

# Mas Dokumentacja

Dariusz Dajcz S21522

## **1.Dziedzina problemowa:**

Aplikacja "AppMobileBank" pozwala bankom kontrolować transakcje, jak i klientom monitorować wydatki.

## **2.Cel:**

Dzięki aplikacji klient będzie miał możliwość wglądu w informacje w związku ze swoim kontem bankowym. Będzie mógł również wykonywać transakcje, bez konieczności robienia tego stacjonarnie w oddziale banku. Wiele funkcjonalności również usprawni pracę banku.

Zakres odpowiedzialności systemu: System powinien przechowywać wszelkie informacje takie jak: stan konta klienta, historia transakcji.

## **4.Użytkownicy:**

Pracownik banku, Klient zarejestrowany, Klient niezarejestrowany, Podsystem czasu

## **5.Wymagania użytkownika:**

1.W systemie przechowywane są informacje o klientach. Każdy klient posiada numer klienta, adres, numer kontaktowy, oraz datę urodzenia. Klientem zarejestrowanym może być osoba (posiadająca konto, dowód osobisty, e-mail oraz informacje czy posiada zdolność kredytową). Klient niezarejestrowany musi posiadać nr Pesel w celu założenia konta

2.Klient może podpisać wiele umów (potrzebny do tego jest podpis klienta oraz zatwierdzenie banku) z Bankiem. Bank posiada nazwę, dokładny adres i możliwość udzielania kredytów (wybrane mogą). Po podpisaniu umowy bank dodaje konto przypisane klientowi. Każde konto zawiera informacje na temat numeru konta, daty założenia, czasu użytkowania oraz możliwość włączenia opcji blik w ustawieniach. Każde konto może być zarazem osobiste (posiada dodatkowo rodzaj pakietu, listę wydatków), studenckie (posiada dodatkowo informacje na temat karty studenckiej, dostępna lista zniżek dla studentów) i premium (posiada informacje na temat zniżek premium do 25%).

Każde konto posiada stan konta z informacjami na temat dostępnych środków. Dodatkowo pobierana jest opłata za konto 20zł miesięcznie (nie dotyczy kont studenckich) oraz bonus dla klientów przyznawany w zależności od rodzaju konta.

3. Z każdego konta można wykonywać transakcje. Każda transakcja posiada unikalny numer powiązany z kontem, kwotę na jaką jest wykonywana, datę wykonania, datę potwierdzenia oraz rodzaj transferu (krajowy lub zagraniczny). Transakcją może być wpłata, wypłata lub przelew. Za wpłatę lub wypłatę pobierana jest prowizja w wysokości 10zł, dodatkowo podawana jest informacja o miejscu gdzie została wykonana usługa. Na temat przelewu wiemy czy był to przelew ekspresowy oraz pobierana jest stała opłata w wysokości 5zł (opłata nie obejmuje konta premium). Transakcje także dzielimy ze względu na formę płatności m.in. gotówka lub karta kredytowa. Gotówka posiada walute (wyłącznie w walutach: (dolar, euro, funty i zł)) oraz ilość bankotów. Karta kredytowa posiada nr karty kredytowej, datę ważności, kod cvv.

---

Oczekuje się, że aplikacja będzie wspomagać użytkownika w realizacji zadań takich jak:

