НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ імені ІГОРЯ СІКОРСЬКОГО»
Факультет інформатики та обчислювальної техніки
(повна назва інституту/факультету)

Кафедра інформатики та програмної інженерії
(повна назва кафедри)

«До захисту допущено»
Завідувач кафедри
_____ Едуард ЖАРІКОВ
(підпис) (ім'я прізвище)
"___"_____2024 р.

Дипломний проєкт

на здобуття ступеня бакалавра

за освітньо-професійною програмою «Інженерія програмного забезпечення інформаційних систем»

спеціальності «121 Інженерія програмного забезпечення»

на тему: Веб-застосунок для управління особистими фінансами

Виконав студент IV курсу, групи ІТ-з01
(шифр групи)
Березовський Владислав Андрійович _____
(прізвище, ім'я, по батькові) (підпис)

Керівник асистент, Соколовський В. В. _____
(посада, науковий ступінь, вчене звання, прізвище та ініціали) (підпис)

Рецензент доцент кафедри ОТ, к.т.н., доц. Волокита А.М. _____
(посада, науковий ступінь, вчене звання, прізвище та ініціали) (підпис)

Засвідчую, що у цій дипломній роботі немає запозичень з праць інших авторів без відповідних посилань.
Студент _____
(підпис)

Київ –2024

Національний технічний університет України
«Київський політехнічний інститут імені Ігоря Сікорського»

Факультет інформатики та обчислювальної техніки

Кафедра інформатики та програмної інженерії

Рівень вищої освіти – перший (бакалаврський)

Спеціальність – 121 Інженерія програмного забезпечення

Освітньо-професійна програма – Інженерія програмного забезпечення інформаційних систем

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ Едуард ЖАРІКОВ
(підпис) (ім'я прізвище)
"___"_____2024 р.

ЗАВДАННЯ
на дипломний проєкт студенту

Березовський Владислав Андрійович
(прізвище, ім'я, по батькові)

1. Тема проєкту Веб-застосунок для управління особистими фінансами
керівник проєкту Соколовський Владислав Володимир, асистент
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджені наказом по університету від «27»квітня 2024 р. №2113-с

2. Термін подання студентом проєкту « 17 » червня 2024 року

3. Вихідні дані до проєкту: технічне завдання

4. Зміст пояснювальної записки
1) Аналіз вимог до програмного забезпечення: загальні положення, опис і аналіз предметної області, аналіз відомих технічних рішень, аналіз засобів розробки, аналіз відомих програмних продуктів, розробка функціональних та нефункціональних вимог, постановка задачі.
2) Моделювання та конструювання програмного забезпечення: моделювання та аналіз програмного забезпечення, архітектура програмного забезпечення, конструювання програмного забезпечення, аналіз безпеки конфіденційних даних.
3) Аналіз якості та тестування програмного забезпечення: аналіз якості програмного забезпечення, опис тестувальних процесів.
4) Розгортання та супровід програмного забезпечення: розгортання програмного забезпечення, супровід програмного забезпечення.
5) Технологічний розділ: керівництво користувача, методика випробувань програмного продукту.

5. Перелік графічного матеріалу

1) Схема структурна варіантів використань

2) Схема бази даних

3) Схема вигляду екранних форм

4) BPMN діаграми

6. Консультанти розділів проєкту

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв

7. Дата видачі завдання «11» березня 2024 року

Календарний план

№ з/п	Назва етапів виконання дипломного проєкту	Термін виконання етапів проєкту	Примітка
1	Вивчення рекомендованої літератури	11.03.2024	
2	Розробка технічного завдання	18.03.2024	
3	Аналіз вимог та уточнення	20.03.2024	
4	специфікацій	30.03.2024	
5	Проектування структури	05.04.2024	
6	програмного забезпечення,	06.04.2024	
7	проектування компонентів	30.04.2024	
8	Програмна реалізація	8.04.2024	
9	програмного забезпечення	12.04.2024	
10	Тестування програмного	16.04.2024	
11	Подання ДП на попередній захист	23.05.2024	
12	Подання ДП рецензенту	12.06.2024	
13	Подання ДП на основний захист	14.06.2024	

Студент _____ Владислав БЕРЕЗОВСЬКИЙ
(підпис) (ініціали, прізвище)

Керівник _____ Владислав СОКОЛОВСЬКИЙ
(підпис) (ініціали, прізвище)

АНОТАЦІЯ

Пояснювальна записка дипломного проєкту складається з чотирьох розділів, містить 34 таблиць, 21 рсунків та 30 джерел – загалом 78 сторінки.

Дипломний проєкт присвячений розробленню програмної системи яка дозволяє зручно відслідковувати свої фінансові надходження та збитки.

Метою є підвищення зручності управління та обліку особистими фінансами та розвиток фінансового контролю.

Об'єкт дослідження: система контролю особистих фінансів.

Предмет дослідження: програмне забезпечення для систем фінансового обліку.

У розділі аналізу вимог до програмного забезпечення розглянуто відповідну літературу, проаналізовано наявні технології та успішні IT-проєкти, а також вимоги до програмного забезпечення.

Розділ моделювання та проектування програмного забезпечення присвячений побудові архітектури застосунка, опису деталей його імплементації, а також визначенню системних вимог й аналізу безпеки додатку.

У розділі аналізу якості та тестуванню наводиться огляд результатів аналізу якості системи, опис методів і ходу тестування, контрольний приклад взаємодії з системою.

У розділі, присвяченому впровадженню та супроводу, описані відповідні алгоритми розгортання додатку, а також рекомендації щодо його підтримки й супроводу.

Програмне забезпечення було успішно розгорнуто на сервісі Microsoft Azure і тепер кожен бажаючий зможе отримати доступ до ресурсу.

КЛЮЧОВІ СЛОВА: ВЕБ-ЗАСТОСУНОК, БАЗА ДАНИХ, ТРАНЗАКЦІЯ, ОБЛІК, ФІНАНСИ, JAVA, TDD, 2FA, Spring, Hibernate, PostgreSQL, JSON, ACID, SSL, React, DOM, AWS, GDPR, HIPPA, PCI DSS, Kubernetes, Jenkins, Docker.

ABSTRACT

The explanatory note of the diploma project consists of four sections, contains 34 tables, 21 figures and 30 sources - a total of 78 pages.

The diploma project is devoted to the development of a software system that allows you to conveniently track your financial income and losses.

The goal is to improve the convenience of managing and accounting for personal finances and the development of financial control.

Object of research: the system of personal finance control.

Research subject: software for financial accounting systems.

The software requirements analysis section reviews relevant literature, analyzes available technologies and successful IT projects, as well as software requirements.

The software modeling and design section is dedicated to building the application architecture, describing the details of its implementation, as well as defining system requirements and application security analysis.

The quality analysis and testing section provides an overview of the results system quality analysis, description of testing methods and progress, control example interactions with the system.

The deployment and maintenance section describes the appropriate deployment algorithms for the application, as well as recommendations for its support and maintenance.

The software was successfully deployed on the Microsoft Azure service and now anyone can access the resource.

KEYWORDS: WEB APPLICATION, DATABASE, TRANSACTION, ACCOUNTING, FINANCE, JAVA, TDD, 2FA, Spring, Hibernate, PostgreSQL, JSON, ACID, SSL, React, DOM, AWS, GDPR, HIPPA, PCI DSS, Kubernetes, Jenkins , Docker.

№ з/п	Формат	Позначення	Найменування	Кількість листів	Примітка
1	А4		Завдання на дипломний проєкт	2	
2	А4	КПІ.ІТ.045440.01.91	Технічне завдання	21	
3	А4	КПІ.ІТ.045440.02.81	Пояснювальна записка	78	
4	А4	КПІ.ІТ.045440.03.12	Текст програми	262	
5	А4	КПІ.ІТ.045440.04.51	Програма та методика тестування	7	
6	А4	КПІ.ІТ.045440.05.34	Керівництво користувача	11	
7	А3	КПІ.ІТ.045440.06.99	Графічний матеріал	9	

Змін.	Арк.	№ докум.	Підп.	Дата	КПІ.ІТ.045440.00.90		
Розроб.		Березовський В.А.			Відомість дипломного проєкту	Літ.	Аркуш 1 / Аркушів
Перевір.							
Н.контр.		Стельмах О.П.				КПІ ім. Ігоря Сікорського ФІОТ каф. ІПІ гр. ІТ-з01	
Затв.		Жаріков Е.В.					

Факультет інформатики та обчислювальної техніки

Кафедра інформатики та програмної інженерії

“ЗАТВЕРДЖЕНО”

Завідувач кафедри

_____ Едуард ЖАРІКОВ

“___” _____ 2024 р.

ВЕБ-ЗАСТОСУНОК ДЛЯ УПРАВЛІННЯ ОСОБИСТИМИ ФІНАНСАМИ

Технічне завдання

КПІ.ІТ.045440.01.91

“ПОГОДЖЕНО”

Керівник проєкту:

_____Владислав СОКОЛОВСЬКИЙ

Нормоконтроль:

_____Олександр СТЕЛЬМАХ

Виконавець:

_____Владислав БЕРЕЗОВСЬКИЙ

Київ – 2024

ЗМІСТ

1 НАЙМЕНУВАННЯ ТА ГАЛУЗЬ ЗАСТОСУВАННЯ 3
2 ПІДСТАВА ДЛЯ РОЗРОБКИ 4
3 ПРИЗНАЧЕННЯ РОЗРОБКИ 5
4 ВИМОГИ ДО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ 6
4.1 Вимоги до функціональних характеристик 6
4.1.1 Аутентифікація 6
4.1.2 Користувацького інтерфейсу 8
4.1.3 Для користувача: 14
4.1.4 Додаткові вимоги: 15
4.2 Вимоги до надійності 15
4.3 Умови експлуатації 16
4.3.1 Вид обслуговування 16
4.3.2 Обслуговуючий персонал 17
4.4 Вимоги до складу і параметрів технічних засобів 17
4.5 Вимоги до інформаційної та програмної сумісності 17
4.5.1 Вимоги до вхідних даних 17
4.5.2 Вимоги до вихідних даних 17
4.5.3 Вимоги до мови розробки 17
4.5.4 Вимоги до середовища розробки 17
4.5.5 Вимоги до представленню вихідних кодів 18
4.6 Вимоги до маркування та пакування 18
4.7 Вимоги до транспортування та зберігання 18
4.8 Спеціальні вимоги 18
5 ВИМОГИ ДО ПРОГРАМНОЇ ДОКУМЕНТАЦІЇ 19
5.1 Попередній склад програмної документації 19
5.2 Спеціальні вимоги до програмної документації 19
6 СТАДІЇ І ЕТАПИ РОЗРОБКИ 20
7 ПОРЯДОК КОНТРОЛЮ ТА ПРИЙМАННЯ 21

1 НАЙМЕНУВАННЯ ТА ГАЛУЗЬ ЗАСТОСУВАННЯ

Назва розробки: Веб-застосунок для управління особистими фінансами

Галузь застосування:

Наведене технічне завдання поширюється на розробку веб-застосунку для управління особистими фінансами «Finio», котрий використовується для обліку доходів і витрат, планування бюджету, створення фінансових звітів та аналізу фінансових показників. Він призначений для використання широким колом користувачів, включаючи індивідуальних користувачів, які бажають керувати своїми фінансами, а також малий бізнес та підприємців, які потребують ефективних інструментів для фінансового планування та контролю.

2 ПІДСТАВА ДЛЯ РОЗРОБКИ

Підставою для розробки веб-застосунку для управління особистими фінансами є завдання на дипломне проєктування, затверджене кафедрою інформатики та програмної інженерії Національного технічного університету України «Київський політехнічний інститут імені Ігоря Сікорського».

3 ПРИЗНАЧЕННЯ РОЗРОБКИ

Метою дипломного проєкту є розробка та впровадження комплексної системи для автоматизації фінансового обліку та управління даними. Цільовий застосунок призначений для фізичних осіб, які мають на меті оптимізувати процеси управління фінансами та забезпечити надійний облік фінансових операцій.

4 ВИМОГИ ДО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

4.1 Вимоги до функціональних характеристик

Програмне забезпечення повинно забезпечувати виконання наступних основних функції:

4.1.1 Аутентифікація

Аутентифікація є критично важливою функцією для забезпечення безпеки та конфіденційності даних користувачів. Вона дозволяє ідентифікувати користувача перед наданням доступу до ресурсів та функцій системи. Нижче наведено вимоги до функції аутентифікації:

1. Логін та Пароль:
   - Користувач повинен мати можливість зареєструвати обліковий запис, вказавши унікальне ім'я користувача та пароль;
   - Після успішної реєстрації користувач повинен мати можливість увійти в систему, ввівши свій логін та пароль;
   - Система повинна здійснювати перевірку введених даних, а у випадку невідповідності – повідомляти користувача про помилку.
2. Захист пароля:
   - Паролі користувачів повинні зберігатися у зашифрованому вигляді з використанням сучасних алгоритмів хешування, таких як bcrypt;
   - Система повинна вимагати від користувача створення складного пароля, який повинен включати мінімум 8 символів, великі і малі літери, цифри та спеціальні символи.
3. Двохфакторна аутентифікація (2FA):
   - Система повинна підтримувати двохфакторну аутентифікацію для додаткового рівня безпеки;

– Під час реєстрації користувачу повинно буде надсилатися посилання для підтвердження на його електронну пошту.

4. Відновлення доступу:

– У випадку, якщо користувач забув пароль, система повинна забезпечувати можливість його відновлення;

– Для цього користувач повинен мати можливість запросити відновлення пароля, ввівши свою електронну адресу. Система повинна надсилати лист з посиланням на сторінку для створення нового пароля.

5. Сесії та управління сеансами:

– Після успішного входу система повинна створювати сесію для користувача з використанням унікального токена;

– Токен сесії повинен бути збережений на стороні клієнта у вигляді захищеного HTTP cookie або у локальному сховищі браузера;

– Система повинна забезпечувати автоматичне завершення сесії після певного періоду неактивності користувача (наприклад, через 60 хвилин).

4.1.2 Користувацького інтерфейсу

– Головна сторінка
  - Відображення поточного балансу користувача;
  - Відображення останніх транзакцій;
  - Графічний аналіз доходів та витрат за певний період;
  - Віджети для швидкого доступу до основних функцій.

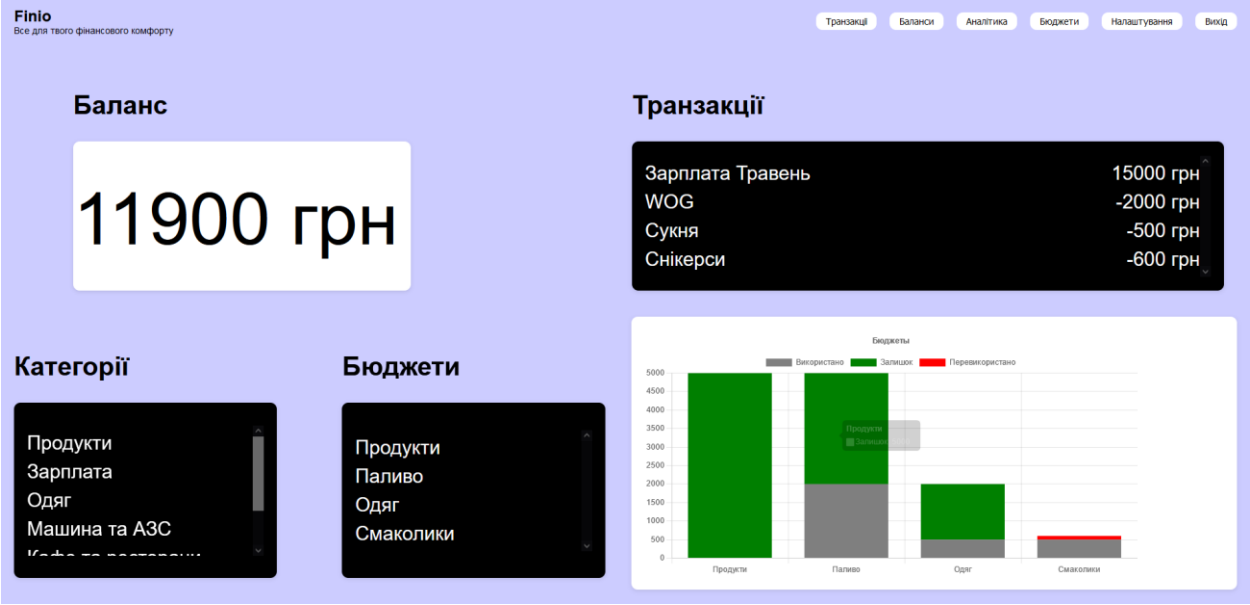


Рисунок 1.1 - Головна сторінка

– Облік доходів і витрат
  - Додавання, редагування та видалення транзакцій;
  - Категоризація доходів та витрат;
  - Введення додаткових деталей транзакцій (дата, сума, категорія, опис).



Рисунок 1.2 – Облік доходів і витрат

– Планування бюджету
  - Створення та редагування бюджетів для різних категорій витрат;
  - Відстеження виконання бюджету з автоматичними попередженнями про перевищення;
  - Аналіз відхилень фактичних витрат від запланованих.



Рисунок 1.3 – Планування бюджетів та витрат

– Фінансові звіти
  - Генерація місячних, квартальних та річних звітів;
  - Звіти по категоріях доходів і витрат;
  - Звіти по періодах та користувацьких налаштуваннях.



Рисунок 1.4 – Фінансові звіти

– Налаштування користувача
  - Управління обліковим записом (редагування профілю, зміна пароля);
  - Налаштування сповіщень (push, email);
  - Налаштування категорій витрат та доходів.

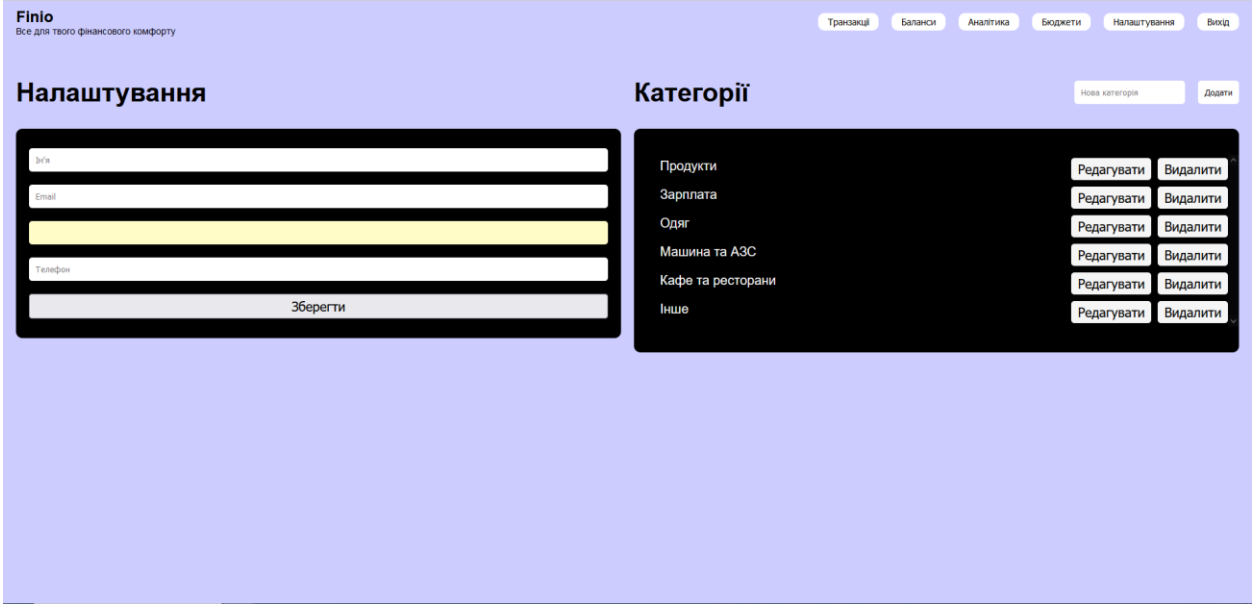


Рисунок 1.5 – Налаштування користувача

– Авторизація
  - Реєстрація нових користувачів;
  - Вхід для існуючих користувачів;
  - Відновлення пароля для користувачів, що його забули.



Рисунок 1.6 – Авторизація

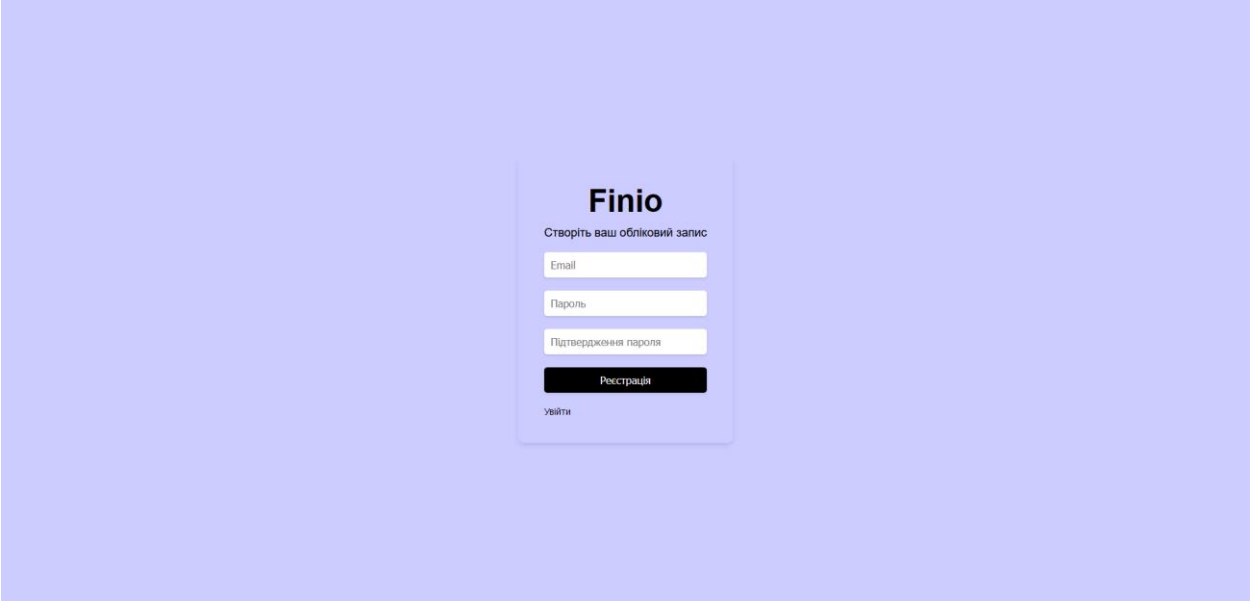


Рисунок 1.7 – Реєстрація

- Налаштування балансів
    - Додавання нових балансів;
    - Редагування та видалення балансів;
    - Відстеження балансів.



Рисунок 1.8 – Налаштування балансів

4.1.3 Для користувача:

– Реєстрація та вхід:
    - Можливість створення нового облікового запису;
    - Можливість входу до системи за допомогою email та пароля;
    - Опція відновлення пароля для забутих облікових даних.
– Персоналізація:
    - Налаштування персональних даних користувача;
    - Зміна пароля та інших налаштувань облікового запису;
    - Налаштування отримання сповіщень (push, email).
– Облік фінансів:
    - Додавання нових транзакцій (доходи та витрати);
    - Редагування та додавання балансів;
    - Редагування та видалення існуючих транзакцій;

- Категоризація транзакцій для зручного обліку.

– Аналіз фінансів:
  - Перегляд аналітики та звітів щодо доходів та витрат;
  - Використання графіків та діаграм для візуалізації фінансових даних;
  - Порівняння фінансових показників за різні періоди.

– Планування бюджету:
  - Створення бюджетів для різних категорій витрат;
  - Відстеження фактичних витрат у порівнянні з запланованими бюджетами;
  - Автоматичні сповіщення про перевищення бюджету.

4.1.4 Додаткові вимоги:

– Кросплатформеність:
  - Забезпечення роботи веб-застосунку на різних браузерах (Chrome, Firefox, Safari, Edge);
  - Оптимізація інтерфейсу для використання на мобільних пристроях (адаптивний дизайн).

– Масштабованість:
  - Можливість розширення функціоналу та додавання нових модулів без значних змін в існуючій архітектурі;
  - Підтримка великої кількості одночасних користувачів без втрати продуктивності.

4.2 Вимоги до надійності

Клієнтська інформація повинна перевірятися на рівні представлення перед обробкою бізнес логікою системи. У разі некоректного вводу, користувач має отримувати відповідь зі статусом 403 (FORBIDDEN) та детальним поясненням помилки. У випадку невалідних даних система не

повинна виконувати жодних дій, окрім формування відповіді для користувача та логування відповідної помилки.

Для забезпечення надійності системи кожна задача повинна виконуватися в ізольованому середовищі, з можливістю обмеження використання системних ресурсів.

Для захисту від некоректних дій користувачів передбачена система ролей, яка перевіряє права на виконання відповідних операцій. Надійність системи контролю доступу забезпечується впровадженням механізму роботи JWT. Паролі в базі даних мають бути хешовані. Access токен не повинен включати вразливу інформацію, щоб запобігти її розповсюдженню поза межами застосунку; ця інформація має бути доступною лише на рівні сервера. Access токен має мати короткий термін валідності(60 хвилин).

Для забезпечення цілісності даних у реляційній базі даних її структура повинна задовольняти 3.5 нормальній формі (включаючи норму Бойса-Кодда). Система повинна перевіряти правильність наявності реляцій між сутностями, наданими користувачем при зверненні до API, враховуючи ієрархічну структуру. Зважаючи на збереження певної інформації в пам'яті сервера автоматизації (зареєстровані відкладені процеси виконання), при його зупинці ця інформація буде втрачена, тому необхідно передбачити механізми збереження та відновлення цих даних.

4.3 Умови експлуатації

Умови експлуатації згідно СанПін 2.2.2.542 – 96.

4.3.1 Вид обслуговування

У ході експлуатації рекомендується регулярно оновлювати версію системи.

4.3.2 Обслуговуючий персонал

Вимоги до обслуговуючого персоналу не висуваються

4.4 Вимоги до складу і параметрів технічних засобів

Мінімальна конфігурація технічних засобів:

– тип процесору: Intel Core i5;
– об‘єм ОЗП: 4 Гб;
– підключення до мережі Інтернет зі швидкістю від 20 мегабіт.

Рекомендована конфігурація технічних засобів:

– тип процесору: Intel Core i7;
– об‘єм ОЗП: 8 Гб;
– підключення до мережі Інтернет зі швидкістю від 100 мегабіт.

4.5 Вимоги до інформаційної та програмної сумісності

Програмне забезпечення повинно працювати під управлінням операційних систем усіх операційних систем, які забезпечують з'єднання з Інтернетом і можуть надати доступ до веб-застосунку через браузер.

4.5.1 Вимоги до вхідних даних

Вхідні дані надаються користувачами за допомогою інтерфейсу

4.5.2 Вимоги до вихідних даних

Результат повертається в графічному форматі на інтерфейсі

4.5.3 Вимоги до мови розробки

Розробку виконати на мові програмування Java та TypeScript

4.5.4 Вимоги до середовища розробки

Розробку виконати на платформі IntelijIDEA

4.5.5 Вимоги до представленню вихідних кодів

Вихідний код програми має бути представлений у вигляді файлів із розширенням .java, .css, .ts, .html

4.6 Вимоги до маркування та пакування

Вимоги до маркування та пакування не висуваються.

4.7 Вимоги до транспортування та зберігання

Вимоги до транспортування та зберігання не висуваються.

4.8 Спеціальні вимоги

Здійснити та описати сценарії розгортання системи в локальному середовищі, а також за допомогою Docker; опублікувати код застосунку в публічному репозиторії, де підтримувати його стабільну та експериментальну версії.

5 ВИМОГИ ДО ПРОГРАМНОЇ ДОКУМЕНТАЦІЇ

5.1 Попередній склад програмної документації

У склад супроводжувальної документації повинні входити наступні документи на аркушах формату А4:

– пояснювальна записка;
– технічне завдання;
– текст програми;
– програма та методика тестування;
– керівництво користувача;
– керівництво програміста.

Графічна частина повинна бути виконана на аркушах формату А3 та містити наступні документи:

– схема структурна варіантів використання;
– схема бази даних;
– схема архітектури програмного забезпечення
– схема структурна діяльності;
– макет вигляду екранних форм.

5.2 Спеціальні вимоги до програмної документації

Програмні модулі, котрі розробляються, повинні бути задокументовані, тобто тексти програм повинні містити всі необхідні коментарі.

6 СТАДІЇ І ЕТАПИ РОЗРОБКИ

№	Назва етапу	Строк	Звітність
1.	Вивчення рекомендованої літератури	11.03.2024	Картка проєкту
2.	Розробка технічного завдання	18.03.2024	Технічне завдання
3.	Аналіз вимог та уточнення специфікацій	20.03.2024	Специфікації програмного забезпечення
4.	Проектування структури програмного забезпечення, проектування компонентів	30.03.2024	Схема структурна програмного забезпечення та специфікація компонентів (діаграма класів, схема алгоритму)
5.	Програмна реалізація програмного забезпечення	05.04.2024	Тексти програмного забезпечення
6.	Тестування програмного забезпечення	06.04.2024	Тести, результати тестування
7.	Розробка матеріалів текстової частини проекту	30.04.2024	Пояснювальна записка
8.	Розробка матеріалів графічної частини проекту	8.04.2024	Графічний матеріал проекту
9.	Оформлення технічної документації проекту	12.04.2024	Технічна документація

7 ПОРЯДОК КОНТРОЛЮ ТА ПРИЙМАННЯ

Тестування розробленого програмного продукту виконується відповідно до “Програми та методики тестування”.

Пояснювальна записка
до дипломного проєкту

на тему: **Веб-застосунок для управління особистими фінансами**

КПІ.ІТ.045440.02.81

Київ – 2024

ЗМІСТ

ВСТУП 6
1 ПЕРЕДПРОЄКТНЕ ОБСТЕЖЕННЯ ПРЕДМЕТНОЇ ОБЛАСТІ 8
1.1 Загальні положення 8
1.2 Змістовний опис та аналіз предметної області 9
1.3 Аналіз існуючих рішень 10
1.3.1 Аналіз відомих програмних продуктів 10
1.3.2 Аналіз відомих алгоритмічних та технічних рішень 12
1.4 Опис бізнес-процесів 15
1.5 Постановка задачі 20
1.6 Варіанти використання програмного забезпечення 21
1.7 Розроблення функціональних вимог 30
1.8 Розроблення нефункціональних вимог 37
1.9 Постановка задачі 37
Висновки до розділу 38
2 КОНСТРУЮВАННЯ ТА РОЗРОБЛЕННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ 39
2.1 Архітектура програмного забезпечення 39
2.2 Обґрунтування вибору засобів розробки 40
2.3 Конструювання програмного забезпечення 44
2.4 Аналіз безпеки даних 51
Висновки до розділу 52
3 АНАЛІЗ ЯКОСТІ ТА ТЕСТУВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ 53
3.1 Аналіз якості ПЗ 53
3.2 Опис процесів тестування 55
3.3 Опис контрольного прикладу 62
Висновки до розділу 63

4 ВПРОВАДЖЕННЯ ТА СУПРОВІД ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ . 65
4.1 Розгортання програмного забезпечення 65
4.2 Супровід програмного забезпечення 71
Висновки до розділу 71
ВИСНОВКИ 73
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ 74
ДОДАТОК А ЗВІТ ПОДІБНОСТІ 76
ДОДАТОК Б АЛГОРИТМ ШИФРУВАННЯ ACCESS ТОКЕНУ 77
ДОДАТОК В ІНСТАНС НА AWS EC2 78

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

IntelliJ IDEA	–	IntelliJ IDEA — комерційне інтегроване середовище розробки для різних мов програмування
API	–	Прикладний програмний інтерфейс (Application Programming Interface)
Java	–	об'єктно-орієнтована мова програмування
IT	–	Інформаційні технології (Information Technology)
ER	–	Діаграма сутність-зв'язок (Entity-Relation diagram)
ОС	–	Операційна система (Operating System)
БД	–	База даних
2FA	–	метод додаткового захисту вашого облікового запису
Spring	–	це програмний каркас (фреймворк) з відкритим кодом та контейнери з підтримкою інверсії керування для платформи Java.
Hibernate	–	засіб відображення між об'єктами та реляційними структурами (object-relational mapping, ORM) для платформи Java
PostgreSQL	–	об'єктно-реляційна система керування базами даних
JSON	–	це текстовий формат обміну даними між комп'ютерами
ACID	–	це набір властивостей, що гарантують надійну роботу транзакцій бази даних: атомарність, узгодженість, ізольованість, довговічність
SSL	–	це засіб захисту особистої інформації користувачів в інтернеті
HTML	–	Мова гіпертекстової розмітки (Hypertext Markup Language)

React	–	відкрита JavaScript бібліотека для створення інтерфейсів користувача, яка покликана вирішувати проблеми часткового оновлення вмісту вебсторінки, з якими стикаються в розробці односторінкових застосунків
DOM	–	це програмний інтерфейс (API), який дає змогу розробникам отримувати доступ до структури та вмісту HTML- або XML-документів

ВСТУП

У сучасному світі управління особистими фінансами стає дедалі важливішим для широкого кола користувачів. Необхідність контролю доходів та витрат вимагають наявності ефективних інструментів для фінансового планування та аналізу. Веб-застосунок для управління особистими фінансами «Finio» розробляється з метою забезпечення користувачів такими інструментами.

Актуальність розробки зумовлена потребою в зручному та функціональному засобі для управління фінансами, який би відповідав сучасним вимогам користувачів. З розвитком цифрових технологій та зростанням кількості фінансових операцій стає все більш актуальною необхідність мати інструмент, що дозволяє ефективно планувати бюджет, відстежувати витрати та доходи, а також аналізувати фінансові дані для прийняття обґрунтованих рішень. Сучасні користувачі очікують від фінансових застосунків не лише базових функцій, але й додаткових можливостей, таких категоризування транзакцій. Таким чином, розробка такого інструменту сприятиме підвищенню фінансової грамотності користувачів, полегшить управління особистими та бізнес-фінансами, а також допоможе уникнути фінансових ризиків і досягти стабільності та процвітання.На сьогодні існує безліч програмних продуктів, спрямованих на управління фінансами, проте більшість із них або надто складні у використанні, або не забезпечують достатньої функціональності для задоволення всіх потреб користувачів.

Застосування «Finio» можливе у широкому колі сфер, такі як індивідуальне управління фінансами, фінансове планування для малих підприємств, а також як допоміжний інструмент для фінансових консультантів. Це програмне забезпечення здатне забезпечити користувачам

зручний інтерфейс, надійні інструменти для аналізу та планування, а також безпеку персональних даних.

Мета розробки - створення зручного інструменту для управління особистими фінансами, який дозволяє користувачам контролювати доходи та витрати, планувати бюджети.

Завдання, які будуть вирішені:

- Надання користувачам можливості додавати, редагувати та видаляти транзакції;
- Забезпечення інструментів для категоризації доходів та витрат;
- Розробка функцій для створення та відстеження виконання бюджетів;
- Розробка зручного та інтуїтивно зрозумілого користувацького інтерфейсу;
- Реалізація функцій для управління обліковим записом та налаштуваннями користувача;
- Забезпечення авторизації користувачів та можливості відновлення пароля.

1 ПЕРЕДПРОЄКТНЕ ОБСТЕЖЕННЯ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Загальні положення

Розробка програмного забезпечення пройшла значну еволюцію з початку 2010-х років. На основі досвіду, накопиченого протягом попередніх десятиліть, з'явилися нові стандарти і процедури, які змінюють підходи до розробки ПЗ. Одним з ключових напрямків стало впровадження методологій, орієнтованих на швидке і якісне реагування на зміни, зокрема DevOps.

DevOps: Витоки і розвиток

DevOps виник як відповідь на потребу у швидкій інтеграції та доставці програмного забезпечення. Відмінність DevOps від традиційних підходів полягає в інтеграції процесів розробки і операційної діяльності. Основним принципом DevOps є співпраця між розробниками і операційними командами для досягнення спільних цілей. Відомий вислів Werner Vogel: "You build it, you run it" став девізом цього підходу.

Continuous Integration (CI) та Continuous Delivery (CD)

CI/CD стали основними компонентами DevOps. Continuous Integration полягає у постійній інтеграції коду з автоматичною перевіркою його якості. Це дозволяє виявляти і виправляти помилки на ранніх етапах розробки. Martin Fowler визначає CI як практику, що вимагає від розробників щоденної інтеграції коду, яка перевіряється автоматичними збірками.

Continuous Delivery, в свою чергу, забезпечує можливість автоматичної доставки програмного забезпечення у будь-який момент. Це означає, що зміни в коді можуть бути швидко та надійно впроваджені в продуктивне середовище.

Поява та поширення хмарних технологій

З впровадженням Amazon Web Services (AWS) у 2006 році, доступність хмарних ресурсів значно спростила методи розгортання ПЗ. Хмарні сервіси

дозволяють уникнути необхідності налаштування і підтримки власних серверів, що особливо важливо для масштабованих додатків.

Infrastructure as Code (IaC)

IaC став невід'ємною частиною сучасної розробки ПЗ. Цей підхід полягає у представленні конфігурації інфраструктури у вигляді програмного коду, який зберігається у репозиторіях і може бути автоматично розгорнутий. IaC дозволяє автоматизувати налаштування і забезпечити повторюваність середовищ розробки, тестування та продуктивності.

1.2 Змістовний опис та аналіз предметної області

Загальний огляд предметної області включає в себе визначення ключових понять, таких як бюджет, дохід, витрати, інвестиції.

- Бюджет - план витрат на певний період часу. Бюджет може включати різні категорії витрат, такі як житло, харчування, розваги, страхування, інвестиції;
- Дохід - сума грошей, яку отримує особа. Дохід може бути заробітною платою, процентами від вкладів, дивідендами від акцій, пенсійними виплатами;
- Витрати - сума грошей, яку особа витрачає на придбання товарів або послуг. Витрати можуть бути пов'язані з життєвими потребами, такими як продукти харчування та житло, або з іншими витратами, такими як розваги, подорожі, медичні послуги;
- Інвестиції - розміщення грошових коштів або інших активів з метою отримання прибутку в майбутньому. Інвестиції можуть бути здійснені у різних формах, включаючи акції, облігації, нерухомість, цінні папери.

Для вирішення задачі дипломного проекту було вирішено використати веб-застосунок.

Веб-застосунок має наступні переваги порівнянно з іншими типами додатків:

- Не потребує додаткового встановлення інших бібліотек чи програм;
- Оновлення застосунку відбувається в реальному часі та непотребує від користувачів додаткових дій;
- Доступ можна отримати з будь-якого пристороюю.

1.3 Аналіз існуючих рішень

Проаналізуємо відоме на сьогодні алгоритмічне забезпечення у даній області та технічні рішення, що допоможуть у реалізації веб-застосунку для управління особистими фінансами Далі будуть розглянуті готові програмні рішення, допоміжні програмні засоби та засоби розробки.

1.3.1 Аналіз відомих програмних продуктів

Існують різноманітні програмні продукти для управління особистими фінансами, які частково або повністю відповідають функціоналу. Це продукти MoneyManager та BudgetTracker

Таблиця 1.3 – Порівняння з аналогом

Функціонал	Веб-застосунок **Finio**	Програмний продукт **MoneyManager**	Програмний продукт **BudgetTracker**	Пояснення
Додавання транзакцій	Так	Так	Так	Застосунок має мати можливість додавати нові транзакції

Продовження таблиці 1.3

Редагування транзакцій	Так	Так	Так	Застосунок має мати можливість редагувати існуючі транзакції
Видалення транзакцій	Так	Так	Так	Застосунок має мати можливість видаляти існуючі транзакції
Категоризація транзакцій	Так	Так	Так	Застосунок має мати можливість додавати категорії до транзакцій
Створення та відстеження бюджетів	Так	Так	Так	Застосунок має мати можливість додавати нові бюджети та відслідковувати прогрес по ним

Продовження таблиці 1.3

Аналітичні інструменти	Так	Так	Так	Застосунок має відображати графіки для аналітики
Користувацький інтерфейс	Зручний	Не дуже зручний	Зручний	Застосунок має мати зручний користувацький інтерфейс
Безпека даних	Парольна аутентифікація, обмежені можливості	2FA, захист даних	2FA, захист даних	Застосунок має забезпечити конфідеційність даних користувачів

1.3.2 Аналіз відомих алгоритмічних та технічних рішень

Потрібно обрати агоритм категоризації транзакцій. Можливі варіанти: класифікація на основі ключових слів, машинне навчання, правила фільтрації. На даному етапі розвитку проекту можна ввести категоризацію за даними отриманими від користувача, а в подальшому використати машинне навчання чи ШІ для категоризації.

Потрібно обрати архітектуру проекту. Можливі варіанти: Моноліт, мікросервісна архутектура, сервісно-орієнтована архітектура. Найкращим варіантом для такого проекту буде мікросурвісна архітектура. Вона дозволяє розділити великий та складний додаток на невеликі та незалежні сервіси, дозволяє використовувати різні технології та мови програмування для кожного сервісу, виконувати швидке відновлення роботи після відмови.

Вибір технологічного стеку:

Java як мова для розробки бекенду:

- Є крос-платформеною мовою програмування, що означає, що програми, написані на Java, можуть працювати на різних операційних системах без змін;
- Має потужні фреймворки та інструменти для розробки веб-додатків, такі як Spring, Hibernate, що дозволяє швидко створювати високоякісні та масштабовані веб-застосунки;
- Має вбудовані механізми безпеки, такі як підписані аплети, управління пам'яттю та безпечні контейнери;
- Має велику та активну спільноту розробників, а також багато ресурсів, бібліотек та інструментів, що спрощують розробку та підтримку веб-застосунків;
- Дозволяє легко інтегрувати веб-застосунок з іншими системами та сервісами, завдяки різноманітним бібліотекам та інструментам для роботи з веб-службами та базами даних;
- Має високу швидкодію та можливості для масштабування.

PostgreSQL як база даних:

- Здатність обробляти великі обсяги даних і високу навантаженість без втрати продуктивності;
- Підтримує складні SQL-запити, транзакції, зберігання та обробку JSON-даних, а також багаторівневу індексацію;
- Має високу репутацію за свою стабільність і відповідність стандартам ACID (атомарність, узгодженість, ізольованість, довговічність);
- Надає потужні засоби для забезпечення безпеки даних, включаючи підтримку SSL, розширені механізми автентифікації та контроль доступу на рівні рядків;

- Має велику та активну спільноту розробників, що забезпечує постійне оновлення, виправлення помилок та розширення функціональних можливостей;
- Підтримує створення користувацьких функцій, типів даних, індексів та інших розширень. Це дозволяє налаштовувати базу даних під специфічні вимоги проекту та розширювати її функціональність відповідно до потреб;
- Легко інтегрується з різними мовами програмування і фреймворками, що спрощує розробку та підтримку застосунку. Вона також підтримує реплікацію та кластеризацію.

React як бібліотека для розробки фронтенду:

- Надає можливість створювати користувацькі інтерфейси веб-застосунків за допомогою компонентної моделі;
- Дозволяє швидко оновлювати та відображати дані на сторінці завдяки використанню віртуального DOM та ефективних алгоритмів оновлення;
- Має велику спільноту розробників та багато ресурсів, бібліотек та інструментів, що спрощують розробку та підтримку веб-застосунків.

AWS для розгортання проекту в хмарному середовищі:

- Надає широкий спектр послуг, що дозволяє гнучко масштабувати ресурси відповідно до потреб вашого веб-застосунку;
- Має глобальну мережу дата-центрів, яка забезпечує високий рівень доступності та надійності для ваших додатків. Системи AWS розроблені з урахуванням вимог до високої доступності та автоматичного відновлення після відмов;
- Надає широкі можливості для захисту даних та інфраструктури, включаючи інструменти для керування доступом, шифрування даних

та моніторингу безпеки. AWS також відповідає ряду міжнародних стандартів безпеки та відповідності, таких як GDPR, HIPAA, PCI DSS та інші;

- Має багато різноманітних сервісів, що можуть бути використані для розгортання та підтримки веб-застосунку, включаючи обчислювальні, мережеві, бази даних, аналітичні та інші сервіси. Крім того, AWS легко інтегрується з іншими сервісами та інструментами, такими як Docker, Kubernetes, Jenkins;
- Пропонує гнучкі та конкурентоспроможні ціни, які дозволяють ефективно використовувати ресурси та зменшувати витрати на інфраструктуру.

1.4 Опис бізнес-процесів

Під час аналізу вимогу було визначено два типи користувачів авторизований та неавторизований. Неавторизований користувач матиме змогу зареєструватись або авторизуватись. Авторизований користувач матиме можливість перегляду бюджетів, транзакцій, поточного балансу, графіків, редагування бюджетів, транзакцій, балансів.



Рисунок 1.1 – Реєстрація користувача

– Реєстрація користувача
    - Користувач переходить на сторінку реєстрації.

- Вводить необхідні дані: ім'я, електронну пошту, пароль;
- Система перевіряє коректність введених даних;
- Якщо дані коректні, система створює обліковий запис та надсилає підтвердження на електронну пошту користувача;
- Користувач підтверджує свою електронну пошту та отримує доступ до системи.

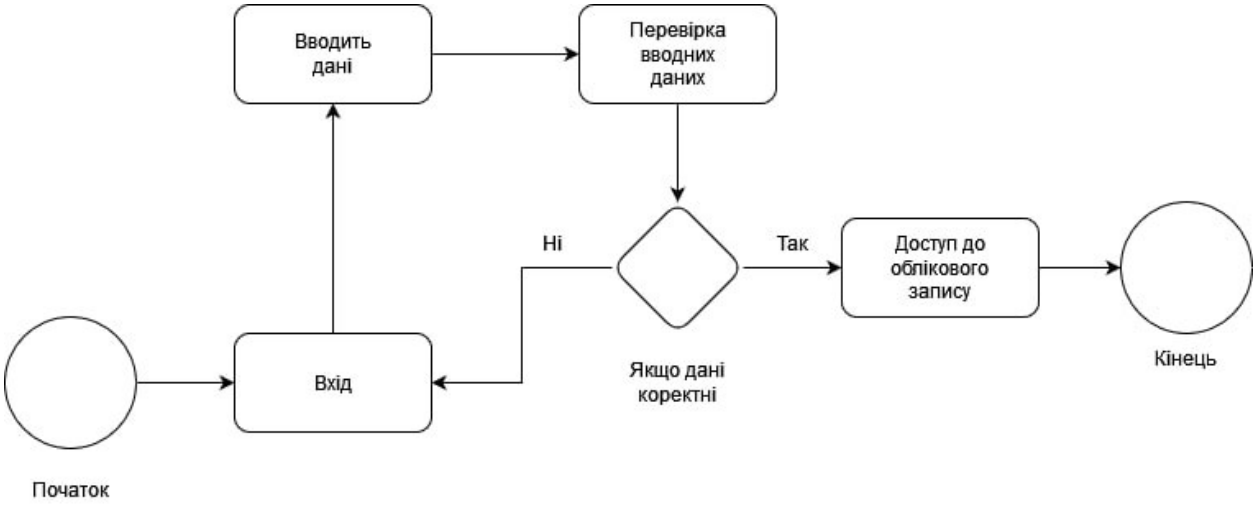


Рисунок 1.2 – Авторизація та вхід у систему

– Авторизація та вхід у систему
  - Користувач переходить на сторінку входу;
  - Вводить свої облікові дані (електронну пошту та пароль);
  - Система перевіряє дані та, у разі успіху, надає доступ до облікового запису користувача.

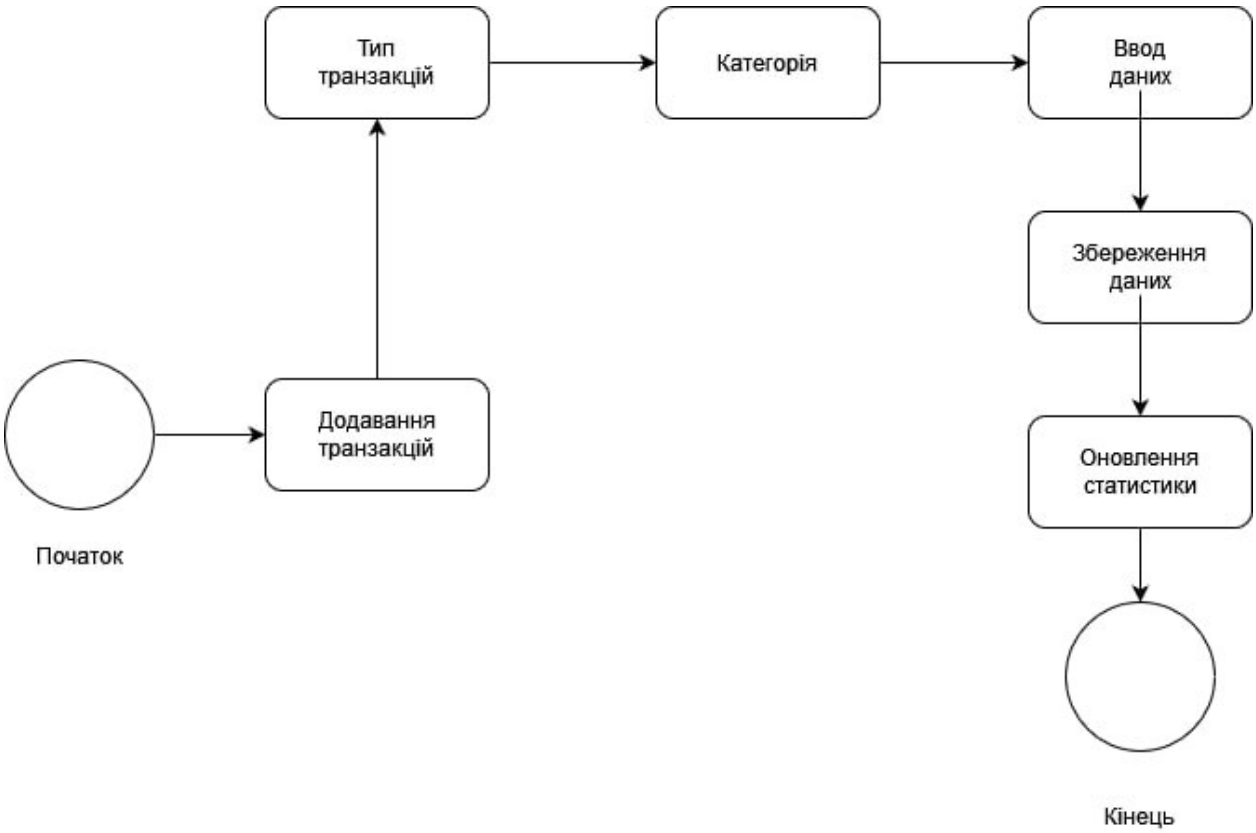


Рисунок 1.3 – Додавання фінансових даних

– Додавання фінансових даних
  - Користувач переходить до розділу додавання транзакцій;
  - Вибирає тип транзакції (дохід чи витрати), категорію, вводить суму, дату та інші деталі;
  - Система зберігає введені дані та оновлює загальну статистику фінансів користувача.

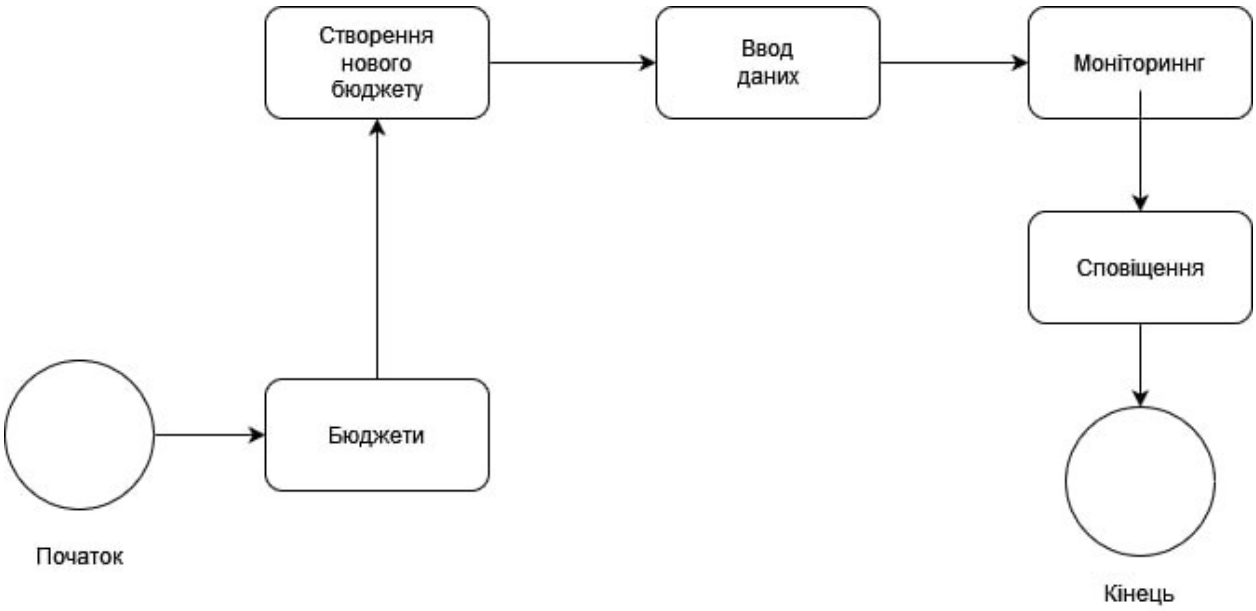


Рисунок 1.4 – Створення та управління бюджетами

– Створення та управління бюджетами
  - Користувач переходить до розділу бюджетів;
  - Створює новий бюджет, вводячи категорії витрат, ліміти та період дії бюджету;
  - Система моніторить витрати користувача та сповіщає його про наближення до ліміту або перевищення бюджету.

