Міністерство освіти і науки України
Національний технічний університет
«Дніпровська політехніка»

Інститут електроенергетики
(інститут)

Факультет інформаційних технологій
(факультет)

Кафедра Програмного забезпечення комп'ютерних систем
(повна назва)

ПОЯСНЮВАЛЬНА ЗАПИСКА
кваліфікаційної роботи ступеня

бакалавра
(назва освітньо-кваліфікаційного рівня)

студента *Швеця Владислава Ігоровича*
(ПІБ)

академічної групи *121-18-1*
(шифр)

спеціальності *121 Інженерія програмного забезпечення*
(код і назва спеціальності)

освітньої програми *Інженерія програмного забезпечення*
(назва освітньої програми)

на тему: *Розробка програмного забезпечення для фінансового контроля і керування банківським рахунком засобами Java, SpringBoot, FasterXML*

Керівники	**Прізвище, ініціали**	**Оцінка за шкалою**		**Підпис**
		рейтинговою	**інституційною**	
кваліфікаційної роботи	*проф.Бердник М.Г.*			
розділів:				
спеціальний	*проф.Бердник М.Г.*			
економічний	*доц. Касьяненко Л.В.*			
Рецензент	*доц. Шедловський І.А.*			
Нормоконтролер	*доц. Гуліна І.Г.*			

Дніпро
2022

Міністерство освіти і науки України
НТУ «Дніпровська політехніка»

ЗАТВЕРДЖЕНО:
завідувач кафедри
програмного забезпечення комп'ютерних систем
(повна назва)

_____ І.М. Удовик
(підпис) (прізвище, ініціали)

« » _____ 2022 року

ЗАВДАННЯ
на кваліфікаційну роботу

бакалавра
(назва освітньо-кваліфікаційного рівня)

студента *121-18-1* (група) *Швеця Владислава Ігоровича* (прізвище та ініціали)

тема кваліфікаційної роботи *Розробка програмного забезпечення для фінансового контроля і керування банківським рахунком засобами Java, SpringBoot, FasterXML*

затверджена наказом ректора НТУ «ДП» від *18 травня 2022р.* № *268-с*

Розділ	Зміст виконання	Термін виконання
Спеціальний	***На основі матеріалів проєктно-технологічної та інших науково-технічних джерел провести аналіз стану рішення проблеми та постановку задачі. Обґрунтувати вибір та здійснити реалізацію методів вирішення проблеми***	***13.05.2022 р.***
Економічний	***Провести розрахунок трудомісткості розробки програмного забезпечення, витрат на створення ПЗ й тривалості його розробки***	***27.05.2022 р.***

Завдання видав _____ (підпис) *проф.Бердник М.Г.* (посада, прізвище, ініціали)

Завдання прийняв до виконання _____ (підпис) *Швець В.І.* (прізвище, ініціали)

Дата видачі завдання: *14.01.2022 р.*

Термін подання кваліфікаційної роботи до ЕК: *13.06.2022 р.*

РЕФЕРАТ

Пояснювальна записка: 39 с., 21 рис., 0 табл., 4 дод., 25 джерел.

Об'єкт розробки: клієнт-серверний додаток.

Мета кваліфікаційної роботи: контроль банківських рахунків за допомогою розробленого додатку.

У вступі розглядається стан проблеми на сьогодняшній день, встановлюється мета кваліфікаційної роботи, актуальність та галузь її застосування, уточнюються цілі завдання.

У першому розділі проаналізовано предметну галузь, визначені актуальні завдання та призначення розробки, сформульовано завдання, зазначено вимоги до програмної реалізації, технологій та програмних засобів.

У другому розділі проаналізовані наявні рішенн, обрано платформу для розробки, спроєктовано та розроблено програмний продукт, описана робота програми, алгоритми і структура її функціонування, а також виклик та завантаження програми, охарактеризовано вихідні та вхідні дані, визначено склад параметрів технічних засобів.

В економічному розділі визначено трудомісткість розробленого програмного продукту, підраховано вартість роботи по створенню програми та розраховано час на його створення.

Практичне значення полягає у створенному додатку, що надає можливість контролюти особові рахунки.

Актуальність програмного продукту визначається великим попитом на подібні розробки, що дає можливості для створення зручного продукту для управління рахунком.

Список ключових слів: КОМП’ЮТЕР, ПРОГРАМНИЙ ПРОДУКТ, МЕНЮ, ДОДАТОК, ПРОЄКТУВАННЯ.

ABSTRACT

Explanatory note: 39 pages, 21 figures, 0 tables, 4 appendices, 25 sources.

Object of development: client-server application.

The purpose of the qualification work: control of bank accounts for using the developed application.

The introduction considers the state of the problem today, establishes the purpose of the qualification work, the relevance and scope of its application, clarifies the objectives.

The first section analyzes the subject area identified current tasks and purposes of development, tasks are formulated, requirements for software implementation, technologies and software are specified funds.

The second section analyzes the available solutions, selected a platform for development, designed and developed a software product, describes the program, algorithms and structure of its operation, as well as calling and loading the program, describes the output and input data, determines the parameters of hardware.

In the economic section the complexity of the developed software product is determined, the cost of work on creation is calculated programs and the estimated time for its creation.

The practical significance lies in the created application that provides an opportunity control personal accounts.

The relevance of the software product is determined by the high demand for similar developments, which provides opportunities to create a convenient product for account management.

List of keywords: COMPUTER, SOFTWARE PRODUCT, MENU, APPENDIX, DESIGN.

СПИСОК УМОВНИХ ПОЗНАЧЕНЬ

ОС – операційна система;

БД – база даних;

ІТ – інформаційні технології;

СУБД – система управління базою даних;

ПК – персональний компю’тер;

API – Application Programming Interface;

MVC – Model-View-Controller;

DAO – Data access object:

ЗМІСТ

ВСТУП

Складно уявити сьогоднішнє життя без інтернету та технологій. Через це у наш час не залишилось областей, в яких не використовують сучасні технології. Використання інформаційних технологій поліпшує не тільки ефективність підприемницької діяльності, але і життя звичайних людей.

Розвиток інтернету вплинув на різні сфери і банківська система не є виключенням. За допомогою різноманітних технологій банки змогли підвищити ефективність виконання фінансових операцій та розширити клієнтську базу. На сьогоднішній день банки по усьому світу по більшості використовують систему для дистанційного надання доступу до послуг и передачі інформації. Такі систему мають назву «Інтернет-банкінг»[1]. З їх допомогою клієнт, лише при наявності доступу до інтернет мережі, може виконувати різноманітні операції з своїм рахунком. При використанні інтернет-банкінгу спрощується консультація та обслуговування клієнта, так як зникає потреба у відвідуванні спеціальних відділень. Через це банки можуть зменшити затрати за допомогою зменшення кількості філіалів і перевести на консультацію через додаток або контакт-центри, що буде вигідніше, ніж оплата оренди та іншого. На цю різницю банки можуть запроваджувати нові фішки для клієнтів або зменшувати комісії за використання рахунків. На цей час у нас в Україні працює і активно розвивається банк який взагалі не має відділень, цей банк надає інформацію по рахунку своїм клієнтам через інтернет-банкінг.

Однак на мій погляд банки сьогодення надають досить обмежену інформацію за рахунками і мають загальний напрямок, так як це зручно для більшості клієнтів, але є певна категорій яким потрібно планувати свої затрати і за допомого інтернет-банкінгу на це потрібно витрачати декілька годин свого часу. Тому метою дипломного проекту є створення додатку, через який клієнт сможе отримувати більш специфічні дані за усіма рахунками та передивлятись аналітичні зведення.

Для розробки програмного застосунку була використана мова Java, так як вона досить зручна і є одною з найпопулярніших мов, через це має велику кількість доступної інформації в інтернеті.

Згідно вище описаної інформації була визначена тема дипломної роботи: ”Розробка програмного забезпечення для фінансового контроля і керування банківським рахунком засобами Java, SpringBoot[3], FasterXML[5]”.

РОЗДІЛ 1
АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ПОСТАНОВКА ЗАДАЧІ

1.1. Загальні відомості с предметної галузі

Інтернет-банкінг[1] – це один із видів дистанційного банківського обслуговування, який надає можливість отримувати доступ до рахунків та банківських операцій через Інтернет. Усього можна виділити чотири основних типи дистанційного банківського обслуговування: PC-Банкінг, SMS-банкінг, телефонний банкінг, та інтернетбанкінг.

PC-Банкінг – система, яка була створена в більшій частині для юридичних осіб. Для використання такої системи потрібне спеціальне обладнання та програмне забезпечення, як зі сторони банку, так і зі сторони клієнту. Взаємодія з банком здійснювалася через спеціальний модем, який напряму з'єднувався з сервером банку. Наразі ця система майже не використовується, оскільки не є зручною та потребує спеціального обладнання.

Телефонний банкінг – це канал доступу, який дозволяє керувати рахунком, та проводити операції з рахунком за допомогою телефону. Ця система навіть на сьогодняшній день користується певною популярністю, оскільки для її використання потрібен лише телефон з наявним сотовим зв'язком. Взаємодія відбувається або через зв'язок з Call-центром, або через голосове меню. Але її недоліком є те, що може бути черга для св'язку с оператором і очікування клієнта на запит може бути значно довшим довгим. А проведення операцій через голосове меню, може бути не зрозумілим та не повним для деяких клієнтів через відсутність зручного графічного інтерфейсу.

SMS банкінг – спілкування між клієнтом і банком здійснюється з використанням SMS-повідомлень. Клієнта інформують про більшість операції. Також клієнт через відправку повідомлень на спеціальний номер може виконувати перекази між своїми рахунками, оплачувати послуги мобільного

зв’язку та інше. Але цей спосіб також не є досконалим через велику кількість обмежень функціоналу та незручність графічної передачі інформації.

Інтернет-банкінг[1] – дав змогу зручного доступу до банківського рахунку через Інтернет за допомогою ПК або телефону на яких є змога встановити вебпереглядач, або спеціальний додаток від банку. Через ці засоби клієнти можуть отримувати усю необхідну інформацію за рахунками та здійснювати необхідні операції. Після впровадження інтернет-банкінгу значно зросла якість та швидкість обслуговування, оскільки зменшились черги у відділеннях банків. Також оскільки система інтернет-банкінгу автоматизована, то клієнт може отримати потрібні послуги в будь-який час, на відміну, від банківских відділень які працюють згідно заданого графіку.

Але окрім цих основних видів дистанційного надання інформації є ще один досить розповсюджений варіант - це персональне API банку.

API (Application Programming Interface)[2] - це опис способів (набір класів, процедур, функцій, структур або констант), якими одна комп'ютерна програма може взаємодіяти з іншою програмою. Зазвичай входить до опису будь-якого інтернет-протоколу, програмного каркасу або стандарту викликів функцій операційної системи. API спрощує процес програмування при створенні програм, абстрагуючи базову реалізацію та надаючи лише об'єкти або дії, необхідні розробнику. Якщо графічний інтерфейс для банківського клієнта може надати користувачеві кнопку, яка виконає всі кроки для вибірки банківських оплат, то API для введення/виводу файлів може дати розробнику функцію, яка копіює файл з одного місця в інше, не вимагаючи від розробника розуміння операцій файлової системи. що відбуваються за лаштунками.

Розглянемо деякі API популярних банків, які використовують системи інтернет-банкінгу, їх переваги та недоліки.

Raiffeisen Bank Aval – на офіційному сайті можна знайти детальну документацію по використанню API (рис. 1.1). При ознайомлені с інформацією я зміг виділити основні переваги, такі як:

- Велика кількість різних запитів як для back-end рішень, так і для front-end, а саме GUI форми, які досить зручні у використанні.
- Детальна документація по можливим відповідям на запит до банківского API.