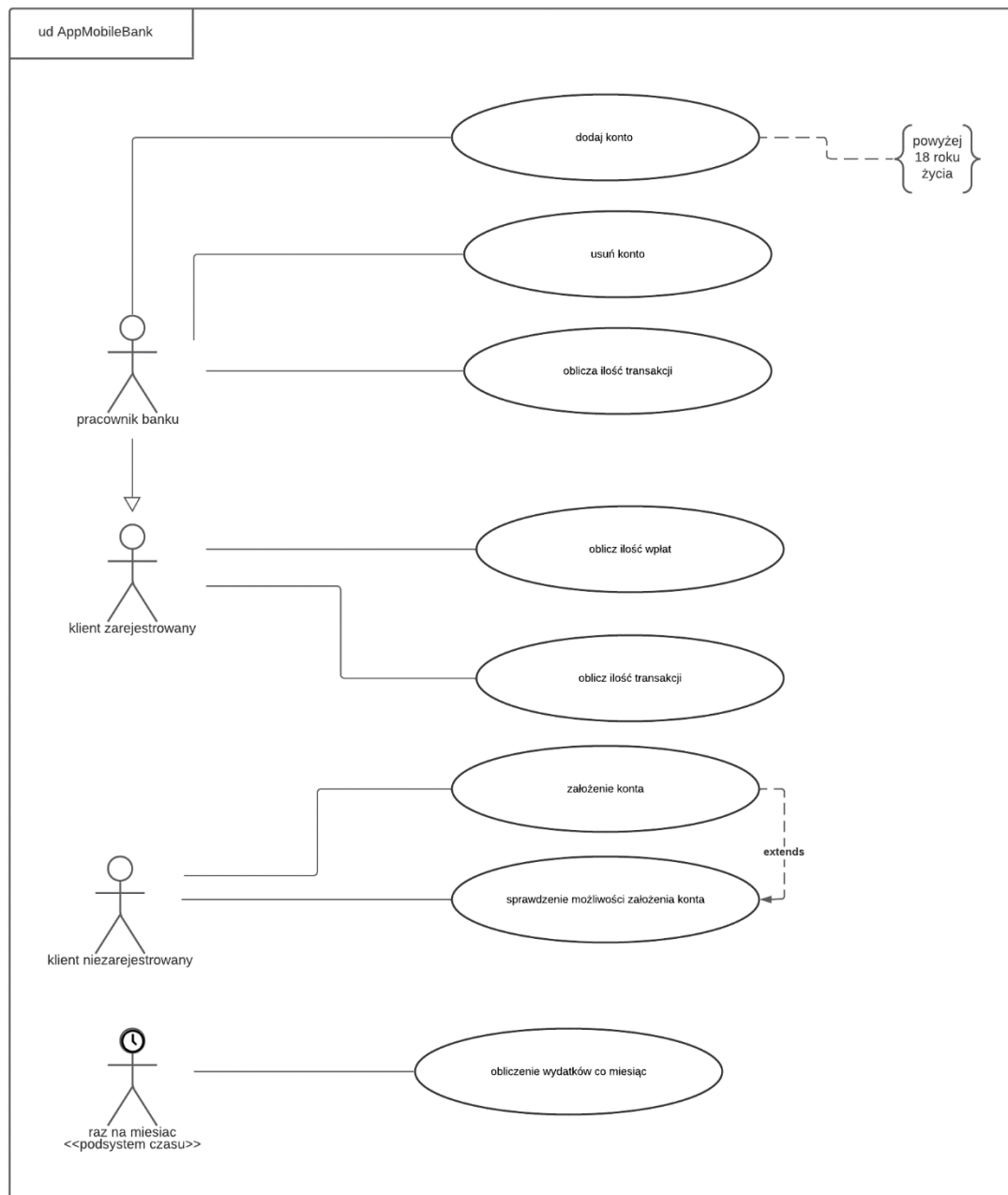
- \* dodanie konta z wcześniejszym sprawdzeniem zdolności kredytowej oraz wieku klienta
- \* usunięcie konta (dokonywane wyłącznie przez pracownika banku po wcześniejszej informacji od klienta)
- \* możliwość wyliczeń transakcyjnych przez pracownika banku
- \* możliwość wyliczeń ilości wpłat oraz ilości transakcji danego konta, również przez klienta
- \* sprawdzenie możliwości założenia nowego konta (dostępne dla wszystkich, również dla gości)
- \* rejestracja konta z możliwością sprawdzenia możliwości założenia konta, dostępne dla gości
- \* cykliczne wyliczenie wydatków miesięcznych