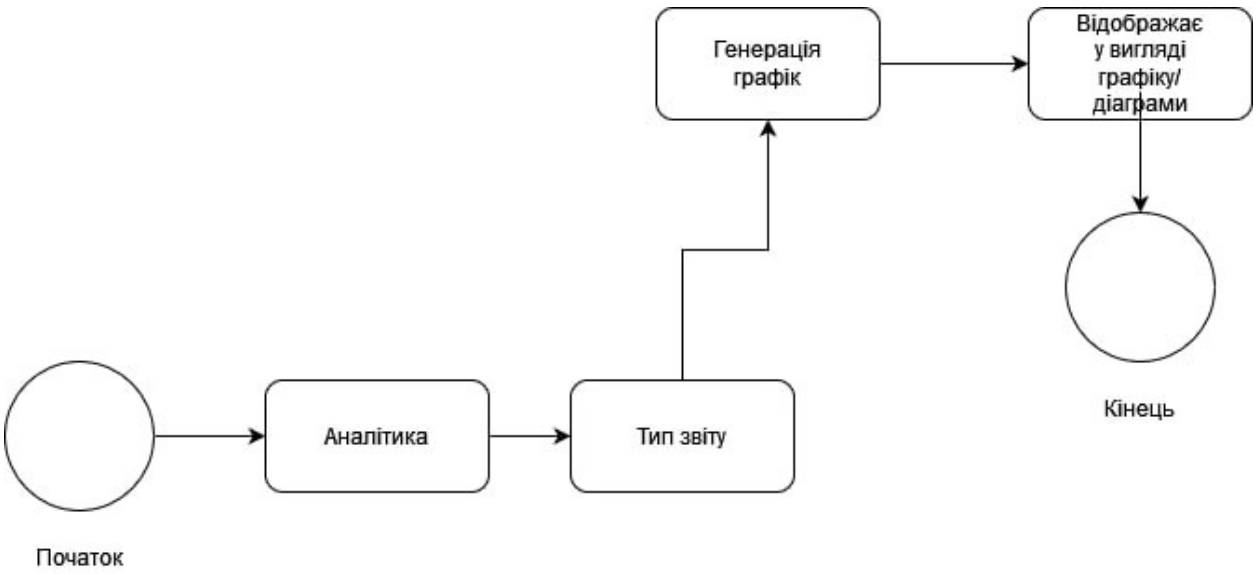


Рисунок 1.5 – Генерація звітів та аналіз фінансових даних

– Генерація звітів та аналіз фінансових даних
    - Користувач переходить до розділу аналітики;
    - Вибирає тип звіту (місячний, річний, по категоріях тощо);
    - Система генерує графік на основі збережених даних та відображає його у вигляді графіків, діаграм та таблиць.

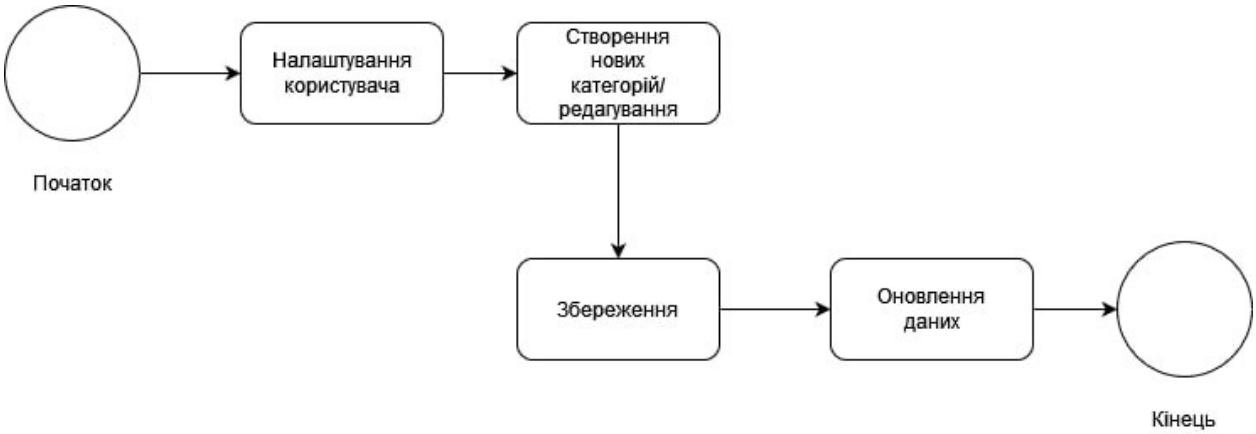


Рисунок 1.6 – Управління категоріями

– Управління категоріями
    - Користувач переходить до розділу налаштувань користувача;
    - Створює нові категорії або редагує існуючі для кращої організації транзакцій;
    - Система зберігає зміни та застосовує їх до відповідних транзакцій.

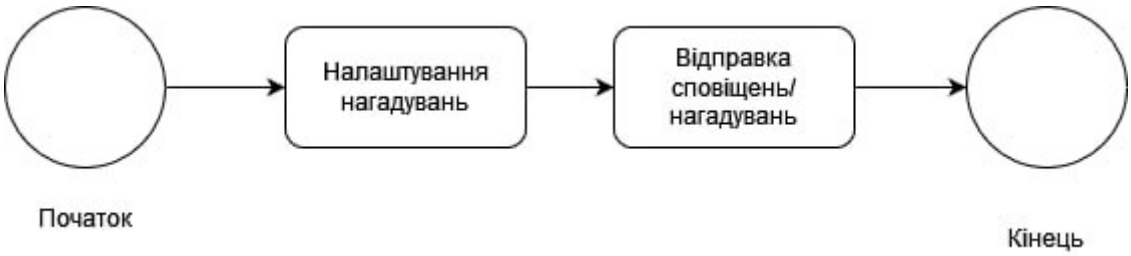


Рисунок 1.7 – Нагадування та сповіщення

– Нагадування та сповіщення
    - Користувач налаштовує нагадування про регулярні платежі та інші фінансові події;
    - Система автоматично надсилає нагадування та сповіщення користувачеві через електронну пошту.

1.5 Постановка задачі

Даний веб-застосунок допоможе проаналізувати витрату коштів та допомогти у розвитку фінансової грамотності.

Буде реалізовано веб-застосунок для полегшення обліку фінансів. В якому користувач зможе відстежувати свої витрати та доходи, додавати та видаляти категорії на які витрачає кошти. Переглядати аналітику за місяць, тиждень, півроку та рік. Також, користувач зможе розподіляти свій бюджет на різні категорії. Переглядати баланс на картках та, навіть, відслідковувати свої готівкові кошти, щоб не втратити та не тримати ці дані в голові.

Задачі, що вирішуються в роботі

Управління бюджетом - користувачі можуть створювати персоналізовані бюджети на певний період. Налаштовувати категорії витрат (продукти, машина, медицина) з визначенням лімітів для кожної категорії.

Моніторинг виконання бюджетів - відображення поточного стану бюджету з можливістю перегляду деталей за категоріями. Візуалізація витрат у вигляді графіків та діаграм. Сповіщення користувачів про перевищення встановлених лімітів у категоріях витрат. Аналіз бюджетів - аналітика про виконання бюджетів за певний період, у вигляді графіків, діаграм.

Управління витратами - користувачі можуть вводити свої витрати, вказуючи суму, дату та категорію витрат.

Аналіз витрат - аналітика за певний період часу (тиждень,місяць, півроку, рік). Візуалізація витрат у вигляді діаграм, графіків..

Відстеження особистих фінансів - введення даних про доходи з різних джерел (заробітна плата, інвестиції, подарунки). Відстеження залишків на рахунках та балансів.

Бюджетування - розподілення витрат на конкретні категорії. Нагадування про заповнення даних про витрати. Сповіщення про наближення

до межі бюджету у певних категоріях. Сповіщення про важливі зміни у фінансовому стані.

1.6 Варіанти використання програмного забезпечення

Головною функцією програмного забезпечення є облік особистих фінансів користувача більше функцій можна побачити на рисунку 1.8.

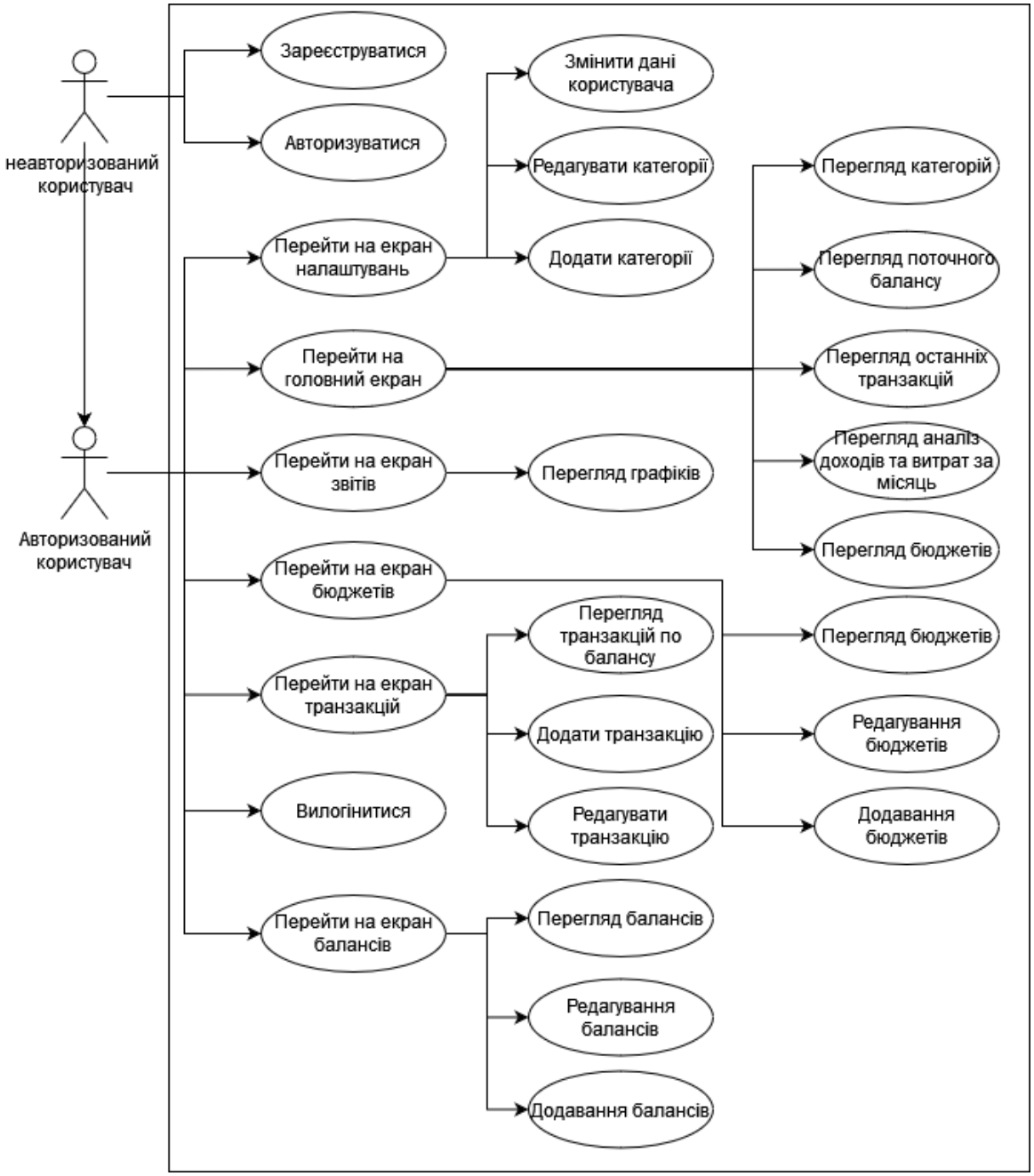


Рисунок 1.8 – Діаграма варіантів використання

В таблицях 1.6.1 – 1.6.14 наведені варіанти використання веб-застосунку.

Таблиця 1.6.1 – Варіант використання UC-01

Use case name	Перегляд сторінки авторизації
Use case ID	UC-01
Goals	Показати сторінку авторизації
Actors	Неавторизований користувач
Trigger	Користувач відкриває сайт
Pre-Conditions	-
Flow of Events	Користувач відкриває сайт. В результаті відображається сторінка з формою для логіну
Extensions	-
Post-Conditions	-

Таблиця 1.6.2 – Варіант використання UC-02

Use case name	Реєстрація користувача
Use case ID	UC-02
Goals	Користувач зареєстрований
Actors	Неавторизований користувач
Trigger	Користувач натискає кнопку «Зареєструватись»
Pre-Conditions	-

Продовження таблиці 1.6.2

Flow of Events	Користувач переходить на сторінку реєстрації. Вводить дані для реєстрації та натискає кнопку реєстрації. У результаті користувач матиме свій аккаунт і зможе авторизуватися
Extensions	У випадку невірно введених даних повинна показатись вікно помилки
Post-Conditions	Користувач отримує свій аккаунт

Таблиця 1.6.3 – Варіант використання UC-03

Use case name	Авторизація користувача
Use case ID	UC-03
Goals	Користувач авторизований та перенаправлений на головну сторінку
Actors	Неавторизований користувач
Trigger	Користувач натискає кнопку «Увійти»
Pre-Conditions	-
Flow of Events	Користувач вводить дані для авторизації та натискає кнопку увійти. У результаті користувач буде авторизований в системі та перенаправлений на головну сторінку

Продовження таблиці 1.6.3

Extensions	У випадку невірно введених даних повинна показатись вікно помилки
Post-Conditions	Користувач авторизований та перенаправлений на головну сторінку

Таблиця 1.6.4 – Варіант використання UC-04

Use case name	Додавання категорії
Use case ID	UC-04
Goals	Додана нова категорія
Actors	Авторизований користувач
Trigger	Користувач натискає кнопку «Додати категорію»
Pre-Conditions	-
Flow of Events	Користувач натискає кнопку «Додати категорію». З’являється сайдбар. Користувач вводить необхідні дані та натискає кнопку створити.
Extensions	У випадку невірно введених даних повинна показатись вікно помилки
Post-Conditions	Нова категорія створена

Таблиця 1.6.5 – Варіант використання UC-05

Use case name	Редагування категорії
Use case ID	UC-05
Goals	Категорія редагована
Actors	Авторизований користувач

Продовження таблиці 1.6.5

Trigger	Користувач натискає на необхідну категорію
Pre-Conditions	-
Flow of Events	Користувач натискає на необхідну категорію. З’являється сайдбар. Користувач вводить необхідні дані та натискає кнопку редагувати.
Extensions	У випадку невірно введених даних повинна показатись вікно помилки
Post-Conditions	Категорія редагована

Таблиця 1.6.6 – Варіант використання UC-06

Use case name	Зміна даних користувача
Use case ID	UC-06
Goals	Дані користувача змінено
Actors	Авторизований користувач
Trigger	Користувач змінює особисті дані
Pre-Conditions	-
Flow of Events	Користувач змінює необхідні дані та натискає кнопку зберегти. Після чого вони зберігаються.
Extensions	У випадку невірно введених даних повинна показатись вікно помилки
Post-Conditions	Дані користувача редаговані

Таблиця 1.6.7 – Варіант використання UC-07

Use case name	Перегляд графіків

Продовження таблиці 1.6.7

Use case ID	UC-07
Goals	Сторінка звітів відкрита
Actors	Авторизований користувач
Trigger	Користувач натискає на графік на головному екрані
Pre-Conditions	-
Flow of Events	Користувач натискає на графік на головному екрані після чого його перенаправляє на сторінку зі звітами
Extensions	-
Post-Conditions	-

Таблиця 1.6.8 – Варіант використання UC-08

Use case name	Додавання транзакції
Use case ID	UC-08
Goals	Транзакцію додано
Actors	Авторизований користувач
Trigger	Користувач натискає «Додати транзакцію»
Pre-Conditions	-
Flow of Events	Користувач натискає «Додати транзакцію». Відкривається сайдбар. Користувач вводить необхідні дані та натискає «Створити»
Extensions	У випадку невірно введених даних повинна показатись вікно помилки
Post-Conditions	Транзакція створена

Таблиця 1.6.9 – Варіант використання UC-09

Use case name	Редагування транзакції
Use case ID	UC-09
Goals	Транзакція редагована
Actors	Авторизований користувач
Trigger	Користувач натискає на необхідну транзакцію
Pre-Conditions	-
Flow of Events	Користувач натискає на необхідну транзакцію. Відкривається сайдбар. Користувач вводить необхідні дані та натискає «Редагувати»
Extensions	У випадку невірно введених даних повинна показатись вікно помилки
Post-Conditions	Транзакція редагована

Таблиця 1.6.10 – Варіант використання UC-10

Use case name	Редагування балансів
Use case ID	UC-10
Goals	Баланс редагований
Actors	Авторизований користувач
Trigger	Користувач натискає на необхідний баланс
Pre-Conditions	-

Продовження таблиці 1.6.10

Flow of Events	Користувач натискає на необхідний баланс. Відкривається сайдбар. Користувач вводить необхідні дані та натискає «Редагувати»
Extensions	У випадку невірно введених даних повинна показатись вікно помилки
Post-Conditions	Баланс редагований

Таблиця 1.6.11 – Варіант використання UC-11

Use case name	Додавання балансів
Use case ID	UC-11
Goals	Баланс створений
Actors	Авторизований користувач
Trigger	Користувач натискає «Додати баланс»
Pre-Conditions	-
Flow of Events	Користувач натискає додати баланс. Відкривається сайдбар. Користувач вводить необхідні дані та натискає «Створити»
Extensions	У випадку невірно введених даних повинна показатись вікно помилки
Post-Conditions	Баланс створений

Таблиця 1.6.12– Варіант використання UC-12

Use case name	Додавання бюджетів
Use case ID	UC-12

Продовження таблиці 1.6.12

Goals	Бюджет створений
Actors	Авторизований користувач
Trigger	Користувач натискає «Додати бюджет»
Pre-Conditions	-
Flow of Events	Користувач натискає додати баланс. Відкривається сайдбар. Користувач вводить необхідні дані та натискає «Створити»
Extensions	У випадку невірно введених даних повинна показатись вікно помилки
Post-Conditions	Бюджет створений

Таблиця 1.6.13 – Варіант використання UC-13

Use case name	Редагування бюджетів
Use case ID	UC-13
Goals	Бюджет редагований
Actors	Авторизований користувач
Trigger	Користувач натискає на необхідний бюджет
Pre-Conditions	-
Flow of Events	Користувач натискає на необхідний бюджет. Відкривається сайдбар. Користувач вводить необхідні дані та натискає «Редагувати»
Extensions	У випадку невірно введених даних повинна показатись вікно помилки

Продовження таблиці 1.6.13

Post-Conditions	Бюджет редагований

Таблиця 1.6.14 – Варіант використання UC-14

Use case name	Вихід
Use case ID	UC-14
Goals	Користувач вилогінений
Actors	Авторизований користувач
Trigger	Користувач натискає «Вихід»
Pre-Conditions	-
Flow of Events	Користувач натискає вихід. Його перекидає на сторінку авторизації.
Extensions	-
Post-Conditions	-

1.7 Розроблення функціональних вимог

Процес аналізу та розроблення вимог є необхідною частиною розробки програмного забезпечення. Загальну модель вимог наведено у таблиці 1.7.1, а у таблицях 1.7.2 – 1.7.13 наведено опис функціональних вимог до програмного забезпечення

Таблиця 1.7.1 – загальна модель вимог

№	Назва	ID вимоги	Пріоритет	Ризик
1	Реєстрація користувача	FR-01	Високий	Низький
2	Авторизація користувача	FR-02	Високий	Низький

Продовження таблиці 1.7.1

3	Додавання фінансових даних	FR-03	Високий	Середній
4	Управління бюджетами	FR-04	Високий	Середній
5	Генерація графіків	FR-05	Середній	Високий
6	Управління категоріями	FR-06	Середній	Низький
7	Нагадування та сповіщення	FR-07	Середній	Низький
8	Захист даних	FR-09	Високий	Високий
9	Інтерфейс користувача	FR-11	Високий	Низький
10	Аналітика	FR-12	Середній	Середній
11	Управління балансами	FR-13	Середній	Середній

Таблиця 1.7.2 – FR-01

Назва	Реєстрація користувача
Опис	Система повинна надати можливість користувачам створювати облікові записи за допомогою заповнення обов'язкових полів та підтвердженням електронної пошти. Після реєстрації користувачі зможуть використовувати інші функціональні можливості застосунку.

Таблиця 1.7.3 – FR-02

Назва	Авторизація користувача
Опис	Система повинна забезпечити можливість входу в обліковий запис користувача за допомогою введення логіна та пароля. Після успішної авторизації користувачеві буде надано доступ до особистого кабінету.

Таблиця 1.7.4 – FR-03

Назва	Додавання фінансових даних
Опис	Користувач повинен мати можливість додавати дані про свої фінансові транзакції, включаючи доходи, витрати та інвестиції, до системи. Система повинна забезпечувати можливість введення різноманітних фінансових операцій та відображення їх в особистому кабінеті користувача.

Таблиця 1.7.5 – FR-04

Назва	Управління бюджетами
Опис	Система повинна надати можливість користувачам створювати, редагувати та видаляти бюджети. Користувачі зможуть встановлювати обмеження на витрати та відстежувати їх відповідно до різних категорій.

Таблиця 1.7.6 – FR-05

Назва	Генерація графіків
Опис	Система повинна забезпечувати можливість генерації графіків та діаграм для візуалізації фінансових даних користувачів. Графіки мають демонструвати зміни у фінансовому стані, витрати за різними категоріями, та іншу корисну інформацію.

Таблиця 1.7.7 – FR-06

Назва	Управління категоріями
Опис	Система повинна дозволяти користувачам створювати, редагувати та видаляти категорії для класифікації своїх фінансових транзакцій. Користувачі зможуть прив'язувати транзакції до певних категорій для кращого аналізу витрат.

Таблиця 1.7.8 – FR-07

Назва	Нагадування та сповіщення
Опис	Система повинна надавати можливість налаштовувати нагадування та сповіщення для користувачів щодо майбутніх платежів, бюджетних обмежень та інших важливих фінансових подій.

Таблиця 1.7.9 – FR-08

Назва	Захист даних
Опис	Система повинна забезпечувати високий рівень захисту конфіденційності та цілісності фінансових даних користувачів. Це включає в себе шифрування даних, захист від несанкціонованого доступу та регулярні резервні копії для запобігання втрати даних.

Таблиця 1.7.10 – FR-09

Назва	Інтерфейс користувача
Опис	Система повинна мати зручний та інтуїтивно зрозумілий інтерфейс користувача, який дозволяє користувачам легко навігувати та взаємодіяти з функціями застосунку. Інтерфейс має бути адаптивним до різних пристроїв, включаючи комп'ютери, планшети та смартфони.

Таблиця 1.7.11 – FR-10

Назва	Аналітика
Опис	Система повинна надавати користувачам можливість аналізу фінансових даних за допомогою різноманітних інструментів та звітів. Це включає в себе відображення статистики доходів та витрат, прогнозування майбутніх фінансових показників та інші аналітичні функції.

Таблиця 1.7.12 – FR-11

Назва	Управління балансами
Опис	Система повинна надавати можливість користувачам відстежувати та управляти своїм фінансовим балансом, включаючи рівень доходів, витрат, заощаджень та інвестицій. Користувачі зможуть переглядати свій фінансовий стан в реальному часі та приймати рішення на основі цієї інформації.

Таблиця 1.7.13 – Матриця трасування вимог

Вимога	UC-01	UC-02	UC-03	UC-04	UC-05	UC-06	UC-07	UC-08	UC-09	UC-10	UC-11	UC-12	UC-13	UC-14
FR-01	Так	Так	Так	Ні	Ні	Ні	Ні	Ні	Ні	Ні	Ні	Ні	Ні	Ні
FR-02	Ні	Так	Ні	Ні	Ні	Ні	Ні	Ні	Ні	Ні	Ні	Ні	Ні	Ні
FR-03	Ні	Ні	Так	Ні	Ні	Ні	Ні	Ні	Ні	Ні	Ні	Ні	Ні	Ні
FR-04	Ні	Ні	Ні	Так	Ні	Ні	Ні	Ні	Ні	Ні	Ні	Ні	Ні	Ні
FR-05	Ні	Ні	Ні	Ні	Так	Ні	Ні	Ні	Ні	Ні	Ні	Ні	Ні	Ні
FR-06	Ні	Ні	Ні	Ні	Ні	Так	Ні	Ні	Ні	Ні	Ні	Ні	Ні	Ні
FR-07	Ні	Ні	Ні	Ні	Ні	Ні	Так	Ні	Ні	Ні	Ні	Ні	Ні	Ні
FR-08	Ні	Ні	Ні	Ні	Ні	Ні	Ні	Так	Ні	Ні	Ні	Ні	Ні	Ні
FR-09	Ні	Ні	Ні	Ні	Ні	Ні	Ні	Ні	Так	Ні	Ні	Ні	Ні	Ні
FR-10	Ні	Ні	Ні	Ні	Ні	Ні	Ні	Ні	Ні	Так	Так	Ні	Ні	Ні
FR-11	Ні	Ні	Ні	Ні	Ні	Ні	Ні	Ні	Ні	Так	Так	Так	Так	Ні

1.8 Розроблення нефункціональних вимог

До веб-застосунку є такі нефункціональні вимоги:

– Веб-браузер версії 110 або вище;
– Мова застосунку – українська;
– Доступ до мережі Internet;
– Використання сучасних інструментів розробки;
– Безпечне збереження конфідеційних даних;
– Простий та зрозумілий користувацьких інтерфейс.

1.9 Постановка задачі

Метою дипломного проєкту є розробка та впровадження комплексної системи для автоматизації фінансового обліку та управління даними. Цільовий застосунок призначений для фізичних осіб, які мають на меті оптимізувати процеси управління фінансами та забезпечити надійний облік фінансових операцій. Завдання проєкту включають:

- Розробка зручного для користувачів формату для опису фінансових операцій та їхніх атрибутів;
- Реалізація механізмів для створення безпечних та ізольованих середовищ для обробки фінансових даних;
- Впровадження механізму для обліку фінансових транзакцій;
- Розробка системи моніторингу та аналізу фінансових даних для забезпечення контролю та оптимізації фінансових процесів;
- Надання можливості налаштування параметрів системи управління фінансами з урахуванням потреб користувачів та вимог бізнесу;
- Реалізація системи управління доступом до фінансових даних з використанням різноманітних ролей та прав доступу для забезпечення безпеки та конфіденційності інформації.

Висновки до розділу

У цьому розділі було проведено детальний аналіз вимог до системи, включаючи функціональні та нефункціональні вимоги. Було визначено ключові характеристики програмного забезпечення.

Були сформовані основні Use cases та Functional Requirements, які значно полегшать, пришвидшать та полегшать процес розробки веб-застосунку.

Також були розглянуті вимоги до бази даних, інтерфейсу користувача, безпеки та продуктивності. На основі аналізу були сформульовані відповідні вимоги, які будуть враховані під час подальшої розробки системи.

Загалом, проведений аналіз вимог дозволить створити ефективну та функціональну систему, що задовольнить потреби користувачів та відповідатиме сучасним стандартам розробки програмного забезпечення.

За результатами розділу сформовано технічне завдання на розробку програмного забезпечення.

2 КОНСТРУЮВАННЯ ТА РОЗРОБЛЕННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

2.1 Архітектура програмного забезпечення

Для програмного забезпечення буде використана архітектура типу "клієнт-сервер", яка базується на мікросервісній архітектурі. У цій архітектурі клієнтська частина представлена застосунком React, що забезпечує інтерактивний та зручний інтерфейс для користувачів. Серверна частина реалізована з використанням Java Spring Boot, яка надає зручність у розробці, швидкість та надійність в роботі.

Для забезпечення безпеки та управління користувачами буде використано механізм аутентифікації та авторизації, реалізований на основі JWT (JSON Web Tokens). Кожен користувач матиме унікальний токен, який буде використовуватися для доступу до ресурсів системи.

Щодо інтеграції з зовнішніми системами, передбачається використання протоколу HTTP або REST API для обміну даними між різними компонентами програмного забезпечення.

Для забезпечення виконання нефункціональних вимог, таких як продуктивність та масштабованість, планується використання хмарних послуг AWS (Amazon Web Services). AWS надасть можливість масштабувати інфраструктуру програмного забезпечення з урахуванням збільшення обсягу користувачів та обробки даних.

В якості бази даних для системи буде використана PostgreSQL. Ця БД забезпечує надійність, стабільність та можливість розширення, що є важливими аспектами для успішного функціонування програмного забезпечення.

Для полегшення розгортання та керування середовищем розробки та виробництва планується використання Docker. Docker дозволить упакувати

програмне забезпечення та всі його залежності в контейнери, що забезпечить консистентність середовища від розробки до виробництва.

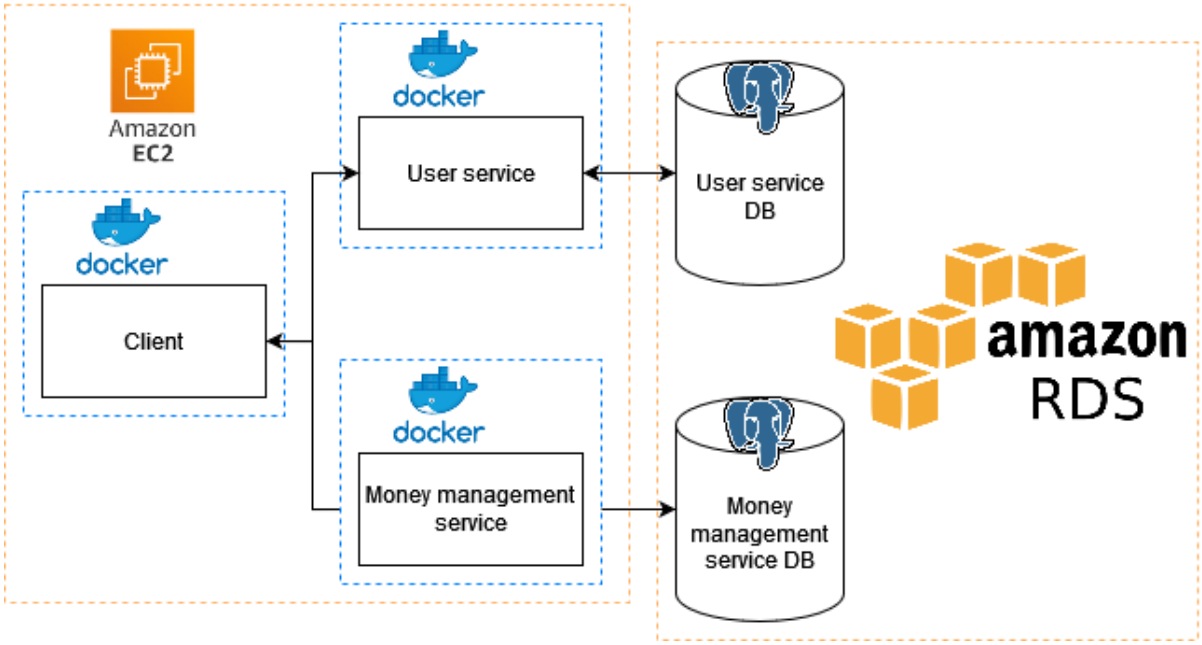


Рисунок 3.1 – Архітектура веб-застосунку

2.2 Обґрунтування вибору засобів розробки

Java проти Python проти C#:

Java:

Переваги:

- Java може працювати на будь-якій операційній системі без змін;
- Потужні фреймворки;
- Вбудовані механізми безпеки;
- Велика та активна спільнота розробників.

Python:

Переваги:

- Простота вивчення та використання: Синтаксис Python більш інтуїтивно зрозумілий;

- Широкий спектр бібліотек для швидкої розробки.

C#:

Переваги:

- Сильна підтримка мови програмування .NET;
- Широке використання у веб-розробці та розробці програмного забезпечення під Windows.

Порівняльний аналіз:

Java вибирається через свою крос-платформенність, потужні фреймворки та широку підтримку для веб-розробки.

React проти Angular проти Vue:

React:

Переваги:

- Легко вивчити та використовувати;
- Велика спільнота та розширюваність за рахунок багатьох сторонніх компонентів та бібліотек.

Angular:

Переваги:

- Комплексний фреймворк, який надає готові рішення для багатьох задач;
- Широкий набір інструментів та бібліотек, вбудований в фреймворк.

Vue:

Переваги:

- Простота та легкість вивчення;
- Висока продуктивність та швидкість роботи.

Порівняльний аналіз:

React вибирається через свою простоту вивчення, велику спільноту та широкі можливості для розширення за рахунок багатьох сторонніх бібліотек.

PostgreSQL проти інших баз даних:

PostgreSQL:

Переваги:

- Здатність обробляти великі обсяги даних та високе навантаження;
- Розширені можливості SQL-запитів та транзакцій;
- Висока стабільність та відповідність стандартам ACID.

Інші бази даних (наприклад, MySQL, MongoDB):

Переваги:

- MySQL: Широке використання та підтримка;
- MongoDB: Гнучка схема даних та гарне масштабування.

Порівняльний аналіз:

PostgreSQL вибирається через свою високу продуктивність, розширені можливості SQL та широкий функціонал.

Spring Boot проти інших фреймворків:

Spring Boot:

Переваги:

- Швидкий та ефективний розвиток завдяки автоконфігурації та вбудованому контейнеру сервлетів;
- Широкий спектр модулів та інтеграційних можливостей для реалізації різноманітних бізнес-логік;
- Висока продуктивність та масштабованість завдяки інтеграції з іншими технологіями та інструментами Spring.

Інші фреймворки (наприклад, Django для Python, .NET для C#):

Переваги:

- Django: Висока швидкодія та ефективність розробки веб-додатків на Python;
- .NET: Широкий набір інструментів для розробки під Windows-платформи та велику підтримку мікросервісної архітектури.

Порівняльний аналіз:

Spring Boot вибирається через свою гнучкість, ефективність та великий екосистем Spring, який надає широкі можливості для реалізації будь-яких вимог проекту.

AWS проти інших хмарних платформ:

AWS:

Переваги:

- Широкий спектр послуг та інфраструктурних можливостей для будь-якого типу застосунку або сервісу;
- Глобальна мережа дата-центрів, що забезпечує високу доступність та надійність;
- Розширені можливості для безпеки даних та інфраструктури.

Інші хмарні платформи (наприклад, Google Cloud Platform, Microsoft Azure):

Переваги:

- Google Cloud Platform: Спеціалізовані послуги для аналітики даних та штучного інтелекту;
- Microsoft Azure: Інтеграція з інструментами Microsoft та великі можливості для розробки під Windows-платформу.

Порівняльний аналіз:

AWS обрано через його широкий спектр послуг, глобальну доступність та високий рівень безпеки.

Docker проти інших технологій віртуалізації:

Docker:

Переваги:

- Легкість використання та швидкість розгортання контейнерів;
- Гнучкість та масштабованість додатків за допомогою контейнеризації;
- Широке використання у великих та складних системах, також як і в мікросервісних архітектурах.

Інші технології віртуалізації (наприклад, Kubernetes, VMWare):

Переваги:

- Kubernetes: Управління контейнерами на великих масштабах та в середовищах хмарних платформ;
- VMWare: Віртуалізація на рівні ОС або апаратного рівня з багатофункціональними можливостями.

Порівняльний аналіз:

Docker вибирається через його простоту, швидкість та широке використання у сучасних системах розробки та розгортання.

2.3 Конструювання програмного забезпечення

Було розроблено кілька оригінальних алгоритмів, спрямованих на оптимізацію роботи системи та забезпечення ефективного управління фінансовими даними користувачів. Основні алгоритми включають:

– Алгоритм оптимізації транзакцій: Цей алгоритм дозволяє ефективно керувати потоком транзакцій, враховуючи їх призначення та пріоритети. Він оптимізує час обробки транзакцій та мінімізує можливі ризики виникнення конфліктів;
– Алгоритм управління категоріями: Цей алгоритм дозволяє користувачам зручно та ефективно організовувати свої фінанси за допомогою категорій. Він класифікує транзакції за категоріями та надає зручні інструменти для їх управління;
– Алгоритм управління балансами: Цей алгоритм забезпечує ефективне управління фінансовими балансами користувачів, враховуючи їх поточний стан та здійснені транзакції. Він допомагає користувачам відстежувати та контролювати їх фінансові активи;

– Алгоритм генерації графіків: Цей алгоритм дозволяє автоматично генерувати графіки на основі фінансових даних користувачів. Він аналізує та візуалізує дані, щоб надати користувачам зрозумілу та інформативну інформацію про їх фінансовий стан.

База даних спроектована для ефективного зберігання та управління фінансовою інформацією користувачів. Вона складається з кількох таблиць, кожна з яких відповідає певній групі даних і має свою структуру.

– Таблиця "Користувачі": Ця таблиця містить інформацію про користувачів системи, таку як ідентифікатор, ім'я, електронна адреса та пароль. Вона використовується для автентифікації та авторизації користувачів;
– Таблиця "Транзакції": У цій таблиці зберігається інформація про фінансові транзакції користувачів, такі як сума, дата, категорія, опис тощо. Кожен запис в цій таблиці відображає одну окрему транзакцію;
– Таблиця "Категорії": Вона містить перелік доступних категорій для класифікації транзакцій, наприклад, їжа, транспорт, розваги тощо. Кожна категорія має унікальний ідентифікатор та назву;
– Таблиця "Баланси": У цій таблиці зберігається інформація про баланси різних рахунків користувачів, таких як готівка, кредитні картки, банківські рахунки тощо. Кожен запис включає ідентифікатор рахунку, залишок та інші відомості;
– Таблиця "Бюджети": Вона містить дані про різні бюджети, які встановлюються користувачами для контролю витрат. Кожен бюджет включає ідентифікатор, назву та суму.

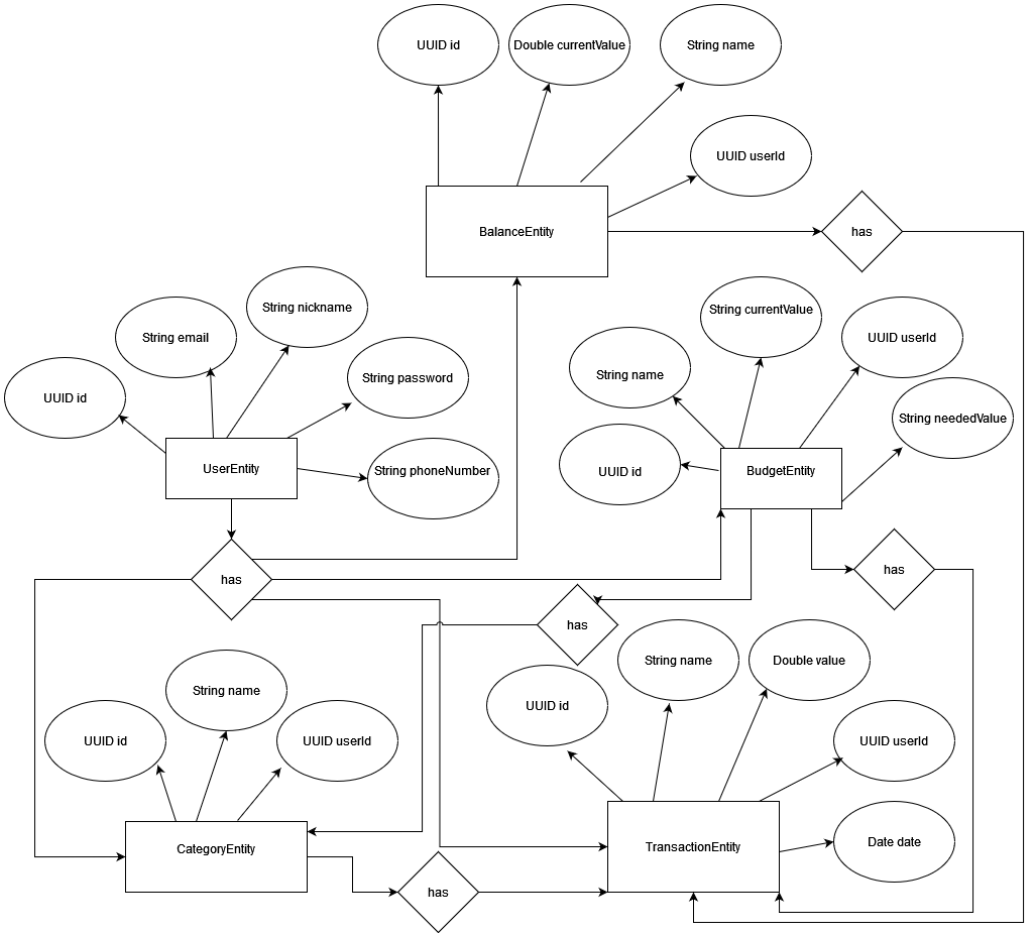


Рисунок 3.2 – ER діаграма сутностей User, Budget, Category, Transaction, Balance

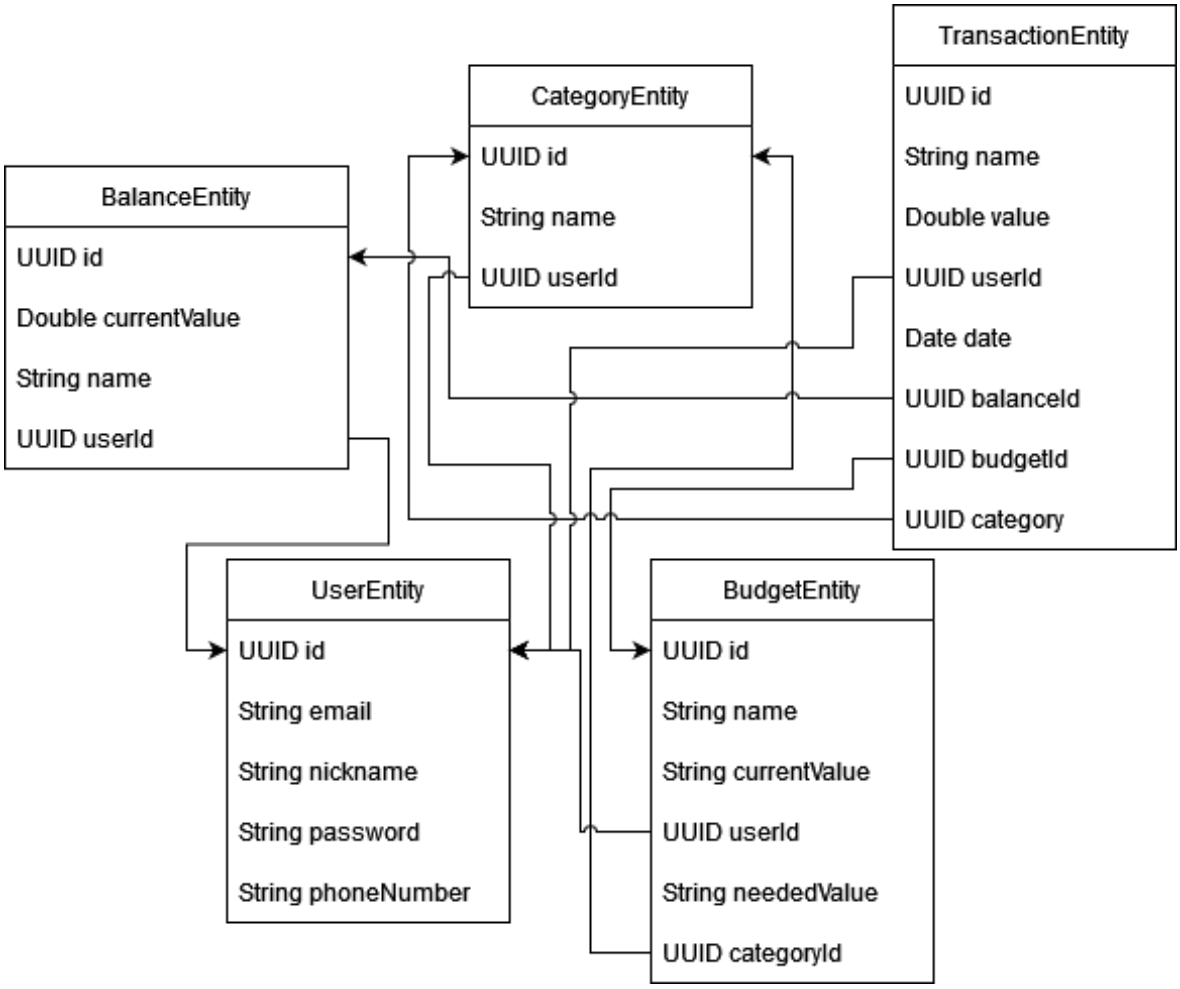


Рисунок 3.3 – Схема структурна класів

В якості фреймфорку для юзер інтерфейсу буде використаний React, який використовується для створення клієнтської частини додатку. React забезпечує розширені можливості створення динамічних та інтерактивних користувацьких інтерфейсів.

В якості фреймворку для розробки веб-додатків на Java буде використаний Spring Boot, який використовується для створення серверної частини нашого додатку. Spring Boot дозволяє швидко створювати робочі застосунки з мінімальною конфігурацією.

Для авторизації буде використано JWT (JSON Web Tokens). JWT дозволяє створювати та перевіряти підписані токени, що дозволяє зберігати стан автентифікації на клієнтському боці.

Chart.js – це бібліотека JavaScript для створення динамічних графіків та діаграм на веб-сторінках. Ми використовуємо Chart.js для відображення фінансової аналітики та статистики у нашому додатку.

В якості бази даних використовується PostgreSQL. Це потужна система управління базами даних, яка використовується для зберігання фінансових даних користувачів. PostgreSQL забезпечує надійне зберігання та оптимальний доступ до даних.

Опис таблиць бази даних наведено у таблицях 3.11 - 3.16. Модель бази даних наведена на рисунку 3.4.

Таблиця 3.11 – Опис таблиці user

Назва поля	Тип даних	Опис
id	UUID	Ідентифікаційний номер користувача
nickname	VARCHAR(255)	Логін користувача
email	VARCHAR(255)	Електронна пошта користувача
password	VARCHAR(255)	Пароль користувача(зашифрований)
phone_number	VARCHAR(255)	Телефоний номер користувача
role	ROLE_TYPE	Роль користувача
enabled	BOOLEAN	Активований аккаунт чи ні

Таблиця 3.12 – Опис таблиці balance

Назва поля	Тип даних	Опис
id	UUID	Ідентифікаційний номер балансу
name	VARCHAR(255)	Ім’я балансу

Продовження таблиці 3.12

user_id	UUID	Ідентифікаційний номер користувача балансу

Таблиця 3.13 – Опис таблиці budget

Назва поля	Тип даних	Опис
id	UUID	Ідентифікаційний номер балансу
name	VARCHAR(255)	Ім’я балансу
current_value	DOUBLE	Поточну використання бюджету
needed_value	DOUBLE	Максимальне використання бюджету
user_id	UUID	Ідентифікаційний номер користувача бюджету
category_id	UUID	Ідентифікаційний номер категорії бюджету

Таблиця 3.14 – Опис таблиці category

Назва поля	Тип даних	Опис
id	UUID	Ідентифікаційний номер балансу
name	VARCHAR(255)	Ім’я категорії
user_id	UUID	Ідентифікаційний номер користувача категорії

Таблиця 3.15 – Опис таблиці transaction

Назва поля	Тип даних	Опис
id	UUID	Ідентифікаційний номер балансу
name	VARCHAR(255)	Ім’я категорії
user_id	UUID	Ідентифікаційний номер користувача категорії
value	DOUBLE	Сума транзакції
balance_id	UUID	Ідентифікаційний номер відповідного балансу
budget_id	UUID	Ідентифікаційний номер відповідного бюджету
category_id	UUID	Ідентифікаційний номер відповідної категорії

Опис утиліт, бібліотек та іншого стороннього програмного забезпечення, що використовується у розробці наведено в таблиці 3.16.

Таблиця 3.16 – Опис утиліт

№ п/п	Назва утиліти	Опис застосування
1	IntelliJ IDEA	Головне середовище розробки програмного забезпечення серверної частини курсової роботи.

Продовження таблиці 3.16

2	Postman	Програмне забезпечення необхідне для тестування rest запитів. Використовувалось для тестування API інтерфейсів, та клієнтських запитів.
3	PgAdmin	Програмне забезпечення яке надає легкий графічний інтерфейс для доступу до бази даних.
	Docker Desktop	Програмне забезпечення яке надає графічний інтерфейс для контейнеризації додатка

Тексти програмного коду наведені в окремому документі «Текст програми».

2.4 Аналіз безпеки даних

Щоб захистити застосунок від атак MITM (man-in-the-middle) та забезпечити цілісність даних і безпечний зв'язок між клієнтом та сервером, ми використовуємо сучасний протокол підключення HTTPS. Цей протокол гарантує шифрування даних та аутентифікацію сервера, що дозволяє уникнути прослуховування та зміни переданих даних.

Усі паролі користувачів зберігаються у захешованому вигляді, щоб уникнути доступу до них у відкритому вигляді. Ми використовуємо бібліотеку для зберігання паролів як результат їх хешування з використанням алгоритму PBKDF2 із HMAC-SHA512 та сіллю. Такий підхід забезпечує безпеку паролів у базі даних.

Крім того, ми встановлюємо обмеження на паролі користувачів, такі як мінімальна довжина пароля, наявність цифр, букв у верхньому регістрі та не буквено-цифрових символів, щоб ускладнити їх перебір атакуючим.

Авторизація користувачів здійснюється за допомогою токенів JWT, що дозволяє нам зробити застосунок stateless та незалежним від сесій. Кожний токен містить інформацію про користувача та штамп безпеки, який змінюється при зміні пароля або адреси електронної пошти користувача. Крім того, токени підписуються ключем, що забезпечує додатковий рівень безпеки та підтверджує автентичність даних.

Висновки до розділу

У розділі "Конструювання програмного забезпечення" ми детально розглянули оригінальні алгоритми та структури даних, що використовуються в нашому застосунку. Ми також надали опис бази даних, включаючи концептуальну, логічну та фізичну моделі, а також сутності та таблиці, що використовуються для збереження даних. Крім того, ми розглянули стороннє програмне забезпечення та бібліотеки для забезпечення безпеки даних та автентифікації.

У розділі "Аналіз безпеки даних" ми обговорили заходи, які вжито для забезпечення безпеки даних у нашому застосунку. Ми використовуємо сучасні протоколи шифрування та аутентифікації, такі як HTTPS та JWT, щоб запобігти атакам MITM та забезпечити цілісність даних. Паролі користувачів зберігаються у захешованому вигляді з використанням безпечного алгоритму хешування. Крім того, ми встановлюємо обмеження на паролі та використовуємо штампи безпеки для додаткового захисту.

Загалом, наші заходи забезпечують високий рівень безпеки даних у нашому застосунку, що дозволяє користувачам впевнено використовувати його для збереження та обробки їх особистих даних.

3 АНАЛІЗ ЯКОСТІ ТА ТЕСТУВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

3.1 Аналіз якості ПЗ

Аналіз якості програмного забезпечення здійснюється за певними метриками, що включають:

– Надійність;
– Продуктивність;
– Зручність використання;
– Функціональність;
– Сумісність;
– Безпека;
– Мобільність;
– Підтримка та обслуговування.

Надійність – це відсоток безвідмовної роботи системи за певний період часу та кількість критичних помилок, що впливають на роботу системи.

Продуктивність – це час відгуку системи на користувацькі запити, швидкість обробки транзакцій та генерації аналітики.

Зручність використання – це рівень задоволеності користувачів інтерфейсом, кількість кліків для виконання основних операцій та наявність інструкцій та підказок.

Функціональність – це відповідність функціональних можливостей заявленим вимогам та повнота реалізації основних та додаткових функцій.

Сумісність – це можливість роботи на різних платформах та інтеграція з іншими програмними продуктами та сервісами.

Безпека – це захищеність даних користувачів від несанкціонованого доступу та наявність механізмів аутентифікації та авторизації користувачів.

Мобільність – це можливість доступу до системи з мобільних пристроїв та адаптивність інтерфейсу для різних розмірів екранів.

Підтримка та обслуговування – це наявність документованих інструкцій для користувачів та доступність технічної підтримки та оновлень

Метою тестування є наступне:

– Виявлення дефектів – знаходження та усунення помилок в програмному забезпеченні, які можуть вплинути на його функціональність, продуктивність та безпеку;

– Перевірка відповідності вимогам – переконання, що програмне забезпечення відповідає усім функціональним та нефункціональним вимогам, визначеним у технічному завданні;

– Забезпечення якості – підтвердження, що програмне забезпечення є надійним, зручним у використанні та безпечним;

– Підтримка стабільності – підтвердження, що нові функції та виправлення не призводять до виникнення нових проблем і не погіршують продуктивність чи надійність системи;

– Оцінка продуктивності – виміряння часу відгуку, швидкості обробки даних;

– Перевірка сумісності – підтвердження, що програмне забезпечення працює належним чином на всіх платформах і сумісне з іншими системами та сервісами;

– Підтвердження безпеки – підтвердження, що програмне забезпечення забезпечує високий рівень захисту даних користувачів;

– Оцінка зручності – перевірка, що інтерфейс користувача є інтуїтивним та зручним.

