# Projekt zespołowy – aplikacja bankomatu

Spis treści

[Projekt zespołowy – aplikacja bankomatu 1](#_Toc44089185)

[Wstęp 2](#_Toc44089186)

[Schemat aplikacji 2](#_Toc44089187)

[Warstwa Bazy Danych 2](#_Toc44089188)

[Warstwa DataAccessLayer 3](#_Toc44089189)

[Warstwa Aplikacji 5](#_Toc44089190)

[Warstwa Formularza 8](#_Toc44089191)

[Modele wytwarzania oprogramowania 13](#_Toc44089192)

[Osoby i role w projekcie 14](#_Toc44089193)

[Diagramy działania aplikacji 15](#_Toc44089194)

[Use Case Diagram 15](#_Toc44089195)

[Activity Diagram 16](#_Toc44089196)

[Sequence Diagram 17](#_Toc44089197)

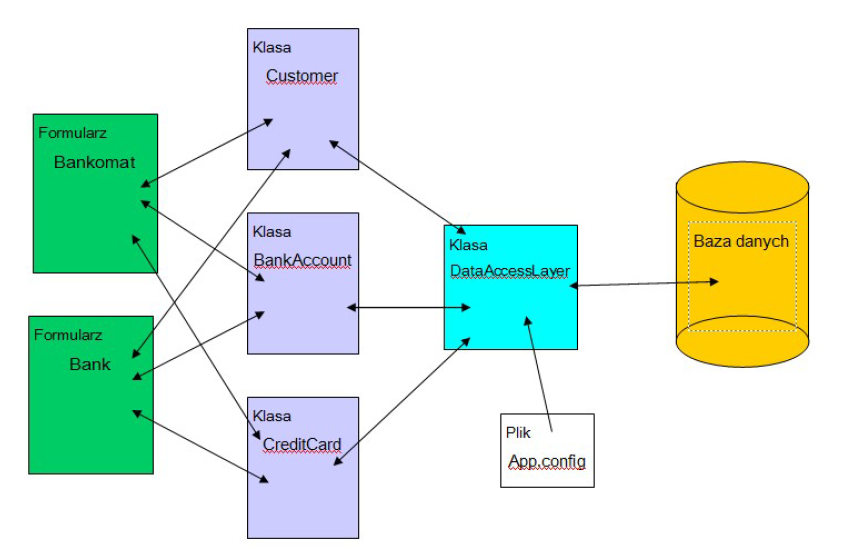
[Class Diagram 18](#_Toc44089198)

[Podsumowanie 18](#_Toc44089199)

## Wstęp

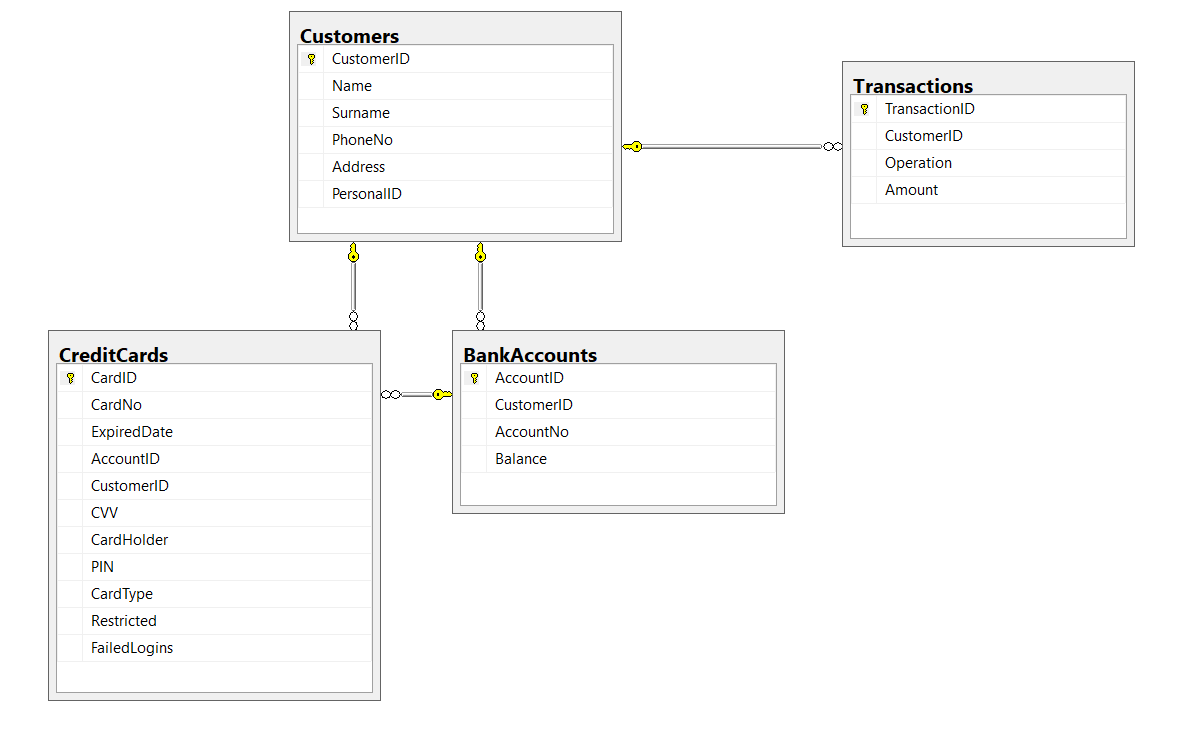
Projekt ma za zadanie stworzenie aplikacji symulującej działanie bankomatu. Aplikacja składa się z 4 warstw: warstwy bazy danych, warstwy dostępu do danych (DataAccessLayer), warstwy aplikacji i warstwy formularza.

