MoneyTracker

Aplikacja do Zarządzania Finansami Osobistymi

Autorzy: Mateusz Kozłowski, Patryk Kustra

8 grudnia 2025

Spis treści

1 Ogólna Charakterystyka Projektu

MoneyTracker to aplikacja webowa służąca do kompleksowego zarządzania finansami osobistymi. System umożliwia użytkownikom śledzenie przepływów pieniężnych (dochodów i wydatków), monitorowanie zobowiązań oraz współdzielenie portfeli z członkami rodziny lub innymi użytkownikami w czasie rzeczywistym.

1.1 Kluczowe Cechy

- **Multi-portfele**: Możliwość tworzenia wielu niezależnych portfeli (np. Osobisty, Domowy, Firmowy, Oszczędnościowy).
- **Współdzielenie**: Zaawansowany system uprawnień pozwalający na dodawanie użytkowników do portfeli z rolami *OWNER* lub *MEMBER*.
- **Transakcje**: Rejestrowanie przychodów i wydatków z możliwością szczegółowej kategoryzacji.
- **Zobowiązania**: Dedykowany moduł do śledzenia pożyczek, kredytów i kart kredytowych.
- **Dashboard**: Centralny panel analityczny prezentujący stan finansów i podsumowania.
- **Dostępność**: Responsywny interfejs webowy (RWD) dostosowany do komputerów i urządzeń mobilnych.

2 Cel Projektu

2.1 Cel Główny

Stworzenie intuicyjnego i bezpiecznego narzędzia webowego, które ułatwia kontrolę nad budżetem domowym oraz wspiera podejmowanie świadomych decyzji finansowych poprzez transparentne współdzielenie danych między użytkownikami.

2.2 Cele Szczegółowe

1. Centralizacja wszystkich operacji finansowych w jednym systemie webowym.
2. Umożliwienie elastycznego zarządzania budżetem grupowym (rodzina/współlokatorzy).
3. Zapewnienie kontroli nad terminowością spłat zobowiązań.
4. Dostarczenie czytelnej analityki finansowej w czasie rzeczywistym.

3 Opis Biznesowy

3.1 Grupa Docelowa

- **Osoby prywatne**: Chcące mieć pełną kontrolę nad własnymi wydatkami.
- **Rodziny i Pary**: Potrzebujące wspólnego wglądu w budżet domowy.
- **Współlokatorzy**: Dzielący rachunki i wydatki na mieszkanie.
- **Freelancerzy**: Potrzebujący separacji finansów osobistych od firmowych.

3.2 Propozycja Wartości (USP)

- **Real-time Collaboration**: Natychmiastowa widoczność wydatków partnera.
- **Bezpieczeństwo**: Standardowe mechanizmy sesyjne Spring Security i haszowanie haseł.
- **Kompleksowość**: Połączenie trackera wydatków z menedżerem długów.

4 Założenia Projektowe

4.1 Stack Technologiczny

- **Backend**: Java 21, Spring Boot 3.x
- **Frontend**: React 18, TypeScript, Redux Toolkit
- **Baza danych**: PostgreSQL 15+
- **Konteneryzacja**: Docker (Docker Compose)

5 Wymagania

5.1 Wymagania Funkcjonalne

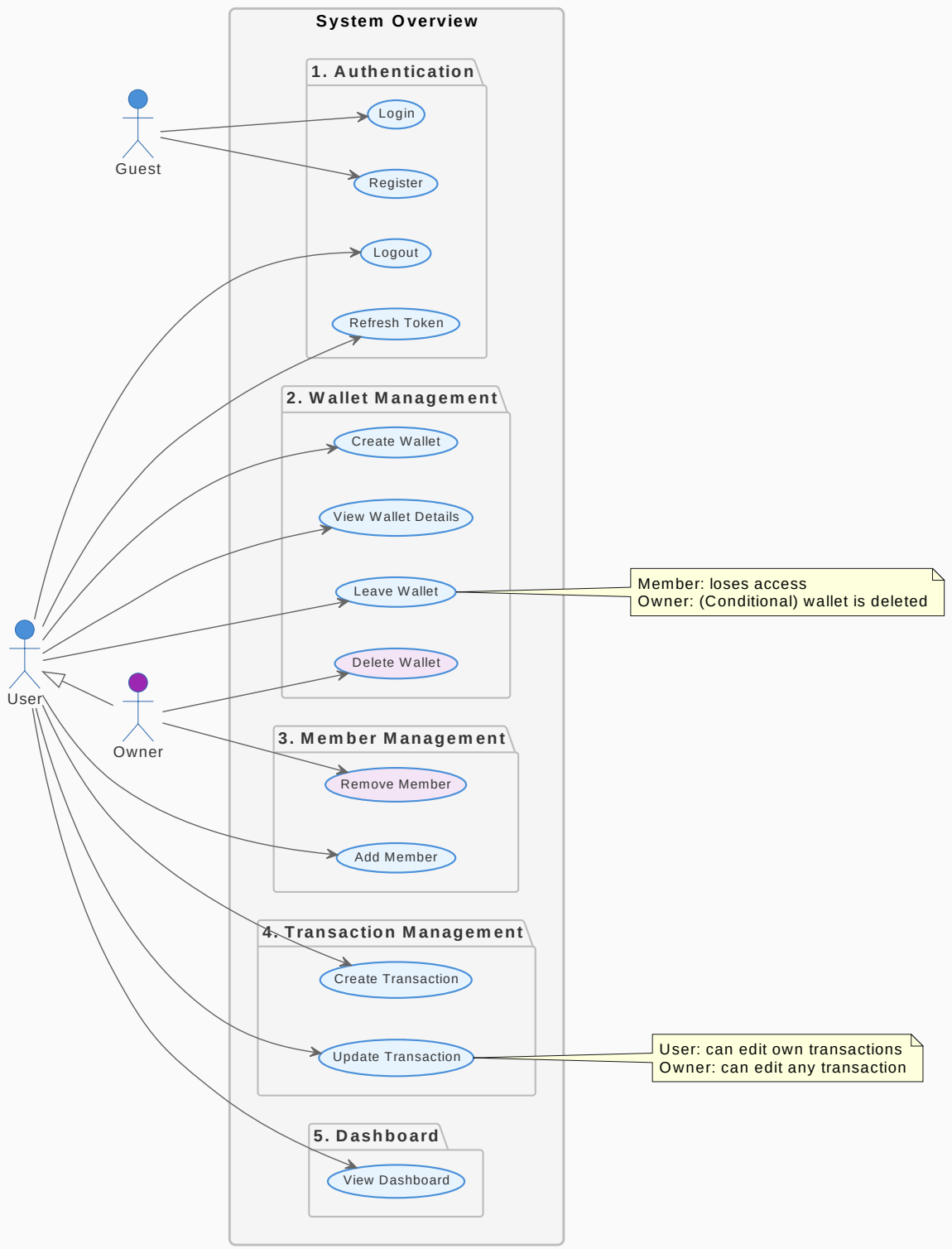
ID	Opis Wymagania
FR1	System umożliwia rejestrację użytkownika z walidacją unikalności adresu email.
FR2	System tworzy bezpieczną sesję użytkownika (Cookie) po poprawnej autentykacji.
FR3	Użytkownik może tworzyć wiele portfeli o różnych typach (np. Family, Business).
FR4	Rola OWNER oraz MEMBER pozwala na dodawanie nowych członków do portfela.
FR5	Wyłącznie rola OWNER posiada uprawnienie do usuwania członków z portfela.
FR6	MEMBER edytuje tylko własne transakcje, OWNER ma pełną edycję.
FR7	Spłata zobowiązania automatycznie generuje transakcję i aktualizuje saldo długu.
FR8	Dashboard agreguje dane asynchronicznie, nie blokując interfejsu użytkownika.