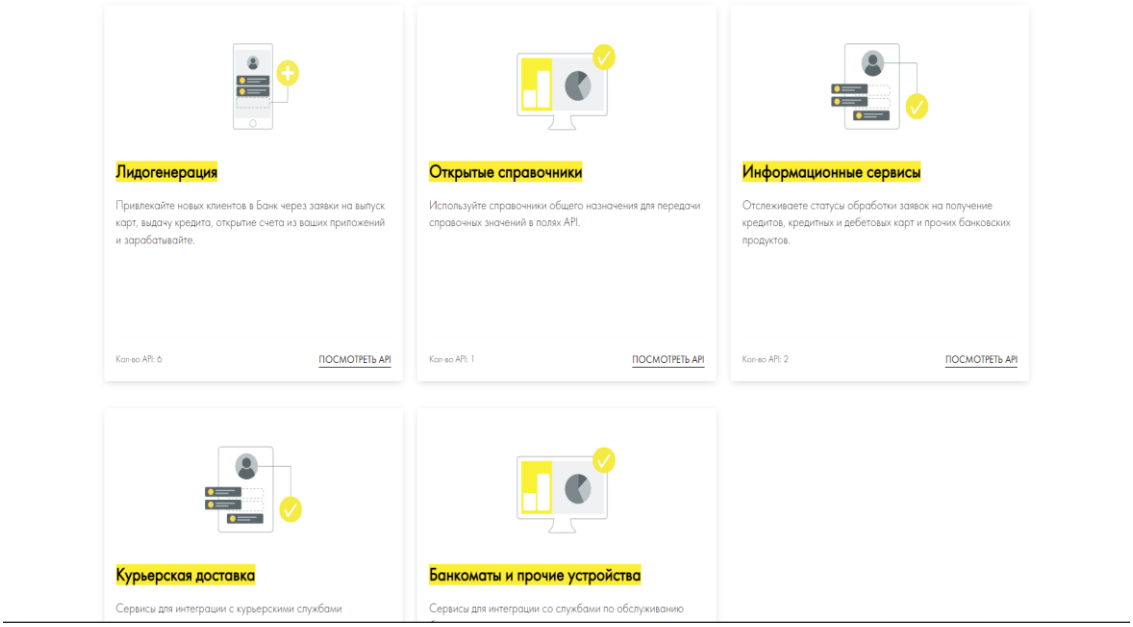

Рис. 1.1. Документація Raiffeisen Bank Aval API

Однак не дивлячись на ці переваги, є досить неприємний недолік, а саме потреба в заключенні спеціального договору на використання API (рис. 1.2). Тому це не зовсім зручний спосіб для звичайного користувача.

Рис. 1.2. Правила під’єднання до Raiffeisen Bank Aval API

Monobank – останнім часом дуже швидко набирає популярність серед користувачів інтернет-банкінгу. Також Monobank є першим банком в Україні, який взагалі не має банківських відділень, тому у цьому випадку їм потрібно було звернути особливу увагу на систему інтернет-банкінгу, так як вона буде основною. В більшій частині мобільний додаток виконує усі потреби клієнта, однак деякої інформації, яка б автоматично підраховувалась, може не вистачати. Ось тут встають у нагоді банківські API, які допоможуть отримати недостатню інформацію.

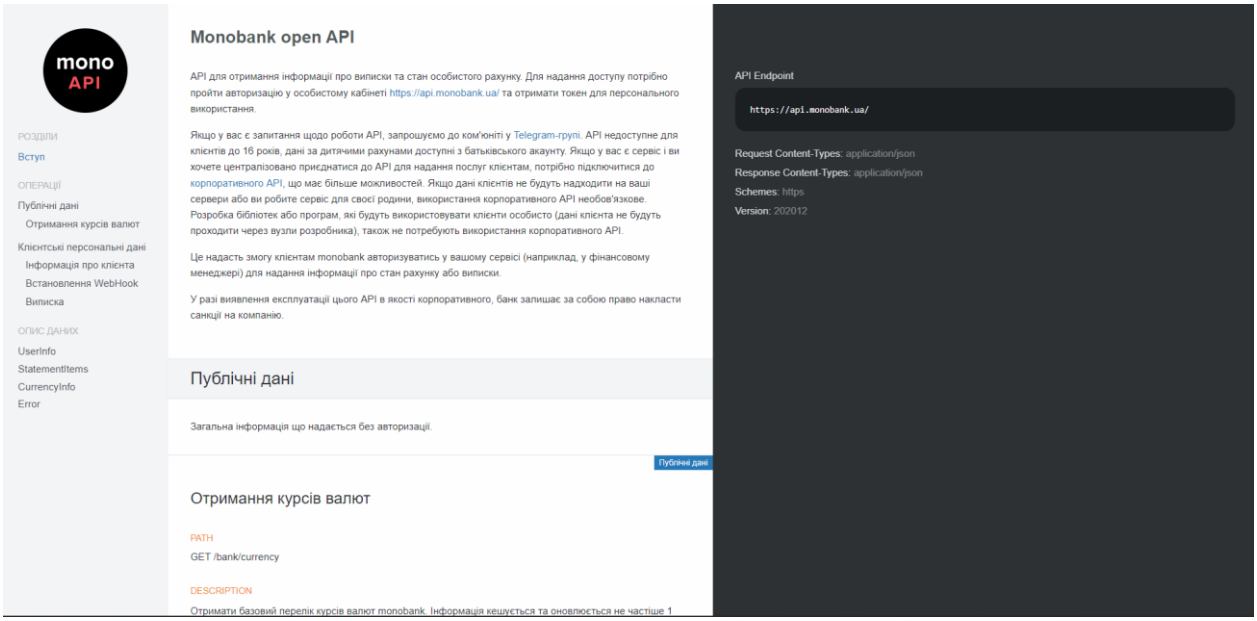

Рис. 1.3. Документація Monobank API

Звернувшись до офіційного сайту, можна легко знайти документацію по API від Monobank (рис. 1.3). Ознайомившись з сторінкою, я помітив певні переваги:

- Простий спосіб написання запитів.
- Одразу біля запиту відображають приклад успішної відповіді від банку.
- Простий спосіб написання запитів.
- Зручний спосіб під’єднання до API, а саме відсутність додаткових реєстрацій для отримання токену.

Якщо казати про недоліки, то я б виділив невелику різноманітність запитів, які надають лише основну інформацію по стану банківських рахунків. Але за допомогою цих даних можна сформувати вибірку, яка буде задовольняти потреби клієнта.

1.2. Призначення розробки та область застосування

Головною метою роботи є створення додатку, за допомогою якої клієнт банку буде мати змогу отримувати персоналізовану інформацію за своїми рахунками. Додаток має бути простим у використанні та надавати потрібний перелік інформації у зрозумілій формі. Програма призначена для:

- Отримання детальної інформації по виписці.
- Управління персональними фільтрами витрат.
- Перегляд залишку на рахунках.

Програма позиціонується як сервер для Web-сайту, за допомогою якого клієнти банку сможуть в особистий кабінет додатку перегляду обрану їм інформацію.

1.3. Підстава для розробки

Відповідно до ОКХ та ОПП, згідно навчального плану та графіків навчального процесу, в кінці навчання студент виконує кваліфікаційну роботу (проект). Тема роботи узгоджується з керівником проекту, випускаючою кафедрою та затверджується наказом ректора.

Отже підставами для розробки (виконання кваліфікаційної роботи) є:

– ОКХ та ОПП за напрямом підготовки 6.050101 «Комп’ютерні науки»;
– навчальний план та графік навчального процесу;
– наказ ректора Національного технічного університету;
– «Дніпровська політехніка» №268-с від 18.05.2022 року;

– завдання на дипломний проект на тему: «Розробка програмного забезпечення для фінансового контроля і керування банківським рахунком засобами Java, SpringBoot, FasterXML ».

1.4. Постановка завдання

Метою дипломного проекта є розробка серверу для Web-сайту, для системи аналізу банківського рахунку. Програма призначена для швидкого отримання актуальної інформації за рахунком та персоналізованого аналізу.

Додаток повинен мати функції:

- Виводити інформацію за певним рахунком.
- Отримання виписки за вказаний період та її аналіз.
- Отримання актуального курсу валют.
- Налаштовувати список персональних категорій.

Для виконання проекту необхідно:

- Розробити структуру системи.
- Розробити зручний доступ до даних через додаток.
- Реалізувати back-end систему.

1.5. Вимоги до програми або програмного виробу

1.5.1. Вимоги до функціональних характеристик

Для виконання поставлених завдань, розроблений програмний продукт повиннен підтримувати виконання таких дій:

- Збереження змін здійснених користувачем.
- Автоматичне оновленя інформації.
- Зручна навігація у додатку.
- Отримувати інформацію з даними про рахунки через банківське API.

Для повного функціювання вище заявлених задач у додатку повинно бути реалізовано:

- Підтримка API та доступ до нього через програму.
- Програмна та апаратна сумісності.
- Конфігурація за допомогою якої застосунок приводиться до експлуатації.

1.5.2. Вимоги до інформаційної безпеки

Для справної роботи програмного комплексу потрібно реалізувати:

- Можливість редагування даних.
- Збереження цілосності даних.

Згідно з поставленими цілями, треба обмежити доступ клієнта до функцій і інформації, які доступні після отримання її від банку.

1.5.3. Вимоги до складу та параметрів технічних засобів

Для працездатності системи необхідно, щоб обчислювальна машина, на якій буде застосовуватись програма, мала наступні рекомендовані характеристиками:

- Маніпулятор “миша”.
- Клавіатура.
- Доступ до мережі інтернет.
- 2 Гб вільного місця на жорсткому диску.
- Процессор AMD Ryzen 3 1200 3.2 GHz
- Не менше 2 Гб оперативної пам’яті.
- Монітор з діагоналлю 15”.

Характеристики наведені вище уявляють собою рекомендовані. При наявності характеристик не нижче зазначених, програмний продукт буде працювати згідно до вимог щодо надійності, безпеки та часу обробки даних.

1.5.4. Вимоги інформаційної та програмної сумісності

Для корректної експлуатації програми необхідне наступне програмне забезпечення, встановлене на ПК:

- Операційна система Windows 7/10/11;
- Java Version 8 Update 333 або більш актуальна.

Back end додатку повинен бути реалізований на мові програмування Java з використанням SpringBoot, FasterXML.

Інформаційна система банківського API може допомогти у надані клієнтам банку додаткової інформації за їх рахунком. У результаті чого може бути покращений досвід від користування послугами. Для розробки вказаної системи потрібно використовувати актуальні способи розробки для виконання поставлених цілей і задоволення клієнтів від користування застосунком.

РОЗДІЛ 2
ПРОЄКТУВАННЯ ТА РОЗРОБКА ІНФОРМАЦІЙНОЇ СИСТЕМИ

2.1. Функціональне призначення програми

Програма призначена для клієнтів, які за допомогою розробленного серверу для Web-сайту, зможуть передивлятися інформацію більш детальну інформацію за власними рахунками. Сервер для Web-сайту повинен реалізовувати наступні функції:

– функціональні кнопки для виведення проаналізованної інформації по рахунку;

– зрозумілий інтерфейс для користувача;

– форму авторизації по API - токену для користувачів;

– функціональні кнопки для виведення загальної інформації по рахунку.

Додаток може бути використаний клієнтами банку, яким потрібні поглиблені данні для проведення обліку рахунку та збирання статистики.

2.2. Опис застосованих математичних методів

Оскільки особливостями предметної галузі програмного продукту для інтернет банкінгу не передбачають застосування математичних методів при розробці, через це математичні методи не були використані.

2.3. Опис використаної архітектури та шаблонів проєктування

При розробці програми була використана структура Модель-вигляд-контролер (MVC – Model-view-controller)[7] - архітектурний шаблон[6], який використовується під час проєктування та розробки програмного забезпечення.

