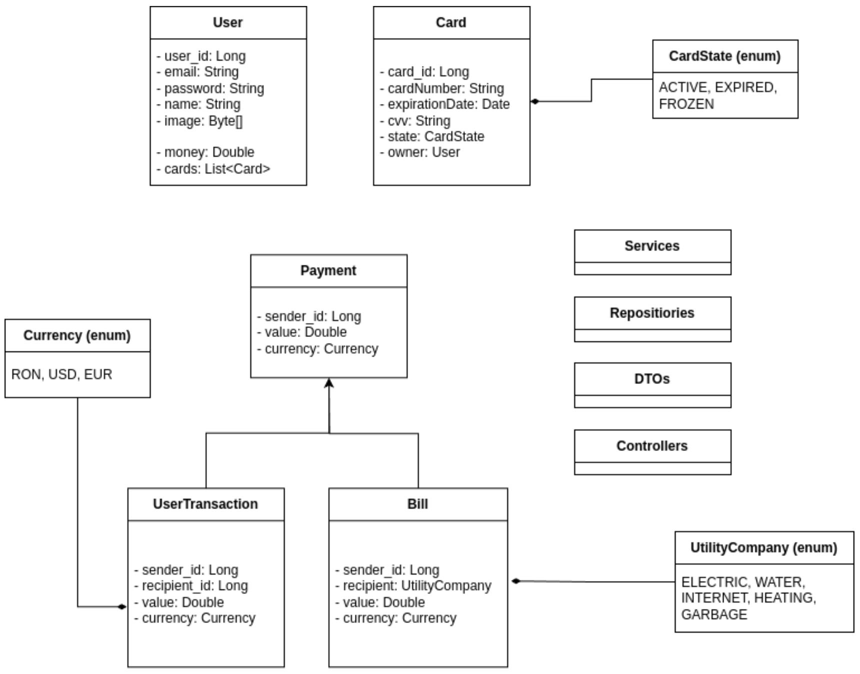
Aditionally, the application should

- Have an aesthetic and easy to use interface
- Run smoothly and efficiently
- Be secure – encrypt sensitive user data (especially passwords)

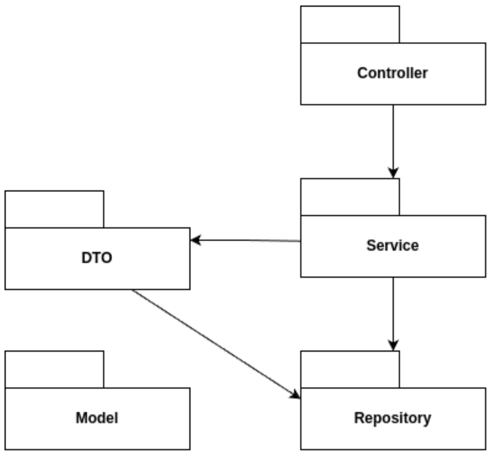
- Include a system for estimating spendings for the next week/month/year
- Have language options
- Include multiple currencies
- Have a currency exchange feature

