Wydział Elektroniki
i Technik Informacyjnych

POLITECHNIKA WARSZAWSKA

APRO2 - Projekt

Rozliczenie wspólnych wydatków

Etap dokumentacja

27 lutego 2025

Artur Brogowicz, Daria Shevchenko, Jan Kozaczuk, Michał Bogusz

Spis treści

1 Temat projektu

Często zdarza się, że będąc w pracy, na wyjeździe, w knajpie czy spotkaniu towarzyskim jedna osoba płaci za wszystkich. Prowadzi to do narastających długów i problemów – kto komu musi oddać pieniądze. Zaproponuj aplikację, która ułatwi zapamiętanie wydanych kwot i ułatwi rozliczenia między ludźmi zarejestrowanymi w aplikacji.

Przykład: Jeżeli osoba A zapłaciła za osobę B 30 zł, następnie B zapłaciła 30 zł za osobę C to w celu uproszenia przepływu pieniędzy aplikacja powinna wskazać, że to osoba C musi oddać 30 zł osobie A.

2 Cele projektu

Celem tego projektu jest stworzenie aplikacji do zarządzania wspólnymi wydatkami oraz ułatwienie rozliczeń między użytkownikami. Projekt ten ma na celu ułatwienie grupie osób zarządzanie finansami poprzez systematyzację i uproszczenie procesu dzielenia się wydatkami oraz uregulowania długów między uczestnikami.
Aplikacja będzie umożliwiała użytkownikom dodawanie, przeglądanie i zarządzanie transakcjami między różnymi osobami.

3 Analiza obiektowa

Analiza obiektowa to proces projektowania systemu informatycznego, który skupia się na identyfikacji, modelowaniu i specyfikowaniu obiektów oraz ich interakcji w systemie. Głównym celem analizy obiektowej jest zrozumienie struktury i zachowania systemu, aby umożliwicz jego skuteczną implemenntację.

3.1 Serwer

Służy jako baza danych oraz komunikacja (zarządzanie użytkownikami, transakcjami, żądaniami transakcji) Zawiera pola takie jak:

- adres serwera (serviceAddress)
- obiekt przechowujący informacje o:
  - użytkownikach (users)
  - transakcjach (transactions)
  - żądania transakcji (requests)

Zawiera metody umożliwiające komunikację takie jak add, get, send różnych parametrów.

3.2 Client

Komunikuje się z serwerem, stanowi interfejs użytkownika. Zawiera jedynie pole User. Wykonuje operacje takie jak:

- rejestracja i logowanie użytkownika (loginUser, registerUser)
- pokazanie wszystkich użytkowników (checkUsers)
- wysyłanie prośby o utworzenie transakcji (sendRequest)
- sprawdzenie dotyczących go próśb oraz transakcji (checkRequests)
- zarządzanie prośbą o utworzenie transakcji (manageRequest)
- finalizacja istniejącej transakcji, czyli jej usunięcie z listy wszystkich transakcji (realizeTransaction)

3.3 User

Reprezentuje pojedynczego użytkownika. Zawiera dane użytkownika, takie jak:

- login użytkownika (userLogin)
- imię i nazwisko użytkownika (name, lastName)
- saldo użytkownika (balance)
- userID (String)

3.4 Users

Kolekcja obiektów User. Reprezentuje zbiorcze informacje o wszystkich użytkownikach. Zawiera pola takie jak:

- dodanie użytkownika (*addUser*)
- pobranie użytkownika na podstawie identyfikatora (*getUser*)
- pobranie użytkownika na podstawie loginu (*getUserByLogin*)
- pobranie listy wszystkich użytkowników (*getUsers*)

3.5 Transaction

Klasa reprezentuje pojedynczą transakcję (innymi słowy: dług). Zawiera następujące pola:

- transactionID (String)
- lenderID (String)
- borrowerID (String)
- sum (int)

Pole *sum* reprezentuje liczbę np. groszy lub centów.