---

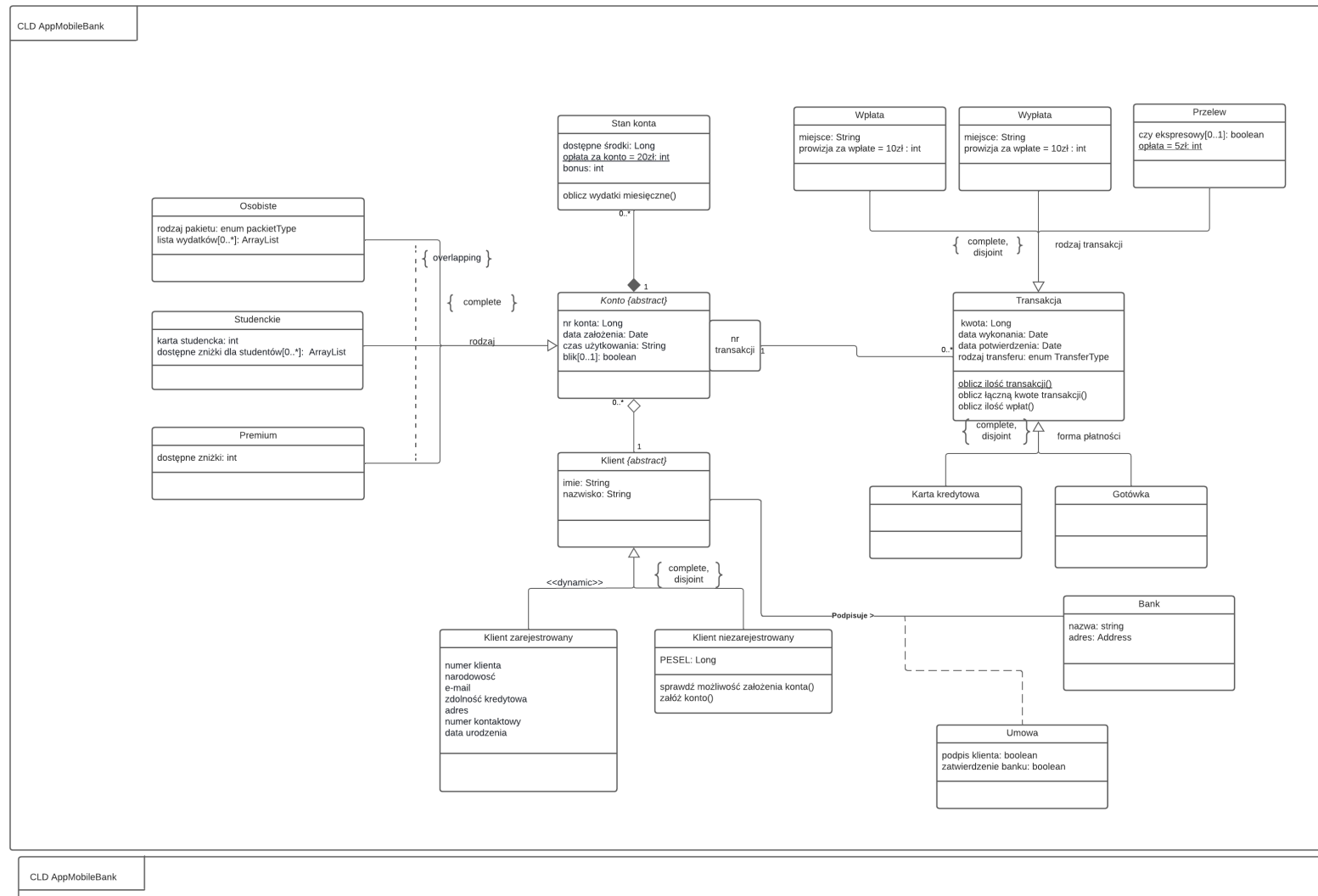
Ograniczenia

Możliwość założenia konta, działalność, zniżka dla kont premium, waluta w transakcjach gotówkowych, transfer, dodanie konta przez bank.