5.2 Wymagania Niefunkcjonalne

ID	Opis Wymagania
NFR1	Średni czas odpowiedzi API wynosi < 500ms dla 95% zapytań.
NFR2	Hasła użytkowników są haszowane algorytmem BCrypt (cost factor 12).
NFR3	Komunikacja z API odbywa się wyłącznie kanałem szyfrowanym (HTTPS).
NFR4	Aplikacja jest w pełni responsywna (RWD) i skaluje się do rozmiaru ekranu.
NFR5	Aplikacja jest łatwa do wdrożenia przy użyciu kontenerów Docker.

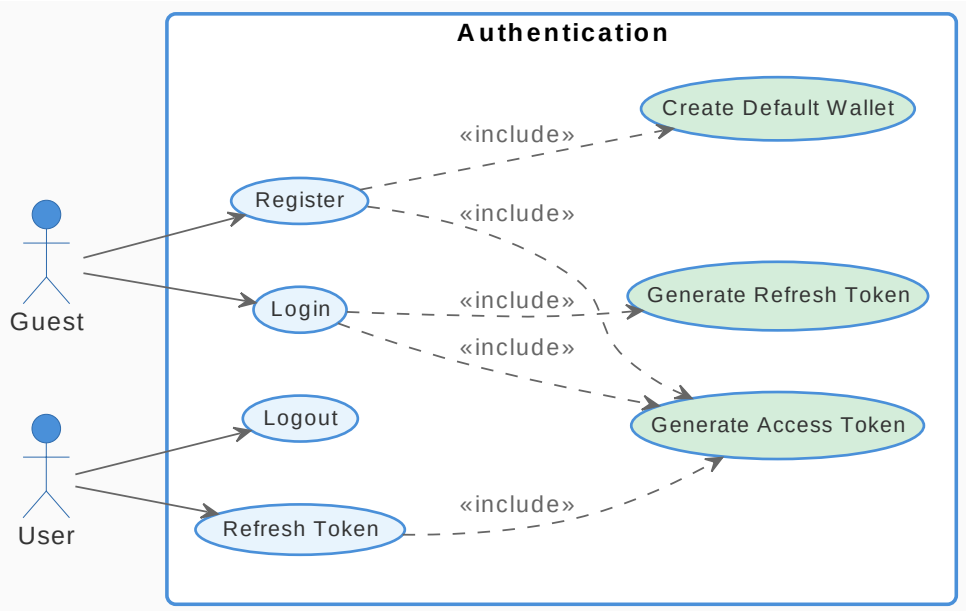
6 Diagramy Przypadków Użycia (Use Case)

Poniższe diagramy ilustrują dostępne funkcjonalności systemu oraz interakcje aktorów z poszczególnymi modułami aplikacji.



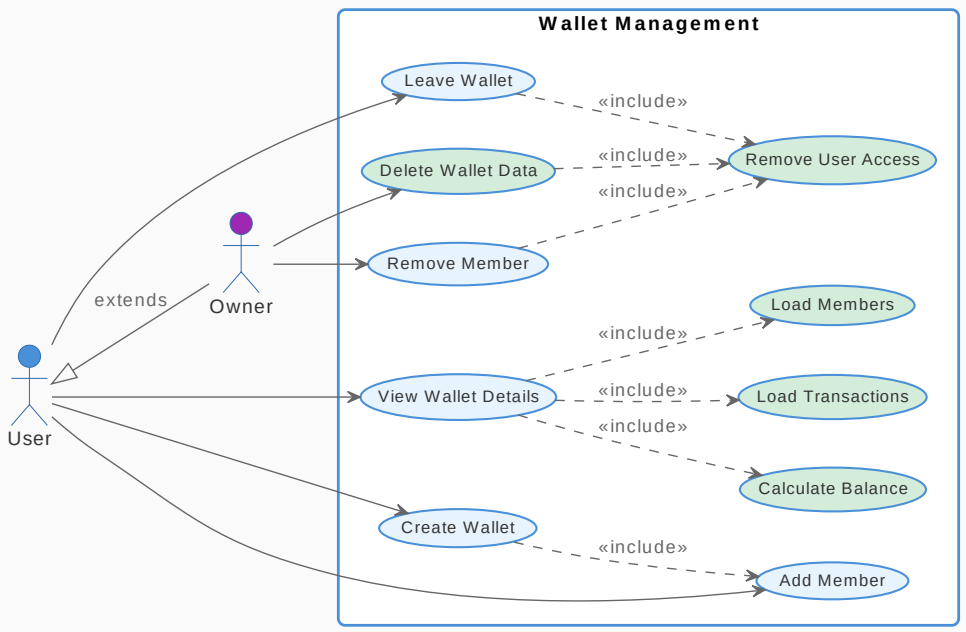
Rysunek 1: Ogólny diagram przypadków użycia

