Bachelor Project

Czech Technical University in Prague

F3

Faculty of Electrical Engineering
Department of Computer Science

Managing personal bank accounts

Dmytro Rastvorov

Supervisor: Ing. Božena Mannová, Ph.D.
May 2024

BACHELOR‘S THESIS ASSIGNMENT

I. Personal and study details

Student's name:	**Rastvorov Dmytro**	Personal ID number:	**503169**
Faculty / Institute:	**Faculty of Electrical Engineering**		
Department / Institute:	**Department of Computer Science**		
Study program:	**Software Engineering and Technology**		

II. Bachelor’s thesis details

Bachelor’s thesis title in English:

Managing personal bank accounts

Bachelor’s thesis title in Czech:

Správa osobních bankovních účtů

Guidelines:

The goal of the work is to create a simple and user-friendly application
for managing personal finances. The application will allow tracking of expenses and
income, creating budgets and tracking them and help keep them within them
order and plan them. Present overviews of financial data graphically.
Solution procedure:
1. Familiarize yourself with the management of personal bank accounts. Analyze existing
solutions available to you for specific applications, compare and evaluate them.
2. Based on the analysis, design the basic functionalities of the proposed application.
3. Choose the application architecture and select the most suitable technologies for
implementation. Justify the choice of technologies.
4. Implement and test the application, including user tests.
5. Evaluate the results and suggest any additional functionality or other improvements.
6. Use appropriate means of software engineering when solving.

Bibliography / sources:

[1] Roger S. Pressmann Bruce Maxim: Software Engineering: A Practitioner's Approach ,
ISBN-10: 9780078022128
[2] Use Case Diagram https://www.ibm.com/docs/en/rational-soft-arch/9.6.1?topic=diagrams-use-case
[3] Česká Spořitelna https://george.csas.cz/

Name and workplace of bachelor’s thesis supervisor:

Ing. Božena Mannová, Ph.D. Center for Software Training FEE

Name and workplace of second bachelor’s thesis supervisor or consultant:

Date of bachelor’s thesis assignment: **16.02.2024** Deadline for bachelor thesis submission: **24.05.2024**

Assignment valid until: **21.09.2025**

Ing. Božena Mannová, Ph.D. Supervisor’s signature	Head of department’s signature	prof. Mgr. Petr Páta, Ph.D. Dean’s signature

III. Assignment receipt

The student acknowledges that the bachelor’s thesis is an individual work. The student must produce his thesis without the assistance of others, with the exception of provided consultations. Within the bachelor’s thesis, the author must state the names of consultants and include a list of references.

.	
Date of assignment receipt	Student’s signature

Acknowledgements

I would like to express my immense gratitude to the teachers for the knowledge I have gained. I would also like to express my special thanks to my bachelor's thesis supervisor Ing. Božena Mannová Ph.D., for timely advice and assistance in writing my bachelor thesis. Finally, I would like to express my gratitude to my family and friends who supported me throughout my studies.

Declaration

I hereby declare that I have used only the information I received during my studies at the university and the sources attached to this bachelor's thesis to complete this bachelor's thesis. I have done this bachelor's thesis on my own, without the help of third parties.

Prague, 16. May 2024

Abstract

This thesis demonstrates the research of managing personal bank accounts, creating the logic of the application and its implementation. The main goal of this project is to create an efficient server part that will allow to perform transactions, transfers and other banking operations rapidly, the possibility to communicate with the users through online chat, as well as the possibility to modify user data without the need for third party confirmation.
The following chapters will analyze the existing solutions, the project development proposals and its implementation. The implementation of the server side will be done by using the Java language and the use of the Spring Boot framework. In addition, monolithic architecture will also be used to achieve the expected result. To demonstrate the result, the Postman application will be used.

Keywords: Banking operations, Server side, Java, Spring Boot, Monolithic architecture, Postman

Supervisor: Ing. Božena Mannová, Ph.D.
Praha 2, Karlovo náměstí 13, E-430

Abstrakt

Tato práce se zabývá správou osobních bankovních účtů. Hlavním cílem tohoto projektu je vytvoření efektivní serverové části, která umožní rychlé provádění transakcí, převodů a dalších bankovních operací, možnost komunikace s uživateli prostřednictvím on-line chatu a také možnost úpravy uživatelských údajů bez nutnosti potvrzení třetí stranou.
V následujících kapitolách budou analyzována stávající řešení, návrhy na vývoj projektu a jeho realizace. Implementace na straně serveru bude provedena pomocí jazyka Java a s využitím frameworku Spring Boot. Kromě toho bude také použita monolitická architektura k dosažení očekávaného výsledku. K demonstraci výsledku bude použita aplikace Postman.

Klíčová slova: Bankovní operace, Strana serveru, Java, Spring Boot, Monolitická architektura, Postman

Překlad názvu: Správa osobních bankovních účtů

Contents

Figures

Chapter 1
Introduction

This chapter describes what the project is about, what aspects will be touched and what the purpose of the project is to accomplish.

1.1 Motivation

Nowadays there are many different banks that try to attract future customers with favorable services. When creating a card, customers are asked to install the bank's mobile app where the user will have full access to their data and accounts, but what many banks don't tell to the customer is that they also have web apps that can also be used to manage their accounts.

Due to this, provided that many banks have a modern interface of their web application, they are also complex. The complexity is that it takes time to fully use the application, for example, if a customer want to make a transaction, he has to click through several pages when it could be done in one: change user data, for example, if the customer's contact number has been changed, the possibility of transferring funds to another account and more.

1.2 Goal

The aim of the work is to analyze the existing web applications of banks and develop a server part for managing personal bank accounts. The program should present the ability to easily and quickly manage bank accounts, getting actual exchange rates, creating new cards, conducting transactions and the ability to change your data. In the course of the project, as well as the bachelor's thesis, a prototype of the server with functionality covering the client's needs will be created.

Chapter 2

Overview of existing solutions

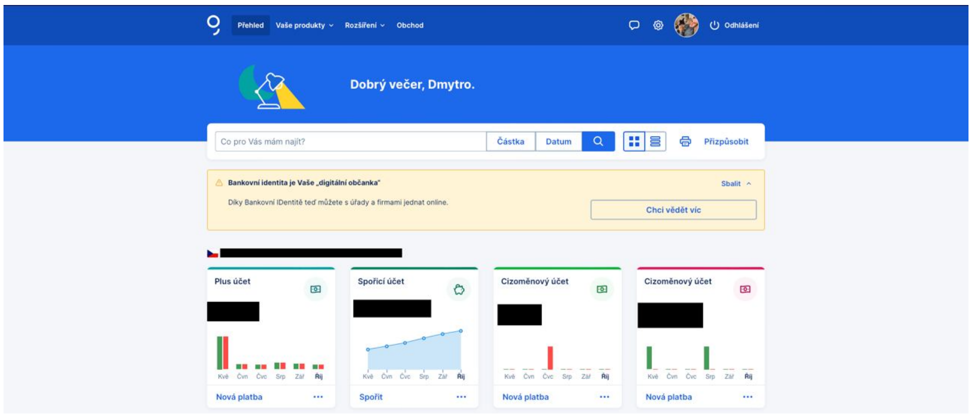
This chapter will analyze several banks from different countries, examining their positive and negative aspects. Based on this, functional and non-functional inquiries will be founded.

2.1 Česká Spořitelna

Česká Spořitelna is the largest bank in the Czech Republic, which is used by the majority of the population of the Czech Republic. The bank offers favorable and pleasant offers to customers, and it is also a partner of several universities and a sponsor of various events.

Upon authorization on the bank's page user will see his personal bank account, in which user can find his active accounts and a graph showing the interaction with the bank account. When working with the bank card user can see the date of purchased goods or received funds, type, name and amount.

User can also change his data, interact with the bank account, change the limit at which user can withdraw money from the account, pay online for goods, add contacts, create templates and new accounts. [10]



Dobrý večer, Dmytro.

Bankovní identita je Vaše „digitální občanka"

Díky Bankovní IDentitě teď můžete s úřady a firmami jednat online.

[redacted]

[redacted]

[redacted]

[redacted]

[redacted]

Figure 2.1: Česká Spořitelná web application

Disadvantages:

1. Navigation Clarity Issue - Navigation is quite complicated and not quite clear for a user who opens this page for the first time. For example, user can observe the account on the main page of the website, but if he hover over "Your Products", user will see the same thing.

2. Too big a set of pages - It would have been better to have all the necessary and important information on 1 page, but in the end it turns out that to get somewhere, user has to open several different pages, which is not exactly pleasant.

3. Exchange Rate Visibility Absence - On the main page does not see the exchange rate, which would be appreciated to see at least in relation to world currencies such as the dollar and euro.

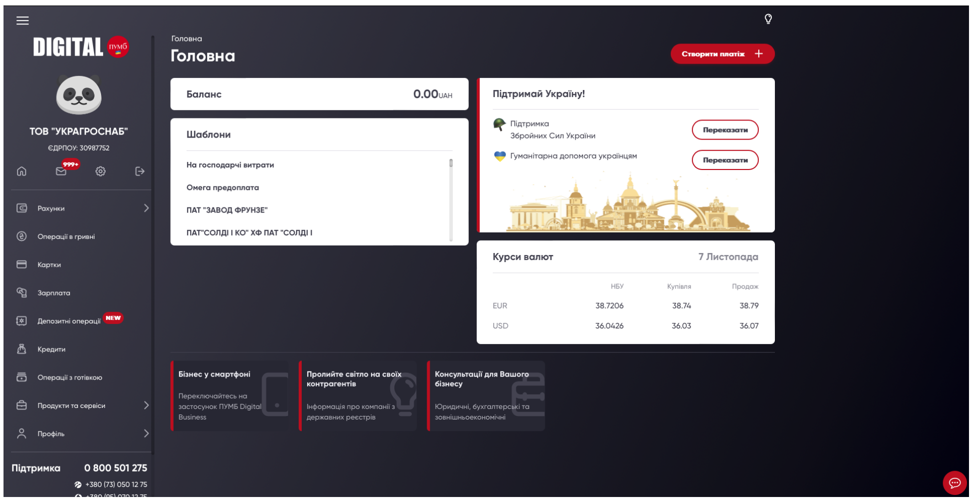
4. One theme type for the website - Would be better if the website had a dark theme in addition to the light theme.

2.2 Digital Pumb

Digital Pumb is a well-known bank in Ukraine, like the Czech bank, the Ukrainian bank offers quite pleasant services for the client. The big plus, unlike the Czech bank, is the direct connection to the application for documents. There is no need to go to the bank to open a Physical Entrepreneur, it is enough just to open a web application, pass identification through the application "Dia”, which is an application for storing documents and sign an online document, which according to the web website, will take less than 5 minutes. Also user can apply for a loan online, which is a positive factor.

In the personal account user can watch the exchange rate, which is quite convenient. The interface in contrast to the Czech bank is quite pleasant and convenient and allows the user to perform the necessary tasks in just 1 click.

The website itself is dynamically one-page, which allows the user to easily perform the necessary financial tasks without navigating through several pages. In this bank, user can interact with his account, set limits, send paychecks, open accounts and cards, take loans, and perform cash transactions. Also in the settings there is an option to change the theme (from dark to light), send passwords to Viber messenger, set a default account and much more. [11]



ТОВ "УКРАГРОСНАБ"
ЄДРПОУ: 30987752

Баланс 0.00UAH

Шаблони
На господарчі витрати
Омега предоплата
ПАТ "ЗАВОД ФРУНЗЕ"
ПАТ"СОЛДІ І КО" ХФ ПАТ "СОЛДІ І

Підтримай Україну!
Підтримка Збройних Сил України
Гуманітарна допомога українцям

Курси валют 7 Листопада

	НБУ	Купівля	Продаж
EUR	38.7206	38.74	38.79
USD	36.0426	36.03	36.07

Бізнес у смартфоні
Переключайтесь на застосунок ПУМБ Digital Business

Пролийте світло на своїх контрагентів
Інформація про компанії з державних реєстрів

Консультації для Вашого бізнесу
Юридичні, бухгалтерські та зовнішньоекономічні

Підтримка 0 800 501 275
+380 (73) 050 12 75

Figure 2.2: Digital Pumb web application

Disadvantages:

1. Language Limitation - In the settings there is no option to choose, for example, English. Therefore, the choice of this bank is less interesting for foreigners.
2. Too narrow a range of possibilities to customize the profile - Apart from changing the password and photo image no other services are offered.

2.3 Privat24

Privat24 is the largest bank in Ukraine, which represents about a quarter of the entire banking system of the country, being systemically important and the largest savings specialized bank, servicing a third of the deposits of the country's population.

The bank has a quite pleasant interface and has the same functions as other banks. There is an opportunity to transfer money to an account at once, transfer to another card, convert the amount of money in order to understand how much user can get for a certain amount (for example, the ratio of hryvnia to dollar).

On the main page of the profile there is a currency rate against the dollar. If user clicks on the rate, he will get a larger list of currency rates. There are also loan products on the page. With the help of this bank user can immediately pay for music listening, parking, withdraw a deposit or even buy a train or airplane ticket and much more. User can set up automatic payments for services, as well as create a template or use it. When user click on a profile, he can also contact his personal banker immediately. A nice touch is the ability to revert the website to an older version for those who are used to it, for example. It is also possible to change the theme (user can also set the interaction wallpaper with the operating system). [12]

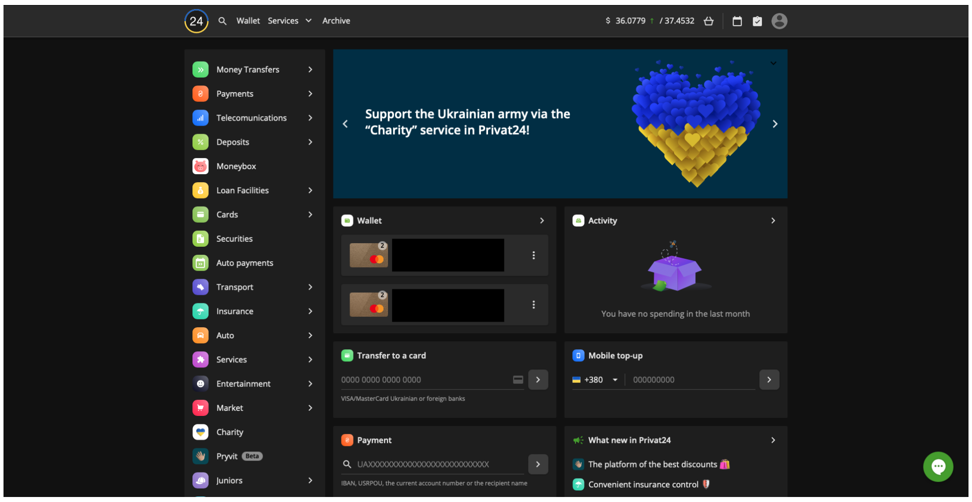


Figure 2.3: Privat24 web application

Disadvantages:

1. Changing profile data - Besides photos and passwords, users can change information only by contacting the bank, which is not so convenient.

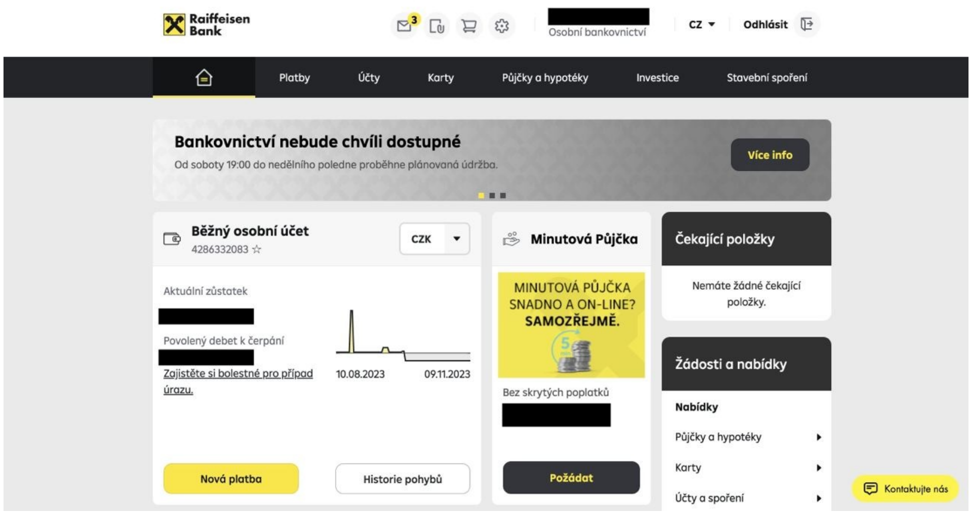
2. Overloaded UI - There are too many options and different options on the page, which makes it very easy to get lost in the interface.

3. Problems with transferring money to another bank (foreign) - When attempting to do so, it will ask the user to come to the bank's department in person, which is not so convenient.

2.4 Raffeisenbank

Raiffeisenbank is an Austrian bank that is one of the largest banks in Europe (about 15 countries, including Austria itself) and serves over 17 million customers.

The bank provides a variety of services to customers that may be tempting to many people. The bank itself also has a web version through which it can interface with its account. When user authorize in his personal profile, user will be able to see the current balance of the amount on the card. It is also possible to switch between cards. In addition, it is possible to view the card history, check the messages received from the bank, make payments, transfer money between accounts, make conversions, open a new account, get a statement of accounts, ask for a new card, take out a loan for a limited amount, make an investment, just like in other banks. It is also possible for user to manage his data. [13]



Bankovnictví nebude chvíli dostupné

Od soboty 19:00 do nedělního poledne proběhne plánovaná údržba.

Nemáte žádné čekající položky.

Figure 2.4: Raffeisen web application

Disadvantages:

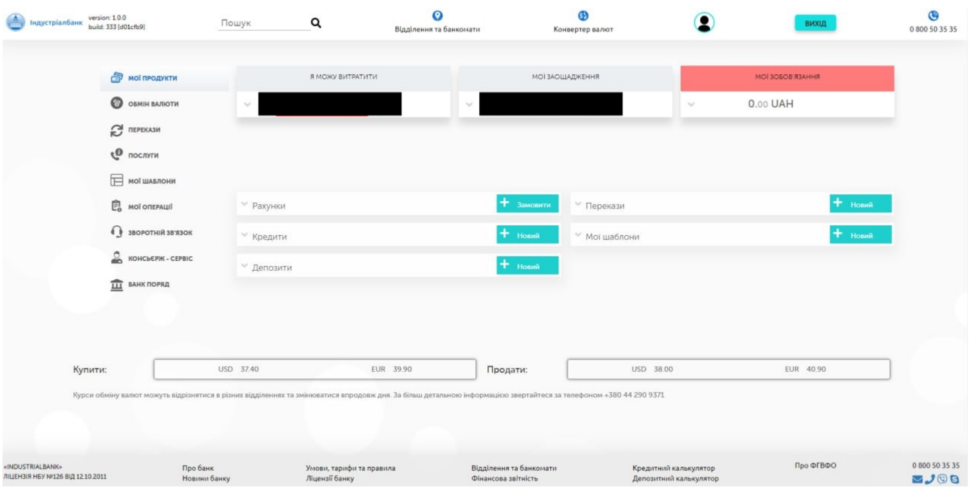
1. Empty home page - The main page contains quite little information. There is nothing else besides the main map and this is a big disadvantage.
2. One type of theme for the website - It would be nice if the website had a dark theme in addition to a light theme.
3. Poor online service - The service is quite slow for issuing documents.

2.5 Industrial bank

Industrial Bank is a Chinese bank that is the largest bank in the country and one of the largest banks in the world in terms of its assets and capital.

This bank also has its departments in Ukraine. The bank provides a variety of banking services, as well as actively implements modern technologies for the convenience of customers. The website of the bank has a type of one-sided dynamic website.

In the personal profile of the web version of the bank user can see his account, debts, exchange rate and various options offered in the left bar: currency exchange, sending funds, services, creation and management of templates, operations, callback, the ability to contact the concierge. On the main page user can create new templates, sending funds, bills, loan, deposits. User also have the possibility to manage his profile, to change it. [14]



[redacted]

[redacted]

0.00 UAH

Купити: USD 37.40 EUR 39.90 Продати: USD 38.00 EUR 40.90

Курси обміну валют можуть відрізнятися в різних відділеннях та змінюватися впродовж дня. За більш детальною інформацією звертайтеся за телефоном +380 44 290 9371

Figure 2.5: Industrial bank web application

Disadvantages:

1. Old interface - The interface is quite old and does not look presentable against other banking websites.

2. One type of theme for the website - It would be better if website had a dark theme in addition to the light theme.

3. Website crashes when zooming in - If the user zooms in on a page, the website crashes.

4. Lack of Home Button - When a user clicks from the personal cabinet, the icon throws the user out of the personal cabinet and opens the website from the beginning.

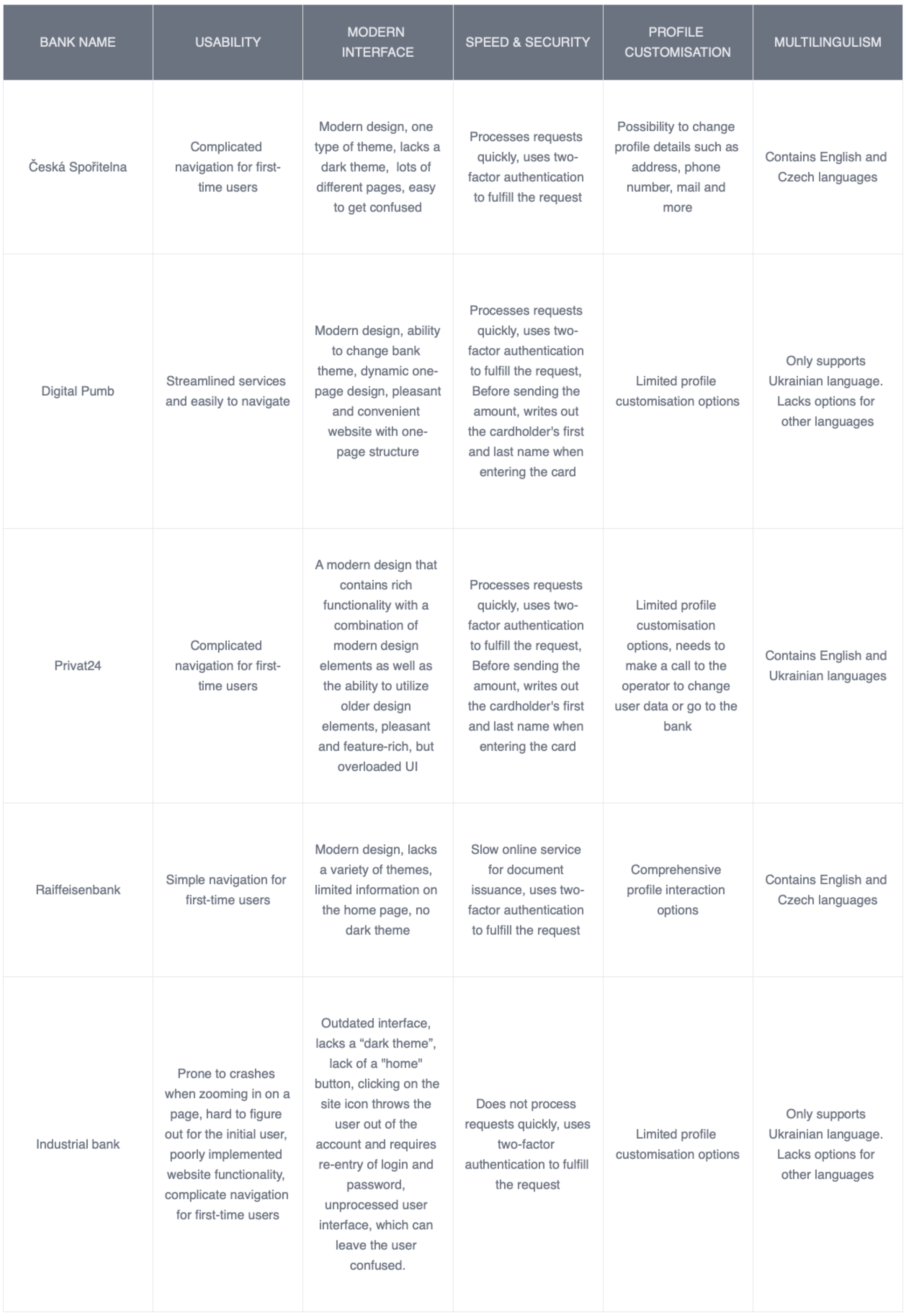
5. Ambiguous Icons in “Utilities" - Within the "Utilities" section, the icons displayed appear unfamiliar and lack clarity.