Метриками для оцінки якості ПЗ обрано наступні:

– MTBF (Mean Time Between Failures): Середній час між відмовами системи;

– Error Rate: Частота виникнення помилок під час роботи системи;

– Response Time: Час відгуку системи на користувацькі запити;

- Throughput: Кількість транзакцій або операцій, виконаних за одиницю часу;
- User Satisfaction Score: Рівень задоволеності користувачів, зазвичай вимірюється за допомогою опитувань;
- Task Completion Rate: Відсоток успішно виконаних користувачами завдань;
- Requirement Coverage: Відсоток реалізованих функцій від загальної кількості вимог;
- Defect Density: Кількість дефектів на одиницю коду (наприклад, на 1000 рядків коду);
- Platform Compatibility: Відсоток підтримуваних платформ, на яких програмне забезпечення працює без проблем;
- Integration Success Rate: Відсоток успішних інтеграцій з іншими системами або сервісами;
- Vulnerability Count: Кількість виявлених вразливостей безпеки;
- Incident Response Time: Час, необхідний для реагування на інциденти безпеки;
- Mobile Compatibility Score: Оцінка адаптивності інтерфейсу для мобільних пристроїв;
- Access Speed on Mobile: Час відгуку системи при доступі з мобільних пристроїв;
- Documentation Coverage: Відсоток покриття функцій документацією.

3.2 Опис процесів тестування

Тестування виконується згідно документу «Програма та методика тестування».

Було виконане мануальне тестування програмного забезпечення, опис відповідних тестів наведено у таблицях 4.3 – 4.15.

Таблиця 4.3 – Тест 1.1 Реєстрація користувача

Початковий стан системи	Користувач знаходиться на сторінці реєстрації
Вхідні дані	Електронна пошта, пароль, підтвердження паролю
Опис проведення тесту	У відповідні поля вводяться: коректна електронна пошта, яка до цього не була зареєстрована в системі, пароль від 10 до 64 символів, який містить хоча б з одну англійську літеру, одне число і один спеціальний символ, і який не входить у топ 10000 найпопулярніших паролей, підтвердження паролю, яке співпадає з раніше введеним паролем. Після цього натискається кнопка підтвердження реєстрації.
Очікуваний результат	Реєстрація проходить успішно, користувач додається у систему і перенаправляється на сторінку авторизації.
Фактичний результат	Збігається з очікуваним.

Таблиця 4.4 – Тест 1.2 Вхід користувача

Початковий стан системи	Користувач знаходиться на сторінці авторизації
Вхідні дані	Електронна пошта, пароль
Опис проведення тесту	У відповідні поля вводяться: коректна електронна пошта, зареєстрована в системі, і відповідний пароль. Натискається кнопка входу.
Очікуваний результат	Користувач успішно авторизується і перенаправляється на головну сторінку (Dashboard).
Фактичний результат	Збігається з очікуваним.

Таблиця 4.5 – Тест 2.1 Додавання нової транзакції

Початковий стан системи	Користувач авторизований і знаходиться на сторінці додавання транзакції
Вхідні дані	Сума, дата, категорія, опис транзакції
Опис проведення тесту	Вводяться коректні дані в усі обов’язкові поля (сума, дата, категорія, опис). Натискається кнопка додавання транзакції.
Очікуваний результат	Транзакція успішно додається, відображається у списку транзакцій на головній сторінці.
Фактичний результат	Збігається з очікуваним.

Таблиця 4.6 – Тест 2.2 Редагування існуючої транзакції

Початковий стан системи	Користувач авторизований і знаходиться на головній сторінці (Dashboard)
Вхідні дані	Оновлені дані транзакції (сума, дата, категорія, опис)
Опис проведення тесту	Вибирається існуюча транзакція для редагування. Вводяться оновлені дані в усі обов’язкові поля. Натискається кнопка збереження змін.
Очікуваний результат	Транзакція успішно оновлюється, зміни відображаються у списку транзакцій на головній сторінці.
Фактичний результат	Збігається з очікуваним.

Таблиця 4.7 – Тест 2.3 Видалення транзакції

Початковий стан системи	Користувач авторизований і знаходиться на головній сторінці (Dashboard)

Продовження таблиці 4.7

Вхідні дані	Немає
Опис проведення тесту	Вибирається існуюча транзакція для видалення. Натискається кнопка видалення і підтверджується дія у вікні підтвердження.
Очікуваний результат	Транзакція успішно видаляється зі списку.
Фактичний результат	Збігається з очікуваним.

Таблиця 4.8 – Тест 3.1 Створення нового бюджету

Початковий стан системи	Користувач авторизований і знаходиться на сторінці планування бюджету
Вхідні дані	Категорія, сума бюджету, період
Опис проведення тесту	Вводяться коректні дані в усі обов’язкові поля (категорія, сума, період). Натискається кнопка створення бюджету.
Очікуваний результат	Бюджет успішно створюється і відображається у списку бюджетів.
Фактичний результат	Збігається з очікуваним.

Таблиця 4.9 – Тест 3.2 Оновлення існуючого бюджету

Початковий стан системи	Користувач авторизований і знаходиться на сторінці планування бюджету
Вхідні дані	Оновлені дані бюджету (категорія, сума, період)

Продовження таблиці 4.9

Опис проведення тесту	Вибирається існуючий бюджет для редагування. Вводяться оновлені дані в усі обов’язкові поля. Натискається кнопка збереження змін.
Очікуваний результат	Бюджет успішно оновлюється, зміни відображаються у списку бюджетів.
Фактичний результат	Збігається з очікуваним.

Таблиця 4.10 – Тест 3.3 Видалення бюджету

Початковий стан системи	Користувач авторизований і знаходиться на сторінці планування бюджету
Вхідні дані	Немає
Опис проведення тесту	Вибирається існуючий бюджет для видалення. Натискається кнопка видалення і підтверджується дія у вікні підтвердження.
Очікуваний результат	Бюджет успішно видаляється зі списку.
Фактичний результат	Збігається з очікуваним.

Таблиця 4.11 – Тест 4.1 Генерація фінансового звіту

Початковий стан системи	Користувач авторизований і знаходиться на сторінці звітів
Вхідні дані	Параметри звіту (період, категорія)
Опис проведення тесту	Вибираються параметри звіту (період, категорія) і натискається кнопка генерації звіту.

Продовження таблиці 4.11

Очікуваний результат	Звіт успішно генерується і відображається на екрані.
Фактичний результат	Збігається з очікуваним.

Таблиця 4.12 – Тест 5.1 Налаштування профілю користувача

Початковий стан системи	Користувач авторизований і знаходиться на сторінці налаштувань профілю
Вхідні дані	Оновлені дані профілю (ім'я, email, пароль)
Опис проведення тесту	Вводяться оновлені дані в усі обов’язкові поля (ім'я, email, новий пароль). Натискається кнопка збереження змін.
Очікуваний результат	Профіль успішно оновлюється, зміни зберігаються в системі.
Фактичний результат	Збігається з очікуваним.

Таблиця 4.13 – Тест 6.1 Відновлення паролю

Початковий стан системи	Користувач знаходиться на сторінці відновлення паролю
Вхідні дані	Електронна пошта, новий пароль, підтвердження паролю
Опис проведення тесту	У відповідні поля вводяться: електронна пошта, новий пароль, підтвердження паролю. Натискається кнопка відновлення паролю.
Очікуваний результат	Користувач отримує електронний лист з посиланням для відновлення паролю, після чого успішно встановлює новий пароль.

Продовження таблиці 4.13

Фактичний результат	Збігається з очікуваним.

Таблиця 4.14 – Тест 7.1 Налаштування сповіщень

Початковий стан системи	Користувач авторизований і знаходиться на сторінці налаштувань сповіщень
Вхідні дані	Параметри сповіщень (тип сповіщення, частота)
Опис проведення тесту	Вибираються параметри сповіщень (тип сповіщення, частота) і натискається кнопка збереження змін.
Очікуваний результат	Сповіщення успішно налаштовуються відповідно до вибраних параметрів.
Фактичний результат	Збігається з очікуваним.

Таблиця 4.15 – Тест 8.1 Перевірка сумісності на різних платформах

Початковий стан системи	Користувач авторизований, тестується на різних платформах (Windows, Unix)
Вхідні дані	Немає
Опис проведення тесту	Перевірка виконання основних функцій (реєстрація, вхід, додавання транзакцій тощо) на різних платформах.
Очікуваний результат	Система успішно працює на всіх підтримуваних платформах без втрати функціональності.
Фактичний результат	Збігається з очікуваним.

3.3 Опис контрольного прикладу

Контрольний приклад: Управління місячним бюджетом

Початковий стан системи:

Користувач авторизований у системі та знаходиться на головній сторінці (Dashboard).

Етапи виконання:

- Додавання нового доходу:
  - Користувач натискає кнопку "Додати дохід";
  - У відповідні поля вводяться дані про дохід: сума (5000 грн), дата (01.05.2024), категорія (Заробітна плата), опис (Заробітна плата за травень);
  - Користувач натискає кнопку підтвердження;
  - Очікуваний результат: Дохід успішно додається до системи, відображається у списку транзакцій та оновлюється баланс;
- Додавання витрати:
  - Користувач натискає кнопку "Додати витрату";
  - У відповідні поля вводяться дані про витрату: сума (200 грн), дата (02.05.2024), категорія (Продукти), опис (Покупка продуктів).
  - Користувач натискає кнопку підтвердження;
  - Очікуваний результат: Витрата успішно додається до системи, відображається у списку транзакцій та оновлюється баланс.
- Планування бюджету:
  - Користувач переходить на сторінку планування бюджету;
  - Вибирається категорія "Продукти", вводиться сума бюджету (3000 грн) на період (травень);
  - Користувач натискає кнопку створення бюджету;

- Очікуваний результат: Бюджет успішно створюється, відображається у списку бюджетів.

– Редагування існуючої транзакції:
  - Користувач вибирає транзакцію з витратою (200 грн на продукти) для редагування;
  - Вносить зміни у суму (300 грн) та натискає кнопку збереження;
  - Очікуваний результат: Транзакція успішно оновлюється, нові дані відображаються у списку транзакцій.
– Генерація фінансового звіту:
  - Користувач переходить на сторінку звітів;
  - Вибирає параметри звіту: період (травень), категорія (усі категорії);
  - Натискає кнопку генерації звіту;
  - Очікуваний результат: Звіт успішно генерується, містить усі доходи та витрати за вибраний період, відображає залишок та перевитрату бюджету.

Висновки до розділу

Проведено мануальне тестування програмного забезпечення, результати якого відображають відповідність фактичних результатів очікуваним. Тести охоплювали різні аспекти функціональності, включаючи реєстрацію, вхід, додавання та управління транзакціями, планування бюджету, генерацію звітів та налаштування сповіщень. Усі функції працювали коректно.

Опис контрольного прикладу демонструє реальний сценарій використання системи, що підтверджує її здатність ефективно підтримувати користувачів у процесі управління фінансами. Контрольний приклад також підтвердив, що програмне забезпечення забезпечує необхідну

функціональність та задовольняє вимоги користувачів щодо надійності, зручності та безпеки.

Таким чином, проведений аналіз та тестування підтвердили, що розроблене програмне забезпечення відповідає заявленим вимогам та готове до впровадження. Подальші покращення можуть бути спрямовані на оптимізацію продуктивності та розширення функціональних можливостей відповідно до відгуків користувачів.

4 ВПРОВАДЖЕННЯ ТА СУПРОВІД ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

4.1 Розгортання програмного забезпечення

1) Створюємо EC2 instance та налаштовуємо

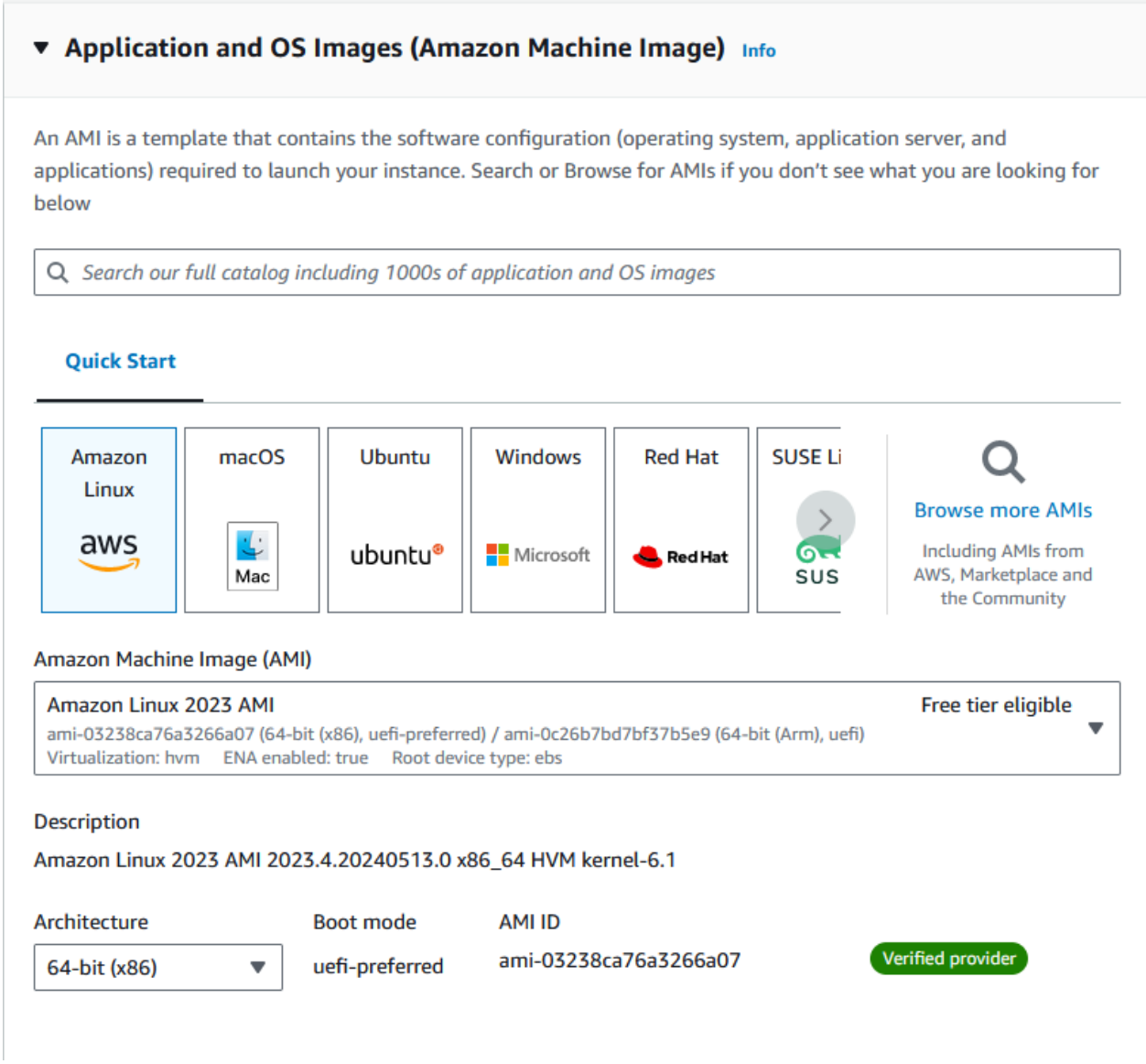


Рисунок 5.1 – Вибір ОС

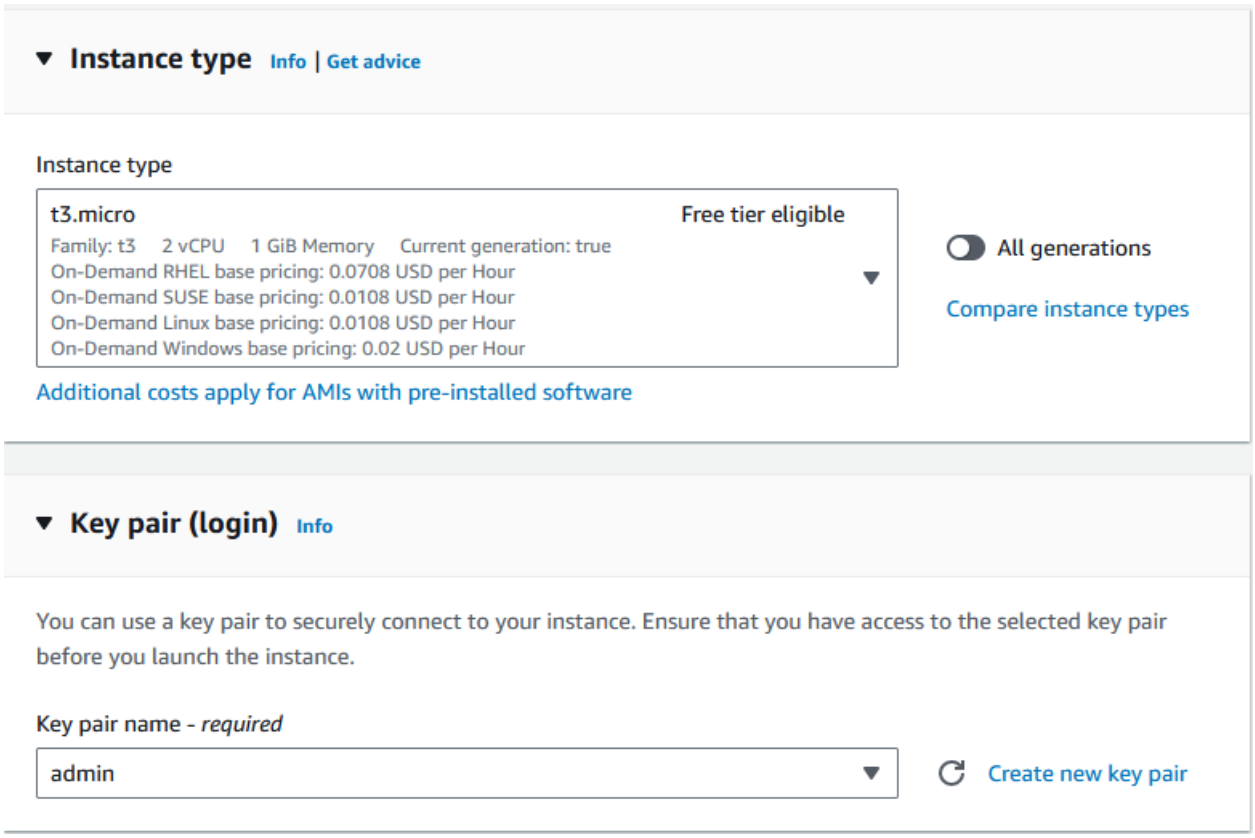


Рисунок 5.2 – Налаштування типу та ключа для логіну на EC2

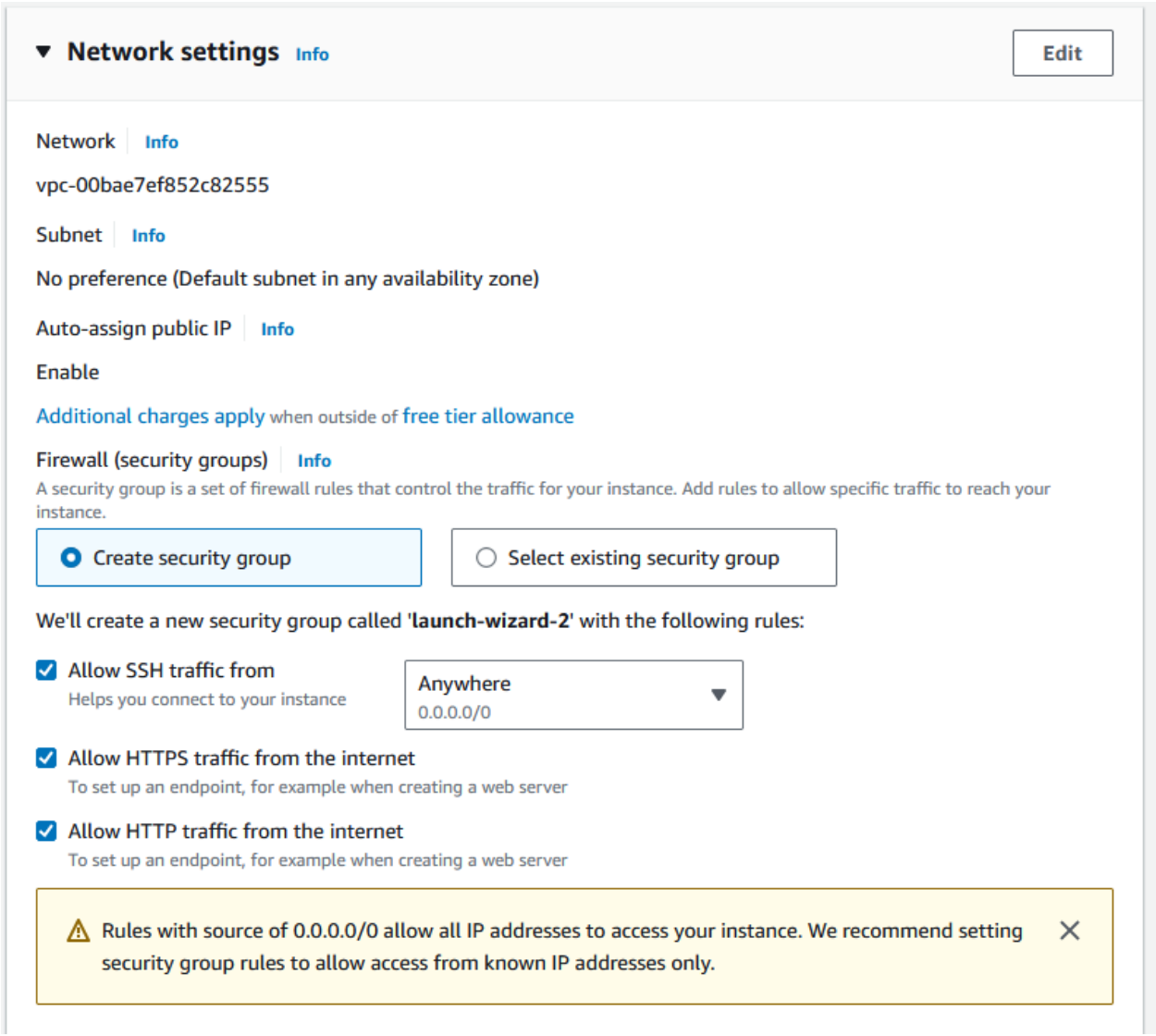


Рисунок 5.3 – Налаштування Network

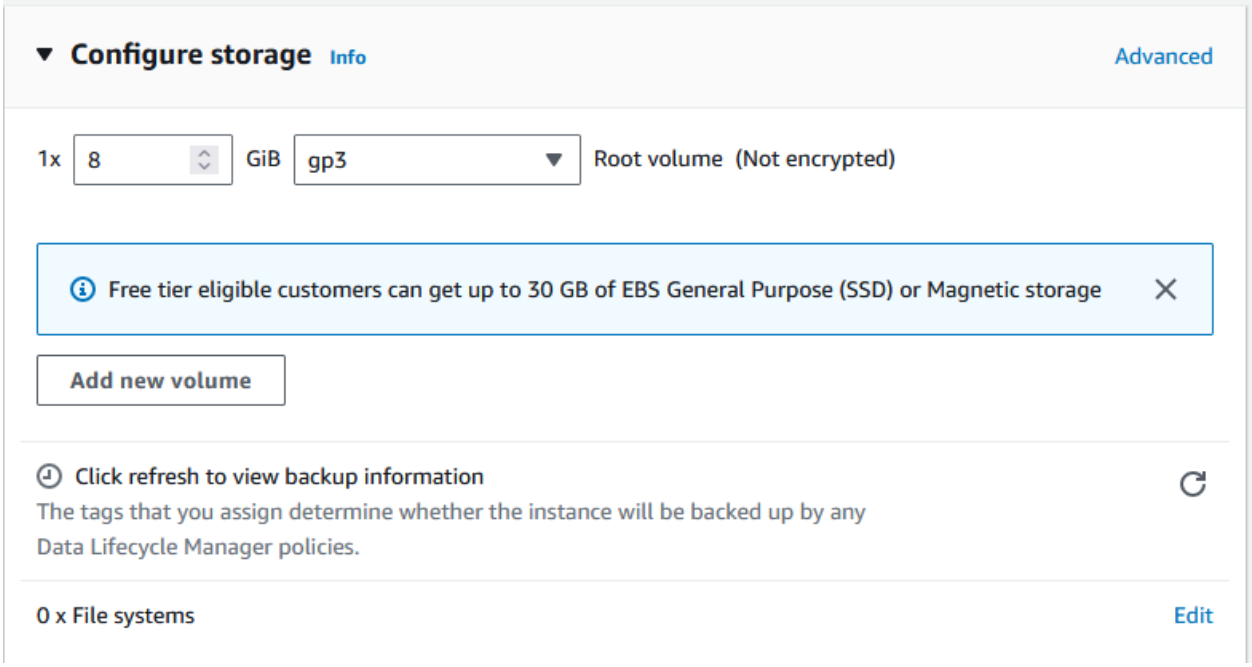


Рисунок 5.4 – Налаштування стореджу

2) Архівуємо проект та переміщаємо його на хмарне середовище

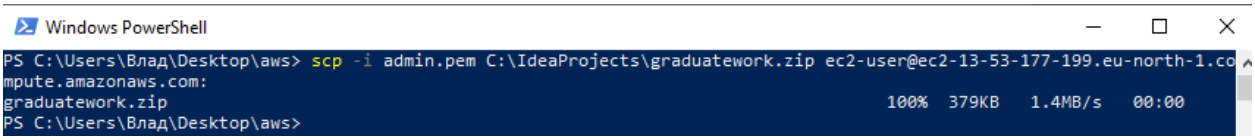


Рисунок 5.5 – Перенесення проекту на EC2

3) Підключаємось до інстансу через SSH

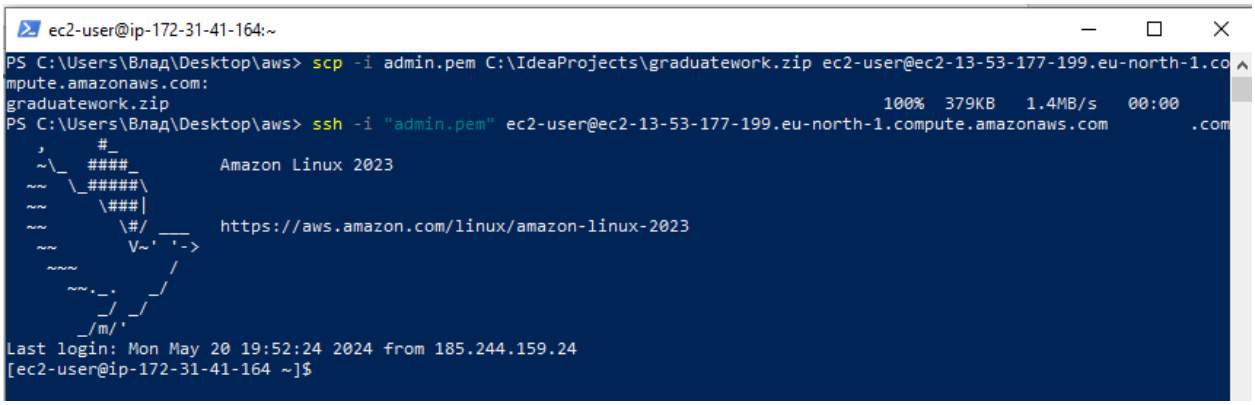


Рисунок 5.6 – Підключення до EC2

4) Розархівовуємо проект

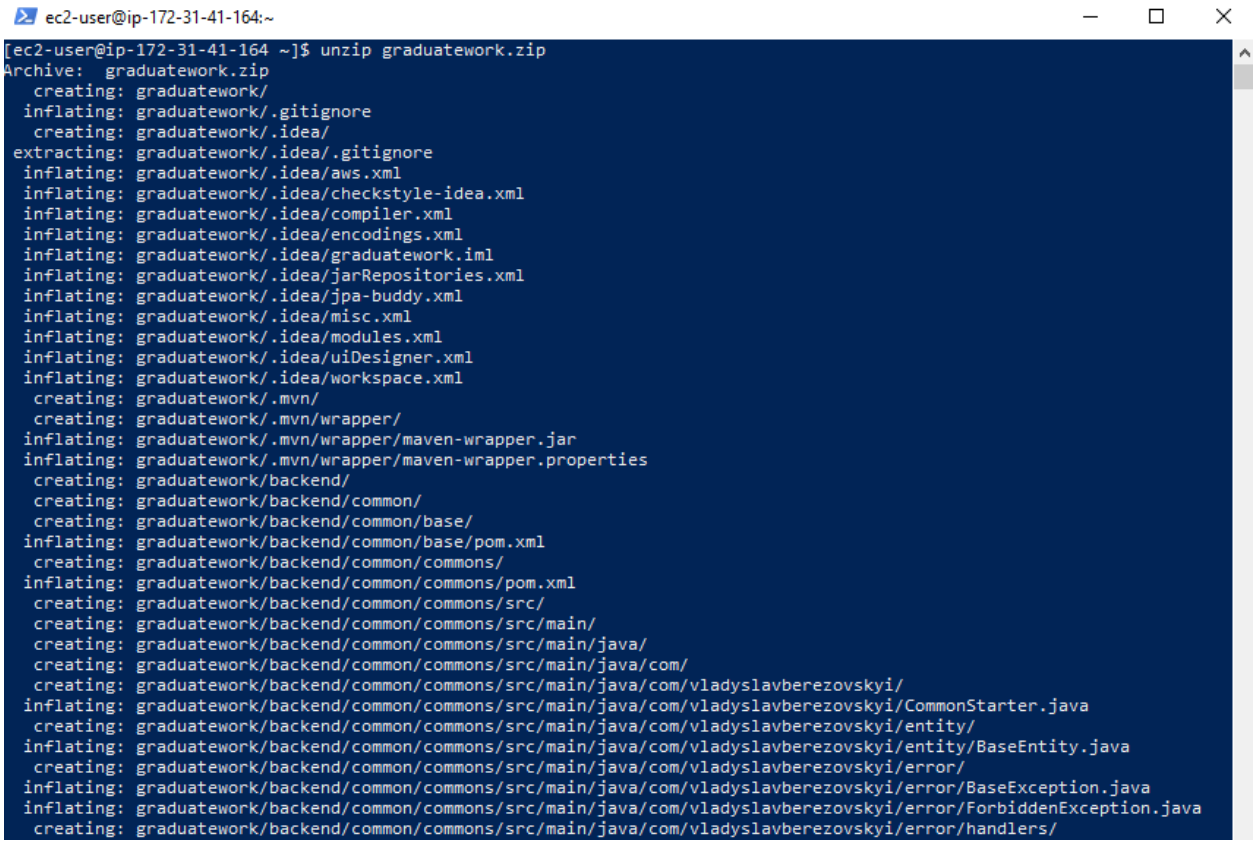


Рисунок 5.7 – Розархівування проекту

5) Підняття докер контейнерів

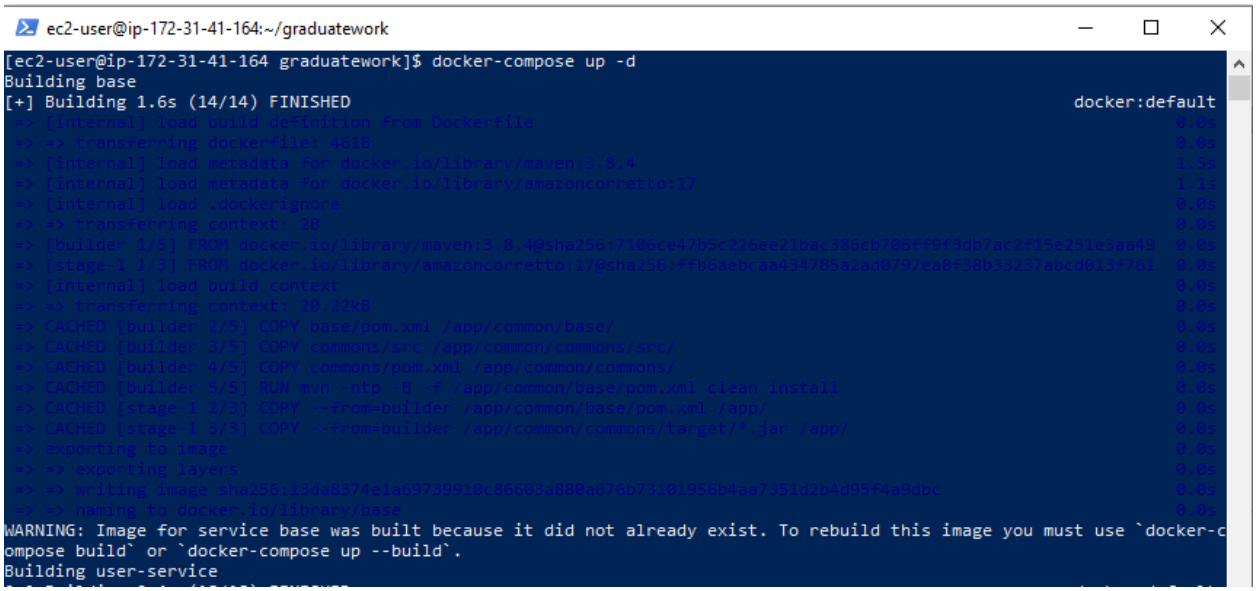


Рисунок 5.8 – Старт процесу підняття контейнерів

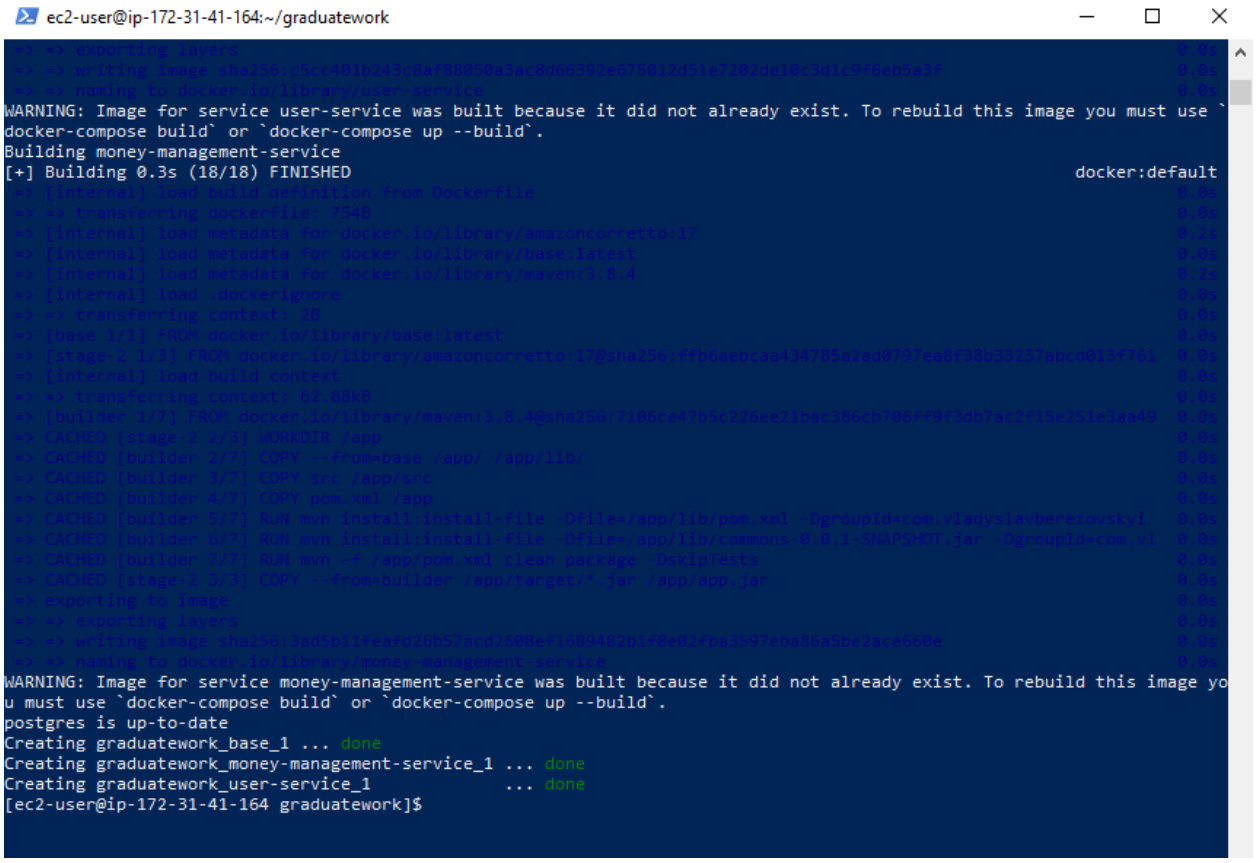


Рисунок 5.9 – Контейнери підняті

6) Проект піднятий до нього можна доступитися за посиланням

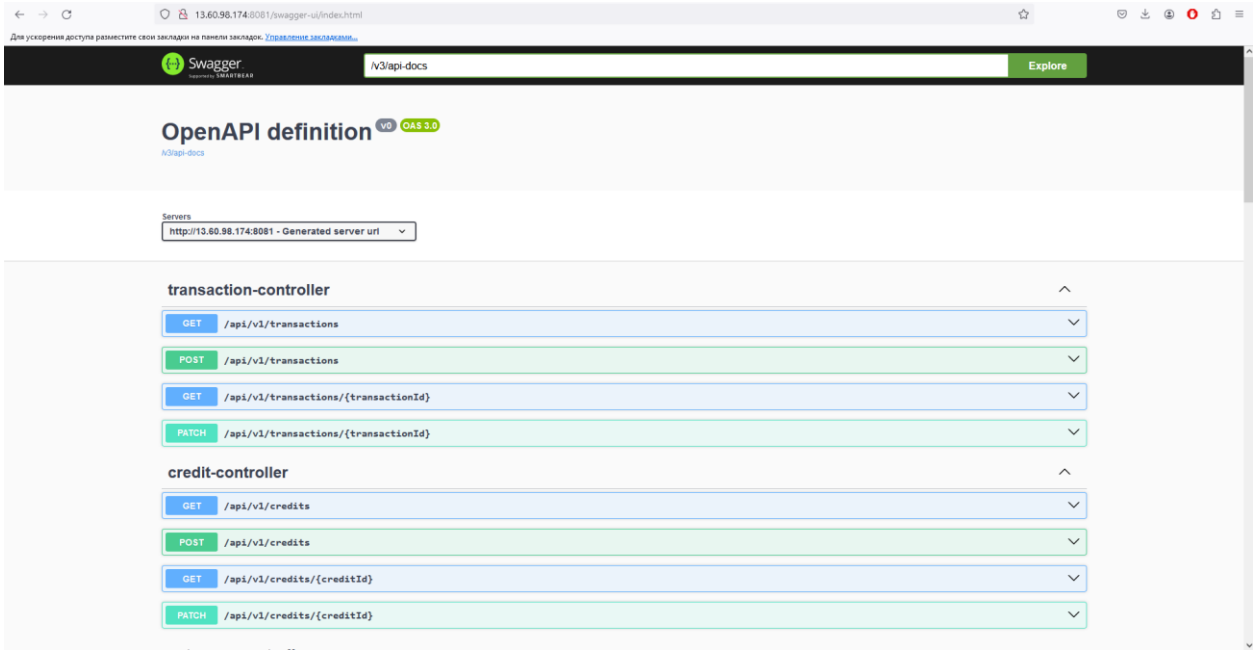


Рисунок 5.10 – Swagger піднятого проекту

4.2 Супровід програмного забезпечення

Створення нового випуску починається, коли нова версія консольного застосунку доставляється у репозиторій у гілку main, тобто коли commit має tag формату “v.*”, де замість “*” знаходиться число. Тоді у середовищі GitHub Actions встановлюється Java. Після цього для проекту встановлюються залежності і проект збирається. Bash скрипт за допомогою бібліотеки mvn генерує виконувані файли (.jar). Після цього проект розгортається в докері на хмарному середовищі.

Нова версія додатку при цьому буде доставлена кінцевим користувачам миттєво та автоматично. Тобто користувачу не потрібно нічого робити, щоб отримати саму останню версію додатку.

Висновки до розділу

В результаті виконання завдань по розгортанню проекту на платформі Amazon Web Services (AWS), вдалося досягти важливих результатів, які значно покращують функціональність та доступність програмного забезпечення.

Процес розгортання включав створення облікового запису AWS та налаштування необхідних прав доступу. Використовуючи Amazon EC2, я створив віртуальну машину для розміщення програмного забезпечення, обрав відповідний образ операційної системи та налаштував параметри, такі як тип інстансу та з'єднання через SSH. База даних була створена за допомогою Amazon RDS, що забезпечило надійне зберігання та обробку даних.

Особлива увага була приділена налаштуванню груп безпеки для захисту інфраструктури та забезпечення належного доступу. Завантаження програмного забезпечення на віртуальну машину було виконано через SSH, після чого були налаштовані необхідні сервіси, такі як веб-сервери.

Після завершення налаштувань, я сконфігурував DNS-записи та забезпечив HTTPS-з'єднання, що гарантує безпечний доступ до додатку через Інтернет. Виконані тестування та налаштування дозволили переконатися в стабільній роботі програмного забезпечення.

Розгортання проекту на AWS значно підвищило його масштабованість та доступність, дозволяючи користувачам безпечно та зручно користуватися додатком. Досягнуті результати відповідають сучасним стандартам та практикам в сфері розробки та розгортання програмного забезпечення, що підтверджує успішність обраного підходу.

ВИСНОВКИ

Протягом роботи над дипломним проєктом був розроблений веб-застосунок для обліку особистих фінансів. Цей веб-застосунок спрощує процес ведення обліку фінансів на різних рахунках, дозволяючи користувачам швидко отримувати інформацію про свої фінанси.

Під час розробки був проведений аналіз існуючих рішень, щоб визначити, що їм не вистачає, та які функції слід реалізувати у дипломному проєкті. Були сформульовані загальні задачі, функціональні та нефункціональні вимоги до системи.

Була спроектована архітектура програмного забезпечення та бази даних, з особливою увагою до безпеки збереження персональних даних користувача. Для розробки були використані сучасні технології, такі як мова програмування Java з фреймворком Spring Boot, React та PostgreSQL.

Для забезпечення якості продукту було проведено мануальне тестування та тестування з використанням Unit тестів. Також були використані статичні засоби аналізу коду для підтримки кодової бази у належному стані.

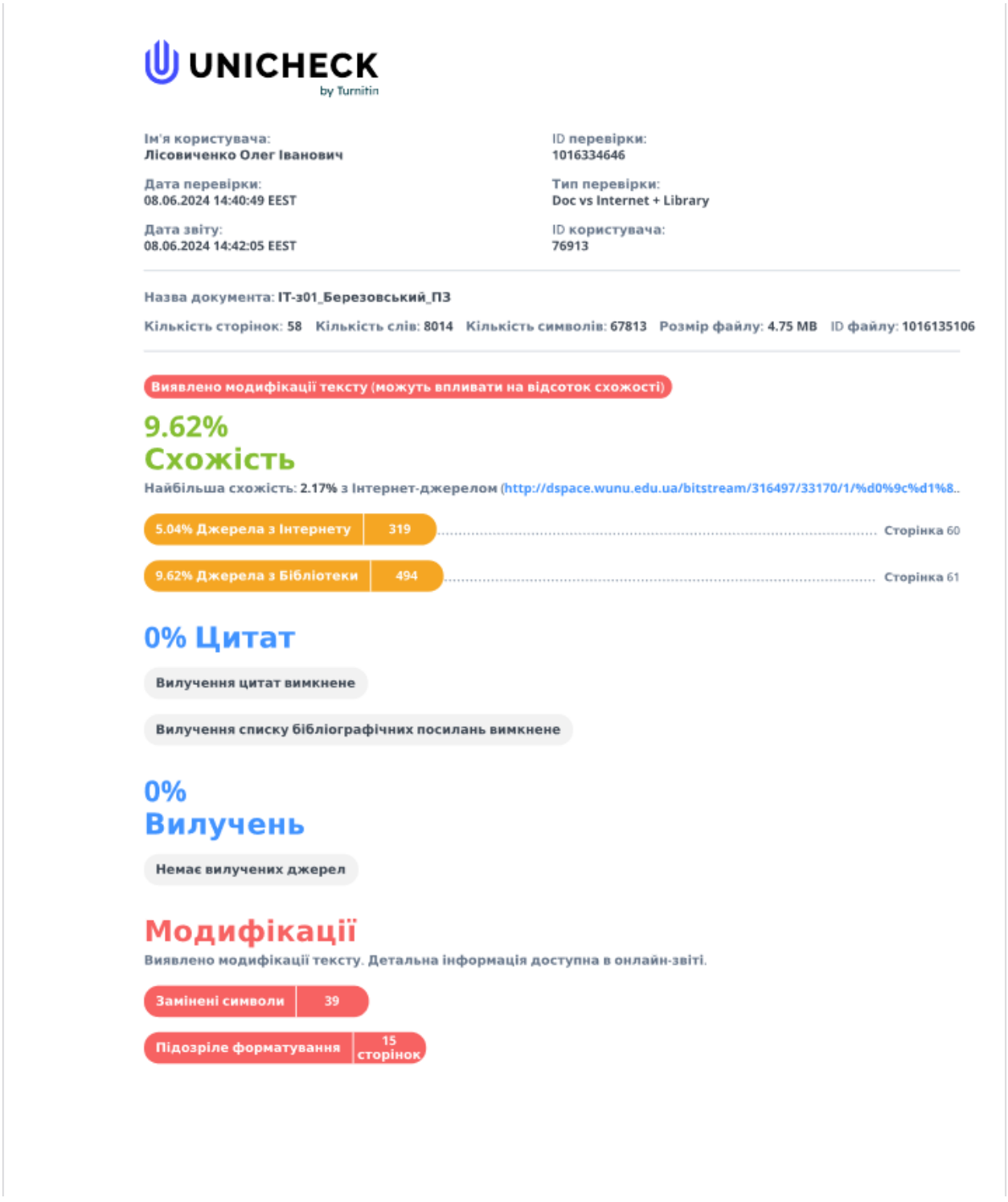
Останнім етапом було впровадження програмного забезпечення на сервісі AWS для забезпечення доступності для користувачів. У майбутньому планується введення монетизації та автоматизація процесу розгортання додатку за допомогою CI/CD процесів. Ці кроки допоможуть розширити клієнтську базу та підвищити конкурентоспроможність продукту.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Sommerville, I. (2011). Software Engineering (9th ed.). Boston: Addison-Wesley.
2. Pressman, R. S., & Maxim, B. R. (2014). Software Engineering: A Practitioner's Approach (8th ed.). McGraw-Hill Education.
3. Cohn, M. (2004). User Stories Applied: For Agile Software Development. Addison-Wesley Professional.
4. Wiegers, K., & Beatty, J. (2013). Software Requirements (3rd ed.). Microsoft Press.
5. Glinz, M. (2007). On Non-Functional Requirements. In 15th IEEE International Requirements Engineering Conference (pp. 21-26). IEEE.
6. Gilb, T. (2005). Competitive Engineering: A Handbook for Systems Engineering, Requirements Engineering, and Software Engineering Using Planguage. Butterworth-Heinemann.
7. Cockburn, A. (2000). Writing Effective Use Cases. Addison-Wesley Professional.
8. Boehm, B. W., & Basili, V. R. (2001). Software Defect Reduction Top 10 List. IEEE Computer, 34(1), 135-137.
9. Nielsen, J. (1993). Usability Engineering. Academic Press.
10. Brooks, F. P. (1995). The Mythical Man-Month: Essays on Software Engineering (Anniversary ed.). Addison-Wesley Professional.
11. McConnell, S. (2004). Code Complete: A Practical Handbook of Software Construction (2nd ed.). Microsoft Press.
12. Bass, L., Clements, P., & Kazman, R. (2012). Software Architecture in Practice (3rd ed.). Addison-Wesley Professional.
13. Fowler, M. (2010). Domain-Specific Languages. Addison-Wesley Professional.
14. Вікіпедія. (н.д.). ACID. Отримано з https://uk.wikipedia.org/wiki/ACID
15. Hostiq.ua. (н.д.). Що таке SSL?. Отримано з https://hostiq.ua/ukr/info/what-is-ssl/
16. Вікіпедія. (н.д.). React. Отримано з https://uk.wikipedia.org/wiki/React

17. Foxminded.ua. (н.д.). Дом це?. Отримано з https://foxminded.ua/dom-tse/
18. Вікіпедія. (н.д.). Загальний регламент про захист даних. Отримано з https://uk.wikipedia.org/wiki/General_Data_Protection_Regulation
19. Вікіпедія. (н.д.). Amazon Web Services. Отримано з https://uk.wikipedia.org/wiki/Amazon_Web_Services
20. Kubernetes.io. (н.д.). Що таке Kubernetes?. Отримано з https://kubernetes.io/uk/docs/concepts/overview/what-is-kubernetes/
21. Вікіпедія. (н.д.). PCI DSS. Отримано з https://uk.wikipedia.org/wiki/PCI_DSS
22. Qagroup.com.ua. (н.д.). Що таке Docker і навіщо він?. Отримано з https://qagroup.com.ua/publications/shcho-take-docker-i-navishcho-vin/
23. Вікіпедія. (н.д.). HIPAA. Отримано з https://uk.wikipedia.org/wiki/HIPAA
24. Вікіпедія. (н.д.). Jenkins. Отримано з https://uk.wikipedia.org/wiki/Jenkins
25. Вікіпедія. (н.д.). JSON. Отримано з https://uk.wikipedia.org/wiki/JSON
26. Вікіпедія. (н.д.). PostgreSQL. Отримано з https://uk.wikipedia.org/wiki/PostgreSQL
27. Вікіпедія. (н.д.). Spring Framework. Отримано з https://uk.wikipedia.org/wiki/Spring_Framework
28. Вікіпедія. (н.д.). Двофакторна аутентифікація. Отримано з https://uk.wikipedia.org/wiki/Two-factor_authentication
29. Вікіпедія. (н.д.). Java. Отримано з https://uk.wikipedia.org/wiki/Java
30. Hostiq.ua. (н.д.). Що таке API?. Отримано з https://hostiq.ua/blog/ukr/what-is-api/

ДОДАТОК А ЗВІТ ПОДІБНОСТІ



UNICHECK
by Turnitin

Ім'я користувача:
Лісовиченко Олег Іванович

ID перевірки:
1016334646

Дата перевірки:
08.06.2024 14:40:49 EEST

Тип перевірки:
Doc vs Internet + Library

Дата звіту:
08.06.2024 14:42:05 EEST

ID користувача:
76913

Назва документа: ІТ-з01_Березовський_ПЗ

Кількість сторінок: 58 Кількість слів: 8014 Кількість символів: 67813 Розмір файлу: 4.75 MB ID файлу: 1016135106

Виявлено модифікації тексту (можуть впливати на відсоток схожості)

9.62%
Схожість

Найбільша схожість: 2.17% з Інтернет-джерелом (http://dspace.wunu.edu.ua/bitstream/316497/33170/1/%d0%9c%d1%8..

5.04% Джерела з Інтернету 319 ..... Сторінка 60

9.62% Джерела з Бібліотеки 494 ..... Сторінка 61

0% Цитат

Вилучення цитат вимкнене

Вилучення списку бібліографічних посилань вимкнене

0%
Вилучень

Немає вилучених джерел

Модифікації

Виявлено модифікації тексту. Детальна інформація доступна в онлайн-звіті.

Замінені символи 39

Підозріле форматування 15 сторінок

ДОДАТОК Б АЛГОРИТМ ШИФРУВАННЯ ACCESS ТОКЕНУ

```
private String generateJWT(UUID id, String email, Role role, boolean enabled) {
        Date now = new Date();
        Date validity = new Date(now.getTime() + tokenExpirationTime);

        return Jwts.builder()
                .claim("id", id.toString())
                .claim("email", email)
                .claim("role", role.toString())
                .claim("enabled", enabled)
                .setIssuedAt(now)
                .setExpiration(validity)
                .signWith(SignatureAlgorithm.HS256, secretKey)
                .compact();
    }
```

ДОДАТОК В ІНСТАНС НА AWS EC2

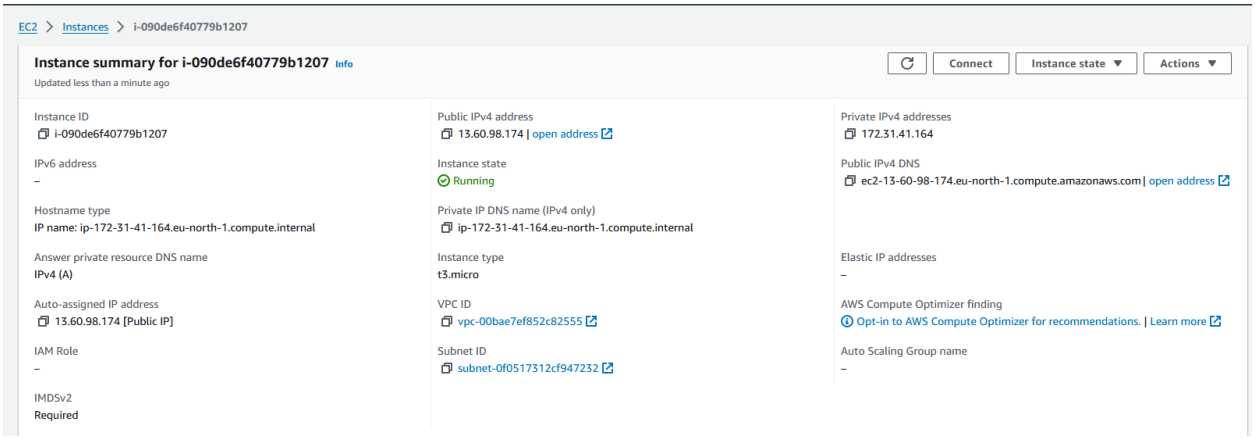


Рисунок А.1 – Інстанс на EC2

Факультет інформатики та обчислювальної техніки
Кафедра інформатики та програмної інженерії

“ЗАТВЕРДЖЕНО”
Завідувач кафедри
_____ Едуард ЖАРІКОВ
“___” _____ 2024 р.