6.1 Autentykacja i Bezpieczeństwo



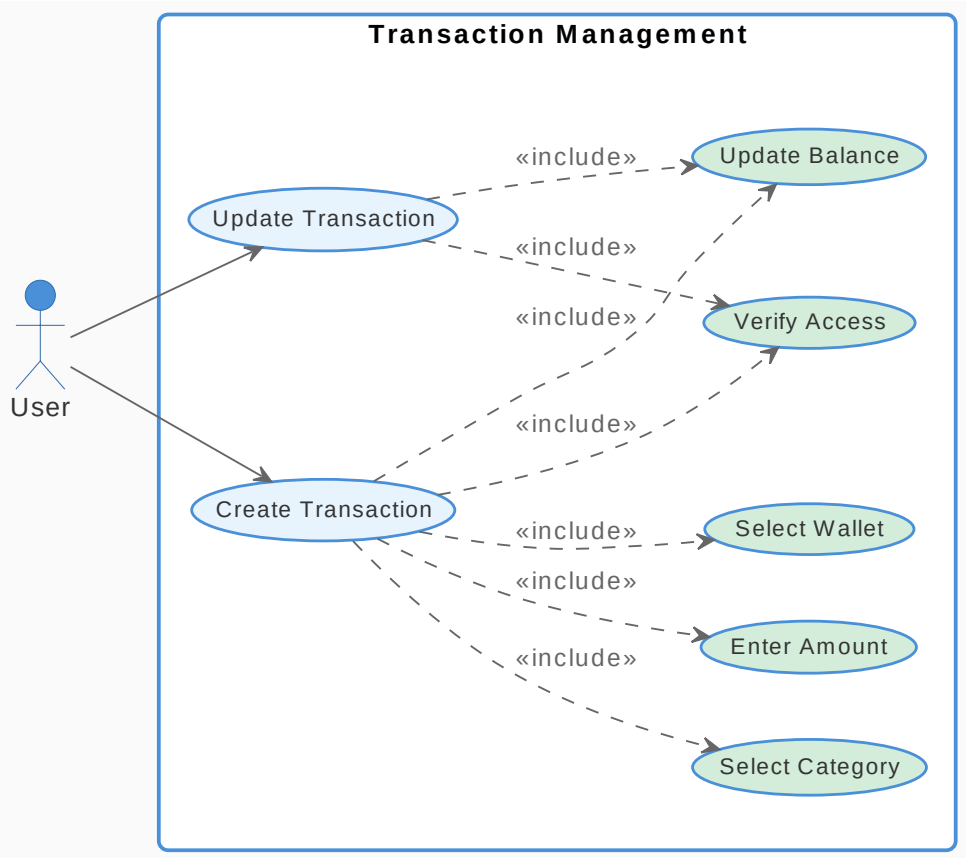
Rysunek 2: Moduł autentykacji

6.2 Zarządzanie Portfelami



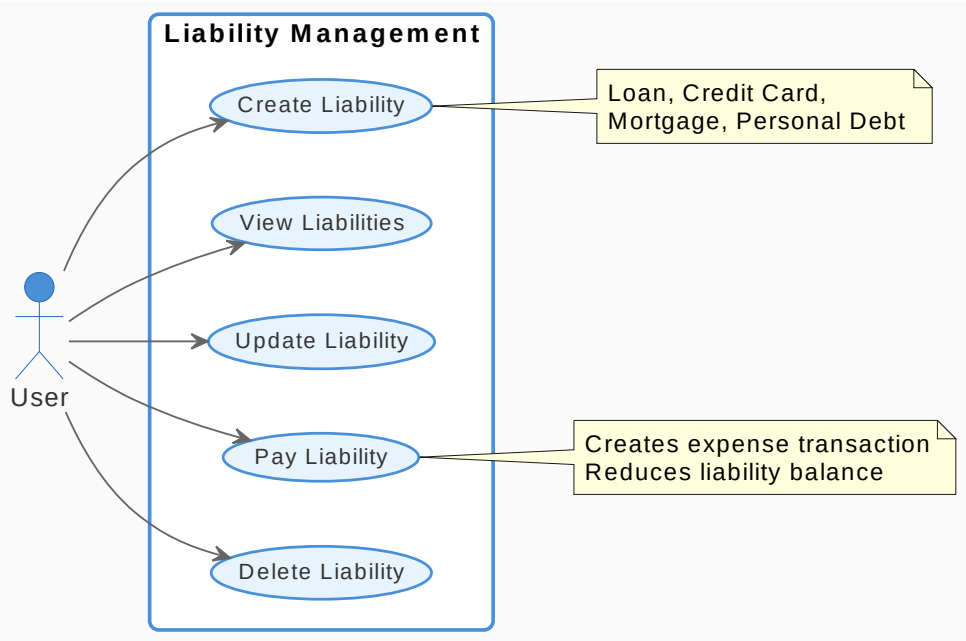
Rysunek 3: Operacje na portfelach

6.3 Zarządzanie Transakcjami



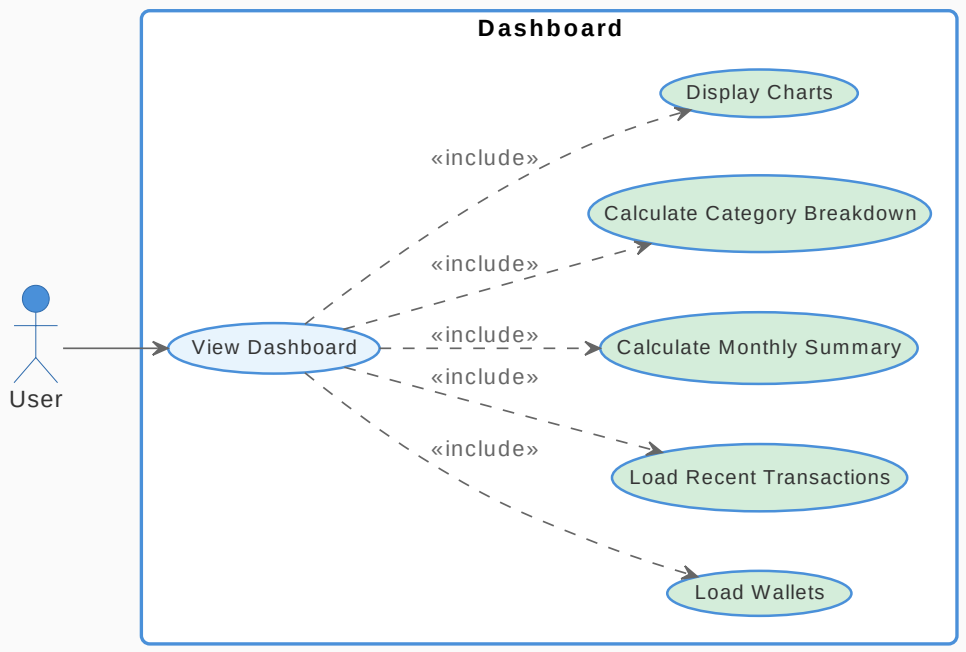
Rysunek 4: Operacje na transakcjach