6. Unclear arrangement of sections - It confuses users on which page they are at.

2.6 Conclusion

The purpose of studying 5 different bank websites was to find out how well they fulfill the main criteria - ease of use, speed and security, profile customisation, modern interface and multilingualism.

The table below shows the final analysis of the banks.



BANK NAME	USABILITY	MODERN INTERFACE	SPEED & SECURITY	PROFILE CUSTOMISATION	MULTILINGULISM
Česká Spořitelna	Complicated navigation for first-time users	Modern design, one type of theme, lacks a dark theme, lots of different pages, easy to get confused	Processes requests quickly, uses two-factor authentication to fulfill the request	Possibility to change profile details such as address, phone number, mail and more	Contains English and Czech languages
Digital Pumb	Streamlined services and easily to navigate	Modern design, ability to change bank theme, dynamic one-page design, pleasant and convenient website with one-page structure	Processes requests quickly, uses two-factor authentication to fulfill the request, Before sending the amount, writes out the cardholder's first and last name when entering the card	Limited profile customisation options	Only supports Ukrainian language. Lacks options for other languages
Privat24	Complicated navigation for first-time users	A modern design that contains rich functionality with a combination of modern design elements as well as the ability to utilize older design elements, pleasant and feature-rich, but overloaded UI	Processes requests quickly, uses two-factor authentication to fulfill the request, Before sending the amount, writes out the cardholder's first and last name when entering the card	Limited profile customisation options, needs to make a call to the operator to change user data or go to the bank	Contains English and Ukrainian languages
Raiffeisenbank	Simple navigation for first-time users	Modern design, lacks a variety of themes, limited information on the home page, no dark theme	Slow online service for document issuance, uses two-factor authentication to fulfill the request	Comprehensive profile interaction options	Contains English and Czech languages
Industrial bank	Prone to crashes when zooming in on a page, hard to figure out for the initial user, poorly implemented website functionality, complicate navigation for first-time users	Outdated interface, lacks a “dark theme”, lack of a "home" button, clicking on the site icon throws the user out of the account and requires re-entry of login and password, unprocessed user interface, which can leave the user confused.	Does not process requests quickly, uses two-factor authentication to fulfill the request	Limited profile customisation options	Only supports Ukrainian language. Lacks options for other languages

Figure 2.6: Conclusion table with the main factors of banking applications.

Chapter 3

Business Analysis

Business analytics describes who the target audience of the application is, what roles exist, their capabilities, how they can interact with the application, and a Use-Case diagram that visually illustrates that.

3.1 Target audience

The target audience of the application is users who will use the web version of the bank in order to manage their bank account. Since the server side of the application will be implemented, it also gives companies the opportunity to use this application for the purpose of implementing the web version of the bank.

3.2 Roles

The application will have several roles: User, Anonymous User, Moderator and Administrator. In this chapter it will be described in more detail how they can interact with the application.

3.2.1 User

User *(USER role)* has the right to manage his/her profile, open new accounts, make bank transactions, take out loans and repay them, deposit money, send messages to other users. A user cannot delete his/her card as long as there are funds on it. Also, the user cannot close his account. To do this, he/she must contact the administrator via chat or email with a request to close the account, after which the administrator will delete it.

3.2.2 Anonymous user

This type of user does not have access to account management and any other services provided by the web version of the bank. An anonymous user can only create an account. Upon creation, he/she is automatically given a card with the national currency.

3.2.3 Moderator

Moderator *(MODERATOR role)* has the ability to see users' accounts and edit them if necessary, has the ability to manage users' cards, loans, deposits and transfers, solve issues through internal chats or emails. Also moderator has the ability to set restrictions on access to users' bank accounts. Has the same capabilities as the **regular user**.

3.2.4 Administrator

The administrator *(ADMIN role)* has the ability to see user accounts and delete them if necessary. A user with the administrator role has the ability to set roles for other users, namely the moderator and administrator roles. He also has the ability to delete bank accounts if there are no funds in them, set restrictions on access to users' bank accounts but does not have access to their loans, and does not have access to users' loans and deposits data. Possesses incomplete **regular user** capabilities.

3.3 Use-Case diagram

The Use-Case diagram describes the existing roles in the application that are inherited from the user and also the anonymous user. [15]

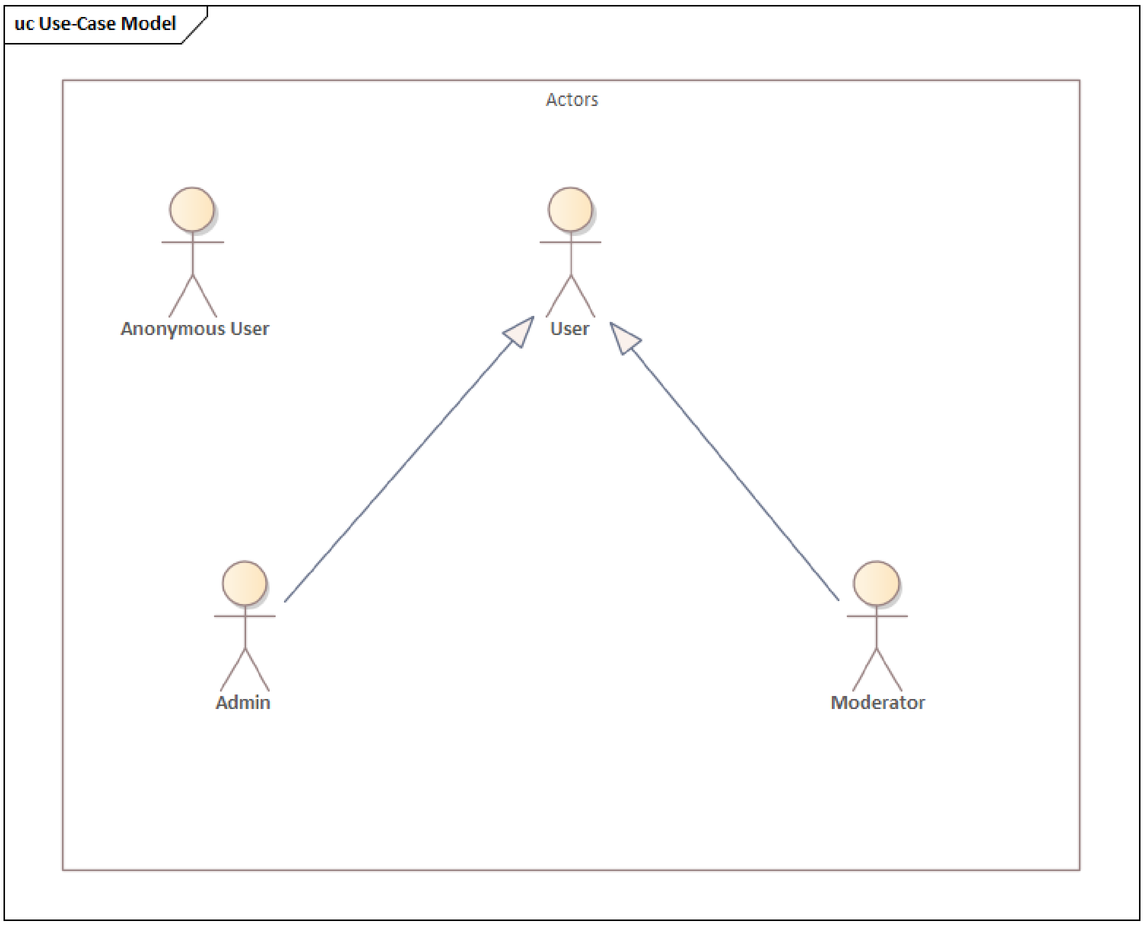


Figure 3.1: Use-Case diagram with actors

This Use-Case diagram describes the interactions between roles and the capabilities available to users with different roles.

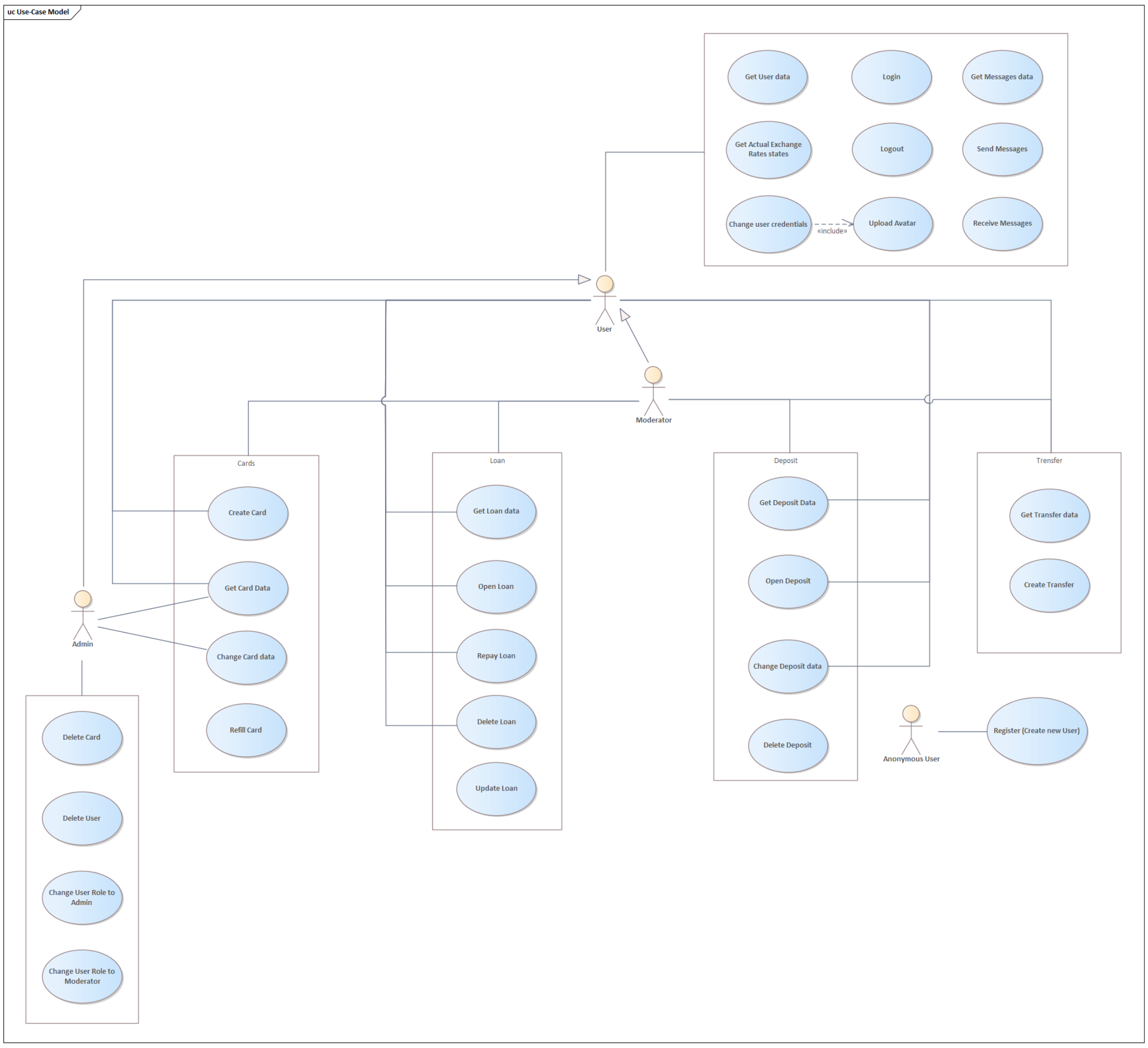


Figure 3.2: Use-Case diagram with interaction between actors

3.4 Conclusion

The Business Analysis chapter delves into defining the target audience, roles, and interactions within the application, alongside presenting a Use-Case diagram illustrating these aspects visually. By detailing user roles such as User, Anonymous User, Moderator, and Administrator, it delineates their respective capabilities and interactions. The Use-Case diagrams visually depict these interactions, offering a comprehensive understanding of the application's functionality and user roles.

Chapter 4

Functional and nonfunctional requirements

This chapter discusses the distinction between functional and non-functional requirements and their impact on the scope and functionality of the project.

4.0.1 Functional requirements

Functional requirements are the requirements that the end user has as core capabilities that the system must provide. All these functionalities must necessarily be included in the system as part of the contract. They are represented or formulated in terms of the input data to be fed into the system, the operations to be performed, and the expected output data. [16]

Based on the analyses that were made during the material review, the project will have the following functional requirements:

FR1: Registration

The program will allow the user to register and set personal information such as - name, surname, date of birth, country of origin, email, password and phone number. After registration user will be able to set his image profile.

FR2: Opening and maintenance of accounts

The program will allow the user to open accounts with different currencies and the ability to manage them, for example, to put on deposit, to withdraw funds from the card, to transfer funds to other accounts of the user.

FR3: Deletion of account

When deleting a bank account, if there is any amount left on it, the application will not allow the user to delete it. To do this, it is necessary to transfer money to another card, so that there are no funds on the account to be deleted.

FR4: Conversions

The main account is automatically opened in Czech crowns upon registration. Provided that the user opens a deposit, the amount transferred to the deposit, if the currency is foreign, will be converted to the currency of the deposit. The user also has the possibility to transfer the amount to his account with both national and foreign currencies. When transferring money to a foreign account, conversion will be done. A user cannot send an amount to another user's account if the currency of their account is not the same as the sender's currency account. Currency conversion will be set based on data coming from third-party API.

FR5: Delete profile

User will be able to delete his profile if he has no money in his account, no loan either on his profile or on his card, and no deposit opened. Also the deletion of the profile must be confirmed by the administrator.

FR6: Loan options

The system will allow the user to take a loan from the bank. The loan will be linked either to the bank account or to the user himself. When an amount is sent to the loan account, the loan will decrease and will soon automatically be removed.

FR7: Currency rate check

The user is able to monitor the exchange rate against the Czech crown.

FR8: Edit profile

User has the ability to edit their profile - photo, phone number, email and password. Changing the user's role can be done only from the administrator's side, as well as blocking and unblocking the user can be from the administrator's and moderator's side.

FR9: Write to other users

The user has the ability to communicate with other users using chat. Moderator and administrator also have such possibilities.

FR10: Account blocking

The administrator and moderator have the right to block the user's account. In this case, the user will not be able to authorize, delete the account and perform various operations. Also he/she will not be able to create a new account if he/she provides the same data as in the previous account.

4.0.2 Nonfunctional requirements

Non-functional requirements are qualitative constraints that the system must satisfy according to the project contract. The priority or degree of realization of these factors varies from project to project. They are also referred to as non-functional requirements. [16]

Based on the analyses that were made during the material review, the project will have the following non-functional requirements:

NFR1: Continuity

The developed program should function in a 24/7 activity mode, ensuring continuous operation without any scheduled or emergency application stops.

NFR2: Safety

The program must have an established authorization, which includes login and password. In this way the program protects the user from the third person.

NFR3: Performance and scalability

The program must be optimized for high performance and be able to scale with changing workloads. It should successfully handle a large number of users and its architecture should allow for efficient expansion when necessary.

NFR4: User application of the program

The program will be easy to use and will not require much effort from the user to understand the application.

NFR5: Testing

The program will be followed by testing to prevent any errors.

NFR6: Portability

The program will not be dependent on the operating system and can be run on other devices of different OS.

NFR7: Reliability and reusability

The program must have full access to the database of all users. The data itself must be reusable.

NFR8: Repairability

The system should be easily maintainable and capable of rapid recovery from system crashes.

NFR9: Flexibility

The program must be flexible and able to adapt to new technologies.

NFR10: Efficient resource management

The program must be able to effectively access and interact with resources.

4.1 Conclusion

The "Functional and Nonfunctional Requirements" chapter outlines the essential functionalities and qualitative constraints of the application. Functional requirements include vital features like user registration, account management, currency conversions, profile editing, and communication capabilities. Nonfunctional requirements emphasize continuous operation, safety measures, performance optimization, user-friendliness, testing protocols, portability, reliability, and flexibility. Together, these requirements shape the project's scope, defining its operational parameters and ensuring functionality and quality across various dimensions.

Chapter 5
Software Analysis

This section discusses the various technologies that will be used to implement this project and the reasons why they were chosen. Among the technologies used are discussed such as the database and programming language.

5.1 Database

A database is a set of data stored in a system and usually managed by a database management system. Databases are stored and in the form of tables, which enables efficient querying and processing of information. Structured Query Language (SQL) is used to manipulate the data. The database is an integral part of the project, providing the requested information when interacting with the application. For example, when an administrator wants to see all existing users, he accesses the database to display a list of all registered users. [17]

5.1.1 PostgreSQL

PostgreSQL is a powerful object-relational database system that is used in many web projects. The language has support for both SQL and JSON, which has gained more interest among developers. Among the many advantages of PostgreSQL can be mentioned reliability, data integrity, a wide range of functions, extensibility and much more. [18]

5.1.2 MySQL

MySQL is one of the most popular databases that is used in various companies such as Facebook, Netflix, Uber and others. MySQL is also an implementation database, where data is stored in separate tables. The main advantages are speed, reliability and easy to use. MySQL also has its disadvantages, such as: inefficiency with the work of very large data as well as in transaction processing, lack of support for SQL control limits, difficult scaling. [19] [20]

5.1.3 Oracle Database

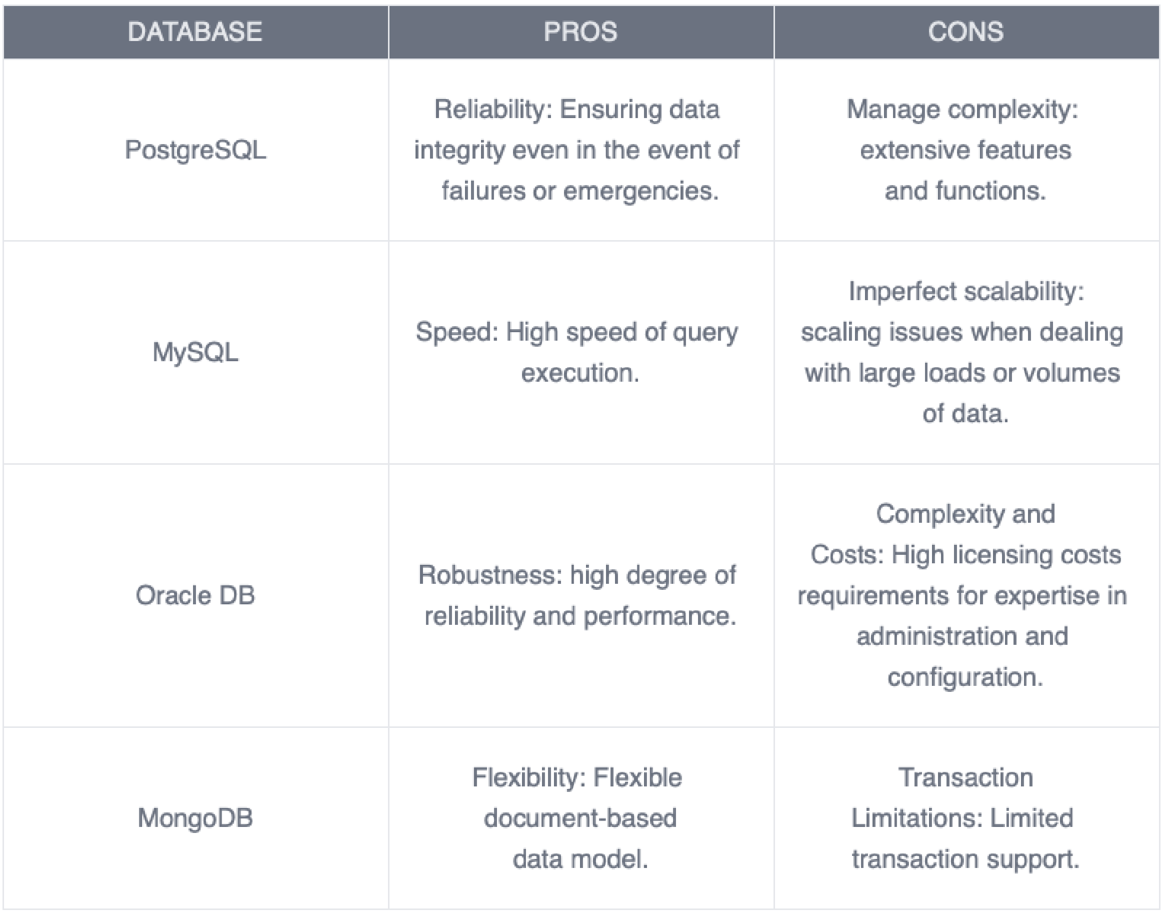
Oracle Database - A relational database management system and is one of the most robust and widely used relational databases for storing, organizing, and searching data by type while preserving relationships between different types. The disadvantages of using Oracle are the complexity of use with architectural and administrative costs, which can put a strain on the use of small Websites. [21]

5.1.4 MongoDB

MongoDB is a database program (tool) to manage document-oriented information, store or retrieve information and has a NoSQL type format. MongoDB is used for big data storage, indexing, load balancing, aggregation and more. MongoDB also has a number of drawbacks such as: limited transaction volume, limited join capabilities, data redundancy and memory use, document size limitation, and nested document layers. [22] [23]

5.1.5 Comparison of databases

Below can be seen a table that demonstrates database comparisons, their advantages and disadvantages.



DATABASE	PROS	CONS
PostgreSQL	Reliability: Ensuring data integrity even in the event of failures or emergencies.	Manage complexity: extensive features and functions.
MySQL	Speed: High speed of query execution.	Imperfect scalability: scaling issues when dealing with large loads or volumes of data.
Oracle DB	Robustness: high degree of reliability and performance.	Complexity and Costs: High licensing costs requirements for expertise in administration and configuration.
MongoDB	Flexibility: Flexible document-based data model.	Transaction Limitations: Limited transaction support.

Figure 5.1: Conclusion table with the main factors of databases

5.2 Backend

The backend is the server side of the website, which is the main part of the project. It stores and organizes data, and makes sure that everything on the client side of the website accomplishes its task. It is the part of the software that does not come in direct contact with the users. The parts and features designed by the backend designers are indirectly available to the users through the front-end application.

The backend is also the thread of communication between the client and the database, because the user, without access to the database, will not be able to get the requested information. Various programming languages are used for its implementation: Java, Kotlin, Python, PHP, C# and others.

5.2.1 Java

Java is a programming language and computing platform first released by Sun Microsystems in 1995 and then purchased by Oracle in 2010. This programming language is considered a dynamically typed language with a C-like syntax. It is a multi-paradigm language but is predominantly object-oriented, also a statically typed, cross-platform general-purpose programming language. The language is used to develop web websites, mobile applications, games, and more.