3.6 Transactions

Kolekcja obiektów Transaction. Reprezentuje listę wszystkich aktywnych długów. Zawiera następujące metody:

- dodaje transakcję (*addTransaction*)
- pobiera transakcję na podstawie identyfikatora (*getTransaction*)
- pobiera listę wszystkich transakcji (*getTransactions*)

3.7 TransactionRequest

Klasa reprezentuje prośbę klienta o utworzenie transakcji. Każde żądanie posiada:

- unikalny identyfikator (*requestId*)
- informację o kierunku transakcji (*direction*)
- identyfikator autora żądania (*authorId*)
- identyfikator odbiorcy żądania (*targetId*)
- przekazywaną kwotę (*sum*)
- status akceptacji (*isAccepted*) - wskazuje, czy prośba może przekształcić się w prawdziwą transakcję, na podstawie decyzji użytkownika, którego ta prośba dotyczy.

3.8 TransactionRequests

Kolekcja obiektów TransactionRequest. Pełni rolę zbiorczą dla nierozpatrzonych jeszcze żądań transakcji w systemie. Posiada następujące metody:

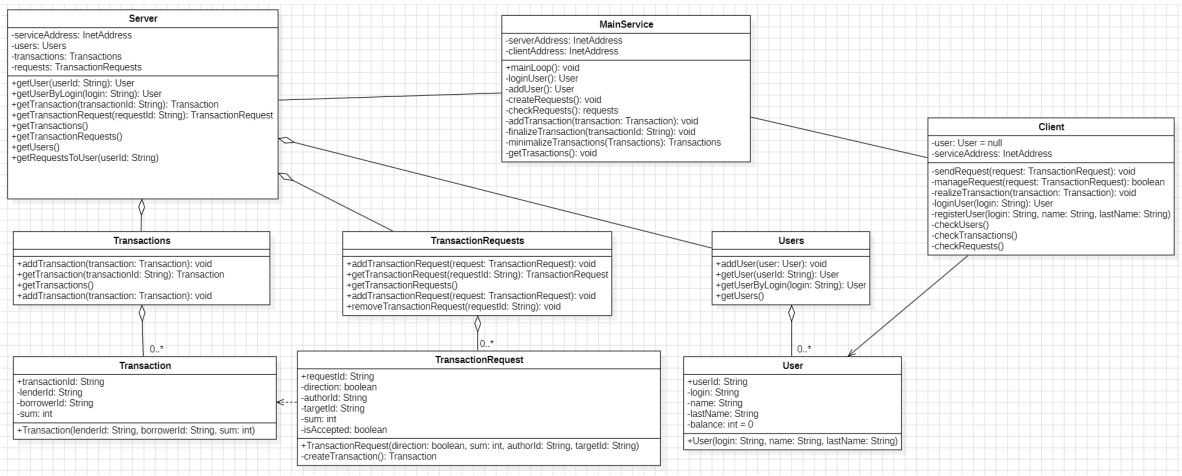
- dodaje żądanie transakcji (*addTransactionRequest*)
- pobiera żądanie transakcji (*getTransactionRequest*)
- pobiera listę wszystkich żądań transakcji (*getTransactionRequests*)