ВЕБ-ЗАСТОСУНОК ДЛЯ УПРАВЛІННЯ ОСОБИСТИМИ ФІНАНСАМИ

Текст програми

КПІ.ІТ-045440.03.12

“ПОГОДЖЕНО”
Керівник проєкту:
_____ Владислав СОКОЛОВСЬКИЙ

Нормоконтроль:
_____Олександр СТЕЛЬМАХ

Виконавець:
_____Владислав БЕРЕЗОВСЬКИЙ

Київ – 2024

Посилання на репозиторій з повним текстом програмного коду
https://github.com/Kingsm01n/graduatework

Файл backend/common/base/pom.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
https://maven.apache.org/xsd/maven-4.0.0.xsd">
   <modelVersion>4.0.0</modelVersion>
   <parent>
      <groupId>org.springframework.boot</groupId>
      <artifactId>spring-boot-starter-parent</artifactId>
      <version>3.2.3</version>
      <relativePath/> <!-- lookup parent from repository -->
   </parent>
   <groupId>com.vladyslavberezovskyi</groupId>
   <artifactId>base</artifactId>
   <version>0.0.1-SNAPSHOT</version>
   <packaging>pom</packaging>
   <name>base</name>
   <description>Graduate Work of Vladyslav Berezovskyi</description>
   <properties>
      <java.version>17</java.version>
   </properties>
   <dependencies>
      <dependency>
         <groupId>org.springframework.boot</groupId>
         <artifactId>spring-boot-starter-actuator</artifactId>
      </dependency>
      <dependency>
         <groupId>org.springframework.boot</groupId>
         <artifactId>spring-boot-starter-data-jpa</artifactId>
      </dependency>
      <dependency>
         <groupId>com.nimbusds</groupId>
         <artifactId>nimbus-jose-jwt</artifactId>
         <version>9.37.3</version>
      </dependency>
      <dependency>
         <groupId>org.springframework.boot</groupId>
```

```
    <artifactId>spring-boot-starter-security</artifactId>
</dependency>
<dependency>
    <groupId>com.h2database</groupId>
    <artifactId>h2</artifactId>
    <scope>test</scope>
</dependency>
<dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-web</artifactId>
</dependency>
<dependency>
    <groupId>org.liquibase</groupId>
    <artifactId>liquibase-core</artifactId>
</dependency>
<dependency>
    <groupId>org.postgresql</groupId>
    <artifactId>postgresql</artifactId>
    <scope>runtime</scope>
</dependency>
<dependency>
    <groupId>org.springdoc</groupId>
    <artifactId>springdoc-openapi-starter-webmvc-ui</artifactId>
    <version>2.5.0</version>
</dependency>
<dependency>
    <groupId>org.projectlombok</groupId>
    <artifactId>lombok</artifactId>
    <optional>true</optional>
</dependency>
<dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-test</artifactId>
    <scope>test</scope>
</dependency>
<dependency>
    <groupId>org.springframework.security</groupId>
    <artifactId>spring-security-test</artifactId>
    <scope>test</scope>
</dependency>
<dependency>
    <groupId>org.mapstruct</groupId>
    <artifactId>mapstruct</artifactId>
```

```
    <version>1.5.3.Final</version>
</dependency>
<dependency>
    <groupId>io.jsonwebtoken</groupId>
    <artifactId>jjwt-api</artifactId>
    <version>0.11.2</version> <!-- Укажите актуальную версию -->
</dependency>
<dependency>
    <groupId>io.jsonwebtoken</groupId>
    <artifactId>jjwt-impl</artifactId>
    <version>0.11.2</version> <!-- Укажите актуальную версию -->
    <scope>runtime</scope>
</dependency>
<dependency>
    <groupId>io.jsonwebtoken</groupId>
    <artifactId>jjwt-jackson</artifactId>
    <version>0.11.2</version> <!-- Укажите актуальную версию -->
    <scope>runtime</scope>
</dependency>
<dependency>
    <groupId>javax.mail</groupId>
    <artifactId>mail</artifactId>
    <version>1.5.0-b01</version>
</dependency>
<dependency>
    <groupId>com.google.api-client</groupId>
    <artifactId>google-api-client</artifactId>
    <version>1.32.1</version>
</dependency>
<dependency>
    <groupId>com.google.oauth-client</groupId>
    <artifactId>google-oauth-client-jetty</artifactId>
    <version>1.36.0</version>
</dependency>
<dependency>
    <groupId>com.google.apis</groupId>
    <artifactId>google-api-services-gmail</artifactId>
    <version>v1-rev20211108-1.32.1</version>
</dependency>
<dependency>
    <groupId>com.google.auth</groupId>
    <artifactId>google-auth-library-oauth2-http</artifactId>
    <version>1.23.0</version>
```

```
    </dependency>
</dependencies>

<dependencyManagement>
    <dependencies>
        <dependency>
            <groupId>com.vladyslavberezovskyi</groupId>
            <artifactId>commons</artifactId>
            <version>0.0.1-SNAPSHOT</version>
        </dependency>
    </dependencies>
</dependencyManagement>

<build>
    <plugins>
        <plugin>
            <groupId>org.apache.maven.plugins</groupId>
            <artifactId>maven-compiler-plugin</artifactId>
            <configuration>
                <source>1.8</source>
                <target>1.8</target>
                <annotationProcessorPaths>
                    <path>
                        <groupId>org.projectlombok</groupId>
                        <artifactId>lombok</artifactId>
                        <version>${lombok.version}</version>
                    </path>
                    <path>
                        <groupId>org.mapstruct</groupId>
                        <artifactId>mapstruct-processor</artifactId>
                        <version>1.5.3.Final</version>
                    </path>
                </annotationProcessorPaths>
                <compilerArgs>
                    <compilerArg>
                        -Amapstruct.defaultComponentModel=spring
                    </compilerArg>
                </compilerArgs>
            </configuration>
        </plugin>
    </plugins>
</build>
```

```
    <modules>
        <module>../commons</module>
    </modules>

</project>
```

Файл backend/common/commons/src/main/java/com/vladyslavberezovskyi/config/SwaggerConfig.java

```
package com.vladyslavberezovskyi.config;

import io.swagger.v3.oas.annotations.enums.SecuritySchemeType;
import io.swagger.v3.oas.annotations.security.SecurityScheme;
import org.springframework.context.annotation.Configuration;

@Configuration
@SecurityScheme(
        name = "Authorization",
        type = SecuritySchemeType.HTTP,
        bearerFormat = "JWT",
        scheme = "bearer"
)
public class SwaggerConfig {
}
```

Файл backend/common/commons/src/main/java/com/vladyslavberezovskyi/entity/BaseEntity.java

```
package com.vladyslavberezovskyi.entity;

import jakarta.persistence.*;
import lombok.*;

import java.util.UUID;

import static jakarta.persistence.InheritanceType.TABLE_PER_CLASS;

@Getter
@Setter
@MappedSuperclass
@NoArgsConstructor
@AllArgsConstructor
public class BaseEntity {
    @Id
    @GeneratedValue(strategy = GenerationType.AUTO)
    @Column(name = "id", nullable = false)
    protected UUID id;
}
```

Файл backend/common/commons/src/main/java/com/vladyslavberezovskyi/error/BaseException.java

```
package com.vladyslavberezovskyi.error;

public class BaseException extends RuntimeException {
}
```

Файл backend/common/commons/src/main/java/com/vladyslavberezovskyi/error/ForbiddenException.java

```
package com.vladyslavberezovskyi.error;

public class ForbiddenException extends BaseException {
}
```

Файл backend/common/commons/src/main/java/com/vladyslavberezovskyi/error/ResourceNotFoundException.java

```
package com.vladyslavberezovskyi.error;

public class ResourceNotFoundException extends BaseException {
}
```

Файл backend/common/commons/src/main/java/com/vladyslavberezovskyi/mapper/PageMapper.java

```
package com.vladyslavberezovskyi.mapper;

import com.vladyslavberezovskyi.util.PageRequest;
import org.springframework.data.domain.Pageable;
import org.springframework.stereotype.Component;

@Component
public class PageMapper {

    public Pageable pageRequestToPageable(PageRequest pageRequest) {
```

```
        return org.springframework.data.domain.PageRequest.of(
                pageRequest.getPage(),
                pageRequest.getSize());
    }
}
```

Файл backend/common/commons/src/main/java/com/vladyslavberezovskyi/security/annotations/IsUser.java

```
package com.vladyslavberezovskyi.security.annotations;

import org.springframework.security.access.prepost.PreAuthorize;

import java.lang.annotation.ElementType;
import java.lang.annotation.Retention;
import java.lang.annotation.RetentionPolicy;
import java.lang.annotation.Target;

@Target(ElementType.METHOD)
@Retention(RetentionPolicy.RUNTIME)
@PreAuthorize("hasRole('USER')")
public @interface IsUser {
}
```

Файл backend/common/commons/src/main/java/com/vladyslavberezovskyi/security/enums/Role.java

```
package com.vladyslavberezovskyi.security.enums;
```

```
import org.springframework.security.core.GrantedAuthority;

public enum Role implements GrantedAuthority {

    ADMIN, USER;

    @Override
    public String getAuthority() {
        return this.name();
    }
}
```

Файл backend/common/commons/src/main/java/com/vladyslavberezovskyi/security/IgnoredPathProperties.java

```
package com.vladyslavberezovskyi.security;

import lombok.Getter;
import org.springframework.http.HttpMethod;
import org.springframework.security.web.util.matcher.AntPathRequestMatcher;

import java.util.*;
import java.util.stream.Collectors;

@Getter
public class IgnoredPathProperties {

    private static final Set<String> IGNORE = new HashSet<>(List.of(new
String[]{"/actuator/**",
```

```
            "/v3/api-docs/**",
            "/swagger-ui/**",
            "/favicon.ico",
            "/api/v1/users",
            "/api/v1/users/auth",
            "/api/v1/invitations/confirmInvitation/**"}));

    private final Set<AntPathRequestMatcher> antMatchers;

    public IgnoredPathProperties() {
        this.antMatchers = IGNORE.stream().map(path -> {
            if (path.contains("users")) {
                return new AntPathRequestMatcher(path, HttpMethod.POST.name());
            } else {
                return new AntPathRequestMatcher(path);
            }
        }).collect(Collectors.toSet());
    }

}
```

Файл backend/common/commons/src/main/java/com/vladyslavberezovskyi/security/SecurityConfig.java

```
package com.vladyslavberezovskyi.security;

import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.Configuration;
import org.springframework.http.HttpMethod;
```

```
import org.springframework.security.config.Customizer;
import org.springframework.security.config.annotation.web.builders.HttpSecurity;
import
org.springframework.security.config.annotation.web.configuration.EnableWebSec
urity;
import
org.springframework.security.config.annotation.web.configurers.AbstractHttpConf
igurer;
import org.springframework.security.crypto.factory.PasswordEncoderFactories;
import org.springframework.security.crypto.password.PasswordEncoder;
import org.springframework.security.web.SecurityFilterChain;
import
org.springframework.security.web.authentication.UsernamePasswordAuthenticatio
nFilter;
import org.springframework.web.cors.CorsConfiguration;
import org.springframework.web.cors.CorsConfigurationSource;
import org.springframework.web.cors.UrlBasedCorsConfigurationSource;
import org.springframework.web.servlet.config.annotation.CorsRegistry;
import org.springframework.web.servlet.config.annotation.EnableWebMvc;
import org.springframework.web.servlet.config.annotation.WebMvcConfigurer;

import java.util.Arrays;
import java.util.List;

@Configuration
@EnableWebMvc
@EnableWebSecurity
public class SecurityConfig implements WebMvcConfigurer {
```

```
    @Bean
    public SecurityFilterChain securityFilterChain(HttpSecurity http) throws
Exception {
        http.authorizeHttpRequests((authorize) -> authorize
                .requestMatchers("/v3/api-docs/**").permitAll()
                .requestMatchers("/swagger-ui/**").permitAll()
                .requestMatchers("/favicon.ico").permitAll()
                .requestMatchers(HttpMethod.POST, "/api/v1/users").permitAll()
                .requestMatchers(HttpMethod.POST, "/api/v1/users/auth").permitAll()
                .requestMatchers(HttpMethod.GET,
"/api/v1/invitations/confirmInvitation/**").permitAll()
                .anyRequest().authenticated()
        );

        http.cors(httpSecurityCorsConfigurer ->
httpSecurityCorsConfigurer.configurationSource(corsConfigurationSource()));
        http.csrf(AbstractHttpConfigurer::disable);

        http.addFilterBefore(new TokenBasedAuthFilter(),
UsernamePasswordAuthenticationFilter.class);

        return http.build();
    }

    @Bean
    CorsConfigurationSource corsConfigurationSource() {
        CorsConfiguration configuration = new CorsConfiguration();
        configuration.setAllowedOrigins(List.of("*"));
        configuration.setAllowedMethods(List.of("*"));
```

```
        configuration.setAllowedHeaders(List.of("*"));
        UrlBasedCorsConfigurationSource source = new
UrlBasedCorsConfigurationSource();
        source.registerCorsConfiguration("/**", configuration);
        return source;
    }

    @Bean
    public PasswordEncoder passwordEncoder() {
        return PasswordEncoderFactories.createDelegatingPasswordEncoder();
    }

    @Override
    public void addCorsMappings(CorsRegistry registry) {
        registry.addMapping("/**")
                .allowedOrigins("*")
                .allowedHeaders("*")
                .allowedMethods("*");
    }
}
```

Файл backend/common/commons/src/main/java/com/vladyslavberezovskyi/security/TokenBasedAuthFilter.java

```
package com.vladyslavberezovskyi.security;

import com.nimbusds.jwt.JWTClaimsSet;
import com.nimbusds.jwt.JWTParser;
import com.vladyslavberezovskyi.error.ForbiddenException;
```

```
import com.vladyslavberezovskyi.security.enums.Role;
import jakarta.servlet.FilterChain;
import jakarta.servlet.ServletException;
import jakarta.servlet.http.HttpServletRequest;
import jakarta.servlet.http.HttpServletResponse;
import lombok.NoArgsConstructor;
import
org.springframework.security.authentication.UsernamePasswordAuthenticationTo
ken;
import org.springframework.security.core.context.SecurityContextHolder;
import org.springframework.web.filter.OncePerRequestFilter;

import java.io.IOException;
import java.text.ParseException;
import java.util.UUID;

import static org.apache.commons.lang3.StringUtils.isNotBlank;

@NoArgsConstructor
public class TokenBasedAuthFilter extends OncePerRequestFilter {

    private final IgnoredPathProperties ignoredPathProperties = new
IgnoredPathProperties();

    @Override
    protected void doFilterInternal(HttpServletRequest request,
                                    HttpServletResponse response,
                                    FilterChain filterChain) throws ServletException,
IOException {
```

```
        boolean ignored =
ignoredPathProperties.getAntMatchers().stream().anyMatch(ignoredPath ->
ignoredPath.matches(request));

        if (!ignored) {
            boolean tokenIsPresent = isNotBlank(request.getHeader("Authorization"));

            if (tokenIsPresent) {
                try {
                    establishUserContext(request);
                } catch (ParseException e) {
                    throw new ForbiddenException();
                }
            } else {
                throw new ForbiddenException();
            }
        }

        filterChain.doFilter(request, response);
    }

    private void establishUserContext(HttpServletRequest request) throws
ParseException {
        String token = request.getHeader("Authorization").replace("Bearer ", "");

        JWTClaimsSet jWtClaimsSet = JWTParser.parse(token).getJWTClaimsSet();
        UserDetails userDetails = UserDetails.builder()
                .id(UUID.fromString(jWtClaimsSet.getStringClaim("id")))
                .email(jWtClaimsSet.getStringClaim("email"))
```

```
                .password(jWtClaimsSet.getStringClaim("password"))
                .role(Role.valueOf(jWtClaimsSet.getStringClaim("role")))
                .enabled(jWtClaimsSet.getBooleanClaim("enabled"))
                .build();

        if (userDetails.isEnabled()) {
            SecurityContextHolder.getContext()
                    .setAuthentication(new
UsernamePasswordAuthenticationToken(userDetails, userDetails,
userDetails.getAuthorities()));
        }
    }
}
```

Файл backend/common/commons/src/main/java/com/vladyslavberezovskyi/security/TokenResponse.java

```
package com.vladyslavberezovskyi.security;

import lombok.AllArgsConstructor;
import lombok.Builder;
import lombok.Data;
import lombok.NoArgsConstructor;

@Data
@Builder
@NoArgsConstructor
@AllArgsConstructor
public class TokenResponse {
```

```
    private String token;
    private Long expiresIn;

}
```

Файл backend/common/commons/src/main/java/com/vladyslavberezovskyi/security/UserDetails.java

```
package com.vladyslavberezovskyi.security;

import com.vladyslavberezovskyi.security.enums.Role;
import lombok.Builder;
import lombok.Data;
import org.springframework.security.core.GrantedAuthority;

import java.util.Collection;
import java.util.Collections;
import java.util.UUID;

@Data
@Builder
public class UserDetails implements
org.springframework.security.core.userdetails.UserDetails {

    private UUID id;
    private String email;
    private String password;
    private Role role;
```

```
private boolean enabled;

@Override
public Collection<? extends GrantedAuthority> getAuthorities() {
   return Collections.singleton(role);
}

@Override
public String getPassword() {
   return password;
}

@Override
public String getUsername() {
   return email;
}

@Override
public boolean isAccountNonExpired() {
   return true;
}

@Override
public boolean isAccountNonLocked() {
   return true;
}

@Override
public boolean isCredentialsNonExpired() {
```

```
        return true;
    }

    @Override
    public boolean isEnabled() {
        return enabled;
    }
}
```

Файл backend/common/commons/src/main/java/com/vladyslavberezovskyi/util/EmailSender.java

```
package com.vladyslavberezovskyi.util;

import com.google.api.client.auth.oauth2.Credential;
import com.google.api.client.extensions.java6.auth.oauth2.AuthorizationCodeInstalledApp;
import com.google.api.client.extensions.jetty.auth.oauth2.LocalServerReceiver;
import com.google.api.client.googleapis.auth.oauth2.GoogleAuthorizationCodeFlow;
import com.google.api.client.googleapis.auth.oauth2.GoogleClientSecrets;
import com.google.api.client.googleapis.javanet.GoogleNetHttpTransport;
import com.google.api.client.googleapis.json.GoogleJsonResponseException;
import com.google.api.client.http.javanet.NetHttpTransport;
import com.google.api.client.json.JsonFactory;
import com.google.api.client.json.gson.GsonFactory;
import com.google.api.client.util.store.FileDataStoreFactory;
import com.google.api.services.gmail.Gmail;
```

```
import com.google.api.services.gmail.GmailScopes;
import com.google.api.services.gmail.model.Message;
import lombok.SneakyThrows;
import org.apache.tomcat.util.codec.binary.Base64;
import org.springframework.beans.factory.annotation.Value;
import org.springframework.stereotype.Component;

import javax.mail.MessagingException;
import javax.mail.Session;
import javax.mail.internet.InternetAddress;
import javax.mail.internet.MimeMessage;
import java.io.*;
import java.security.GeneralSecurityException;
import java.util.Collections;
import java.util.List;
import java.util.Properties;
import java.util.UUID;

@Component
public class EmailSender {

    @Value("${email.username}")
    private String emailAddress;
    @Value("${email.confirmation.url}")
    private String confirmationUrl;
    @Value("${email.confirmation.path}")
    private String confirmationPath;

    private static final String APPLICATION_NAME = "Graduatework";
```

```
    private static final JsonFactory JSON_FACTORY =
GsonFactory.getDefaultInstance();
    private static final String TOKENS_DIRECTORY_PATH = "tokens";
    private static final List<String> SCOPES =
Collections.singletonList(GmailScopes.GMAIL_SEND);
    private static final String CREDENTIALS_FILE_PATH = "/credentials.json";

    @SneakyThrows
    public void sendInvitation(String toEmailAddress, UUID invitationId) {
        NetHttpTransport HTTP_TRANSPORT =
GoogleNetHttpTransport.newTrustedTransport();
        // Create the gmail API client
        Gmail service = new Gmail.Builder(HTTP_TRANSPORT,
JSON_FACTORY, getCredentials(HTTP_TRANSPORT))
                .setApplicationName(APPLICATION_NAME)
                .build();

        // Encode as MIME message
        Properties props = new Properties();
        Session session = Session.getDefaultInstance(props, null);
        MimeMessage email = createInvitation(toEmailAddress, invitationId);

        ByteArrayOutputStream buffer = new ByteArrayOutputStream();
        email.writeTo(buffer);
        byte[] bytes = buffer.toByteArray();
        String encodedEmail = Base64.encodeBase64URLSafeString(bytes);
        Message message = new Message();
        message.setRaw(encodedEmail);
```

```
        try {
            // Create send message
            service.users()
                    .messages()
                    .send("me", message)
                    .execute();
        } catch (GoogleJsonResponseException e) {
            throw new RuntimeException();
        }
    }

    public void sendWarning(String toEmailAddress, String budgetName, String
budgetValue) throws Exception {
        NetHttpTransport HTTP_TRANSPORT =
GoogleNetHttpTransport.newTrustedTransport();
        // Create the gmail API client
        Gmail service = new Gmail.Builder(HTTP_TRANSPORT,
JSON_FACTORY, getCredentials(HTTP_TRANSPORT))
                .setApplicationName(APPLICATION_NAME)
                .build();

        // Encode as MIME message
        Properties props = new Properties();
        Session session = Session.getDefaultInstance(props, null);
        MimeMessage email = createWarning(toEmailAddress, budgetName,
budgetValue);

        ByteArrayOutputStream buffer = new ByteArrayOutputStream();
        email.writeTo(buffer);
```

```
        byte[] bytes = buffer.toByteArray();
        String encodedEmail = Base64.encodeBase64URLSafeString(bytes);
        Message message = new Message();
        message.setRaw(encodedEmail);

        try {
            // Create send message
            service.users()
                    .messages()
                    .send("me", message)
                    .execute();
        } catch (GoogleJsonResponseException e) {
            throw new Exception();
        }
    }

    private MimeMessage createInvitation(String toEmailAddress, UUID
invitationId) throws MessagingException {
        Properties props = new Properties();
        Session session = Session.getDefaultInstance(props, null);

        MimeMessage email = new MimeMessage(session);

        email.setFrom(new InternetAddress(emailAddress));
        email.addRecipient(javax.mail.Message.RecipientType.TO,
                new InternetAddress(toEmailAddress));
        email.setSubject("Email Verification");
        email.setText("For email verification click this link: " +
generateUrl(invitationId));
```

```
        return email;
    }

    private MimeMessage createWarning(String toEmailAddress, String
budgetName, String budgetValue) throws MessagingException {
        Properties props = new Properties();
        Session session = Session.getDefaultInstance(props, null);

        MimeMessage email = new MimeMessage(session);

        email.setFrom(new InternetAddress(emailAddress));
        email.addRecipient(javax.mail.Message.RecipientType.TO,
                new InternetAddress(toEmailAddress));
        email.setSubject("Budget overflow WARNING");
        email.setText("You exceeded your budget " + budgetName + " limit " +
budgetValue);
        return email;
    }

    private String generateUrl(UUID invitationId) {
        return confirmationUrl + confirmationPath + "/" + invitationId;
    }

    private static Credential getCredentials(final NetHttpTransport
HTTP_TRANSPORT)
            throws IOException {
        // Load client secrets.
        InputStream in =
EmailSender.class.getResourceAsStream(CREDENTIALS_FILE_PATH);
```

```
        if (in == null) {
            throw new FileNotFoundException("Resource not found: " +
CREDENTIALS_FILE_PATH);
        }
        GoogleClientSecrets clientSecrets =
                GoogleClientSecrets.load(JSON_FACTORY, new
InputStreamReader(in));

        // Build flow and trigger user authorization request.
        GoogleAuthorizationCodeFlow flow = new
GoogleAuthorizationCodeFlow.Builder(
                HTTP_TRANSPORT, JSON_FACTORY, clientSecrets, SCOPES)
                .setDataStoreFactory(new FileDataStoreFactory(new
File(TOKENS_DIRECTORY_PATH)))
                .setAccessType("offline")
                .build();
        LocalServerReceiver receiver = new
LocalServerReceiver.Builder().setPort(8888).build();
        //returns an authorized Credential object.
        return new AuthorizationCodeInstalledApp(flow,
receiver).authorize("user1");
    }

}
```

Файл backend/common/commons/src/main/java/com/vladyslavberezovskyi/util/PageRequest.java

```
package com.vladyslavberezovskyi.util;
```

```
import lombok.Builder;
import lombok.Data;

@Data
@Builder
public class PageRequest {

    private int page;
    private int size;
    private String[] sort;

}
```

Файл backend/common/commons/src/main/java/com/vladyslavberezovskyi/CommonStarter.java

```
package com.vladyslavberezovskyi;

import org.springframework.boot.autoconfigure.AutoConfiguration;
import org.springframework.context.annotation.ComponentScan;

@ComponentScan
@AutoConfiguration
public class CommonStarter {
}
```

Файл backend/common/commons/src/main/resources/application.yaml

```
email:
```

```
  username: ${EMAIL_USERNAME}
  confirmation:
    url: ${EMAIL_CONFIRMATION_URL}
    path: /api/v1/invitations/confirmInvitation
```

Файл backend/common/commons/pom.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
https://maven.apache.org/xsd/maven-4.0.0.xsd">
    <modelVersion>4.0.0</modelVersion>
    <parent>
        <groupId>com.vladyslavberezovskyi</groupId>
        <artifactId>base</artifactId>
        <version>0.0.1-SNAPSHOT</version>
        <relativePath>../base</relativePath>
    </parent>

    <artifactId>commons</artifactId>
    <version>0.0.1-SNAPSHOT</version>
    <name>commons</name>

</project>
```

Файл backend/common/Dockerfile

```
FROM maven:3.8.4 as builder

COPY base/pom.xml /app/common/base/
```

```
COPY commons/src /app/common/commons/src/
COPY commons/pom.xml /app/common/commons/

RUN mvn -ntp -B -f /app/common/base/pom.xml clean install

FROM amazoncorretto:17

COPY --from=builder /app/common/base/pom.xml /app/
COPY --from=builder /app/common/commons/target/*.jar /app/
```

Файл backend/service/money-management-service/src/main/java/com/vladyslavberezovskyi/controller/BalanceController.java

```
package com.vladyslavberezovskyi.controller;

import com.vladyslavberezovskyi.dto.balance.CreateBalanceRequest;
import com.vladyslavberezovskyi.dto.balance.PatchBalanceRequest;
import com.vladyslavberezovskyi.mapper.PageMapper;
import com.vladyslavberezovskyi.security.annotations.IsUser;
import com.vladyslavberezovskyi.service.BalanceService;
import com.vladyslavberezovskyi.dto.balance.BalanceResponse;
import com.vladyslavberezovskyi.mapper.BalanceMapper;
import com.vladyslavberezovskyi.util.PageRequest;
import io.swagger.v3.oas.annotations.security.SecurityRequirement;
import lombok.RequiredArgsConstructor;
import org.springframework.data.domain.Page;
import org.springframework.web.bind.annotation.*;
```

```
import java.util.List;
import java.util.UUID;

@RestController
@RequiredArgsConstructor
@RequestMapping(path = "/api/v1/balances")
@SecurityRequirement(name = "Authorization")
public class BalanceController {

    private final BalanceService service;
    private final BalanceMapper mapper;
    private final PageMapper pageMapper;

    @GetMapping(path = "/{balanceId}")
    public BalanceResponse getBalanceById(@PathVariable UUID balanceId) {
        return mapper.modelToDto(service.getBalanceById(balanceId));
    }

    @GetMapping
    public List<BalanceResponse> getAllBalances() {
        return mapper.modelToDto(service.getAllBalances());
    }

    @GetMapping("/paged")
    public Page<BalanceResponse> getAllBalances(PageRequest pageRequest) {
        return
mapper.modelToDto(service.getAllBalances(pageMapper.pageRequestToPageable
(pageRequest)));
    }
```

```
    @PostMapping
    public BalanceResponse createBalance(@RequestBody CreateBalanceRequest
createBalanceRequest) {
        return
mapper.modelToDto(service.createBalance(mapper.dtoToModel(createBalanceRe
quest)));
    }

    @PatchMapping(path = "/{balanceId}")
    public BalanceResponse updateBalance(@PathVariable UUID balanceId,
                                         @RequestBody PatchBalanceRequest
patchBalanceRequest) {
        return mapper.modelToDto(service.updateBalance(balanceId,
mapper.dtoToModel(patchBalanceRequest)));
    }

    @DeleteMapping(path = "/{balanceId}")
    public void deleteBalance(@PathVariable UUID balanceId) {
        service.deleteBalance(balanceId);
    }
}
```

Файл backend/service/money-management-service/src/main/java/com/vladyslavberezovskyi/controller/BudgetController.java

```
package com.vladyslavberezovskyi.controller;

import com.vladyslavberezovskyi.dto.budget.BudgetResponse;
```

```
import com.vladyslavberezovskyi.dto.budget.CreateBudgetRequest;
import com.vladyslavberezovskyi.dto.budget.PatchBudgetRequest;
import com.vladyslavberezovskyi.mapper.BudgetMapper;
import com.vladyslavberezovskyi.mapper.PageMapper;
import com.vladyslavberezovskyi.security.annotations.IsUser;
import com.vladyslavberezovskyi.service.BudgetService;
import com.vladyslavberezovskyi.util.PageRequest;
import io.swagger.v3.oas.annotations.security.SecurityRequirement;
import lombok.RequiredArgsConstructor;
import org.springframework.data.domain.Page;
import org.springframework.http.HttpStatus;
import org.springframework.security.core.Authentication;
import org.springframework.security.core.context.SecurityContextHolder;
import org.springframework.web.bind.annotation.*;

import java.util.List;
import java.util.UUID;

@RestController
@RequiredArgsConstructor
@RequestMapping("/api/v1/budgets")
@SecurityRequirement(name = "Authorization")
public class BudgetController {

    private final BudgetService service;
    private final BudgetMapper mapper;
    private final PageMapper pageMapper;

    @IsUser
```

```
    @GetMapping("/{budgetId}")
    public BudgetResponse getBudgetById(@PathVariable UUID budgetId) {
        return mapper.modelToDto(service.getBudgetById(budgetId));
    }

    @IsUser
    @GetMapping
    public List<BudgetResponse> getAllBudgets() {
        return mapper.modelToDto(service.getAllBudgets());
    }

    @IsUser
    @GetMapping("/paged")
    public Page<BudgetResponse> getAllBudgets(PageRequest pageRequest) {
        return
mapper.modelToDto(service.getAllBudgets(pageMapper.pageRequestToPageable(
pageRequest)));
    }

    @IsUser
    @PostMapping
    public BudgetResponse createBudget(@RequestBody CreateBudgetRequest
createBudgetRequest) {
        return
mapper.modelToDto(service.createBudget(mapper.dtoToModel(createBudgetRequ
est)));
    }

    @IsUser
```

```
    @PatchMapping("/{budgetId}")
    public BudgetResponse updateBudget(@PathVariable UUID budgetId,
                                       @RequestBody PatchBudgetRequest
patchBudgetRequest) {
        return mapper.modelToDto(service.updateBudget(budgetId,
mapper.dtoToModel(patchBudgetRequest)));
    }

    @IsUser
    @DeleteMapping("/{budgetId}")
    public void deleteBudget(@PathVariable UUID budgetId) {
        service.deleteBudget(budgetId);
    }
}
```

Файл backend/service/money-management-service/src/main/java/com/vladyslavberezovskyi/controller/CategoryController.java

```
package com.vladyslavberezovskyi.controller;

import com.vladyslavberezovskyi.dto.category.CategoryResponse;
import com.vladyslavberezovskyi.dto.category.CreateCategoryRequest;
import com.vladyslavberezovskyi.dto.category.PatchCategoryRequest;
import com.vladyslavberezovskyi.mapper.CategoryMapper;
import com.vladyslavberezovskyi.mapper.PageMapper;
import com.vladyslavberezovskyi.security.annotations.IsUser;
import com.vladyslavberezovskyi.service.CategoryService;
import com.vladyslavberezovskyi.util.PageRequest;
import io.swagger.v3.oas.annotations.security.SecurityRequirement;
```

```
import lombok.RequiredArgsConstructor;
import org.springframework.data.domain.Page;
import org.springframework.web.bind.annotation.*;

import java.util.List;
import java.util.UUID;

@RestController
@RequiredArgsConstructor
@RequestMapping("/api/v1/categories")
@SecurityRequirement(name = "Authorization")
public class CategoryController {

    private final CategoryService service;
    private final CategoryMapper mapper;
    private final PageMapper pageMapper;

    @IsUser
    @GetMapping("/{categoryId}")
    public CategoryResponse getCategoryById(@PathVariable UUID categoryId) {
        return mapper.modelToDto(service.getCategoryById(categoryId));
    }

    @IsUser
    @GetMapping
    public List<CategoryResponse> getAllCategories() {
        return mapper.modelToDto(service.getAllCategories());
    }
```

```
    @IsUser
    @GetMapping("/paged")
    public Page<CategoryResponse> getAllCategories(PageRequest pageRequest) {
        return
mapper.modelToDto(service.getAllCategories(pageMapper.pageRequestToPageab
le(pageRequest)));
    }

    @IsUser
    @PostMapping
    public CategoryResponse createCategory(@RequestBody
CreateCategoryRequest createCategoryRequest) {
        return
mapper.modelToDto(service.createCategory(mapper.dtoToModel(createCategory
Request)));
    }

    @IsUser
    @PatchMapping("/{categoryId}")
    public CategoryResponse updateCategory(@PathVariable UUID categoryId,
                                           @RequestBody PatchCategoryRequest
patchCategoryRequest) {
        return mapper.modelToDto(service.updateCategory(categoryId,
mapper.dtoToModel(patchCategoryRequest)));
    }

    @IsUser
    @DeleteMapping("/{categoryId}")
    public void deleteCategory(@PathVariable UUID categoryId) {
```

```
        service.deleteCategory(categoryId);
    }

}
```

Файл backend/service/money-management-service/src/main/java/com/vladyslavberezovskyi/controller/TransactionController.java

```
package com.vladyslavberezovskyi.controller;

import com.vladyslavberezovskyi.dto.transaction.CreateTransactionRequest;
import com.vladyslavberezovskyi.dto.transaction.PatchTransactionRequest;
import com.vladyslavberezovskyi.dto.transaction.TransactionResponse;
import com.vladyslavberezovskyi.mapper.PageMapper;
import com.vladyslavberezovskyi.mapper.TransactionMapper;
import com.vladyslavberezovskyi.security.annotations.IsUser;
import com.vladyslavberezovskyi.service.TransactionService;
import com.vladyslavberezovskyi.util.PageRequest;
import io.swagger.v3.oas.annotations.security.SecurityRequirement;
import lombok.RequiredArgsConstructor;
import org.springframework.data.domain.Page;
import org.springframework.lang.Nullable;
import org.springframework.web.bind.annotation.*;

import java.util.List;
import java.util.UUID;

@RestController
@RequiredArgsConstructor
```

```
@RequestMapping("/api/v1/transactions")
@SecurityRequirement(name = "Authorization")
public class TransactionController {

    private final TransactionService service;
    private final TransactionMapper mapper;
    private final PageMapper pageMapper;

    @IsUser
    @GetMapping("/{transactionId}")
    public TransactionResponse getTransactionById(@PathVariable UUID
transactionId) {
        return mapper.modelToDto(service.getTransactionById(transactionId));
    }

    @IsUser
    @GetMapping
    public List<TransactionResponse> getAllTransactions(@RequestParam
@Nullable UUID balanceId) {
        return mapper.modelToDto(service.getAllTransactions(balanceId));
    }

    @IsUser
    @GetMapping("/paged")
    public Page<TransactionResponse> getAllTransactions(PageRequest
pageRequest) {
        return
mapper.modelToDto(service.getAllTransactions(pageMapper.pageRequestToPage
able(pageRequest)));
```

```
    }

    @IsUser
    @PostMapping
    public TransactionResponse createTransaction(@RequestBody
CreateTransactionRequest createTransactionRequest) {
        return
mapper.modelToDto(service.createTransaction(mapper.dtoToModel(createTransac
tionRequest)));
    }

    @IsUser
    @PatchMapping("/{transactionId}")
    public TransactionResponse updateTransaction(@PathVariable UUID
transactionId,
                                                 @RequestBody PatchTransactionRequest
patchTransactionRequest) {
        return mapper.modelToDto(service.updateTransaction(transactionId,
mapper.dtoToModel(patchTransactionRequest)));
    }

    @IsUser
    @DeleteMapping("/{transactionId}")
    public void deleteTransaction(@PathVariable UUID transactionId) {
        service.deleteTransaction(transactionId);
    }
}
```

Файл backend/service/money-management-service/src/main/java/com/vladyslavberezovskyi/dao/entity/BalanceEntity.java

```
package com.vladyslavberezovskyi.dao.entity;

import com.vladyslavberezovskyi.entity.BaseEntity;
import jakarta.persistence.CascadeType;
import jakarta.persistence.Entity;
import jakarta.persistence.OneToMany;
import lombok.*;

import java.util.ArrayList;
import java.util.List;
import java.util.UUID;

@Builder
@Getter
@Setter
@NoArgsConstructor
@AllArgsConstructor
@Entity(name = "balance")
public class BalanceEntity extends BaseEntity {

    private String name;
    private Double currentValue;
    private UUID userId;
    @OneToMany(mappedBy = "balance", cascade = CascadeType.ALL,
orphanRemoval = true)
    private List<TransactionEntity> transactions = new ArrayList<>();
```

```
}
```

Файл backend/service/money-management-service/src/main/java/com/vladyslavberezovskyi/dao/entity/BudgetEntity.java

```
package com.vladyslavberezovskyi.dao.entity;

import com.vladyslavberezovskyi.entity.BaseEntity;
import jakarta.persistence.*;
import lombok.*;

import java.util.ArrayList;
import java.util.List;
import java.util.UUID;

@Builder
@Getter
@Setter
@NoArgsConstructor
@AllArgsConstructor
@Entity(name = "budget")
public class BudgetEntity extends BaseEntity {

    private String name;
    private Double currentValue;
    private Double neededValue;
    private UUID userId;
    @ManyToOne
    @JoinColumn(name = "category_id")
```

```
    private CategoryEntity category;

}
```

Файл backend/service/money-management-service/src/main/java/com/vladyslavberezovskyi/dao/entity/CategoryEntity.java

```
package com.vladyslavberezovskyi.dao.entity;

import com.vladyslavberezovskyi.entity.BaseEntity;
import jakarta.persistence.Entity;
import jakarta.persistence.OneToMany;
import jakarta.persistence.OneToOne;
import lombok.*;

import java.util.ArrayList;
import java.util.List;
import java.util.UUID;

@Builder
@NoArgsConstructor
@Getter
@Setter
@Entity(name = "category")
@AllArgsConstructor
public class CategoryEntity extends BaseEntity {

    private String name;
    private UUID userId;
```

```
    @OneToMany(mappedBy = "category", orphanRemoval = true)
    private List<BudgetEntity> budgets = new ArrayList<>();
    @OneToMany(mappedBy = "category", orphanRemoval = true)
    private List<TransactionEntity> transactions = new ArrayList<>();

}
```

Файл backend/service/money-management-service/src/main/java/com/vladyslavberezovskyi/dao/entity/TransactionEntity.java

```
package com.vladyslavberezovskyi.dao.entity;

import com.vladyslavberezovskyi.entity.BaseEntity;
import jakarta.persistence.Entity;
import jakarta.persistence.JoinColumn;
import jakarta.persistence.ManyToOne;
import lombok.*;
import org.hibernate.annotations.CreationTimestamp;

import java.util.Date;
import java.util.UUID;

@Builder
@NoArgsConstructor
@Getter
@Setter
@Entity(name = "transaction")
@AllArgsConstructor
public class TransactionEntity extends BaseEntity {
```

```
    private String name;
    private Double value;
    private UUID userId;
    @CreationTimestamp
    private Date date;
    @ManyToOne
    @JoinColumn(name = "balance_id")
    private BalanceEntity balance;
    @ManyToOne
    @JoinColumn(name = "category_id")
    private CategoryEntity category;

}
```

Файл backend/service/money-management-service/src/main/java/com/vladyslavberezovskyi/dao/repository/BalanceRepository.java

```
package com.vladyslavberezovskyi.dao.repository;

import com.vladyslavberezovskyi.dao.entity.BalanceEntity;
import org.springframework.data.jpa.repository.JpaRepository;
import org.springframework.stereotype.Repository;

import java.util.UUID;

@Repository
public interface BalanceRepository extends JpaRepository<BalanceEntity, UUID>
{
```

```
}
```

Файл backend/service/money-management-service/src/main/java/com/vladyslavberezovskyi/dao/repository/BudgetRepository.java

```
package com.vladyslavberezovskyi.dao.repository;

import com.vladyslavberezovskyi.dao.entity.BudgetEntity;
import com.vladyslavberezovskyi.dao.entity.CategoryEntity;
import org.springframework.data.jpa.repository.JpaRepository;
import org.springframework.stereotype.Repository;

import java.util.List;
import java.util.UUID;

@Repository
public interface BudgetRepository extends JpaRepository<BudgetEntity, UUID> {

    List<BudgetEntity> findAllByCategory(CategoryEntity categoryEntity);

}
```

Файл backend/service/money-management-service/src/main/java/com/vladyslavberezovskyi/dao/repository/CategoryRepository.java

```
package com.vladyslavberezovskyi.dao.repository;

import com.vladyslavberezovskyi.dao.entity.CategoryEntity;
import org.springframework.data.jpa.repository.JpaRepository;
```

```
import org.springframework.stereotype.Repository;

import java.util.UUID;

@Repository
public interface CategoryRepository extends JpaRepository<CategoryEntity,
UUID> {
}
```

Файл backend/service/money-management-service/src/main/java/com/vladyslavberezovskyi/dao/repository/TransactionRepository.java

```
package com.vladyslavberezovskyi.dao.repository;

import com.vladyslavberezovskyi.dao.entity.CategoryEntity;
import com.vladyslavberezovskyi.dao.entity.TransactionEntity;
import org.springframework.data.jpa.repository.JpaRepository;
import org.springframework.stereotype.Repository;

import java.util.List;
import java.util.UUID;

@Repository
public interface TransactionRepository extends JpaRepository<TransactionEntity,
UUID> {

    List<TransactionEntity> findAllByCategory(CategoryEntity categoryEntity);

    List<TransactionEntity> findAllByBalanceId(UUID balanceId);
```

```
}
```

Файл backend/service/money-management-service/src/main/java/com/vladyslavberezovskyi/dto/balance/BalanceResponse.java

```
package com.vladyslavberezovskyi.dto.balance;

import lombok.AllArgsConstructor;
import lombok.Builder;
import lombok.Data;
import lombok.NoArgsConstructor;

import java.util.List;
import java.util.UUID;

@Data
@Builder
@NoArgsConstructor
@AllArgsConstructor
public class BalanceResponse {

    private UUID id;
    private String name;
    private Double currentValue;
    private List<UUID> transactions;

}
```

Файл backend/service/money-management-service/src/main/java/com/vladyslavberezovskyi/dto/balance/CreateBalanceRequest.java

```
package com.vladyslavberezovskyi.dto.balance;

import lombok.AllArgsConstructor;
import lombok.Builder;
import lombok.Data;
import lombok.NoArgsConstructor;

@Data
@Builder
@NoArgsConstructor
@AllArgsConstructor//TODO add validation
public class CreateBalanceRequest {

    private String name;
    private Double currentValue;

}
```

Файл backend/service/money-management-service/src/main/java/com/vladyslavberezovskyi/dto/balance/PatchBalanceRequest.java

```
package com.vladyslavberezovskyi.dto.balance;

import lombok.AllArgsConstructor;
import lombok.Builder;
import lombok.Data;
```

```
import lombok.NoArgsConstructor;

@Data
@Builder
@NoArgsConstructor
@AllArgsConstructor//TODO make objects nullable
      //TODO add validation
public class PatchBalanceRequest {

   private String name;
   private Double currentValue;

}
```

Файл backend/service/money-management-service/src/main/java/com/vladyslavberezovskyi/dto/budget/BudgetResponse.java

```
package com.vladyslavberezovskyi.dto.budget;

import lombok.AllArgsConstructor;
import lombok.Builder;
import lombok.Data;
import lombok.NoArgsConstructor;

import java.util.List;
import java.util.UUID;

@Data
@Builder
```

```
@NoArgsConstructor
@AllArgsConstructor
public class BudgetResponse {

    private UUID id;
    private String name;
    private Double currentValue;
    private Double neededValue;
    private List<UUID> transactions;
    private UUID category;
}
```

Файл backend/service/money-management-service/src/main/java/com/vladyslavberezovskyi/dto/budget/CreateBudgetRequest.java

```
package com.vladyslavberezovskyi.dto.budget;

import lombok.AllArgsConstructor;
import lombok.Builder;
import lombok.Data;
import lombok.NoArgsConstructor;

import java.util.UUID;

@Data
@Builder
@NoArgsConstructor
@AllArgsConstructor
public class CreateBudgetRequest {
```

```
    private String name;
    private Double currentValue;
    private Double neededValue;
    private UUID category;

}
```

Файл backend/service/money-management-service/src/main/java/com/vladyslavberezovskyi/dto/budget/PatchBudgetRequest.java

```
package com.vladyslavberezovskyi.dto.budget;

import lombok.AllArgsConstructor;
import lombok.Builder;
import lombok.Data;
import lombok.NoArgsConstructor;

import java.util.UUID;

@Data
@Builder
@NoArgsConstructor
@AllArgsConstructor
public class PatchBudgetRequest {

    private String name;
    private Double currentValue;
    private Double neededValue;
```

```
    private UUID category;

}
```

Файл backend/service/money-management-service/src/main/java/com/vladyslavberezovskyi/dto/category/CategoryResponse.java

```
package com.vladyslavberezovskyi.dto.category;

import lombok.AllArgsConstructor;
import lombok.Builder;
import lombok.Data;
import lombok.NoArgsConstructor;

import java.util.UUID;

@Data
@Builder
@NoArgsConstructor
@AllArgsConstructor//TODO
public class CategoryResponse {

    private UUID id;
    private String name;

}
```

Файл backend/service/money-management-service/src/main/java/com/vladyslavberezovskyi/dto/category/CreateCategoryRequest.java

```
package com.vladyslavberezovskyi.dto.category;

import lombok.AllArgsConstructor;
import lombok.Builder;
import lombok.Data;
import lombok.NoArgsConstructor;

import java.util.UUID;

@Data
@Builder
@NoArgsConstructor
@AllArgsConstructor//TODO
public class CreateCategoryRequest {

    private String name;

}
```

Файл backend/service/money-management-service/src/main/java/com/vladyslavberezovskyi/dto/category/PatchCategoryRequest.java

```
package com.vladyslavberezovskyi.dto.category;

import lombok.AllArgsConstructor;
import lombok.Builder;
```

```
import lombok.Data;
import lombok.NoArgsConstructor;

@Data
@Builder
@NoArgsConstructor
@AllArgsConstructor
public class PatchCategoryRequest {

    private String name;
}
```

Файл backend/service/money-management-service/src/main/java/com/vladyslavberezovskyi/dto/transaction/CreateTransactionRequest.java

```
package com.vladyslavberezovskyi.dto.transaction;

import lombok.AllArgsConstructor;
import lombok.Builder;
import lombok.Data;
import lombok.NoArgsConstructor;

import java.util.Date;
import java.util.UUID;

@Data
@Builder
@NoArgsConstructor
@AllArgsConstructor
```

```
public class CreateTransactionRequest {

    private String name;
    private Double value;
    private UUID balance;
    private UUID credit;
    private Date date;
    private UUID category;

}
```

Файл backend/service/money-management-service/src/main/java/com/vladyslavberezovskyi/dto/transaction/PatchTransactionRequest.java

```
package com.vladyslavberezovskyi.dto.transaction;

import lombok.AllArgsConstructor;
import lombok.Builder;
import lombok.Data;
import lombok.NoArgsConstructor;

import java.util.Date;
import java.util.UUID;

@Data
@Builder
@NoArgsConstructor
@AllArgsConstructor
public class PatchTransactionRequest {
```

```
    private String name;
    private Double value;
    private UUID balance;
    private UUID credit;
    private Date date;
    private UUID category;

}
```

Файл backend/service/money-management-service/src/main/java/com/vladyslavberezovskyi/dto/transaction/TransactionResponse.java

```
package com.vladyslavberezovskyi.dto.transaction;

import lombok.AllArgsConstructor;
import lombok.Builder;
import lombok.Data;
import lombok.NoArgsConstructor;

import java.util.Date;
import java.util.UUID;

@Data
@Builder
@NoArgsConstructor
@AllArgsConstructor
public class TransactionResponse {
```

```
    private UUID id;
    private String name;
    private Double value;
    private UUID balance;
    private UUID credit;
    private UUID category;
    private Date date;

}
```

Файл backend/service/money-management-service/src/main/java/com/vladyslavberezovskyi/mapper/BalanceMapper.java

```
package com.vladyslavberezovskyi.mapper;

import com.vladyslavberezovskyi.dao.entity.BalanceEntity;
import com.vladyslavberezovskyi.dao.entity.TransactionEntity;
import com.vladyslavberezovskyi.dto.balance.BalanceResponse;
import com.vladyslavberezovskyi.dto.balance.CreateBalanceRequest;
import com.vladyslavberezovskyi.dto.balance.PatchBalanceRequest;
import com.vladyslavberezovskyi.model.Balance;
import org.mapstruct.Mapper;
import org.mapstruct.Mapping;
import org.mapstruct.MappingTarget;
import org.springframework.data.domain.Page;
import org.springframework.lang.Nullable;

import java.util.List;
import java.util.UUID;
import java.util.stream.Collectors;
```

```
import static org.mapstruct.MappingConstants.ComponentModel.SPRING;

@Mapper(componentModel = SPRING, uses =
{TransactionMapper.class})//TODO map ignores
public interface BalanceMapper {

   BalanceResponse modelToDto(Balance balance);

   default List<BalanceResponse> modelToDto(@Nullable List<Balance>
balances) {
      if (balances == null) {
         return null;
      }

      return balances.stream()
            .map(this::modelToDto)
            .toList();
   }

   default Page<BalanceResponse> modelToDto(@Nullable Page<Balance>
balances) {
      if (balances == null) {
         return null;
      }

      return balances.map(this::modelToDto);
   }
```

```
    Balance dtoToModel(CreateBalanceRequest createBalanceRequest);

    Balance dtoToModel(PatchBalanceRequest patchBalanceRequest);

    Balance entityToModel(BalanceEntity balanceEntity);

    default List<Balance> entityToModel(@Nullable List<BalanceEntity>
balanceEntities) {
        if (balanceEntities == null) {
            return null;
        }

        return balanceEntities.stream()
                .map(this::entityToModel)
                .toList();
    }

    default Page<Balance> entityToModel(@Nullable Page<BalanceEntity>
balanceEntities) {
        if (balanceEntities == null) {
            return null;
        }

        return balanceEntities.map(this::entityToModel);
    }

    @Mapping(target = "transactions", ignore = true)
    BalanceEntity modelToEntity(Balance balance);
```

```
    @Mapping(target = "id", ignore = true)
    @Mapping(target = "transactions", ignore = true)
    void update(@MappingTarget BalanceEntity balanceEntity, Balance balance);
}
```