### Schemat aplikacji



### Warstwa Bazy Danych

Warstwa bazy danych zawiera jedną bazę składającą się z 4 tabel: **Customer**, **BankAccounts**, **CreditCards** oraz **Transactions**. Poniżej widoczny jest schemat bazy danych wraz z powiązaniami pomiędzy tabelami.



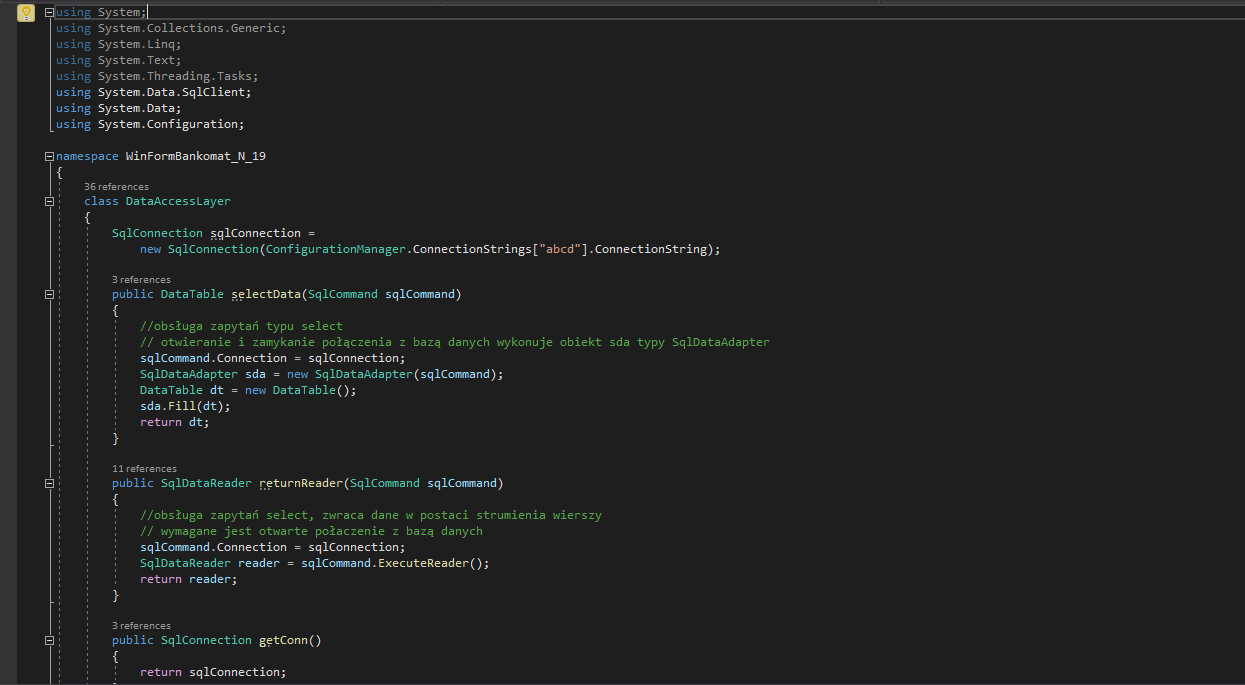
### Warstwa DataAccessLayer

Warstwa dostępu do danych - **DataAccessLayer** - pozwala na dostęp aplikacji do bazy danych. Umożliwia to globalny obiekt sqlConnection. Łańcuch połączenia z bazą danych (connectionString) znajduje się pliku konfiguracyjnym App.config. Odczytanie łańcucha połączenia z pliku konfiguracyjnego aplikacji umożliwia klasa ConfigurationManager znajdująca się w przestrzeni nazw System.Configuration.

Klasa DataAccessLayer zawiera ponadto 5 publicznych metod, które wystarczają do obsługi bazy danych.