6.4 Zobowiązania (Liabilities)



Rysunek 5: Obsługa długów i pożyczek

6.5 Dashboard

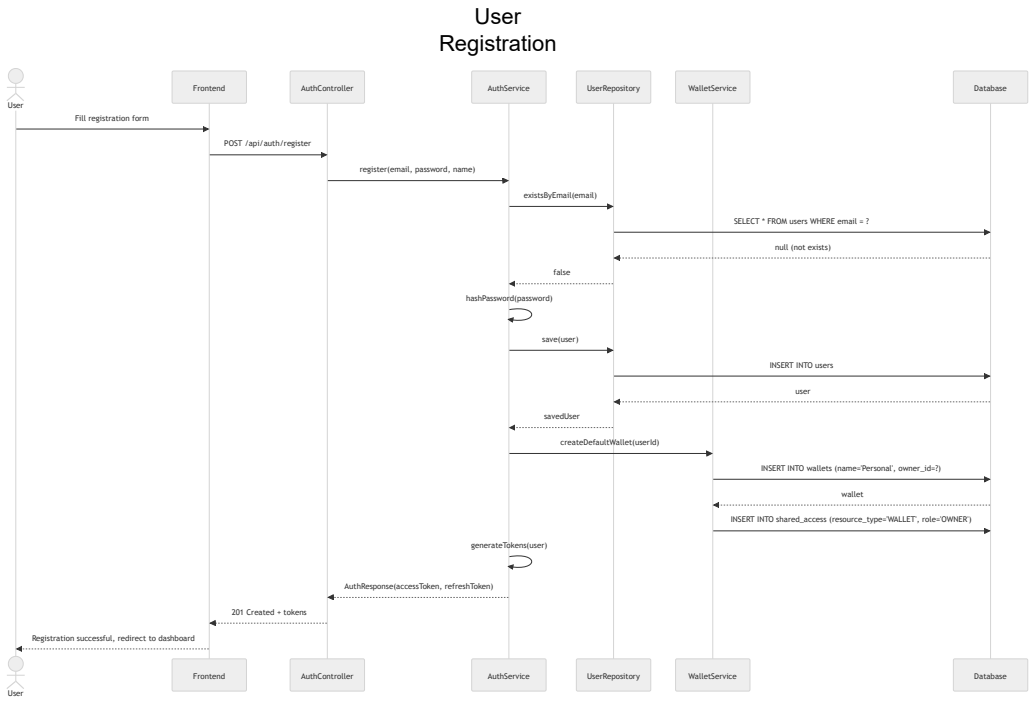


Rysunek 6: Logika wyświetlania dashboardu

7 Diagramy Sekwencji (Logic Flow)

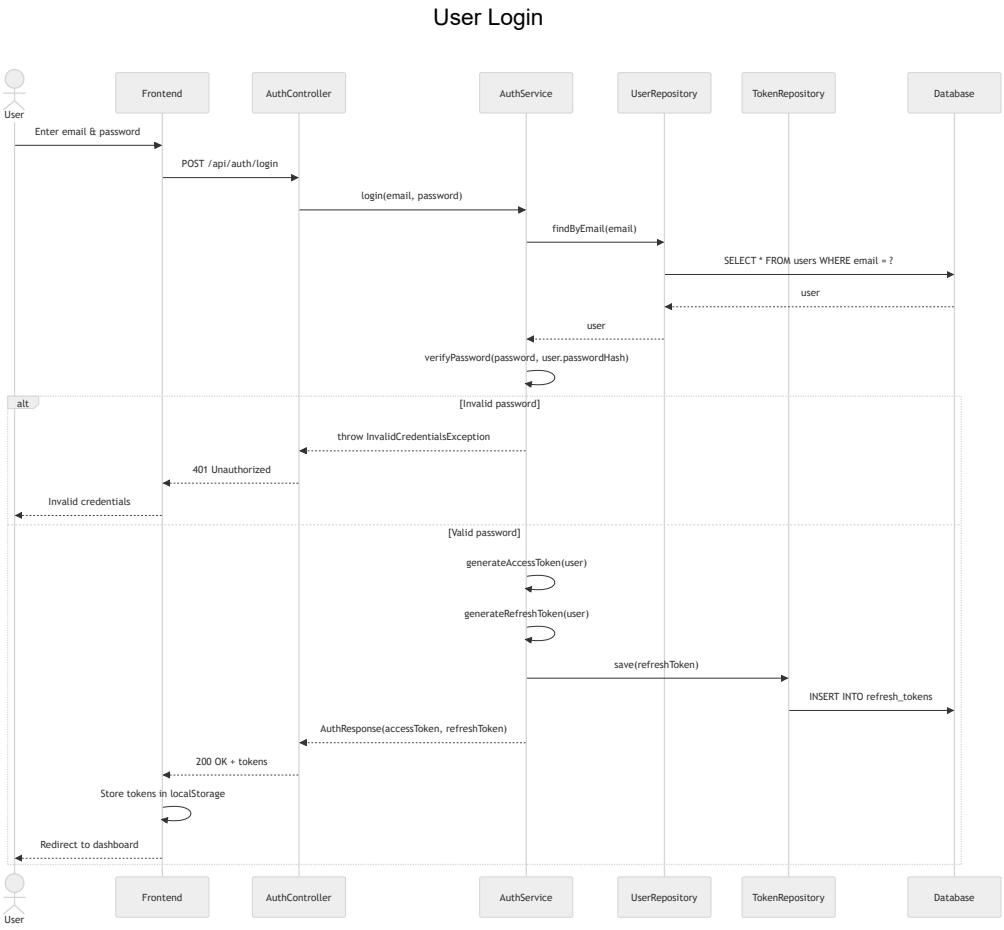
Poniższe diagramy prezentują przepływ sterowania dla kluczowych procesów biznesowych.