The main advantage of the Java language is its multiplatform nature, which allows user to run applications written in this language on any platform using JVM (Java Virtual Machine). [24]

5.2.2 Kotlin

Kotlin is a cross-platform, statically typed, general-purpose high-level programming language with type inference. The language appeared in 2011 and was developed by JetBrains. Kotlin is mainly JVM-oriented, but also compiles to JavaScript. Kotlin is a popular language due to its simplicity and efficiency. It has powerful tools for creating server-side, client-side, mobile as well as desktop applications. [25]

5.2.3 PHP

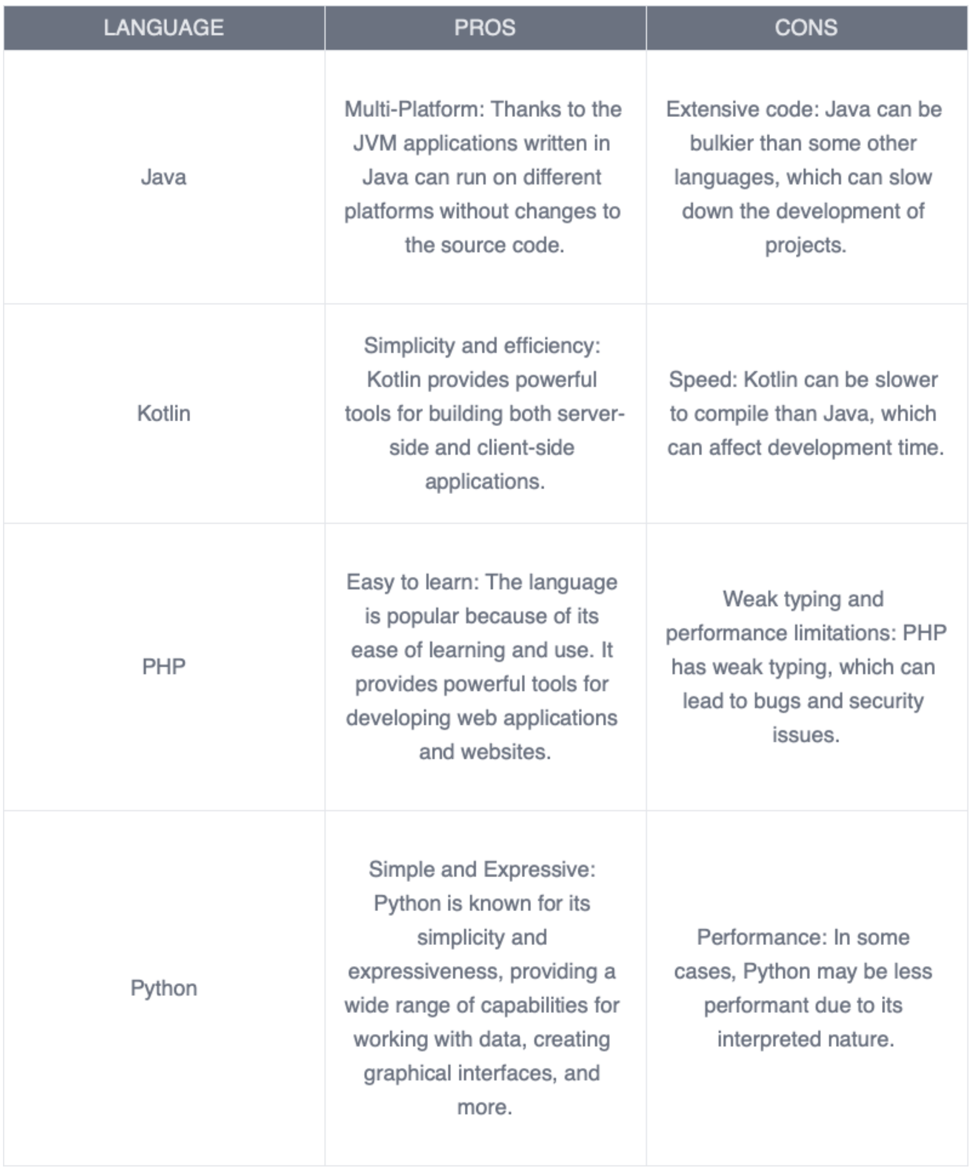
PHP is an open source server-side scripting language that many developers use for web development. The language was developed in 1995 by The PHP Group. PHP is a general-purpose language that can be used to create many projects, including graphical user interfaces (GUIs). PHP is one of the popular languages for beginner programmers due to its ease of learning and use. It provides powerful tools for processing forms, working with databases, creating dynamic web pages and much more. The language also has its drawbacks, such as weak typing, limitations in performance and scalability when working with large and complex applications. [26]

5.2.4 Python

Python is a high-level interpreted object-oriented programming language with dynamic semantics, created by the Python Software Foundation in 1991. Its built-in data structures, dynamic typing, garbage-collected and linking make it ideal for developing a variety of applications and use as a scripting language. It can be used for both functional and dynamic programming, making it unique. Python supports modules and packages, making it easy to compartmentalize programs and reuse code. [27] [28]

5.2.5 Comparison of languages

Below can be seen a table that demonstrates the comparison of programming languages, their advantages and disadvantages.



LANGUAGE	PROS	CONS
Java	Multi-Platform: Thanks to the JVM applications written in Java can run on different platforms without changes to the source code.	Extensive code: Java can be bulkier than some other languages, which can slow down the development of projects.
Kotlin	Simplicity and efficiency: Kotlin provides powerful tools for building both server-side and client-side applications.	Speed: Kotlin can be slower to compile than Java, which can affect development time.
PHP	Easy to learn: The language is popular because of its ease of learning and use. It provides powerful tools for developing web applications and websites.	Weak typing and performance limitations: PHP has weak typing, which can lead to bugs and security issues.
Python	Simple and Expressive: Python is known for its simplicity and expressiveness, providing a wide range of capabilities for working with data, creating graphical interfaces, and more.	Performance: In some cases, Python may be less performant due to its interpreted nature.

Figure 5.2: Conclusion table with the main factors of languages

5.3 Conclusion

After studying all available technologies, it was decided to use PostgreSQL as a database language and Java as a language for developing the server side of the application. PostgreSQL language was chosen because it is convenient and easy to use. The Java language was chosen because it is very popular for implementing the server side of projects, and it is also a clear and accessible programming language due to its syntax and support for object-oriented programming (OOP).

These technologies were chosen because of the experience gained in their use, and also because they were studied in detail during my university studies.

Chapter 6

Proposal

This section discusses the microservice, monolithic architecture, used APIs, Kafka, Docker Image and Hazelcast. Among other things, UML diagrams are shown to show what these entities will be and how they will interact with each other.

6.1 Architecture

The application architecture describes the system components, the relationships and interactions between them.

6.1.1 Microservice architecture

Microservices are a type of architecture that is used to implement distributed and loosely coupled applications, where changing one microservice will not suspend the entire application. A microservice consists of a single deployment that is independent of other deployments and processes that supports a specific business function. [29]

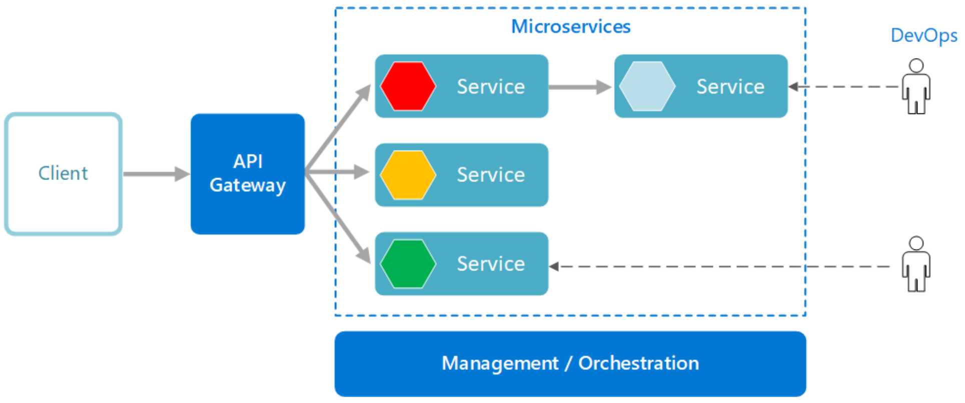


Figure 6.1: Demonstration of microservice architecture [29]

6.1.2 Monolithic architecture

Monolithic architecture is a traditional model of software that presents itself as a single network that integrates all business tasks. In order to make changes to an application with this architecture, user need to update the entire stack, access the database and after all the changes, run the project. [30]

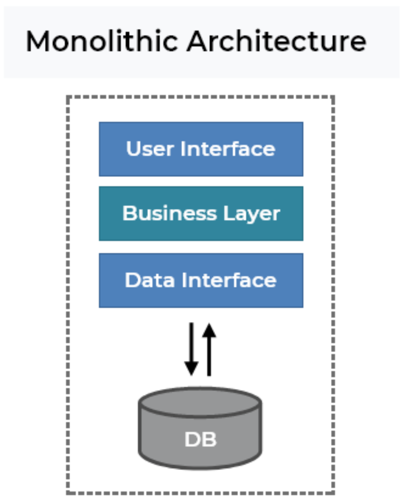


Figure 6.2: Demonstration of monolith architecture [31]

6.1.3 Deployment diagram

Deployment diagrams are diagrams that model the physical architecture of a system. Their main purpose is to show the relationships between the software and hardware components of the system and the distribution of processing between them. [32]

Below is a deployment diagram showing the basic process of communication between nodes.

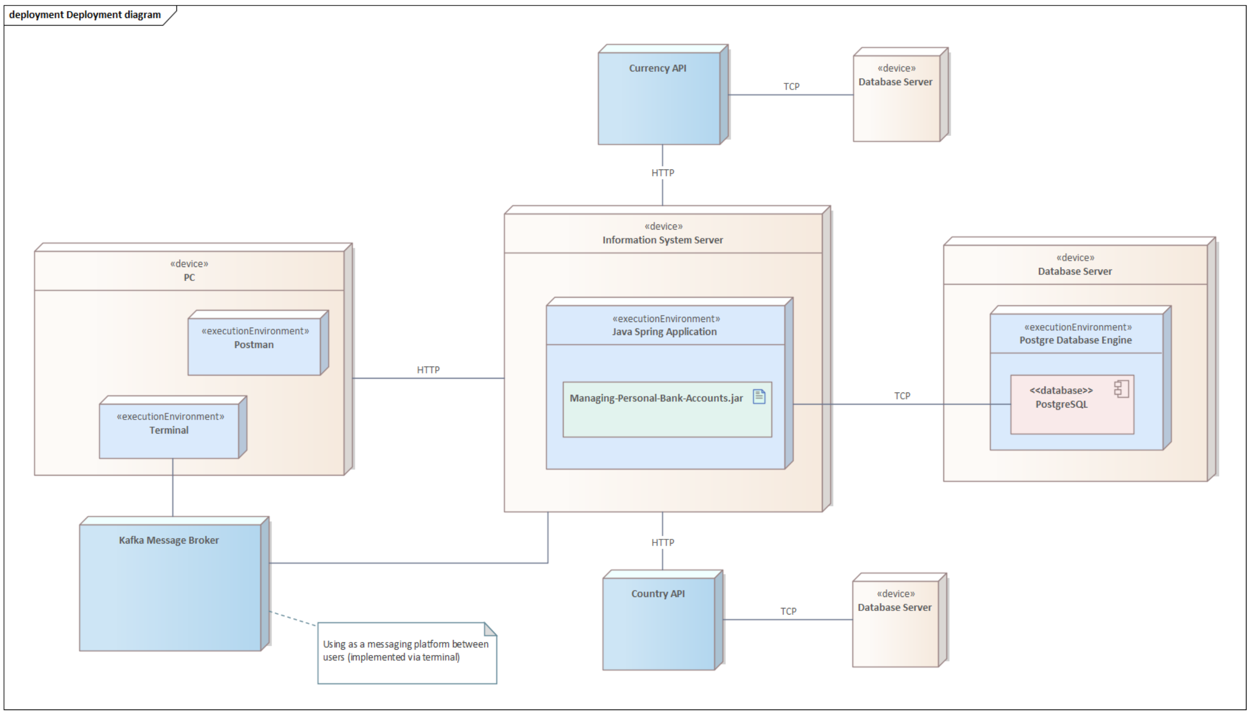


Figure 6.3: Deployment diagram

The first node in the diagram represents the client side of the application. It contains applications to run the necessary applications such as Terminal and Postman. Also for communication between users, the Kafka Message Broker will be used, which will work through the terminal and send messages to the server, which will then store the data in a database.

The second node represents the server part of the application. The server part is the main part of the application because it connects the user and the database. In addition, the server uses APIs such as *"ExchangeRate-API"* and *"Country API"* to retrieve data about exchange rates and existing countries. It also has a link to Kafka, through which it processes and stores messages in the server.

The third node is the database. It stores data about users, transactions, deposits, credits, etc. in the form of tables. When users interact with the application, they always access the server to retrieve, save, or modify data.

Below is a deployment diagram where two nodes are used. The first node is a PC, which represents the client side or user interface. The second node represents the Docker Image which contains a copy of the application in its deployment version. We also use Postman to interact and test the application.

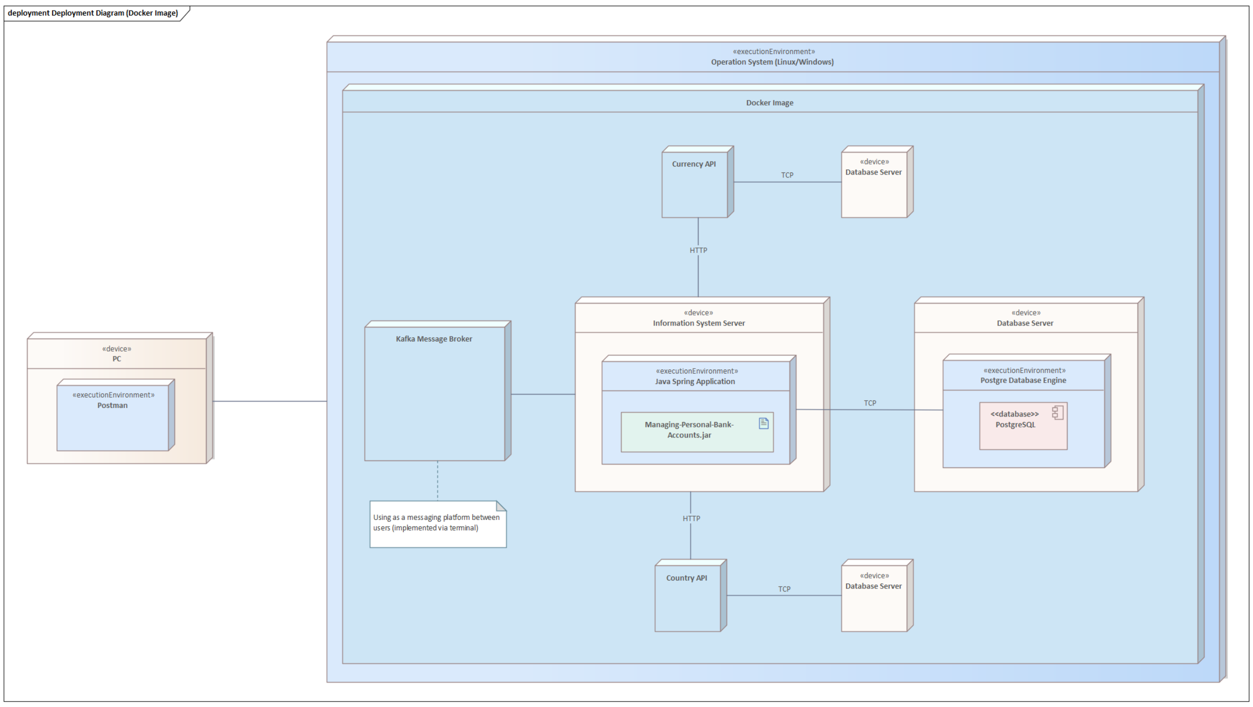

deployment Deployment Diagram (Docker Image)
«executionEnvironment» Operation System (Linux/Windows)
Docker Image
Currency API
TCP
«device» Database Server
HTTP
«device» PC
«executionEnvironment» Postman
Kafka Message Broker
«device» Information System Server
«executionEnvironment» Java Spring Application
Managing-Personal-Bank-Accounts.jar
TCP
«device» Database Server
«executionEnvironment» Postgre Database Engine
<<database>> PostgreSQL
Using as a messaging platform between users (implemented via terminal)
HTTP
Country API
TCP
«device» Database Server

Figure 6.4: Deployment diagram (Docker Image)

6.1.4 UML Diagram

The UML diagram is a graphical demonstration of the server-side concept, the relationships between classes, and what attributes they have.

In the diagram below it can be seen what classes are available in the project, as well as enumerative classes that will allow to set a status or process.

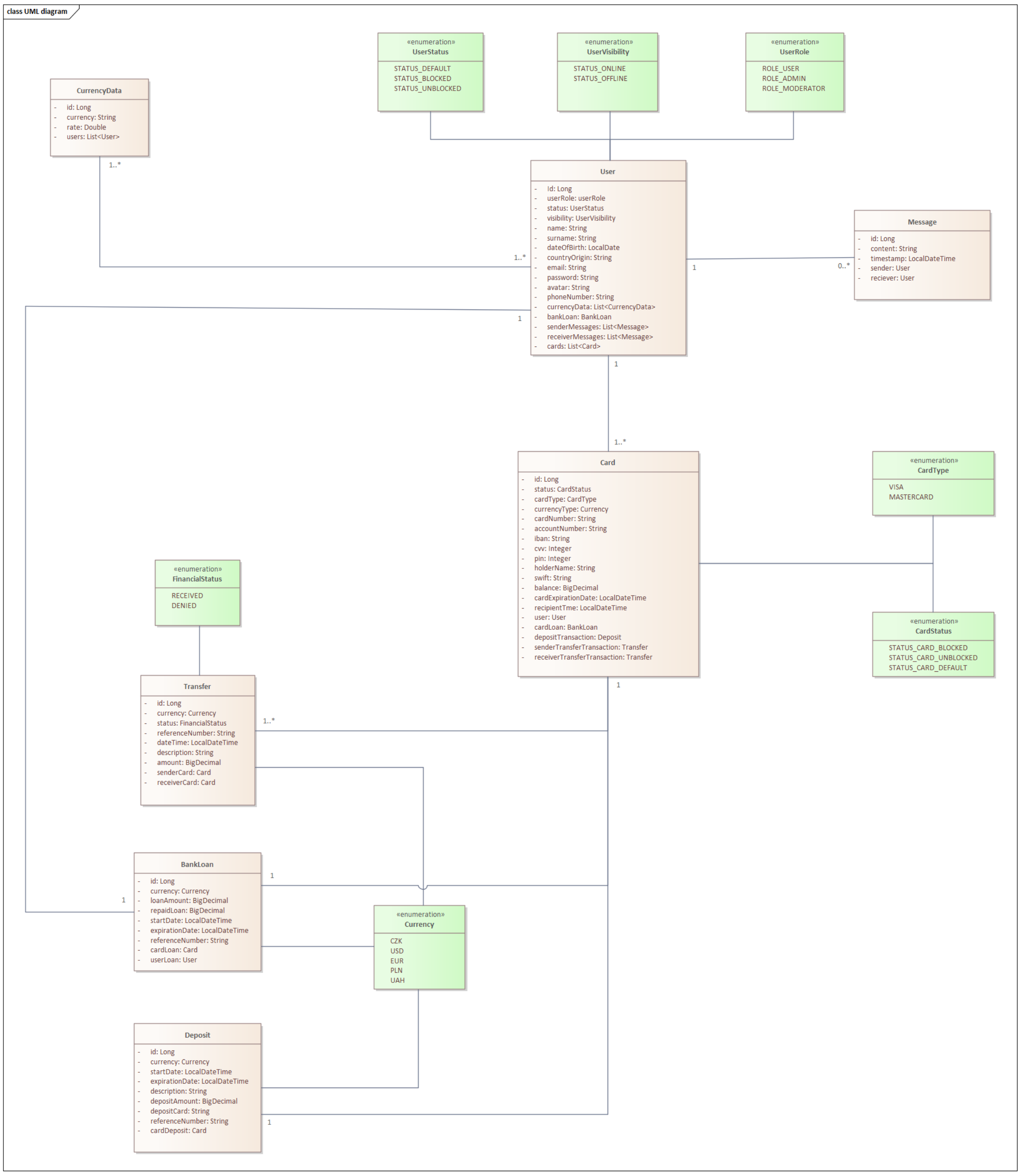


Figure 6.5: UML diagram

6.1.5 Server side

The server side of the application is written on the Spring Boot framework using the Java language. **Spring Boot framework** is a popular framework for developing the server side of web applications. It allows user to create a web application without much effort, so customizing and writing code seems to be easier. Spring Boot has a lot of functionality, among which its most significant features can be considered: dependency management and automatic configuration.

Below is a component diagram describing the interaction with the server side of the application.

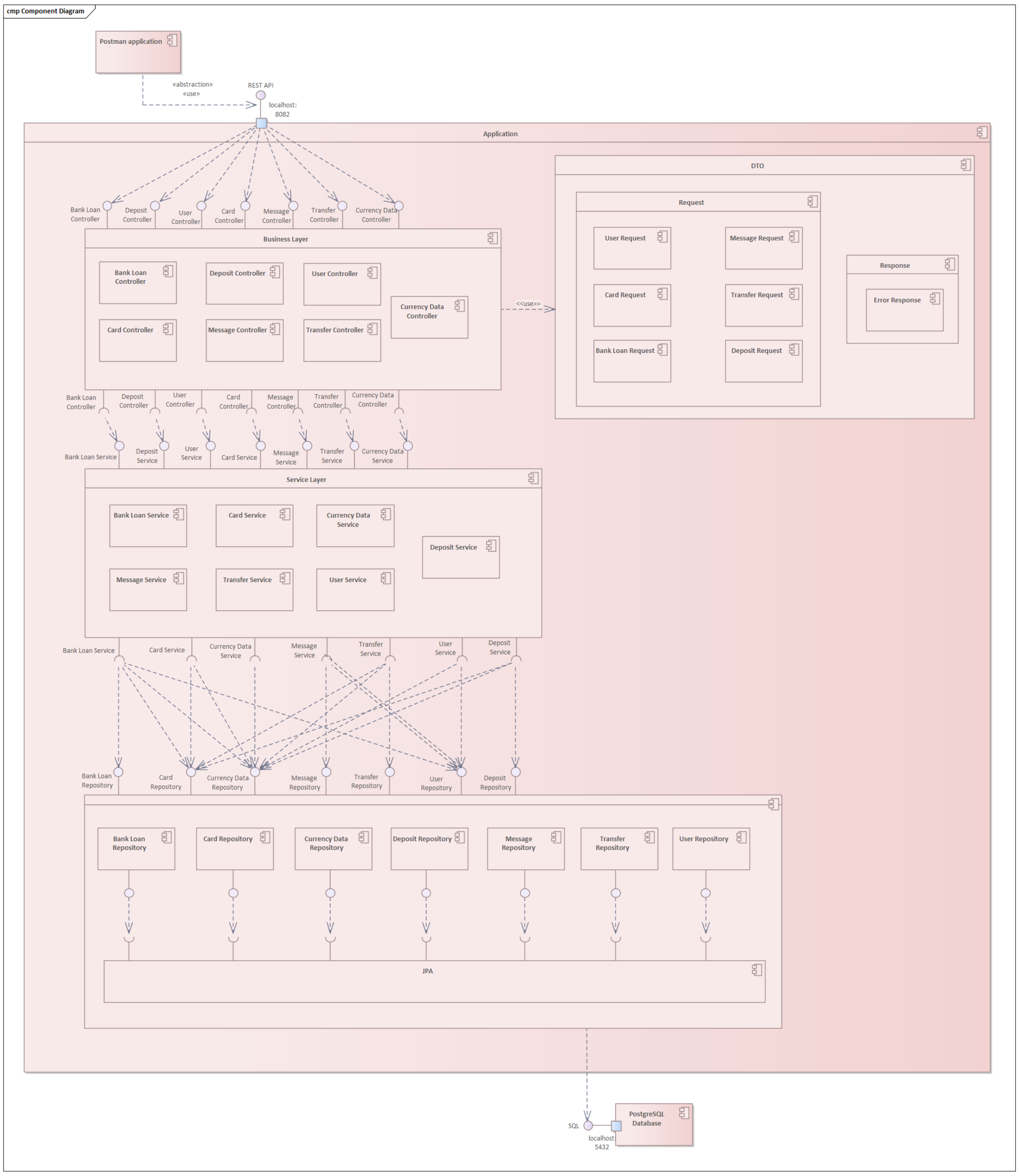


Figure 6.6: Component diagram

6.1.6 Security

Security is a major aspect in apps, especially when it comes to bank accounts. The application uses Spring Security. **Spring Security** is a unique framework that is used for authorization and authentication. Its benefit is also that it can be easily augmented with the desired functionality. The application uses **Basic Auth** for authentication.