Цей шаблон передбачає поділ системи на три взаємопов'язані частини: модель даних, вигляд (інтерфейс користувача) та модуль управління (рис. 2.1.). З допомогою цього шаблону можно легко відокремити основну модель від користувацького інтерфейсу.

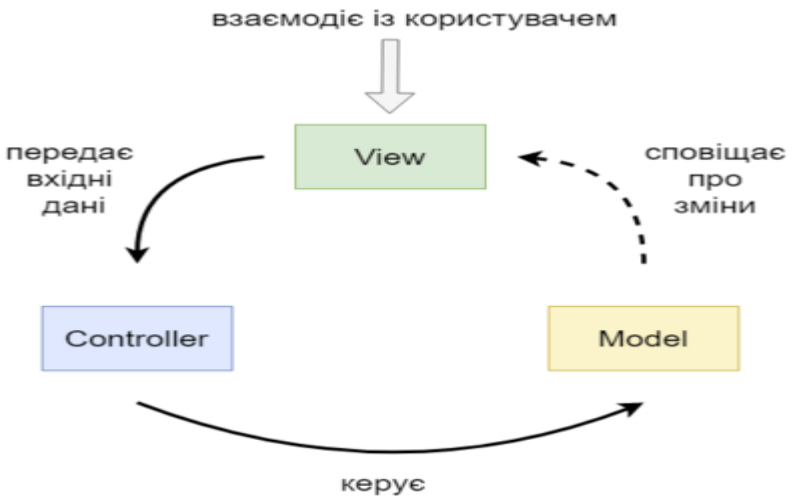

Рис. 2.1. Діаграма взаємодії між компонентами шаблону MVC

2.4. Опис використаних технологій та мов програмування

Для розробки серверу для Web-сайту була обрана мова програмування Java. У якості середовища розробки - IntelJIdea 2022 Ultimate edition[8]

Java — це високорівнева, заснована на класах, об’єктно-орієнтована мова програмування, яка розроблена так, щоб мати якомога менше залежностей реалізації. Це мова програмування загального призначення, призначена для того, щоб програмісти могли писати один раз, запускати в будь-якому місці, що означає, що скомпільований код Java може працювати на всіх платформах, які підтримують Java, без необхідності перекомпіляції. Програми Java зазвичай компілюються у байт-код, який може працювати на

будь-якій віртуальній машині Java (JVM) незалежно від базової архітектури комп'ютера.

Для використання сторонніх API у програмі, була використана додаткова бібліотека залежностей – FasterXML, так як більшість API повертає відповідь у форматі JSON, ці дані потрібно буде перетворювати на об’єкти програми і ця бібліотека надає найпростіший та найшвидший варіант взаємодії з цими даними.

Для перевірки алгоритмів програми був використаний фреймворк Junit[9], з його допомогою можливо проводити модульне тестування. Модульне тестування - процес у програмуванні, що дозволяє перевірити на коректність окремі модулі вихідного коду програми.

Для взаємодії серверної частини з моделями керування використовується документо-орієнтовна база даних MongoDB[4], оскільки основними вхідними та вихідними даними для цієї БД є документи у форматі JSON, які використовуються при використанні API.

2.5. Опис структури програми та алгоритмів її функціонування

Структура проекту була розроблена за стандартами Java EE.

На рис. 2.2. наведена структура файлів розробленого проекту.

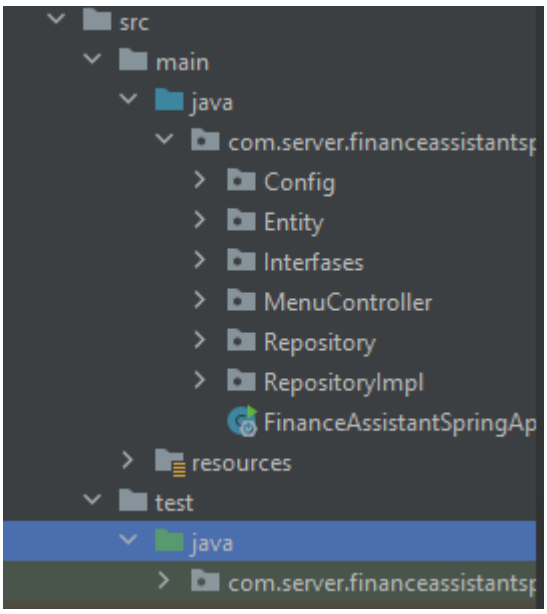

Рис. 2.2. Основна структура файлів програми

На рис. 2.3. відображена структура файлів Config. В цьому пакеті зберігаються додаткові налаштування серверної частини[10].

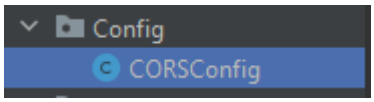

Рис. 2.3. Структура пакету Config

На рис. 2.4. відображена структура файлів Entity. Entity це основні об’єкти, які будуть зберігати або опрацьовувати дані користувача отримані через сторонні API.

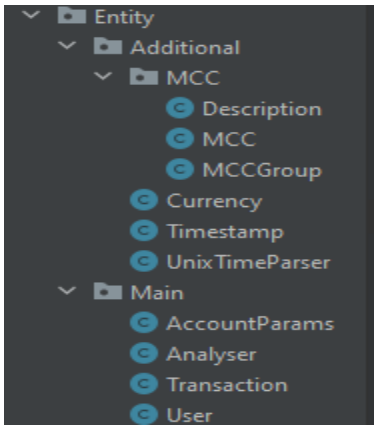

Рис. 2.4. Структура пакету Entity

На рис. 2.5. відображена структура файлів Interfaces[11]. В об’єктах цього пакету файлів описані функціональні можливості кожного меню користувача, якими буде виконуватись пересування по програмі.

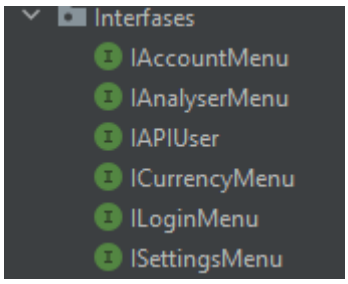

Рис. 2.5. Структура пакету Interfaces

На рис. 2.6. відображена структура файлів MenuController. Об’єкти пакету меню реалізовують вказаний функціонал в пакеті Interfaces за допомогою RestAPI[12].

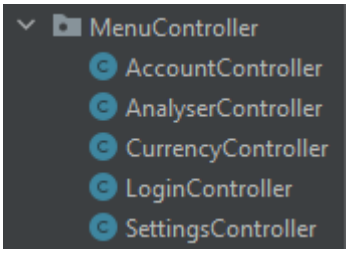

Рис. 2.6. Структура пакету Menu

На рис. 2.7. відображена структура файлів Repository[13]. Repository виступає у ролі посередника між шаром домену та шаром відображення реляційних даних. Він виконує роль колекції об'єктів домену в оперативній пам'яті. Таким чином, репозиторій є більш високим рівень абстракції над шаром відображення даних.

Repository є орієнтованим на модель предметної області, тоді як DAO більше орієнтований на джерело даних.

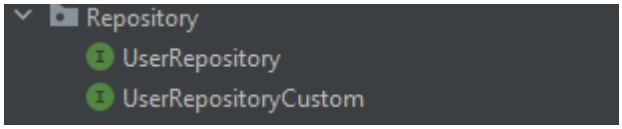

Рис. 2.7. Структура пакету Repository

На рис. 2.8. відображена структура файлів RepositoryImpl. У цьому пакеті реалізовуються алгоритми звернення та редагування полів у базі даних[14].

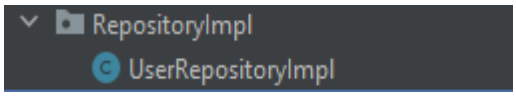

Рис. 2.8. Структура пакету RepositoryImpl

На рис. 2.9. відображена структура файлів resources. В рамках цього пакету зберігаються основні налаштування серверу[15], файл с загальноприйнятими категоріями витрат[4] та файл з резервними курсами валют[16] на випадок якщо Monobank API не зможе повернути актуальні дані.

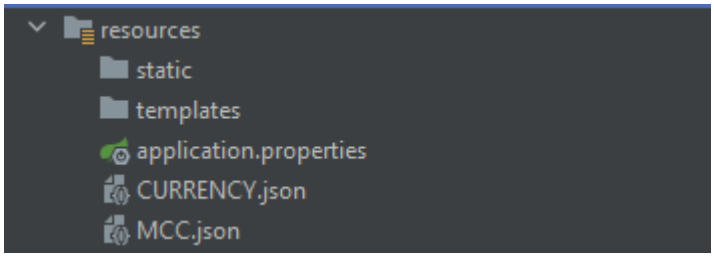

Рис. 2.9. Структура пакету resources

На рис. 2.10. відображена структура файлів test. Для модельного тестування створюються спеціальні об’єкти які будуть виконувати функції основних об’єктів і перевіряти коректність їх роботи.

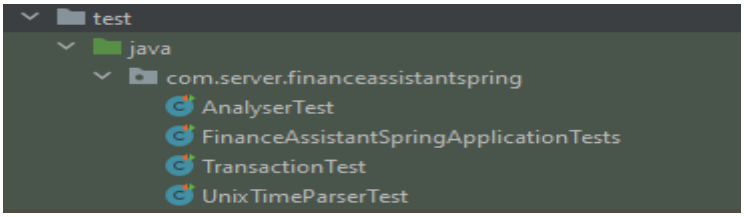

Рис 2.10. Структура пакету test

2.6. Обгрунтування та організація вхідних та вихідних даних програми

Згідно з цілями та методами розробки у якості вхідних даних, у розробленому додатку, будуть використовуватись дані отримані в результаті звернення до банківського API. Також у програмі будуть використовуватись дані, які будуть вводитись користувачем через клавіатуру на сайті.

Вихідні дані уявлятимуть собой текстову інформацію, яка буде відображатись на Web-сайті, який буде отримувати дані від серверу.

2.7. Опис розробленої системи

2.7.1. Використані технічні засоби

Розроблена програма може виконуватись на пристроях з будь якою операційною системою, яка підтримує Java, на різних типах персональних комп’ютерів чи ноутбуків.

Для запуску сервер для Web-сайту, потрібен ПК який зможе забезпечити належну продуктивність для відповідної роботи. У нього мають бути наступні мінімальні характеристики:

- Доступ до мережі інтернет.
- 2 Гб вільного місця на жорсткому диску.
- Процессор AMD Ryzer 3 1200 3,20 GHz.
- Не менше 2 Гб оперативної пам’яті.
- Монітор з діагоналлю 15".

2.7.2. Використані програмні засоби

Для запуску та експлуатації програми потрібні наступні програмні засоби:

- Встановлене середовище виконання IntelliJ IDEA 2022;
- Встановлене JDK(Java Development Kit) 17.0.3.1;
- Веб-браузер.

2.7.3. Виклик та завантаження програми

Для початку роботи з програмою, потрібно запустити сервер для Web-сайту за допомогою програмного забезпечення IntelliJ IDEA 2022. Потрібно запустити скрипт з назвою “FinanceAssistantSpringApplication”. Після в IntelliJ IDEA повинно з’явитись вікно с консоллю, яка автоматично запустить розроблений сервер. Після виконання цих маніпуляцій сервер повністю готовий до використання.

2.7.4. Опис інтерфейсу користувача

Після повного запуску можливо використовувати усі функції серверу. При переході на веб-сторінку повинен відображатись екран авторизації користувача. На цьому екрані, якщо клієнт ще не авторизований, потрібно ввести персональний токен для доступу до Monobank API(рис. 2.11.).