7.1 Rejestracja Użytkownika



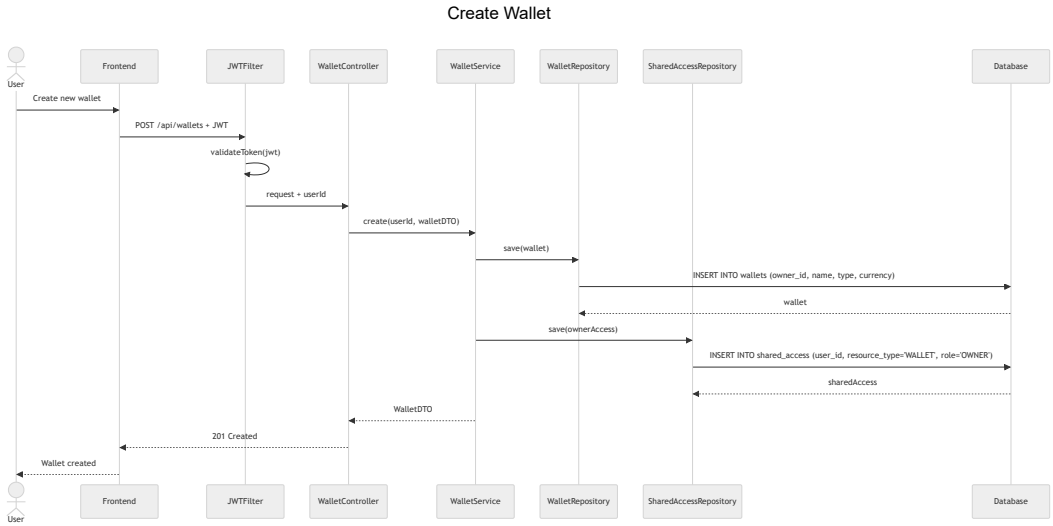
Rysunek 7: Przepływ procesu rejestracji

7.2 Logowanie



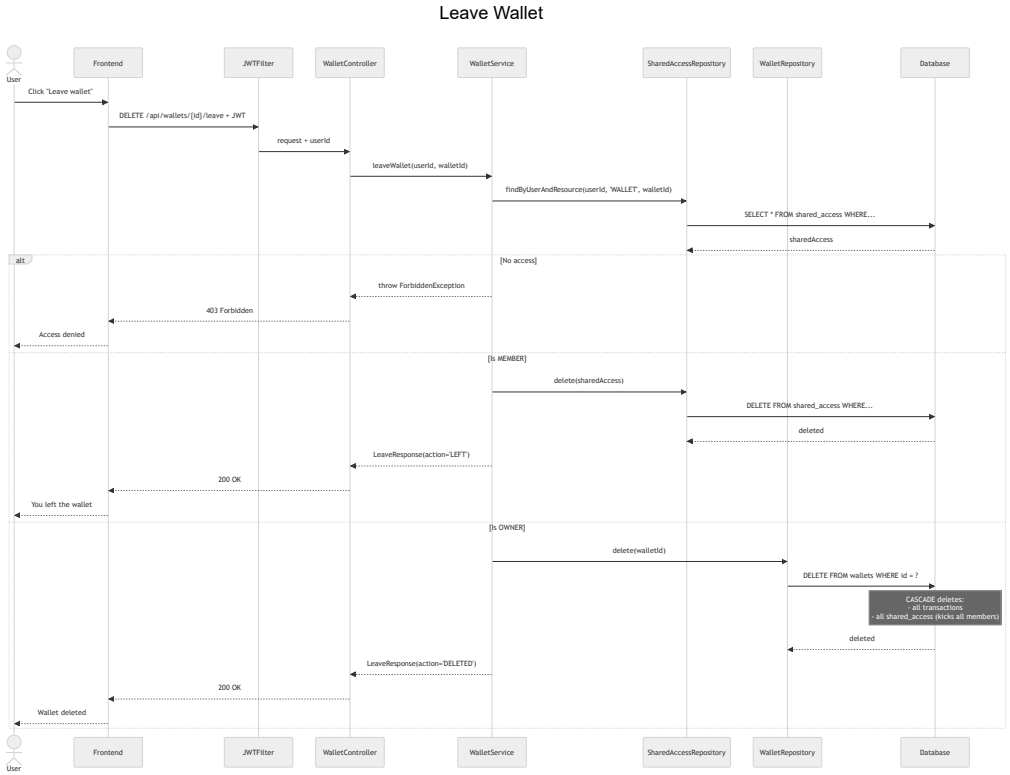
Rysunek 8: Przepływ procesu logowania (Sesja)