3.9 Services

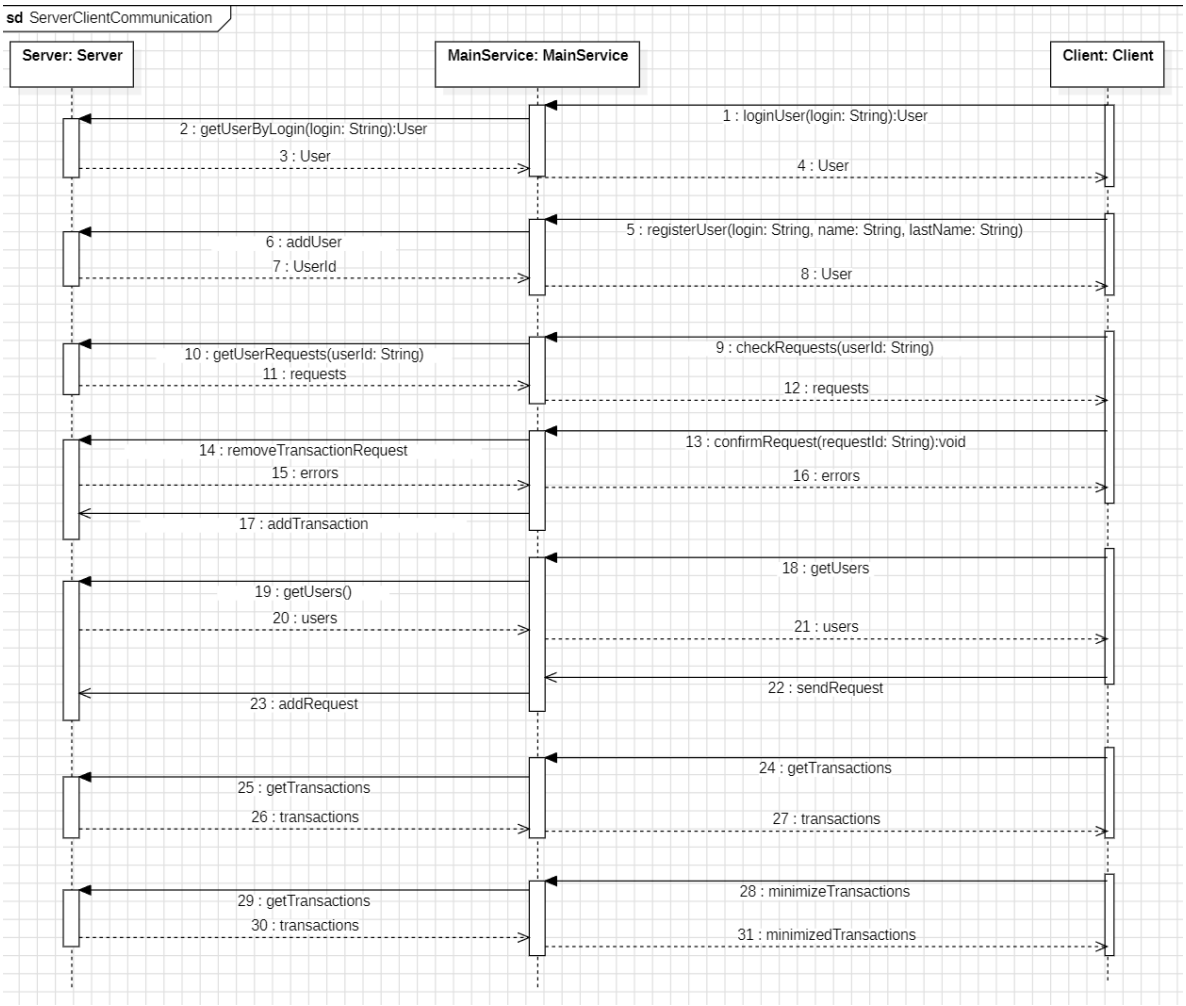
Klasa komunikująca się z klientem za pośrednictwem serwera, zawiera niezbędne usługi do realizacji operacji między użytkownikami. Zawiera następujące metody:

- zwracanie listy próśb dla wskazanego usera (getRequestList)
- potwierdzanie próśb (confirmRequest)
- logowanie użytkownika (loginUser)
- rejestracja użytkownika (registerUser)

4 Projekt diagramu klas (UML)



Rysunek 1: Diagram klas UML



Rysunek 2: Diagram sekwencyjny komunikacji klienta z serwerem

Literatura

[1] Verhoeff, T. (2004). Settling multiple debts efficiently : an invitation to computing science. Informatics in Education, 3(1), 105-126.

[2] Aplikacja Tricount