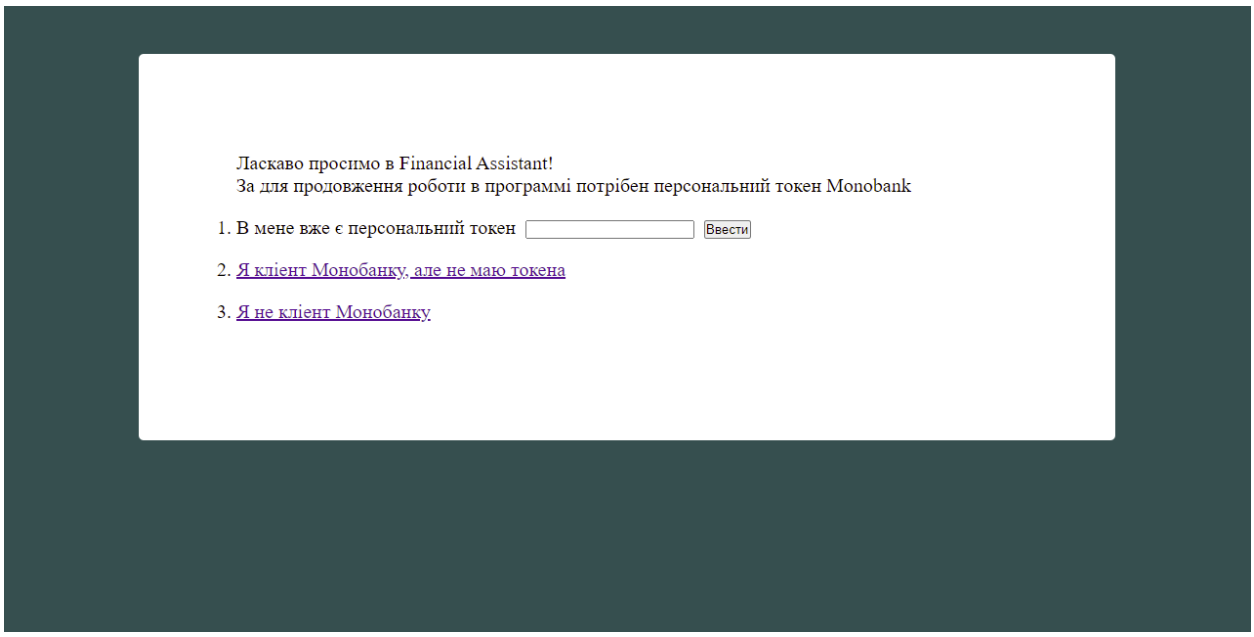

Рис. 2.11. Вікно авторизації

Код реалізації входу користувача до системи наведений нижче:

```
@PostMapping("/login")
@Override
public User Login(@RequestParam(value = "token") String token) throws IOException {
   User client = new User();
   client.setPersonalToken(token);
   client.setMccListByGroup(MCC.mccFill());
   try{
   JsonNode data = new ObjectMapper().readTree(client.apiCall());
   client = client.mapper.readValue(data.toString(), User.class);
   client.setPersonalToken(token);
   if (userRepository.findByClientId(client.getId())!= null){
      User dbclient = userRepository.findByClientId(client.getId());
      client.setPersonalMCC(dbclient.getPersonalMCC());
   }
   else userRepository.save(client);
   return client;
   }
   catch (Exception exception){
      throw new ResponseStatusException(HttpStatus.CONFLICT, "login.post.conflict");
   }
}
```

У разі якщо користувач введе невірний токен, сервер поверне помилку[17] (рис. 2.12.).

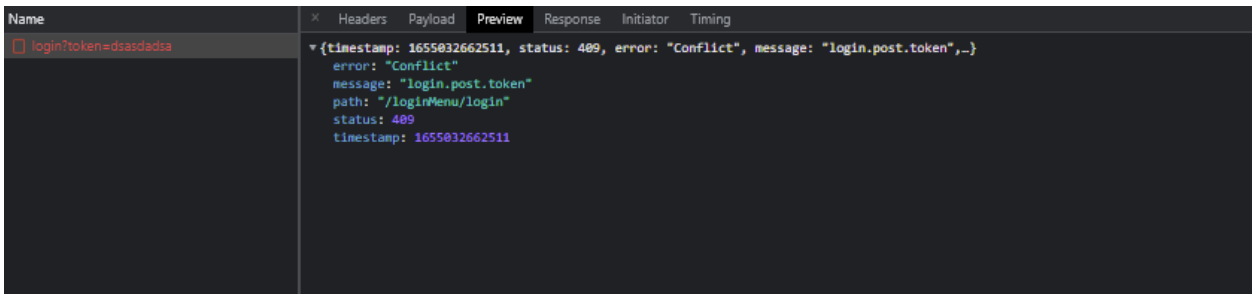

Рис. 2.12. Помилка від серверу

Після авторизації клієнта переводить на головне меню в якому відображається стан балансу по всім рахункам і стан заборгованості по кредитному ліміту. Окрім загальної інформації відображатимуться кнопки для переходу в інші розділи меню(рис. 2.13.).

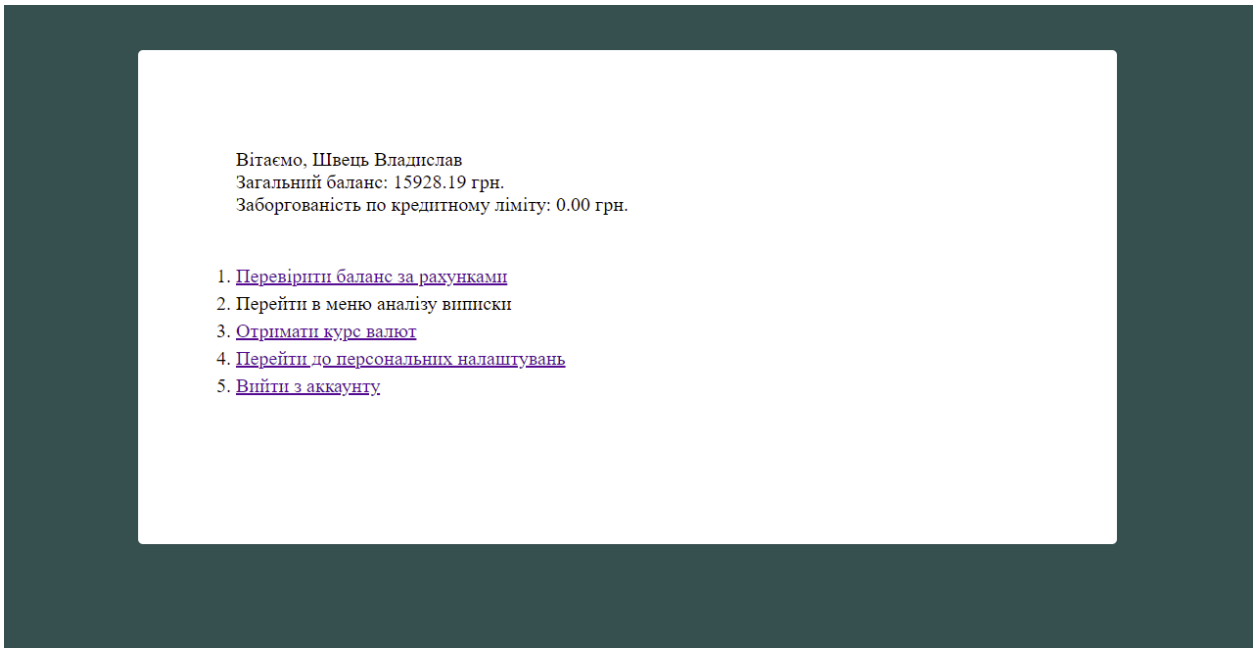

Рис. 2.13. Головний екран веб-інтерфейсу

Для отримання балансу по кожній окремій карті клієнта потрібно перейти у відповідний розділ. В цьому розділі відображається номер рахунку, валюта в якій зберігаються кошти і баланс самої карти (рис. 2.14.).

Рис. 2.14. Інформація за рахунками

Щоб перевірити курс валют клієнт натискає на відповідну кнопку меню, після цього з’являється таблиця з актуальними курсами (рис. 2.15.).

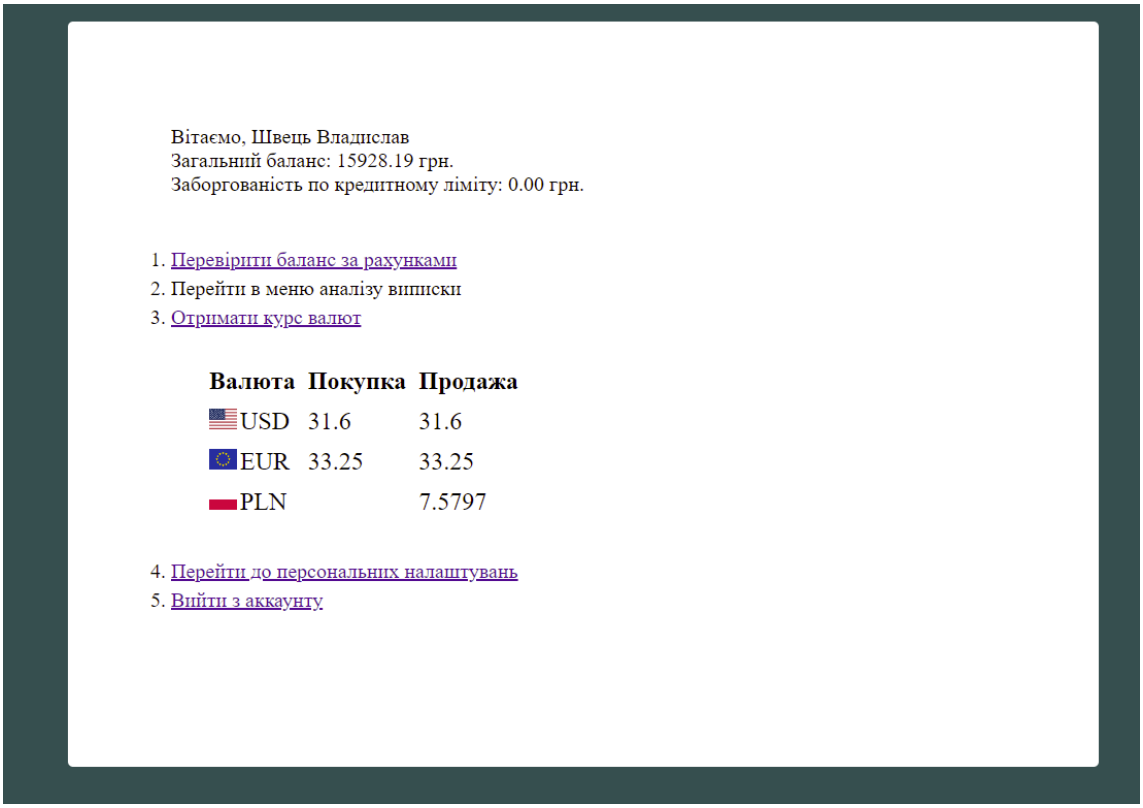

Рис. 2.15. Курс валют

У розділі персональних налаштувань користувач зможе додати, видалити або редагувати існуючи категорії витрат (рис. 2.16.).