7.3 Tworzenie Nowego Portfela



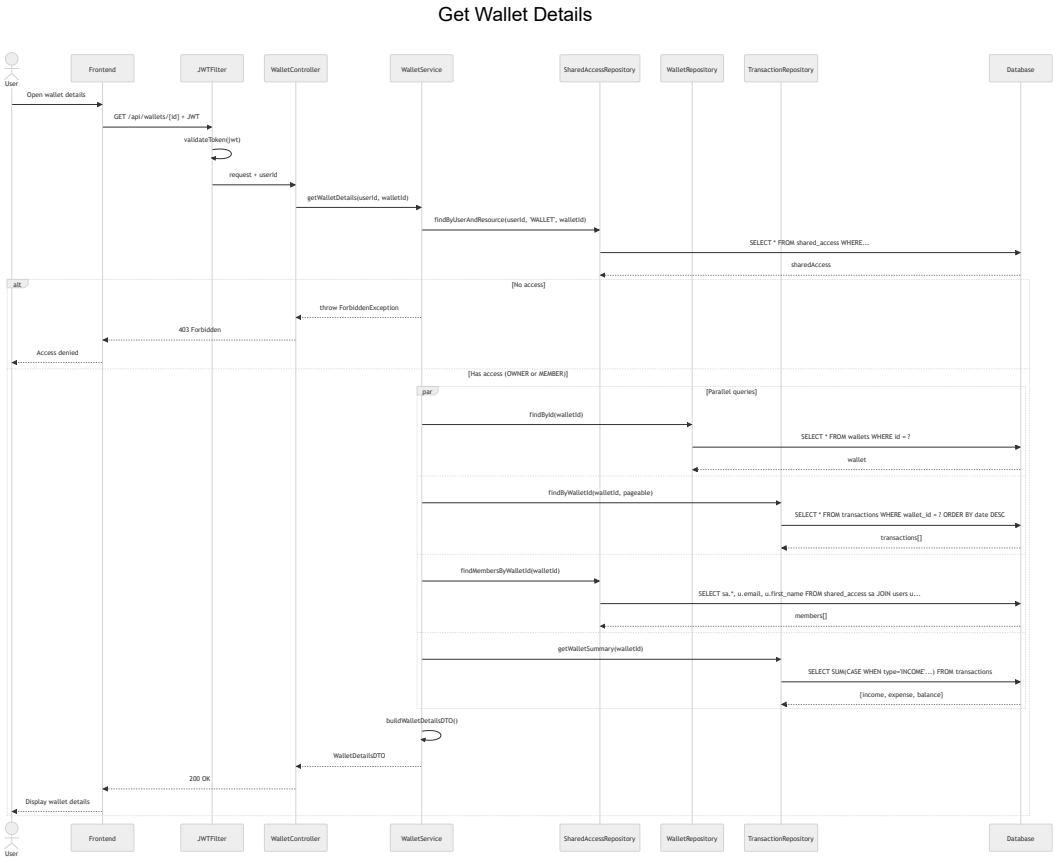
Rysunek 9: Logika tworzenia portfela

7.4 Opuszczanie Portfela



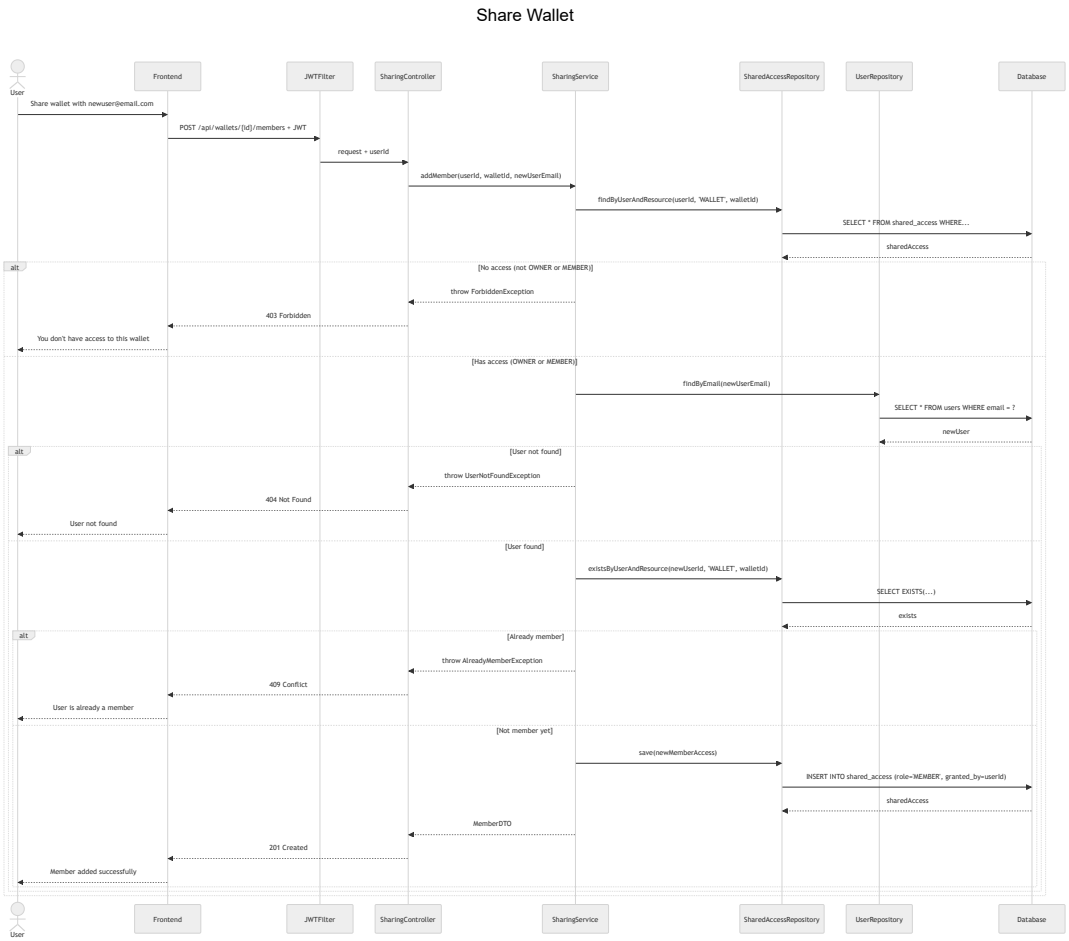
Rysunek 10: Logika opuszczania portfela przez użytkownika

7.5 Pobieranie Szczegółów Portfela



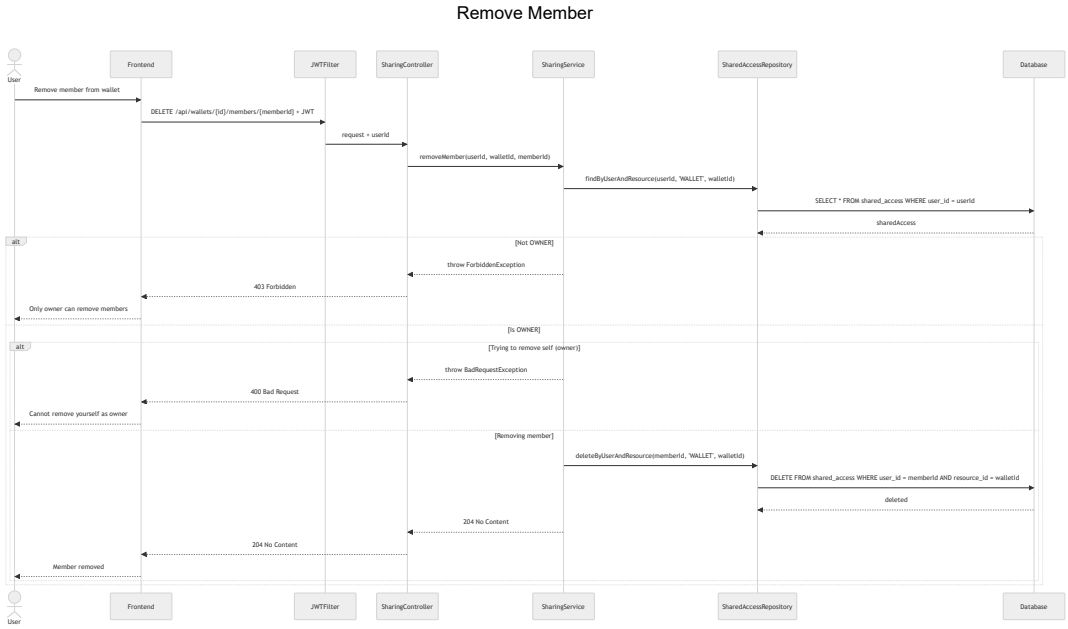
Rysunek 11: Proces pobierania danych o konkretnym portfelu