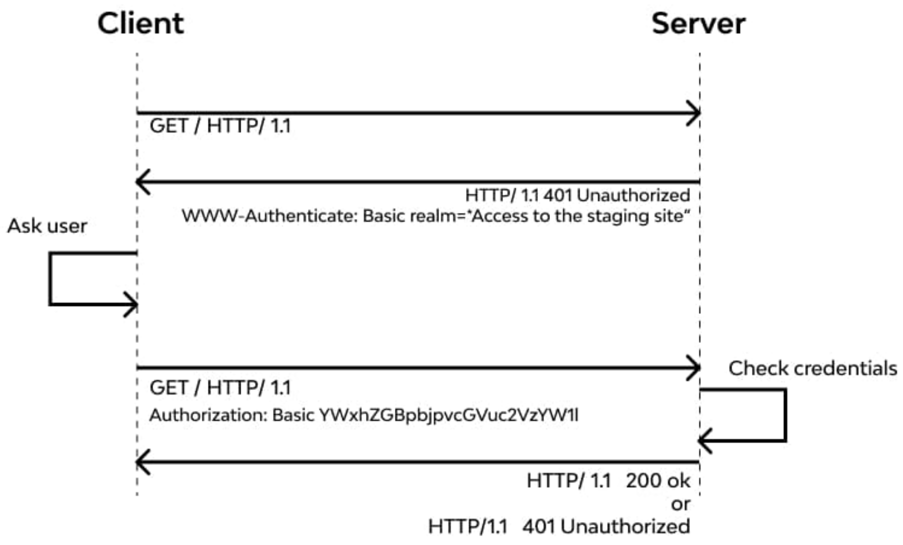
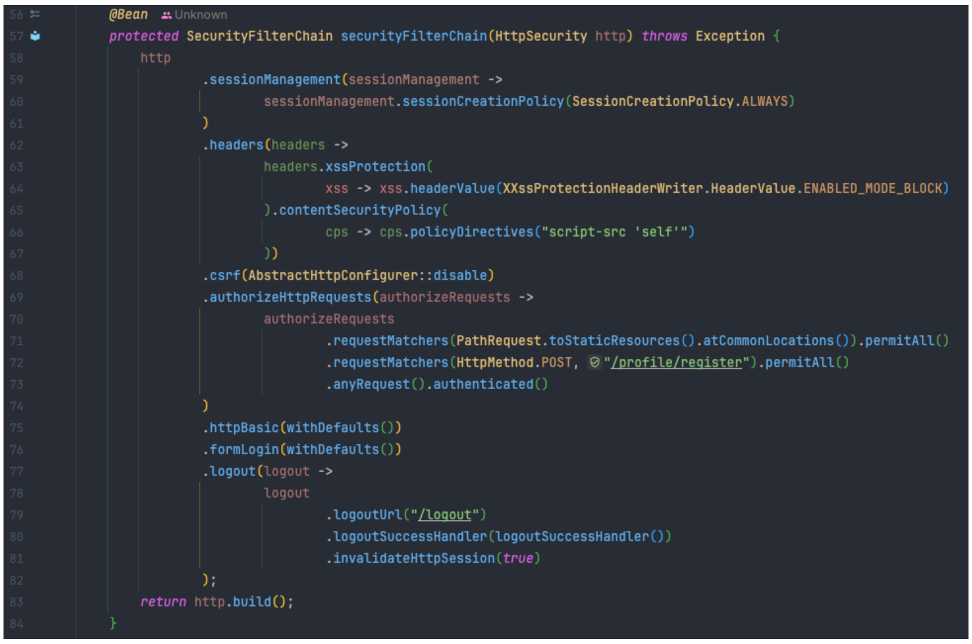

Client
Server
GET / HTTP/ 1.1
HTTP/ 1.1 401 Unauthorized
WWW-Authenticate: Basic realm="Access to the staging site"
Ask user
Check credentials
GET / HTTP/ 1.1
Authorization: Basic YWxhZGBpbjpvcGVuc2VzYW1l
HTTP/ 1.1 200 ok
or
HTTP/1.1 401 Unauthorized

Figure 6.7: Basic Auth diagram demonstration [33]

Basic Auth is one of the authentication methods and is a simple way to secure the REST API. It uses an HTTP header to pass the username and password when requesting a server. The disadvantage of Basic Auth is that it does not require the use of cookies to manage user sessions. This can affect application performance due to the need to re-authenticate each time a request is made. One way to solve this problem could be through the use of a session manager. By using a session manager, user can manage user sessions more efficiently and reduce the time spent authenticating and loading data to the page. [34]

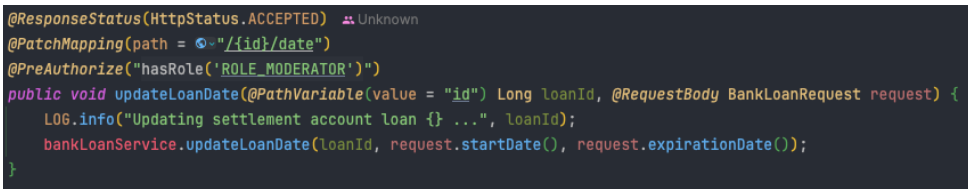
Below is the implementation of "Basic Auth" authentication in the application.



```java
@Bean  Unknown
protected SecurityFilterChain securityFilterChain(HttpSecurity http) throws Exception {
    http
            .sessionManagement(sessionManagement ->
                    sessionManagement.sessionCreationPolicy(SessionCreationPolicy.ALWAYS)
            )
            .headers(headers ->
                    headers.xssProtection(
                            xss -> xss.headerValue(XXssProtectionHeaderWriter.HeaderValue.ENABLED_MODE_BLOCK)
                    ).contentSecurityPolicy(
                            cps -> cps.policyDirectives("script-src 'self'")
                    ))
            .csrf(AbstractHttpConfigurer::disable)
            .authorizeHttpRequests(authorizeRequests ->
                    authorizeRequests
                            .requestMatchers(PathRequest.toStaticResources().atCommonLocations()).permitAll()
                            .requestMatchers(HttpMethod.POST, "/profile/register").permitAll()
                            .anyRequest().authenticated()
            )
            .httpBasic(withDefaults())
            .formLogin(withDefaults())
            .logout(logout ->
                    logout
                            .logoutUrl("/logout")
                            .logoutSuccessHandler(logoutSuccessHandler())
                            .invalidateHttpSession(true)
            );
    return http.build();
}
```

Figure 6.8: Security chain implementation demonstration

Also, in the application, users have been divided into roles, and only users with a specific role have access to certain methods. This can be achieved by using the *@PreAuthorize* annotation. Below is a demonstration of how this is implemented in the application.

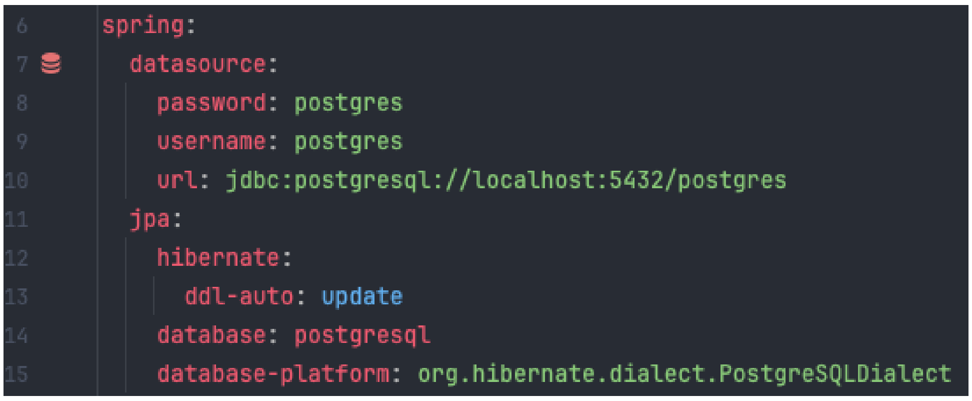


```java
@ResponseStatus(HttpStatus.ACCEPTED)  Unknown
@PatchMapping(path = "/{id}/date")
@PreAuthorize("hasRole('ROLE_MODERATOR')")
public void updateLoanDate(@PathVariable(value = "id") Long loanId, @RequestBody BankLoanRequest request) {
    LOG.info("Updating settlement account loan {} ...", loanId);
    bankLoanService.updateLoanDate(loanId, request.startDate(), request.expirationDate());
}
```

Figure 6.9: PreAuthorize implementation demonstration

6.1.7 Database

The PostgreSQL database was selected for the application development. This database is popular among users due to its powerful, open-source, object-relational database system. Additionally, it was studied and utilized during university courses, which provided a solid understanding of its syntax.
Below is a demonstration of how PostgreSQL is configured in the application.



```
spring:
  datasource:
    password: postgres
    username: postgres
    url: jdbc:postgresql://localhost:5432/postgres
  jpa:
    hibernate:
      ddl-auto: update
    database: postgresql
    database-platform: org.hibernate.dialect.PostgreSQLDialect
```

Figure 6.10: PostgreSQL configuration for accessing to the database

6.1.8 Hibernate

Hibernate is a Java framework whose goal is to simplify database relationships and the use of data in code. Hibernate resides in the freely available Object Relational Mapping tool, or ORM for short. Hibernate allows user to write less code to manipulate the database, making the code more readable. It is also part of the Java Persistence API (JPA) implementing its specifications for data persistence. [35]

6.1.9 JPA

The Java Persistence API (JPA) is a Java language specification that is used to persist Java objects in a relational database. In this way, JPA can be referred to as the node that connects databases to our code. ORM tools such as Hibernate, which implement the JPA specifics to persist data in the database. [36]

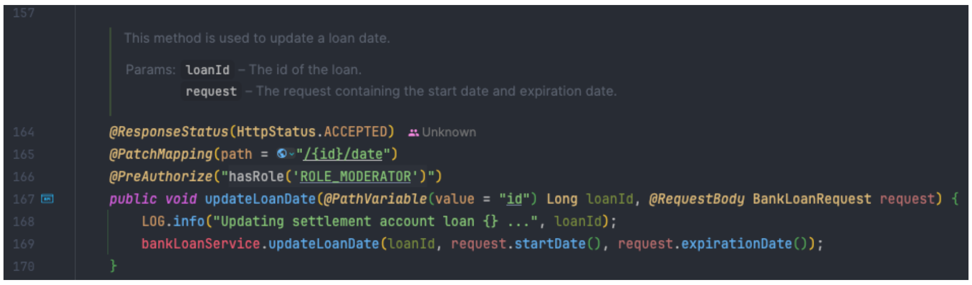
6.1.10 Maven

Maven is a powerful tool that allows user to manage projects. Its functionality includes such mechanics as building projects, working with validation, cleaning up unnecessary dependencies, rebuilding files, testing, building into a package, installing new dependencies and much more. Maven makes the daily work of Java developers easier and generally helps to understand the essence of any Java project. However, it is also worth mentioning that Maven is an important part of the project, without which it will be impossible to run the application. Maven allows user to use it as a controller to control a Java application. [37]

6.1.11 REST API

REST (Representational state transfer) is a style of software architecture that is used to create APIs (Application Persistence Interface) in the modern world. REST allows user to use information using HTTP requests, thanks to which user can call information, write it to a database, modify and delete it.

The most common HTTP requests used for manipulation are GET, POST, PATCH, PUT, DELETE. Each piece of information is uniquely defined by a global identifier such as a URL. Each URL in turn has a strictly defined format. Below is an example from an application where an HTTP request is used to perform manipulations on data in a database.



```
    This method is used to update a loan date.
    Params: loanId – The id of the loan.
            request – The request containing the start date and expiration date.

@ResponseStatus(HttpStatus.ACCEPTED)  Unknown
@PatchMapping(path = "/{id}/date")
@PreAuthorize("hasRole('ROLE_MODERATOR')")
public void updateLoanDate(@PathVariable(value = "id") Long loanId, @RequestBody BankLoanRequest request) {
    LOG.info("Updating settlement account loan {} ...", loanId);
    bankLoanService.updateLoanDate(loanId, request.startDate(), request.expirationDate());
}
```

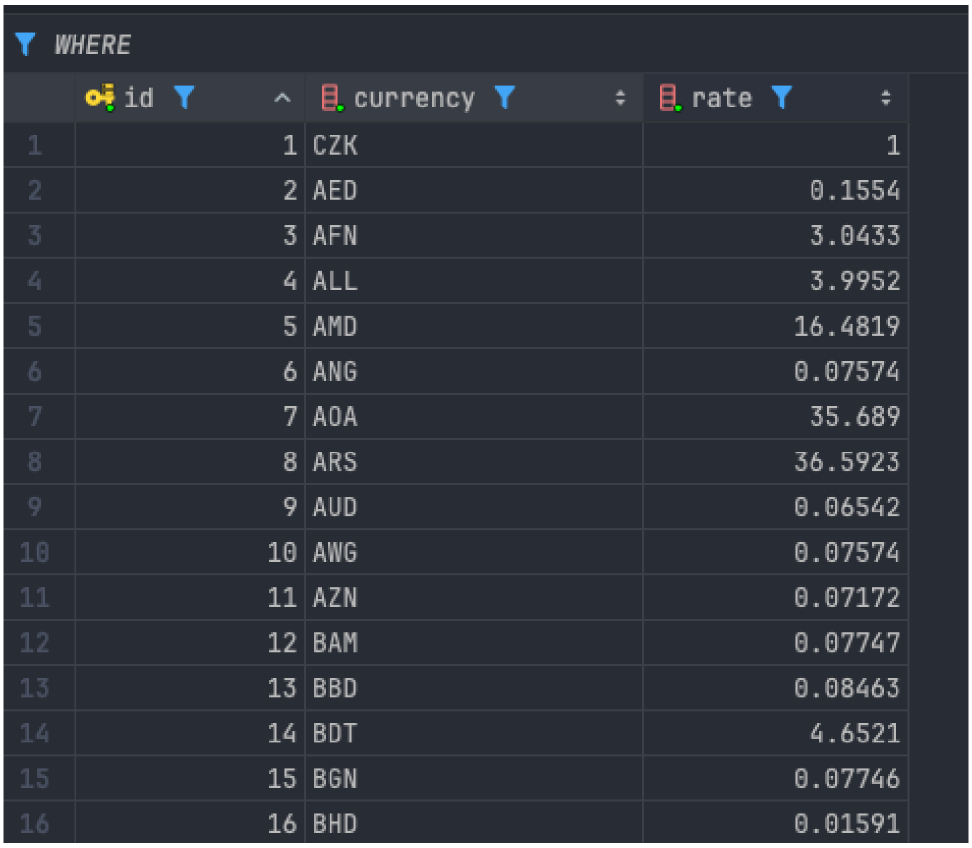
Figure 6.11: Demonstration of REST call in the application

6.2 Used Third-party API's

Third-party APIs were used in the application, through which the application can get up-to-date information about currencies and existing countries.

6.2.1 Currency API

Currency API (or *"ExchangeRate-API"*) is a free, publicly available API that allows user to get up-to-date currency rates from over 162 countries around the world. In the application the main currency is Czech crown, so rates of other currencies will refer to Czech crown, so for example 1 crown is 0.04232 dollars. In the application, the exchange rate is updated every 24 hours so that the user can see the current exchange rate today. Below user can see how information about currency rates is stored in the application database. [38]



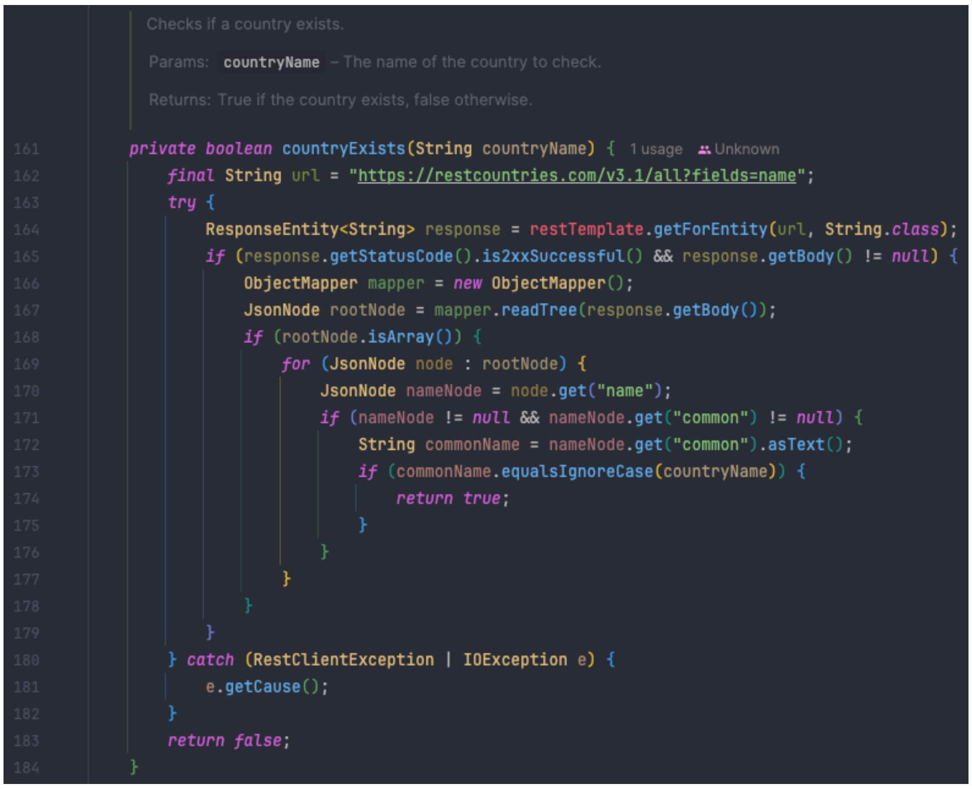
WHERE

	id	currency	rate
1	1	CZK	1
2	2	AED	0.1554
3	3	AFN	3.0433
4	4	ALL	3.9952
5	5	AMD	16.4819
6	6	ANG	0.07574
7	7	AOA	35.689
8	8	ARS	36.5923
9	9	AUD	0.06542
10	10	AWG	0.07574
11	11	AZN	0.07172
12	12	BAM	0.07747
13	13	BBD	0.08463
14	14	BDT	4.6521
15	15	BGN	0.07746
16	16	BHD	0.01591

Figure 6.12: Demonstration of the ExhcangeRate-API usage

6.2.2 Country API

The *"REST Countries API"* is used to verify that a user has entered an existing country when registering. Below is a demonstration of how the country entered by the user is validated in the application. [39]

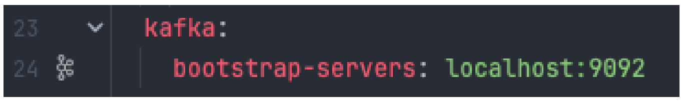


```
Checks if a country exists.
Params: countryName – The name of the country to check.
Returns: True if the country exists, false otherwise.
private boolean countryExists(String countryName) {  1 usage  Unknown
    final String url = "https://restcountries.com/v3.1/all?fields=name";
    try {
        ResponseEntity<String> response = restTemplate.getForEntity(url, String.class);
        if (response.getStatusCode().is2xxSuccessful() && response.getBody() != null) {
            ObjectMapper mapper = new ObjectMapper();
            JsonNode rootNode = mapper.readTree(response.getBody());
            if (rootNode.isArray()) {
                for (JsonNode node : rootNode) {
                    JsonNode nameNode = node.get("name");
                    if (nameNode != null && nameNode.get("common") != null) {
                        String commonName = nameNode.get("common").asText();
                        if (commonName.equalsIgnoreCase(countryName)) {
                            return true;
                        }
                    }
                }
            }
        }
    } catch (RestClientException | IOException e) {
        e.getCause();
    }
    return false;
}
```

Figure 6.13: Demonstration of the REST Countries API usage

6.3 Apache Kafka

Apache Kafka is a distributed platform for streaming data that is continuously generated during real-time use of an application. The application uses Apache Kafka as a robust and scalable Message Broker that enables messaging between users. The use of the customized configuration for Kafka in application.yml file is demonstrated below. [40]

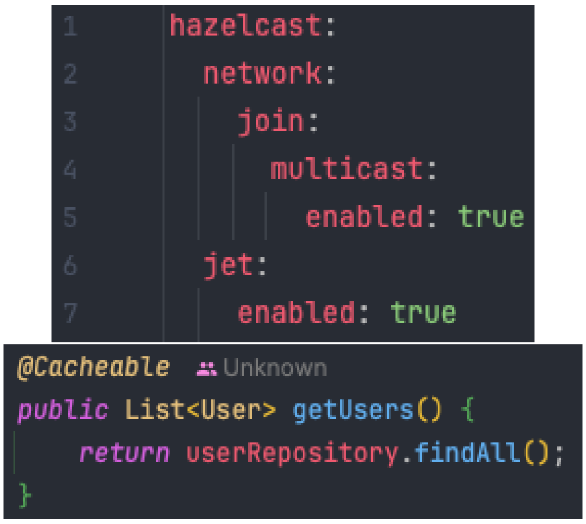


```
kafka:
  bootstrap-servers: localhost:9092
```

Figure 6.14: Demonstration of the Apache Kafka configuration

6.4 Hazelcast

Hazelcast is a distributed In-Memory Data Grid platform for Java. In an application, Hazelcast acts as query caching, allowing users to retrieve data instantly when the data is re-processed, without having to execute the query again. Below is an example of the configuration that can be found in the hazelcast.yml file, as well as a demonstration of the use of the caching annotations that Hazelcast works with. [41]



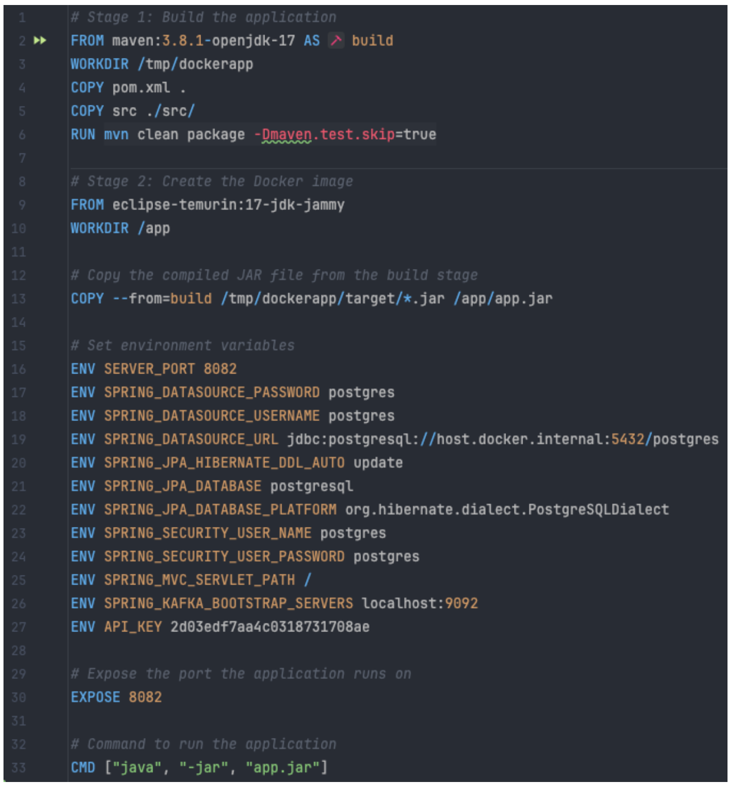
```
hazelcast:
  network:
    join:
      multicast:
        enabled: true
  jet:
    enabled: true
```

```
@Cacheable  Unknown
public List<User> getUsers() {
    return userRepository.findAll();
}
```

Figure 6.15: Demonstration of the Hazelcast configuration

6.5 Docker Image

A Docker image is a file that is used to execute code in a Docker container. Docker images act as a set of instructions to create a Docker container, similar to a template. In a Docker image file, we specify the commands to build the application, create the Docker image, specify variables for environment, port, etc. Below is a Dockerfile that runs the commands described above. [42]



```
# Stage 1: Build the application
FROM maven:3.8.1-openjdk-17 AS build
WORKDIR /tmp/dockerapp
COPY pom.xml .
COPY src ./src/
RUN mvn clean package -Dmaven.test.skip=true

# Stage 2: Create the Docker image
FROM eclipse-temurin:17-jdk-jammy
WORKDIR /app

# Copy the compiled JAR file from the build stage
COPY --from=build /tmp/dockerapp/target/*.jar /app/app.jar

# Set environment variables
ENV SERVER_PORT 8082
ENV SPRING_DATASOURCE_PASSWORD postgres
ENV SPRING_DATASOURCE_USERNAME postgres
ENV SPRING_DATASOURCE_URL jdbc:postgresql://host.docker.internal:5432/postgres
ENV SPRING_JPA_HIBERNATE_DDL_AUTO update
ENV SPRING_JPA_DATABASE postgresql
ENV SPRING_JPA_DATABASE_PLATFORM org.hibernate.dialect.PostgreSQLDialect
ENV SPRING_SECURITY_USER_NAME postgres
ENV SPRING_SECURITY_USER_PASSWORD postgres
ENV SPRING_MVC_SERVLET_PATH /
ENV SPRING_KAFKA_BOOTSTRAP_SERVERS localhost:9092
ENV API_KEY 2d03edf7aa4c0318731708ae

# Expose the port the application runs on
EXPOSE 8082

# Command to run the application
CMD ["java", "-jar", "app.jar"]
```

Figure 6.16: Demonstration of Docker Image

6.6 Conclusion

In this chapter, all technologies and frameworks used in the application were analyzed. After studying this title it is possible to conclude why these technologies and frameworks were used, what are their advantages and what role they play in the application.

Chapter 7
Implementation

This chapter will describe the development of the application, what framework is used and the structure of the application.

7.1 Application implementation

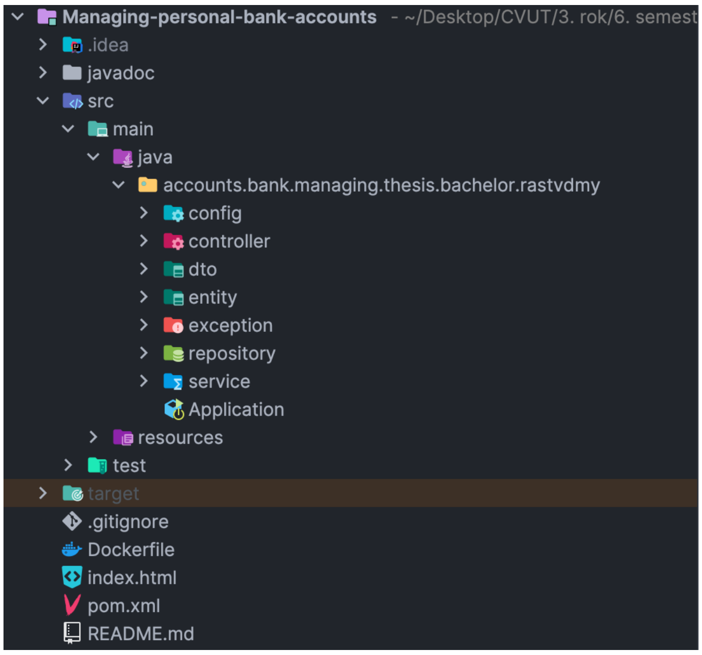
The application itself is the server side of the web application, which plays a very important role as the information that is manipulated will later be written out in the client side of the application.