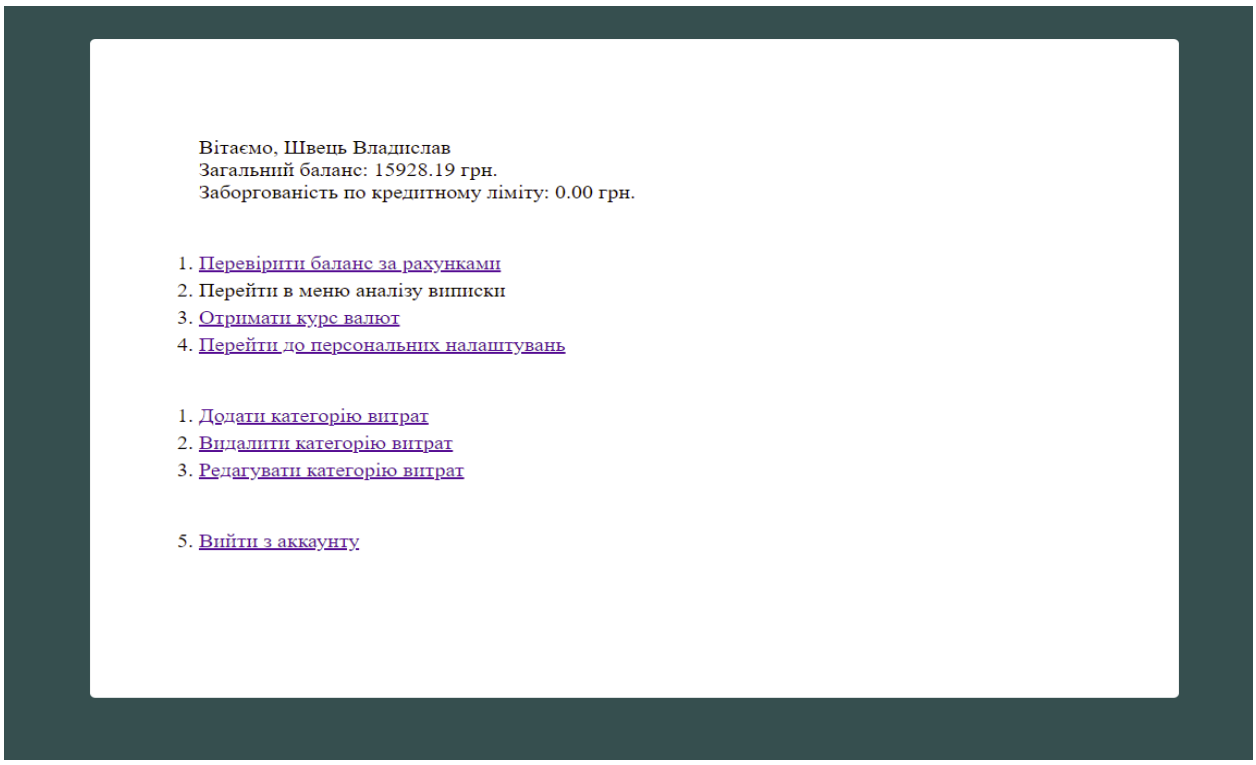

Рис. 2.16. Меню персональних налаштувань

Останнім і головним меню є меню аналізу виписки. В цьому меню клієнт зможе обрати рахунок за яким він хоче перевірити дані та вказати період[18] за яким буде аналізуватись дані (рис. 2.17. та рис. 2.18.).

Доступні варіанти аналізу даних:

- Сума поповнень/витрат.
- Оборот коштів.
- Порівняння поповнень/витрат/обороту коштів.
- Прогнозований розрахунок.
- Затрати по певним категоріям.
- Список витрат за вказаний період.

Рис. 2.17. Меню аналізу виписки

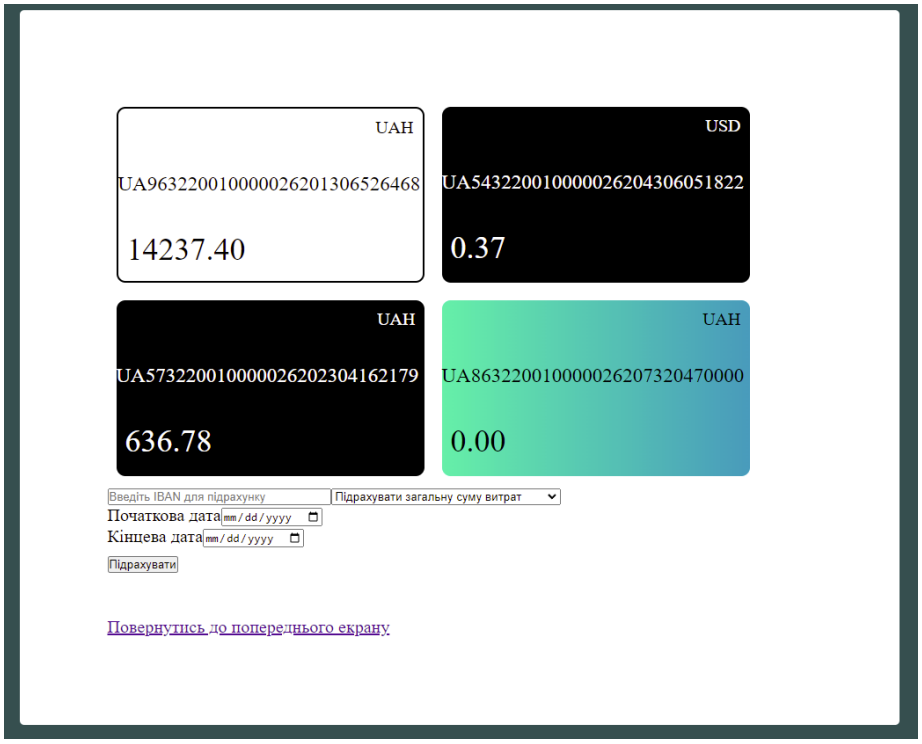

Рис. 2.18. Меню підрахунку виписки

РОЗДІЛ 3
ЕКОНОМІЧНИЙ РОЗДІЛ

Вихідні дані розробки програмного забезпечення:

1) Передбачуване число операторів – 1339;
2) Коефіцієнт складності програми – 1,30;
3) Коефіцієнт корекції програми в ході її розробки – 0,067;
4) Середня годинна заробітна плата програміста, грн/год – 126. Середня зарплата за годину роботи програміста була вирахувати виходячи з даних статистики зарплат професії "Junior Java developer в Україні"[19]. Середньо-українська заробітна плата програміста, який пише програми за допомогою мови програмування Java і з досвідом роботи двох місяців дорівнює 22120 грн в місяць. При восьмигодинному робочому дні (176 годин в місяць в середньому) середня заробітня плата за годину буде становити 126 грн;
5) коефіцієнт кваліфікації програміста, обумовлений від стажу – 0,8;
6) вартість машино-години ЕОМ, грн/год – 4,3;

3.1. Визначення трудомісткості розробки програмного забезпечення

Трудомісткість розробки ПЗ можна розрахувати за формулою:

$$t = t_0 + t_u + t_a + t_n + t_{отл} + t_д, \text{ людино-годин,} \quad (3.1)$$

де *to* – затрати праці на підготовку й опис визначеної задачі(приймається 60 людино-годин);

tu – затрати праці на дослідження алгоритму вирішення задачі,

ta – затрати праці на розробку блок-схеми алгоритму,

tn – затрати праці на програмування по готовій блок-схемі,

тотл – затрати праці на налаштуваня програми на ЕОМ,

tд – затрати праці на підготовку документації.

Складові затрати праці визначаються через умовне число операторів у ПЗ, яке розробляється.

Умовне число операторів (підпрограм):

$$Q = q \cdot C(1 + p), \text{ людино-годин,} \quad (3.2)$$

де q – передбачуване число операторів,

C – коефіцієнт складності програми,

p – коефіцієнт кореляції програми в ході її розробки.

$Q = 1339 \cdot 1{,}30(1 + 0{,}067) = 1857{,}33$

Затрати праці на вивчення опису задачі *tu* визначається з урахуванням уточнення опису і кваліфікації програміста:

$$t_u = \frac{Q \cdot B}{(75..85) \cdot k}, \text{людино} - \text{годин}, \quad (3.3)$$

де B, яке дорівнює 1,35, – коефіцієнт збільшення витрат праці внаслідок недостатнього опису задачі;

k, яке дорівнює 0,8, – коефіцієнт кваліфікації програміста, обумовлений стажем роботи з даної спеціальності.

$$t_u = \frac{1857{,}33 \cdot 1{,}35}{79 \cdot 0{,}8} = 39{,}67, \text{ людино-годин}$$

Затрати праці на розробку алгоритму рішення задачі:

$$t_a = \frac{Q}{(20..25) \cdot k}; \quad (3.4)$$

$$t_a = \frac{1857{,}33}{22 \cdot 0{,}8} = 105{,}53, \text{ людино-годин}$$

Затрати на складання програми по готовій блок-схемі:

$$t_n = \frac{Q}{(20..25) \cdot k}, \quad (3.5)$$

$$t_n = \frac{1857{,}33}{23 \cdot 0{,}8} = 100{,}94, \text{ людино-годин}$$

Затрати праці на налагодження програми на ЕОМ:
– за умови автономного налагодження одного завдання:

$$t_{отл} = \frac{Q}{(4..5) \cdot k}; \quad (3.6)$$

$$t_{отл} = \frac{1857{,}33}{4 \cdot 0{,}8} = 580{,}42, \text{ людино-годин}$$

– за умови комплексного налагодження завдання:

$$t_{отл}^{к} = 1{,}4 \cdot t_{отл} \quad (3.7)$$

$$t_{отл}^{к} = 1{,}4 \cdot 580{,}42 = 812{,}59, \text{ людино-годин}$$

Затрати праці на підготовку документації:

$$t_д = t_{др} + t_{до} \quad (3.8)$$

де *tдр* – трудомісткість підготовки матеріалів і рукопису

$$t_{др} = \frac{Q}{(15..20)\cdot k} \quad (3.9)$$

$$t_{др} = \frac{1857,33}{20\cdot 0,8} = 116,08, \text{ людино-годин}$$

де *tдо* – трудомісткість редагування, печатки й оформлення документації

$$t_{до} = 0,75 \cdot t_{др} \quad (3.10)$$

$$t_{до} = 0,75 \cdot 116,08 = 87,06, \text{ людино-годин}$$

$$t_д = 116,08 + 87,06 = 203,14, \text{ людино-годим}$$

Отримаємо трудомісткість розробки програмного забезпечення:

$$t = 60 + 39,67 + 105,53 + 100,94 + 580,42 + 203,14 = 1089,70,$$

людино-годин.

У результаті ми розрахували, що в загальній складності необхідно близько 1089,70 людино-години, щоб розробити даний програмний продукт.

3.2. Розрахунок витрат на створення програми

Затрати на створення ПЗ *КПО* включають затрати на заробітну плату виконавця програми *ЗЗП* і витрат машинного часу, необхідного для налагодження програми на ЕОМ.

$$К_{ПО} = З_{ЗП} + З_{МВ} \quad (3.11)$$

де *ЗЗП* – заробітна плата виконавців, яка визначається за формулою:

$$З_{ЗП} = t \cdot C_{ПР}, \text{ грн}, \quad (3.12)$$

де *t* – загальна трудомісткість, людино-годин,

СПР – середня годинна заробітна плата програміста, грн/година.

$$З_{ЗП} = 1089,70 \cdot 126 = 137302,20, \text{ грн}.$$

ЗМВ – вартість машинного часу, необхідного для налагодження програми на ЕОМ:

$$З_{МВ} = t_{отл} \cdot C_{МЧ}, \text{ грн}, \quad (3.13)$$

де *тотл* – трудомісткість налагодження програми на ЕОМ, год.

СМЧ – вартість машино-години ЕОМ, грн/год.

$$З_{МВ} = 580{,}42 \cdot 4{,}3 = 2495{,}81, \text{ грн},$$

$$К_{ПО} = 137302{,}20 + 2495{,}81 = 139798{,}01, \text{ грн}.$$

Очікуваний період створення ПЗ:

$$Т = \frac{t}{B_k \cdot F_p}, \text{ мес}, \quad (3.14)$$

де *Bk* – число виконавців, *Fp*

– місячний фонд робочого часу (при 40 годинному робочому тижні Fp = 176 годин).

$$Т = \frac{1089.70}{1 * 176} \approx 6{,}19 \text{ міс}.$$

Висновки: для розробки цього програмного продукту потребується 1089,70 людино-годин. Таким чином, очікувана тривалість розробки складе 6,19 місяця при 40 годинному робочому тижні і 176-годинному робочому місяці, а затрати на створення при розробці програмного забезпечення складатимуть 139798,01 грн.