7.6 Dodawanie Członka do Portfela



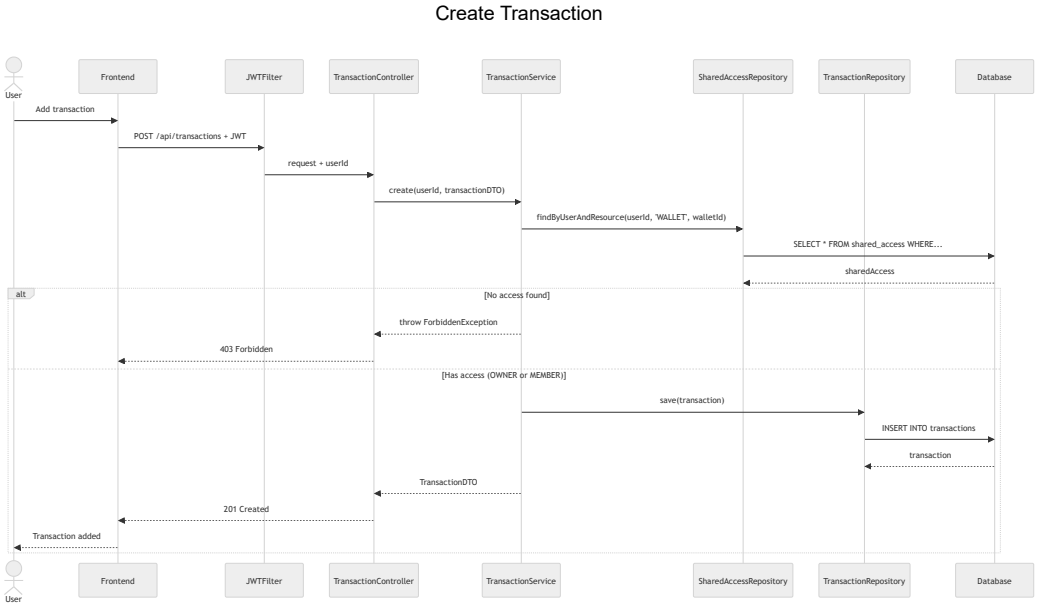
Rysunek 12: Logika zapraszania użytkownika do portfela

7.7 Usuwanie Członka z Portfela



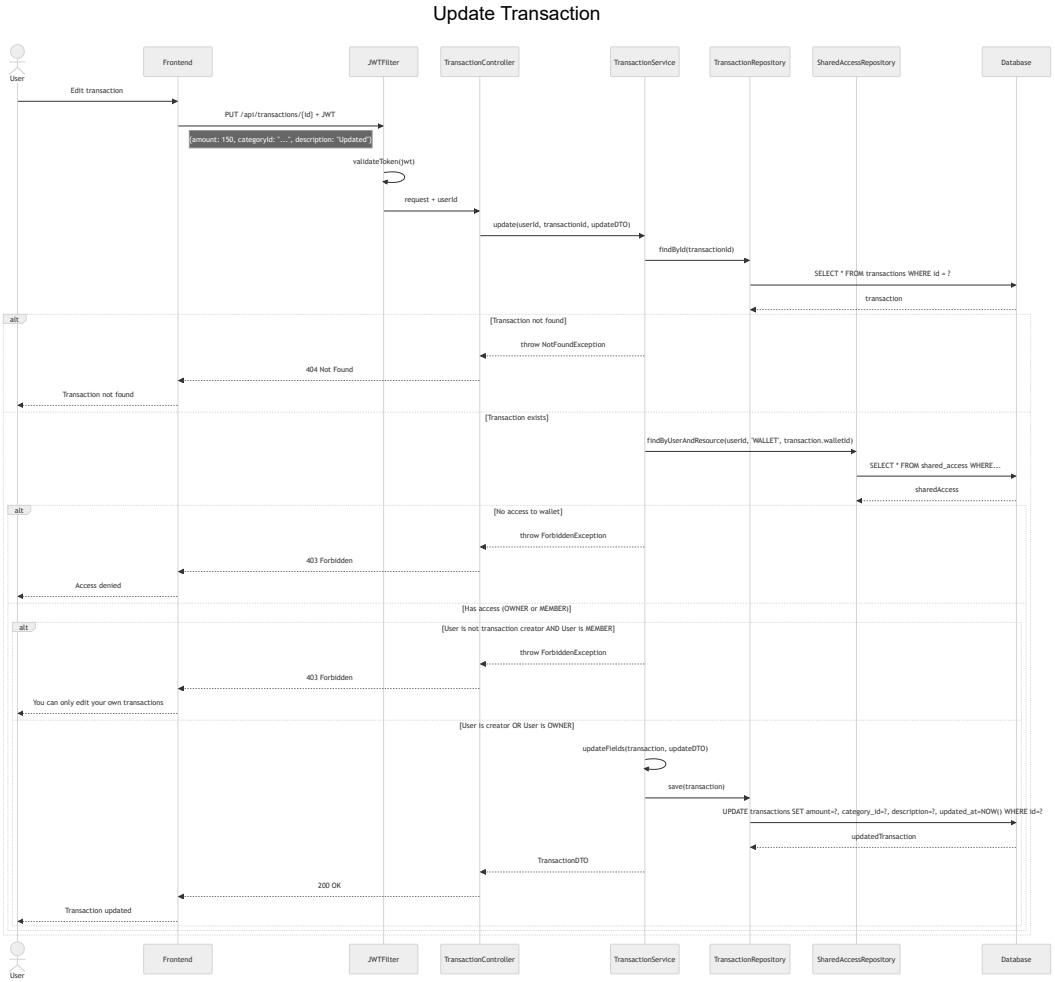
Rysunek 13: Logika usuwania członka przez właściciela