Файл backend/service/money-management-service/target/classes/com/vladyslavberezovskyi/mapper/BalanceMapperImpl.class

```
//
// Source code recreated from a .class file by IntelliJ IDEA
// (powered by FernFlower decompiler)
//

package com.vladyslavberezovskyi.mapper;

import com.vladyslavberezovskyi.dao.entity.BalanceEntity;
import com.vladyslavberezovskyi.dto.balance.BalanceResponse;
import com.vladyslavberezovskyi.dto.balance.CreateBalanceRequest;
import com.vladyslavberezovskyi.dto.balance.PatchBalanceRequest;
import com.vladyslavberezovskyi.model.Balance;
import java.util.ArrayList;
import java.util.List;
import java.util.UUID;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Component;

@Component
public class BalanceMapperImpl implements BalanceMapper {
    @Autowired
```

```
    private TransactionMapper transactionMapper;

    public BalanceMapperImpl() {
    }

    public BalanceResponse modelToDto(Balance balance) {
        if (balance == null) {
            return null;
        } else {
            BalanceResponse.BalanceResponseBuilder balanceResponse =
BalanceResponse.builder();
            balanceResponse.id(balance.getId());
            balanceResponse.name(balance.getName());
            balanceResponse.currentValue(balance.getCurrentValue());
            List<UUID> list = balance.getTransactions();
            if (list != null) {
                balanceResponse.transactions(new ArrayList(list));
            }

            return balanceResponse.build();
        }
    }

    public Balance dtoToModel(CreateBalanceRequest createBalanceRequest) {
        if (createBalanceRequest == null) {
            return null;
        } else {
            Balance.BalanceBuilder balance = Balance.builder();
            balance.name(createBalanceRequest.getName());
```

```
            balance.currentValue(createBalanceRequest.getCurrentValue());
            return balance.build();
        }
    }

    public Balance dtoToModel(PatchBalanceRequest patchBalanceRequest) {
        if (patchBalanceRequest == null) {
            return null;
        } else {
            Balance.BalanceBuilder balance = Balance.builder();
            balance.name(patchBalanceRequest.getName());
            balance.currentValue(patchBalanceRequest.getCurrentValue());
            return balance.build();
        }
    }

    public Balance entityToModel(BalanceEntity balanceEntity) {
        if (balanceEntity == null) {
            return null;
        } else {
            Balance.BalanceBuilder balance = Balance.builder();
            balance.id(balanceEntity.getId());
            balance.name(balanceEntity.getName());
            balance.currentValue(balanceEntity.getCurrentValue());
            balance.userId(balanceEntity.getUserId());

balance.transactions(this.transactionMapper.getIds(balanceEntity.getTransactions()
));
            return balance.build();
```

```
        }
    }

    public BalanceEntity modelToEntity(Balance balance) {
        if (balance == null) {
            return null;
        } else {
            BalanceEntity.BalanceEntityBuilder balanceEntity =
BalanceEntity.builder();
            balanceEntity.name(balance.getName());
            balanceEntity.currentValue(balance.getCurrentValue());
            balanceEntity.userId(balance.getUserId());
            return balanceEntity.build();
        }
    }

    public void update(BalanceEntity balanceEntity, Balance balance) {
        if (balance != null) {
            balanceEntity.setName(balance.getName());
            balanceEntity.setCurrentValue(balance.getCurrentValue());
            balanceEntity.setUserId(balance.getUserId());
        }
    }
}
```

Файл backend/service/money-management-service/src/main/java/com/vladyslavberezovskyi/mapper/BudgetMapper.java

```
package com.vladyslavberezovskyi.mapper;
```

```
import com.vladyslavberezovskyi.dao.entity.BudgetEntity;
import com.vladyslavberezovskyi.dto.budget.BudgetResponse;
import com.vladyslavberezovskyi.dto.budget.CreateBudgetRequest;
import com.vladyslavberezovskyi.dto.budget.PatchBudgetRequest;
import com.vladyslavberezovskyi.model.Budget;
import org.mapstruct.Mapper;
import org.mapstruct.Mapping;
import org.mapstruct.MappingTarget;
import org.springframework.data.domain.Page;
import org.springframework.lang.Nullable;

import java.util.List;
import java.util.UUID;

import static org.mapstruct.MappingConstants.ComponentModel.SPRING;

@Mapper(componentModel = SPRING, uses =
{TransactionMapper.class})//TODO map ignores
public interface BudgetMapper {
   BudgetResponse modelToDto(Budget budget);

   default List<BudgetResponse> modelToDto(@Nullable List<Budget> budgets)
{
       if (budgets == null) {
           return null;
       }

       return budgets.stream()
               .map(this::modelToDto)
```

```
            .toList();
    }

    default Page<BudgetResponse> modelToDto(@Nullable Page<Budget>
budgets) {
        if (budgets == null) {
            return null;
        }

        return budgets.map(this::modelToDto);
    }

    Budget dtoToModel(CreateBudgetRequest createBudgetRequest);

    Budget dtoToModel(PatchBudgetRequest patchBudgetRequest);

    @Mapping(target = "category", source = "category.id")
    Budget entityToModel(BudgetEntity budgetEntity);

    default List<Budget> entityToModel(@Nullable List<BudgetEntity>
budgetEntities){
        if (budgetEntities == null) {
            return null;
        }

        return budgetEntities.stream()
                .map(this::entityToModel)
                .toList();
    }
```

```
    default Page<Budget> entityToModel(@Nullable Page<BudgetEntity>
budgetEntities){
        if (budgetEntities == null) {
            return null;
        }

        return budgetEntities.map(this::entityToModel);
    }

    @Mapping(target = "category", ignore = true)
    BudgetEntity modelToEntity(Budget budget);

    @Mapping(target = "id", ignore = true)
    @Mapping(target = "currentValue", ignore = true)
    @Mapping(target = "category", ignore = true)
    void update(@MappingTarget BudgetEntity budgetEntity, Budget budget);

    default List<UUID> getIds(@Nullable List<BudgetEntity> entities) {
        if (entities == null) {
            return null;
        }

        return entities.stream()
                .map(BudgetEntity::getId)
                .toList();
    }
}
```

Файл backend/service/money-management-service/target/classes/com/vladyslavberezovskyi/mapper/BudgetMapperImpl.class

```
//
// Source code recreated from a .class file by IntelliJ IDEA
// (powered by FernFlower decompiler)
//

package com.vladyslavberezovskyi.mapper;

import com.vladyslavberezovskyi.dao.entity.BudgetEntity;
import com.vladyslavberezovskyi.dao.entity.CategoryEntity;
import com.vladyslavberezovskyi.dto.budget.BudgetResponse;
import com.vladyslavberezovskyi.dto.budget.CreateBudgetRequest;
import com.vladyslavberezovskyi.dto.budget.PatchBudgetRequest;
import com.vladyslavberezovskyi.model.Budget;
import java.util.ArrayList;
import java.util.List;
import java.util.UUID;
import org.springframework.stereotype.Component;

@Component
public class BudgetMapperImpl implements BudgetMapper {
   public BudgetMapperImpl() {
   }

   public BudgetResponse modelToDto(Budget budget) {
      if (budget == null) {
         return null;
```

```
        } else {
            BudgetResponse.BudgetResponseBuilder budgetResponse =
BudgetResponse.builder();
            budgetResponse.id(budget.getId());
            budgetResponse.name(budget.getName());
            budgetResponse.currentValue(budget.getCurrentValue());
            budgetResponse.neededValue(budget.getNeededValue());
            List<UUID> list = budget.getTransactions();
            if (list != null) {
               budgetResponse.transactions(new ArrayList(list));
            }

            budgetResponse.category(budget.getCategory());
            return budgetResponse.build();
        }
    }

    public Budget dtoToModel(CreateBudgetRequest createBudgetRequest) {
        if (createBudgetRequest == null) {
            return null;
        } else {
            Budget.BudgetBuilder budget = Budget.builder();
            budget.name(createBudgetRequest.getName());
            budget.currentValue(createBudgetRequest.getCurrentValue());
            budget.neededValue(createBudgetRequest.getNeededValue());
            budget.category(createBudgetRequest.getCategory());
            return budget.build();
        }
    }
```

```
public Budget dtoToModel(PatchBudgetRequest patchBudgetRequest) {
    if (patchBudgetRequest == null) {
        return null;
    } else {
        Budget.BudgetBuilder budget = Budget.builder();
        budget.name(patchBudgetRequest.getName());
        budget.currentValue(patchBudgetRequest.getCurrentValue());
        budget.neededValue(patchBudgetRequest.getNeededValue());
        budget.category(patchBudgetRequest.getCategory());
        return budget.build();
    }
}

public Budget entityToModel(BudgetEntity budgetEntity) {
    if (budgetEntity == null) {
        return null;
    } else {
        Budget.BudgetBuilder budget = Budget.builder();
        budget.category(this.budgetEntityCategoryId(budgetEntity));
        budget.id(budgetEntity.getId());
        budget.name(budgetEntity.getName());
        budget.currentValue(budgetEntity.getCurrentValue());
        budget.neededValue(budgetEntity.getNeededValue());
        budget.userId(budgetEntity.getUserId());
        return budget.build();
    }
}
```

```
public BudgetEntity modelToEntity(Budget budget) {
    if (budget == null) {
        return null;
    } else {
        BudgetEntity.BudgetEntityBuilder budgetEntity = BudgetEntity.builder();
        budgetEntity.name(budget.getName());
        budgetEntity.currentValue(budget.getCurrentValue());
        budgetEntity.neededValue(budget.getNeededValue());
        budgetEntity.userId(budget.getUserId());
        return budgetEntity.build();
    }
}

public void update(BudgetEntity budgetEntity, Budget budget) {
    if (budget != null) {
        budgetEntity.setName(budget.getName());
        budgetEntity.setNeededValue(budget.getNeededValue());
        budgetEntity.setUserId(budget.getUserId());
    }
}

private UUID budgetEntityCategoryId(BudgetEntity budgetEntity) {
    if (budgetEntity == null) {
        return null;
    } else {
        CategoryEntity category = budgetEntity.getCategory();
        if (category == null) {
            return null;
        } else {
```

```
            UUID id = category.getId();
            return id == null ? null : id;
        }
    }
  }
}
```

Файл backend/service/money-management-service/src/main/java/com/vladyslavberezovskyi/mapper/CategoryMapper.java

```
package com.vladyslavberezovskyi.mapper;

import com.vladyslavberezovskyi.dao.entity.CategoryEntity;
import com.vladyslavberezovskyi.dto.category.CategoryResponse;
import com.vladyslavberezovskyi.dto.category.CreateCategoryRequest;
import com.vladyslavberezovskyi.dto.category.PatchCategoryRequest;
import com.vladyslavberezovskyi.model.Category;
import org.mapstruct.Mapper;
import org.mapstruct.Mapping;
import org.mapstruct.MappingTarget;
import org.springframework.data.domain.Page;
import org.springframework.lang.Nullable;

import java.util.List;

import static org.mapstruct.MappingConstants.ComponentModel.SPRING;

@Mapper(componentModel = SPRING)//TODO map ignores
public interface CategoryMapper {
```

```
    CategoryResponse modelToDto(Category category);

    default List<CategoryResponse> modelToDto(@Nullable List<Category>
categories) {
        if (categories == null) {
            return null;
        }

        return categories.stream()
                .map(this::modelToDto)
                .toList();
    }

    default Page<CategoryResponse> modelToDto(@Nullable Page<Category>
categories) {
        if (categories == null) {
            return null;
        }

        return categories.map(this::modelToDto);
    }

    Category dtoToModel(CreateCategoryRequest createCategoryRequest);

    Category dtoToModel(PatchCategoryRequest patchCategoryRequest);

    Category entityToModel(CategoryEntity categoryEntity);
```

```
    default List<Category> entityToModel(@Nullable List<CategoryEntity>
categoryEntities) {
        if (categoryEntities == null) {
            return null;
        }

        return categoryEntities.stream()
                .map(this::entityToModel)
                .toList();
    }

    default Page<Category> entityToModel(@Nullable Page<CategoryEntity>
categoryEntities) {
        if (categoryEntities == null) {
            return null;
        }

        return categoryEntities.map(this::entityToModel);
    }

    CategoryEntity modelToEntity(Category category);

    @Mapping(target = "id", ignore = true)
    void update(@MappingTarget CategoryEntity categoryEntity, Category
category);
}
```

Файл backend/service/money-management-service/target/classes/com/vladyslavberezovskyi/mapper/CategoryMapperImpl.class

```
//
// Source code recreated from a .class file by IntelliJ IDEA
// (powered by FernFlower decompiler)
//

package com.vladyslavberezovskyi.mapper;

import com.vladyslavberezovskyi.dao.entity.CategoryEntity;
import com.vladyslavberezovskyi.dto.category.CategoryResponse;
import com.vladyslavberezovskyi.dto.category.CreateCategoryRequest;
import com.vladyslavberezovskyi.dto.category.PatchCategoryRequest;
import com.vladyslavberezovskyi.model.Category;
import org.springframework.stereotype.Component;

@Component
public class CategoryMapperImpl implements CategoryMapper {
   public CategoryMapperImpl() {
   }

   public CategoryResponse modelToDto(Category category) {
      if (category == null) {
         return null;
      } else {
         CategoryResponse.CategoryResponseBuilder categoryResponse =
CategoryResponse.builder();
         categoryResponse.id(category.getId());
```

```
        categoryResponse.name(category.getName());
        return categoryResponse.build();
    }
}

public Category dtoToModel(CreateCategoryRequest createCategoryRequest) {
    if (createCategoryRequest == null) {
        return null;
    } else {
        Category.CategoryBuilder category = Category.builder();
        category.name(createCategoryRequest.getName());
        return category.build();
    }
}

public Category dtoToModel(PatchCategoryRequest patchCategoryRequest) {
    if (patchCategoryRequest == null) {
        return null;
    } else {
        Category.CategoryBuilder category = Category.builder();
        category.name(patchCategoryRequest.getName());
        return category.build();
    }
}

public Category entityToModel(CategoryEntity categoryEntity) {
    if (categoryEntity == null) {
        return null;
    } else {
```

```
            Category.CategoryBuilder category = Category.builder();
            category.id(categoryEntity.getId());
            category.name(categoryEntity.getName());
            return category.build();
        }
    }

    public CategoryEntity modelToEntity(Category category) {
        if (category == null) {
            return null;
        } else {
            CategoryEntity.CategoryEntityBuilder categoryEntity =
CategoryEntity.builder();
            categoryEntity.name(category.getName());
            return categoryEntity.build();
        }
    }

    public void update(CategoryEntity categoryEntity, Category category) {
        if (category != null) {
            categoryEntity.setName(category.getName());
        }
    }
}
```

Файл backend/service/money-management-service/src/main/java/com/vladyslavberezovskyi/mapper/TransactionMapper.java

```
package com.vladyslavberezovskyi.mapper;
```

```
import com.vladyslavberezovskyi.dao.entity.TransactionEntity;
import com.vladyslavberezovskyi.dto.transaction.CreateTransactionRequest;
import com.vladyslavberezovskyi.dto.transaction.PatchTransactionRequest;
import com.vladyslavberezovskyi.dto.transaction.TransactionResponse;
import com.vladyslavberezovskyi.model.Transaction;
import org.mapstruct.Mapper;
import org.mapstruct.Mapping;
import org.mapstruct.MappingTarget;
import org.springframework.data.domain.Page;
import org.springframework.lang.Nullable;

import java.util.List;
import java.util.Optional;
import java.util.UUID;

import static org.mapstruct.MappingConstants.ComponentModel.SPRING;

@Mapper(componentModel = SPRING, uses = {BudgetMapper.class})//TODO
map ignores
public interface TransactionMapper {
   TransactionResponse modelToDto(Transaction transaction);

   default List<TransactionResponse> modelToDto(@Nullable List<Transaction>
transactions) {
       if (transactions == null) {
          return null;
       }
```

```
        return transactions.stream()
                .map(this::modelToDto)
                .toList();
    }

    default Page<TransactionResponse> modelToDto(@Nullable
Page<Transaction> transactions) {
        if (transactions == null) {
            return null;
        }

        return transactions.map(this::modelToDto);
    }

    Transaction dtoToModel(CreateTransactionRequest createTransactionRequest);

    Transaction dtoToModel(PatchTransactionRequest patchTransactionRequest);

    @Mapping(target = "balance", source = "balance.id")
    @Mapping(target = "category", source = "category.id")
    Transaction entityToModel(TransactionEntity transactionEntity);

    default List<Transaction> entityToModel(@Nullable List<TransactionEntity>
transactionEntities) {
        if (transactionEntities == null) {
            return null;
        }

        return transactionEntities.stream()
```

```
                .map(this::entityToModel)
                .toList();
    }

    default Page<Transaction> entityToModel(@Nullable Page<TransactionEntity>
transactionEntities) {
        if (transactionEntities == null) {
            return null;
        }

        return transactionEntities.map(this::entityToModel);
    }

    @Mapping(target = "balance", ignore = true)
    @Mapping(target = "category", ignore = true)
    TransactionEntity modelToEntity(Transaction transaction);

    @Mapping(target = "id", ignore = true)
    @Mapping(target = "balance", ignore = true)
    @Mapping(target = "category", ignore = true)
    void update(@MappingTarget TransactionEntity transactionEntity, Transaction
transaction);

    default List<UUID> getIds(@Nullable List<TransactionEntity> entities) {
        if (entities == null) {
            return null;
        }

        return entities.stream()
```

```
            .map(TransactionEntity::getId)
            .toList();
    }
}
```

Файл backend/service/money-management-service/target/classes/com/vladyslavberezovskyi/mapper/TransactionMapperImpl.class

```
//
// Source code recreated from a .class file by IntelliJ IDEA
// (powered by FernFlower decompiler)
//

package com.vladyslavberezovskyi.mapper;

import com.vladyslavberezovskyi.dao.entity.BalanceEntity;
import com.vladyslavberezovskyi.dao.entity.CategoryEntity;
import com.vladyslavberezovskyi.dao.entity.TransactionEntity;
import com.vladyslavberezovskyi.dto.transaction.CreateTransactionRequest;
import com.vladyslavberezovskyi.dto.transaction.PatchTransactionRequest;
import com.vladyslavberezovskyi.dto.transaction.TransactionResponse;
import com.vladyslavberezovskyi.model.Transaction;
import java.util.UUID;
import org.springframework.stereotype.Component;

@Component
public class TransactionMapperImpl implements TransactionMapper {
    public TransactionMapperImpl() {
    }
```

```
    public TransactionResponse modelToDto(Transaction transaction) {
        if (transaction == null) {
            return null;
        } else {
            TransactionResponse.TransactionResponseBuilder transactionResponse =
TransactionResponse.builder();
            transactionResponse.id(transaction.getId());
            transactionResponse.name(transaction.getName());
            transactionResponse.value(transaction.getValue());
            transactionResponse.balance(transaction.getBalance());
            transactionResponse.credit(transaction.getCredit());
            transactionResponse.category(transaction.getCategory());
            transactionResponse.date(transaction.getDate());
            return transactionResponse.build();
        }
    }

    public Transaction dtoToModel(CreateTransactionRequest
createTransactionRequest) {
        if (createTransactionRequest == null) {
            return null;
        } else {
            Transaction.TransactionBuilder transaction = Transaction.builder();
            transaction.name(createTransactionRequest.getName());
            transaction.value(createTransactionRequest.getValue());
            transaction.balance(createTransactionRequest.getBalance());
            transaction.credit(createTransactionRequest.getCredit());
            transaction.date(createTransactionRequest.getDate());
```

```
            transaction.category(createTransactionRequest.getCategory());
            return transaction.build();
        }
    }

    public Transaction dtoToModel(PatchTransactionRequest
patchTransactionRequest) {
        if (patchTransactionRequest == null) {
            return null;
        } else {
            Transaction.TransactionBuilder transaction = Transaction.builder();
            transaction.name(patchTransactionRequest.getName());
            transaction.value(patchTransactionRequest.getValue());
            transaction.balance(patchTransactionRequest.getBalance());
            transaction.credit(patchTransactionRequest.getCredit());
            transaction.date(patchTransactionRequest.getDate());
            transaction.category(patchTransactionRequest.getCategory());
            return transaction.build();
        }
    }

    public Transaction entityToModel(TransactionEntity transactionEntity) {
        if (transactionEntity == null) {
            return null;
        } else {
            Transaction.TransactionBuilder transaction = Transaction.builder();
            transaction.balance(this.transactionEntityBalanceId(transactionEntity));
            transaction.category(this.transactionEntityCategoryId(transactionEntity));
            transaction.id(transactionEntity.getId());
```

```
            transaction.name(transactionEntity.getName());
            transaction.value(transactionEntity.getValue());
            transaction.userId(transactionEntity.getUserId());
            transaction.date(transactionEntity.getDate());
            return transaction.build();
        }
    }

    public TransactionEntity modelToEntity(Transaction transaction) {
        if (transaction == null) {
            return null;
        } else {
            TransactionEntity.TransactionEntityBuilder transactionEntity =
TransactionEntity.builder();
            transactionEntity.name(transaction.getName());
            transactionEntity.value(transaction.getValue());
            transactionEntity.userId(transaction.getUserId());
            transactionEntity.date(transaction.getDate());
            return transactionEntity.build();
        }
    }

    public void update(TransactionEntity transactionEntity, Transaction transaction)
{
        if (transaction != null) {
            transactionEntity.setName(transaction.getName());
            transactionEntity.setValue(transaction.getValue());
            transactionEntity.setUserId(transaction.getUserId());
            transactionEntity.setDate(transaction.getDate());
```

```
        }
    }

    private UUID transactionEntityBalanceId(TransactionEntity transactionEntity) {
        if (transactionEntity == null) {
            return null;
        } else {
            BalanceEntity balance = transactionEntity.getBalance();
            if (balance == null) {
                return null;
            } else {
                UUID id = balance.getId();
                return id == null ? null : id;
            }
        }
    }

    private UUID transactionEntityCategoryId(TransactionEntity transactionEntity)
{
        if (transactionEntity == null) {
            return null;
        } else {
            CategoryEntity category = transactionEntity.getCategory();
            if (category == null) {
                return null;
            } else {
                UUID id = category.getId();
                return id == null ? null : id;
            }
```

```
        }
    }
}
```

Файл backend/service/money-management-service/src/main/java/com/vladyslavberezovskyi/model/Balance.java

```
package com.vladyslavberezovskyi.model;

import lombok.AllArgsConstructor;
import lombok.Builder;
import lombok.Data;
import lombok.NoArgsConstructor;

import java.util.List;
import java.util.UUID;

@Data
@Builder
@NoArgsConstructor
@AllArgsConstructor
public class Balance {

    private UUID id;
    private String name;
    private Double currentValue;
    private UUID userId;
    private List<UUID> transactions;

}
```

Файл backend/service/money-management-service/src/main/java/com/vladyslavberezovskyi/model/Budget.java

```
package com.vladyslavberezovskyi.model;

import lombok.AllArgsConstructor;
import lombok.Builder;
import lombok.Data;
import lombok.NoArgsConstructor;

import java.util.List;
import java.util.UUID;

@Data
@Builder
@NoArgsConstructor
@AllArgsConstructor
public class Budget {

    private UUID id;
    private String name;
    private Double currentValue;
    private Double neededValue;
    private UUID userId;
    private List<UUID> transactions;
    private UUID category;

}
```

Файл backend/service/money-management-service/src/main/java/com/vladyslavberezovskyi/model/Category.java

```
package com.vladyslavberezovskyi.model;

import lombok.AllArgsConstructor;
import lombok.Builder;
import lombok.Data;
import lombok.NoArgsConstructor;

import java.util.UUID;

@Data
@Builder
@NoArgsConstructor
@AllArgsConstructor//TODO
public class Category {

    private UUID id;
    private String name;

}
```

Файл backend/service/money-management-service/src/main/java/com/vladyslavberezovskyi/model/Credit.java

```
package com.vladyslavberezovskyi.model;

import lombok.AllArgsConstructor;
import lombok.Builder;
import lombok.Data;
```

```
import lombok.NoArgsConstructor;

import java.util.UUID;

@Data
@Builder
@NoArgsConstructor
@AllArgsConstructor//TODO
public class Credit {

    private UUID id;
}
```

Файл backend/service/money-management-service/src/main/java/com/vladyslavberezovskyi/model/Transaction.java

```
package com.vladyslavberezovskyi.model;

import lombok.AllArgsConstructor;
import lombok.Builder;
import lombok.Data;
import lombok.NoArgsConstructor;

import java.util.Date;
import java.util.UUID;

@Data
@Builder
@NoArgsConstructor
@AllArgsConstructor
```

```
public class Transaction {

    private UUID id;
    private String name;
    private Double value;
    private UUID balance;
    private UUID credit;
    private UUID userId;
    private Date date;
    private UUID category;

}
```

Файл backend/service/money-management-service/src/main/java/com/vladyslavberezovskyi/service/BalanceService.java

```
package com.vladyslavberezovskyi.service;

import com.vladyslavberezovskyi.model.Balance;
import org.springframework.data.domain.Page;
import org.springframework.data.domain.Pageable;

import java.util.List;
import java.util.UUID;

public interface BalanceService {

    Balance getBalanceById(UUID balanceId);

    List<Balance> getAllBalances();
```

```
    Page<Balance> getAllBalances(Pageable pageable);

    Balance createBalance(Balance balance);

    Balance updateBalance(UUID balanceId, Balance balance);

    void deleteBalance(UUID balanceId);
}
```

Файл backend/service/money-management-service/src/main/java/com/vladyslavberezovskyi/service/impl/BalanceServiceImpl.java

```
package com.vladyslavberezovskyi.service.impl;

import com.vladyslavberezovskyi.dao.entity.BalanceEntity;
import com.vladyslavberezovskyi.dao.entity.TransactionEntity;
import com.vladyslavberezovskyi.dao.repository.BalanceRepository;
import com.vladyslavberezovskyi.dao.repository.TransactionRepository;
import com.vladyslavberezovskyi.error.ResourceNotFoundException;
import com.vladyslavberezovskyi.mapper.BalanceMapper;
import com.vladyslavberezovskyi.model.Balance;
import com.vladyslavberezovskyi.service.BalanceService;
import lombok.RequiredArgsConstructor;
import org.springframework.data.domain.Page;
import org.springframework.data.domain.Pageable;
import org.springframework.stereotype.Service;

import java.util.List;
```

```
import java.util.UUID;

@Service
@RequiredArgsConstructor //TODO add userId to queries
public class BalanceServiceImpl implements BalanceService {

    private final BalanceRepository repository;
    private final BalanceMapper mapper;
    private final TransactionRepository transactionRepository;

    @Override
    public Balance getBalanceById(UUID balanceId) {
        return mapper.entityToModel(repository.findById(balanceId)
                .orElseThrow(ResourceNotFoundException::new));
    }

    @Override
    public List<Balance> getAllBalances() {
        return mapper.entityToModel(repository.findAll());
    }

    @Override
    public Page<Balance> getAllBalances(Pageable pageable) {
        return mapper.entityToModel(repository.findAll(pageable));
    }

    @Override
    public Balance createBalance(Balance balance) {
        BalanceEntity balanceEntity = mapper.modelToEntity(balance);
```

```
        balanceEntity = repository.saveAndFlush(balanceEntity);

        return mapper.entityToModel(balanceEntity);
    }

    @Override
    public Balance updateBalance(UUID balanceId, Balance balance) {
        BalanceEntity balanceEntity = repository.findById(balanceId)
                .orElseThrow(ResourceNotFoundException::new);
        mapper.update(balanceEntity, balance);
        repository.saveAndFlush(balanceEntity);

        return mapper.entityToModel(balanceEntity);
    }

    @Override
    public void deleteBalance(UUID balanceId) {
        BalanceEntity balanceEntity = repository.findById(balanceId)
                .orElseThrow(ResourceNotFoundException::new);

        List<TransactionEntity> transactionEntities =
transactionRepository.findAllByBalanceId(balanceId);
        transactionRepository.deleteAllInBatch(transactionEntities);

        repository.delete(balanceEntity);
    }
}
```

Файл backend/service/money-management-service/src/main/java/com/vladyslavberezovskyi/service/BudgetService.java

```
package com.vladyslavberezovskyi.service;

import com.vladyslavberezovskyi.model.Budget;
import org.springframework.data.domain.Page;
import org.springframework.data.domain.Pageable;

import java.util.List;
import java.util.UUID;

public interface BudgetService {

    Budget getBudgetById(UUID budgetId);

    List<Budget> getAllBudgets();

    Page<Budget> getAllBudgets(Pageable pageable);

    Budget createBudget(Budget budget);

    Budget updateBudget(UUID budgetId, Budget budget);

    void deleteBudget(UUID budgetId);
}
```

Файл backend/service/money-management-service/src/main/java/com/vladyslavberezovskyi/service/impl/BudgetServiceImpl.java

```
package com.vladyslavberezovskyi.service.impl;

import com.vladyslavberezovskyi.dao.entity.BudgetEntity;
import com.vladyslavberezovskyi.dao.entity.CategoryEntity;
import com.vladyslavberezovskyi.dao.entity.TransactionEntity;
import com.vladyslavberezovskyi.dao.repository.BudgetRepository;
import com.vladyslavberezovskyi.dao.repository.CategoryRepository;
import com.vladyslavberezovskyi.dao.repository.TransactionRepository;
import com.vladyslavberezovskyi.error.ResourceNotFoundException;
import com.vladyslavberezovskyi.mapper.BudgetMapper;
import com.vladyslavberezovskyi.model.Budget;
import com.vladyslavberezovskyi.security.UserDetails;
import com.vladyslavberezovskyi.service.BudgetService;
import com.vladyslavberezovskyi.util.EmailSender;
import lombok.RequiredArgsConstructor;
import org.springframework.data.domain.Page;
import org.springframework.data.domain.Pageable;
import org.springframework.security.core.context.SecurityContextHolder;
import org.springframework.stereotype.Service;

import java.util.List;
import java.util.UUID;

@Service
@RequiredArgsConstructor//TODO add userId to queries
public class BudgetServiceImpl implements BudgetService {

    private final BudgetMapper mapper;
    private final BudgetRepository repository;
```

```
    private final CategoryRepository categoryRepository;
    private final TransactionRepository transactionRepository;
    private final EmailSender emailSender;

    @Override
    public Budget getBudgetById(UUID budgetId) {
        return mapper.entityToModel(repository.findById(budgetId)
                .orElseThrow(ResourceNotFoundException::new));
    }

    @Override
    public List<Budget> getAllBudgets() {
        return mapper.entityToModel(repository.findAll());
    }

    @Override
    public Page<Budget> getAllBudgets(Pageable pageable) {
        return mapper.entityToModel(repository.findAll(pageable));
    }

    @Override
    public Budget createBudget(Budget budget) {
        BudgetEntity budgetEntity = mapper.modelToEntity(budget);

        CategoryEntity categoryEntity =
categoryRepository.findById(budget.getCategory())
                        .orElseThrow(ResourceNotFoundException::new);
        budgetEntity.setCategory(categoryEntity);
```

```
        List<TransactionEntity> transactionEntities =
transactionRepository.findAllByCategory(categoryEntity);

        for (TransactionEntity transactionEntity : transactionEntities) {
            budgetEntity.setCurrentValue(budgetEntity.getCurrentValue() -
transactionEntity.getValue());
        }

        budgetEntity = repository.saveAndFlush(budgetEntity);

        return mapper.entityToModel(budgetEntity);
    }

    @Override
    public Budget updateBudget(UUID budgetId, Budget budget) {
        BudgetEntity budgetEntity = repository.findById(budgetId)
                .orElseThrow(ResourceNotFoundException::new);
        mapper.update(budgetEntity, budget);
        repository.saveAndFlush(budgetEntity);

        if (budgetEntity.getCurrentValue() >= budgetEntity.getNeededValue()) {
            String email = ((UserDetails) SecurityContextHolder.getContext()
                    .getAuthentication()
                    .getPrincipal())
                    .getUsername();
            try {
                emailSender.sendWarning(email,
                        budgetEntity.getName(),
```

```
                    budgetEntity.getCurrentValue() + "/" +
budgetEntity.getNeededValue());
            } catch (Exception e) {
                throw new RuntimeException(e);
            }
        }

        return mapper.entityToModel(budgetEntity);
    }

    @Override
    public void deleteBudget(UUID budgetId) {
        BudgetEntity budgetEntity = repository.findById(budgetId)
                .orElseThrow(ResourceNotFoundException::new);

        repository.delete(budgetEntity);
    }
}
```

Файл backend/service/money-management-service/src/main/java/com/vladyslavberezovskyi/service/CategoryService.java

```
package com.vladyslavberezovskyi.service;

import com.vladyslavberezovskyi.model.Category;
import org.springframework.data.domain.Page;
import org.springframework.data.domain.Pageable;

import java.util.List;
import java.util.UUID;
```

```
public interface CategoryService {

    Category getCategoryById(UUID categoryId);

    List<Category> getAllCategories();

    Page<Category> getAllCategories(Pageable pageable);

    Category createCategory(Category category);

    Category updateCategory(UUID categoryId, Category category);

    void deleteCategory(UUID categoryId);
}
```

Файл backend/service/money-management-service/src/main/java/com/vladyslavberezovskyi/service/impl/CategoryServiceImpl.java

```
package com.vladyslavberezovskyi.service.impl;

import com.vladyslavberezovskyi.dao.entity.BalanceEntity;
import com.vladyslavberezovskyi.dao.entity.BudgetEntity;
import com.vladyslavberezovskyi.dao.entity.CategoryEntity;
import com.vladyslavberezovskyi.dao.entity.TransactionEntity;
import com.vladyslavberezovskyi.dao.repository.BudgetRepository;
import com.vladyslavberezovskyi.dao.repository.CategoryRepository;
import com.vladyslavberezovskyi.dao.repository.TransactionRepository;
import com.vladyslavberezovskyi.error.ResourceNotFoundException;
```

```
import com.vladyslavberezovskyi.mapper.CategoryMapper;
import com.vladyslavberezovskyi.model.Category;
import com.vladyslavberezovskyi.service.CategoryService;
import lombok.RequiredArgsConstructor;
import org.springframework.data.domain.Page;
import org.springframework.data.domain.Pageable;
import org.springframework.stereotype.Service;

import java.util.HashSet;
import java.util.List;
import java.util.Set;
import java.util.UUID;
import java.util.stream.Collectors;

@Service
@RequiredArgsConstructor//TODO add userId to queries
public class CategoryServiceImpl implements CategoryService {

    private final CategoryRepository repository;
    private final CategoryMapper mapper;
    private final TransactionRepository transactionRepository;
    private final BudgetRepository budgetRepository;

    @Override
    public Category getCategoryById(UUID categoryId) {
        return mapper.entityToModel(repository.findById(categoryId)
                .orElseThrow(ResourceNotFoundException::new));
    }
```

```
@Override
public List<Category> getAllCategories() {
    return mapper.entityToModel(repository.findAll());
}

@Override
public Page<Category> getAllCategories(Pageable pageable) {
    return mapper.entityToModel(repository.findAll(pageable));
}

@Override
public Category createCategory(Category category) {
    CategoryEntity categoryEntity = mapper.modelToEntity(category);
    categoryEntity = repository.saveAndFlush(categoryEntity);

    return mapper.entityToModel(categoryEntity);
}

@Override
public Category updateCategory(UUID categoryId, Category category) {
    CategoryEntity categoryEntity = repository.findById(categoryId)
            .orElseThrow(ResourceNotFoundException::new);
    mapper.update(categoryEntity, category);
    categoryEntity = repository.saveAndFlush(categoryEntity);

    return mapper.entityToModel(categoryEntity);
}

@Override
```

```
    public void deleteCategory(UUID categoryId) {
        CategoryEntity categoryEntity = repository.findById(categoryId)
                .orElseThrow(ResourceNotFoundException::new);

        List<TransactionEntity> transactionEntities =
transactionRepository.findAllByCategory(categoryEntity);
        transactionRepository.deleteAll(transactionEntities);

        repository.delete(categoryEntity);
    }
}
```

Файл backend/service/money-management-service/src/main/java/com/vladyslavberezovskyi/service/TransactionService.java

```
package com.vladyslavberezovskyi.service;

import com.vladyslavberezovskyi.model.Transaction;
import org.springframework.data.domain.Page;
import org.springframework.data.domain.Pageable;

import java.util.List;
import java.util.UUID;

public interface TransactionService {

    Transaction getTransactionById(UUID transactionId);

    List<Transaction> getAllTransactions(UUID balanceId);
```

```
    Page<Transaction> getAllTransactions(Pageable pageable);

    Transaction createTransaction(Transaction transaction);

    Transaction updateTransaction(UUID transactionId, Transaction transaction);

    void deleteTransaction(UUID transactionId);
}
```

Файл backend/service/money-management-service/src/main/java/com/vladyslavberezovskyi/service/impl/TransactionServiceImpl.java

```
package com.vladyslavberezovskyi.service.impl;

import com.vladyslavberezovskyi.dao.entity.BalanceEntity;
import com.vladyslavberezovskyi.dao.entity.BudgetEntity;
import com.vladyslavberezovskyi.dao.entity.CategoryEntity;
import com.vladyslavberezovskyi.dao.entity.TransactionEntity;
import com.vladyslavberezovskyi.dao.repository.BalanceRepository;
import com.vladyslavberezovskyi.dao.repository.BudgetRepository;
import com.vladyslavberezovskyi.dao.repository.CategoryRepository;
import com.vladyslavberezovskyi.dao.repository.TransactionRepository;
import com.vladyslavberezovskyi.error.ResourceNotFoundException;
import com.vladyslavberezovskyi.mapper.TransactionMapper;
import com.vladyslavberezovskyi.model.Transaction;
import com.vladyslavberezovskyi.security.UserDetails;
import com.vladyslavberezovskyi.service.TransactionService;
import com.vladyslavberezovskyi.util.EmailSender;
```

```
import lombok.RequiredArgsConstructor;
import org.springframework.data.domain.Page;
import org.springframework.data.domain.Pageable;
import org.springframework.security.core.context.SecurityContextHolder;
import org.springframework.stereotype.Service;

import java.util.List;
import java.util.UUID;

@Service
@RequiredArgsConstructor//TODO add userId to queries
public class TransactionServiceImpl implements TransactionService {

    private final TransactionRepository repository;
    private final TransactionMapper mapper;
    private final CategoryRepository categoryRepository;
    private final BudgetRepository budgetRepository;
    private final BalanceRepository balanceRepository;
    private final EmailSender emailSender;

    @Override
    public Transaction getTransactionById(UUID transactionId) {
        return mapper.entityToModel(repository.findById(transactionId)
                .orElseThrow(ResourceNotFoundException::new));
    }

    @Override
    public List<Transaction> getAllTransactions(UUID balanceId) {
        if (balanceId != null) {
```

```
            return mapper.entityToModel(repository.findAllByBalanceId(balanceId));
        } else {
            return mapper.entityToModel(repository.findAll());
        }
    }

    @Override
    public Page<Transaction> getAllTransactions(Pageable pageable) {
        return mapper.entityToModel(repository.findAll(pageable));
    }

    @Override
    public Transaction createTransaction(Transaction transaction) {
        BalanceEntity balanceEntity =
balanceRepository.findById(transaction.getBalance())
                .orElseThrow(ResourceNotFoundException::new);

        if ((balanceEntity.getCurrentValue() + transaction.getValue()) >= 0) {
            balanceEntity.setCurrentValue(balanceEntity.getCurrentValue() +
transaction.getValue());
            balanceRepository.saveAndFlush(balanceEntity);
        } else {
            throw new RuntimeException();
        }

        TransactionEntity transactionEntity = mapper.modelToEntity(transaction);

        transactionEntity.setBalance(balanceEntity);
```

```
        CategoryEntity categoryEntity =
categoryRepository.findById(transaction.getCategory())
                .orElseThrow(ResourceNotFoundException::new);
        transactionEntity.setCategory(categoryEntity);

        List<BudgetEntity> budgetEntities =
budgetRepository.findAllByCategory(categoryEntity);
        budgetEntities.forEach(budgetEntity ->
                budgetEntity.setCurrentValue(budgetEntity.getCurrentValue() -
transaction.getValue()));
        budgetRepository.saveAllAndFlush(budgetEntities);

        budgetEntities.forEach(budgetEntity -> {
           if (budgetEntity.getCurrentValue() >= budgetEntity.getNeededValue()) {
              String email = ((UserDetails) SecurityContextHolder.getContext()
                    .getAuthentication()
                    .getPrincipal())
                    .getUsername();
              try {
                 emailSender.sendWarning(email,
                       budgetEntity.getName(),
                       budgetEntity.getCurrentValue() + "/" +
budgetEntity.getNeededValue());
              } catch (Exception e) {
                 throw new RuntimeException(e);
              }
           }
        });
```

```
        transactionEntity = repository.saveAndFlush(transactionEntity);

        return mapper.entityToModel(transactionEntity);
    }

    @Override
    public Transaction updateTransaction(UUID transactionId, Transaction
transaction) {
        TransactionEntity transactionEntity = repository.findById(transactionId)
                .orElseThrow(ResourceNotFoundException::new);

        List<BudgetEntity> oldBudgetEntities =
budgetRepository.findAllByCategory(transactionEntity.getCategory());
        for (BudgetEntity oldBudgetEntity : oldBudgetEntities) {
            oldBudgetEntity.setCurrentValue(oldBudgetEntity.getCurrentValue() +
transactionEntity.getValue());
        }
        budgetRepository.saveAllAndFlush(oldBudgetEntities);

        BalanceEntity oldBalanceEntity =
balanceRepository.findById(transactionEntity.getBalance().getId())
                .orElseThrow(ResourceNotFoundException::new);
        oldBalanceEntity.setCurrentValue(oldBalanceEntity.getCurrentValue() -
transactionEntity.getValue());
        balanceRepository.saveAndFlush(oldBalanceEntity);

        mapper.update(transactionEntity, transaction);
```

```
        CategoryEntity newCategoryEntity =
categoryRepository.findById(transaction.getCategory())
                .orElseThrow(ResourceNotFoundException::new);

        List<BudgetEntity> newBudgetEntities =
budgetRepository.findAllByCategory(newCategoryEntity);
        newBudgetEntities.forEach(budgetEntity ->
budgetEntity.setCurrentValue(budgetEntity.getCurrentValue() -
transaction.getValue()));
        budgetRepository.saveAllAndFlush(newBudgetEntities);

        newBudgetEntities.forEach(budgetEntity -> {
            if (budgetEntity.getCurrentValue() >= budgetEntity.getNeededValue()) {
                String email = ((UserDetails) SecurityContextHolder.getContext()
                        .getAuthentication()
                        .getPrincipal())
                        .getUsername();
                try {
                    emailSender.sendWarning(email,
                            budgetEntity.getName(),
                            budgetEntity.getCurrentValue() + "/" +
budgetEntity.getNeededValue());
                } catch (Exception e) {
                    throw new RuntimeException(e);
                }
            }
        });
```

```
        BalanceEntity newBalanceEntity =
balanceRepository.findById(transaction.getBalance())
                .orElseThrow(ResourceNotFoundException::new);
        newBalanceEntity.setCurrentValue(newBalanceEntity.getCurrentValue() +
transaction.getValue());
        balanceRepository.saveAndFlush(newBalanceEntity);

        transactionEntity.setBalance(newBalanceEntity);
        transactionEntity.setCategory(newCategoryEntity);

        transactionEntity = repository.saveAndFlush(transactionEntity);

        return mapper.entityToModel(transactionEntity);
    }

    @Override
    public void deleteTransaction(UUID transactionId) {
        TransactionEntity transactionEntity = repository.findById(transactionId)
                .orElseThrow(ResourceNotFoundException::new);

        List<BudgetEntity> oldBudgetEntities =
budgetRepository.findAllByCategory(transactionEntity.getCategory());
        for (BudgetEntity oldBudgetEntity : oldBudgetEntities) {
            oldBudgetEntity.setCurrentValue(oldBudgetEntity.getCurrentValue() +
transactionEntity.getValue());
        }
        budgetRepository.saveAllAndFlush(oldBudgetEntities);
```

```
        BalanceEntity oldBalanceEntity =
balanceRepository.findById(transactionEntity.getBalance().getId())
                .orElseThrow(ResourceNotFoundException::new);
        oldBalanceEntity.setCurrentValue(oldBalanceEntity.getCurrentValue() -
transactionEntity.getValue());
        balanceRepository.saveAndFlush(oldBalanceEntity);

        repository.delete(transactionEntity);
    }
}
```

Файл backend/service/money-management-service/src/main/java/com/vladyslavberezovskyi/MoneyManagementService.java

```
package com.vladyslavberezovskyi;

import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;

@SpringBootApplication
public class MoneyManagementService {

    public static void main(String[] args) {
        SpringApplication.run(MoneyManagementService.class, args);
    }

}
```

Файл backend/service/money-management-service/src/main/resources/sql/ddl/000_create_balance_table.sql

```
CREATE TABLE balance (
  id UUID,
  name VARCHAR(255),
  current_value DOUBLE PRECISION,
  user_id UUID,
  CONSTRAINT pk_balance PRIMARY KEY (id)
);
```

Файл backend/service/money-management-service/src/main/resources/sql/ddl/001_create_category_table.sql

```
CREATE TABLE category (
  id UUID NOT NULL,
  name VARCHAR(255),
  user_id UUID,
  CONSTRAINT pk_category PRIMARY KEY (id)
);
```

Файл backend/service/money-management-service/src/main/resources/sql/ddl/002_create_budget_table.sql

```
CREATE TABLE budget (
 id UUID NOT NULL,
  name VARCHAR(255),
  current_value DOUBLE PRECISION,
  needed_value DOUBLE PRECISION,
  user_id UUID,
  category_id UUID,
  CONSTRAINT pk_budget PRIMARY KEY (id)
```

```
);

ALTER TABLE budget ADD CONSTRAINT FK_BUDGET_ON_CATEGORY
FOREIGN KEY (category_id) REFERENCES category (id);
```

Файл backend/service/money-management-service/src/main/resources/sql/ddl/003_create_transaction_table.sql

```
CREATE TABLE transaction (
  id UUID NOT NULL,
   name VARCHAR(255),
   value DOUBLE PRECISION,
   user_id UUID,
   date TIMESTAMP WITHOUT TIME ZONE,
   balance_id UUID,
   category_id UUID,
   CONSTRAINT pk_transaction PRIMARY KEY (id)
);

ALTER TABLE transaction ADD CONSTRAINT
FK_TRANSACTION_ON_BALANCE FOREIGN KEY (balance_id)
REFERENCES balance (id);

ALTER TABLE transaction ADD CONSTRAINT
FK_TRANSACTION_ON_CATEGORY FOREIGN KEY (category_id)
REFERENCES category (id);
```

Файл backend/service/money-management-service/src/main/resources/sql/changelog-master.yaml

```
databaseChangeLog:
```

```
  - include:
      file: sql/ddl/000_create_balance_table.sql
      context: local,default,integration-test
  - include:
      file: sql/ddl/001_create_category_table.sql
      context: local,default,integration-test
  - include:
      file: sql/ddl/002_create_budget_table.sql
      context: local,default,integration-test
  - include:
      file: sql/ddl/003_create_transaction_table.sql
      context: local,default,integration-test
```

Файл backend/service/money-management-service/src/main/resources/application.yaml

```
spring:
  liquibase:
    change-log: classpath:sql/changelog-master.yaml
    contexts: ${spring.profiles.active:default}

email:
  username: ${EMAIL_USERNAME}
  confirmation:
    url: ${EMAIL_CONFIRMATION_URL}
    path: /api/v1/invitations/confirmInvitation
```

Файл backend/service/money-management-service/src/main/resources/application-local.yaml

```
spring:
```

```
  application:
    name: money-management-service-local
  datasource:
    username: postgres
    password: postgres
    #?stringtype=unspecified need to convert string into enum type in db
    url: jdbc:postgresql://localhost:5432/money-management-
service?stringtype=unspecified

email:
  username: test
  confirmation:
    url: http://localhost:8080
    path: /api/v1/invitations/confirmInvitation
```

Файл backend/service/money-management-service/Dockerfile

```
ARG BASE_NAME=base

FROM ${BASE_NAME} AS base
FROM maven:3.8.4 AS builder

COPY --from=base /app/ /app/lib/
COPY src /app/src
COPY pom.xml /app

RUN mvn install:install-file -Dfile=/app/lib/pom.xml -
DgroupId=com.vladyslavberezovskyi -DartifactId=base -Dversion=0.0.1-
SNAPSHOT -Dpackaging=pom
```

```
RUN mvn install:install-file -Dfile=/app/lib/commons-0.0.1-SNAPSHOT.jar -
DgroupId=com.vladyslavberezovskyi -DartifactId=commons -Dversion=0.0.1-
SNAPSHOT -Dpackaging=jar

RUN mvn -f /app/pom.xml clean package -DskipTests

FROM amazoncorretto:17

WORKDIR /app

COPY --from=builder /app/target/*.jar /app/app.jar

CMD ["java", "-jar", "app.jar"]
```

Файл backend/service/money-management-service/pom.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
https://maven.apache.org/xsd/maven-4.0.0.xsd">
    <modelVersion>4.0.0</modelVersion>
    <parent>
        <groupId>com.vladyslavberezovskyi</groupId>
        <artifactId>base</artifactId>
        <version>0.0.1-SNAPSHOT</version>
        <relativePath/> <!-- lookup parent from repository -->
    </parent>
    <artifactId>money-management</artifactId>
    <version>0.0.1-SNAPSHOT</version>
```

```
<name>money-management-service</name>
<properties>
    <java.version>17</java.version>
</properties>

<dependencies>
    <dependency>
        <groupId>com.vladyslavberezovskyi</groupId>
        <artifactId>commons</artifactId>
        <version>0.0.1-SNAPSHOT</version>
    </dependency>
</dependencies>

<build>
    <plugins>
        <plugin>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-maven-plugin</artifactId>
            <configuration>
                <excludes>
                    <exclude>
                        <groupId>org.projectlombok</groupId>
                        <artifactId>lombok</artifactId>
                    </exclude>
                </excludes>
            </configuration>
        </plugin>
    </plugins>
</build>
```

```
</project>
```

Файл backend/service/user-service/src/main/java/com/vladyslavberezovskyi/controller/InvitationController.java

```
package com.vladyslavberezovskyi.controller;

import com.vladyslavberezovskyi.dto.invitation.InvitationResponse;
import
com.vladyslavberezovskyi.dto.invitation.PhoneNumberConfirmationRequest;
import com.vladyslavberezovskyi.mapper.InvitationMapper;
import com.vladyslavberezovskyi.service.InvitationService;
import lombok.RequiredArgsConstructor;
import org.springframework.web.bind.annotation.*;

import java.util.UUID;

@RestController
@RequiredArgsConstructor
@RequestMapping(path = "/api/v1/invitations")
public class InvitationController {

    private final InvitationMapper invitationMapper;
    private final InvitationService invitationService;

    @PostMapping(path = "/confirmPhoneNumber/{id}")
    public InvitationResponse confirmPhoneNumber(@PathVariable UUID id) {
```

```
        return
invitationMapper.modelToDto(invitationService.confirmPhoneNumber(id));
    }

    @PostMapping(path = "/sendPhoneNumberConfirmationRequest")
    public InvitationResponse
sendPhoneNumberConfirmationRequest(@RequestBody
PhoneNumberConfirmationRequest phoneNumberConfirmationRequest) {
        return
invitationMapper.modelToDto(invitationService.sendPhoneNumberConfirmationR
equest(invitationMapper.dtoToModel(phoneNumberConfirmationRequest)));
    }

    @GetMapping(path = "/{id}")
    public InvitationResponse getInvitationById(@PathVariable UUID id) {
        return
invitationMapper.modelToDto(invitationService.getInvitationById(id));
    }

    @GetMapping(path = "/confirmInvitation/{id}")
    public InvitationResponse confirmInvitation(@PathVariable UUID id) {
        return
invitationMapper.modelToDto(invitationService.confirmInvitation(id));
    }
}
```

Файл backend/service/user-service/src/main/java/com/vladyslavberezovskyi/controller/UserController.java

```
package com.vladyslavberezovskyi.controller;

import com.vladyslavberezovskyi.dto.user.AuthUserRequest;
import com.vladyslavberezovskyi.dto.user.CreateUserRequest;
import com.vladyslavberezovskyi.dto.user.UpdateUserRequest;
import com.vladyslavberezovskyi.dto.user.UserResponse;
import com.vladyslavberezovskyi.mapper.UserMapper;
import com.vladyslavberezovskyi.security.TokenResponse;
import com.vladyslavberezovskyi.security.annotations.IsUser;
import com.vladyslavberezovskyi.service.UserService;
import io.swagger.v3.oas.annotations.security.SecurityRequirement;
import lombok.RequiredArgsConstructor;
import org.springframework.web.bind.annotation.*;

@RestController
@RequiredArgsConstructor
@RequestMapping(path = "/api/v1")
public class UserController {

    private final UserService service;
    private final UserMapper mapper;

    @IsUser
    @GetMapping("/user")
    @SecurityRequirement(name = "Authorization")
    public UserResponse getUser() {
        return mapper.modelToDto(service.getUserById());
    }
```

```
    @PostMapping("/users")
    private UserResponse createUser(@RequestBody CreateUserRequest
createUserRequest) {
        return
mapper.modelToDto(service.createUser(mapper.dtoToModel(createUserRequest))
);
    }

    @PostMapping("/users/auth")
    public TokenResponse authenticateUser(@RequestBody AuthUserRequest
authUserRequest) {
        return service.auth(mapper.dtoToModel(authUserRequest));
    }

    @PatchMapping("/user")
    @SecurityRequirement(name = "Authorization")
    public UserResponse updateUser(@RequestBody UpdateUserRequest
updateUserRequest) {
        return
mapper.modelToDto(service.updateUser(mapper.dtoToModel(updateUserRequest)
));
    }
}
```

Файл backend/service/user-service/src/main/java/com/vladyslavberezovskyi/dao/entity/InvitationEntity.java

```
package com.vladyslavberezovskyi.dao.entity;
```

```
import com.vladyslavberezovskyi.entity.BaseEntity;
import jakarta.persistence.Column;
import jakarta.persistence.Entity;
import jakarta.persistence.OneToOne;
import lombok.*;

@Builder
@NoArgsConstructor
@Getter
@Setter
@Entity(name = "invitation")
@AllArgsConstructor
public class InvitationEntity extends BaseEntity {

    private String email;
    private boolean emailConfirmed;
    private String phoneNumber;
    private boolean phoneNumberConfirmed;

    @OneToOne(mappedBy = "invitation")
    private UserEntity user;

}
```

Файл backend/service/user-service/src/main/java/com/vladyslavberezovskyi/dao/entity/UserEntity.java

```
package com.vladyslavberezovskyi.dao.entity;

import com.vladyslavberezovskyi.security.enums.Role;
```

```
import com.vladyslavberezovskyi.entity.BaseEntity;
import jakarta.persistence.*;
import lombok.*;