## 6. Diagram przypadków użycia:



## 7.Diagram klas analityczny:



```

classDiagram
    class StanKonta {
        dostepne_srodki: Long
        oplata_za_konto = 20zł: int
        bonus: int
    }
    class Konto {
        nr_konta: Long
        data_zalozenia: Date
        czas_uzytkowania: String
        blik[0..1]: boolean
        rodzaj_pakietu: enum
        rodzaj_konta: Rodzaj_konta
        //Osobiste
        lista_wydatkow[0..*]: ArrayList
        //Studenckie
        karta_studencka: int
        dostepne_znizki_dla_studentow[0..*]: ArrayList
        //Premium
        dostepne_znizki: int
    }
    class Klient {
        <<abstract>>
        imie: String
        nazwisko: String
    }
    class KlientZarejestrowany {
        numer_klienta: Long
        narodowosc: String
        e-mail: String
        zdolnosc_kredytowa: boolean
        adres: Address
        numer_kontaktowy: String
        data_urodzenia: Date
        usun_konto()
    }
    class KlientNiezarejestrowany {
        PESEL: Long
        sprawdź_możliwość_zalozenia_konta()
        załóż_konto()
    }
    class Wpłata {
        miejsce: String
        prowizja_za_wplate = 10zł: int
    }
    class Wyplata {
        miejsce: String
        prowizja_za_wplate = 10zł: int
    }
    class Przelew {
        czy_ekspresowy[0..1]: boolean
        oplata = 5zł: int
    }
    class Transakcja {
        kwota: Long
        data_wykonania: Date
        data_potwierdzenia: Date
        rodzaj_transferu: enum
        forma_platnosci: Enum
    }
    class Bank {
        nazwa: String
        adres: Address
    }
    class Umowa {
        podpis_klienta: boolean
        zatwierdzenie_banku: boolean
    }

    StanKonta "0..1" --> "1" Konto : kompozycja
    Konto "1" --> "0..1" Klient : asocjacja kwalifikowana
    Klient "1" --> "0..1" KlientZarejestrowany : dynamic
    Klient "1" --> "0..1" KlientNiezarejestrowany : dynamic
    KlientZarejestrowany "1" --> "0..1" KlientNiezarejestrowany : complete, disjoint
    Konto "1" --> "0..1" Wpłata : asocjacja kwalifikowana
    Konto "1" --> "0..1" Wyplata : asocjacja kwalifikowana
    Konto "1" --> "0..1" Przelew : asocjacja kwalifikowana
    Wpłata "0..1" --> "0..1" Transakcja : complete, disjoint
    Wyplata "0..1" --> "0..1" Transakcja : complete, disjoint
    Przelew "0..1" --> "0..1" Transakcja : complete, disjoint
    Klient "1" --> "0..1" Transakcja : asocjacja kwalifikowana
    KlientZarejestrowany "1" --> "0..1" Transakcja : asocjacja kwalifikowana
    KlientNiezarejestrowany "1" --> "0..1" Transakcja : asocjacja kwalifikowana
    Klient "1" --> "0..1" Bank : asocjacja kwalifikowana
    KlientZarejestrowany "1" --> "0..1" Bank : asocjacja kwalifikowana
    KlientNiezarejestrowany "1" --> "0..1" Bank : asocjacja kwalifikowana
    Klient "1" --> "0..1" Umowa : asocjacja kwalifikowana
    KlientZarejestrowany "1" --> "0..1" Umowa : asocjacja kwalifikowana
    KlientNiezarejestrowany "1" --> "0..1" Umowa : asocjacja kwalifikowana
  
```

The diagram illustrates the structure of the CLD AppMobileBank system. It features several classes and their relationships:

- Stan konta**: A class representing account status with attributes `dostępne środki: Long`, `opłata za konto = 20zł: int`, and `bonus: int`. It is associated with **Konto** via a composition relationship.
- Konto**: A central class with attributes like `nr konta: Long`, `data założenia: Date`, `czas użytkowania: String`, `blik[0..1]: boolean`, `rodzaj pakietu: enum`, and `rodzaj konta: Rodzaj konta`. It also contains role-specific attributes for `Osobiste`, `Studenckie`, and `Premium`. It is associated with **Klient** and **Transakcja**.
- Klient (abstract)**: An abstract class with attributes `imie: String` and `nazwisko: String`. It has two subclasses: **Klient zarejestrowany** and **Klient niezarejestrowany**, which are dynamically associated with the abstract class.
- Klient zarejestrowany**: Contains attributes like `numer klienta: Long`, `narodowość: String`, `e-mail: String`, `zdolność kredytowa: boolean`, `adres: Address`, `numer kontaktowy: String`, and `data urodzenia: Date`. It has a `usun konto()` method.
- Klient niezarejestrowany**: Contains attributes `PESEL: Long` and methods `sprawdź możliwość założenia konta()` and `załóż konto()`.
- Wpłata**, **Wyplata**, and **Przelew**: Transaction types with attributes like `miejsce: String` and `prowizja za wpłate = 10zł: int`. They are associated with **Transakcja**.
- Transakcja**: A transaction class with attributes `kwota: Long`, `data wykonania: Date`, `data potwierdzenia: Date`, `rodzaj transferu: enum`, and `forma płatności: Enum`.
- Bank**: A class with attributes `nazwa: String` and `adres: Address`.
- Umowa**: A contract class with attributes `podpis klienta: boolean` and `zatwierdzenie banku: boolean`.

