**Sprawozdanie z Projektu Uproszczonego Systemu Bankowego Klient-Serwer**

Data: 28.05.2025  
Autorzy: Illia Bazyliv, Artur Hryban

**Część 1: Wstęp i Architektura**

**1. Wstęp**

Projekt przedstawia uproszczony system bankowy w architekturze klient-serwer, umożliwiający klientom logowanie, sprawdzanie salda, wpłaty, wypłaty i przelewy, a bankierom zarządzanie danymi klientów i kontami poprzez aplikacje konsolowe. Celem było stworzenie funkcjonalnego rozwiązania, które pokazuje zastosowanie programowania sieciowego i bazodanowego w Javie, wybranej ze względu na niezawodność, wsparcie dla wielowątkowości i wieloplatformowość dzięki JVM. Projekt ma też wartość edukacyjną, ilustrując integrację komunikacji sieciowej, wielowątkowości i zarządzania bazą danych.

**2. Architektura**

System został zaprojektowany w oparciu o klasyczny model klient-serwer, który jest powszechnie stosowany w aplikacjach rozproszonych. W tym modelu serwer pełni rolę centralnego komponentu, odpowiedzialnego za odbieranie i obsługę żądań pochodzących od wielu klientów jednocześnie. Aby zapewnić efektywną obsługę wielu połączeń, zaimplementowano wielowątkowość – każde połączenie klienta jest obsługiwane w osobnym wątku. Taki mechanizm pozwala uniknąć blokowania serwera i zapewnia płynne działanie nawet przy większej liczbie użytkowników.

Komunikacja między klientami a serwerem opiera się na prostym, tekstowym protokole, w którym wiadomości są formatowane za pomocą średników i przesyłane przez gniazda TCP/IP. Protokół ten, mimo swojej prostoty, jest skuteczny i umożliwia łatwe przesyłanie komend oraz danych w ustandaryzowany sposób. Przykładowo, żądanie logowania może mieć postać: LOGIN;user\_id;password, na co serwer odpowiada: OK;LOGIN\_SUCCESSFUL w przypadku powodzenia lub ERROR;INVALID\_CREDENTIALS w razie niepowodzenia. Użycie TCP/IP gwarantuje niezawodność i uporządkowaną transmisję danych, co jest kluczowe dla systemu bankowego.

Kod źródłowy projektu został podzielony na trzy główne pakiety, co zwiększa jego czytelność i ułatwia zarządzanie:

* common: Zawiera klasy i modele danych wspólne dla klienta i serwera, takie jak definicje struktur danych używanych w komunikacji.
* client: Obejmuje logikę aplikacji klienckich, w tym interfejs użytkownika i obsługę żądań wysyłanych do serwera.
* server: Zawiera implementację serwera, w tym mechanizmy obsługi połączeń, wielowątkowość oraz przetwarzanie żądań.

Taka modularna struktura nie tylko poprawia przejrzystość kodu, ale również ułatwia jego dalsze utrzymanie i rozbudowę, co jest istotne w projektach o większej skali. Dodatkowo, zastosowanie wielowątkowości w połączeniu z protokołem TCP/IP pozwala na elastyczne i niezawodne działanie systemu.

**Część 2: Działanie Systemu, Użyte Technologie i Podsumowanie**

**3. Działanie Systemu**

Serwer w systemie bankowym jest zaprojektowany tak, aby odbierać żądania od klientów i przetwarzać je w sposób asynchroniczny, co realizowane jest poprzez obsługę każdego połączenia w osobnym wątku. Dzięki temu system może jednocześnie obsługiwać wielu użytkowników bez utraty wydajności. Komunikacja z bazą danych MySQL odbywa się za pomocą JDBC (Java Database Connectivity), co pozwala na przechowywanie, aktualizację i pobieranie informacji o kontach oraz użytkownikach w sposób zorganizowany i bezpieczny.

Baza danych została zaprojektowana z użyciem dwóch głównych tabel:

* clients: Przechowuje dane klientów, takie jak identyfikator użytkownika i dane logowania.
* accounts: Zawiera informacje o kontach bankowych, powiązanych z klientami relacją jeden-do-wielu, co oznacza, że jeden klient może posiadać wiele kont.

Aplikacje klienckie występują w dwóch wersjach, dostosowanych do różnych grup użytkowników:

* ClientApp: Dedykowana klientom indywidualnym, oferuje konsolowy interfejs z menu umożliwiającym logowanie, sprawdzanie salda, dokonywanie wpłat, wypłat i przelewów.
* BankerApp: Przeznaczona dla bankierów, pozwala na administracyjne zarządzanie danymi klientów, w tym dodawanie nowych użytkowników i edytowanie ich kont.

Przykładowy przepływ operacji przelewu wygląda następująco:

1. Użytkownik wybiera opcję przelewu w menu aplikacji ClientApp.
2. Aplikacja prosi o podanie numeru konta źródłowego, docelowego oraz kwoty przelewu.
3. Żądanie w formacie TRANSFER;from\_account;to\_account;amount jest wysyłane do serwera.
4. Serwer weryfikuje dostępność środków na koncie źródłowym, wykonuje transakcję w bazie danych i aktualizuje salda.
5. Odpowiedź, np. OK;TRANSFER\_SUCCESSFUL lub ERROR;INSUFFICIENT\_FUNDS, jest zwracana do klienta i wyświetlana w interfejsie.

Taki proces pokazuje, jak system integruje komunikację sieciową z operacjami bazodanowymi, zapewniając użytkownikom intuicyjne i niezawodne narzędzie.

**4. Użyte Technologie**

Do realizacji projektu wykorzystano następujące technologie i narzędzia:

* Język programowania: Java 17
* Narzędzia i biblioteki:
  + Maven;
  + MySQL;
  + TCP/IP;
  + ExecutorService: Mechanizm zarządzania pulą wątków, który upraszcza implementację wielowątkowości na serwerze, zwiększając jego skalowalność;
  + java.util.logging: Biblioteka do rejestrowania zdarzeń, która ułatwia monitorowanie działania serwera i debugowanie w razie problemów.

**5. Podsumowanie**

Projekt skutecznie realizuje podstawowe funkcje systemu bankowego, takie jak zarządzanie kontami, przelewy i administracja użytkownikami, łącząc prostotę z użytecznością w Javie. Jest stabilny, z potencjałem do rozbudowy o funkcje jak uwierzytelnianie dwuskładnikowe czy cache’owanie dla lepszej wydajności. Ma wartość edukacyjną, przystępnie pokazując wielowątkowość, komunikację sieciową i zarządzanie bazami danych, stanowiąc bazę do dalszej nauki lub tworzenia złożonych systemów.