@Builder
@NoArgsConstructor
@Getter
@Setter
@Entity(name = "user")
@Table(name = "\"user\"")
@AllArgsConstructor
public class UserEntity extends BaseEntity {

    private String email;
    private String nickname;
    private String password;
    private String phoneNumber;
    @Enumerated(EnumType.STRING)
    private Role role = Role.USER;
    private boolean enabled;

    @OneToOne(cascade = CascadeType.ALL, orphanRemoval = true)
    @JoinColumn(name = "invitation_id")
    private InvitationEntity invitation;

}
```

Файл backend/service/user-service/src/main/java/com/vladyslavberezovskyi/dao/repository/InvitationRepository.java

```
package com.vladyslavberezovskyi.dao.repository;

import com.vladyslavberezovskyi.dao.entity.InvitationEntity;
import org.springframework.data.jpa.repository.JpaRepository;

import java.util.UUID;

public interface InvitationRepository extends JpaRepository<InvitationEntity, UUID> {
    InvitationEntity findByEmail(String phoneNumber);
}
```

Файл backend/service/user-service/src/main/java/com/vladyslavberezovskyi/dao/repository/UserRepository.java

```
package com.vladyslavberezovskyi.dao.repository;

import com.vladyslavberezovskyi.dao.entity.UserEntity;
import org.springframework.data.jpa.repository.JpaRepository;

import java.util.Optional;
import java.util.UUID;

public interface UserRepository extends JpaRepository<UserEntity, UUID> {

    Optional<UserEntity> findByEmail(String email);
```

```
    boolean existsByEmail(String email);

    boolean existsByPhoneNumber(String phoneNumber);
}
```

Файл backend/service/user-service/src/main/java/com/vladyslavberezovskyi/dto/invitation/InvitationResponse.java

```
package com.vladyslavberezovskyi.dto.invitation;

import lombok.AllArgsConstructor;
import lombok.Builder;
import lombok.Data;
import lombok.NoArgsConstructor;

import java.util.UUID;

@Data
@Builder
@NoArgsConstructor
@AllArgsConstructor
public class InvitationResponse {

    private UUID id;
    private String email;
    private boolean emailConfirmed;
    private String phoneNumber;
    private boolean phoneNumberConfirmed;
```

```
    private UUID user;

}
```

Файл backend/service/user-service/src/main/java/com/vladyslavberezovskyi/dto/invitation/PhoneNumberConfirmationRequest.java

```
package com.vladyslavberezovskyi.dto.invitation;

import lombok.AllArgsConstructor;
import lombok.Builder;
import lombok.Data;
import lombok.NoArgsConstructor;

@Data
@Builder
@NoArgsConstructor
@AllArgsConstructor
public class PhoneNumberConfirmationRequest {

    private String phoneNumber;

}
```

Файл backend/service/user-service/src/main/java/com/vladyslavberezovskyi/dto/user/AuthUserRequest.java

```
package com.vladyslavberezovskyi.dto.user;
```

```
import lombok.AllArgsConstructor;
import lombok.Builder;
import lombok.Data;
import lombok.NoArgsConstructor;

@Data
@Builder
@NoArgsConstructor
@AllArgsConstructor
public class AuthUserRequest {

    private String email;
    private String password;

}
```

Файл backend/service/user-service/src/main/java/com/vladyslavberezovskyi/dto/user/CreateUserRequest.java

```
package com.vladyslavberezovskyi.dto.user;

import lombok.AllArgsConstructor;
import lombok.Builder;
import lombok.Data;
import lombok.NoArgsConstructor;

@Data
@Builder
@NoArgsConstructor
```

```
@AllArgsConstructor
public class CreateUserRequest {

    private String email;
    private String password;
    private String phoneNumber;

}
```

Файл backend/service/user-service/src/main/java/com/vladyslavberezovskyi/dto/user/UpdateUserRequest.java

```
package com.vladyslavberezovskyi.dto.user;

import lombok.AllArgsConstructor;
import lombok.Builder;
import lombok.Data;
import lombok.NoArgsConstructor;

@Data
@Builder
@NoArgsConstructor
@AllArgsConstructor
public class UpdateUserRequest {

    private String name;
    private String email;
    private String password;
    private String phoneNumber;
```

```
}
```

Файл backend/service/user-service/src/main/java/com/vladyslavberezovskyi/dto/user/UserResponse.java

```
package com.vladyslavberezovskyi.dto.user;

import lombok.AllArgsConstructor;
import lombok.Builder;
import lombok.Data;
import lombok.NoArgsConstructor;

@Data
@Builder
@NoArgsConstructor
@AllArgsConstructor
public class UpdateUserRequest {

    private String name;
    private String email;
    private String password;
    private String phoneNumber;

}
```

Файл backend/service/user-service/src/main/java/com/vladyslavberezovskyi/error/UserAlreadyExistsException.java

```
package com.vladyslavberezovskyi.error;
```

```
public class UserAlreadyExistsException extends BaseException {
   //TODO add reasons or codes
}
```

Файл backend/service/user-service/src/main/java/com/vladyslavberezovskyi/mapper/InvitationMapper.java

```
package com.vladyslavberezovskyi.mapper;

import com.vladyslavberezovskyi.dao.entity.InvitationEntity;
import com.vladyslavberezovskyi.dto.invitation.InvitationResponse;
import
com.vladyslavberezovskyi.dto.invitation.PhoneNumberConfirmationRequest;
import com.vladyslavberezovskyi.model.Invitation;
import org.mapstruct.Mapper;
import org.mapstruct.Mapping;

import static org.mapstruct.MappingConstants.ComponentModel.SPRING;

@Mapper(componentModel = SPRING)
public interface InvitationMapper {

   InvitationResponse modelToDto(Invitation invitation);

   @Mapping(target = "user", source = "user.id")
   Invitation entityToModel(InvitationEntity invitationEntity);
```

```
    Invitation dtoToModel(PhoneNumberConfirmationRequest
phoneNumberConfirmationRequest);

    @Mapping(target = "user", ignore = true)
    InvitationEntity modelToEntity(Invitation invitation);
}
```

Файл backend/service/user-service/target/classes/com/vladyslavberezovskyi/mapper/InvitationMapperImpl.class

```
//
// Source code recreated from a .class file by IntelliJ IDEA
// (powered by FernFlower decompiler)
//

package com.vladyslavberezovskyi.mapper;

import com.vladyslavberezovskyi.dao.entity.InvitationEntity;
import com.vladyslavberezovskyi.dao.entity.UserEntity;
import com.vladyslavberezovskyi.dto.invitation.InvitationResponse;
import
com.vladyslavberezovskyi.dto.invitation.PhoneNumberConfirmationRequest;
import com.vladyslavberezovskyi.model.Invitation;
import java.util.UUID;
import org.springframework.stereotype.Component;

@Component
public class InvitationMapperImpl implements InvitationMapper {
    public InvitationMapperImpl() {
```

```
    }

    public InvitationResponse modelToDto(Invitation invitation) {
        if (invitation == null) {
            return null;
        } else {
            InvitationResponse.InvitationResponseBuilder invitationResponse =
InvitationResponse.builder();
            invitationResponse.id(invitation.getId());
            invitationResponse.email(invitation.getEmail());
            invitationResponse.emailConfirmed(invitation.isEmailConfirmed());
            invitationResponse.phoneNumber(invitation.getPhoneNumber());

invitationResponse.phoneNumberConfirmed(invitation.isPhoneNumberConfirmed
());
            invitationResponse.user(invitation.getUser());
            return invitationResponse.build();
        }
    }

    public Invitation entityToModel(InvitationEntity invitationEntity) {
        if (invitationEntity == null) {
            return null;
        } else {
            Invitation.InvitationBuilder invitation = Invitation.builder();
            invitation.user(this.invitationEntityUserId(invitationEntity));
            invitation.id(invitationEntity.getId());
            invitation.email(invitationEntity.getEmail());
            invitation.emailConfirmed(invitationEntity.isEmailConfirmed());
```

```
            invitation.phoneNumber(invitationEntity.getPhoneNumber());

invitation.phoneNumberConfirmed(invitationEntity.isPhoneNumberConfirmed());
            return invitation.build();
        }
    }

    public Invitation dtoToModel(PhoneNumberConfirmationRequest
phoneNumberConfirmationRequest) {
        if (phoneNumberConfirmationRequest == null) {
            return null;
        } else {
            Invitation.InvitationBuilder invitation = Invitation.builder();

invitation.phoneNumber(phoneNumberConfirmationRequest.getPhoneNumber());
            return invitation.build();
        }
    }

    public InvitationEntity modelToEntity(Invitation invitation) {
        if (invitation == null) {
            return null;
        } else {
            InvitationEntity.InvitationEntityBuilder invitationEntity =
InvitationEntity.builder();
            invitationEntity.email(invitation.getEmail());
            invitationEntity.emailConfirmed(invitation.isEmailConfirmed());
            invitationEntity.phoneNumber(invitation.getPhoneNumber());
```

```
invitationEntity.phoneNumberConfirmed(invitation.isPhoneNumberConfirmed());
            return invitationEntity.build();
        }
    }

    private UUID invitationEntityUserId(InvitationEntity invitationEntity) {
        if (invitationEntity == null) {
            return null;
        } else {
            UserEntity user = invitationEntity.getUser();
            if (user == null) {
                return null;
            } else {
                UUID id = user.getId();
                return id == null ? null : id;
            }
        }
    }
}
```

Файл backend/service/user-service/src/main/java/com/vladyslavberezovskyi/mapper/UserMapper.java

```
package com.vladyslavberezovskyi.mapper;

import com.vladyslavberezovskyi.dto.user.AuthUserRequest;
import com.vladyslavberezovskyi.dto.user.CreateUserRequest;
import com.vladyslavberezovskyi.dto.user.UpdateUserRequest;
import com.vladyslavberezovskyi.dto.user.UserResponse;
```

```
import com.vladyslavberezovskyi.dao.entity.UserEntity;
import com.vladyslavberezovskyi.model.User;
import org.mapstruct.Mapper;
import org.mapstruct.Mapping;
import org.mapstruct.MappingTarget;

import static org.mapstruct.MappingConstants.ComponentModel.SPRING;

@Mapper(componentModel = SPRING)
public interface UserMapper {

    UserResponse modelToDto(User user);

    @Mapping(target = "name", source = "nickname")
    User entityToModel(UserEntity userEntity);

    User dtoToModel(CreateUserRequest createUserRequest);

    User dtoToModel(UpdateUserRequest updateUserRequest);

    UserEntity modelToEntity(User user);

    User dtoToModel(AuthUserRequest authUserRequest);

    @Mapping(target = "id", ignore = true)
    @Mapping(target = "nickname", source = "name")
    @Mapping(target = "password", ignore = true)
    @Mapping(target = "role", ignore = true)
    void update(@MappingTarget UserEntity userEntity, User user);
```

```
}
```

Файл backend/service/user-service/target/classes/com/vladyslavberezovskyi/mapper/UserMapperImpl.class

```
//
// Source code recreated from a .class file by IntelliJ IDEA
// (powered by FernFlower decompiler)
//

package com.vladyslavberezovskyi.mapper;

import com.vladyslavberezovskyi.dao.entity.UserEntity;
import com.vladyslavberezovskyi.dto.user.AuthUserRequest;
import com.vladyslavberezovskyi.dto.user.CreateUserRequest;
import com.vladyslavberezovskyi.dto.user.UpdateUserRequest;
import com.vladyslavberezovskyi.dto.user.UserResponse;
import com.vladyslavberezovskyi.model.User;
import org.springframework.stereotype.Component;

@Component
public class UserMapperImpl implements UserMapper {
   public UserMapperImpl() {
   }

   public UserResponse modelToDto(User user) {
      if (user == null) {
         return null;
      } else {
```

```
            UserResponse.UserResponseBuilder userResponse =
UserResponse.builder();
            userResponse.id(user.getId());
            userResponse.name(user.getName());
            userResponse.email(user.getEmail());
            userResponse.phoneNumber(user.getPhoneNumber());
            return userResponse.build();
        }
    }

    public User entityToModel(UserEntity userEntity) {
        if (userEntity == null) {
            return null;
        } else {
            User.UserBuilder user = User.builder();
            user.name(userEntity.getNickname());
            user.id(userEntity.getId());
            user.email(userEntity.getEmail());
            user.password(userEntity.getPassword());
            user.phoneNumber(userEntity.getPhoneNumber());
            return user.build();
        }
    }

    public User dtoToModel(CreateUserRequest createUserRequest) {
        if (createUserRequest == null) {
            return null;
        } else {
            User.UserBuilder user = User.builder();
```

```
        user.email(createUserRequest.getEmail());
        user.password(createUserRequest.getPassword());
        user.phoneNumber(createUserRequest.getPhoneNumber());
        return user.build();
    }
}

public User dtoToModel(UpdateUserRequest updateUserRequest) {
    if (updateUserRequest == null) {
        return null;
    } else {
        User.UserBuilder user = User.builder();
        user.name(updateUserRequest.getName());
        user.email(updateUserRequest.getEmail());
        user.password(updateUserRequest.getPassword());
        user.phoneNumber(updateUserRequest.getPhoneNumber());
        return user.build();
    }
}

public UserEntity modelToEntity(User user) {
    if (user == null) {
        return null;
    } else {
        UserEntity.UserEntityBuilder userEntity = UserEntity.builder();
        userEntity.email(user.getEmail());
        userEntity.password(user.getPassword());
        userEntity.phoneNumber(user.getPhoneNumber());
        return userEntity.build();
```

```
        }
    }

    public User dtoToModel(AuthUserRequest authUserRequest) {
        if (authUserRequest == null) {
            return null;
        } else {
            User.UserBuilder user = User.builder();
            user.email(authUserRequest.getEmail());
            user.password(authUserRequest.getPassword());
            return user.build();
        }
    }

    public void update(UserEntity userEntity, User user) {
        if (user != null) {
            userEntity.setNickname(user.getName());
            userEntity.setEmail(user.getEmail());
            userEntity.setPhoneNumber(user.getPhoneNumber());
        }
    }
}
```

Файл backend/service/user-service/src/main/java/com/vladyslavberezovskyi/model/Invitation.java

```
package com.vladyslavberezovskyi.model;

import lombok.AllArgsConstructor;
import lombok.Builder;
```

```
import lombok.Data;
import lombok.NoArgsConstructor;

import java.util.UUID;

@Data
@Builder
@NoArgsConstructor
@AllArgsConstructor
public class Invitation {

    private UUID id;
    private String email;
    private boolean emailConfirmed;
    private String phoneNumber;
    private boolean phoneNumberConfirmed;
    private UUID user;

}
```

Файл backend/service/user-service/src/main/java/com/vladyslavberezovskyi/model/User.java

```
package com.vladyslavberezovskyi.model;

import lombok.AllArgsConstructor;
import lombok.Builder;
import lombok.Data;
import lombok.NoArgsConstructor;
```

```
import java.util.UUID;

@Data
@Builder
@NoArgsConstructor
@AllArgsConstructor
public class User {

    private UUID id;
    private String name;
    private String email;
    private String password;
    private String phoneNumber;

}
```

Файл backend/service/user-service/src/main/java/com/vladyslavberezovskyi/service/InvitationService.java

```
package com.vladyslavberezovskyi.service;

import com.vladyslavberezovskyi.dao.entity.UserEntity;
import
com.vladyslavberezovskyi.dto.invitation.PhoneNumberConfirmationRequest;
import com.vladyslavberezovskyi.model.Invitation;

import java.util.UUID;

public interface InvitationService {
```

```
    void sendInvitation(UserEntity savedUser);

    Invitation getInvitationById(UUID id);

    Invitation confirmPhoneNumber(UUID id);

    Invitation sendPhoneNumberConfirmationRequest(Invitation invitation);

    Invitation confirmInvitation(UUID id);
}
```

Файл backend/service/user-service/src/main/java/com/vladyslavberezovskyi/service/impl/InvitationServiceImpl.java

```
package com.vladyslavberezovskyi.service.impl;

import com.vladyslavberezovskyi.dao.entity.InvitationEntity;
import com.vladyslavberezovskyi.dao.entity.UserEntity;
import com.vladyslavberezovskyi.dao.repository.InvitationRepository;
import com.vladyslavberezovskyi.dao.repository.UserRepository;
import com.vladyslavberezovskyi.error.ResourceNotFoundException;
import com.vladyslavberezovskyi.error.UserAlreadyExistsException;
import com.vladyslavberezovskyi.mapper.InvitationMapper;
import com.vladyslavberezovskyi.model.Invitation;
import com.vladyslavberezovskyi.service.InvitationService;
import com.vladyslavberezovskyi.util.EmailSender;
import lombok.RequiredArgsConstructor;
import org.springframework.stereotype.Service;
```

```
import java.util.UUID;

@Service
@RequiredArgsConstructor
public class InvitationServiceImpl implements InvitationService {

    private final InvitationRepository invitationRepository;
    private final InvitationMapper mapper;
    private final UserRepository userRepository;
    private final EmailSender emailSender;

    @Override
    public void sendInvitation(UserEntity user) {
        InvitationEntity invitationEntity = new InvitationEntity();
        invitationEntity.setEmail(user.getEmail());

        user.setInvitation(invitationEntity);

        invitationEntity = invitationRepository.saveAndFlush(invitationEntity);
        userRepository.saveAndFlush(user);

        try {
            emailSender.sendInvitation(user.getEmail(), invitationEntity.getId());
        } catch (Exception e) {
            userRepository.delete(user);
            invitationRepository.delete(invitationEntity);
        }
    }
```

```
    @Override
    public Invitation getInvitationById(UUID id) {
        InvitationEntity invitationEntity = invitationRepository.findById(id)
                .orElseThrow(ResourceNotFoundException::new);

        return mapper.entityToModel(invitationEntity);
    }

    @Override
    public Invitation confirmPhoneNumber(UUID id) {
        InvitationEntity invitationEntity = invitationRepository.findById(id)
                .orElseThrow(ResourceNotFoundException::new);

        invitationEntity.setPhoneNumberConfirmed(true);
        invitationEntity = invitationRepository.saveAndFlush(invitationEntity);

        return mapper.entityToModel(invitationEntity);
    }

    @Override
    public Invitation sendPhoneNumberConfirmationRequest(Invitation invitation)
{
        if (userRepository.existsByPhoneNumber(invitation.getPhoneNumber())) {
            throw new UserAlreadyExistsException();
        }

        InvitationEntity invitationEntity = mapper.modelToEntity(invitation);
        invitationEntity = invitationRepository.saveAndFlush(invitationEntity);
```

```
        //TODO send msg

        return mapper.entityToModel(invitationEntity);
    }

    @Override
    public Invitation confirmInvitation(UUID id) {
        InvitationEntity invitationEntity = invitationRepository.findById(id)
                .orElseThrow(ResourceNotFoundException::new);

        invitationEntity.setEmailConfirmed(true);
        invitationEntity = invitationRepository.saveAndFlush(invitationEntity);

        UserEntity user = invitationEntity.getUser();
        user.setEnabled(true);
        userRepository.saveAndFlush(user);

        return mapper.entityToModel(invitationEntity);
    }

}
```

Файл backend/service/user-service/src/main/java/com/vladyslavberezovskyi/service/UserService.java

```
package com.vladyslavberezovskyi.service;

import com.vladyslavberezovskyi.model.User;
import com.vladyslavberezovskyi.security.TokenResponse;
```

```
import java.util.UUID;

public interface UserService {
    User getUserById();

    User createUser(User user);

    TokenResponse auth(User model);

    User updateUser(User user);
}
```

Файл backend/service/user-service/src/main/java/com/vladyslavberezovskyi/service/impl/UserServiceImpl.java

```
package com.vladyslavberezovskyi.service.impl;

import com.vladyslavberezovskyi.dao.entity.InvitationEntity;
import com.vladyslavberezovskyi.dao.entity.UserEntity;
import com.vladyslavberezovskyi.dao.repository.InvitationRepository;
import com.vladyslavberezovskyi.dao.repository.UserRepository;
import com.vladyslavberezovskyi.error.ForbiddenException;
import com.vladyslavberezovskyi.error.ResourceNotFoundException;
import com.vladyslavberezovskyi.error.UserAlreadyExistsException;
import com.vladyslavberezovskyi.mapper.UserMapper;
import com.vladyslavberezovskyi.model.User;
import com.vladyslavberezovskyi.security.TokenResponse;
import com.vladyslavberezovskyi.security.UserDetails;
import com.vladyslavberezovskyi.security.enums.Role;
```

```
import com.vladyslavberezovskyi.service.InvitationService;
import com.vladyslavberezovskyi.service.UserService;
import com.vladyslavberezovskyi.util.EmailSender;
import io.jsonwebtoken.Jwts;
import io.jsonwebtoken.SignatureAlgorithm;
import lombok.RequiredArgsConstructor;
import org.springframework.beans.factory.annotation.Value;
import org.springframework.security.core.context.SecurityContextHolder;
import org.springframework.security.crypto.password.PasswordEncoder;
import org.springframework.stereotype.Service;

import java.util.Date;
import java.util.UUID;

@Service
@RequiredArgsConstructor
public class UserServiceImpl implements UserService {

    @Value("${constants.secretKey}")
    private String secretKey;
    @Value("${constants.tokenExpirationTime}")
    private Long tokenExpirationTime;

    private final UserRepository repository;
    private final UserMapper mapper;
    private final InvitationService invitationService;
    private final InvitationRepository invitationRepository;
    private final PasswordEncoder encoder;
    private final EmailSender emailSender;
```

```
    @Override
    public User getUserById() {
        UUID id = ((UserDetails)
SecurityContextHolder.getContext().getAuthentication().getPrincipal()).getId();
        UserEntity userEntity = repository.findById(id)
                .orElseThrow(ResourceNotFoundException::new);

        return mapper.entityToModel(userEntity);
    }

    @Override
    public User createUser(User user) {
        if (repository.existsByEmail(user.getEmail())) {
            throw new UserAlreadyExistsException();
        }

        UserEntity newUser = mapper.modelToEntity(user);
        newUser.setRole(Role.USER);
        newUser.setPassword(encoder.encode(newUser.getPassword()));
        UserEntity savedUser = repository.saveAndFlush(newUser);

        invitationService.sendInvitation(savedUser);

        return mapper.entityToModel(savedUser);
    }

    @Override
    public TokenResponse auth(User user) {
```

```
        UserEntity userEntity = repository.findByEmail(user.getEmail())
                .orElseThrow(ForbiddenException::new);

        if (!userEntity.isEnabled()) {
            throw new ForbiddenException();
        }

        if (!encoder.matches(user.getPassword(), userEntity.getPassword())) {
            throw new ForbiddenException();
        }

        TokenResponse tokenResponse = new TokenResponse();
        tokenResponse.setToken(generateJWT(userEntity.getId(),
userEntity.getEmail(), userEntity.getPassword(), userEntity.getRole(),
userEntity.isEnabled()));
        tokenResponse.setExpiresIn(tokenExpirationTime);
        return tokenResponse;
    }

    @Override
    public User updateUser(User user) {
        UUID id = ((UserDetails)
SecurityContextHolder.getContext().getAuthentication().getPrincipal()).getId();
        UserEntity userEntity = repository.findById(id)
                .orElseThrow(ResourceNotFoundException::new);

        if (user.getEmail() == null) {
            user.setEmail(userEntity.getEmail());
        }
```

```
        if (!user.getEmail().equals(userEntity.getEmail())) {
            InvitationEntity invitationEntity =
invitationRepository.findByEmail(userEntity.getEmail());
            invitationEntity.setEmail(user.getEmail());
            invitationEntity.setEmailConfirmed(false);
            invitationRepository.saveAndFlush(invitationEntity);

            emailSender.sendInvitation(user.getEmail(), invitationEntity.getId());
            userEntity.setEnabled(false);
        }

        mapper.update(userEntity, user);
        if (user.getPassword() != null && !user.getPassword().isBlank()) {
            userEntity.setPassword(encoder.encode(user.getPassword()));
        }

        userEntity = repository.saveAndFlush(userEntity);

        return mapper.entityToModel(userEntity);
    }

    private String generateJWT(UUID id, String email, String password, Role role,
boolean enabled) {
        Date now = new Date();
        Date validity = new Date(now.getTime() + tokenExpirationTime);

        return Jwts.builder()
                .claim("id", id.toString())
```

```
            .claim("email", email)
            .claim("password", password)
            .claim("role", role.toString())
            .claim("enabled", enabled)
            .setIssuedAt(now)
            .setExpiration(validity)
            .signWith(SignatureAlgorithm.HS256, secretKey)
            .compact();
    }
}
```

Файл backend/service/user-service/src/main/java/com/vladyslavberezovskyi/UserService.java

```
package com.vladyslavberezovskyi;

import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;

@SpringBootApplication
public class UserService {

    public static void main(String[] args) {
        SpringApplication.run(UserService.class, args);
    }

}
```

Файл backend/service/user-service/src/main/resources/sql/ddl/000_create_invitation_table.sql

```
CREATE TABLE invitation (
  id UUID NOT NULL,
   email VARCHAR(255),
   email_confirmed BOOLEAN NOT NULL,
   phone_number VARCHAR(255),
   phone_number_confirmed BOOLEAN NOT NULL,
   CONSTRAINT pk_invitation PRIMARY KEY (id)
);
```

Файл backend/service/user-service/src/main/resources/sql/ddl/001_create_user_table.sql

```
CREATE TABLE "user" (
  id UUID NOT NULL,
   email VARCHAR(255),
   nickname VARCHAR(255),
   password VARCHAR(255),
   phone_number VARCHAR(255),
   role VARCHAR(255),
   enabled BOOLEAN NOT NULL,
   invitation_id UUID,
   CONSTRAINT pk_user PRIMARY KEY (id)
);

ALTER TABLE "user" ADD CONSTRAINT FK_USER_ON_INVITATION
FOREIGN KEY (invitation_id) REFERENCES invitation (id);
```

Файл backend/service/user-service/src/main/resources/sql/changelog-master.yaml

```
databaseChangeLog:
```

```
  - include:
      file: sql/ddl/000_create_invitation_table.sql
      context: local,default,integration-test
  - include:
      file: sql/ddl/001_create_user_table.sql
      context: local,default,integration-test
```

Файл backend/service/user-service/src/main/resources/application.yaml

```
spring:
  liquibase:
    change-log: classpath:sql/changelog-master.yaml
    contexts: ${spring.profiles.active:default}

constants:
  secretKey: ${CONSTANTS_SECRET_KEY}
  tokenExpirationTime: ${CONSTANTS_TOKEN_EXPIRATION_TIME}

email:
  username: ${EMAIL_USERNAME}
  confirmation:
    url: ${EMAIL_CONFIRMATION_URL}
    path: /api/v1/invitations/confirmInvitation
```

Файл backend/service/user-service/src/main/resources/application-local.yaml

```
spring:
  application:
    name: user-service-local
  datasource:
    username: postgres
```

```
    password: postgres
    #?stringtype=unspecified need to convert string into enum type in db
    url: jdbc:postgresql://localhost:5432/user-service?stringtype=unspecified

constants:
  secretKey: "secretKeySecretKeySecretKeySecretKeySecretKeySecretKey"
  tokenExpirationTime: 3600000

email:
  username: test
  confirmation:
    url: http://localhost:8080
    path: /api/v1/invitations/confirmInvitation
```

Файл frontend/public/index.html

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="utf-8" />
    <link rel="icon" href="%PUBLIC_URL%/favicon.ico" />
    <meta name="viewport" content="width=device-width, initial-scale=1" />
    <meta name="theme-color" content="#000000" />
    <meta
      name="description"
      content="Web site created using create-react-app"
    />
    <link rel="apple-touch-icon" href="%PUBLIC_URL%/logo192.png" />
    <!--
      manifest.json provides metadata used when your web app is installed on a
```

```
    user's mobile device or desktop. See
https://developers.google.com/web/fundamentals/web-app-manifest/
  -->
  <link rel="manifest" href="%PUBLIC_URL%/manifest.json" />
  <!--
    Notice the use of %PUBLIC_URL% in the tags above.
    It will be replaced with the URL of the `public` folder during the build.
    Only files inside the `public` folder can be referenced from the HTML.

    Unlike "/favicon.ico" or "favicon.ico", "%PUBLIC_URL%/favicon.ico" will
    work correctly both with client-side routing and a non-root public URL.
    Learn how to configure a non-root public URL by running `npm run build`.
  -->
  <title>React App</title>
 </head>
 <body>
  <noscript>You need to enable JavaScript to run this app.</noscript>
  <div id="root"></div>
  <!--
    This HTML file is a template.
    If you open it directly in the browser, you will see an empty page.

    You can add webfonts, meta tags, or analytics to this file.
    The build step will place the bundled scripts into the <body> tag.

    To begin the development, run `npm start` or `yarn start`.
    To create a production bundle, use `npm run build` or `yarn build`.
  -->
 </body>
```

```
</html>
```

Файл frontend/public/manifest.json

```
{
  "short_name": "React App",
  "name": "Create React App Sample",
  "icons": [
    {
      "src": "favicon.ico",
      "sizes": "64x64 32x32 24x24 16x16",
      "type": "image/x-icon"
    },
    {
      "src": "logo192.png",
      "type": "image/png",
      "sizes": "192x192"
    },
    {
      "src": "logo512.png",
      "type": "image/png",
      "sizes": "512x512"
    }
  ],
  "start_url": ".",
  "display": "standalone",
  "theme_color": "#000000",
  "background_color": "#ffffff"
}
```

Файл frontend/src/analytics/Analytics.css

```
body {
    font-family: Arial, sans-serif;
    background-color: #ccf;
    margin: 0;
    padding: 0;
}

.analytics {
    display: flex;
    flex-direction: column;
    height: 100%;
    width: 100%;
    text-align: left;
}

.content {
    display: flex;
    justify-content: center;
    padding: 20px;
}

.analytics-section {
    width: 100%;
    border-radius: 10px;
    height: 100%;
}

.analytics-section h2 {
```

```
    margin-top: 0;
    color: #000;
}

.charts {
    display: flex;
    justify-content: space-between;
    gap: 20px;
}

.chart {
    width: 48%;
}
```

Файл frontend/src/analytics/Analytics.js

```
import React, { useEffect, useState } from 'react';
import './Analytics.css';
import Header from '../header/Header'
import BudgetChart from '../charts/BudgetChart'
import TransactionsChart from '../charts/TransactionsChart'

const Analytics = () => {
    const [budgets, setBudgets] = useState(null);
    const [transactions, setTransactions] = useState(null);
    const [balances, setBalances] = useState(null);
    const token = localStorage.getItem('token');

    useEffect(() => {
        const fetchTransactions = async () => {
```

```
    try {
        const response = await fetch('http://localhost:8081/api/v1/transactions', {
            method: 'GET',
            headers: {
                'Content-Type': 'application/json',
                'Authorization': `Bearer ${token}`
            }
        });

        if (response.ok) {
            const data = await response.json();
            setTransactions(data);
        } else {
        }
    } catch (error) {
    }
};
const fetchBudgets = async () => {
    try {
        const response = await fetch('http://localhost:8081/api/v1/budgets', {
            method: 'GET',
            headers: {
                'Content-Type': 'application/json',
                'Authorization': `Bearer ${token}`
            }
        });

        if (response.ok) {
            const data = await response.json();
```

```
            setBudgets(data);
        } else {
        }
    } catch (error) {
    }
};
const fetchBalances = async () => {
    try {
        const response = await fetch('http://localhost:8081/api/v1/balances', {
            method: 'GET',
            headers: {
                'Content-Type': 'application/json',
                'Authorization': `Bearer ${token}`
            }
        });

        if (response.ok) {
            const data = await response.json();
            setBalances(data);
        } else {
        }
    } catch (error) {
    }
};

fetchBalances();
fetchTransactions();
fetchBudgets();
}, [token]);
```

```
  return (
    <div className="analytics">
      <Header/>
      <div className="content">
        <div className="analytics-section">
          <h2>Аналітика</h2>
          <div className="charts">
            <div className="chart">
              {transactions !== null && balances !== null ?
<TransactionsChart balances={balances} transactions={transactions} /> :
'Загрузка...'}
            </div>
            <div className="chart">
              {budgets !== null ? <BudgetChart budgets={budgets} /> :
'Загрузка...'}
            </div>
          </div>
        </div>
      </div>
  );
};

export default Analytics;
```

Файл frontend/src/balances/Balances.css

```
body {
  margin: 0;
```

```
    font-family: Arial, sans-serif;
    background-color: #ccf;
}

.balances-container {
    display: flex;
    flex-direction: column;
}

.balances-content {
    width: 94%;
    padding: 3%;
    padding-top: 0;
}

.balances-filter {
    display: flex;
    justify-content: space-between;
    margin-bottom: 20px;
}

.balances-filter select {
    width: 160px;
    font-size: 20px;
}

.add-button {
    padding: 10px 20px;
    font-size: 20px;
```

```
    cursor: pointer;
    background-color: #fff;
    border: none;
    border-radius: 5px;
    box-shadow: 0 2px 4px rgba(0, 0, 0, 0.1);
}

.balances-list {
    list-style-type: none;
    padding: 0;
    padding: 10px;
    background-color: #000;
    border-radius: 15px;
}

.balances-list ul {
    max-height: 580px;
    overflow-y: scroll;
}

.balance-item {
    display: flex;
    font-size: 30px;
    justify-content: space-between;
    align-items: center;
    padding: 10px 20px;
    background-color: #000;
    color: #fff;
    border-radius: 5px;
```

```
    margin-bottom: 10px;
}

.balance-item span {
    flex: 1;
}

.edit-button, .delete-button {
    padding: 5px 10px;
    font-size: 20px;
    cursor: pointer;
    background-color: #fff;
    border: none;
    border-radius: 5px;
    box-shadow: 0 2px 4px rgba(0, 0, 0, 0.1);
    margin-left: 10px;
}
```

Файл frontend/src/balances/Balances.js

```
import React, { useEffect, useState } from 'react';
import Header from '../header/Header'
import CreateBalancesSideBar from './CreateBalancesSideBar'
import EditBalancesSideBar from './EditBalancesSideBar'
import './Balances.css';

const Balances = () => {
    const [balances, setBalances] = useState(null);
    const [categories, setCategories] = useState(null);
    const [selectedBalance, setSelectedBalance] = useState('');
    const [isSidebarOpen, setIsSidebarOpen] = useState(false);
```

```
    const [isEditSidebarOpen, setIsEditSidebarOpen] = useState(false);
    const [balanceToEdit, setBalanceToEdit] = useState(null);
    const token = localStorage.getItem('token');

    const handleAddButtonClick = () => {
        setIsSidebarOpen(true);
    };

    const handleEditClick = (balance) => {
        setBalanceToEdit(balance);
        setIsEditSidebarOpen(true);
    };

    const handleDeleteClick = async (deletedBalance) => {
        try {
            const response = await fetch('http://localhost:8081/api/v1/balances/' +
deletedBalance.id, {
                method: 'DELETE',
                headers: {
                    'Content-Type': 'application/json',
                    'Authorization': `Bearer ${token}`
                }
            });

            if (response.ok) {
                setBalances((prevBalances) =>
                    prevBalances
                        .filter((balance) => balance.id !== deletedBalance.id)
                );
```

```
        } else {
        }
    } catch (error) {
    }
};

const handleSidebarClose = () => {
    setIsSidebarOpen(false);
};

const handleAddBalance = async (balance) => {
    try {
        const response = await fetch('http://localhost:8081/api/v1/balances', {
            method: 'POST',
            headers: {
                'Content-Type': 'application/json',
                'Authorization': `Bearer ${token}`
            },
            body: JSON.stringify(balance)
        });

        if (response.ok) {
            const data = await response.json();
            setBalances((prevBalances) => [...prevBalances, data]);
        } else {
        }
    } catch (error) {
    }
};
```

```
    const handleEditBalance = async (updatedBalance) => {
        try {
            const response = await fetch('http://localhost:8081/api/v1/balances/' +
updatedBalance.id, {
                method: 'PATCH',
                headers: {
                    'Content-Type': 'application/json',
                    'Authorization': `Bearer ${token}`
                },
                body: JSON.stringify(updatedBalance)
            });

            if (response.ok) {
                const data = await response.json();
                setBalances((prevBalances) =>
                    prevBalances
                        .map((balance) => balance.id === updatedBalance.id ? data :
balance)
                );
            } else {
            }
        } catch (error) {
        }
    };

    useEffect(() => {
        const fetchData = async () => {
            try {
```

```
            const response = await fetch('http://localhost:8081/api/v1/balances', {
                method: 'GET',
                headers: {
                    'Content-Type': 'application/json',
                    'Authorization': `Bearer ${token}`
                }
            });

            if (response.ok) {
                const data = await response.json();
                setBalances(data);
            } else {
            }
        } catch (error) {
        }
    };

    fetchData();
}, [token]);

return (
    <div className="balances-container">
        <Header/>
        <div className="balances-content">
            <h2>Баланси</h2>
            <div className="balances-filter">
                <button className="add-button"
onClick={handleAddButtonClick}>Додати</button>
            </div>
```

```
            <div className="balances-list">
              <ul>
                {balances !== null ? balances.map((balance, index) => (
                  <li className="balance-item">
                    <span>• {balance.name}</span>
                    <span>{balance.currentValue} грн</span>
                    <button className="edit-button" onClick={() =>
handleEditClick(balance)}>Редагувати</button>
                    <button className="delete-button" onClick={() =>
handleDeleteClick(balance)}>Видалити</button>
                  </li>
                )) : 'Загрузка...'}
              </ul>
            </div>
            {isSidebarOpen && (
              <CreateBalancesSideBar
                onClose={handleSidebarClose}
                onAddBalance={handleAddBalance}
              />
            )}
            {isEditSidebarOpen && (
              <EditBalancesSideBar
                balance={balanceToEdit}
                onClose={() => setIsEditSidebarOpen(false)}
                onEditBalance={handleEditBalance}
              />
            )}
          </div>
        </div>
```

```
    );
};

export default Balances;
```

Файл frontend/src/balances/BalancesSideBar.css

```
.sidebar-form {
    position: fixed;
    right: 0;
    top: 0;
    width: 300px;
    height: 100%;
    background-color: #fff;
    box-shadow: -2px 0 5px rgba(0, 0, 0, 0.5);
    padding: 20px;
    z-index: 1000;
    overflow-y: auto;
    display: flex;
    flex-direction: column;
}

.close-button {
    align-self: flex-end;
    background-color: rgba(255, 0, 0, 0.5);
    color: black;
    border: none;
    padding: 10px;
    cursor: pointer;
    border-radius: 5px;
    box-shadow: 0 2px 4px rgba(0, 0, 0, 0.1);
```

```
    margin-bottom: 20px;
}

.form-group {
    margin-bottom: 20px;
}

.form-group label {
    display: block;
    margin-bottom: 5px;
    font-size: 16px;
}

.form-group input, .form-group select {
    width: 100%;
    padding: 10px;
    font-size: 16px;
    box-sizing: border-box;
    border: 1px solid #ccc;
    border-radius: 5px;
    box-shadow: inset 0 2px 4px rgba(0, 0, 0, 0.1);
}

.submit-button {
    align-self: center;
    background-color: green;
    color: white;
    border: none;
    padding: 10px 20px;
```

```
    cursor: pointer;
    border-radius: 5px;
    box-shadow: 0 2px 4px rgba(0, 0, 0, 0.1);
    font-size: 16px;
}
```

Файл frontend/src/balances/CreateBalancesSideBar.js

```
import React, { useState } from 'react';
import './BalancesSideBar.css';

const CreateBalancesSideBar = ({ onClose, onAddBalance }) => {
    const [balanceName, setBalanceName] = useState('');
    const [balanceCurrentValue, setBalanceCurrentValue] = useState('');

    const handleSubmit = (e) => {
        e.preventDefault();
        onAddBalance({
            name: balanceName,
            currentValue: parseFloat(balanceCurrentValue)
        });
        onClose();
    };

    return (
        <div className="sidebar-form">
            <button className="close-button" onClick={onClose}>Закрити</button>
            <form onSubmit={handleSubmit}>
                <div className="form-group">
                    <label>Назва балансу:</label>
```

```
            <input
              type="text"
              value={balanceName}
              onChange={(e) => setBalanceName(e.target.value)}
              required
            />
          </div>
          <div className="form-group">
            <label>Сума:</label>
            <input
              type="number"
              value={balanceCurrentValue}
              onChange={(e) => setBalanceCurrentValue(e.target.value)}
              required
            />
          </div>
          <button type="submit" className="submit-button">Додати</button>
        </form>
      </div>
    );
};

export default CreateBalancesSideBar;
```

Файл frontend/src/balances/EditBalancesSideBar.js

```
import React, { useState, useEffect } from 'react';
import './BalancesSideBar.css';

const EditBalancesSideBar = ({ balance, onClose, onEditBalance }) => {
```

```
    const [balanceName, setBalanceName] = useState(balance.name);
    const [balanceCurrentValue, setBalanceCurrentValue] =
useState(balance.currentValue);

    const handleSubmit = (e) => {
        e.preventDefault();
        onEditBalance({
            ...balance,
            name: balanceName,
            currentValue: parseFloat(balanceCurrentValue)
        });
        onClose();
    };

    return (
        <div className="sidebar-form">
            <button className="close-button" onClick={onClose}>Закрити</button>
            <form onSubmit={handleSubmit}>
                <div className="form-group">
                    <label>Назва балансу:</label>
                    <input
                        type="text"
                        value={balanceName}
                        onChange={(e) => setBalanceName(e.target.value)}
                        required
                    />
                </div>
                <div className="form-group">
                    <label>Сума:</label>
```

```
                <input
                    type="number"
                    value={balanceCurrentValue}
                    onChange={(e) => setBalanceCurrentValue(e.target.value)}
                    required
                />
            </div>
            <button type="submit" className="submit-button">Зберегти</button>
        </form>
    </div>
  );
};

export default EditBalancesSideBar;
```

Файл frontend/src/budgets/Budgets.css

```
body {
  font-family: Arial, sans-serif;
  background-color: #ccf;
  margin: 0;
  padding: 0;
}

.budget {
  display: flex;
  flex-direction: column;
  height: 100%;
  width: 100%;
  text-align: left;
```

```
}

.budget-header {
    height: 10%;
    width: 50%;
    display: flex;
    justify-content: space-between;
    align-items: center;
    padding: 10px;
}

.content {
    display: flex;
    flex-direction: column;
    padding: 20px;
    height: 100%;
}

.budget-container {
    display: flex;
    flex-direction: row;
    width: 100%;
    gap: 20px;
    height: 100%;
}

.budget-section {
    background-color: #000;
    color: #fff;
```

```
    padding: 20px;
    border-radius: 10px;
    width: 50%;
    max-height: 90%;
}

.budget-section h2 {
    margin-top: 0;
}

.add-button {
    background-color: #f5f5f5;
    border: none;
    padding: 10px 20px;
    cursor: pointer;
    border-radius: 5px;
    box-shadow: 0 2px 4px rgba(0, 0, 0, 0.1);
}

.add-button:hover {
    background-color: #e0e0e0;
}

.budget-list {
    font-size: 20px;
    overflow-y: scroll;
    height: 100%;
}
```

```
.budget-list ul {
    list-style: none;
    padding: 0;
    margin: 0;
}

.budget-list li {
    display: flex;
    justify-content: space-between;
    align-items: center;
    padding: 10px 0;
}

.budget-buttons {
    display: flex;
}

.budget-list div {
    width: 33%
}

.budget-list span {
    margin-right: 20px;
}

.edit-button, .delete-button {
    background-color: #f5f5f5;
    border: none;
    padding: 5px 10px;
```

```
    cursor: pointer;
    border-radius: 5px;
    box-shadow: 0 2px 4px rgba(0, 0, 0, 0.1);
    margin-left: 10px;
}

.edit-button:hover, .delete-button:hover {
    background-color: #e0e0e0;
}

.chart {
  background-color: #ffffff;
  border-radius: 10px;
  padding: 20px;
  box-shadow: 0 2px 5px rgba(0, 0, 0, 0.1);
  max-height: 90%;
  flex: 1;
  min-width: 0;
  display: flex;
  flex-direction: column;
  justify-content: center;
}
```

Файл frontend/src/budgets/Budgets.js

```
import React, { useEffect, useState } from 'react';
import './Budgets.css';
import Header from '../header/Header';
import CreateBudgetsSideBar from './CreateBudgetsSideBar'
import EditBudgetsSideBar from './EditBudgetsSideBar'
```

```
import BudgetChart from '../charts/BudgetChart';

const Budgets = () => {
    const [budgets, setBudgets] = useState(null);
    const [categories, setCategories] = useState(null);
    const [isSidebarOpen, setIsSidebarOpen] = useState(false);
    const [isEditSidebarOpen, setIsEditSidebarOpen] = useState(false);
    const [budgetToEdit, setBudgetToEdit] = useState(null);
    const token = localStorage.getItem('token');

    const handleAddButtonClick = () => {
        setIsSidebarOpen(true);
    };

    const handleAddBudget = async (budget) => {
        try {
            const response = await fetch('http://localhost:8081/api/v1/budgets', {
                method: 'POST',
                headers: {
                    'Content-Type': 'application/json',
                    'Authorization': `Bearer ${token}`
                },
                body: JSON.stringify(budget)
            });

            if (response.ok) {
                const data = await response.json();
                setBudgets((prevBudgets) => [...prevBudgets, data]);
            } else {
```

```
            }
        } catch (error) {
        }
    };

    const handleSidebarClose = () => {
        setIsSidebarOpen(false);
    };

    const handleEditClick = (budget) => {
        setBudgetToEdit(budget);
        setIsEditSidebarOpen(true);
    };

    const handleEditBudget = async (updatedBudget) => {
        try {
            const response = await fetch('http://localhost:8081/api/v1/budgets/' +
updatedBudget.id, {
                method: 'PATCH',
                headers: {
                    'Content-Type': 'application/json',
                    'Authorization': `Bearer ${token}`
                },
                body: JSON.stringify(updatedBudget)
            });

            if (response.ok) {
                const data = await response.json();
                setBudgets((prevBudgets) =>
```

```
                prevBudgets
                    .map((budget) => budget.id === updatedBudget.id ? data : budget)
            );
        } else {
        }
    } catch (error) {
    }
};

const handleDeleteClick = async (deletedBudget) => {
    try {
        const response = await fetch('http://localhost:8081/api/v1/budgets/' +
deletedBudget.id, {
            method: 'DELETE',
            headers: {
                'Content-Type': 'application/json',
                'Authorization': `Bearer ${token}`
            }
        });

        if (response.ok) {
            setBudgets((prevBudgets) =>
                prevBudgets
                    .filter((budget) => budget.id !== deletedBudget.id)
            );
        } else {
        }
    } catch (error) {
    }
```

```
};

useEffect(() => {
    const fetchBudgets = async () => {
        try {
            const response = await fetch('http://localhost:8081/api/v1/budgets', {
                method: 'GET',
                headers: {
                    'Content-Type': 'application/json',
                    'Authorization': `Bearer ${token}`
                }
            });

            if (response.ok) {
                const data = await response.json();
                setBudgets(data);
            } else {
            }
        } catch (error) {
        }
    };
    const fetchCategories = async () => {
        try {
            const response = await fetch('http://localhost:8081/api/v1/categories', {
                method: 'GET',
                headers: {
                    'Content-Type': 'application/json',
                    'Authorization': `Bearer ${token}`
                }
```

```
            });

            if (response.ok) {
                const data = await response.json();
                setCategories(data);
            } else {
            }
        } catch (error) {
        }
    };

    fetchCategories();
    fetchBudgets();
}, [token]);

return (
    <div className="budget">
        <Header/>
        <div className="content">
            <div className="budget-header">
                <h2>Бюджети</h2>
                <button className="add-button"
onClick={handleAddButtonClick}>Додати</button>
            </div>
            <div className="budget-container">
                <div className="budget-section">
                    <div className="budget-list">
                        <ul>
                            {budgets !== null ? budgets.map((budget, index) => (
```

```
                        <li key={index}>
                            <div>{budget.name}</div>

<div><span>{budget.currentValue}/{budget.neededValue}</span></div>
                            <div className="budget-buttons">
                                <button className="edit-button" onClick={() =>
handleEditClick(budget)}>Редагувати</button>
                                <button className="delete-button" onClick={() =>
handleDeleteClick(budget)}>Видалити</button>
                            </div>
                        </li>
                    )) : 'Загрузка...'}
                </ul>
            </div>
        </div>
        <div className="chart">
            {budgets !== null ? <BudgetChart budgets={budgets} /> :
'Загрузка...'}
        </div>
    </div>
    {isSidebarOpen && (
        <CreateBudgetsSideBar
            categories={categories}
            onClose={handleSidebarClose}
            onAddBudget={handleAddBudget}
        />
    )}
    {isEditSidebarOpen && (
        <EditBudgetsSideBar
```

```
                budget={budgetToEdit}
                categories={categories}
                onClose={() => setIsEditSidebarOpen(false)}
                onEditBudget={handleEditBudget}
              />
            )}
          </div>
        </div>
    );
};

export default Budgets;
```

Файл frontend/src/budgets/BudgetsSideBar.css

```
.sidebar-form {
    position: fixed;
    right: 0;
    top: 0;
    width: 300px;
    height: 100%;
    background-color: #fff;
    box-shadow: -2px 0 5px rgba(0, 0, 0, 0.5);
    padding: 20px;
    z-index: 1000;
    overflow-y: auto;
    display: flex;
    flex-direction: column;
}
```

```
.close-button {
    align-self: flex-end;
    background-color: rgba(255, 0, 0, 0.5);
    color: black;
    border: none;
    padding: 10px;
    cursor: pointer;
    border-radius: 5px;
    box-shadow: 0 2px 4px rgba(0, 0, 0, 0.1);
    margin-bottom: 20px;
}

.form-group {
    margin-bottom: 20px;
}

.form-group label {
    display: block;
    margin-bottom: 5px;
    font-size: 16px;
}

.form-group input, .form-group select {
    width: 100%;
    padding: 10px;
    font-size: 16px;
    box-sizing: border-box;
    border: 1px solid #ccc;
    border-radius: 5px;
```

```
    box-shadow: inset 0 2px 4px rgba(0, 0, 0, 0.1);
}

.submit-button {
    align-self: center;
    background-color: green;
    color: white;
    border: none;
    padding: 10px 20px;
    cursor: pointer;
    border-radius: 5px;
    box-shadow: 0 2px 4px rgba(0, 0, 0, 0.1);
    font-size: 16px;
}
```

Файл frontend/src/budgets/CreateBudgetsSideBar.js

```
import React, { useState } from 'react';
import './BudgetsSideBar.css';

const CreateBudgetsSideBar = ({ categories, balances, onClose, onAddBudget })
=> {
    const [budgetName, setBudgetName] = useState('');
    const [budgetCategory, setBudgetCategory] = useState('');
    const [budgetNeededValue, setBudgetNeededValue] = useState('');

    const handleSubmit = (e) => {
        e.preventDefault();
        onAddBudget({
            name: budgetName,
```

```
        category: budgetCategory,
        currentValue: parseFloat(0),
        neededValue: parseFloat(budgetNeededValue)
    });
    onClose();
};

return (
    <div className="sidebar-form">
        <button className="close-button" onClick={onClose}>Закрити</button>
        <form onSubmit={handleSubmit}>
            <div className="form-group">
                <label>Назва бюджету:</label>
                <input
                    type="text"
                    value={budgetName}
                    onChange={(e) => setBudgetName(e.target.value)}
                    required
                />
            </div>
            <div className="form-group">
                <label>Категорія:</label>
                <select
                    value={budgetCategory}
                    onChange={(e) => setBudgetCategory(e.target.value)}
                    required
                >
                    <option value="">Виберіть категорію</option>
                    {categories.map((category) => (
```

```
                    <option key={category.id} value={category.id}>
                      {category.name}
                    </option>
                  ))}
                </select>
              </div>
              <div className="form-group">
                <label>Ліміт:</label>
                <input
                  type="number"
                  value={budgetNeededValue}
                  onChange={(e) => setBudgetNeededValue(e.target.value)}
                  required
                />
              </div>
              <button type="submit" className="submit-button">Додати</button>
            </form>
          </div>
    );
};

export default CreateBudgetsSideBar;
```

Файл frontend/src/budgets/EditBudgetsSideBar.js

```
import React, { useState, useEffect } from 'react';
import './BudgetsSideBar.css';

const EditBudgetsSideBar = ({ budget, categories, onClose, onEditBudget }) => {
    const [budgetName, setBudgetName] = useState(budget.name);
```

```
    const [budgetCategory, setBudgetCategory] = useState(budget.category);
    const [budgetNeededValue, setBudgetNeededValue] =
useState(budget.neededValue);

    const handleSubmit = (e) => {
        e.preventDefault();
        onEditBudget({
            ...budget,
            name: budgetName,
            category: budgetCategory,
            currentValue: parseFloat(0),
            neededValue: parseFloat(budgetNeededValue)
        });
        onClose();
    };

    return (
        <div className="sidebar-form">
            <button className="close-button" onClick={onClose}>Закрити</button>
            <form onSubmit={handleSubmit}>
                <div className="form-group">
                    <label>Назва бюджету:</label>
                    <input
                        type="text"
                        value={budgetName}
                        onChange={(e) => setBudgetName(e.target.value)}
                        required
                    />
                </div>
```

```
            <div className="form-group">
              <label>Категорія:</label>
              <select
                value={budgetCategory}
                onChange={(e) => setBudgetCategory(e.target.value)}
                required
              >
                <option value="">Виберіть категорію</option>
                {categories.map((category) => (
                  <option key={category.id} value={category.id}>
                    {category.name}
                  </option>
                ))}
              </select>
            </div>
            <div className="form-group">
              <label>Сума:</label>
              <input
                type="number"
                value={budgetNeededValue}
                onChange={(e) => setBudgetNeededValue(e.target.value)}
                required
              />
            </div>
            <button type="submit" className="submit-button">Зберегти</button>
          </form>
        </div>
    );
};
```

```
export default EditBudgetsSideBar;
```