ВИСНОВКИ

Метою дипломного проекту є створення додатку, за допомогою якого клієнт зміг би отримувати більш персоналізовану інформацію за рахунками. Основною мовою для розробки використовувалась Java, по причині того що ця мова програмуваня є зручною та однією з популярніших на ринку технологій.

Програма може бути корисною для людей, яким не зручно підраховати та аналізувати свої витрати самостійно.

Для досягнення мети були вирішені наступні завдання:

- аналіз предметної галузі;
- уточнення вимог до програмного продукту;
- уточнення вимог до форматів додатку;
- розробка алгоритмів аналізу та обробки даних;

Відповідно до проведеного аналізу та поставленого завдання в роботі, вирішено всі функціональні задачі та дотримано вимоги до розробленого додатку, а саме:

- зручний і інформативний графічний інтерфейс;
- оновлення інформації у режимі онлайн;
- можливість отримувати інформацію з даними про рахунки через банківське API;
- можливість переглядати інформацію за власними рахунками;
- можливість входу до особистого кабінету;

Визначено приблизні значення трудомісткості розробленої інформаційної системи (1089,70 людино-годин), проведений підрахунок можливої вартості роботи по створенню програми (139798,01 грн) та розраховано час на його створення (6,19 міс).

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Інтернет-банкінг. [Електронний ресурс]. URL: https://uk.wikipedia.org/wiki/%D0%86%D0%BD%D1%82%D0%B5%D1%80%D0%BD%D0%B5%D1%82-%D0%B1%D0%B0%D0%BD%D0%BA%D1%96%D0%BD%D0%B3 (дата звернення 08.05.2022).

2. Common Application Properies. [Електронний ресурс].URL: https://docs.spring.io/spring-boot/docs/current/reference/html/application-properties.html(дата звернення 15.05.2022).

3. Використання фреймворку SpringBoot.[Електронний ресурс].URL: https://spring.io/projects/spring-boot(дата звернення 12.05.2022)

4. Merchant Category Code. [Електронний ресурс].URL: https://ru.wikipedia.org/wiki/Merchant_Category_Code(дата звернення 18.05.2022).

5. Робимо серіалізацію в JSON та назад з Jackson. [Електронний ресурс].URL: https://blog.ithillel.ua/ru/articles/delaem-serializatsiyu-v-json-i-obratno-c-jackson(дата звернення 15.05.2022).

6. Паттерни проєктування.[Електронний ресурс].URL: https://refactoring.guru/ru/design-patterns (дата звернення 14.05.2022).

7. Паттерн проєктування MVC.[Електронний ресурс]. URL: https://uk.wikipedia.org/wiki/%D0%9C%D0%BE%D0%B4%D0%B5%D0%BB%D1%8C-%D0%B2%D0%B8%D0%B4-%D0%BA%D0%BE%D0%BD%D1%82%D1%80%D0%BE%D0%BB%D0%B5%D1%80 (дата звернення 14.05.2022).

8. Додання бібліотек у проект Intellij IDEA. [Електронний ресурс].URL: https://javadevblog.com/kak-dobavitbiblioteku-jar-fajl-v-proekt-intellij-idea.html (дата звернення 11.05.2022).

9. JUnit Tutorial. [Електронний ресурс].URL: https://www.javatpoint.com/junit-tutorial(дата звернення 23.05.2022).

10. Налаштування CORS за допомогою SpringBoot.[Електронний ресурс].URL: https://www.baeldung.com/spring-cors (дата звернення 08.06.2022).

11. Інтерфейси в Java. [Електронний ресурс].URL: https://www.fandroid.info/interfejsy/(дата звернення 23.05.2022).

12. Введення в REST API. [Електронний ресурс].URL: https://habr.com/ru/post/483202/(дата звернення 10.05.2022).

13. Паттерн Repository у мові Java [Електронний ресурс]. URL: https://habr.com/ru/post/248505/ (дата звернення 20.05.2022).

14. Запити до MongoDB. [Електронний ресурс].URL: https://habr.com/ru/post/134590/(дата звернення 19.05.2022).

15. Common Application Properies. [Електронний ресурс].URL: https://docs.spring.io/spring-boot/docs/current/reference/html/application-properties.html(дата звернення 15.05.2022).

16. Класифікація валют. [Електронний ресурс].URL: https://uk.wikipedia.org/wiki/%D0%9A%D0%BB%D0%B0%D1%81%D0%B8%D1%84%D1%96%D0%BA%D0%B0%D1%86%D1%96%D1%8F_%D0%B2%D0%B0%D0%BB%D1%8E%D1%82_(ISO_4217)(дата звернення 22.05.2022).

17. Список кодів стану HTTP. [Електронний ресурс].URL: https://ru.wikipedia.org/wiki/%D0%A1%D0%BF%D0%B8%D1%81%D0%BE%D0%BA_%D0%BA%D0%BE%D0%B4%D0%BE%D0%B2_%D1%81%D0%BE%D1%81%D1%82%D0%BE%D1%8F%D0%BD%D0%B8%D1%8F_HTTP(дата звернення 12.05.2022).

18. Об’єкт Date в Java.[Електронний ресурс].URL: https://www.w3schools.com/java/java_date.asp(дата звернення 14.05.2022).

19. Заробітна плата Junior Java Developer в Украине [Електронний ресурс]. URL: https://dou.ua/lenta/articles/salary-report-devs-june-2020/ (дата звернення 13.06.2022).

20. Створення REST API за допомогою SpringBoot.[Електронний ресурс].URL: https://www.baeldung.com/rest-with-spring-series(дата звернення 12.05.2022).
21. Створення Git-репозиторія. [Електронний ресурс].URL: https://git-scm.com/docs/git (дата звернення 10.06.2022).
22. Unit тести в Java. [Електронний ресурс].URL: https://blog.ithillel.ua/ru/articles/unit-testy-v-java.-kratkoe-rukovodstvo(дата звернення 22.05.2022).
23. Merchant Category Code. [Електронний ресурс].URL: https://ru.wikipedia.org/wiki/Merchant_Category_Code(дата звернення 18.05.2022).
24. Unix-час. [Електронний ресурс].URL: https://ru.wikipedia.org/wiki/Unix-%D0%B2%D1%80%D0%B5%D0%BC%D1%8F(дата звернення 19.05.2022).
25. Maven Introduction. [Електронний ресурс].URL: https://maven.apache.org/what-is-maven.html(дата звернення 11.05.2022).
26. Звернення до стороннього API в Java. [Електронний ресурс].URL: https://habr.com/ru/company/umbrellaitcom/blog/423591/(дата звернення 16.05.2022).

ДОДАТОК А

КОД ПРОГРАМИ

КОД ПРОГРАМИ

Description.java

```
package com.server.financeassistantspring.Entity.Additional.MCC;

import com.fasterxml.jackson.annotation.JsonIgnoreProperties;
import com.fasterxml.jackson.annotation.JsonProperty;
import com.fasterxml.jackson.databind.ObjectMapper;

@JsonIgnoreProperties(ignoreUnknown = true)
public class Description {
   static ObjectMapper mapper = new ObjectMapper();
   @JsonProperty("uk")
   private String uk;
   @JsonProperty("en")
   private String en;
   @JsonProperty("ru")
   private String ru;

   public String getUk() {
      return uk;
   }

   public void setUk(String uk) {
      this.uk = uk;
   }

   public String getEn() {
      return en;
   }

   public void setEn(String en) {
      this.en = en;
   }

   public String getRu() {
      return ru;
   }

   public void setRu(String ru) {
      this.ru = ru;
   }
}
```

MCC.java

```
package com.server.financeassistantspring.Entity.Additional.MCC;


import com.fasterxml.jackson.annotation.JsonIgnoreProperties;
import com.fasterxml.jackson.annotation.JsonProperty;
import com.fasterxml.jackson.databind.JsonNode;
```

```
import com.fasterxml.jackson.databind.ObjectMapper;

import java.io.IOException;
import java.nio.file.Files;
import java.nio.file.Paths;

@JsonIgnoreProperties(ignoreUnknown = true)
public class MCC {

    static ObjectMapper mapper = new ObjectMapper();
    @JsonProperty("mcc")
    private String mcc;
    @JsonProperty("group")
    private MCCGroup group;

    public String getMcc() {
        return mcc;
    }

    public void setMcc(String mcc) {
        this.mcc = mcc;
    }

    public MCCGroup getGroup() {
        return group;
    }

    public void setGroup(MCCGroup group) {
        this.group = group;
    }

    static public MCC[] mccFill() throws IOException {
        String JsonStr = new String(Files.readAllBytes(Paths.get("src/main/resources/MCC.json")));
        JsonNode data = new ObjectMapper().readTree(JsonStr);
        MCC[] mccList = mapper.readValue(data.toString(),MCC[].class);
        return mccList;
    }

    static public MCC createPersonalGroup(String thisMcc, String thisType, String thisDescription,
String thisFullDescription, String thisShortDescription){
        MCC personalGroup = new MCC();
        personalGroup.setGroup(new MCCGroup());
        personalGroup.getGroup().setDescription(new Description());
        personalGroup.getGroup().setFullDescription(new Description());
        personalGroup.getGroup().setShortDescription(new Description());
        personalGroup.setMcc(thisMcc);
        personalGroup.getGroup().setType(thisType);
        personalGroup.getGroup().getDescription().setUk(thisDescription);
        personalGroup.getGroup().getFullDescription().setUk(thisFullDescription);
        personalGroup.getGroup().getShortDescription().setUk(thisShortDescription);
        return personalGroup;
    }
}
```

MCCGroup.java

```
package com.server.financeassistantspring.Entity.Additional.MCC;

import com.fasterxml.jackson.annotation.JsonIgnoreProperties;
```

```
import com.fasterxml.jackson.annotation.JsonProperty;
import com.fasterxml.jackson.databind.ObjectMapper;

@JsonIgnoreProperties(ignoreUnknown = true)
public class MCCGroup {

    static ObjectMapper mapper = new ObjectMapper();
    @JsonProperty("type")
    private String type;
    @JsonProperty("description")
    private Description description;
    @JsonProperty("fullDescription")
    private Description fullDescription;
    @JsonProperty("shortDescription")
    private Description shortDescription;

    public String getType() {
        return type;
    }

    public void setType(String type) {
        this.type = type;
    }

    public Description getDescription() {
        return description;
    }

    public void setDescription(Description description) {
        this.description = description;
    }

    public Description getFullDescription() {
        return fullDescription;
    }

    public void setFullDescription(Description fullDescription) {
        this.fullDescription = fullDescription;
    }

    public Description getShortDescription() {
        return shortDescription;
    }

    public void setShortDescription(Description shortDescription) {
        this.shortDescription = shortDescription;
    }
}
```