Annotations include:

- atribut klasowy**: Points to the `opłata za konto` attribute in **Stan konta**.
- kompozycja**: Points to the composition relationship between **Stan konta** and **Konto**.
- asocjacja kwalifikowana**: Points to the association between **Konto** and **Transakcja**.
- klasa abstrakcyjna**: Points to the **Klient (abstract)** class.
- dynamic**: Points to the dynamic association between **Klient** and its subclasses.
- complete, disjoint**: Points to the relationship between **Klient zarejestrowany** and **Klient niezarejestrowany**.
- atribut opcjonalny** and **atribut zlozony**: Point to the `numer klienta` and `data urodzenia` attributes in **Klient zarejestrowany**.

## **Omówienie decyzji projektowych i skutków analizy dynamicznej:**

W związku z podjętą implementacją projektu w języku Java zostały podjęte następujące decyzje projektowe:

- klasa `Konto` została zmieniona z abstrakcyjnej na zwykłą na rzecz implementacji rodzaju dziedziczenia `overlapping` w której wszystkie inwarianty umieszczamy w jednej nadklasie oraz dodajemy do niej dyskryminator który nas informuje o rodzaju obiektu ( `enum Rodzaj konta`)
- większość asocjacji zostanie zastąpiona kompozycją
- dziedziczenie wieloaspektowe w klasie nadrzędnej `Transakcja` zostało zamienione na zwykłe rozszerzenie klas z wykluczeniem
- dziedziczenie dynamiczne będzie możliwe za pomocą zaimplementowanych metod bezpośrednio w klasach (`klient niezarejestrowany`, `klient zarejestrowany`)

## **9.Wymagania niefunkcjonalne:**

\*możliwość założenia konta: powyżej 18 roku życia

\*działalność: wybieramy z oficjalnej listy PKD

\*zniżka dla kont premium: nie większe niż 25%

\*waluta w transakcjach gotówkowych: możliwa wyłącznie w dolarach, euro, funtach, złotych

\*transfer: krajowy lub zagraniczny

\*dodanie konta przez bank: dla klienta powyżej 18 roku życia

## **10.Opis przyszłej ewolucji systemu:**

W przyszłości planowane jest dodanie opcji przyznawania kredytu przez bank.

## **11.Słownik:**

\* transakcja - usługa bankowa, wspomagająca obrót pieniędzmi, oraz zarządzaniem płatnościami

\* klient - podmiot, na którego bank świadczy co najmniej jedną z usług transakcyjnych

\* bank - instytucja prowadząca działalność polegającą na przyjmowaniu środków pieniężnych oraz wykonywaniu czynności w statusie banku ( transakcje bankowe) oraz zamianę klas z kategorii rodzaj płatności na enum z podaną informacją na ten temat

\* konto (rachunek bankowy) - podstawowe narzędzie, którym posługują się banki do zapisywania należności klienta

\* karta kredytowa - karta płatnicza, której wydanie jest związane z przyznaniem limitu kredytowego przez bank

\* gotówka - forma pieniądza w fizycznej postaci

\* umowa - wzajemne uzgodnienie dwóch stron (klienta i banku) na mocy której bank zobowiązuje się do przechowywania środków pieniężnych oraz do udostępnieniu klientowi możliwości dokonywania transakcji