7.1.1 Spring Boot

Spring Boot is an open source framework that makes it easy to build web applications. It provides user-friendly tools and automated configurations, which makes the development process fast and enjoyable. Spring Boot is widely used in the industry and is a popular choice among web application developers. It supports not only Java but also more modern programming languages such as Kotlin and Scala, allowing user to choose the right tool for each project.

7.1.2 Application structure

The main files are located in the folder
src/main/java/accounts/bank/managing/thesis/bachelor/rastvdmy/.
During development, the code was structured and divided into several folders, each of which contains a specific part of the application responsible for a particular program process.
Below demonstrates the structure of the application.



```
Managing-personal-bank-accounts - ~/Desktop/CVUT/3. rok/6. semest
  .idea
  javadoc
  src
    main
      java
        accounts.bank.managing.thesis.bachelor.rastvdmy
          config
          controller
          dto
          entity
          exception
          repository
          service
          Application
      resources
    test
  target
  .gitignore
  Dockerfile
  index.html
  pom.xml
  README.md
```

Figure 7.1: Demonstration of the application structure

/config/ - This folder contains the configurations needed to initialize the application, run Kafka, and install security. Also inside this folder there is a folder */config/utils/*, which contains Kafka listeners - a component acting as a listener, thanks to which it is possible to see the status of the message when communicating.

/controller/ - A folder that stores files used to manipulate data by performing certain operations defined in REST controllers and displaying them on the screen. Inside this folder user can also find the folder */controller/handler/*, which contains the Error handler, which when processing messages, in case of error will display the page with errors, which are located in the folder */resources/templates/*.

/dto/ - The application uses two types of data transfer objects, known as DTO, provided in the dto folder. A DTO is the process by which the necessary data is transferred between the client and server parts of an application. */dto/request/* - Responsible for transferring data from the client part to the server part. This type of data transfer object is used to send requests to the server and transfer the data for processing.
/dto/response/ - Responsible for transferring data from the server part to the client part. This type of data transfer object is used to send responses

from the server to the client and transfer the data for display to the user.

Each type of DTO is represented in a corresponding class within the specified folder.

/entity/ - This folder contains entities that are a directly important part of the application. As the application runs, there will be constant interaction with the entities, including assigning new roles, deleting, modifying, and other operations.

/exception/ - The exception folder contains the application's error class, which is used to write an error to the screen if the user for example makes a mistake while entering data.

/repository/ - This folder contains files through which it is possible to manipulate the database, namely to retrieve certain data, to save and delete from the database, and to modify data. The folder links the business logic with the database logic.

/service/ - This folder contains the files that implement the business logic. In this folder, services interact with entities, controllers, errors and repositories, thus making themselves the "heart" of the application. The business logic implements such details as: getting up-to-date exchange rates, transferring money, opening a deposit, creating a profile, opening a new account, and more. This folder also contains the */service/component/* folder, which stores: The *Administrator Initializer* - every time user try to create a database, an administrator is automatically created, as well as the *Generator* - which stores the logic for generating the necessary data for the map, namely: IBAN, Swift, account number and reference number.

/resources/ - This folder contains the application configuration files, the Hazelcast hashing configuration, and the */resources/templates/* folder, which contains the files the user sees when errors are displayed.

/test/ - The tests folder contains files that test the application before launching it so that there are no unnecessary errors.

7.2 Conclusion

The "Implementation" chapter outlines the development process, highlighting the utilization of Spring Boot for web application development. It provides a comprehensive overview of the application's structure, detailing each folder's purpose and functionality. Key components such as configurations, controllers, DTOs, entities, exceptions, repositories, services, and test files are discussed, underscoring the meticulous organization of the codebase. This chapter underscores the development effort and emphasizes the significance of the application's structure and core components.

Chapter 8

Development environment

This chapter will describe all the development environments used to create the application.

8.1 IntelliJ IDEA Ultimate

IntelliJ IDEA is an integrated development environment (IDE) developed by JetBrains that is used for application development. IntelliJ allows user to write applications in different languages such as Java, Kotlin, Scala, JavaScript and many others, which makes it unique. Also the main feature is plugins, which are unique tools that allow developers to make IntelliJ a unique sandbox. [43]

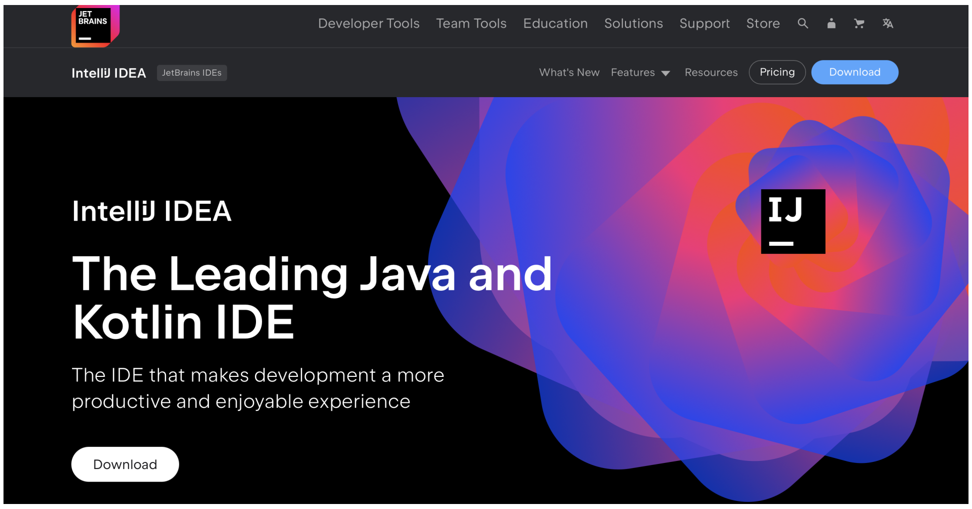
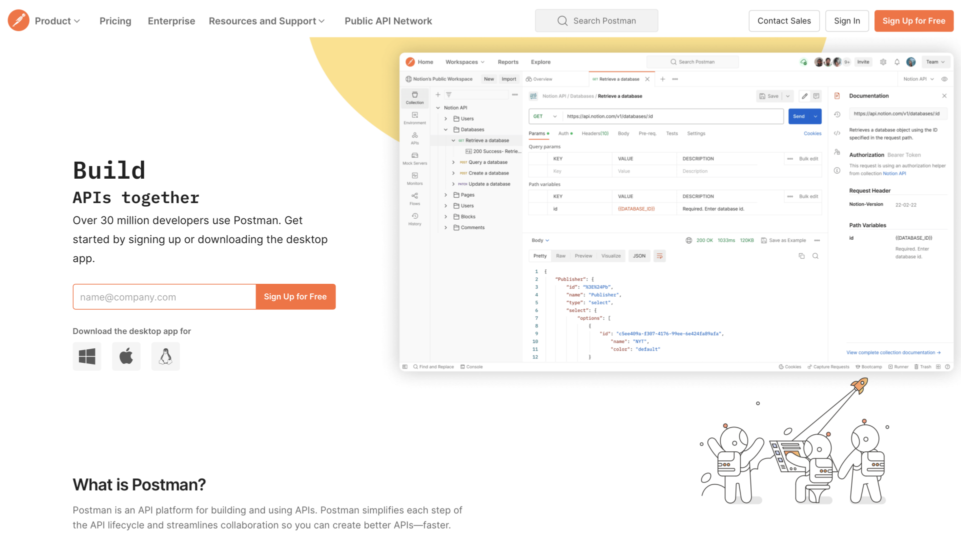


Figure 8.1: IntelliJ IDEA web page

8.2 Postman

Postman is a platform developed by Postdot Technologies for creating and using APIs. It allows users to test the server side of an application by creating different REST requests, facilitating interaction with both the server side and the database. Additionally, Postman is a unique tool for creating REST API documentation, enabling the generation of documentation once all requests have been written. [44]



Build
APIs together

Over 30 million developers use Postman. Get started by signing up or downloading the desktop app.

name@company.com

https://api.notion.com/v1/databases/:id

Query params

KEY	VALUE	DESCRIPTION
Key	Value	Description

Path variables

KEY	VALUE	DESCRIPTION
id	{{DATABASE_ID}}	Required. Enter database id.

```
{
    "Publisher": {
        "id": "%3E%24Pb",
        "name": "Publisher",
        "type": "select",
        "select": {
            "options": [
                {
                    "id": "c5ee409a-f307-4176-99ee-6e424fa89afa",
                    "name": "NYT",
                    "color": "default"
                }
```

https://api.notion.com/v1/databases/:id

Retrieves a database object using the ID specified in the request path.

Authorization Bearer Token

This request is using an authorization helper from collection Notion API

Request Header

Notion-Version 22-02-22

Path Variables

id {{DATABASE_ID}} Required. Enter database id.

What is Postman?

Postman is an API platform for building and using APIs. Postman simplifies each step of the API lifecycle and streamlines collaboration so you can create better APIs—faster.

Figure 8.2: Postman web page

8.3 Docker

Docker is an open platform for developing, delivering, and running applications. It allows applications to be decoupled from the infrastructure, enabling faster software delivery. In an application, Docker is used to create images that can be executed independently. Docker includes builds that can run on their own, independent of the underlying application. [45]

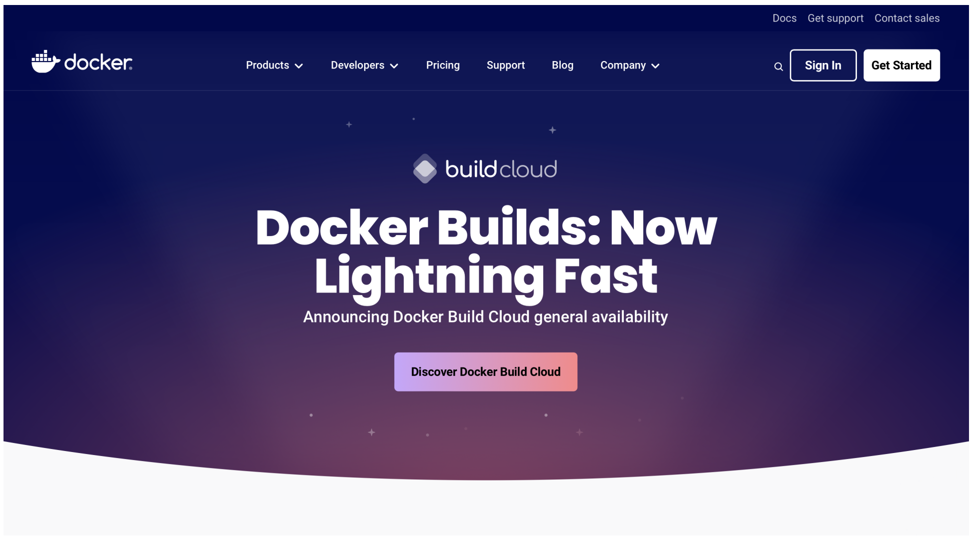

Docs
Get support
Contact sales
docker
Products
Developers
Pricing
Support
Blog
Company
Sign In
Get Started
build cloud
Docker Builds: Now Lightning Fast
Announcing Docker Build Cloud general availability
Discover Docker Build Cloud

Figure 8.3: Docker web page

8.4 Conclusion

The "Development Environment" chapter highlights the key tools used in application development. IntelliJ IDEA Ultimate provides a robust integrated development environment supporting various languages. Postman simplifies API testing and documentation, while Docker enables efficient application development, delivery, and execution. These tools collectively streamline the development process, emphasizing efficiency and productivity.

Chapter 9

Versioning

Project versioning is an important part of project implementation, because with its help you can define the project status, dividing each change in the project into bug fixes, minor changes and major changes. Project versioning is also an important part of team-based application development, because this way we can divide the work between each team member, specifying their own branch version, and after implementation and validation, merge it with the main version.

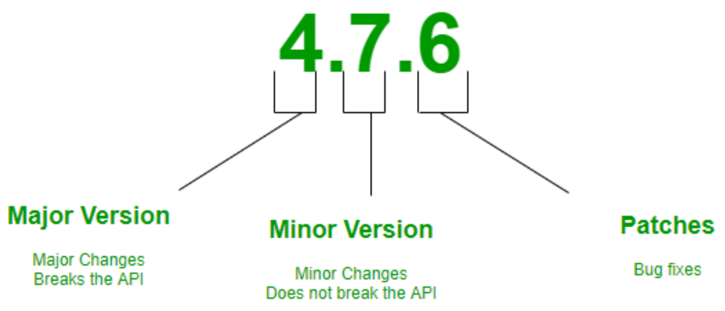
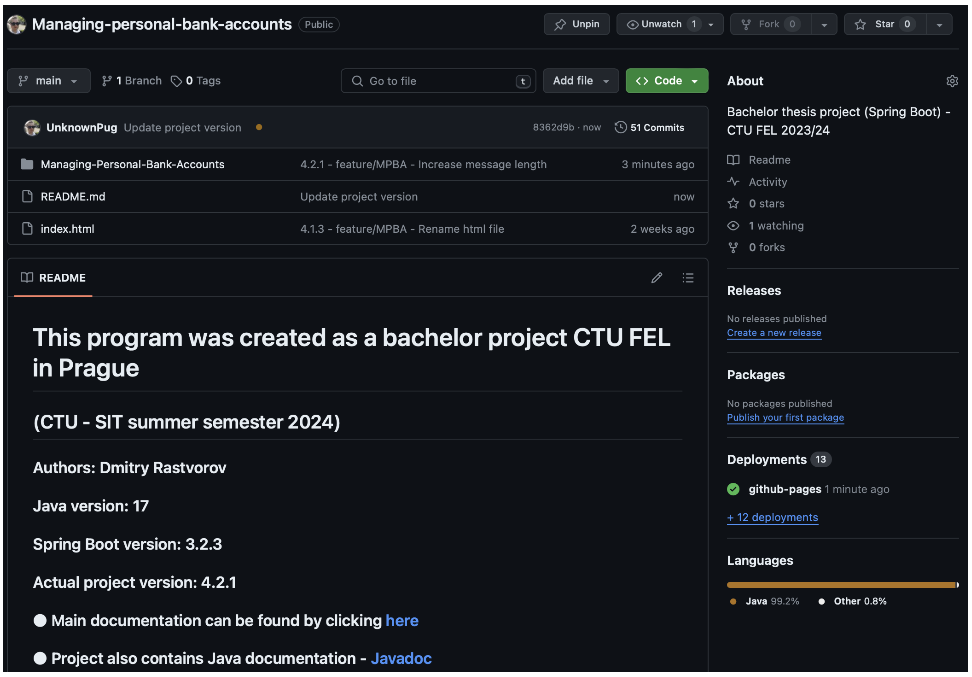


Figure 9.1: Demonstration of an example of project versioning [46]

9.1 GitHub

GitHub is a free web service that acts as a repository for open source IT projects as well as a service for collaborative application development. With GitHub, users can share projects, manage them, change access to them, and much more. At the moment, this web service is the most popular among developers. Due to the fact that GitHub supports the distributed version

control system git, it became possible to use the knowledge gained during university studies, namely creating separate branches, merging them and much more. GitHub's unique functionality, GitHub Pages, was also utilized. GitHub Pages allows the user to create a static site that uses code written in the application. Thus, it was possible to realize Java documentation of the project and publish it as a web page.
Here is link to the project repository https://github.com/UnknownPug/Managing-personal-bank-accounts, which is in *public state*. Below is a picture of what the repository page of the Managing Personal Bank Accounts application looks like.



UnknownPug Update project version · 8362d9b · now · 51 Commits

Managing-Personal-Bank-Accounts	4.2.1 - feature/MPBA - Increase message length	3 minutes ago
README.md	Update project version	now
index.html	4.1.3 - feature/MPBA - Rename html file	2 weeks ago

This program was created as a bachelor project CTU FEL in Prague

(CTU - SIT summer semester 2024)

Authors: Dmitry Rastvorov

Java version: 17

Spring Boot version: 3.2.3

Actual project version: 4.2.1

- Main documentation can be found by clicking here
- Project also contains Java documentation - Javadoc

About

Bachelor thesis project (Spring Boot) - CTU FEL 2023/24

Readme
Activity
0 stars
1 watching
0 forks

Releases

No releases published
Create a new release

Packages

No packages published
Publish your first package

Deployments 13

github-pages 1 minute ago

+ 12 deployments

Languages

Java 99.2% · Other 0.8%

Figure 9.2: Demonstration of project repository located in GitHub

9.2 Conclusion

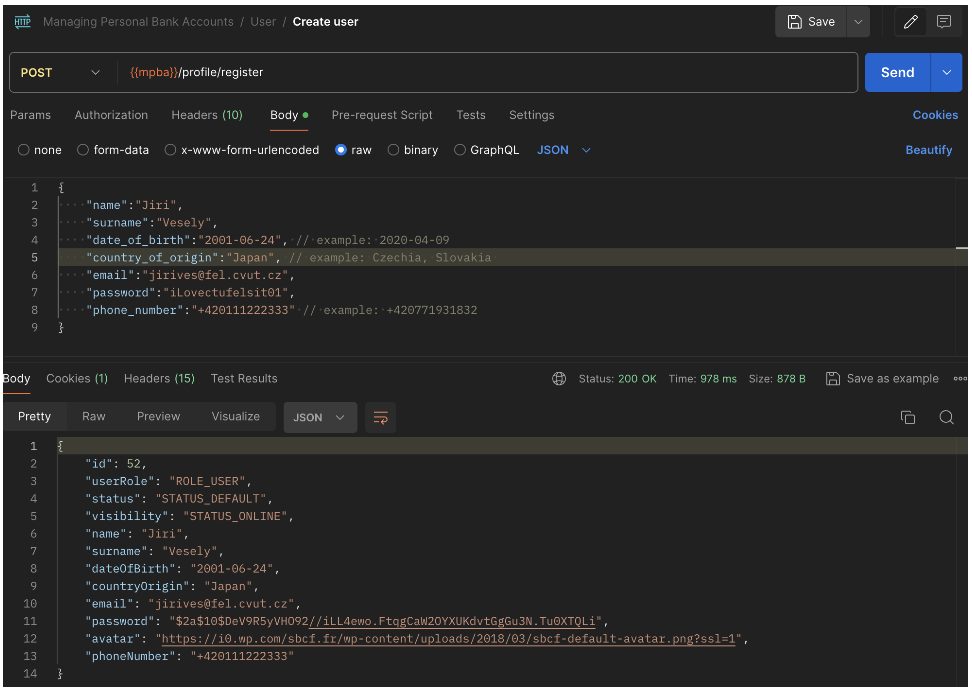
In this chapter, all the applications that are used to implement the project, their functionality and features have been broken down in detail. The next chapters will describe the application, its testing and the next steps in the project realization.

Chapter 10
Application description

This chapter describes the functionality of the application, its features that the application currently has. Below can be seen a few basic scenarios of the application.

10.1 Registration

To use the functionality of the app user need to register in it, when registering the user will need to set the data that is wanted from them and then the program will check if the data has been entered correctly. If so, the user will be created and a Czech currency bank card will be automatically linked to it. In case the user tries to enter incorrect data, such as invalid country, invalid phone number, name, email and other data, then the user will not be able to enter the correct data. If user wrote some wrong data, he will be able to edit them. Also, only the administrator can delete a user as well as change roles.



POST {{mpba}}/profile/register

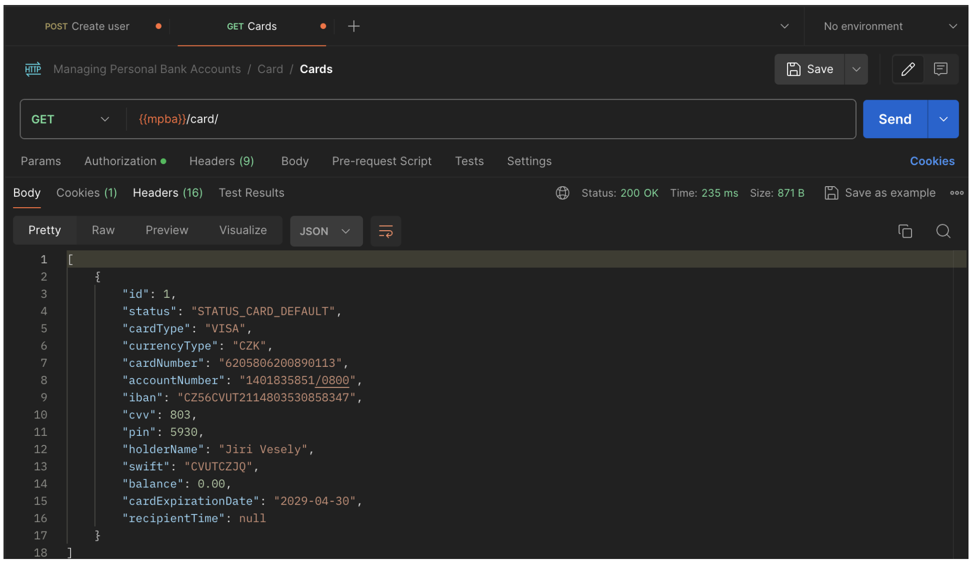
none | form-data | x-www-form-urlencoded | [✓ raw] | binary | GraphQL | JSON

```
{
    "name":"Jiri",
    "surname":"Vesely",
    "date_of_birth":"2001-06-24", // example: 2020-04-09
    "country_of_origin":"Japan", // example: Czechia, Slovakia
    "email":"jirives@fel.cvut.cz",
    "password":"iLovectufelsit01",
    "phone_number":"+420111222333" // example: +420771931832
}
```

Body | Cookies (1) | Headers (15) | Test Results — Status: 200 OK Time: 978 ms Size: 878 B

```
{
    "id": 52,
    "userRole": "ROLE_USER",
    "status": "STATUS_DEFAULT",
    "visibility": "STATUS_ONLINE",
    "name": "Jiri",
    "surname": "Vesely",
    "dateOfBirth": "2001-06-24",
    "countryOrigin": "Japan",
    "email": "jirives@fel.cvut.cz",
    "password": "$2a$10$DeV9R5yVHO92//iLL4ewo.FtqgCaW2OYXUKdvtGgGu3N.Tu0XTQLi",
    "avatar": "https://i0.wp.com/sbcf.fr/wp-content/uploads/2018/03/sbcf-default-avatar.png?ssl=1",
    "phoneNumber": "+420111222333"
}
```

Figure 10.1: Demonstration of user registration



GET {{mpba}}/card/

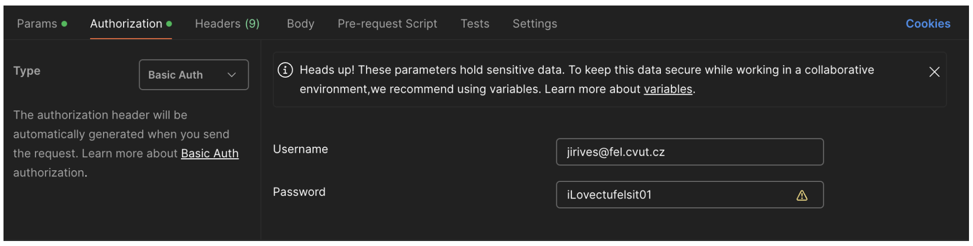
Body | Cookies (1) | Headers (16) | Test Results — Status: 200 OK Time: 235 ms Size: 871 B

```
[
    {
        "id": 1,
        "status": "STATUS_CARD_DEFAULT",
        "cardType": "VISA",
        "currencyType": "CZK",
        "cardNumber": "6205806200890113",
        "accountNumber": "1401835851/0800",
        "iban": "CZ56CVUT2114803530858347",
        "cvv": 803,
        "pin": 5930,
        "holderName": "Jiri Vesely",
        "swift": "CVUTCZJQ",
        "balance": 0.00,
        "cardExpirationDate": "2029-04-30",
        "recipientTime": null
    }
]
```

Figure 10.2: Demonstration of automatically added card after registration

10.2 Log in

For user authorization the standard Basic Auth is used, which has a login page. Authorization can be done using Postman. There is also a generated authorization page that can be used to simulate authorization. In order to try the authorization simulation user should enter the ip that was specified in the application (*standard is localhost*), then the symbol *":"*, then the port that was also specified in the application settings and the last is endpoint, which is */login*. *Example: localhost:8082/login.* If the user logs out of the application */logout*, then his status will change to *OFFLINE* and user will be redirect to */login* page. If user decides to log in later, then after authorization his status will change to *ONLINE* and he will be redirect to the page with Json info message. Below user can see what authorization looks like in Postman, as well as a Simulation of the login page with user status changes.