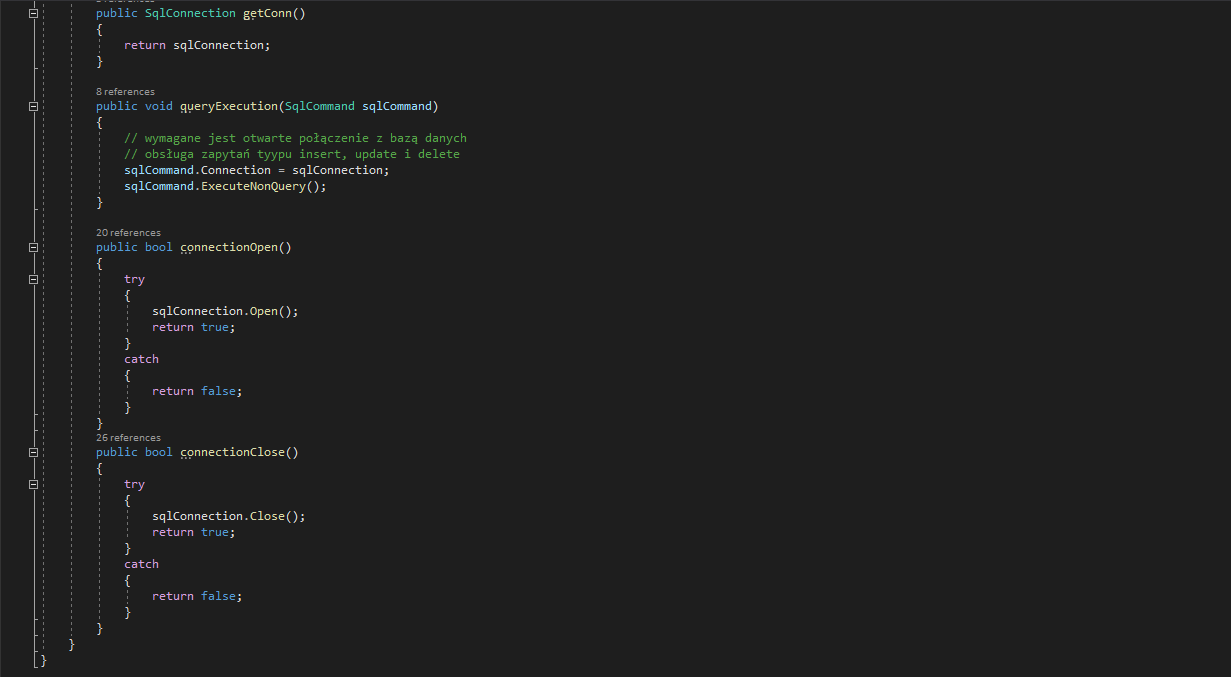
Metoda selectData(SqlCommand sqlCommand) obsługuje zapytania typu *select* i zwraca wynik tego zapytania w tabeli danych (DataTable).

Metoda selectReader(sqlCommand sql Command) również obsługuje zapytania typu *select*, ale zwraca wynik w postaci strumienia wierszy (SqlDataReader). Metodę tę można wykorzystać do znalezienia pojedyńczej wartości w bazie danych.



Metody connectionOpen() oraz connectionClose() pozwalają na otwieranie i zamykanie połączenia z bazą danych.

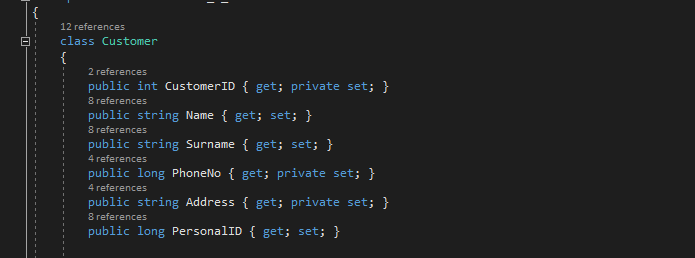
Metoda queryExecution(sqlCommand sqlCommand) służy do wykonywania zapytań typu *insert, update* i *delete*. Metodę tę wykorzystuje się do aktualizacji salda na rachunku czy wprowadzania danych nowego klienta, rachunku lub karty.