7.8 Tworzenie Transakcji



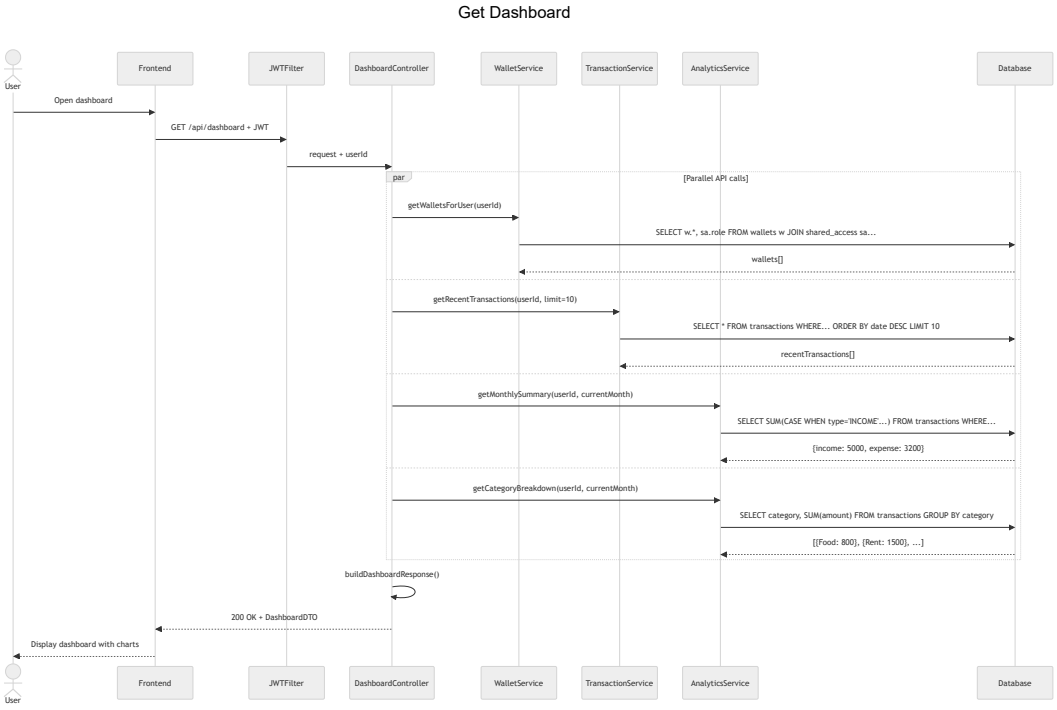
Rysunek 14: Zapisywanie nowej transakcji w bazie danych

7.9 Edycja Transakcji



Rysunek 15: Proces aktualizacji istniejącej transakcji

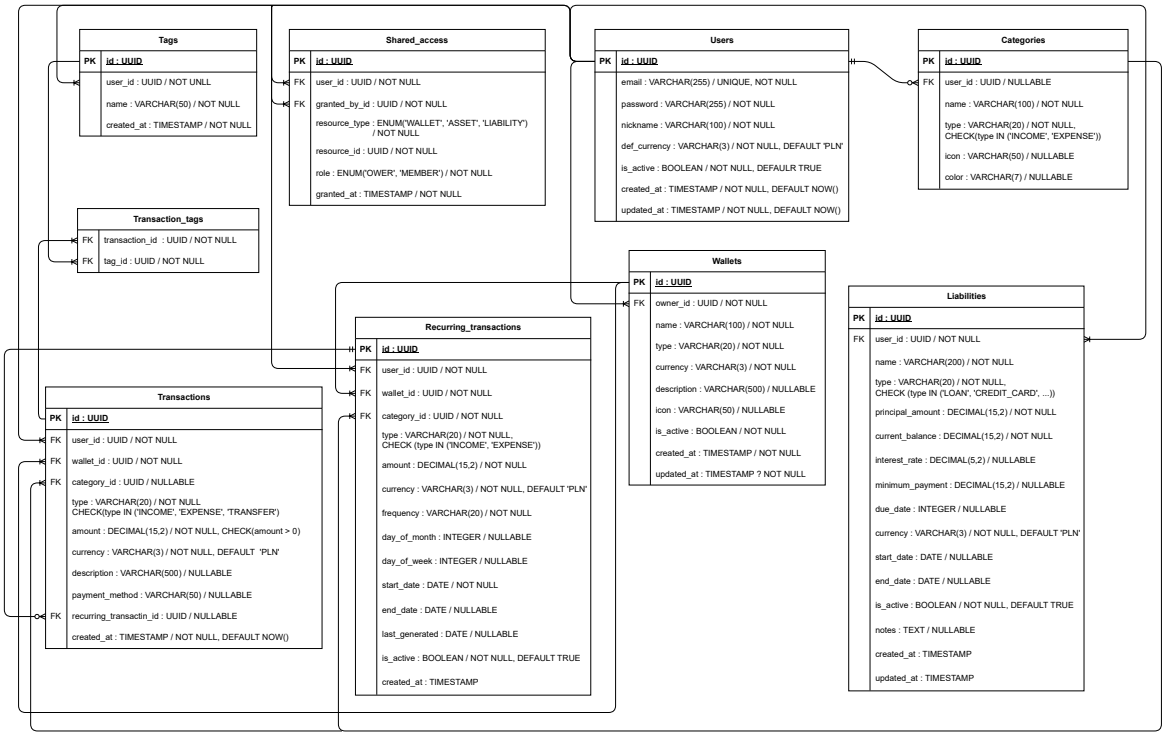
7.10 Pobieranie Danych Dashboardu



Rysunek 16: Agregacja danych finansowych na dashboard

8 Model Danych

Poniższy diagram przedstawia strukturę relacyjnej bazy danych PostgreSQL zaprojektowanej dla potrzeb systemu.



Rysunek 17: Diagram Entity-Relationship (ERD)

Opis kluczowych encji:

- **users**: Przechowuje dane uwierzytelniające oraz profilowe.
- **wallets**: Centralna encja grupująca finanse; definiuje walutę i właściciela.
- **transactions**: Rejestr operacji finansowych powiązany z portfelem i kategorią.
- **categories**: Słownik kategorii (systemowych oraz zdefiniowanych przez użytkownika).
- **liabilities**: Tabela śledząca stan zadłużenia i harmonogram spłat.

9 Plan Wdrożenia (Roadmap)

1. **Faza 1: Fundamenty**
    - Konfiguracja repozytorium i Docker Compose.
    - Konfiguracja szkieletu aplikacji (Spring Boot + React).
    - Projekt bazy danych PostgreSQL i migracje (Flyway).
2. **Faza 2: Core Logic**
    - Logika biznesowa portfeli (Backend).
    - CRUD transakcji.
    - Podstawowy Frontend (React Web).
3. **Faza 3: Zaawansowane Funkcje**
    - Współdzielenie portfeli (logika uprawnień).
    - Moduł zobowiązań.
    - Rozbudowa Dashboardu i wykresów.
4. **Faza 4: Stabilizacja**
    - Konteneryzacja finalna (Docker).
    - Poprawki błędów (Bugfixing) i deployment.