Type: Basic Auth

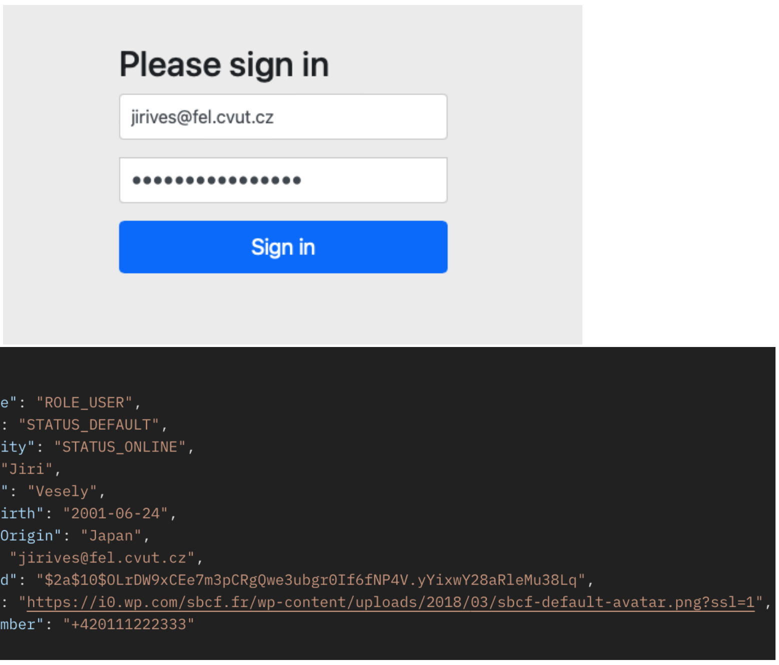
The authorization header will be automatically generated when you send the request. Learn more about Basic Auth authorization.

Heads up! These parameters hold sensitive data. To keep this data secure while working in a collaborative environment,we recommend using variables. Learn more about variables.

Username: jirives@fel.cvut.cz

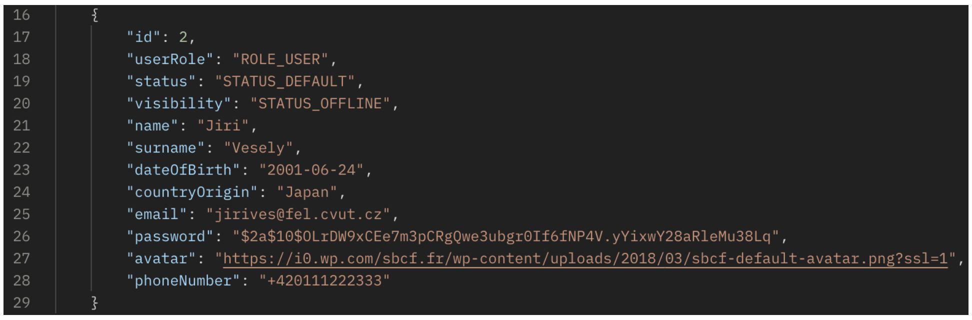
Password: iLovectufelsit01

Figure 10.3: Demonstration of authorization via Postman



```
{
    "id": 2,
    "userRole": "ROLE_USER",
    "status": "STATUS_DEFAULT",
    "visibility": "STATUS_ONLINE",
    "name": "Jiri",
    "surname": "Vesely",
    "dateOfBirth": "2001-06-24",
    "countryOrigin": "Japan",
    "email": "jirives@fel.cvut.cz",
    "password": "$2a$10$OLrDW9xCEe7m3pCRgQwe3ubgr0If6fNP4V.yYixwY28aRleMu38Lq",
    "avatar": "https://i0.wp.com/sbcf.fr/wp-content/uploads/2018/03/sbcf-default-avatar.png?ssl=1",
    "phoneNumber": "+420111222333"
}
```

Figure 10.4: Demonstration of simulate authorization Log In

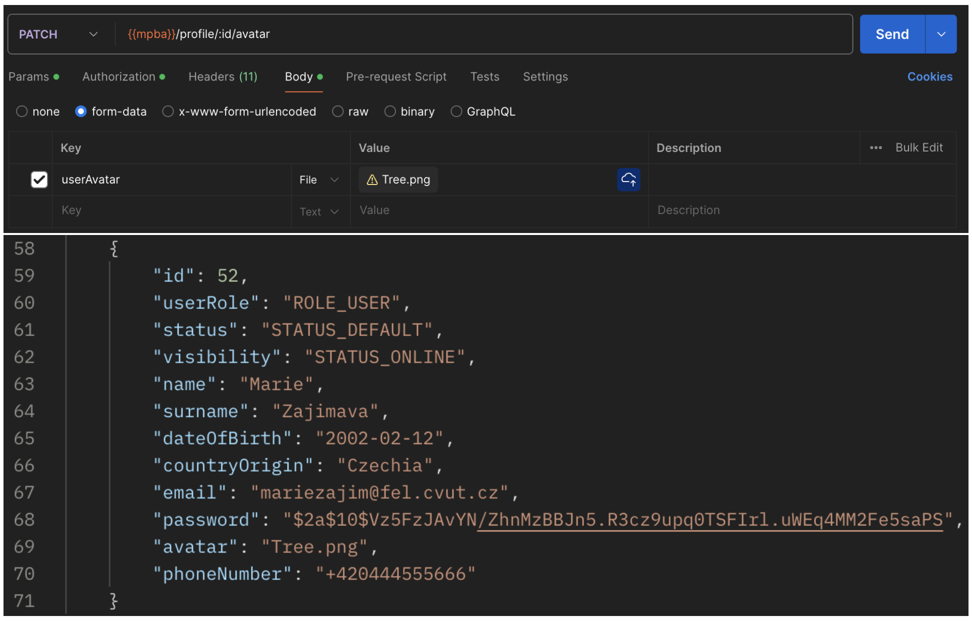


```
{
    "id": 2,
    "userRole": "ROLE_USER",
    "status": "STATUS_DEFAULT",
    "visibility": "STATUS_OFFLINE",
    "name": "Jiri",
    "surname": "Vesely",
    "dateOfBirth": "2001-06-24",
    "countryOrigin": "Japan",
    "email": "jirives@fel.cvut.cz",
    "password": "$2a$10$OLrDW9xCEe7m3pCRgQwe3ubgr0If6fNP4V.yYixwY28aRleMu38Lq",
    "avatar": "https://i0.wp.com/sbcf.fr/wp-content/uploads/2018/03/sbcf-default-avatar.png?ssl=1",
    "phoneNumber": "+420111222333"
}
```

Figure 10.5: Demonstration of simulate authorization Log Out

10.3 Adding User photo

The user can also customize their photo. To do this, upload a file of type *jpg*, *png*, *jpeg*. If the user tries to upload files that are not images, an error will appear.



```
PATCH {{mpba}}/profile/:id/avatar
```

Body: form-data

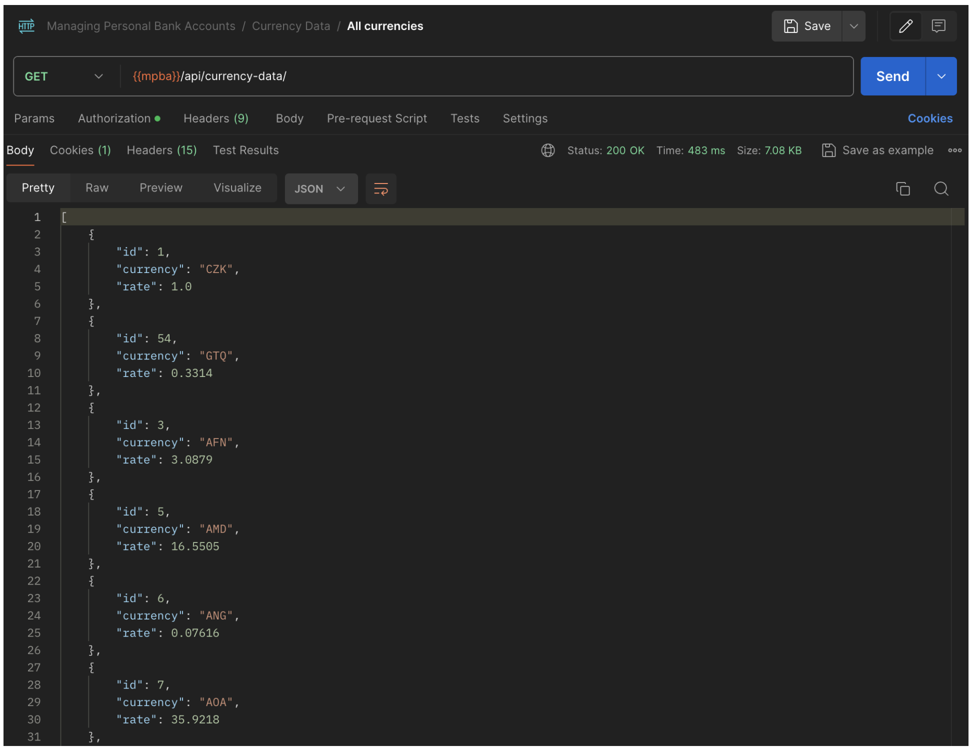
	Key		Value	Description
✓	userAvatar	File	Tree.png	
	Key	Text	Value	Description

```
{
    "id": 52,
    "userRole": "ROLE_USER",
    "status": "STATUS_DEFAULT",
    "visibility": "STATUS_ONLINE",
    "name": "Marie",
    "surname": "Zajimava",
    "dateOfBirth": "2002-02-12",
    "countryOrigin": "Czechia",
    "email": "mariezajim@fel.cvut.cz",
    "password": "$2a$10$Vz5FzJAvYN/ZhnMzBBJn5.R3cz9upq0TSFIrl.uWEq4MM2Fe5saPS",
    "avatar": "Tree.png",
    "phoneNumber": "+420444555666"
}
```

Figure 10.6: Demonstration of uploading user avatar

10.4 Currency data

Any user can view the exchange rate available at the bank. Thanks to the ExchangeRate-API, the bank has 162 exchange rates available.

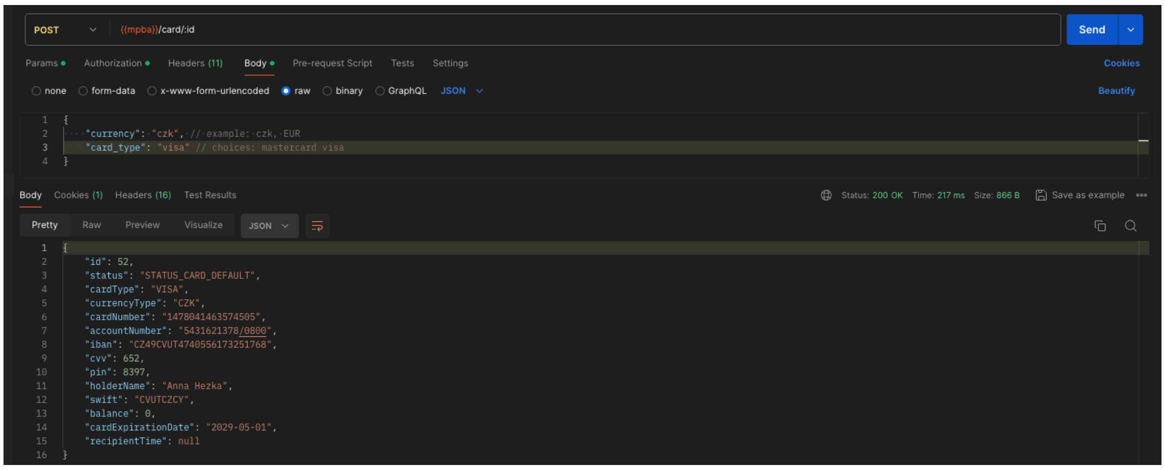


```
[
    {
        "id": 1,
        "currency": "CZK",
        "rate": 1.0
    },
    {
        "id": 54,
        "currency": "GTQ",
        "rate": 0.3314
    },
    {
        "id": 3,
        "currency": "AFN",
        "rate": 3.0879
    },
    {
        "id": 5,
        "currency": "AMD",
        "rate": 16.5505
    },
    {
        "id": 6,
        "currency": "ANG",
        "rate": 0.07616
    },
    {
        "id": 7,
        "currency": "AOA",
        "rate": 35.9218
    },
```

Figure 10.7: Demonstration of getting available currencies

10.5 Card creation and management

The user can open a card by specifying its currency from the available 5 (Czech crowns, Dollars, Euros, Zloty, Hryvnias) and its type (MasterCard or Visa). The card can also be refilled. The moderator can change its type, as well as block or unblock it. A card can only be deleted by a user with the administrator role and provided there is no loan, no deposit and the card itself has no funds on it. Below can be seen a demonstration of creating a card.



```
{{mpba}}/card/:id
```

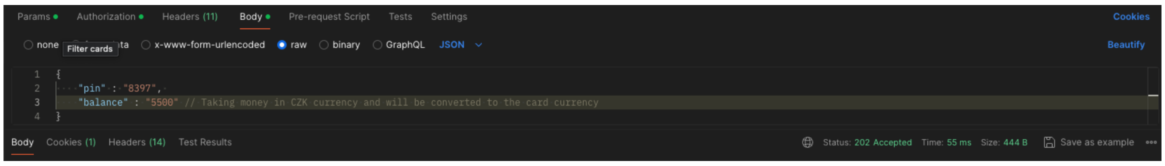
```
{
    "currency": "czk", // example: czk, EUR
    "card_type": "visa" // choices: mastercard visa
}
```

Status: 200 OK Time: 217 ms Size: 866 B

```
{
    "id": 52,
    "status": "STATUS_CARD_DEFAULT",
    "cardType": "VISA",
    "currencyType": "CZK",
    "cardNumber": "1478041463574505",
    "accountNumber": "5431621378/0800",
    "iban": "CZ49CVUT4740556173251768",
    "cvv": 652,
    "pin": 8397,
    "holderName": "Anna Hezka",
    "swift": "CVUTCZCY",
    "balance": 0,
    "cardExpirationDate": "2029-05-01",
    "recipientTime": null
}
```

Figure 10.8: Demonstration of creating card

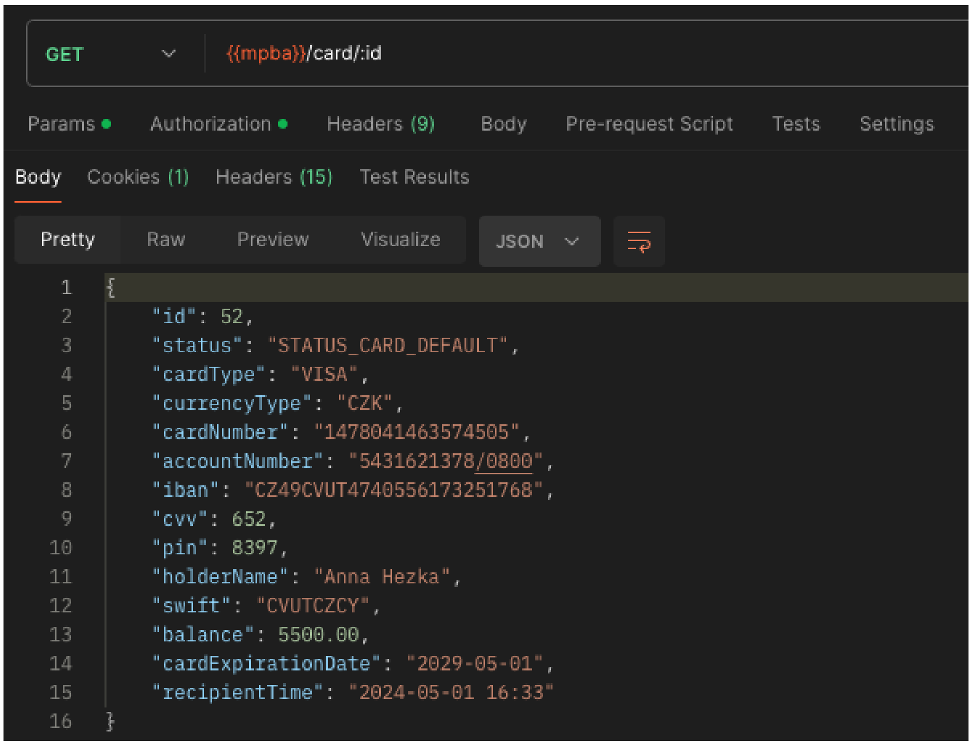
We can also refill the card by entering its pin code and the amount we want to deposit. When the user refill the account, the amount is automatically converted into Czech crowns. The following shows how to refill the card.



```
{
    "pin" : "8397",
    "balance" : "5500" // Taking money in CZK currency and will be converted to the card currency
}
```

Status: 202 Accepted Time: 55 ms Size: 444 B

Figure 10.9: Demonstration of refilling the card

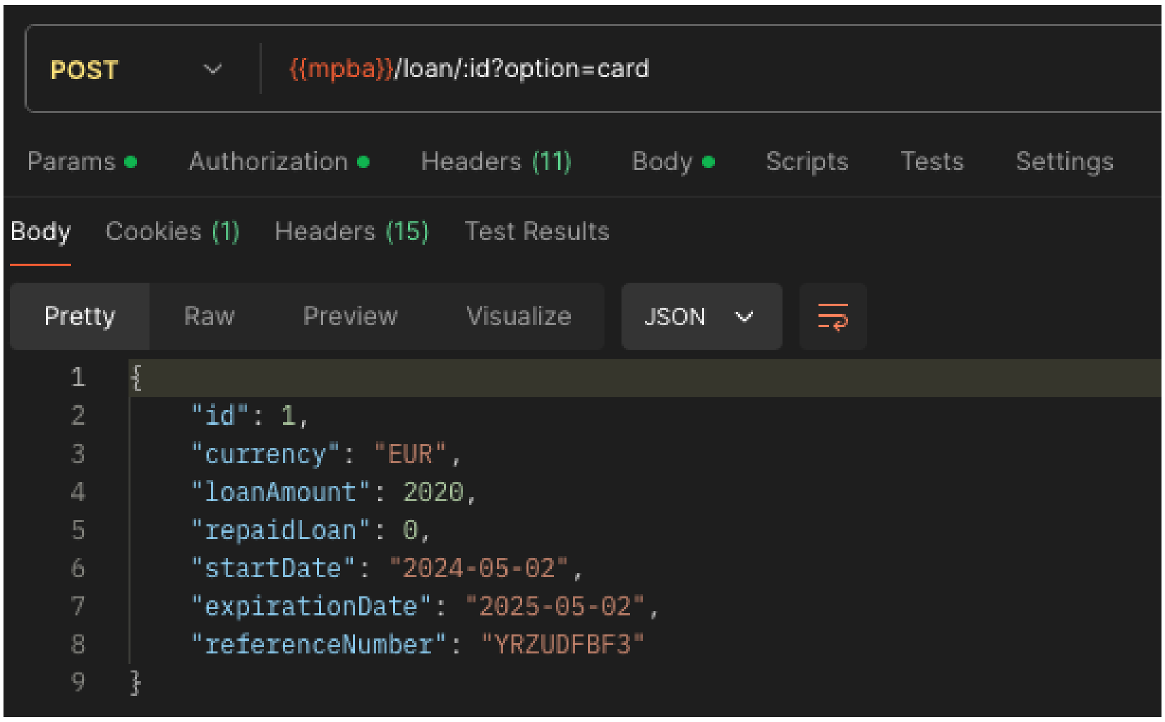


```
{
    "id": 52,
    "status": "STATUS_CARD_DEFAULT",
    "cardType": "VISA",
    "currencyType": "CZK",
    "cardNumber": "1478041463574505",
    "accountNumber": "5431621378/0800",
    "iban": "CZ49CVUT4740556173251768",
    "cvv": 652,
    "pin": 8397,
    "holderName": "Anna Hezka",
    "swift": "CVUTCZCY",
    "balance": 5500.00,
    "cardExpirationDate": "2029-05-01",
    "recipientTime": "2024-05-01 16:33"
}
```

Figure 10.10: Demonstration of the result of refilling the card

10.6 Opening loan and Repayment

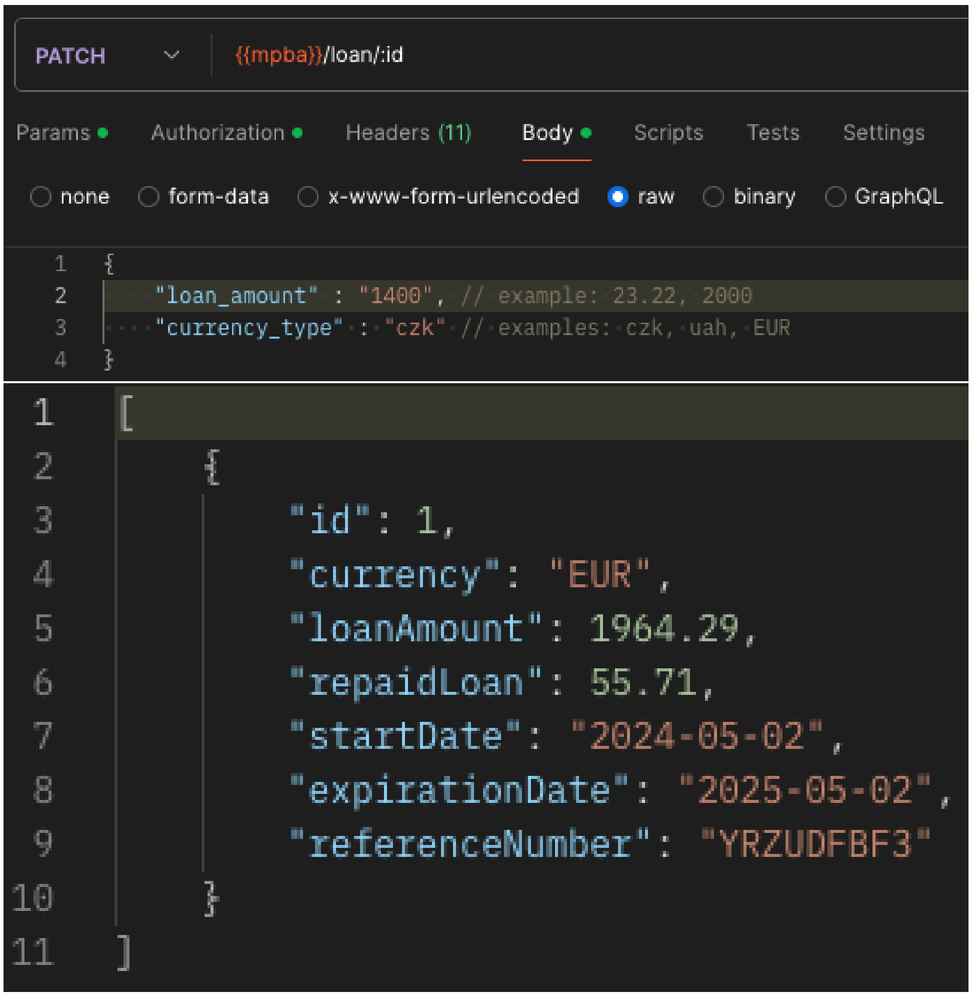
The user can also take money on loan. To do this, he/she will need to specify the currency from the available bank currencies and the amount. In addition, the user must choose where the loan will be recorded - on the user's card or in his name. After that, he will be given a deadline for which he must pay it off. The user cannot take more than one loan. Also the end date of payment can be increased, but this can only be done by the moderator. Below is a demonstration of how the loan is opened.



```
{
    "id": 1,
    "currency": "EUR",
    "loanAmount": 2020,
    "repaidLoan": 0,
    "startDate": "2024-05-02",
    "expirationDate": "2025-05-02",
    "referenceNumber": "YRZUDFBF3"
}
```

Figure 10.11: Demonstration of opening loan

The user can pay for the loan in any currency. Upon payment, the amount is converted to the currency of the loan. Once the loan is paid off, it will automatically be removed from the user's view. Below is a demonstration of a loan.