Файл frontend/src/charts/BudgetChart.js

```
import React from 'react';
import { Bar } from 'react-chartjs-2';
import { Chart as ChartJS, BarElement, CategoryScale, LinearScale, Title, Tooltip,
Legend } from 'chart.js';

ChartJS.register(BarElement, CategoryScale, LinearScale, Title, Tooltip, Legend);

const BudgetChart = ({ budgets }) => {
   const labels = budgets.map(budget => budget.name);
   const data = {
      labels,
      datasets: [
         {
            label: 'Використано',
            data: budgets.map(budget => Math.min(budget.currentValue,
budget.neededValue)),
            backgroundColor: 'gray'
         },
         {
            label: 'Залишок',
            data: budgets.map(budget => Math.max(0, budget.neededValue -
budget.currentValue)),
            backgroundColor: 'green'
         },
         {
```

```
            label: 'Перевикористано',
            data: budgets.map(budget => Math.max(0, budget.currentValue -
budget.neededValue)),
            backgroundColor: 'red'
        }
    ]
};

const options = {
    responsive: true,
    plugins: {
        legend: {
            position: 'top'
        },
        title: {
            display: true,
            text: 'Бюджети'
        }
    },
    scales: {
        x: {
            stacked: true
        },
        y: {
            stacked: true
        }
    }
};
```

```
  return <Bar data={data} options={options} />;
};

export default BudgetChart;
```

Файл frontend/src/charts/TransactionsChart.js

```
import React from 'react';
import { Line } from 'react-chartjs-2';
import { Chart as ChartJS, CategoryScale, LinearScale, PointElement,
LineElement, Title, Tooltip, Legend } from 'chart.js';
import { format, fromUnixTime } from 'date-fns';

ChartJS.register(
  CategoryScale,
  LinearScale,
  PointElement,
  LineElement,
  Title,
  Tooltip,
  Legend
);

const TransactionsChart = ({ balances, transactions }) => {
  const labels = transactions.map(transaction => format(new Date(transaction.date),
'd MMM'));

  const incomeData = transactions
    .filter(transaction => transaction.value > 0)
    .reduce((acc, tx) => {
```

```
    const date = format(new Date(tx.date), 'd MMM');
    acc[date] = (acc[date] || 0) + tx.value;
    return acc;
  }, {});

const expenseData = transactions
  .filter(transaction => transaction.value < 0)
  .reduce((acc, tx) => {
    const date = format(new Date(tx.date), 'd MMM');
    acc[date] = (acc[date] || 0) + tx.value;
    return acc;
  }, {});

const incomeDataset = labels.map(date => incomeData[date] || 0);
const expenseDataset = labels.map(date => expenseData[date] || 0);

const data = {
  labels,
  datasets: [
    {
      label: 'Доходи',
      data: incomeDataset,
      borderColor: 'green',
      backgroundColor: 'rgba(0, 255, 0, 0.3)',
      fill: false,
    },
    {
      label: 'Розходи',
      data: expenseDataset,
```

```
      borderColor: 'red',
      backgroundColor: 'rgba(255, 0, 0, 0.3)',
      fill: false,
    },
  ],
};

const options = {
  responsive: true,
  plugins: {
    legend: {
      position: 'top',
    },
    title: {
      display: true,
      text: 'Графік транзакцій',
    },
  },
  scales: {
    x: {
      type: 'category',
      labels: labels,
    },
  },
};

return <Line data={data} options={options} />;
};
```

```
export default TransactionsChart;
```

Файл frontend/src/dashboard/Dashboard.css

```
body {
  font-family: 'Arial', sans-serif;
  background-color: #ccf;
  color: #000;
  margin: 10;
  padding: 10;
  height: 100vh;
  display: flex;
  flex-direction: column;
}

.dashboard {
  display: flex;
  flex-direction: column;
  height: 100%;
  width: 100%;
  text-align: left;
}

.dashboard-main {
  display: flex;
  flex-direction: column;
  padding: 20px;
  gap: 20px;
  flex-grow: 1;
}
```

```
.top-row, .bottom-row {
  display: flex;
  justify-content: space-between;
  gap: 20px;
  width: 100%;
  flex-grow: 1;
}

.transactions-box {
  border-radius: 10px;
  padding: 20px;
  padding-left: 0px;
  flex: 1;
  min-width: 0;
  display: flex;
  flex-direction: column;
  justify-content: center;
}

.transactions {
  background-color: #ffffff;
  border-radius: 10px;
  padding: 20px;
  font-size: 30px;
  box-shadow: 0 2px 5px rgba(0, 0, 0, 0.1);
  flex: 1;
  min-width: 0;
  display: flex;
```

```
  flex-direction: column;
  justify-content: center;
}

.balance-box {
  border-radius: 10px;
  padding: 20px;
  padding-left: 0px;
  flex: 1;
  min-width: 0;
  display: flex;
  flex-direction: column;
  justify-content: center;
}

h2 {
  font-size: 40px;
}

.balance-box h2 {
  margin-left: 10%;
}

.balance {
  background-color: #ffffff;
  border-radius: 10px;
  padding: 20px;
  margin-left: 10%;
  padding-left: 0px;
```

```
  box-shadow: 0 2px 5px rgba(0, 0, 0, 0.1);
  flex: 1;
  min-width: 0;
  display: flex;
  flex-direction: column;
  justify-content: center;
  width: fit-content;
}

.chart {
  background-color: #ffffff;
  border-radius: 10px;
  padding: 20px;
  box-shadow: 0 2px 5px rgba(0, 0, 0, 0.1);
  max-height: 90%;
  flex: 1;
  min-width: 0;
  display: flex;
  flex-direction: column;
  justify-content: center;
}

.categories-box, .budgets-box {
  border-radius: 10px;
  padding: 20px;
  padding-left: 0px;
  flex: 1;
  min-width: 0;
  display: flex;
```

```
  flex-direction: column;
  justify-content: center;
}

.categories, .budgets {
  background-color: #ffffff;
  border-radius: 10px;
  padding: 20px;
  font-size: 30px;
  box-shadow: 0 2px 5px rgba(0, 0, 0, 0.1);
  flex: 1;
  min-width: 0;
  display: flex;
  flex-direction: column;
  justify-content: center;
}

.category-and-budget-row {
  display: flex;
  justify-content: space-between;
  gap: 80px;
  width: 2%;
  flex-grow: 1;
}

.transactions, .categories, .budgets {
  background-color: #000000;
  color: #ffffff;
}
```

```
.balance-amount {
  font-size: 7em;
  margin-top: 10px;
}

.transactions ul, .categories ul, .budgets ul {
  list-style-type: none;
  padding: 0;
  max-height: 200px;
  overflow-y: scroll;
  text-align: left;
  margin: 0;
}

.transactions li, .categories li, .budgets li {
  margin: 10px 0;
  display: flex;
  justify-content: space-between;
}

.chart {
  background-color: #f5f5f5;
}

.chart h2 {
  margin-bottom: 20px;
}
```

Файл frontend/src/dashboard/Dashboard.js

```
import React, { useEffect, useState } from 'react';
import './Dashboard.css';
import Header from '../header/Header';
import { useNavigate } from 'react-router-dom';
import BudgetChart from '../charts/BudgetChart';

const Dashboard = () => {
    const navigate = useNavigate();
    const [balance, setBalance] = useState(null);
    const [transactions, setTransactions] = useState(null);
    const [categories, setCategories] = useState(null);
    const [budgets, setBudgets] = useState(null);
    const token = localStorage.getItem('token');

    useEffect(() => {
        const fetchBalance = async () => {
            try {
                const response = await fetch('http://localhost:8081/api/v1/balances', {
                    method: 'GET',
                    headers: {
                        'Content-Type': 'application/json',
                        'Authorization': `Bearer ${token}`
                    }
                });

                if (response.ok) {
                    const data = await response.json();
```

```
                const totalBalance = data.reduce((total, balance) => total +
balance.currentValue, 0);
                setBalance(totalBalance);
            } else {
            }
        } catch (error) {
        }
    };
    const fetchTransactions = async () => {
        try {
            const response = await fetch('http://localhost:8081/api/v1/transactions', {
                method: 'GET',
                headers: {
                    'Content-Type': 'application/json',
                    'Authorization': `Bearer ${token}`
                }
            });

            if (response.ok) {
                const data = await response.json();
                setTransactions(data);
            } else {
            }
        } catch (error) {
        }
    };
    const fetchCategories = async () => {
        try {
            const response = await fetch('http://localhost:8081/api/v1/categories', {
```

```
            method: 'GET',
            headers: {
                'Content-Type': 'application/json',
                'Authorization': `Bearer ${token}`
            }
        });

        if (response.ok) {
            const data = await response.json();
            setCategories(data);
        } else {
        }
    } catch (error) {
    }
};
const fetchBudgets = async () => {
    try {
        const response = await fetch('http://localhost:8081/api/v1/budgets', {
            method: 'GET',
            headers: {
                'Content-Type': 'application/json',
                'Authorization': `Bearer ${token}`
            }
        });

        if (response.ok) {
            const data = await response.json();
            setBudgets(data);
        } else {
```

```
            }
        } catch (error) {
        }
    };

    fetchTransactions();
    fetchCategories();
    fetchBudgets();
    fetchBalance();
  }, [token]);

  return (
    <div className="dashboard">
      <Header/>
      <main className="dashboard-main">
        <div className="top-row">
          <div className="balance-box">
            <h2>Баланс</h2>
            <div className="balance">
              <div className="balance-amount">{balance !== null ? `${balance}
грн` : 'Загрузка...'}</div>
            </div>
          </div>
          <div className="transactions-box">
            <h2>Транзакції</h2>
            <div className="transactions">
              <ul>
                {transactions !== null ? transactions.map((transaction, index) => (
                  <li key={index}>
```

```
                {transaction.name} <span>{transaction.value} грн</span>
              </li>
            )) : 'Загрузка...'}
          </ul>
        </div>
      </div>
    </div>
    <div className="bottom-row">
      <div className="category-and-budget-row">
        <div className="categories-box">
          <h2>Категорії</h2>
          <div className="categories">
            <ul>
              {categories !== null ? categories.map((category, index) => (
                <li key={index}>
                  {category.name}
                </li>
              )) : 'Загрузка...'}
            </ul>
          </div>
        </div>
        <div className="budgets-box">
          <h2>Бюджети</h2>
          <div className="budgets">
            <ul>
              {budgets !== null ? budgets.map((budget, index) => (
                <li key={index}>
                  {budget.name}
                </li>
```

```
                    )) : 'Загрузка...'}
                  </ul>
                </div>
              </div>
            </div>
            <div className="chart">
              {budgets !== null ? <BudgetChart budgets={budgets} />  : 'Загрузка...'}
            </div>
          </div>
        </main>
      </div>
    );
};

export default Dashboard;
```

Файл frontend/src/header/Header.css

```
.App-header {
  background-color: #ccf;
  padding: 10px 20px;
  display: flex;
  justify-content: space-between;
  align-items: center;
}

.header-content {
  text-align: left;
}
```

```
.App-header h1 {
  margin: 0;
  font-size: 1.5em;
}

.App-header p {
  margin: 0;
  font-size: 0.9em;
}

.header-buttons {
  display: flex;
  gap: 20px;
}

.transactions-button, .balances-button, .analytic-button, .budgets-button, .settings-
button, .logout-button {
  padding: 5px 15px;
  border-radius: 10px;
  border: none;
  background-color: #ffffff;
  cursor: pointer;
  font-size: 0.9em;
}
```

Файл frontend/src/header/Header.js

```
import React from 'react';
import './Header.css';
import { useNavigate } from 'react-router-dom';
```

```
const Header = () => {
    const navigate = useNavigate();

    const handleHome = () => {
        navigate('/home');
    }
    const handleTransactions = () => {
        navigate('/transactions');
    }
    const handleBalances = () => {
        navigate('/balances');
    }
    const handleAnalytic = () => {
        navigate('/analytics');
    }
    const handleBudgets = () => {
        navigate('/budgets');
    }
    const handleSettings = () => {
        navigate('/settings');
    }
    const handleLogout = () => {
        localStorage.removeItem('token');
        navigate('/');
    };

    return (
        <header className="App-header">
```

```
        <div className="header-content">
          <h1 onClick={handleHome}>Finio</h1>
          <p>Все для твого фінансового комфорту</p>
        </div>
        <div className="header-buttons">
          <button className="transactions-button"
onClick={handleTransactions}>Транзакції</button>
          <button className="balances-button"
onClick={handleBalances}>Баланси</button>
          <button className="analytic-button"
onClick={handleAnalytic}>Аналітика</button>
          <button className="budgets-button"
onClick={handleBudgets}>Бюджети</button>
          <button className="settings-button"
onClick={handleSettings}>Налаштування</button>
          <button className="logout-button"
onClick={handleLogout}>Вихід</button>
        </div>
      </header>
    );
}

export default Header;
```

Файл frontend/src/login/Login.css

```
body {
    margin: 0;
    font-family: Arial, sans-serif;
    background-color: #ccf;
```

```
    display: flex;
    justify-content: center;
    align-items: center;
    height: 100vh;
}

.login-container {
    display: flex;
    justify-content: center;
    align-items: center;
    height: 100%;
    width: 100%;
}

.login-box {
    background-color: #ccf;
    padding: 40px;
    border-radius: 10px;
    text-align: center;
}

.login-box h1 {
    margin: 0;
    font-size: 48px;
}

.login-box p {
    margin: 10px 0 20px;
    font-size: 18px;
```

```
}

.input-group {
    margin-bottom: 20px;
    display: flex;
    justify-content: center;
}

.input-group input {
    width: 100%;
    max-width: 300px;
    padding: 10px;
    border: none;
    border-radius: 5px;
    font-size: 16px;
    box-shadow: 0 2px 4px rgba(0, 0, 0, 0.1);
}

.login-button {
    width: 100%;
    max-width: 300px;
    padding: 10px;
    background-color: #000;
    color: #fff;
    border: none;
    border-radius: 5px;
    cursor: pointer;
    font-size: 16px;
    box-shadow: 0 2px 4px rgba(0, 0, 0, 0.1);
```

```
}

.login-links {
    margin-top: 20px;
    display: flex;
    justify-content: space-between;
    font-size: 14px;
}

.login-links a {
    color: #000;
    text-decoration: none;
}

.login-links a:hover {
    text-decoration: underline;
}
```

Файл frontend/src/login/Login.js

```
import React, { useState } from 'react';
import { useNavigate } from 'react-router-dom';
import './Login.css';

const Login = () => {
    const navigate = useNavigate();
    const [email, setEmail] = useState('');
    const [password, setPassword] = useState('');

    const handleSubmit = async (e) => {
```

```
    e.preventDefault();

    try {
        const response = await fetch('http://localhost:8080/api/v1/users/auth', {
            method: 'POST',
            headers: {
                'Content-Type': 'application/json',
                'Access-Control-Allow-Origin': '*'
            },
            body: JSON.stringify({ email, password })
        });

        if (response.ok) {
            const data = await response.json();
            localStorage.setItem('token', data.token);
            navigate('/home');
            console.log('Успешный вход!');
        } else {
        }
    } catch (error) {
    }
};

return (
    <div className="container">
        <div className="box">
            <h1>Finio</h1>
            <p>Все для твого фінансового комфорту</p>
            <form onSubmit={handleSubmit}>
```

```
            <div className="input-group">
              <input
                type="email"
                placeholder="Email"
                value={email}
                onChange={(e) => setEmail(e.target.value)}
                required
              />
            </div>
            <div className="input-group">
              <input
                type="password"
                placeholder="Пароль"
                value={password}
                onChange={(e) => setPassword(e.target.value)}
                required
              />
            </div>
            <button type="submit" className="button">Увійти</button>
          </form>
          <div className="links">
            <a href="/register">Реєстрація</a>
          </div>
        </div>
      </div>
    );
};

export default Login;
```

Файл frontend/src/register/Register.css

```
body {
    margin: 0;
    font-family: Arial, sans-serif;
    background-color: #ccf;
    display: flex;
    justify-content: center;
    align-items: center;
    height: 100vh;
}

.container {
    display: flex;
    justify-content: center;
    align-items: center;
    height: 100%;
    width: 100%;
}

.box {
    background-color: #ccf;
    padding: 40px;
    border-radius: 10px;
    box-shadow: 0 4px 8px rgba(0, 0, 0, 0.1);
    text-align: center;
}

.box h1 {
```

```
    margin: 0;
    font-size: 48px;
}

.box p {
    margin: 10px 0 20px;
    font-size: 18px;
}

.input-group {
    margin-bottom: 20px;
    display: flex;
    justify-content: center;
}

.input-group input {
    width: 100%;
    max-width: 300px;
    padding: 10px;
    border: none;
    border-radius: 5px;
    font-size: 16px;
    box-shadow: 0 2px 4px rgba(0, 0, 0, 0.1);
}

.input-group .input-error {
    width: 100%;
    max-width: 300px;
    padding: 10px;
```

```
    border: none;
    border-radius: 5px;
    background-color: rgba(255, 0, 0, 0.2);
    font-size: 16px;
    box-shadow: 0 2px 4px rgba(0, 0, 0, 0.1);
}


.button {
    width: 100%;
    max-width: 300px;
    padding: 10px;
    background-color: #000;
    color: #fff;
    border: none;
    border-radius: 5px;
    cursor: pointer;
    font-size: 16px;
    box-shadow: 0 2px 4px rgba(0, 0, 0, 0.1);
}

.links {
    margin-top: 20px;
    display: flex;
    justify-content: space-between;
    font-size: 14px;
}

.links a {
```

```
    color: #000;
    text-decoration: none;
}

.links a:hover {
    text-decoration: underline;
}
```

Файл frontend/src/register/Register.js

```
import React, { useState } from 'react';
import { useNavigate } from 'react-router-dom';
import './Register.css';

const Register = () => {
    const navigate = useNavigate();
    const [email, setEmail] = useState('');
    const [password, setPassword] = useState('');
    const [repeatPassword, setRepeatPassword] = useState('');
    const [passwordMatch, setPasswordMatch] = useState(true);

    const handleRepeatPasswordChange = async (e) => {
        setRepeatPassword(e)

        if (e !== password) {
            console.log(repeatPassword !== password, e, password)
            setPasswordMatch(false);
            return;
        } else {
            setPasswordMatch(true);
```

```
    }
}

const handleSubmit = async (e) => {
    e.preventDefault();

    if (repeatPassword !== password) {
        return;
    }

    try {
        const response = await fetch('http://localhost:8080/api/v1/users', {
            method: 'POST',
            headers: {
                'Content-Type': 'application/json',
                'Access-Control-Allow-Origin': '*'
            },
            body: JSON.stringify({ email, password })
        });

        if (response.ok) {
            navigate('/');
        } else {
        }
    } catch (error) {
    }
};

return (
```

```
<div className="container">
  <div className="box">
    <h1>Finio</h1>
    <p>Створіть ваш обліковий запис</p>
    <form onSubmit={handleSubmit}>
      <div className="input-group">
        <input
          id="repeat-password"
          type="email"
          placeholder="Email"
          value={email}
          onChange={(e) => setEmail(e.target.value)}
          required
        />
      </div>
      <div className="input-group">
        <input
          type="password"
          placeholder="Пароль"
          value={password}
          onChange={(e) => setPassword(e.target.value)}
          required
        />
      </div>
      <div className="input-group">
        <input
          type="password"
          placeholder="Підтвердження пароля"
          value={repeatPassword}
```

```
                    onChange={(e) =>
handleRepeatPasswordChange(e.target.value)}
                    className={!passwordMatch ? 'input-error' : ''}
                    required
                  />
                </div>
                <button type="submit" className="button">Реєстрація</button>
              </form>
              <div className="links">
                <a href="/">Увійти</a>
              </div>
            </div>
          </div>
    );
};

export default Register;
```

Файл frontend/src/settings/Settings.css

```
.settings-container {
  display: flex;
  flex-direction: column;
  background-color: #ccf;
  height: 100vh;
  font-family: Arial, sans-serif;
}

.settings-content {
  display: flex;
```

```
  height: 100%;
  padding: 20px;
  gap: 20px;
}

.profile-block {
    width: 50%;
}

.category-block {
    width: 50%;
}

.category-settings {
  background-color: black;
  padding: 20px;
  border-radius: 10px;
  color: white;
}

.profile-settings {
  background-color: black;
  padding: 20px;
  border-radius: 10px;
  color: white;
  display: flex;
  flex-direction: column;
}
```

```
.profile-settings h2,
.category-settings h2 {
  font-size: 1.5em;
}

.category-settings ul {
  display: flex;
  flex-direction: column;
}

.profile-settings button {
  font-size: 20px;
  padding: 5px;
  margin: 10px 0;
}

.profile-settings input {
  margin: 10px 0;
  padding: 10px;
  border: none;
  border-radius: 5px;
}

.category-header input {
  padding: 10px;
  border: none;
  border-radius: 5px;
}
```

```
.category-header {
  display: flex;
  align-items: center;
  justify-content: space-between;
  flex-direction: row;
}

.category-header div {
  display: flex;
  gap: 20px;
}

.category-settings ul {
  list-style-type: disc;
  padding-left: 20px;
  font-size: 20px;
  display: flex;
  width: 100%;
  flex-direction: column;
  max-height: 500px;
  overflow-y: scroll;
}

.category-settings li {
  margin: 5px 0;
  display: flex;
  flex-direction: row;
  justify-content: space-between;
}
```

```
.category-header button {
  padding: 10px;
  border: none;
  border-radius: 5px;
  background-color: white;
  cursor: pointer;
}

.edit-button, .delete-button, .save-button {
    background-color: #f5f5f5;
    border: none;
    font-size: 20px;
    padding: 5px 10px;
    cursor: pointer;
    border-radius: 5px;
    box-shadow: 0 2px 4px rgba(0, 0, 0, 0.1);
    margin-left: 10px;
}

.category-settings input {
  padding: 10px;
  border: none;
  border-radius: 5px;
}
```

Файл frontend/src/settings/Settings.js

```
import React, { useEffect, useState } from 'react';
import './Settings.css';
```

```
import Header from '../header/Header';

const Settings = () => {
  const [name, setName] = useState('');
  const [password, setPassword] = useState('');
  const [phone, setPhone] = useState('');
  const [email, setEmail] = useState('');
  const [categories, setCategories] = useState(null);
  const [newCategory, setNewCategory] = useState('');
  const [editCategoryId, setEditCategoryId] = useState(null);
  const [editCategoryName, setEditCategoryName] = useState('');
  const token = localStorage.getItem('token');

  const handleAddCategory = async () => {
    try {
      const response = await fetch('http://localhost:8081/api/v1/categories', {
        method: 'POST',
        headers: {
          'Content-Type': 'application/json',
          'Authorization': `Bearer ${token}`
        },
        body: JSON.stringify({ name: newCategory })
      });

      if (response.ok) {
        const data = await response.json();
        setCategories([...categories, data]);
        setNewCategory('');
      } else {
```

```
    }
   } catch (error) {
   }
  };

  const handleEditClick = (category) => {
   setEditCategoryId(category.id);
   setEditCategoryName(category.name);
  };

  const handleSaveUser = async () => {
   try {
    const response = await fetch('http://localhost:8080/api/v1/user', {
     method: 'PATCH',
     headers: {
      'Content-Type': 'application/json',
      'Authorization': `Bearer ${token}`
     },
     body: JSON.stringify({ name: name, password: password, email: email,
phoneNumber: phone })
    });

    if (response.ok) {
     const updatedCategory = await response.json();
     setCategories(categories.map(cat => (cat.id === updatedCategory.id ?
updatedCategory : cat)));
     setEditCategoryId(null);
     setEditCategoryName('');
    } else {
```

```
    }
   } catch (error) {
   }
  };

  const handleSaveClick = async (category) => {
   try {
    const response = await fetch('http://localhost:8081/api/v1/categories/' +
category.id, {
     method: 'PATCH',
     headers: {
      'Content-Type': 'application/json',
      'Authorization': `Bearer ${token}`
     },
     body: JSON.stringify({ name: editCategoryName })
    });

    if (response.ok) {
     const updatedCategory = await response.json();
     setCategories(categories.map(cat => (cat.id === updatedCategory.id ?
updatedCategory : cat)));
     setEditCategoryId(null);
     setEditCategoryName('');
    } else {
    }
   } catch (error) {
   }
  };
```

```
  const handleDeleteClick = async (deletedCategory) => {
    try {
      const response = await fetch('http://localhost:8081/api/v1/categories/' +
deletedCategory.id, {
        method: 'DELETE',
        headers: {
          'Content-Type': 'application/json',
          'Authorization': `Bearer ${token}`
        }
      });

      if (response.ok) {
        setCategories(categories.filter(category => category.id !==
deletedCategory.id));
      } else {
      }
    } catch (error) {
    }
  };

  useEffect(() => {
    const fetchUser = async () => {
      try {
        const response = await fetch('http://localhost:8080/api/v1/user', {
          method: 'GET',
          headers: {
            'Content-Type': 'application/json',
            'Authorization': `Bearer ${token}`
          }
```

```
      });

      if (response.ok) {
        const data = await response.json();
        setName(data.name);
        setPassword('')
        setPhone(data.phoneNumber);
        setEmail(data.email);
      } else {
      }
    } catch (error) {
    }
}
const fetchCategories = async () => {
  try {
    const response = await fetch('http://localhost:8081/api/v1/categories', {
      method: 'GET',
      headers: {
        'Content-Type': 'application/json',
        'Authorization': `Bearer ${token}`
      }
    });

    if (response.ok) {
      const data = await response.json();
      setCategories(data);
    } else {
    }
  } catch (error) {
```

```
    }
  };

  fetchUser();
  fetchCategories();
 }, [token]);

 return (
  <div className="settings-container">
   <Header />
   <div className="settings-content">
    <div className="profile-block">
     <h2>Налаштування</h2>
     <div className="profile-settings">
       <input type="text" placeholder="Ім'я" value={name} onChange={(e) =>
setName(e.target.value)} />
       <input type="email" placeholder="Email" value={email} onChange={(e)
=> setEmail(e.target.value)} />
       <input type="password" placeholder="Новий пароль" value={password}
onChange={(e) => setPassword(e.target.value)} />
       <input type="text" placeholder="Телефон" value={phone}
onChange={(e) => setPhone(e.target.value)} />
       <button onClick={handleSaveUser}>Зберегти</button>
     </div>
    </div>
    <div className="category-block">
     <div className="category-header">
      <h2>Категорії</h2>
      <div>
```

```
            <input
              type="text"
              placeholder="Нова категорія"
              value={newCategory}
              onChange={(e) => setNewCategory(e.target.value)}
            />
            <button onClick={handleAddCategory}>Додати</button>
          </div>
        </div>
        <div className="category-settings">
          <ul>
            {categories !== null ? categories.map((category) => (
              <li key={category.id}>
                {editCategoryId === category.id ? (
                  <input
                    type="text"
                    value={editCategoryName}
                    onChange={(e) => setEditCategoryName(e.target.value)}
                  />
                ) : (
                  category.name
                )}
                <div className="category-buttons">
                  {editCategoryId === category.id ? (
                    <button className="save-button" onClick={() =>
handleSaveClick(category)}>Зберегти</button>
                  ) : (
                    <button className="edit-button" onClick={() =>
handleEditClick(category)}>Редагувати</button>
```

```
                )}
                <button className="delete-button" onClick={() =>
handleDeleteClick(category)}>Видалити</button>
              </div>
            </li>
          )) : 'Загрузка...'}
        </ul>
      </div>
    </div>
  </div>
 );
};

export default Settings;
```

Файл frontend/src/transactions/CreateTransactionsSideBar.js

```
import React, { useState } from 'react';
import './TransactionsSideBar.css';

const CreateTransactionsSideBar = ({ categories, balances, onClose,
onAddTransaction }) => {
  const [transactionName, setTransactionName] = useState('');
  const [transactionBalance, setTransactionBalance] = useState('');
  const [transactionCategory, setTransactionCategory] = useState('');
  const [transactionValue, setTransactionValue] = useState('');

  const handleSubmit = (e) => {
    e.preventDefault();
```

```
    onAddTransaction({
      name: transactionName,
      balance: transactionBalance,
      category: transactionCategory,
      value: parseFloat(transactionValue)
    });
    onClose();
};

return (
    <div className="sidebar-form">
      <button className="close-button" onClick={onClose}>Закрити</button>
      <form onSubmit={handleSubmit}>
        <div className="form-group">
          <label>Назва Транзакції:</label>
          <input
            type="text"
            value={transactionName}
            onChange={(e) => setTransactionName(e.target.value)}
            required
          />
        </div>
        <div className="form-group">
          <label>Баланс:</label>
          <select
            value={transactionBalance}
            onChange={(e) => setTransactionBalance(e.target.value)}
            required
          >
```

```
        <option value="">Виберіть баланс</option>
        {balances.map((balance) => (
          <option key={balance.id} value={balance.id}>
            {balance.name}
          </option>
        ))}
      </select>
    </div>
    <div className="form-group">
      <label>Категорія:</label>
      <select
        value={transactionCategory}
        onChange={(e) => setTransactionCategory(e.target.value)}
        required
      >
        <option value="">Виберіть категорію</option>
        {categories.map((category) => (
          <option key={category.id} value={category.id}>
            {category.name}
          </option>
        ))}
      </select>
    </div>
    <div className="form-group">
      <label>Сума:</label>
      <input
        type="number"
        value={transactionValue}
        onChange={(e) => setTransactionValue(e.target.value)}
```

```
                    required
                  />
                </div>
                <button type="submit" className="submit-button">Додати</button>
            </form>
        </div>
    );
};

export default CreateTransactionsSideBar;
```

Файл frontend/src/transactions/EditTransactionsSideBar.js

```
import React, { useState, useEffect } from 'react';
import './TransactionsSideBar.css';

const EditTransactionsSideBar = ({ transaction, balances, categories, onClose,
onEditTransaction }) => {
    const [transactionName, setTransactionName] = useState(transaction.name);
    const [transactionCategory, setTransactionCategory] =
useState(transaction.category);
    const [transactionBalance, setTransactionBalance] =
useState(transaction.balance);
    const [transactionValue, setTransactionValue] = useState(transaction.value);

    const handleSubmit = (e) => {
        e.preventDefault();
        onEditTransaction({
            ...transaction,
            name: transactionName,
```

```
        category: transactionCategory,
        balance: transactionBalance,
        value: parseFloat(transactionValue)
    });
    onClose();
};

return (
    <div className="sidebar-form">
        <button className="close-button" onClick={onClose}>Закрити</button>
        <form onSubmit={handleSubmit}>
            <div className="form-group">
                <label>Назва Транзакції:</label>
                <input
                    type="text"
                    value={transactionName}
                    onChange={(e) => setTransactionName(e.target.value)}
                    required
                />
            </div>
            <div className="form-group">
                <label>Баланс:</label>
                <select
                    value={transactionBalance}
                    onChange={(e) => setTransactionBalance(e.target.value)}
                    required
                >
                    <option value="">Виберіть баланс</option>
                    {balances.map((balance) => (
```

```
            <option key={balance.id} value={balance.id}>
              {balance.name}
            </option>
          ))}
        </select>
      </div>
      <div className="form-group">
        <label>Категорія:</label>
        <select
          value={transactionCategory}
          onChange={(e) => setTransactionCategory(e.target.value)}
          required
        >
          <option value="">Виберіть категорію</option>
          {categories.map((category) => (
            <option key={category.id} value={category.id}>
              {category.name}
            </option>
          ))}
        </select>
      </div>
      <div className="form-group">
        <label>Сума:</label>
        <input
          type="number"
          value={transactionValue}
          onChange={(e) => setTransactionValue(e.target.value)}
          required
        />
```

```
            </div>
            <button type="submit" className="submit-
button">Сохранить</button>
          </form>
        </div>
    );
};

export default EditTransactionsSideBar;
```

Файл frontend/src/transactions/Transactions.css

```
body {
    margin: 0;
    font-family: Arial, sans-serif;
    background-color: #ccf;
}

.transactions-container {
    display: flex;
    flex-direction: column;
}

.transactions-content {
    width: 94%;
    padding: 3%;
    padding-top: 0;
}

.transactions-filter {
```

```
    display: flex;
    justify-content: space-between;
    margin-bottom: 20px;
}

.transactions-filter select {
    width: 160px;
    font-size: 20px;
}

.add-button {
    padding: 10px 20px;
    font-size: 20px;
    cursor: pointer;
    background-color: #fff;
    border: none;
    border-radius: 5px;
    box-shadow: 0 2px 4px rgba(0, 0, 0, 0.1);
}

.transactions-list {
    list-style-type: none;
    padding: 0;
    padding: 10px;
    background-color: #000;
    border-radius: 15px;
}

.transactions-list ul {
```

```
    max-height: 580px;
    overflow-y: scroll;
}

.transaction-item {
    display: flex;
    font-size: 30px;
    justify-content: space-between;
    align-items: center;
    padding: 10px 20px;
    background-color: #000;
    color: #fff;
    border-radius: 5px;
    margin-bottom: 10px;
}

.transaction-item span {
    flex: 1;
}

.edit-button, .delete-button {
    padding: 5px 10px;
    font-size: 20px;
    cursor: pointer;
    background-color: #fff;
    border: none;
    border-radius: 5px;
    box-shadow: 0 2px 4px rgba(0, 0, 0, 0.1);
    margin-left: 10px;
```

```
}
```

Файл frontend/src/transactions/Transactions.js

```
import React, { useEffect, useState } from 'react';
import Header from '../header/Header'
import CreateTransactionsSideBar from './CreateTransactionsSideBar'
import EditTransactionsSideBar from './EditTransactionsSideBar'
import './Transactions.css';

const Transactions = () => {
    const [balances, setBalances] = useState(null);
    const [transactions, setTransactions] = useState(null);
    const [categories, setCategories] = useState(null);
    const [selectedBalance, setSelectedBalance] = useState('');
    const [isSidebarOpen, setIsSidebarOpen] = useState(false);
    const [isEditSidebarOpen, setIsEditSidebarOpen] = useState(false);
    const [transactionToEdit, setTransactionToEdit] = useState(null);
    const token = localStorage.getItem('token');

    const handleAddButtonClick = () => {
        setIsSidebarOpen(true);
    };

    const handleEditClick = (transaction) => {
        setTransactionToEdit(transaction);
        setIsEditSidebarOpen(true);
    };

    const handleDeleteClick = async (deletedTransaction) => {
```

```
        try {
            const response = await fetch('http://localhost:8081/api/v1/transactions/' +
deletedTransaction.id, {
                method: 'DELETE',
                headers: {
                    'Content-Type': 'application/json',
                    'Authorization': `Bearer ${token}`
                }
            });

            if (response.ok) {
                setTransactions((prevTransactions) =>
                    prevTransactions
                        .filter((transaction) => transaction.id !== deletedTransaction.id)
                );
            } else {
            }
        } catch (error) {
        }
    };

    const handleSidebarClose = () => {
        setIsSidebarOpen(false);
    };

    const handleAddTransaction = async (transaction) => {
        try {
            const response = await fetch('http://localhost:8081/api/v1/transactions', {
                method: 'POST',
```

```
            headers: {
                'Content-Type': 'application/json',
                'Authorization': `Bearer ${token}`
            },
            body: JSON.stringify(transaction)
        });

        if (response.ok) {
            const data = await response.json();
            if (selectedBalance === data.balance) {
                setTransactions((prevTransactions) => [...prevTransactions, data]);
            }
        } else {
        }
    } catch (error) {
    }
};

const handleEditTransaction = async (updatedTransaction) => {
    try {
        const response = await fetch('http://localhost:8081/api/v1/transactions/' +
updatedTransaction.id, {
            method: 'PATCH',
            headers: {
                'Content-Type': 'application/json',
                'Authorization': `Bearer ${token}`
            },
            body: JSON.stringify(updatedTransaction)
        });
```

```
            if (response.ok) {
                const data = await response.json();
                setTransactions((prevTransactions) =>
                    prevTransactions
                        .map((transaction) => transaction.id === updatedTransaction.id ?
updatedTransaction : transaction)
                        .filter((transaction) => transaction.balance === selectedBalance)
                );
            } else {
            }
        } catch (error) {
        }
    };

    const handleSelectChange = async (e) => {
        try {
            const balanceId = e;
            setSelectedBalance(balanceId);
            const response = await
fetch('http://localhost:8081/api/v1/transactions?balanceId=' + balanceId, {
                method: 'GET',
                headers: {
                    'Content-Type': 'application/json',
                    'Authorization': `Bearer ${token}`
                }
            });

            if (response.ok) {
```

```
            const data = await response.json();
            setTransactions(data);
        } else {
        }
    } catch (error) {
    }
}
useEffect(() => {
    const fetchData = async () => {
        try {
            const response = await fetch('http://localhost:8081/api/v1/balances', {
                method: 'GET',
                headers: {
                    'Content-Type': 'application/json',
                    'Authorization': `Bearer ${token}`
                }
            });

            if (response.ok) {
                const data = await response.json();
                setBalances(data);

                try {
                    const balanceId = data[0].id;
                    setSelectedBalance(data[0].id);
                    const response = await
fetch('http://localhost:8081/api/v1/transactions?balanceId=' + balanceId, {
                        method: 'GET',
                        headers: {
```

```
                    'Content-Type': 'application/json',
                    'Authorization': `Bearer ${token}`
                }
            });

            if (response.ok) {
                const data = await response.json();
                setTransactions(data);
            } else {
            }
        } catch (error) {
        }
    } else {
    }
} catch (error) {
}
};

const fetchCategories = async () => {
    try {
        const response = await fetch('http://localhost:8081/api/v1/categories', {
            method: 'GET',
            headers: {
                'Content-Type': 'application/json',
                'Authorization': `Bearer ${token}`
            }
        });

        if (response.ok) {
```

```
                const data = await response.json();
                setCategories(data);
            } else {
            }
        } catch (error) {
        }
    };

    fetchCategories();
    fetchData();
}, [token]);

return (
    <div className="transactions-container">
        <Header/>
        <div className="transactions-content">
            <h2>Транзакції</h2>
            <div className="transactions-filter">
                <select onChange={(e) => handleSelectChange(e.target.value)}>
                    {balances !== null ? balances.map((balance, index) => (
                        <option value={balance.id}>{balance.name}</option>
                    )) : 'Загрузка...'}
                </select>
                <button className="add-button"
onClick={handleAddButtonClick}>Додати</button>
            </div>
            <div className="transactions-list">
                <ul>
                    {transactions !== null ? transactions.map((transaction, index) => (
```

```
                    <li className="transaction-item">
                        <span>• {transaction.name}</span>
                        <span>{categories.find(cat => cat.id ===
transaction.category).name}</span>
                        <span>{transaction.value} грн</span>
                        <button className="edit-button" onClick={() =>
handleEditClick(transaction)}>Редагувати</button>
                        <button className="delete-button" onClick={() =>
handleDeleteClick(transaction)}>Видалити</button>
                    </li>
                )) : 'Загрузка...'}
            </ul>
        </div>
        {isSidebarOpen && (
            <CreateTransactionsSideBar
                categories={categories}
                balances={balances}
                onClose={handleSidebarClose}
                onAddTransaction={handleAddTransaction}
            />
        )}
        {isEditSidebarOpen && (
            <EditTransactionsSideBar
                transaction={transactionToEdit}
                categories={categories}
                balances={balances}
                onClose={() => setIsEditSidebarOpen(false)}
                onEditTransaction={handleEditTransaction}
            />
```

```
            )}
          </div>
        </div>
    );
};

export default Transactions;
```

Файл frontend/src/transactions/TransactionsSideBar.css

```
.sidebar-form {
    position: fixed;
    right: 0;
    top: 0;
    width: 300px;
    height: 100%;
    background-color: #fff;
    box-shadow: -2px 0 5px rgba(0, 0, 0, 0.5);
    padding: 20px;
    z-index: 1000;
    overflow-y: auto;
    display: flex;
    flex-direction: column;
}

.close-button {
    align-self: flex-end;
    background-color: rgba(255, 0, 0, 0.5);
    color: black;
    border: none;
```

```
    padding: 10px;
    cursor: pointer;
    border-radius: 5px;
    box-shadow: 0 2px 4px rgba(0, 0, 0, 0.1);
    margin-bottom: 20px;
}

.form-group {
    margin-bottom: 20px;
}

.form-group label {
    display: block;
    margin-bottom: 5px;
    font-size: 16px;
}

.form-group input, .form-group select {
    width: 100%;
    padding: 10px;
    font-size: 16px;
    box-sizing: border-box;
    border: 1px solid #ccc;
    border-radius: 5px;
    box-shadow: inset 0 2px 4px rgba(0, 0, 0, 0.1);
}

.submit-button {
    align-self: center;
```

```
    background-color: green;
    color: white;
    border: none;
    padding: 10px 20px;
    cursor: pointer;
    border-radius: 5px;
    box-shadow: 0 2px 4px rgba(0, 0, 0, 0.1);
    font-size: 16px;
}
```

Файл frontend/src/App.js

```
import React from 'react';
import { BrowserRouter as Router, Route, Routes } from 'react-router-dom';
import Login from './login/Login';
import Register from './register/Register';
import Dashboard from './dashboard/Dashboard';
import Transactions from './transactions/Transactions';
import Budgets from './budgets/Budgets';
import Balances from './balances/Balances';
import Analytics from './analytics/Analytics';
import Settings from './settings/Settings';
import PrivateRoute from './PrivateRoute';

const App = () => {
    return (
        <Router>
            <Routes>
                <Route path="/register" element={<Register />} />
                <Route path="/" element={<Login />} />
```

```
            <Route element={<PrivateRoute />}>
              <Route path="/home" element={<Dashboard />} />
              <Route path="/transactions" element={<Transactions />} />
              <Route path="/budgets" element={<Budgets />} />
              <Route path="/balances" element={<Balances />} />
              <Route path="/analytics" element={<Analytics />} />
              <Route path="/settings" element={<Settings />} />
            </Route>
          </Routes>
        </Router>
    );
};

export default App;
```

Файл frontend/src/index.css

```
body {
  height: 100%;
  width: 100%;
  margin: 0;
  font-family: -apple-system, BlinkMacSystemFont, 'Segoe UI', 'Roboto', 'Oxygen',
    'Ubuntu', 'Cantarell', 'Fira Sans', 'Droid Sans', 'Helvetica Neue',
    sans-serif;
  -webkit-font-smoothing: antialiased;
  -moz-osx-font-smoothing: grayscale;
}

#root {
  height: 100%;
```

```
  width: 100%;
}

code {
  font-family: source-code-pro, Menlo, Monaco, Consolas, 'Courier New',
    monospace;
}
```

Файл frontend/src/index.js

```
import React from 'react';
import ReactDOM from 'react-dom/client';
import './index.css';
import App from './App';
import reportWebVitals from './reportWebVitals';

const root = ReactDOM.createRoot(document.getElementById('root'));
root.render(
  <React.StrictMode>
    <App />
  </React.StrictMode>
);

// If you want to start measuring performance in your app, pass a function
// to log results (for example: reportWebVitals(console.log))
// or send to an analytics endpoint. Learn more: https://bit.ly/CRA-vitals
reportWebVitals();
```

Файл frontend/src/PrivateRoute.js

```
import React from 'react';
```

```
import { Navigate, Outlet } from 'react-router-dom';

const PrivateRoute = () => {
   const token = localStorage.getItem('token');

   return token ? <Outlet /> : <Navigate to="/" />;
};

export default PrivateRoute;
```

Файл frontend/src/reportWebVitals.js

```
const reportWebVitals = onPerfEntry => {
  if (onPerfEntry && onPerfEntry instanceof Function) {
    import('web-vitals').then(({ getCLS, getFID, getFCP, getLCP, getTTFB }) => {
      getCLS(onPerfEntry);
      getFID(onPerfEntry);
      getFCP(onPerfEntry);
      getLCP(onPerfEntry);
      getTTFB(onPerfEntry);
    });
  }
};

export default reportWebVitals;
```

Файл frontend/package.json

```
{
  "name": "frontend",
  "version": "0.1.0",
```

```
"private": true,
"dependencies": {
  "@testing-library/jest-dom": "^5.17.0",
  "@testing-library/react": "^13.4.0",
  "@testing-library/user-event": "^13.5.0",
  "axios": "^1.7.2",
  "chart.js": "^4.4.3",
  "date-fns": "^3.6.0",
  "react": "^18.3.1",
  "react-chartjs-2": "^5.2.0",
  "react-dom": "^18.3.1",
  "react-router-dom": "^6.23.1",
  "react-scripts": "5.0.1",
  "web-vitals": "^2.1.4"
},
"scripts": {
  "start": "react-scripts start",
  "build": "react-scripts build",
  "test": "react-scripts test",
  "eject": "react-scripts eject"
},
"eslintConfig": {
  "extends": [
    "react-app",
    "react-app/jest"
  ]
},
"browserslist": {
  "production": [
```

```
    ">0.2%",
    "not dead",
    "not op_mini all"
  ],
  "development": [
    "last 1 chrome version",
    "last 1 firefox version",
    "last 1 safari version"
  ]
 }
}
```

Файл .gitignore

```
HELP.md
target/
!.mvn/wrapper/maven-wrapper.jar
!**/src/main/**/target/
!**/src/test/**/target/
!**/src/main/**/credentials.json

### STS ###
.apt_generated
.classpath
.factorypath
.project
.settings
.springBeans
.sts4-cache
```

```
### IntelliJ IDEA ###
.idea
*.iws
*.iml
*.ipr

### NetBeans ###
/nbproject/private/
/nbbuild/
/dist/
/nbdist/
/.nb-gradle/
build/
!**/src/main/**/build/
!**/src/test/**/build/

### VS Code ###
.vscode/
/tokens/
/frontend/node_modules/
```

Файл docker-compose.yaml

```
version: '3.7'
services:
  postgres:
    image: postgres
    container_name: postgres
    restart: always
    environment:
```

```
      - POSTGRES_USER=postgres
      - POSTGRES_PASSWORD=postgres
    ports:
      - "5432:5432"
    volumes:
      - ./init.sql:/docker-entrypoint-initdb.d/init.sql

  base:
    build:
      context: ./backend/common/.
      dockerfile: Dockerfile
    image: base
    environment:
      - EMAIL_USERNAME=test
      - EMAIL_CONFIRMATION_URL=http://localhost:8080


  user-service:
    build:
      context: ./backend/service/user-service/.
      dockerfile: Dockerfile
    image: user-service
    environment:
      - SPRING_DATASOURCE_PASSWORD=postgres
      - SPRING_DATASOURCE_URL=jdbc:postgresql://postgres:5432/user-
service?stringtype=unspecified
      - SPRING_DATASOURCE_USERNAME=postgres
```

```
      -
CONSTANTS_SECRET_KEY=secretKeySecretKeySecretKeySecretKeySecretKe
ySecretKey
      - CONSTANTS_TOKEN_EXPIRATION_TIME=36000
      - EMAIL_USERNAME=test
      - EMAIL_CONFIRMATION_URL=http://localhost:8080
    ports:
      - "8080:8080"
      - "8888:8888"
    depends_on:
      - postgres
      - base

  money-management-service:
    build:
      context: ./backend/service/money-management-service/.
      dockerfile: Dockerfile
    image: money-management-service
    environment:
      - SPRING_DATASOURCE_PASSWORD=postgres
      - SPRING_DATASOURCE_URL=jdbc:postgresql://postgres:5432/money-
management-service?stringtype=unspecified
      - SPRING_DATASOURCE_USERNAME=postgres
      - EMAIL_USERNAME=test
      - EMAIL_CONFIRMATION_URL=http://localhost:8080
    ports:
      - "8081:8080"
      - "8889:8888"
    depends_on:
```

```
    - postgres
    - base

#  bank-integration-service:
#    build:
#      context: ./backend/service/bank-integration-service/.
#      dockerfile: Dockerfile
#    environment:
#      - SPRING_DATASOURCE_PASSWORD=postgres
#      - SPRING_DATASOURCE_URL=jdbc:postgresql://postgres:5432/bank-
integration-service
#      - SPRING_DATASOURCE_USERNAME=postgres
#    ports:
#      - "8082:8080"
#    depends_on:
#      - postgres
```

Файл init.sql

```
CREATE DATABASE "user-service";
CREATE DATABASE "money-management-service";
CREATE DATABASE "bank-integration-service";
```

Факультет інформатики та обчислювальної техніки
Кафедра інформатики та програмної інженерії

“ЗАТВЕРДЖЕНО”
Завідувач кафедри
_____ Едуард ЖАРІКОВ
“___” _____ 2024 р.

ВЕБ-ЗАСТОСУНОК ДЛЯ УПРАВЛІННЯ ОСОБИСТИМИ ФІНАНСАМИ

Програма та методика тестування

КПІ.ІТ.045440.05.34

“ПОГОДЖЕНО”
Керівник проєкту:
_____ Владислав СОКОЛОВСЬКИЙ

Нормоконтроль:
_____ Олександр СТЕЛЬМАХ

Виконавець:
_____ Владислав БЕРЕЗОВСЬКИЙ

Київ – 2024

ЗМІСТ

1 ОБ’ЄКТ ВИПРОБУВАНЬ 3
2 МЕТА ТЕСТУВАННЯ 4
3 МЕТОДИ ТЕСТУВАННЯ 5
4 ЗАСОБИ ТА ПОРЯДОК ТЕСТУВАННЯ 6

1 ОБ’ЄКТ ВИПРОБУВАНЬ

Об'єктом випробувань є розроблена веб-платформа для обліку особистих фінансів. Програмне забезпечення призначене для використання на різних платформах, включаючи операційні системи Windows, macOS і Linux. Крім того, воно підтримує роботу в різних веб-браузерах, таких як Google Chrome, Mozilla Firefox, Safari і Microsoft Edge.

2 МЕТА ТЕСТУВАННЯ

Метою тестування є:

- Перевірка відповідності роботи програмного забезпечення функціональним вимогам;
- Перевірка коректного збереження даних;
- Впевненість у сумісності веб-додатку з останніми версіями сучасних браузерів, таких як Chrome, Opera, Firefox і інші;
- Визначення сумісності застосунку з різними операційними системами, такими як Windows, Linux і інші;
- Виявлення та усунення проблем, помилок і недоліків у програмному забезпеченні;
- Оцінка зручності використання графічного інтерфейсу.

3 МЕТОДИ ТЕСТУВАННЯ

Для тестування програмного забезпечення використовуються такі методи:

- Функціональне тестування: перевірка роботи різних функцій веб-застосунку, таких як додавання транзакцій, перегляд звітів, редагування профілю користувача тощо;
- Відповідність до специфікації: перевірка того, чи відповідає робота веб-застосунку вимогам і специфікаціям, що були визначені на етапі розробки;
- Сумісність з різними браузерами: перевірка того, як веб-застосунок працює на різних веб-переглядачах, таких як Chrome, Firefox, Opera тощо;
- Тестування відповідності до дизайну: перевірка, чи відображається інтерфейс веб-застосунку правильно на різних пристроях і розмірах екранів;
- Мануальне тестування: виконання тест-кейсів вручну для перевірки функцій, які не піддаються автоматизації або для виявлення непередбачених проблем;
- Тестування безпеки: перевірка вразливостей системи, захист від вторгнень та зловмисного використання особистих даних;
- Тестування швидкодії: вимірювання швидкості відкриття сторінок, завантаження даних та виконання операцій для забезпечення оптимальної продуктивності веб-застосунку;
- Тестування на різних операційних системах: перевірка роботи веб-застосунку на різних операційних системах, таких як Windows, Linux, macOS тощо.

4 ЗАСОБИ ТА ПОРЯДОК ТЕСТУВАННЯ

Тестування виконується мануально з використанням наскрізного тестування (End-to-end, E2E, Chain testing) та написанням unit-тестів, з метою знаходження помилок та недоліків як у функціональній частині програмного забезпечення так і в зручності користування. Для того, щоб перевірити працездатність та відмовостійкість застосунку, необхідно провести наступні тестування:

- Мануальне тестування: Виконання тест-кейсів на основні функціональні можливості веб-застосунку. Перевірка зручності взаємодії з користувачем. Тестування різних сценаріїв використання для виявлення непередбачених проблем;
- Написання unit-тестів: Розробка автоматизованих тестів для перевірки окремих модулів або компонентів веб-застосунку. Використання фреймворків для тестування, таких як JUnit, для забезпечення надійності та стабільності коду;
- Тестування на різних пристроях та операційних системах: Динамічне тестування на різних версіях веб-переглядачів (Chrome, Firefox, Safari, Edge тощо). Перевірка сумісності з різними версіями операційних систем (Windows, macOS, Android, iOS). Тестування на пристроях з різною роздільною здатністю екрану для впевненості у коректному відображенні веб-застосунку;
- Тестування функціональності: Перевірка відповідності функціональних можливостей веб-застосунку вимогам, зазначеним у специфікації. Тестування зміни орієнтації екрану на мобільних пристроях. Тестування реакції програми на відсутність мережевого з'єднання та виведення відповідних повідомлень про помилку;
- Тестування безпеки: Перевірка наявності захисту від потенційних загроз, таких як вразливості введення даних, атаки XSS або SQL-

ін'єкції. Тестування забезпечення конфіденційності особистих фінансових даних користувачів.