6. Diagrams

6.1. Class diagram

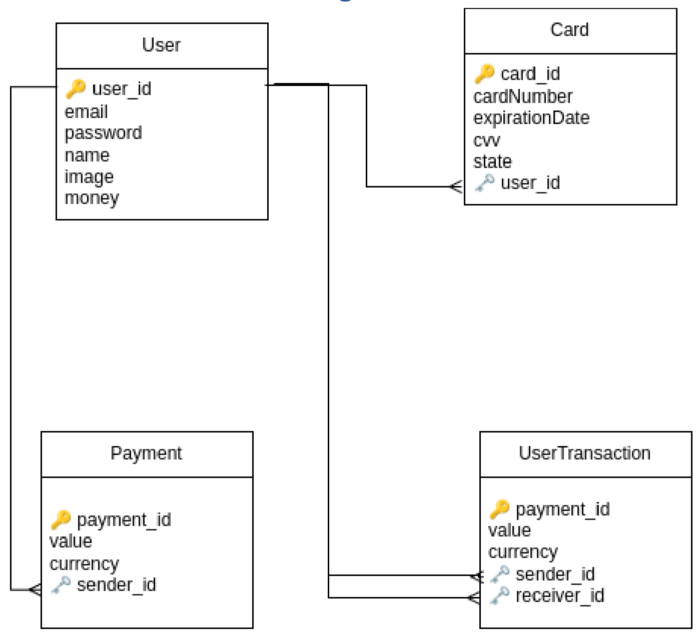


6.2. Package Diagram



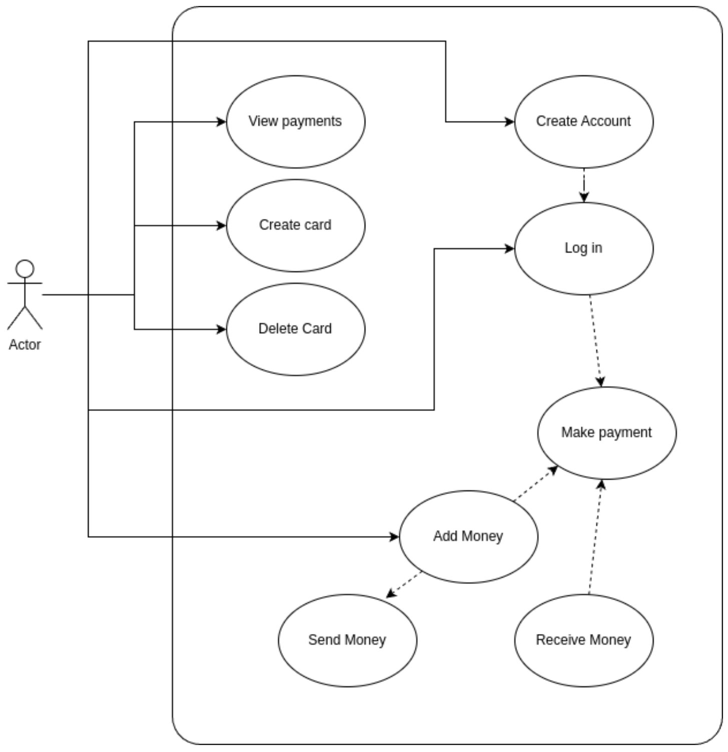
The packagee diagram highlights the layered architecture, with the three main layers being clearly visible. The package containing DTOs is a link between the Data and Business layer. While some data is passed directly to the Business layer, other data is passed through DTOs, in order to avoid unnecessary data from being sent.

6.3. Database Diagram

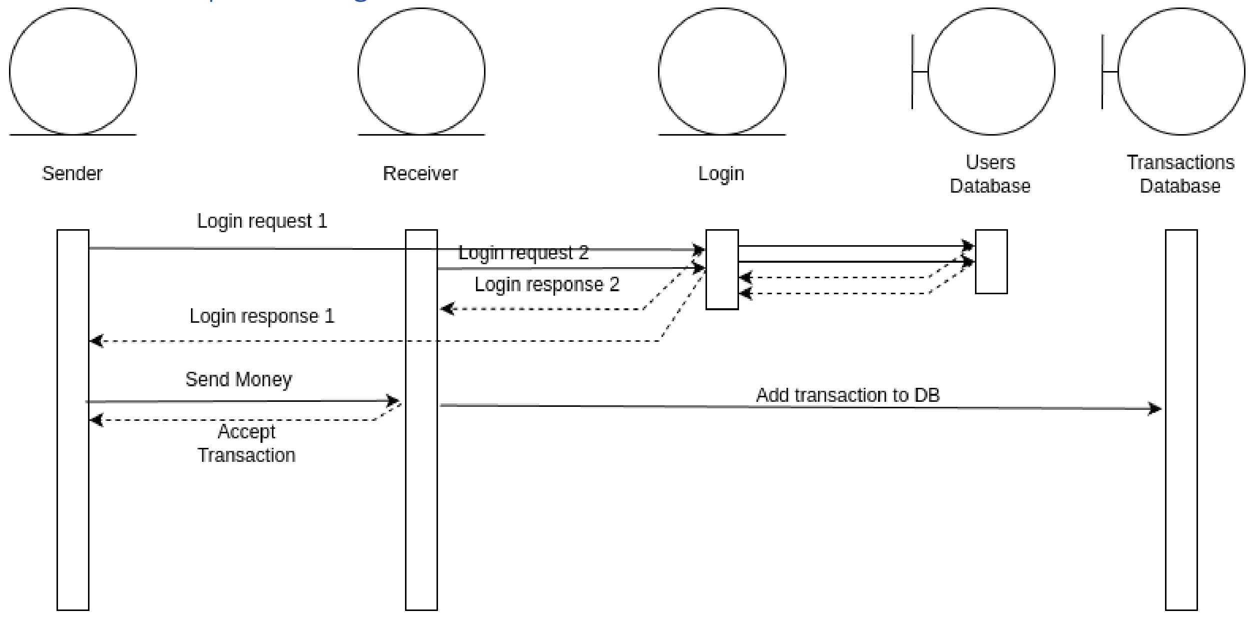


Simple database design, only including one-to-many relations. Payment table include bill payments made by users, while the UserTransaction table holds the transactions completed between two users.

6.4. Use Case Diagram



6.5. Sequence Diagram



Sequence diagram for a transaction between two users – a sender and a receiver.

7. Use Case Definition

Create Account:

- Description: Allows new users to sign up for an account.

Log In:

- Description: Allows registered users to access their accounts.

Create Card:

- Description: Lets users generate new cards.

Delete Card:

- Description: Enables users to remove existing cards from their account.

Add Money:

- Description: Allows users to add funds to their account balance.

Make Payment:

- Description: Enables users to send money to others.

View Payments:

- Description: Allows users to see their past transactions.

Receive Money:

- Description: Lets users receive funds from others.

Send Money:

- Description: Enables users to transfer funds to others.