PATCH {{mpba}}/loan/:id

Params ● Authorization ● Headers (11) Body ● Scripts Tests Settings

none form-data x-www-form-urlencoded [✓ raw] binary GraphQL

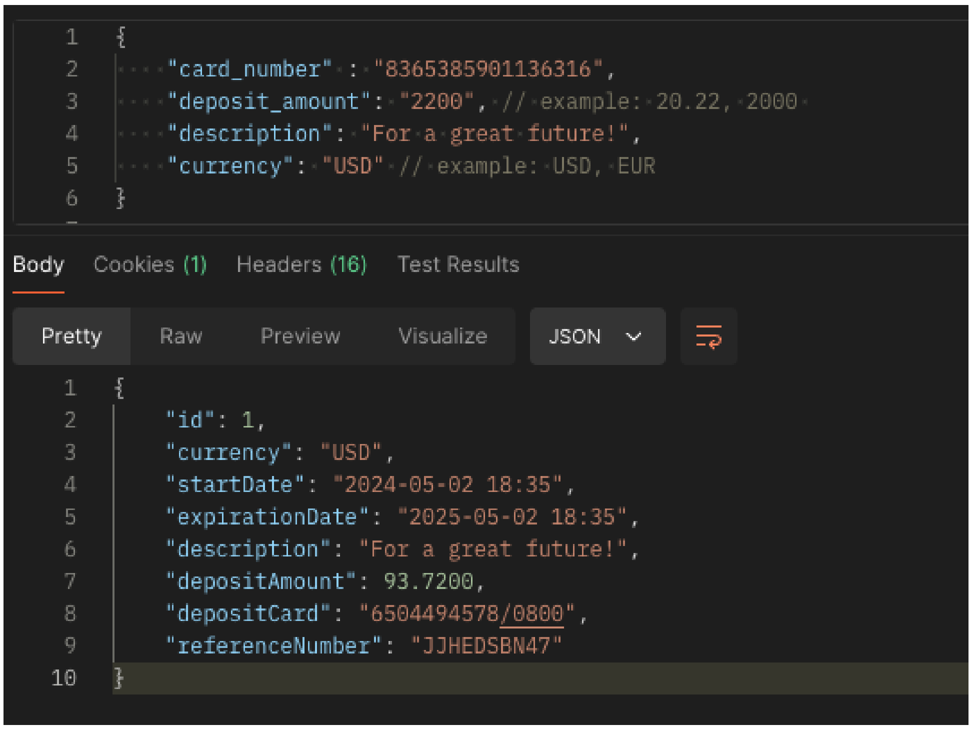
```
{
    "loan_amount" : "1400", // example: 23.22, 2000
    "currency_type" : "czk" // examples: czk, uah, EUR
}
```

```
[
    {
        "id": 1,
        "currency": "EUR",
        "loanAmount": 1964.29,
        "repaidLoan": 55.71,
        "startDate": "2024-05-02",
        "expirationDate": "2025-05-02",
        "referenceNumber": "YRZUDFBF3"
    }
]
```

Figure 10.12: Demonstration of repaying loan

10.7 Opening a deposit

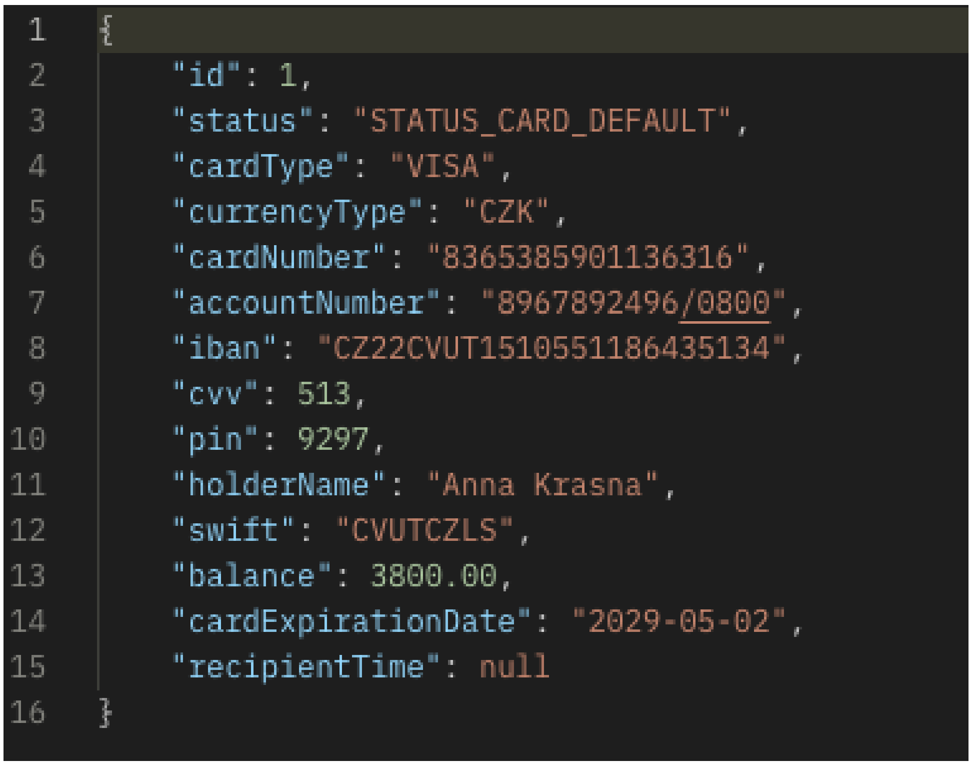
The user can also open a deposit account. When depositing money to the deposit account, the user must transfer money from the specified card, after which it will be debited and transferred to the deposit account. After the deposit time expires, the user will be able to return the money to the specified card with a 5 percent bonus. The user can also return the amount before the expiration of the deposit period, but the bonus will not be accrued. The deposit can be opened in 5 available currencies. When transferring money of one currency to a deposit in another currency, the money is converted. Below is an example of deposit replenishment and the condition that if the money is requested back immediately, then without the 5 percent bonus.



```
{
    "card_number" : "8365385901136316",
    "deposit_amount": "2200", // example: 20.22, 2000
    "description": "For a great future!",
    "currency": "USD" // example: USD, EUR
}
```

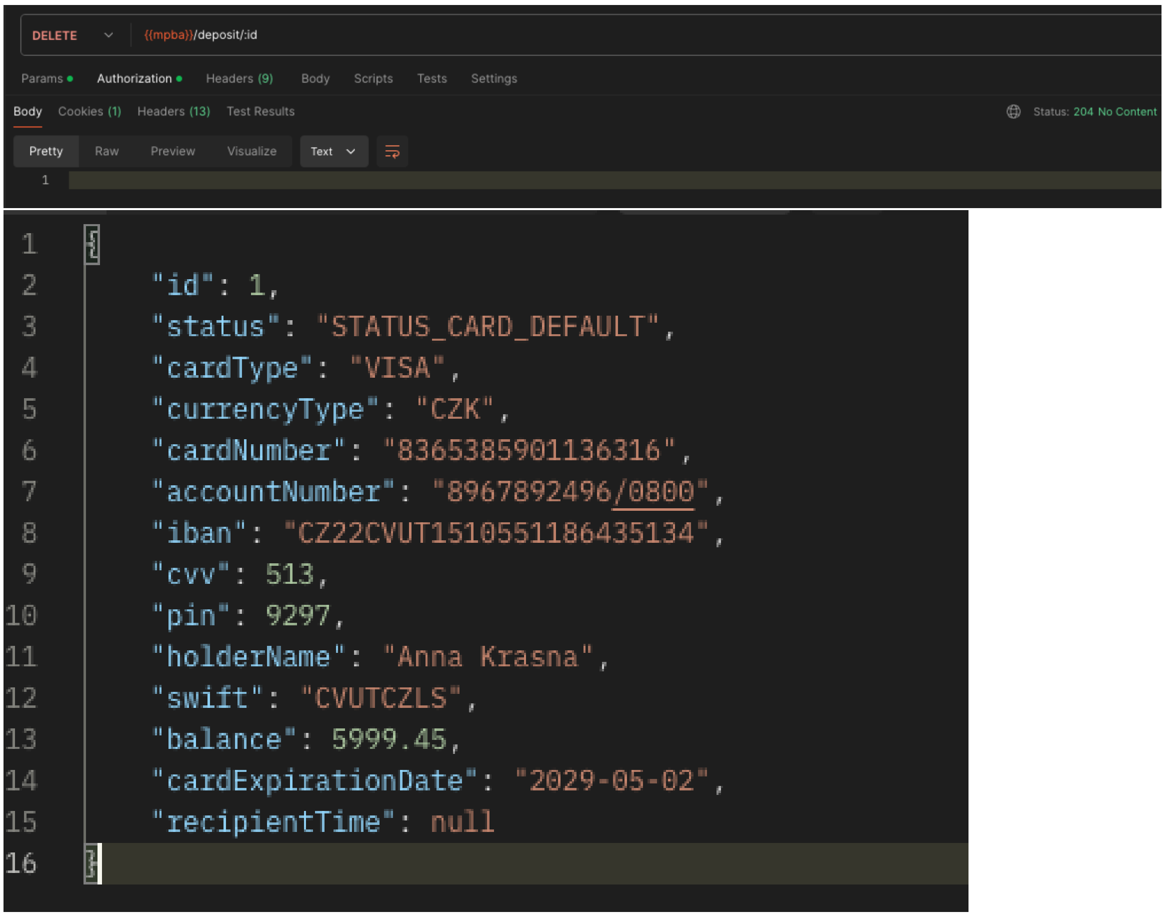
```
{
    "id": 1,
    "currency": "USD",
    "startDate": "2024-05-02 18:35",
    "expirationDate": "2025-05-02 18:35",
    "description": "For a great future!",
    "depositAmount": 93.7200,
    "depositCard": "6504494578/0800",
    "referenceNumber": "JJHEDSBN47"
}
```

Figure 10.13: Demonstration of the openning deposit



```
{
    "id": 1,
    "status": "STATUS_CARD_DEFAULT",
    "cardType": "VISA",
    "currencyType": "CZK",
    "cardNumber": "8365385901136316",
    "accountNumber": "8967892496/0800",
    "iban": "CZ22CVUT1510551186435134",
    "cvv": 513,
    "pin": 9297,
    "holderName": "Anna Krasna",
    "swift": "CVUTCZLS",
    "balance": 3800.00,
    "cardExpirationDate": "2029-05-02",
    "recipientTime": null
}
```

Figure 10.14: Demonstration of the user's balance after opening a deposit



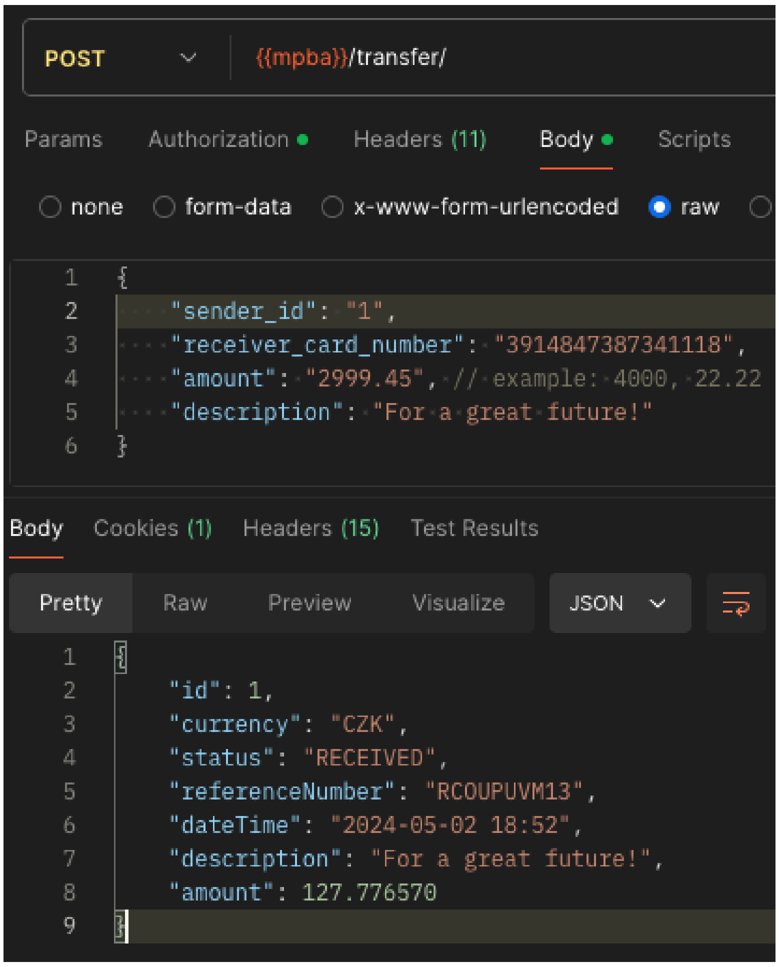
```
DELETE {{mpba}}/deposit/:id
Status: 204 No Content
```

```
{
    "id": 1,
    "status": "STATUS_CARD_DEFAULT",
    "cardType": "VISA",
    "currencyType": "CZK",
    "cardNumber": "8365385901136316",
    "accountNumber": "8967892496/0800",
    "iban": "CZ22CVUT1510551186435134",
    "cvv": 513,
    "pin": 9297,
    "holderName": "Anna Krasna",
    "swift": "CVUTCZLS",
    "balance": 5999.45,
    "cardExpirationDate": "2029-05-02",
    "recipientTime": null
}
```

Figure 10.15: Demonstration of the deleting deposit and user's balance

10.8 Transfer to user account

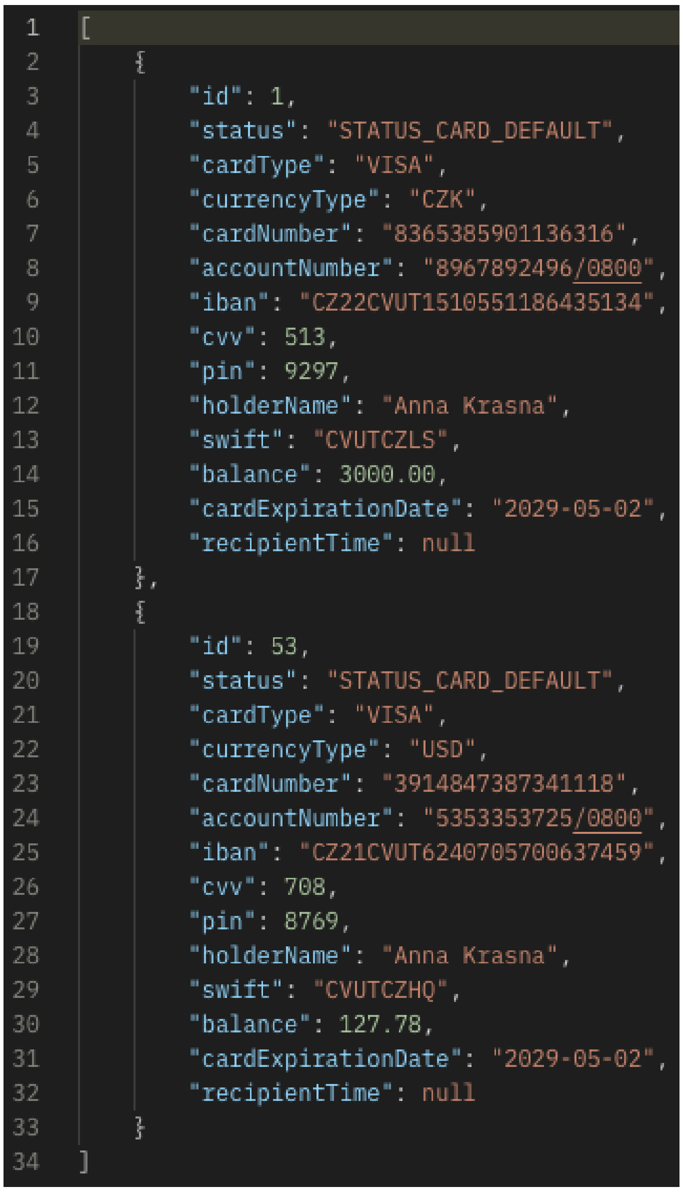
Users can send money to each other's cards. To do this, you need to specify the sender's ID, the recipient's card number, the amount and a description. Below is an example where a user with a Czech card sends money to his dollar card. The specified amount will be converted to the currency of the card to which the amount of money is sent. This operation cannot be done between users. Only transfer with the same currency is available for users.



```
{
    "sender_id": "1",
    "receiver_card_number": "3914847387341118",
    "amount": "2999.45", // example: 4000, 22.22
    "description": "For a great future!"
}
```

```
{
    "id": 1,
    "currency": "CZK",
    "status": "RECEIVED",
    "referenceNumber": "RCOUPUVM13",
    "dateTime": "2024-05-02 18:52",
    "description": "For a great future!",
    "amount": 127.776570
}
```

Figure 10.16: Demonstration of transferring money to another card

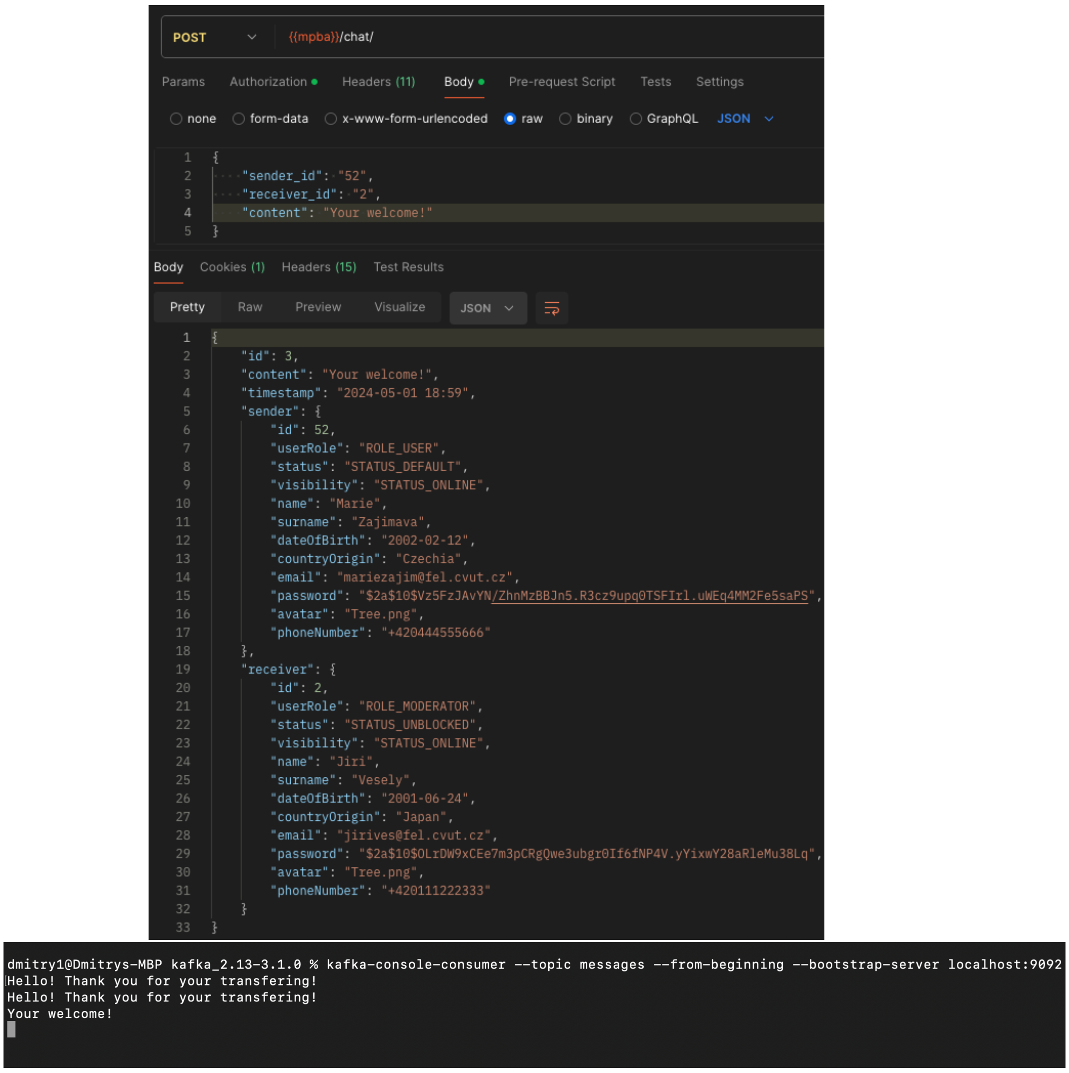


```
[
    {
        "id": 1,
        "status": "STATUS_CARD_DEFAULT",
        "cardType": "VISA",
        "currencyType": "CZK",
        "cardNumber": "8365385901136316",
        "accountNumber": "8967892496/0800",
        "iban": "CZ22CVUT1510551186435134",
        "cvv": 513,
        "pin": 9297,
        "holderName": "Anna Krasna",
        "swift": "CVUTCZLS",
        "balance": 3000.00,
        "cardExpirationDate": "2029-05-02",
        "recipientTime": null
    },
    {
        "id": 53,
        "status": "STATUS_CARD_DEFAULT",
        "cardType": "VISA",
        "currencyType": "USD",
        "cardNumber": "3914847387341118",
        "accountNumber": "5353353725/0800",
        "iban": "CZ21CVUT6240705700637459",
        "cvv": 708,
        "pin": 8769,
        "holderName": "Anna Krasna",
        "swift": "CVUTCZHQ",
        "balance": 127.78,
        "cardExpirationDate": "2029-05-02",
        "recipientTime": null
    }
]
```

Figure 10.17: Demonstration of user cards after transfer

10.9 Sending messages

Users can also send messages to each other, thus communicating to solve various issues. The following is a demonstration of how users send messages to each other and can be seen using Kafka Message Broker.



```
{
    "sender_id": "52",
    "receiver_id": "2",
    "content": "Your welcome!"
}
```

```
{
    "id": 3,
    "content": "Your welcome!",
    "timestamp": "2024-05-01 18:59",
    "sender": {
        "id": 52,
        "userRole": "ROLE_USER",
        "status": "STATUS_DEFAULT",
        "visibility": "STATUS_ONLINE",
        "name": "Marie",
        "surname": "Zajimava",
        "dateOfBirth": "2002-02-12",
        "countryOrigin": "Czechia",
        "email": "mariezajim@fel.cvut.cz",
        "password": "$2a$10$Vz5FzJAvYN/ZhnMzBBJn5.R3cz9upq0TSFIrl.uWEq4MM2Fe5saPS",
        "avatar": "Tree.png",
        "phoneNumber": "+420444555666"
    },
    "receiver": {
        "id": 2,
        "userRole": "ROLE_MODERATOR",
        "status": "STATUS_UNBLOCKED",
        "visibility": "STATUS_ONLINE",
        "name": "Jiri",
        "surname": "Vesely",
        "dateOfBirth": "2001-06-24",
        "countryOrigin": "Japan",
        "email": "jirives@fel.cvut.cz",
        "password": "$2a$10$OLrDW9xCEe7m3pCRgQwe3ubgr0If6fNP4V.yYixwY28aRleMu38Lq",
        "avatar": "Tree.png",
        "phoneNumber": "+420111222333"
    }
}
```

```
dmitry1@Dmitrys-MBP kafka_2.13-3.1.0 % kafka-console-consumer --topic messages --from-beginning --bootstrap-server localhost:9092
Hello! Thank you for your transfering!
Hello! Thank you for your transfering!
Your welcome!
```

Figure 10.18: Demonstration of messaging between users

10.10 Conclusion

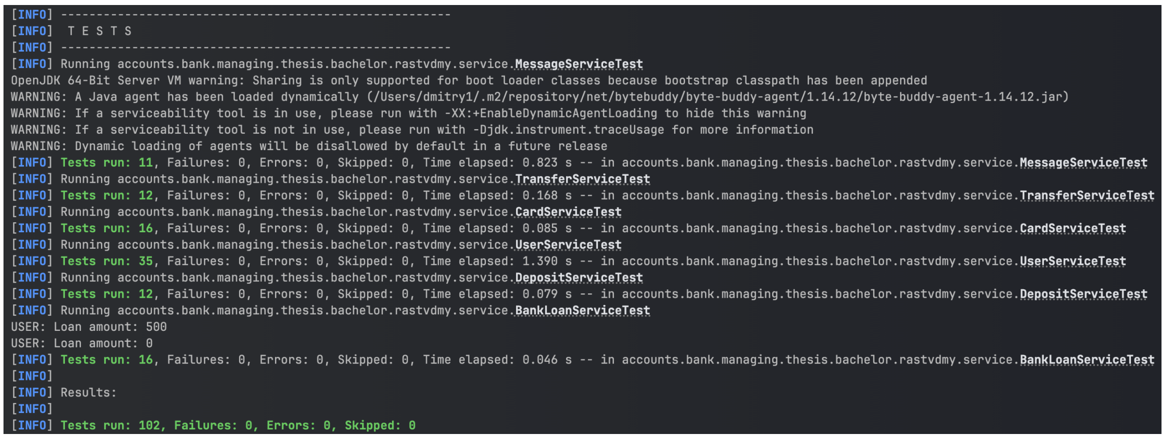
This chapter provides an overview of the application's functionality, covering registration, login, photo customization, currency data retrieval, card management, loan handling, deposit management, money transfers, and messaging. It demonstrates the application's versatility and usability across different scenarios.