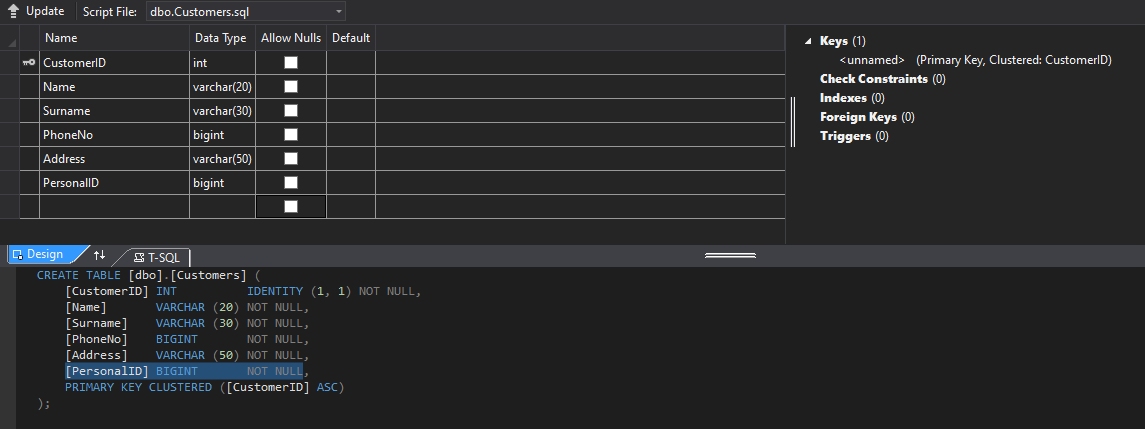


### Warstwa Aplikacji

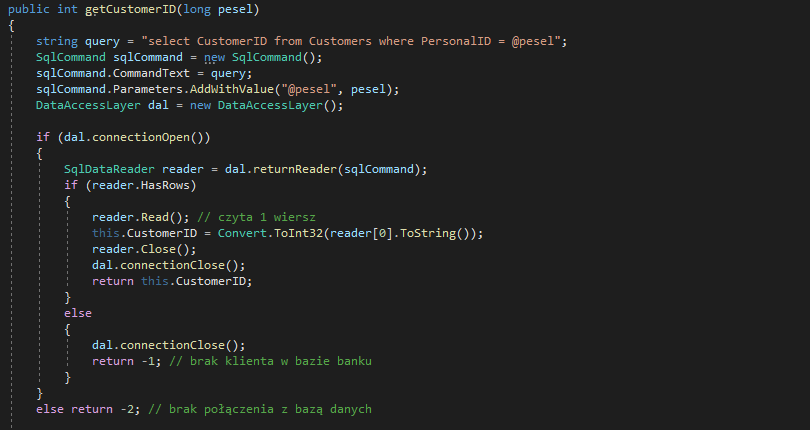
Warstwa aplikacji składa się z 4 klas: **Customer**, **CreditCard**, **BankAccount** i **Transaction**. Metody tych klas związane są z odpowiadającymi im tabelami w bazie danych. Wszystkie kolumny tabel odpowiadają właściwościom ich klas. Ponadto zawierają one szereg metod umożliwiających odczyt i zapis danych do bazy.

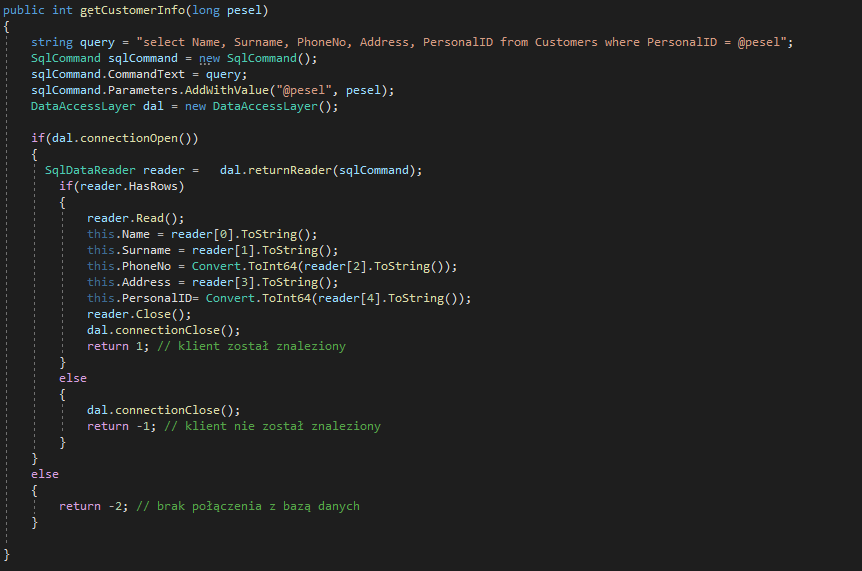
Właściwości klasy **Customer** odpowiadają kolumnom w tabeli **Customers**.