## SCENARIUSZ

Nazwa przypadku użycia	Dodaj konto
Warunek początkowy	Klient powyżej 18 roku życia
Główny przepływ zdarzeń	<ol style="list-style-type: none"><li>1. Użycie funkcjonalności „dodaj konto” przez aktora Klient / Pracownik banku</li><li>2. System wysyła zapytanie do Klienta o nr PESEL</li><li>3. Na podstawie nr PESEL System weryfikuje datę urodzenia. Jeśli klient powyżej 18 roku życia, przesyła umowę do aktora Pracownika banku</li><li>4. Aktor Pracownik Banku zatwierdza umowę i odsyła do akceptacji przez aktora Klient</li><li>5. Aktor klient podpisuje umowę</li><li>6. System dodaje konto z danymi Klienta</li></ol>
Alternatywny przepływ zdarzeń	<p>3a. System weryfikuje datę urodzenia klienta. Aktor klient nie ukończył 18 roku życia.</p> <p>4a. Aktor Pracownik banku nie decyduje się na zatwierdzenie umowy, ze względu na niejasności w danych osobowych</p> <p>5a. Aktor Klient rozmyśla się i nie decyduje się na podpisanie umowy</p>

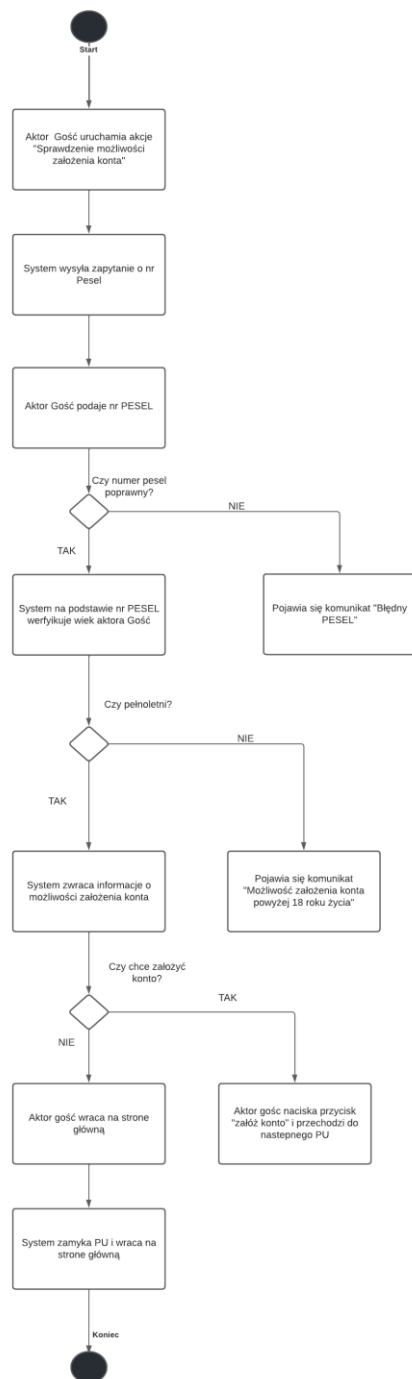
	6a. Proces dodawania konta nie przeszedł pomyślnie. Błędy walidacyjne
Zakończenie	W każdym momencie
Warunek końcowy	Zatwierdzenie umowy przez aktora Pracownik banku, podpisanie umowy przez aktora Klient