Currency.java

```
package com.server.financeassistantspring.Entity.Additional;

import com.fasterxml.jackson.annotation.JsonProperty;
import com.server.financeassistantspring.Interfases.IAPIUser;

import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStreamReader;
```

```
import java.net.HttpURLConnection;
import java.net.URL;
import java.nio.file.Files;
import java.nio.file.Paths;
import java.util.Arrays;
import java.util.List;

public class Currency implements IAPIUser {
    @JsonProperty("currencyCodeA")
    private int currencyCodeA;

    @JsonProperty("currencyCodeB")
    private int currencyCodeB;

    @JsonProperty("date")
    private long date;

    @JsonProperty("rateSell")
    private double rateSell;

    @JsonProperty("rateBuy")
    private double rateBuy;

    @JsonProperty("rateCross")
    private float rateCross;

    public int getCurrencyCodeA() {
        return currencyCodeA;
    }

    public void setCurrencyCodeA(int currencyCodeA) {
        this.currencyCodeA = currencyCodeA;
    }

    public int getCurrencyCodeB() {
        return currencyCodeB;
    }

    public void setCurrencyCodeB(int currencyCodeB) {
        this.currencyCodeB = currencyCodeB;
    }

    public long getDTM() {
        return date;
    }

    public void setDTM(long DTM) {
        this.date = DTM;
    }

    public double getRateSell() {
        return rateSell;
    }

    public void setRateSell(double rateSell) {
        this.rateSell = rateSell;
    }
```

```
    public double getRateBuy() {
        return rateBuy;
    }

    public void setRateBuy(double rateBuy) {
        this.rateBuy = rateBuy;
    }

    public float getRateCross() {
        return rateCross;
    }

    public void setRateCross(float rateCross) {
        this.rateCross = rateCross;
    }

    @Override
    public String toString() {
        return
getCurrencyCodeA()+","+getCurrencyCodeB()+","+getRateBuy()+","+getRateSell()+","+getDTM()
;
    }

    @Override
    public String apiCall() throws IOException {
        String JsonStr = null;
        final URL url = new URL(BASE_URL + "bank/currency");
        final HttpURLConnection con = (HttpURLConnection) url.openConnection();
        con.setRequestMethod("GET");
        con.setRequestProperty("Content-Type", "application/json");
        con.setConnectTimeout(50000);
        con.setReadTimeout(7000000);

        try (final BufferedReader in = new BufferedReader(new
InputStreamReader(con.getInputStream()))) {
            String inputLine;
            final StringBuilder content = new StringBuilder();
            while ((inputLine = in.readLine()) != null) {
                content.append(inputLine);
            }
            JsonStr = content.toString();
        }
        catch (final Exception ex) {
            //ex.printStackTrace();
            JsonStr = new
String(Files.readAllBytes(Paths.get("src/main/resources/CURRENCY.json")));
        }
        return JsonStr;
    }
    static public List<Currency> currencyFill() throws IOException {
        Currency caller = new Currency();
        Currency[] currList = Currency.mapper.readValue(caller.apiCall(),Currency[].class);
        return Arrays.asList(currList);
    }
}
```

PersonalSettings.java

```
package com.server.financeassistantspring.Entity.Additional;

import com.server.financeassistantspring.Entity.Additional.MCC.MCC;
import org.springframework.data.mongodb.core.mapping.Document;
import org.springframework.data.mongodb.core.mapping.Field;

import java.util.List;

@Document(collection = "PersonalSettings")
public class PersonalSettings {
    @Field(value = "clientId")
    private String clientId;
    @Field(value = "personalSettings")
    private List<MCC> personalSettings;

    public PersonalSettings(String clientId, List<MCC> personalSettings) {
        this.clientId = clientId;
        this.personalSettings = personalSettings;
    }

    public String getClientId() {
        return clientId;
    }

    public void setClientId(String clientId) {
        this.clientId = clientId;
    }

    public List<MCC> getPersonalSettings() {
        return personalSettings;
    }

    public void setPersonalSettings(List<MCC> personalSettings) {
        this.personalSettings = personalSettings;
    }
}
```

AccountParams.java

```
package com.server.financeassistantspring.Entity.Main;

import com.fasterxml.jackson.annotation.JsonIgnoreProperties;

//TODO: Fix pan card fill from API
@JsonIgnoreProperties( ignoreUnknown = true )
public class AccountParams {
    //@JsonProperty("id")
    private String id;
    //@JsonProperty("balance")
    private int balance;
    //@JsonProperty("creditLimit")
    private int creditLimit;
    //@JsonProperty("type")
    private String type;
    //@JsonProperty("cashbackType")
    private String cashbackType;
    //@JsonProperty("currencyCode")
    private int currencyCode;
    //@JsonProperty("sendId")
```

```
private String sendID;
//@JsonProperty("maskedPan")
private String panCard;
//@JsonProperty("iban")
private String iban;

public int getCurrencyCode() {
   return currencyCode;
}

public void setCurrencyCode(int currencyCode) {
   this.currencyCode = currencyCode;
}

public String getCashbackType() {
   return cashbackType;
}

public void setCashbackType(String cashbackType) {
   this.cashbackType = cashbackType;
}

public String getType() {
   return type;
}

public void setType(String type) {
   this.type = type;
}

public double getCreditLimit() {
   double result = 0;
   StringBuffer temp = new StringBuffer(String.valueOf(this.creditLimit));
   if (temp.length()>2) {
      temp.insert(temp.length()-2,".");
      result = Double.parseDouble(temp.toString());
   }
   else {
      if(temp.length() == 2 & temp.charAt(0) == '-'){
         temp.insert(1,"0.0");
         result = Double.parseDouble(temp.toString());
      }
      else if (temp.length() == 2 & temp.charAt(0) != '-'){
         temp.insert(0,"0.");
         result = Double.parseDouble(temp.toString());
      }

      else {
         temp.insert(0,"0.0");
         result = Double.parseDouble(temp.toString());
      }
   }
   return result;
}

public void setCreditLimit(int creditLimit) {
   this.creditLimit = creditLimit;
```

```
}

public String getId() {
   return id;
}

public void setId(String id) {
   this.id = id;
}

public double getBalance() {
   double result = 0;
   StringBuffer temp = new StringBuffer(String.valueOf(this.balance));
   if (temp.length()>2) {
      temp.insert(temp.length()-2,".");
      result = Double.parseDouble(temp.toString());
   }
   else {
      if(temp.length() == 2 & temp.charAt(0) == '-'){
         temp.insert(1,"0.0");
         result = Double.parseDouble(temp.toString());
      }
      else if (temp.length() == 2 & temp.charAt(0) != '-'){
         temp.insert(0,"0.");
         result = Double.parseDouble(temp.toString());
      }

      else {
         temp.insert(0,"0.0");
         result = Double.parseDouble(temp.toString());
      }
   }
   return result;
}

public void setBalance(int balance) {
   this.balance = balance;
}

public String getSendID() {
   return sendID;
}

public void setSendID(String sendID) {
   this.sendID = sendID;
}

public String getIban() {
   return iban;
}

public void setIban(String iban) {
   this.iban = iban;
}

public String getPanCard() {
   return panCard;
```

```
    }

    public void setPanCard(String panCard) {
        this.panCard = panCard;
    }
}
```

Analyser.java

```
package com.server.financeassistantspring.Entity.Main;


import com.server.financeassistantspring.Entity.Additional.MCC.MCC;
import com.server.financeassistantspring.Entity.Additional.UnixTimeParser;

import java.io.IOException;
import java.text.SimpleDateFormat;
import java.util.*;
import java.util.concurrent.TimeUnit;
public class Analyser{

    public double calculateWriteOff(Transaction[] extract){
        double result = 0;
        try{
            for (Transaction transction: extract) {
                if (transction.getAmount() < 0){
                    result += transction.getAmount();
                }
            }
            return result;
        }
        catch (ArrayIndexOutOfBoundsException exception){
            return result;
        }
    }

    public double calculateRefills(Transaction[] extract){
        double result = 0;
        try {
            for (Transaction transction: extract) {
                if (transction.getAmount() > 0){
                    result += transction.getAmount();
                }
            }
            return result;
        }
        catch (ArrayIndexOutOfBoundsException exception){
            return result;
        }
    }

    public Transaction[] amountForPeriod(Transaction[] extract, long from, long to){
        if (extract[0] != null) {
            ArrayList<Transaction> result = new ArrayList<>();
            for (Transaction transaction: extract) {
                if(transaction.getTime() <= to & transaction.getTime() >= from){
                    result.add(transaction);
                }
            }
```

```
            return result.toArray(new Transaction[0]);
        }
        else return null;
    }

    public List<Transaction> sortByMCC(Transaction[] extract, List<MCC> mccList, String group){
        ArrayList<Transaction> result = new ArrayList<>();
        if (extract[0] != null){
            for (Transaction transaction: extract) {
                for (MCC mcc: mccList) {
                    if (Objects.equals(mcc.getGroup().getType(), group) & Objects.equals(mcc.getMcc(),
String.valueOf(transaction.getMcc()))){
                        result.add(transaction);
                    }
                }

            }
            return result;
        }
        else return null;
    }

    public boolean isWriteoffBigger(Transaction[] extract1, Transaction[] extract2){
        if (extract1 != null & extract2 != null){
            if (calculateWriteOff(extract1) < calculateWriteOff(extract2)){
                return true;
            }
            return false;
        }
        else return false;
    }

    public boolean isWriteoffEquals(Transaction[] extract1, Transaction[] extract2){
        if (extract1 != null & extract2 != null){
            if (calculateWriteOff(extract1) == calculateWriteOff(extract2)){
                return true;
            }
            else return false;
        }
        else return false;
    }

    public boolean isRefillsBigger(Transaction[] extract1, Transaction[] extract2){
        if (extract1 != null & extract2 != null){
            if (calculateRefills(extract1) > calculateRefills(extract2)){
                return true;
            }
            return false;
        }
        else return false;
    }
    public boolean isRefillsEquals(Transaction[] extract1, Transaction[] extract2){
        if (extract1 != null & extract2 != null){
            if (calculateRefills(extract1) == calculateRefills(extract2)){
                return true;
            }
            else return false;
```

```
        }
        else return false;
    }

    public double prognosis(User client) throws IOException, InterruptedException {
        double result = 0;
        double ratio = 0;
        double writeoffRatio;
        double reffilsRatio;
        Transaction[] temp1;
        Transaction[] temp2;
        Transaction[] temp3;
        Date date = new Date ();
        SimpleDateFormat sdf = new SimpleDateFormat("dd/MM/yyyy HH:mm:ss");
        Transaction caller = new Transaction();
        for (int i = 0; i < client.getAccount().size(); i++){
            temp1 = Transaction.mapper.readValue(caller.apiCall(client.getAccount().get(i).getId(),
(UnixTimeParser.timeParse(sdf.format(date.getTime())) - (4*UnixTimeParser.oneWeek)),
UnixTimeParser.timeParse(sdf.format(date.getTime())),client.getPersonalToken()),
Transaction[].class);
            TimeUnit.SECONDS.sleep(70);
            temp2 = Transaction.mapper.readValue(caller.apiCall(client.getAccount().get(i).getId(),
(UnixTimeParser.timeParse(sdf.format(date.getTime())) - (8*UnixTimeParser.oneWeek)),
(UnixTimeParser.timeParse(sdf.format(date.getTime())) -
(4*UnixTimeParser.oneWeek)),client.getPersonalToken()), Transaction[].class);
            TimeUnit.SECONDS.sleep(70);
            temp3 = Transaction.mapper.readValue(caller.apiCall(client.getAccount().get(i).getId(),
(UnixTimeParser.timeParse(sdf.format(date.getTime())) - (12*UnixTimeParser.oneWeek)),
(UnixTimeParser.timeParse(sdf.format(date.getTime())) -
(8*UnixTimeParser.oneWeek)),client.getPersonalToken()), Transaction[].class);
            writeoffRatio = ((calculateWriteOff(temp2)/calculateWriteOff(temp1)) +
(calculateWriteOff(temp3)/calculateWriteOff(temp2))/2);
            reffilsRatio = ((calculateRefills(temp2)/calculateRefills(temp1)) +
(calculateRefills(temp3)/calculateRefills(temp2))/2);
            ratio += (reffilsRatio - writeoffRatio);
        }
        if (ratio > 0){
            for (int i = 0; i < client.getAccount().size(); i++){
                TimeUnit.SECONDS.sleep(70);
                temp1 = Transaction.mapper.readValue(caller.apiCall(client.getAccount().get(i).getId(),
(UnixTimeParser.timeParse(sdf.format(date.getTime())) - (4*UnixTimeParser.oneWeek)),
UnixTimeParser.timeParse(sdf.format(date.getTime())),client.getPersonalToken()),
Transaction[].class);
                result += (calculateRefills(temp1)*ratio - calculateWriteOff(temp1));
            }
            return result;
        }
        else if(ratio < 0){
            for (int i = 0; i < client.getAccount().size(); i++){
                TimeUnit.SECONDS.sleep(70);
                temp1 = Transaction.mapper.readValue(caller.apiCall(client.getAccount().get(i).getId(),
(UnixTimeParser.timeParse(sdf.format(date.getTime())) - (4*UnixTimeParser.oneWeek)),
UnixTimeParser.timeParse(sdf.format(date.getTime())),client.getPersonalToken()),
Transaction[].class);
                result += (calculateRefills(temp1) - (-1*(calculateWriteOff(temp1)*ratio)));
            }
            return result;
```

```
            }
            else return result;
        }
        public List<Transaction> sortByMcc(List<Transaction> extract, List<MCC> mccList){
            return null;
        }
}
```