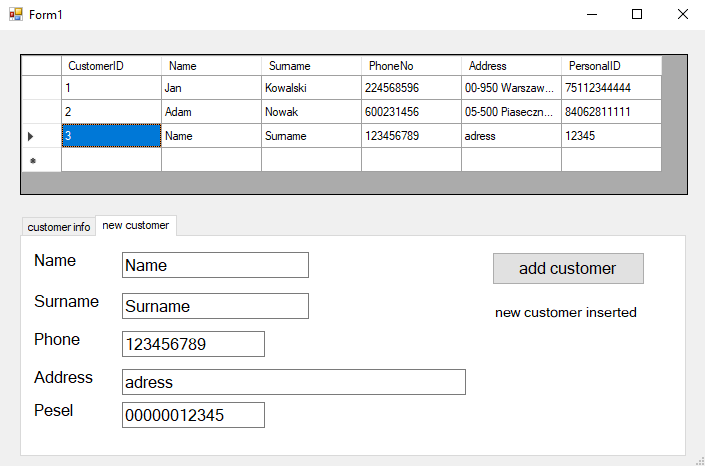
Oprócz właściwości, w klasie **Customer** znajdują się 2 publiczne metody getCustomerID(long pesel) oraz getCustomerInfo(long pesel), które jako argument przyjmują nr PESEL. Pierwsza metoda zwraca wartość klucza głównego z tabeli **Customers** szukanego klienta, druga ustawia właściowści obiektu typu Customer. Obie metody korzystają z obiektu sqlCommand przypisując jego właściwości sqlCommand.CommandText odpowiednie zapytanie typu *select*, a następnie wywołują metodę dal.returnReader(sqlCommand) na obiekcie dal typu DataAccessLayer. Metoda returnReader(sqlCommand) zwraca obiekt typu SqlDataReader (reader), gdzie dostęp do poszczególnych wierszy uzyskuje się poprzez wywołanie metody read().





Ponadto, w klasie **Customer** znajduje się publiczna metoda addCustomer(), która pobiera dane z formularza. Jak wcześniej, metoda korzysta z obiektu sqlCommand przypisując jego właściwości sqlCommand.CommandText zapytanie typu *insert*, a następnie wywołuje metodę dal.queryExecution(sqlCommand).



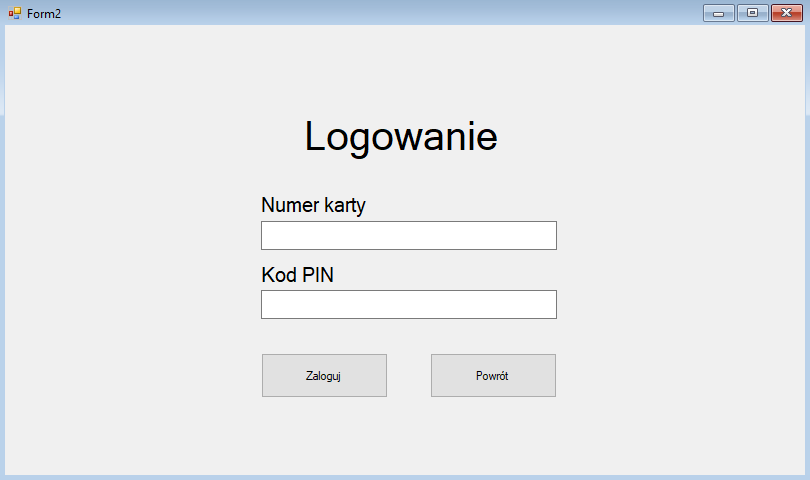


### Warstwa Formularza

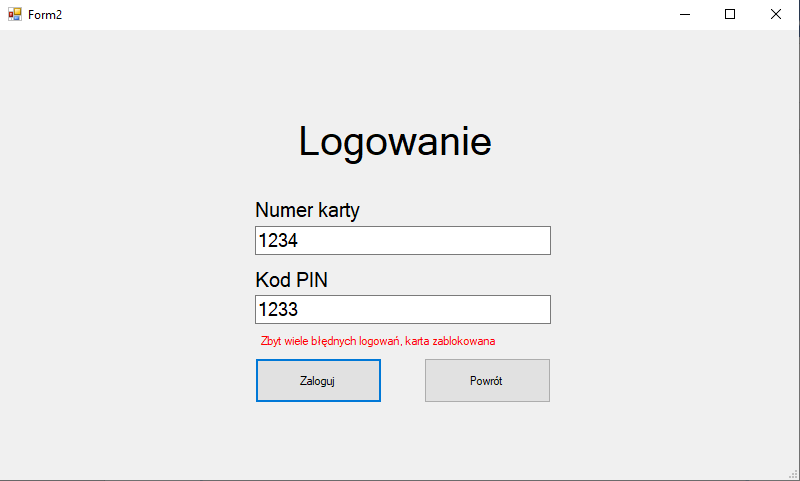
Warstwa Formularza składa się z formularza Bankomat, który ma za zadanie symulować działanie bankomatu, oraz formularza Bank, w pewnym sensie symulujący działanie okienka kasowego w banku.