## SCENARIUSZ

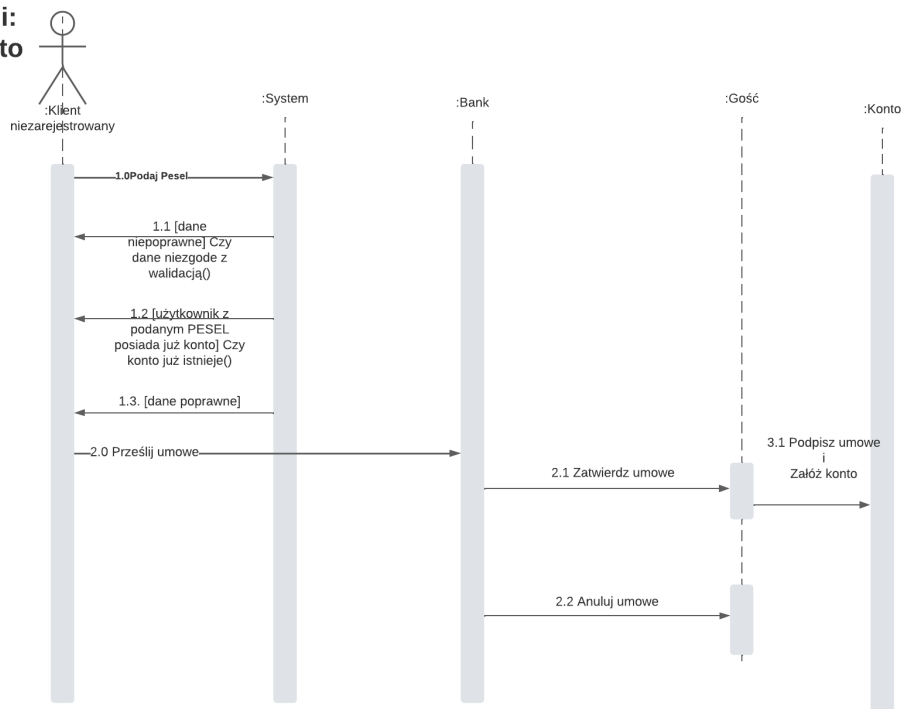
Nazwa przypadku użycia	Sprawdza możliwości założenia konta
Warunek początkowy	Gość niezarejestrowany
Główny przepływ zdarzeń	<ol style="list-style-type: none"> <li>1. Użycie funkcjonalności „Sprawdź możliwości założenia konta” przez aktora Gość</li> <li>2. System wysyła zapytanie do Klienta o nr PESEL</li> <li>3. Aktor gość podaje nr PESEL i zatwierdza akcje</li> <li>4. Na podstawie nr PESEL System weryfikuje datę urodzenia. Jeśli powyżej 18 roku życia pojawia się komunikat „Jest możliwość założenia konta” oraz odblokowuje się przycisk „Założ konto”</li> <li>5. Aktor Gość kończy weryfikację i za pomocą przycisku „Powrót do menu głównego” wraca na stronę główną aplikacji</li> <li>6. System kończy PU</li> </ol>
Alternatywny przepływ zdarzeń	<p>3a. Podany nr pesel nie przechodzi walidacji, pojawia się komunikat „Błędny pesel”</p> <p>4a. System weryfikuje datę urodzenia klienta. Aktor klient nie ukończył 18 roku życia.</p> <p>5a. Aktor gość decyduje się na założenie konta i uruchamia przypadek użycia Założenie konta</p>
Zakończenie	W każdym momencie
Warunek końcowy	Otrzymanie informacji o możliwości założenia konta