Transaction.java

```
package com.server.financeassistantspring.Entity.Main;


import com.fasterxml.jackson.annotation.JsonIgnoreProperties;

import com.server.financeassistantspring.Interfases.IAPIUser;


import java.io.BufferedReader;

import java.io.IOException;

import java.io.InputStreamReader;

import java.net.HttpURLConnection;

import java.net.URL;


import static com.server.financeassistantspring.Interfases.IAPIUser.BASE_URL;


@JsonIgnoreProperties( ignoreUnknown = true )

public class Transaction implements IAPIUser {

    private String id;

    private int time;

    private String description;

    private int mcc;

    private boolean hold;

    private int amount;

    private int operationAmount;

    private int currencyCode;

    private int commissionRate;

    private int cashbackAmount;
```

```
    private int balance;
    private String comment;
    private String receiptId;
    private String counterEdrpou;
    private String counterIban;

    //====================================

    @Override
    public String apiCall() throws IOException {
        System.out.println("Неверный набор параметров запуска");
        return null;
    }

    public String apiCall(String account, long from, String token) throws IOException {
        String JsonStr = null;
        final URL url = new URL(BASE_URL + "personal/statement/"+account+"/"+from);
        final HttpURLConnection con = (HttpURLConnection) url.openConnection();
        con.setRequestMethod("GET");
        con.setRequestProperty("Content-Type", "application/json");
        con.setRequestProperty("X-Token", token);
        con.setConnectTimeout(50000);
        con.setReadTimeout(7000000);

        try (final BufferedReader in = new BufferedReader(new InputStreamReader(con.getInputStream())))
{
            String inputLine;
            final StringBuilder content = new StringBuilder();
            while ((inputLine = in.readLine()) != null) {
                content.append(inputLine);
```

```
            }
            JsonStr = content.toString();
        }
        catch (final Exception ex) {
            ex.printStackTrace();
            System.out.println("ex");
        }
        return JsonStr;
    }

    public String apiCall(String account, long from, long to, String token) throws IOException {
        String JsonStr = null;
        final URL url = new URL(BASE_URL + "personal/statement/" + account + "/" + from + "/" + to);
        final HttpURLConnection con = (HttpURLConnection) url.openConnection();
        con.setRequestMethod("GET");
        con.setRequestProperty("Content-Type", "application/json");
        con.setRequestProperty("X-Token", token);
        con.setConnectTimeout(50000);
        con.setReadTimeout(7000000);

        try (final BufferedReader in = new BufferedReader(new InputStreamReader(con.getInputStream())))
{
            String inputLine;
            final StringBuilder content = new StringBuilder();
            while ((inputLine = in.readLine()) != null) {
                content.append(inputLine);
            }
            JsonStr = content.toString();
        }
        catch (final Exception ex) {
```

```
        ex.printStackTrace();
        System.out.println("ex");
    }
    return JsonStr;
}

//=====================================

public String getId() {
    return id;
}

public void setId(String id) {
    this.id = id;
}

public int getTime() {
    return time;
}

public void setTime(int time) {
    this.time = time;
}

public String getDescription() {
    return description;
}

public void setDescription(String description) {
    this.description = description;
```

```
}

public int getMcc() {
    return mcc;
}

public void setMcc(int mcc) {
    this.mcc = mcc;
}

public boolean isHold() {
    return hold;
}

public void setHold(boolean hold) {
    this.hold = hold;
}

public double getAmount() {
    double result = 0;
    StringBuffer temp = new StringBuffer(String.valueOf(this.amount));
    if (temp.length()>2) {
        temp.insert(temp.length()-2,".");
        result = Double.parseDouble(temp.toString());
    }
    else {
        if(temp.length() == 2 & temp.charAt(0) == '-'){
            temp.insert(1,"0.0");
            result = Double.parseDouble(temp.toString());
        }
```

```
        else if (temp.length() == 2 & temp.charAt(0) != '-'){
            temp.insert(0,"0.");
            result = Double.parseDouble(temp.toString());
        }

        else {
            temp.insert(0,"0.0");
            result = Double.parseDouble(temp.toString());
        }
    }
    return result;
}

public void setAmount(int amount) {
    this.amount = amount;
}

public int getOperationAmount() {
    return operationAmount;
}

public void setOperationAmount(int operationAmount) {
    this.operationAmount = operationAmount;
}

public int getCurrencyCode() {
    return currencyCode;
}

public void setCurrencyCode(int currencyCode) {
```

```
    this.currencyCode = currencyCode;
}

public int getCommissionRate() {
    return commissionRate;
}

public void setCommissionRate(int commissionRate) {
    this.commissionRate = commissionRate;
}

public int getCashbackAmount() {
    return cashbackAmount;
}

public void setCashbackAmount(int cashbackAmount) {
    this.cashbackAmount = cashbackAmount;
}

public int getBalance() {
    return balance;
}

public void setBalance(int balance) {
    this.balance = balance;
}

public String getComment() {
    return comment;
}
```

```
    public void setComment(String comment) {
        this.comment = comment;
    }

    public String getReceiptId() {
        return receiptId;
    }

    public void setReceiptId(String receiptId) {
        this.receiptId = receiptId;
    }

    public String getCounterEdrpou() {
        return counterEdrpou;
    }

    public void setCounterEdrpou(String counterEdrpou) {
        this.counterEdrpou = counterEdrpou;
    }

    public String getCounterIban() {
        return counterIban;
    }

    public void setCounterIban(String counterIban) {
        this.counterIban = counterIban;
    }
}
```

User.java

```
package com.server.financeassistantspring.Entity.Main;

import com.fasterxml.jackson.annotation.JsonIgnoreProperties;
import com.fasterxml.jackson.annotation.JsonProperty;
import com.server.financeassistantspring.Entity.Additional.Currency;
import com.server.financeassistantspring.Entity.Additional.MCC.MCC;
import com.server.financeassistantspring.Entity.Additional.PersonalSettings;
import com.server.financeassistantspring.Interfases.IAPIUser;
import org.springframework.data.mongodb.core.mapping.Document;
import org.springframework.data.mongodb.core.mapping.Field;

import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStreamReader;
import java.net.HttpURLConnection;
import java.net.URL;
import java.util.ArrayList;
import java.util.List;
import java.util.Objects;

@Document(collection = "User")
@JsonIgnoreProperties( ignoreUnknown = true )
public class User implements IAPIUser {
    private String personalToken;

    @Field("clientId")
    @JsonProperty("clientId")
    private String clientId;

    @Field("name")
```

```
@JsonProperty("name")
private String name;

@JsonProperty("webHookUrl")
private String webHookUrl;

@JsonProperty("accounts")
private List<AccountParams> account;

@JsonProperty("permissions")
private String permission;

private MCC[] mccListByGroup;

@Field("personalMcc")
private PersonalSettings personalSettings;
private Currency[] currList;

//OTHER METHOD
@Override
public String apiCall() throws IOException {
    String JsonStr = null;

    final URL url = new URL(BASE_URL + "personal/client-info");
    final HttpURLConnection con = (HttpURLConnection) url.openConnection();
    con.setRequestMethod("GET");
    con.setRequestProperty("Content-Type", "application/json");
    con.setRequestProperty("X-Token", this.personalToken);
    con.setConnectTimeout(50000);
    con.setReadTimeout(7000000);
```

```
        try (final BufferedReader in = new BufferedReader(new InputStreamReader(con.getInputStream())))
{
            String inputLine;
            final StringBuilder content = new StringBuilder();
            while ((inputLine = in.readLine()) != null) {
                content.append(inputLine);
            }
            JsonStr = content.toString();
        }
        return JsonStr;
    }

    public double getTotalBalance(List<Currency> currencies){
        double balance = 0;
        for (int i = 0; i < this.account.size(); i++) {
            if (this.account.get(i).getCurrencyCode() == 980) {
                balance += this.account.get(i).getBalance();
            } else {
                for (Currency currency : currencies) {
                    if (currency.getCurrencyCodeA() == this.account.get(i).getCurrencyCode()) {
                        balance += this.account.get(i).getBalance() / currency.getRateSell();
                    }
                }
            }
        }
        return balance;
    }
    // GETTER AND SETTER
```

```
public List<AccountParams> getAccount() {
    return account;
}

public void setAccount(List<AccountParams> account) {
    this.account = account;
}

public String getWebHookUrl() {
    return webHookUrl;
}

public void setWebHookUrl(String webHookUrl) {
    this.webHookUrl = webHookUrl;
}

public String getName() {
    return name;
}

public void setName(String name) {
    this.name = name;
}

public String getId() {
    return clientId;
}

public void setId(String id) {
    this.clientId = id;
```

```
}

public String getPermission() {
    return permission;
}

public void setPermission(String permission) {
    this.permission = permission;
}

public String getPersonalToken() {
    return personalToken;
}

public void setPersonalToken(String personalToken) {
    this.personalToken = personalToken;
}

public MCC[] getMccListByGroup() {
    return mccListByGroup;
}

public void setMccListByGroup(MCC[] mccListByGroup) throws IOException {
    this.mccListByGroup = MCC.mccFill();
}
public Currency[] getCurrList() {
    return currList;
}

public void setCurrList(Currency[] currList) {
```

```
        this.currList = currList;
    }

    public PersonalSettings getPersonalSettings() {
        return personalSettings;
    }

    public void setPersonalSettings(PersonalSettings personalSettings) {
        this.personalSettings = personalSettings;
    }
}
...
```

ДОДАТОК Б

ВІДГУК КЕРІВНИКА ЕКОНОМІЧНОГО РОЗДІЛУ

ДОДАТОК В

ПЕРЕЛІК ФАЙЛІВ НА ДИСКУ

Ім’я файлу	Опис
Появнювальні документи	
	Пояснювальна записка кваліфікаційної роботи. Документ Word.
	Пояснювальна записка кваліфікаційної роботи в форматі PDF.
Програма	
	Архів. Містить коди програми.
Презентація	
	Презентація кваліфікаційної роботи.

ДОДАТОК Г

ВІДГУК
на кваліфікаційну роботу бакалавра
на тему:
"Розробка програмного забезпечення для фінансового контроля і керування банківським рахунком засобами Java, SpringBoot, FasterXML"
студентки групи 121-18-1 Швець Владислав Ігорович

ТЕКСТ

Керівник кваліфікаційної роботи
професор каф. ПЗКС, д.т.н. **М.Г. Бердник**

ДОДАТОК Д

РЕЦЕНЗІЯ
на кваліфікаційну роботу бакалавра
на тему:
«Розробка програмного забезпечення для організації роботи автосалону»
студента групи 121-18-1 Іванова Олександра Руслановича

ТЕКСТ

Рецензент дипломного проекту
доц., кафедри ПКЗС к.т.н. **І. А. Шедловський**