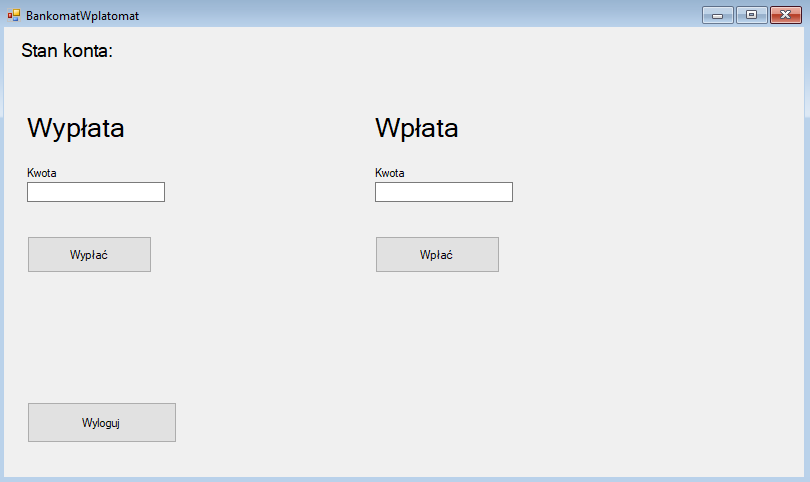
Użytkownik ma możliwość wyboru, czy chce zalogować się do aplikacji jako Klient czy jako Pracownik. W przypadku wyboru logowania jako Klient, aplikacja umożliwi wpisanie nr karty i kodu PIN w odpowiednie pola edycyjne oraz zatwierdzenie operacji przyciskiem.



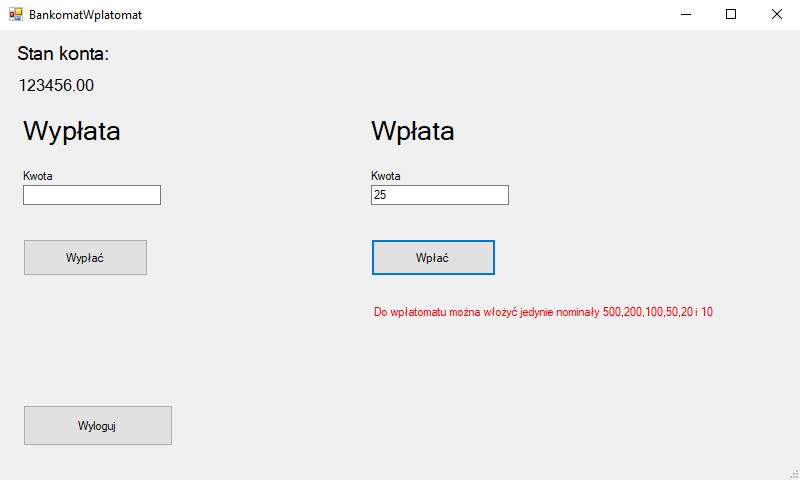
Aplikacja umożliwia zasymulowanie sytuacji, w której karta jest przeterminowana, zablokowana lub skradziona, co oznajmi odpowiednim komunikatem. Trzykrotne wpisanie błędnego PINu również skutkuje zablokowaniem karty i wyświetleniem odpowiedniego komunikatu.



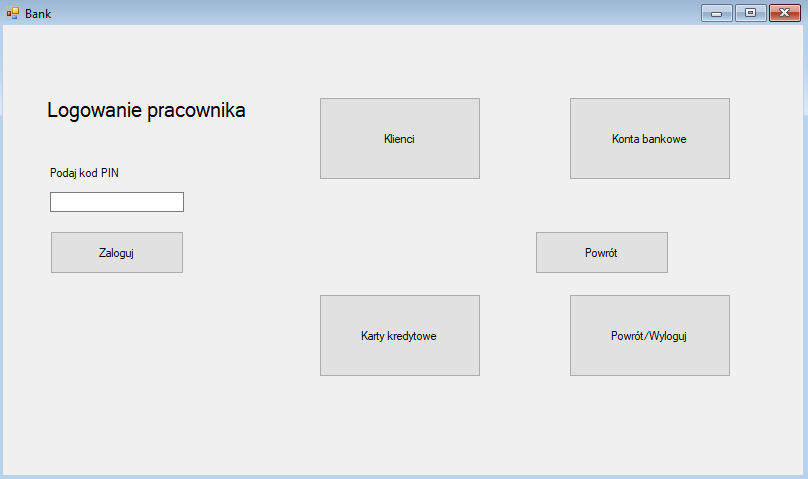
Po zalogowaniu jako Klient, użytkownik ma dostęp do informacji o stanie środków na koncie oraz ma możliwość wypłaty i wpłaty gotówki.