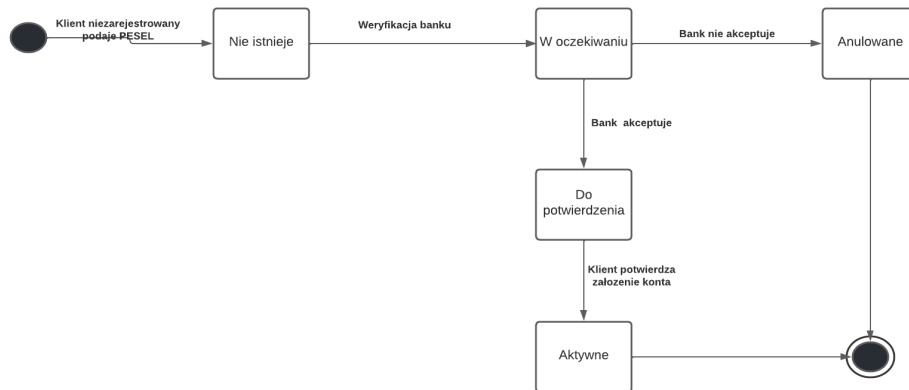
## Diagram aktywności



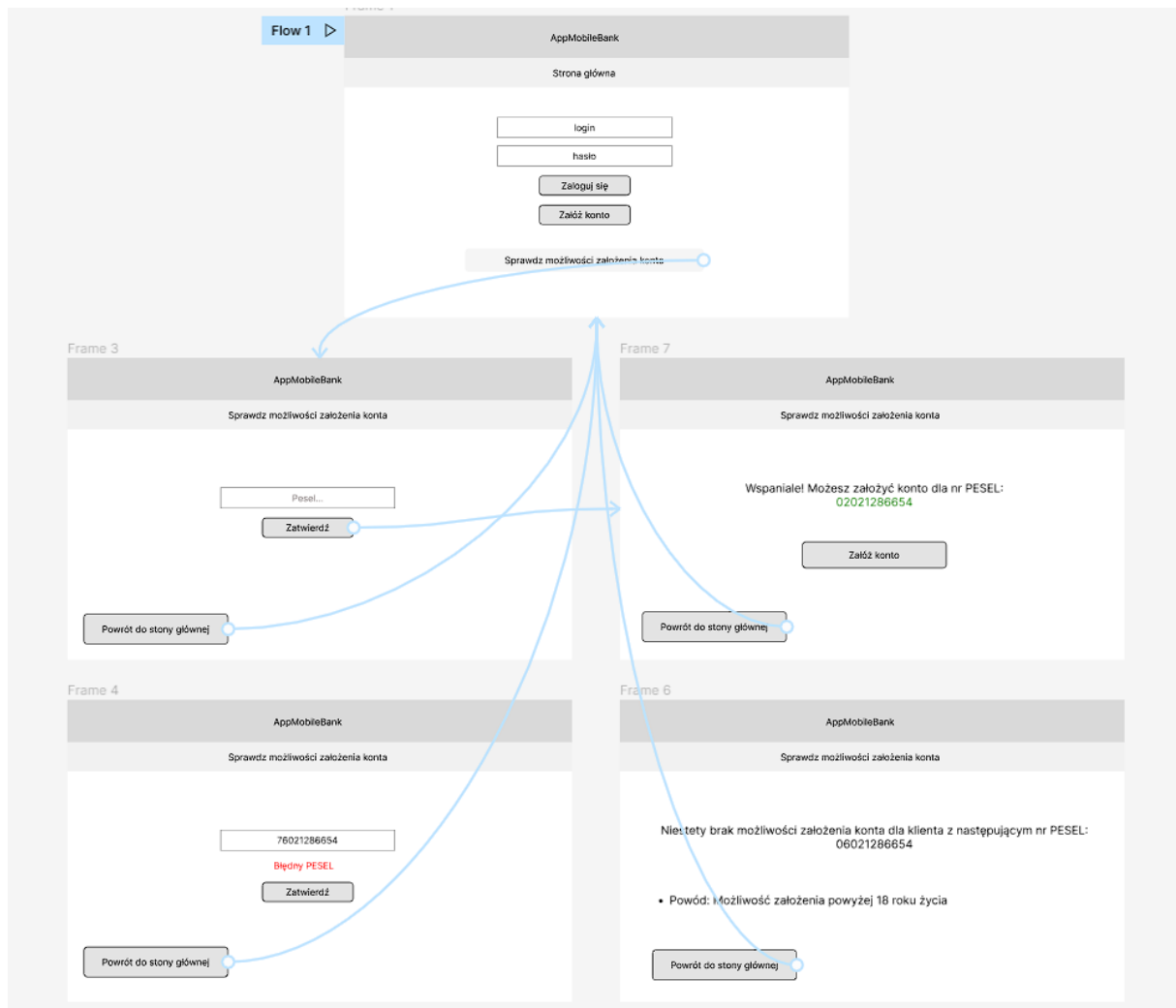
**Diagram  
sekwencji:  
Założ konto**



### Diagram stanu dla klasy Konto



Projekt interfejsu użytkownika dla przypadku użycia: Sprawdź możliwości założenia konta



## Zastosowane narzędzia

- Lucid.app -diagramy klas: analityczny i projektowy, diagramy przypadków użycia, diagram stanu, diagram aktywności, projekt GUI
- Microsoft Word

## Źródła wiedzy

- MAS informacje – dokument opisujący zakres dokumentacji
- PRI wykłady
- Byt wykłady
- Internet