Факультет інформатики та обчислювальної техніки
Кафедра інформатики та програмної інженерії

“ЗАТВЕРДЖЕНО”
Завідувач кафедри
_____ Едуард ЖАРІКОВ
“___” _____ 2024 р.

ВЕБ-ЗАСТОСУНОК ДЛЯ УПРАВЛІННЯ ОСОБИСТИМИ ФІНАНСАМИ

Керівництво користувача

КПІ.ІТ.045440.05.34

“ПОГОДЖЕНО”
Керівник проєкту:
_____ Владислав СОКОЛОВСЬКИЙ

Нормоконтроль:
_____ Олександр СТЕЛЬМАХ

Виконавець:
_____ Владислав БЕРЕЗОВСЬКИЙ

Київ – 2024

ЗМІСТ

1 ПРИЗНАЧЕННЯ ПРОГРАМИ 3
2 ПІДГОТОВКА ДО РОБОТИ З ПРОГРАМНИМ ЗАБЕЗПЕЧЕННЯМ 5
2.1 Системні вимоги для коректної роботи 5
2.2 Дії, необхідні для отримання доступу до веб-застосунку 5
2.3 Перевірка коректної роботи 6
3 ВИКОНАННЯ ПРОГРАМИ 8

1 ПРИЗНАЧЕННЯ ПРОГРАМИ

Короткий опис програмного забезпечення

Веб-застосунок для обліку особистих фінансів є інтерактивною платформою, призначеною для управління особистими доходами та витратами. Основна мета застосунку — допомогти користувачам ефективно контролювати свої фінанси, аналізувати фінансові дані та приймати обґрунтовані рішення щодо витрат.

Кінцева збірка веб-застосунку включає в себе наступні компоненти:

- Інтерфейс користувача (UI): Побудований за допомогою сучасних фреймворків React для забезпечення інтерактивності та зручності використання;
- Серверна частина: Реалізована на основі Java Spring Boot, яка обробляє запити від клієнтської частини, керує базою даних і забезпечує бізнес-логіку застосунку;
- База даних: Реляційна база даних PostgreSQL для зберігання фінансових даних користувачів;
- Система автентифікації: Забезпечує захищений вхід користувачів і управління їхніми даними;
- API: Забезпечує взаємодію між клієнтською та серверною частинами, дозволяючи безпечно обмінюватися даними.

Інсталяційна версія веб-застосунку включає:

- Архів з вихідним кодом застосунку;
- Скрипти для розгортання застосунку на веб-сервері;
- Файл налаштувань docker-compose.yaml з прикладами необхідних параметрів.

Репозиторій веб-застосунку містить наступні основні директорії та файли:

- /backend/service/src: Вихідний код застосунку, включаючи модулі серверної частини, утиліти;
- /frontend/src: Статичні файли, такі як JS, CSS;
- /frontend/public: Статичні файли, такі як зображення;
- /backend/service/src/main/resources: Конфігураційні файли для налаштування різних середовищ (розробка, тестування, продуктивність);
- /backend/service/src/test: Тестові кейси для різних видів тестування (unit-тести, інтеграційні тести).

2 ПІДГОТОВКА ДО РОБОТИ З ПРОГРАМНИМ ЗАБЕЗПЕЧЕННЯМ

2.1 Системні вимоги для коректної роботи

Мінімальна конфігурація технічних засобів:

– тип процесору: Intel Core i5;
– об‘єм ОЗП: 4 Гб;
– підключення до мережі Інтернет зі швидкістю від 20 мегабіт.

Рекомендована конфігурація технічних засобів:

– тип процесору: Intel Core i7;
– об‘єм ОЗП: 8 Гб;
– підключення до мережі Інтернет зі швидкістю від 100 мегабіт.

2.2 Дії, необхідні для отримання доступу до веб-застосунку

Щоб відкрити веб-застосунок для обліку особистих фінансів у вашому браузері, виконайте наступні дії:

1) Запустіть будь-який сучасний веб-браузер, наприклад, Google Chrome, Mozilla Firefox, Microsoft Edge або Safari;
2) В адресному рядку веб-браузера введіть URL-адресу веб-застосунку. Натисніть клавішу Enter;
3) Введіть свої облікові дані (ім'я користувача та пароль) у відповідні поля на сторінці входу;
4) Натисніть кнопку "Увійти" для доступу до функціональності застосунку;
5) Після входу ви потрапите на головну сторінку веб-застосунку. Використовуйте меню або інші навігаційні елементи для доступу до різних розділів та функцій, таких як:
   – Додавання нових фінансових транзакцій;
   – Перегляд графіків про доходи та витрати;
   – Налаштування бюджетів та фінансових цілей;

- Аналіз фінансових даних.

6) Після завершення роботи з веб-застосунком не забудьте вийти зі свого облікового запису. Для цього потрібно натиснути на опцію "Вийти" у верхньому правому куті сторінки;

Застосунок оптимізовано для роботи у наступних веб-браузерах:

- Google Chrome (остання версія);
- Mozilla Firefox (остання версія);
- Microsoft Edge (остання версія);
- Safari (остання версія).

Для забезпечення найкращого користувацького досвіду рекомендується використовувати одну з останніх версій цих браузерів.

2.3 Перевірка коректної роботи

Дії, необхідні для перевірки коректності процесу установки чи розгортання програмного забезпечення

1) Перевірка доступності веб-застосунку:
   - Відкрийте веб-браузер;
   - Введіть URL-адресу веб-застосунку в адресний рядок і натисніть клавішу Enter;
   - Переконайтеся, що завантажується головна сторінка застосунку без помилок.
2) Перевірка авторизації:
   - На сторінці входу введіть облікові дані (ім'я користувача та пароль);
   - Натисніть кнопку "Увійти";
   - Переконайтеся, що ви потрапляєте на головну сторінку застосунку після успішної авторизації.
3) Перевірка основних функцій застосунку:

a. Додавання нової транзакції:
    i. Перейдіть до розділу додавання транзакції;
    ii. Введіть дані про нову транзакцію (сума, категорія, опис);
    iii. Натисніть кнопку "Зберегти" або аналогічну;
    iv. Переконайтеся, що нова транзакція з’являється у списку транзакцій.
b. Перегляд графіків:
    i. Перейдіть до розділу аналітики;
    ii. Перевірте, що графіки генеруються правильно, відображаються коректно.
c. Налаштування бюджетів:
    i. Перейдіть до розділу налаштувань бюджетів;
    ii. Додайте або змініть бюджет;
    iii. Переконайтеся, що зміни зберігаються та відображаються правильно.

4) Перевірка роботи на різних браузерах:

Повторіть кроки 1-3 на різних веб-браузерах (Google Chrome, Mozilla Firefox, Microsoft Edge, Safari) для переконання в сумісності застосунку.

5) Перевірка роботи на різних операційних системах:

Повторіть кроки 1-3 на різних операційних системах (Windows, Linux, macOS) для переконання в сумісності застосунку.

3 ВИКОНАННЯ ПРОГРАМИ

1. Запуск програми

1. **Відкриття веб-браузера:**
    - Відкрийте будь-який сучасний веб-браузер, який підтримується (Google Chrome, Mozilla Firefox, Microsoft Edge, Safari).
2. **Доступ до веб-застосунку:**
    - У рядок адреси введіть URL-адресу веб-застосунку і натисніть клавішу Enter.

2. Авторизація

1. **Вхід до системи:**
    - На сторінці входу введіть свої облікові дані (ім'я користувача та пароль);



Рисунок 3.1 – Сторінка авторизації

- Натисніть кнопку "Увійти";
- Після успішної авторизації ви потрапите на головну сторінку застосунку.

3. Основні функції

1. **Додавання нової транзакції:**

- Перейдіть до розділу "Транзакції" у навігаційному меню;
- Натисніть “Додати”;
- Введіть необхідні дані про транзакцію: сума, категорія, опис;

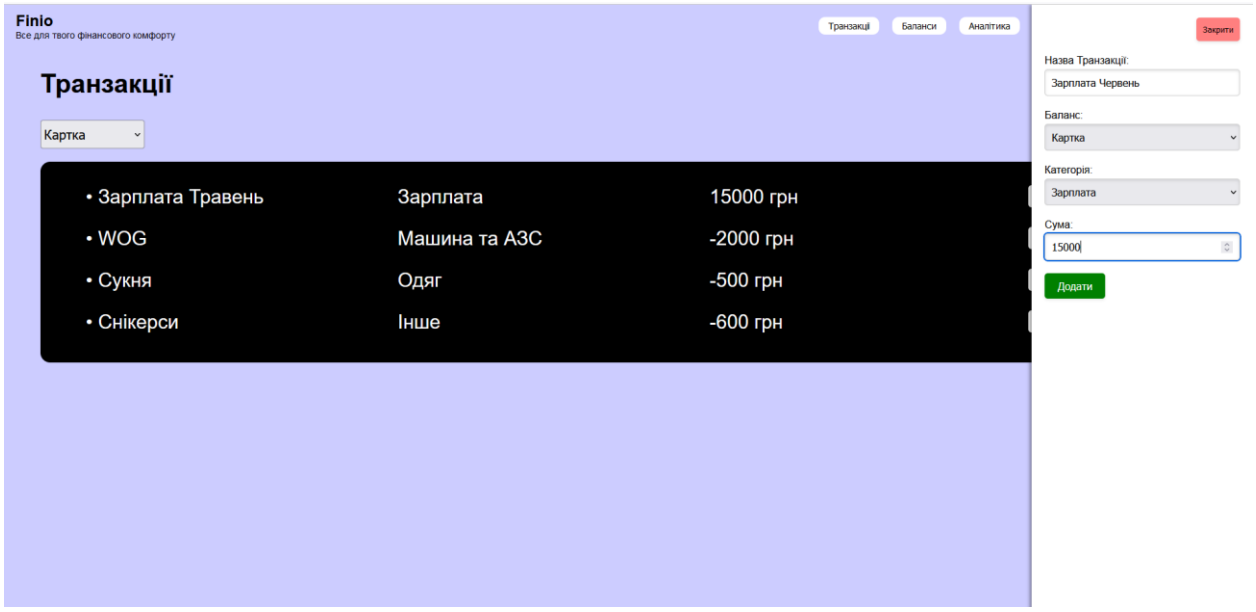


Рисунок 3.2 – Додавання транзакції

- Натисніть кнопку "Додати";
- Переконайтеся, що нова транзакція з’явилася у списку транзакцій.

2. **Перегляд та аналіз транзакцій:**
   - Перейдіть до розділу "Транзакції" у навігаційному меню;
   - Використовуйте фільтри для сортування та пошуку транзакцій за різними параметрами (дата, категорія, сума);
   - Переглядайте деталі окремих транзакцій за потреби.
3. **Створення та управління бюджетами:**
   - Перейдіть до розділу "Бюджети" у навігаційному меню;
   - Натисніть кнопку "Додати";
   - Введіть параметри бюджету: назва, категорія, ліміт;



Рисунок 3.3 – Додавання бюджету

- Натисніть кнопку "Додати";
- Переконайтеся, що новий бюджет з’явився у списку бюджетів.

4. **Перегляд графіків:**
   - Перейдіть до розділу "Аналітика" у навігаційному меню;
   - Ознайомтесь з графіками та аналітикою, що відображають ваші фінансові дані.



Рисунок 3.4 – Сторінка аналітики

4. Налаштування

1. **Налаштування профілю:**
    - Перейдіть до розділу "Налаштування" у навігаційному меню;
    - Оновіть особисту інформацію, таку як ім'я, електронна пошта, пароль;
    - Збережіть зміни, натиснувши кнопку "Оновити".
2. **Налаштування категорій:**
    - Перейдіть до розділу "Категорії" у навігаційному меню;
    - Додайте нові категорії або редагуйте існуючі для кращого обліку транзакцій;
    - Збережіть зміни, натиснувши кнопку "Зберегти".

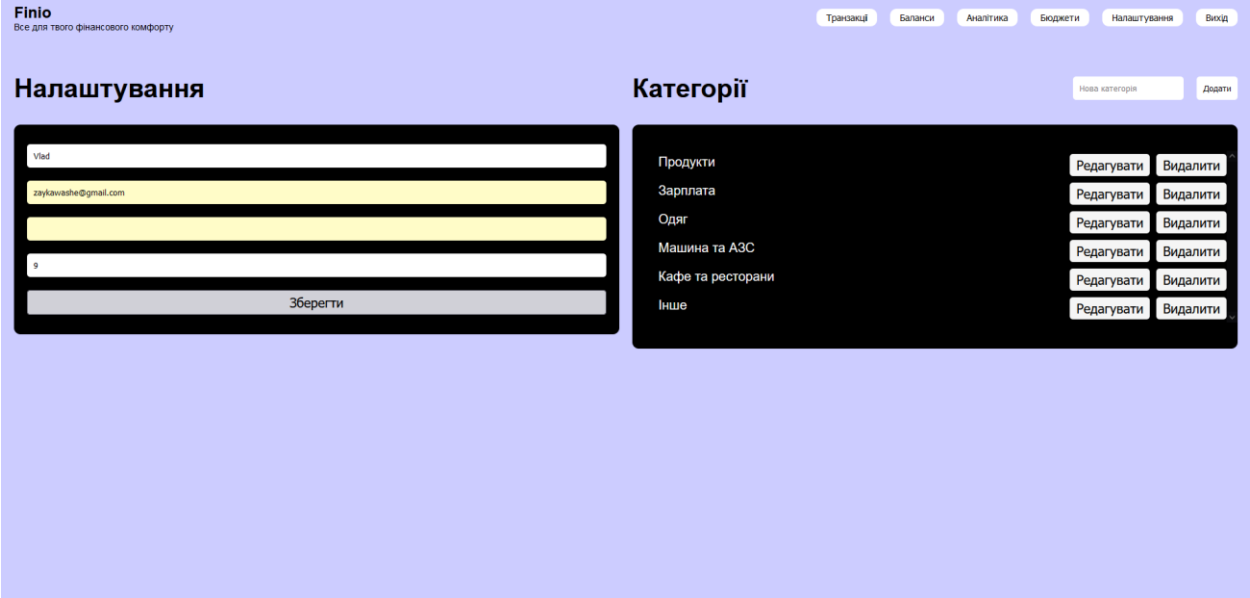


Рисунок 3.5 – Сторінка налаштувань

5. Завершення роботи

1. **Вихід з системи:**
    - Для завершення роботи натисніть “Вийти” у верхньому правому куті інтерфейсу;
    - Після виходу ви повернетеся на сторінку входу.
2. **Закриття веб-браузера:**
    - Закрийте вкладку з веб-застосунком або закрийте сам веб-браузер.

Факультет інформатики та обчислювальної техніки
Кафедра інформатики та програмної інженерії

“ЗАТВЕРДЖЕНО”
Завідувач кафедри
_____ Едуард ЖАРІКОВ
“___” _____ 2024 р.

ВЕБ-ЗАСТОСУНОК ДЛЯ УПРАВЛІННЯ ОСОБИСТИМИ ФІНАНСАМИ

Графічний матеріал

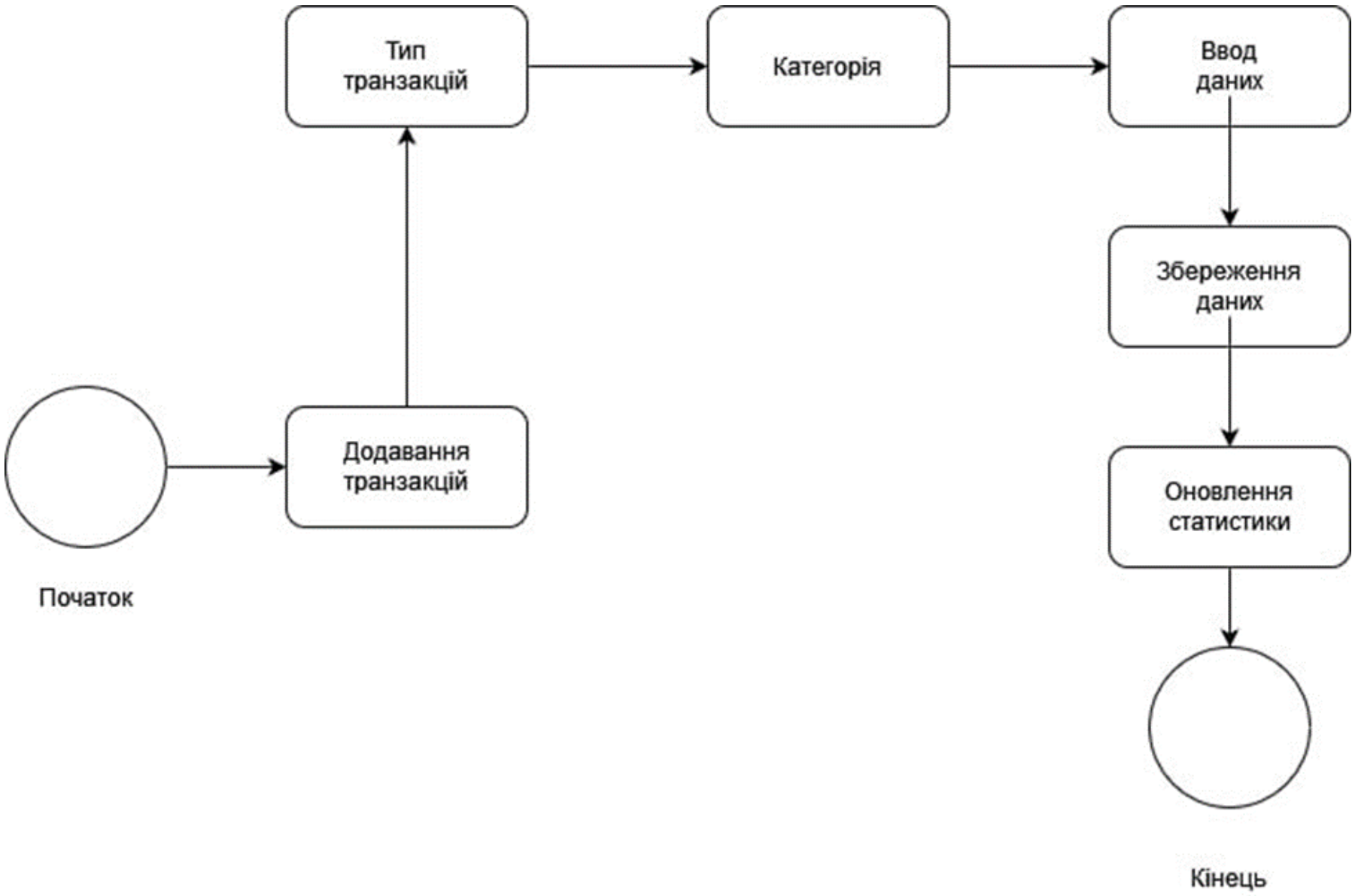
КПІ.ІТ.045440.06.99

“ПОГОДЖЕНО”
Керівник проєкту:
_____ Владислав СОКОЛОВСЬКИЙ

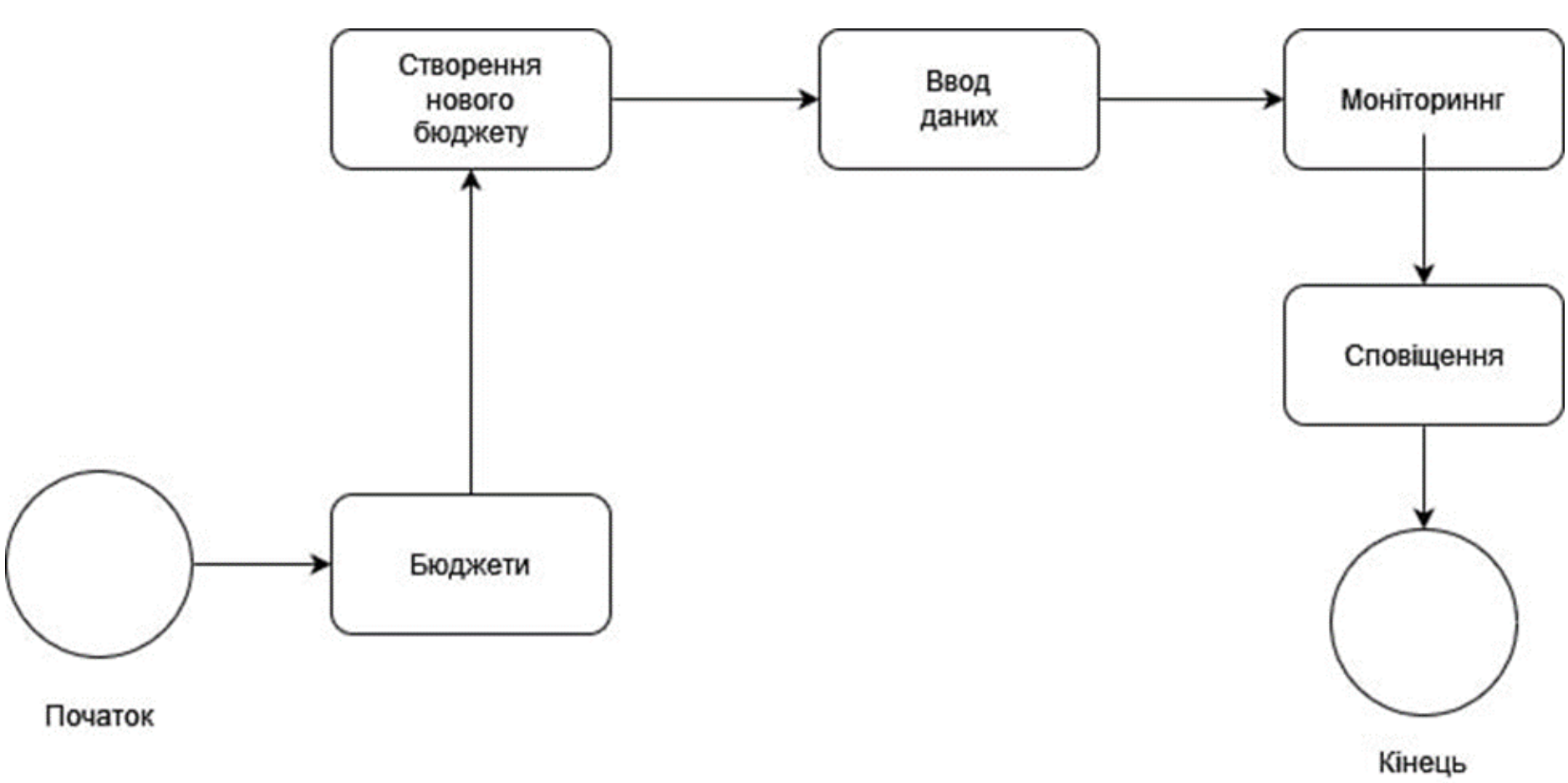
Нормоконтроль:
_____ Олександр СТЕЛЬМАХ

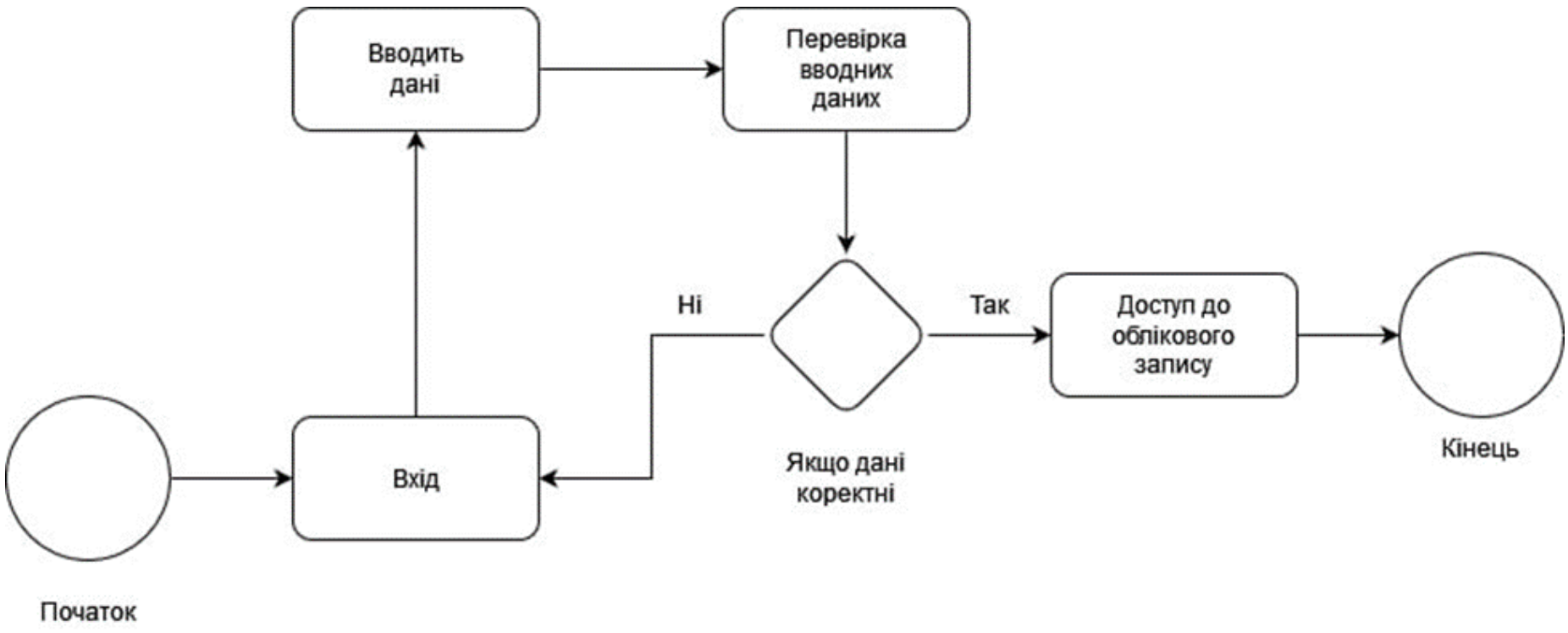
Виконавець:
_____ Владислав БЕРЕЗОВСЬКИЙ

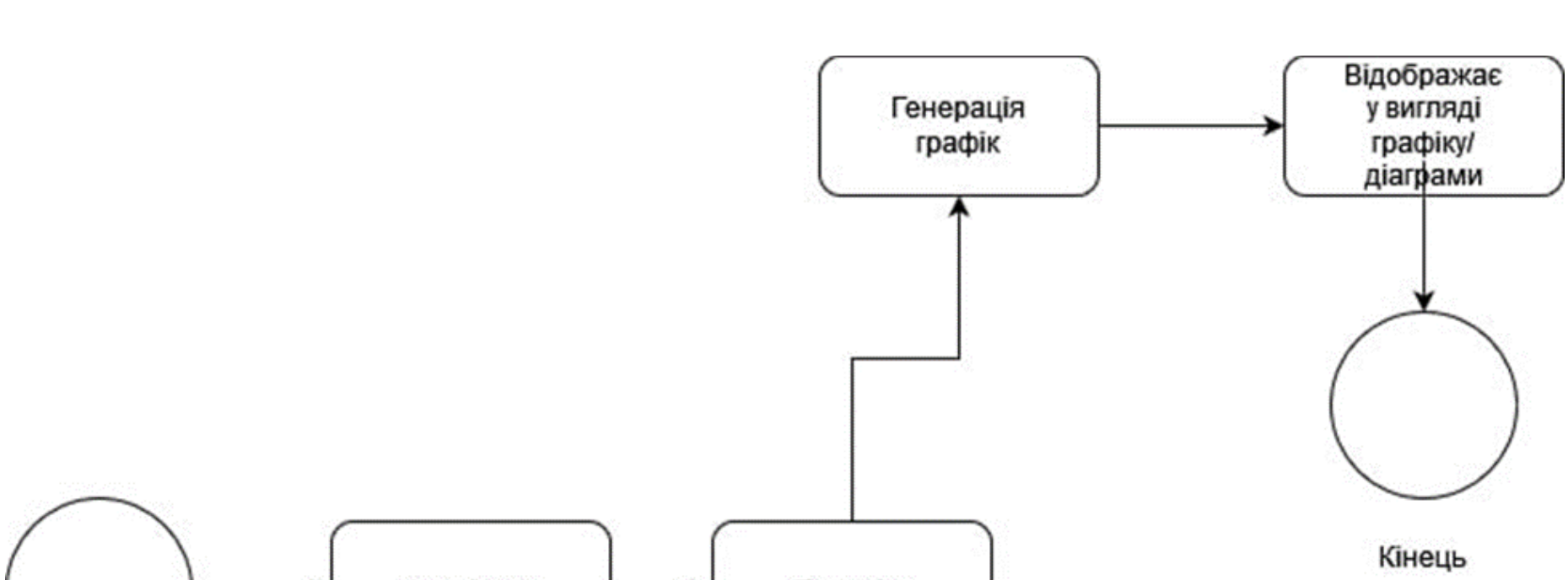
Київ – 2024


Тип транзакцій
Категорія
Ввод даних
Збереження даних
Оновлення статистики
Додавання транзакцій
Початок
Кінець


Вводить дані
Перевірка вводних даних
Ні
Так
Створення облікового запису
Реєстрація
Якщо дані коректні
Відправка підтвердження на пошту
Підтвердження пошти
Початок
Кінець

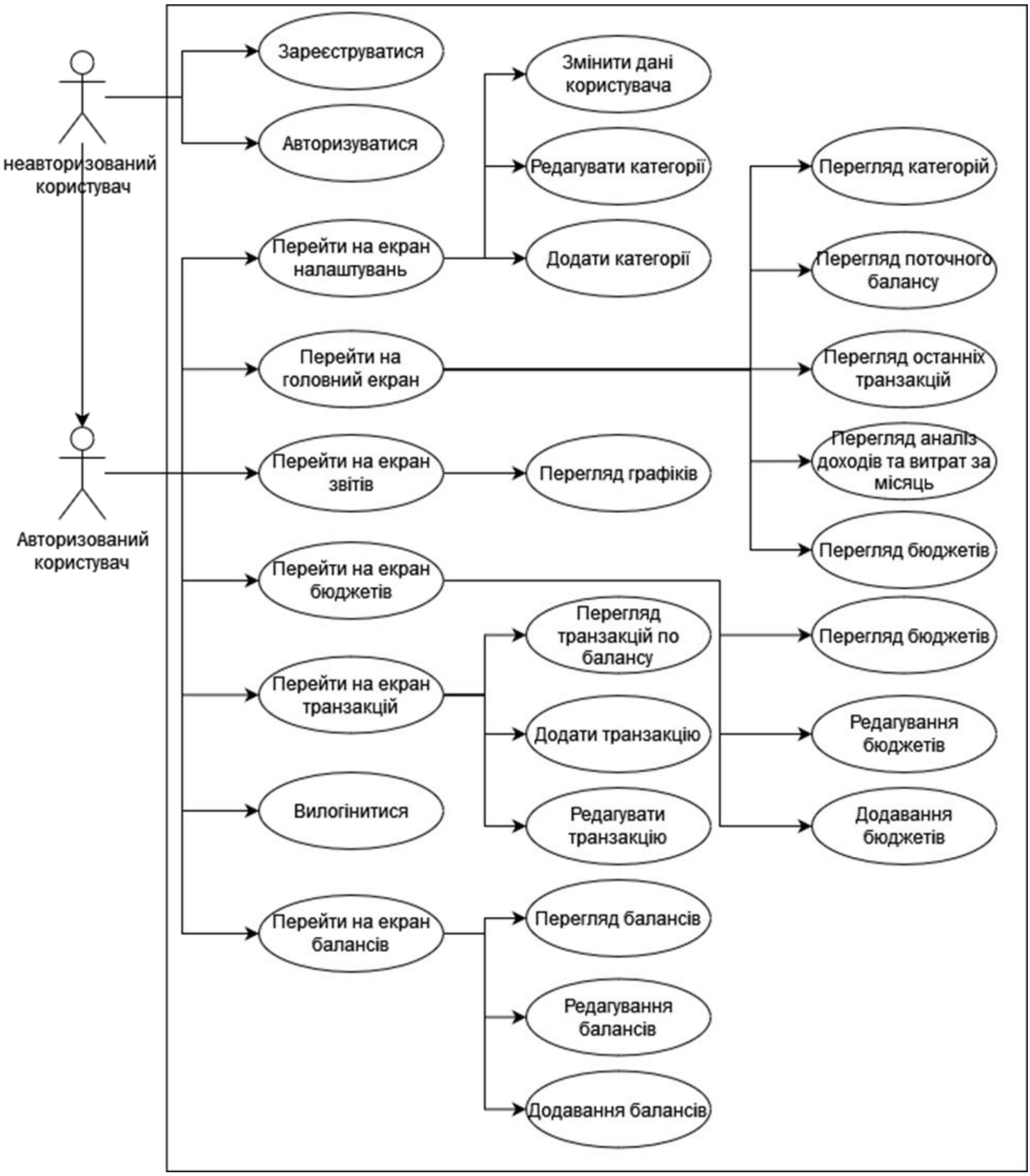

Створення нового бюджету
Ввод даних
Моніторинг
Сповіщення
Бюджети
Початок
Кінець

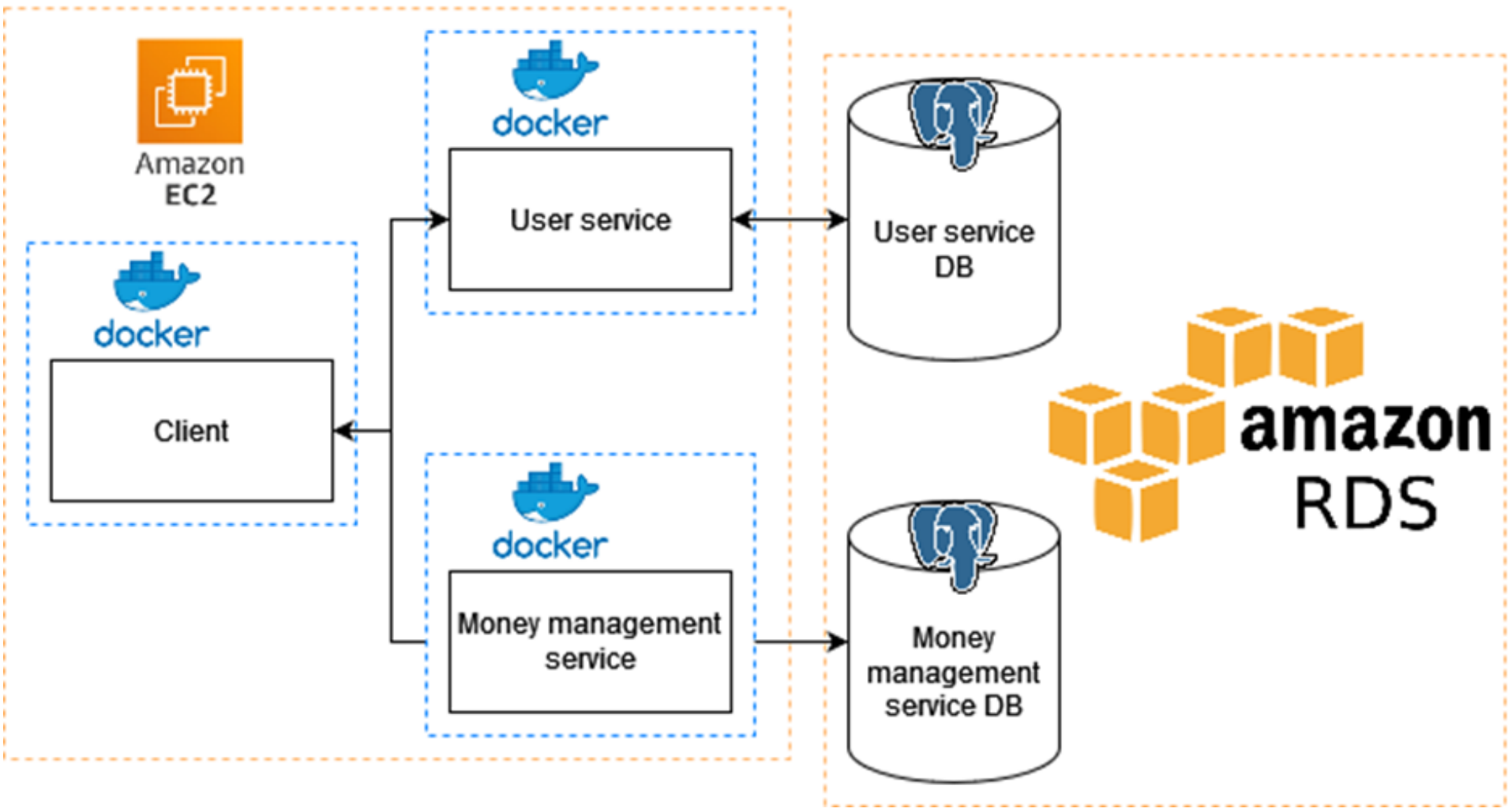

Вводить дані
Перевірка вводних даних
Ні
Так
Доступ до облікового запису
Початок
Вхід
Якщо дані коректні
Кінець

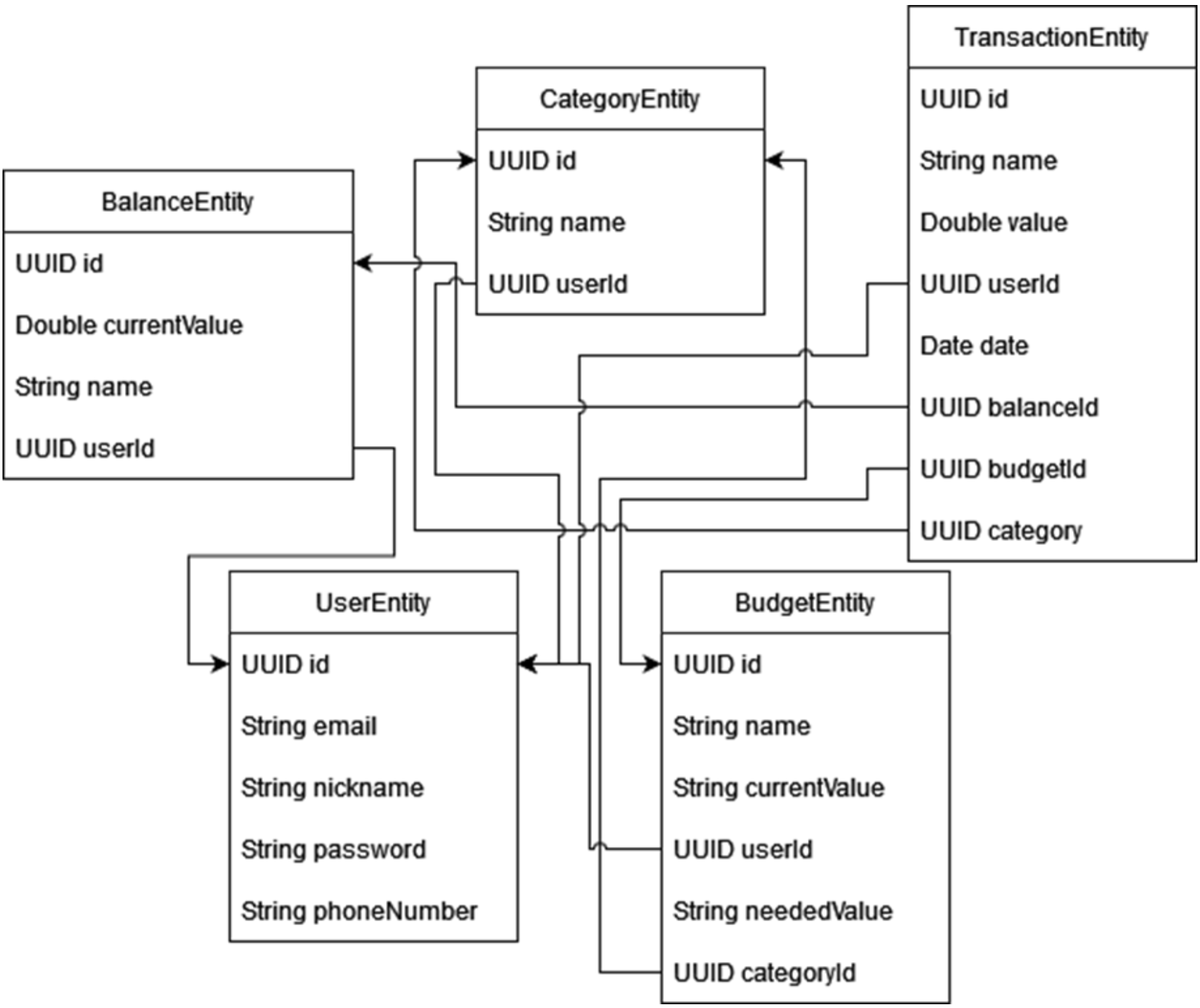

Генерація графік
Відображає у вигляді графіку/ діаграми
Початок
Аналітика
Тип звіту
Кінець

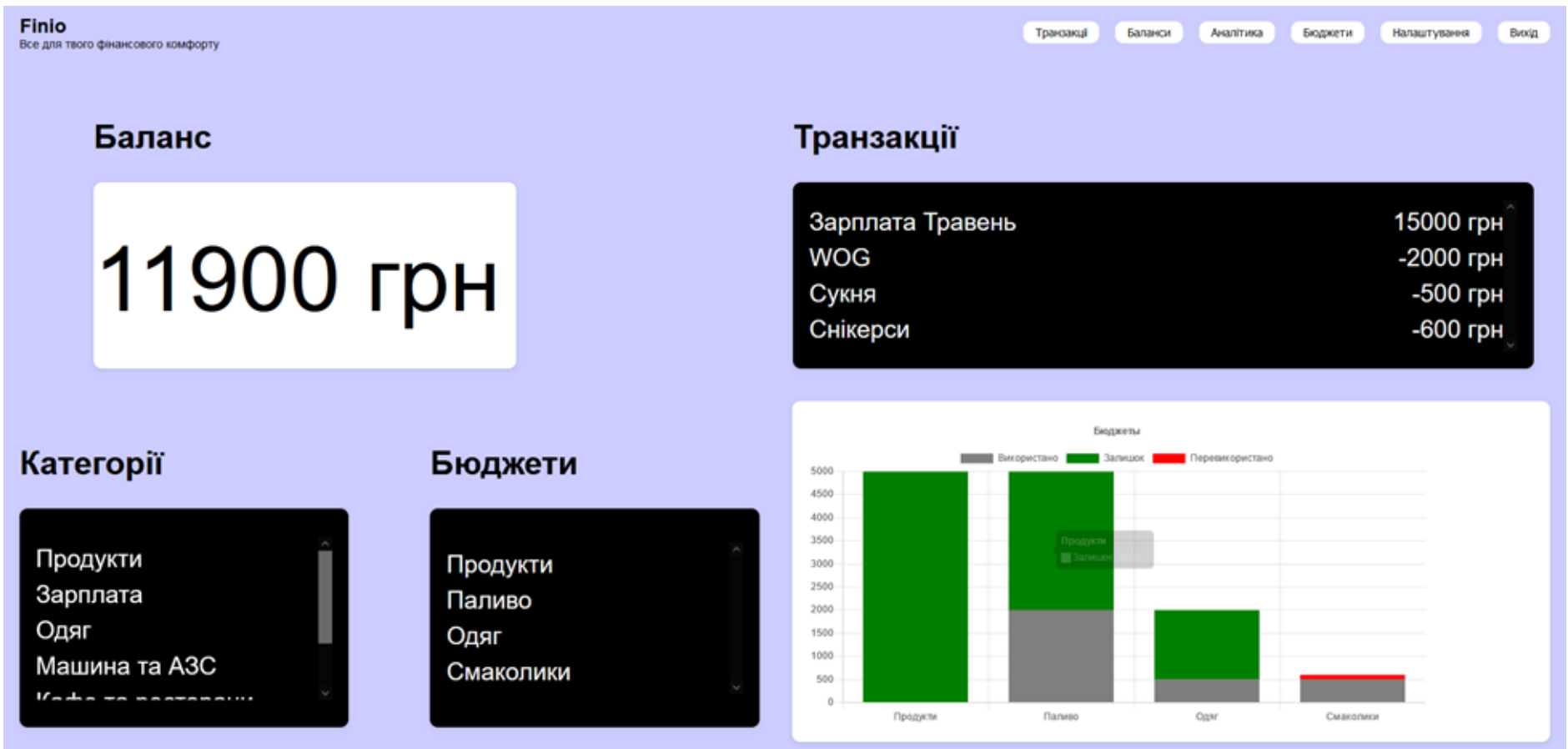

Початок
Налаштування користувача
Створення нових категорій/ редагування
Збереження
Оновлення даних
Кінець


Початок
Налаштування нагадувань
Відправка сповіщень/ нагадувань
Кінець


неавторизований користувач
Авторизований користувач
Зареєструватися
Авторизуватися
Перейти на екран налаштувань
Змінити дані користувача
Редагувати категорії
Додати категорії
Перейти на головний екран
Перейти на екран звітів
Перегляд графіків
Перейти на екран бюджетів
Перейти на екран транзакцій
Перегляд транзакцій по балансу
Додати транзакцію
Редагувати транзакцію
Вилогінитися
Перейти на екран балансів
Перегляд балансів
Редагування балансів
Додавання балансів
Перегляд категорій
Перегляд поточного балансу
Перегляд останніх транзакцій
Перегляд аналіз доходів та витрат за місяць
Перегляд бюджетів
Перегляд бюджетів
Редагування бюджетів
Додавання бюджетів


Amazon
EC2
docker
User service
User service
DB
docker
Client
docker
Money management
service
Money
management
service DB
amazon
RDS

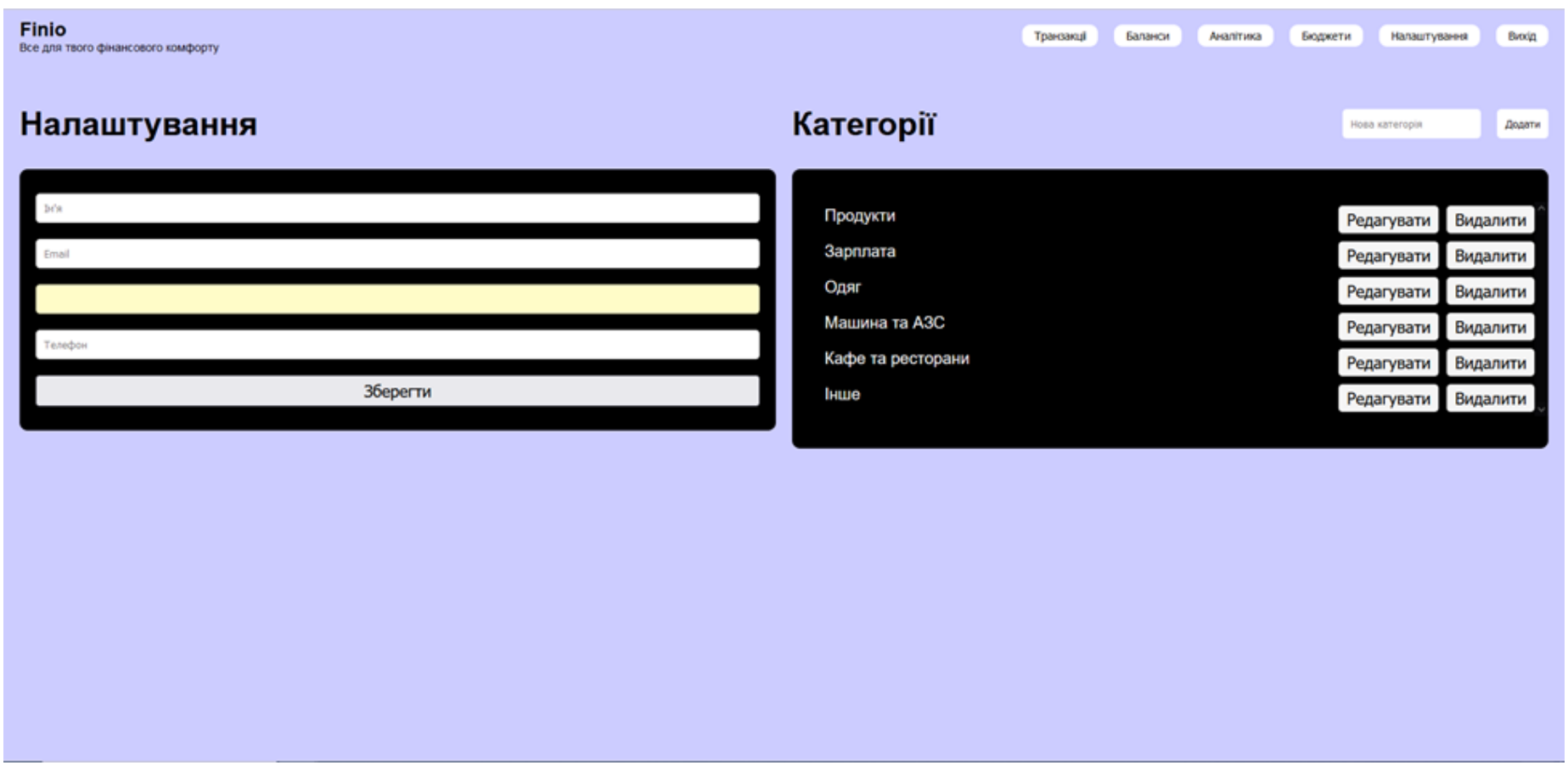

BalanceEntity
UUID id
Double currentValue
String name
UUID userId
CategoryEntity
UUID id
String name
UUID userId
TransactionEntity
UUID id
String name
Double value
UUID userId
Date date
UUID balanceId
UUID budgetId
UUID category
UserEntity
UUID id
String email
String nickname
String password
String phoneNumber
BudgetEntity
UUID id
String name
String currentValue
UUID userId
String neededValue
UUID categoryId

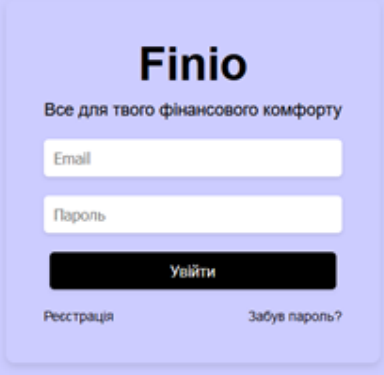

Finio
Все для твого фінансового комфорту
Транзакції
Баланси
Аналітика
Бюджети
Налаштування
Вихід
Баланс
11900 грн
Транзакції
Зарплата Травень 15000 грн
WOG -2000 грн
Сукня -500 грн
Снікерси -600 грн
Категорії
Продукти
Зарплата
Одяг
Машина та АЗС
Бюджети
Продукти
Паливо
Одяг
Смаколики
Бюджеты
Використано
Залишок
Перевикористано
5000
4500
4000
3500
3000
2500
2000
1500
1000
500
0
Продукти
Паливо
Одяг
Смаколики

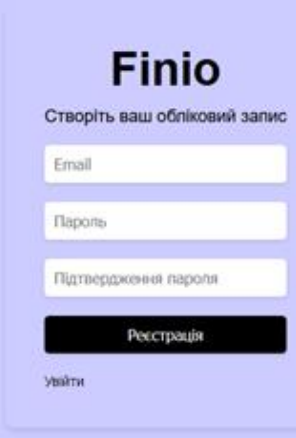

Finio
Все для твого фінансового комфорту
Транзакції
Баланси
Аналітика
Бюджети
Налаштування
Вихід
Транзакції
Картка
Додати
• Зарплата Травень
Зарплата
15000 грн
Редагувати
Видалити
• WOG
Машина та АЗС
-2000 грн
Редагувати
Видалити
• Сукня
Одяг
-500 грн
Редагувати
Видалити
• Снікерси
Інше
-600 грн
Редагувати
Видалити


Finio
Все для твого фінансового комфорту
Транзакції
Баланси
Аналітика
Бюджети
Налаштування
Вихід
Бюджети
Додати
Продукти 0/5000 Редагувати Видалити
Паліво 2000/5000 Редагувати Видалити
Одяг 500/2000 Редагувати Видалити
Смаколики 600/500 Редагувати Видалити
Бюджети
Використано
Залишок
Перевикористано
5000
4500
4000
3500
3000
2500
2000
1500
1000
500
0
Продукти
Паліво
Одяг
Смаколики


Finio
Все для твого фінансового комфорту
Транзакції
Баланси
Аналітика
Бюджети
Налаштування
Вихід
Аналітика
Графік транзакцій
Доходи
Розходи
16 000
14 000
12 000
10 000
8000
6000
4000
2000
0
-2000
-4000
31 May
31 May
31 May
31 May
Бюджети
Використано
Залишок
Перевикористано
5000
4500
4000
3500
3000
2500
2000
1500
1000
500
0
Продукти
Паліво
Одяг
Смаколики


Finio
Все для твого фінансового комфорту
Транзакції
Баланси
Аналітика
Бюджети
Налаштування
Вихід
Налаштування
Ім'я
Email
Телефон
Зберегти
Категорії
Нова категорія
Додати
Продукти Редагувати Видалити
Зарплата Редагувати Видалити
Одяг Редагувати Видалити
Машина та АЗС Редагувати Видалити
Кафе та ресторани Редагувати Видалити
Інше Редагувати Видалити


Finio
Все для твого фінансового комфорту
Email
Пароль
Увійти
Реєстрація
Забув пароль?


Finio
Створіть ваш обліковий запис
Email
Пароль
Підтвердження пароля
Реєстрація
Увійти


Finio
Все для твого фінансового комфорту
Транзакції
Баланси
Аналітика
Бюджети
Налаштування
Вихід
Баланси
Додати
• Готівка
0 грн
Редагувати
Видалити
• Картка
11900 грн
Редагувати
Видалити