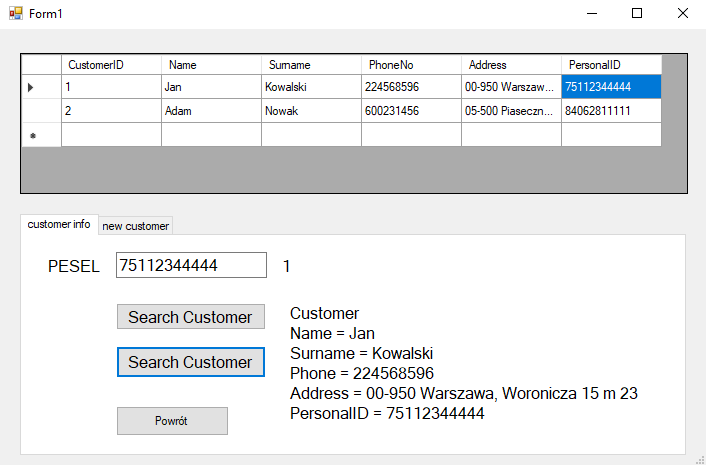
System pozwala na wypłaty i wpłaty wyłącznie banknotów, a więc kwota, jaką chcemy wpłacić lub wypłacić musi być możliwa do zaprezentowania jako suma nominałów. Nie spełnienie tego warunku skutkuje wyświetleniem odpowiedniego komunikatu. Również próba wypłacenia większej ilości pieniędzy, niż jest na koncie, jest niemożliwa i skutkuje wyświetleniem informacji z błędem.



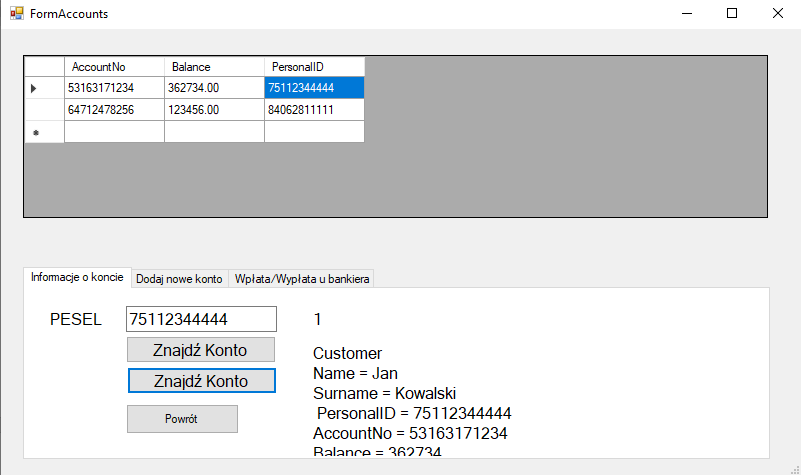
Zalogowanie jako Pracownik pozwala na uzyskanie szeregu informacji o Klientach, kontach bankowych i kartach kredytowych.



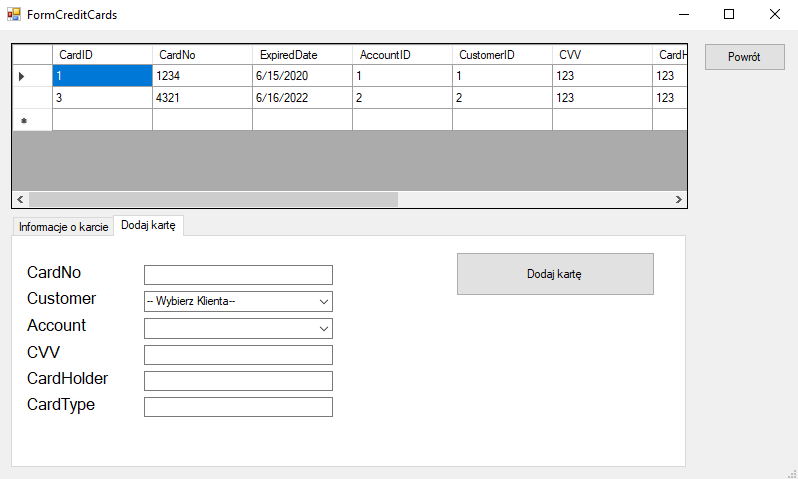
Opcja „Klienci” pozwala na szukanie Klientów według różnych kryteriów oraz na dodawanie nowych Klientów.



Opcja „Konta bankowe” umożliwia wyszukiwanie informacji o istniejących kontach, dodawanie nowych kont oraz dokonywanie wypłat i wpłat u bankiera.



Opcja „Karty kredytowe” wyswietla informacje o kartach kredytowych, umożliwia szukanie konkretych kart według kryteriów wyszukiwania oraz na dodawanie nowych kart.



## Modele wytwarzania oprogramowania

W procesie tworzenia oprogramowania możemy wyróżnić dwa podstawowe modele: model kaskadowy oraz model cykliczny. W obu modelach możemy wyróżnić następujące kroki: planowanie, analiza, projektowanie, implementacja (developement), testowanie oraz utrzymywanie. Model kaskadowy polega na wykonywaniu podstawowych czynności jako odrębnych faz projektowych, kolejno po sobie. Jeśli któraś z faz zwróci niesatysfakcjonujący produkt cofamy się wykonując kolejne iteracje aż do momentu kiedy otrzymamy satysfakcjonujący produkt. W modelu cyklicznym wykonujemy następujące kroki w cyklach, aż do momentu otrzymania zadowalającego produktu. Na każdym etapie możliwa jest rewizja procesu i dostosowanie produktu do ciągle zmieniających się potrzeb i uwag Klienta.