Chapter 11
Testing

In this chapter, the processes of testing an application are studied in detail. Testing is one of the major factors in application development. Tests can be used to check various scenarios in an application and if any errors are found, they can be fixed. The purpose of testing is to verify that the application performs the required functions and that the application itself works without any errors. In the chapter, the types of testing are analyzed and a comparison is made between one of the banks and the application.

11.1 Server-side testing

The server side of an application plays a key role in its functionality, so thorough testing is necessary to ensure that it meets user expectations. For this purpose, tools such as JUnit and Mockito are often used, which allow testing the operability and reliability of the code, as well as emulating various use scenarios. Below is a demonstration of the results of application tests, the task of which is to test the service part of the application.

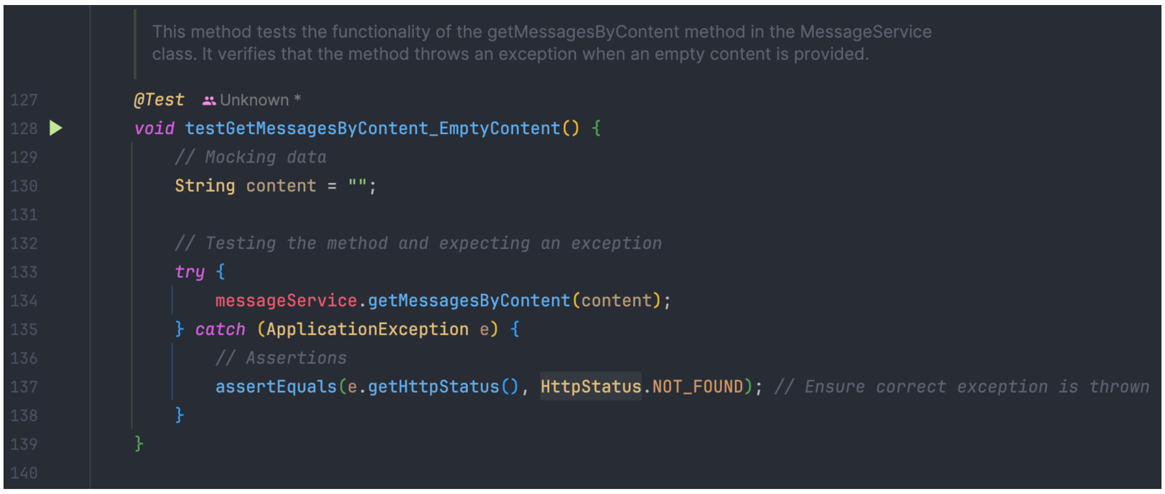


```
[INFO] -------------------------------------------------------
[INFO]  T E S T S
[INFO] -------------------------------------------------------
[INFO] Running accounts.bank.managing.thesis.bachelor.rastvdmy.service.MessageServiceTest
OpenJDK 64-Bit Server VM warning: Sharing is only supported for boot loader classes because bootstrap classpath has been appended
WARNING: A Java agent has been loaded dynamically (/Users/dmitry1/.m2/repository/net/bytebuddy/byte-buddy-agent/1.14.12/byte-buddy-agent-1.14.12.jar)
WARNING: If a serviceability tool is in use, please run with -XX:+EnableDynamicAgentLoading to hide this warning
WARNING: If a serviceability tool is not in use, please run with -Djdk.instrument.traceUsage for more information
WARNING: Dynamic loading of agents will be disallowed by default in a future release
[INFO] Tests run: 11, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 0.823 s -- in accounts.bank.managing.thesis.bachelor.rastvdmy.service.MessageServiceTest
[INFO] Running accounts.bank.managing.thesis.bachelor.rastvdmy.service.TransferServiceTest
[INFO] Tests run: 12, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 0.168 s -- in accounts.bank.managing.thesis.bachelor.rastvdmy.service.TransferServiceTest
[INFO] Running accounts.bank.managing.thesis.bachelor.rastvdmy.service.CardServiceTest
[INFO] Tests run: 16, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 0.085 s -- in accounts.bank.managing.thesis.bachelor.rastvdmy.service.CardServiceTest
[INFO] Running accounts.bank.managing.thesis.bachelor.rastvdmy.service.UserServiceTest
[INFO] Tests run: 35, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 1.390 s -- in accounts.bank.managing.thesis.bachelor.rastvdmy.service.UserServiceTest
[INFO] Running accounts.bank.managing.thesis.bachelor.rastvdmy.service.DepositServiceTest
[INFO] Tests run: 12, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 0.079 s -- in accounts.bank.managing.thesis.bachelor.rastvdmy.service.DepositServiceTest
[INFO] Running accounts.bank.managing.thesis.bachelor.rastvdmy.service.BankLoanServiceTest
USER: Loan amount: 500
USER: Loan amount: 0
[INFO] Tests run: 16, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 0.046 s -- in accounts.bank.managing.thesis.bachelor.rastvdmy.service.BankLoanServiceTest
[INFO]
[INFO] Results:
[INFO]
[INFO] Tests run: 102, Failures: 0, Errors: 0, Skipped: 0
```

Figure 11.1: Demonstration of testing result during running Maven

11.2 JUnit

JUnit is an open source unit testing framework for the Java programming language that plays an important role in development and is used for testing applications. JUnit offers the user a list of annotations and functions that will allow testing different parts of the code, comparing results, error dropdowns and more. Below is a demonstration of the JUnit testing method from the project. [47]



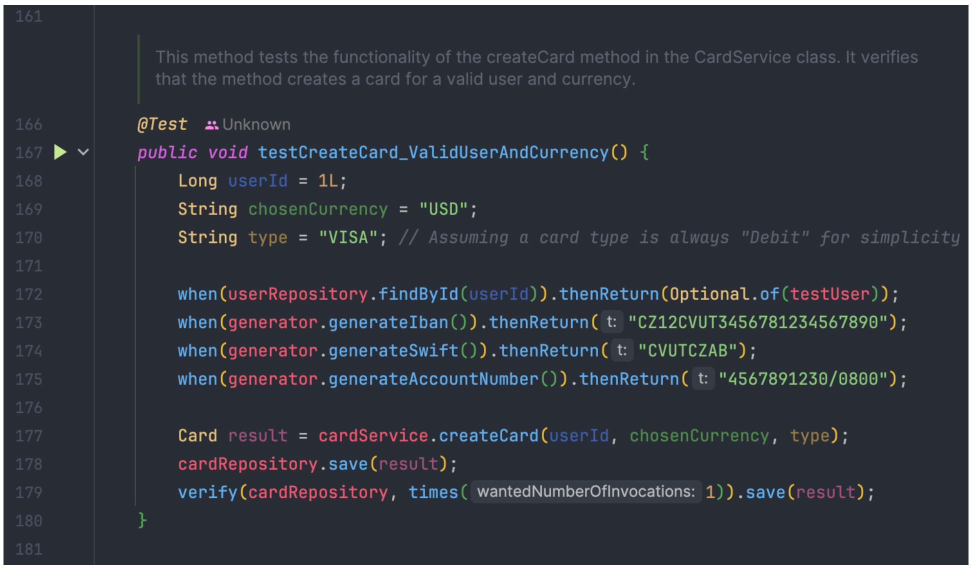
This method tests the functionality of the getMessagesByContent method in the MessageService class. It verifies that the method throws an exception when an empty content is provided.

```
@Test  Unknown *
void testGetMessagesByContent_EmptyContent() {
    // Mocking data
    String content = "";

    // Testing the method and expecting an exception
    try {
        messageService.getMessagesByContent(content);
    } catch (ApplicationException e) {
        // Assertions
        assertEquals(e.getHttpStatus(), HttpStatus.NOT_FOUND); // Ensure correct exception is thrown
    }
}
```

Figure 11.2: Demonstration of JUnit testing

11.3 Mockito

Mockito testing is a framework that is used to write tests efficiently while using special mock objects. Thus Mockito allows user not to implement production class objects, instead it creates a "mock" which is not an existing object but just a shell. This makes it possible to check, for example, if a class has been run, how many times it has been run, and much more. Below is a demonstration of the Mockito testing method from the project. [48]



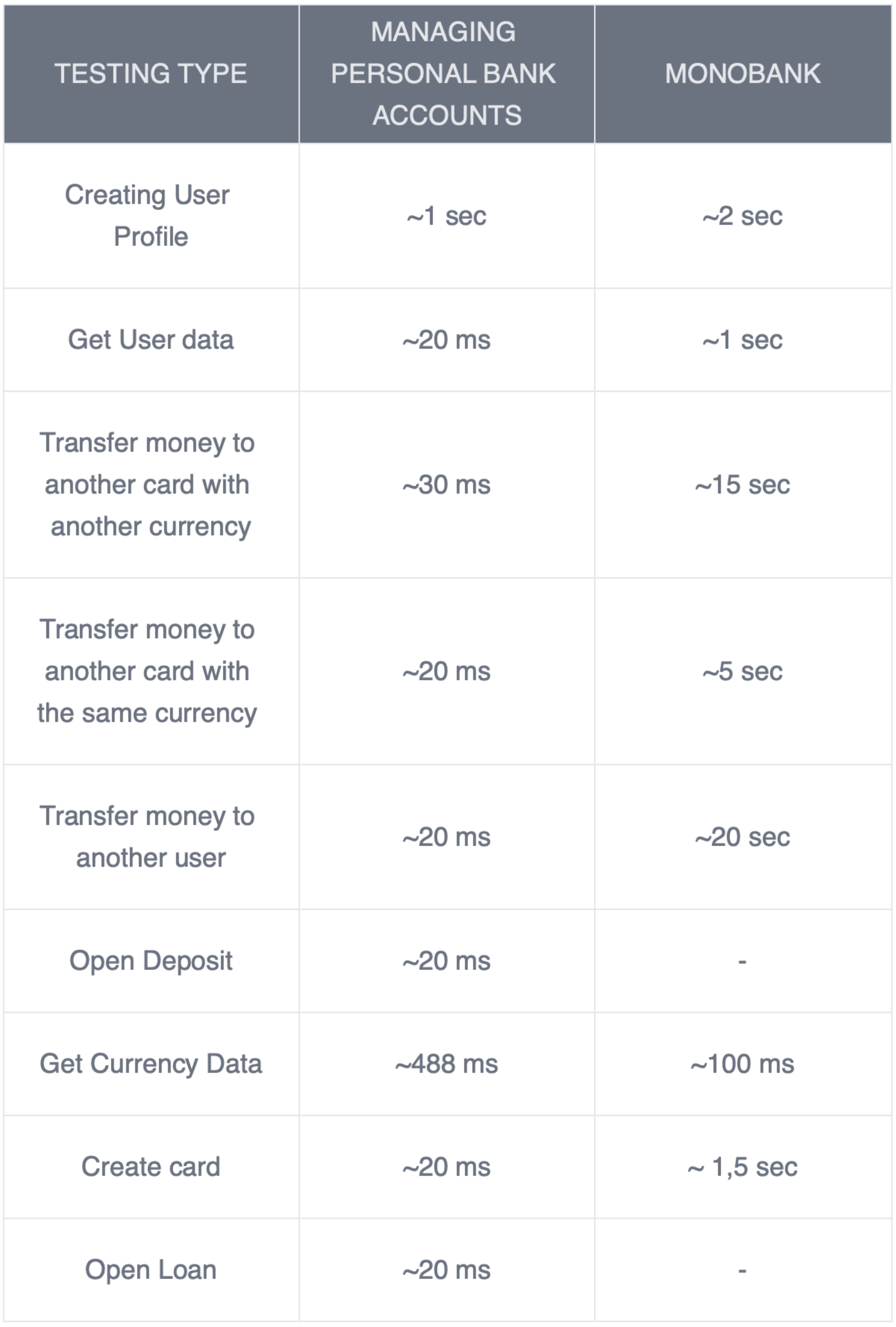
This method tests the functionality of the createCard method in the CardService class. It verifies that the method creates a card for a valid user and currency.

```java
@Test  Unknown
public void testCreateCard_ValidUserAndCurrency() {
    Long userId = 1L;
    String chosenCurrency = "USD";
    String type = "VISA"; // Assuming a card type is always "Debit" for simplicity

    when(userRepository.findById(userId)).thenReturn(Optional.of(testUser));
    when(generator.generateIban()).thenReturn( t: "CZ12CVUT3456781234567890");
    when(generator.generateSwift()).thenReturn( t: "CVUTCZAB");
    when(generator.generateAccountNumber()).thenReturn( t: "4567891230/0800");

    Card result = cardService.createCard(userId, chosenCurrency, type);
    cardRepository.save(result);
    verify(cardRepository, times( wantedNumberOfInvocations: 1)).save(result);
}
```

Figure 11.3: Demonstration of Mockito testing

11.4 Bank vs Application

In this chapter, a table displaying the results of comparisons between the different operations performed during the testing of the "Managing Personal Bank Accounts" and "Monobank" applications can be found below.



TESTING TYPE	MANAGING PERSONAL BANK ACCOUNTS	MONOBANK
Creating User Profile	~1 sec	~2 sec
Get User data	~20 ms	~1 sec
Transfer money to another card with another currency	~30 ms	~15 sec
Transfer money to another card with the same currency	~20 ms	~5 sec
Transfer money to another user	~20 ms	~20 sec
Open Deposit	~20 ms	-
Get Currency Data	~488 ms	~100 ms
Create card	~20 ms	~ 1,5 sec
Open Loan	~20 ms	-

Figure 11.4: Table with the all testing speed results

11.5 Conclusion

This chapter sampled the types of testing used in the application, analyzed the comparison of application and bank testing. The following chapters describe the next steps to improve the application and the main result of the work performed.

Chapter 12
Next steps

As a result of the work performed, it can be said that the application fulfills the above described functional and non-functional requirements that were set for the implementation of the application. It should also be noted that the application has opportunities for improvement and is open for modifications. The following aspects can be realized in this application:

1. Transferring the application to microservice architecture.
2. Replacing Basic Auth with OAuth 2.0.
3. Optimizing the application for a large user flow.
4. Frontend implementation using React or Angular.
5. Using Kubernetes for container management.
6. Using Cloud service.
7. Adding more choice of currencies.

Chapter 13
Conclusion

In this bachelor's project, the topic of the project was studied in detail, comparisons were made with other existing analogs of applications, and programs were selected with which the application is implemented. Then the application was implemented using Java and PostgreSQL languages, Spring Boot framework, caching using Hazelcast, image creation using Docker and messenger was created using Apache Kafka. In addition, tests of the application were written using JUnit and Mockito frameworks to test the application and test different situations.

This analysis of the assignment gave a better understanding of the implementation of the application.

The result of the bachelor's thesis is the design and implementation of managing personal bank accounts.

There is also a video demonstration of the application, where it is possible to get familiar with the main features of the project. To view the video, click on the quote number or find the reference in the bibliography under the quote number. [49]

Chapter 14

List of used abbreviations

IT - Information Technology

API - Application Persistence Interface

UML - Unified Modeling Language

SQL - Structured Query Language

REST - Representational State Transfer

DB - Database

UI - User Interface

JPA - Java Persistence API

OS - Operating System

JSON - JavaScript Object Notation

JVM - Java Virtual Machine

GUI - Graphical User Interface

OOP - Object-Oriented Programming

HTTP - Hypertext Transfer Protocol

TCP - Transmission Control Protocol

Auth - Authentication

IDE	- Integrated Development Environment
ORM	- Object–Relational Mapping
URL	- Uniform Resource Locator
DTO	- Data Transfer Object
IBAN	- International Bank Account Number
ID	- Identification
MPBA	- Managing Personal Bank Accounts

Chapter 15

Bibliography

[1] Roger S. Pressmann Bruce Maxim: *"Software Engineering: A Practitioner's Approach"*, ISBN-10: 9780078022128

[2] doc. Ing. Miroslav Bureš, Ph.D. ČVUT. *"Software Testing"* [online]. 2021 [quoted. 2023-10-20]. Available from: https://moodle.fel.cvut.cz/course/view.php?id=6692

[3] Ing. Martin Řimnáč, Ph.D. ČVUT. *"Database systems"* [online]. 2021 [quoted 2023-10-25]. Available from: https://cw.fel.cvut.cz/b212/courses/b0b36dbs/start

[4] Ing. Petr Křemen, Ph.D. ČVUT. *"Enterprise architecture"* [online]. 2022 [quoted 2023-11-01]. Available from: https://cw.fel.cvut.cz/b221/courses/b6b36ear/lectures/start

[5] Ing David Kadleček, Ph.D. ČVUT. *"Object modeling"* [online]. 2021 [quoted 2023-11-03]. Available from: https://cw.fel.cvut.cz/b211/courses/b6b36omo/prednasky

[6] Ing. Jiří Šebek. ČVUT. *"Design of software systems"* [online]. 2022 [quoted 2023-11-18]. Available from: https://cw.fel.cvut.cz/b222/courses/b6b36nss/start

[7] Baeldung. *"Build Your REST API with Spring"* [online]. 2020 [quoted 2023-11-19]. Available from: https://www.baeldung.com/rest-api-spring-guide

[8] Regina O. Obe, Leo S. Hsu. *"PostgreSQL: Up and Running"* [online]. 2017 [quoted 2024-03-20]. Available from: https://www.oreilly.com/library/view/postgresql-up-and/9781491963401/

[9] Erich Gamma, John Vlissides, Richard Helm, Ralph Johnson. *"Design patterns: elements of reusable object-oriented software"* [online]. 1994 [quoted 2024-04-25]. Available from: https://www.oreilly.com/library/view/design-patterns-elements/0201633612/

[10] Erste Group. *"Česká Spořitelna"* [online]. [quoted 2023-10-16]. Available from: https://george.csas.cz/

[11] Dragon Capital. *"Digital Pumb"* [online]. [quoted 2023-10-16]. Available from: https://www.digital.pumb.ua/

[12] PrivatBank. *"Privat24"* [online]. [quoted 2023-10-16]. Available from: https://next.privat24.ua/

[13] Raiffeisen Bank International. *"Raiffeisenbank"* [online]. [quoted 2023-10-16]. Available from: https://www.rb.cz/en

[14] Joint-Stock. *"Industrial bank"* [online]. [quoted 2023-06-10]. Available from: https://industrialbank.ua/en/business

[15] IBM. *"Use Case Diagram"* [online]. [quoted 2023-11-19]. Available from: https://www.ibm.com/docs/en/rational-soft-arch/9.6.1?topic=diagrams-use-case

[16] GeekforGeeks. *"Functional vs Non Functional Requirements"* [online]. [quoted 2024-02-24]. Available from: https://www.geeksforgeeks.org/functional-vs-non-functional-requirements/

[17] GeekforGeeks. *"What is Database?"* [online]. [quoted 2023-11-21]. Available from: https://www.geeksforgeeks.org/what-is-database/

[18] PostgreSQL. *"What is PostgreSQL?"* [online]. [quoted 2024-02-09]. Available from: https://www.postgresql.org/about/

[19] MySQL. *"What is MySQL?"* [online]. [quoted 2024-02-15]. Available from: https://www.oracle.com/mysql/what-is-mysql/#mysql-benefits

[20] GeekforGeeks. *"Disadvantages of using MySQL"* [online]. [quoted 2024-02-22]. Available from: https://www.geeksforgeeks.org/disadvantages-of-using-mysql/

[21] Margaret Rouse. Technopedia. *"Oracle Database"* [online]. [quoted 2024-02-23]. Available from: https://www.techopedia.com/definition/8711/oracle-database

[22] Alexander S. Gillis, Bridget Botelho. TechTarget. *"What is MongoDB?"* [online]. [quoted 2024-02-24]. Available from: https://www.techtarget.com/searchdatamanagement/definition/MongoDB

[23] GeekforGeeks. *"MongoDB advantages and disadvantages"* [online]. [quoted 2024-02-25]. Available from: https://www.geeksforgeeks.org/mongodb-advantages-disadvantages/

[24] Java. *"What is Java?"* [online]. [quoted 2024-05-12]. Available from: https://www.java.com/en/download/help/whatis_java.html

[25] Wikipedia. *"Kotlin programming language"* [online]. [quoted 2024-03-24]. Available from: https://en.wikipedia.org/wiki/Kotlin_(programming_language)

[26] Kolade Chris. freeCodeCamp. *"What is PHP?"* [online]. [quoted 2024-03-03]. Available from: https://www.freecodecamp.org/news/what-is-php-the-php-programming-language-meaning-explained/

[27] Wikipedia. *"Python programming language"* [online]. [quoted 2024-03-03]. Available from: https://en.wikipedia.org/wiki/Python_(programming_language)

[28] Python.org. *"What is Python? Executive Summary"* [online]. [quoted 2024-03-03]. Available from: https://www.python.org/doc/essays/blurb/

[29] Microsoft. *"Microservices architecture design"* [online]. [quoted 2024-03-05]. Available from: https://learn.microsoft.com/en-us/azure/architecture/microservices/

[30] Chandler Harris. Atlassian. *"Microservices vs. monolithic architecture"* [online]. [quoted 2024-03-05]. Available from: https://www.atlassian.com/microservices/microservices-architecture/microservices-vs-monolith

[31] ZenTao. ZenTao Blog. *"Monolithic Architecture"* [online]. [quoted 2024-03-04]. Available from: https://www.zentao.pm/blog/where-cloud-native-comes-from-and-where-its-going-1104.html

[32] IBM. *"Deployment diagrams"* [online]. [quoted 2024-03-04]. Available from: https://www.ibm.com/docs/en/rational-soft-arch/9.7.0?topic=diagrams-deployment

[33] Sato Kenta. Dev. *"The Simplicity of Basic Authentication"* [online]. [quoted 2024-03-06]. Available from: https://dev.to/satokenta/understanding-the-benefits-of-oauth-over-basic-authentication-36lj

[34] Ayush Shrivastava. LinkedIn. *"Spring Boot Security Implementation With Basic Auth"* [online]. [quoted 2024-04-04]. Available from: https://www.linkedin.com/pulse/spring-boot -security-implementation-basic-auth-ayush-shrivastava-plr6c/

[35] Javatpoint. *"Hibernate Tutorial"* [online]. [quoted 2024-04-05]. Available from: https://www.javatpoint.com/hibernate-tutorial

[36] Javatpoint. *"JPA Introduction"* [online]. [quoted 2024-04-05]. Available from: https://www.javatpoint.com/jpa-introduction

[37] GeekforGeeks. *"Apache Maven"* [online]. [quoted 2024-04-05]. Available from: https://www.geeksforgeeks.org/apache-maven/

[38] AYR Tech (Pty) Ltd. *"ExchangeRate-API"* [online]. [quoted 2024-05-03]. Available from: https://app.exchangerate-api.com/

[39] Alejandro Matos. *"REST Countries"* [online]. [quoted 2024-05-03]. Available from: https://restcountries.com/

[40] Red Hat. *"What is Apache Kafka?"* [online]. [quoted 2024-04-13]. Available from: https://www.redhat.com/en/topics/integration/what-is-apache-kafka

[41] Baeldung. *"Guide to Hazelcast with Java"* [online]. [quoted 2024-04-15]. Available from: https://www.baeldung.com/java-hazelcast

[42] Alexander S. Gillis. Techtarget. *"What is Docker Image?"* [online]. [quoted 2024-04-17]. Available from: https://www.techtarget.com/searchitoperations/definition/Docker-image

[43] JetBrains. *"IntelliJ IDEA"* [online]. [quoted 2024-04-19]. Available from: https://www.jetbrains.com/idea/

[44] Postdot Technologies. *"Postman"* [online]. [quoted 2024-04-19]. Available from: https://www.postman.com/

[45] *"Docker"* [online]. [quoted 2024-04-19]. Available from: https://www.docker.com/

[46] GeekforGeeks. *"Introduction to Semantic Versioning"* [online]. [quoted 2024-04-20]. Available from:
https://www.geeksforgeeks.org/introduction-semantic-versioning/

[47] Tutorialspoint. *"JUnit Overview"* [online]. [quoted 2024-04-25]. Available from: https://www.tutorialspoint.com/junit/junit_overview.htm

[48] Skillcombo. Medium. *"The Difference Between JUnit and Mockito"* [online]. [quoted 2024-04-25].
Available from: https://medium.com/@skillcombo/the-difference-between-junit-and-mockito-detailed-comparison-f5b86e062a25

[49] Dmytro Rastvorov. YouTube. *"MPBA - video demonstration"* [online]. [quoted 2024-05-05]. Available from: https://youtu.be/Q0lxQ8F4bYs