## Osoby i role w projekcie

Zadania do wykonania w projekcie zostały przydzielone do konkretnych osób ze względu na ich preferencje oraz na posiadane umiejętności. Każda osoba z grupy pełni określoną rolę w projekcie:

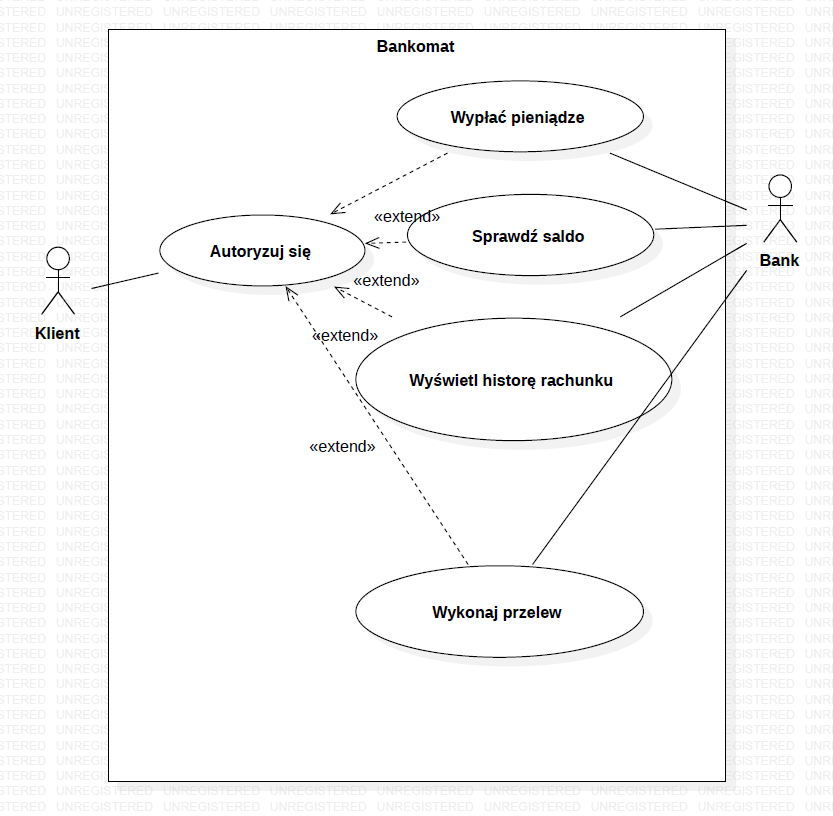
1. Kamil Prokopiuk – Project Manager – zarządzanie projektem, przydzielanie zadań, przegląd i kontrola zmian w kodzie, sporządzanie dokumentacji.
2. Adam Szymański – Analityk – dba o to aby projekt spełniał wymagania biznesowe i osiągał zamierzone cele, istotne wsparcie w trakcie wybierania rozwiązań, określania celów i nadania projektowi odpowiedniego kierunku, również sporządzanie dokumentacji w postaci diagramów.
3. Hanna Kasprzak – Architekt baz danych – projekt, implementacja i utrzymanie bazy danych spełniającej wymagania aplikacji, stworzenie schematu bazy danych.
4. Artur Rosiński – UI/UX Designer – osoba odpowiedzialna na stworzenie nowoczesnego i intuicyjnego GUI aplikacji.
5. Sylwester Kwiatkowski – Programista i Architekt Systemu – jedna z najważniejszych osób w zespole, najbardziej doświadczony programista, który dba o całokształt aplikacji, jakość kodu, wybiera najlepsze rozwiązania, technologie i narzędzia, rewizja zmian proponowanych przez programistów, refaktoryzacja kodu, logika systemu.
6. Mateusz Bugowski – Programista – odpowiada za kod od strony Backendu, czyli za logikę systemu, sporządzanie diagramów.
7. Adam Żuk – Programista – zajmuje się częścią kodu, z którą interakcję prowadzi użytkownik, autor wielu nowych funkcjonalności.
8. Dawid Ciecierski – Tester – testowanie zmian i nowych funkcjonalności aplikacji na różnych etapach implementacji kodu.
9. Aleksandra Kuźniar – Tester – testowanie finalnej wersji aplikacji z punktu widzenia Klienta.

Wszytkie osoby w grupie brały aktywny i czynny udział w całym procesie tworzenia aplikacji i przygotowywania niezbędnej dokumentacji.

## Diagramy działania aplikacji

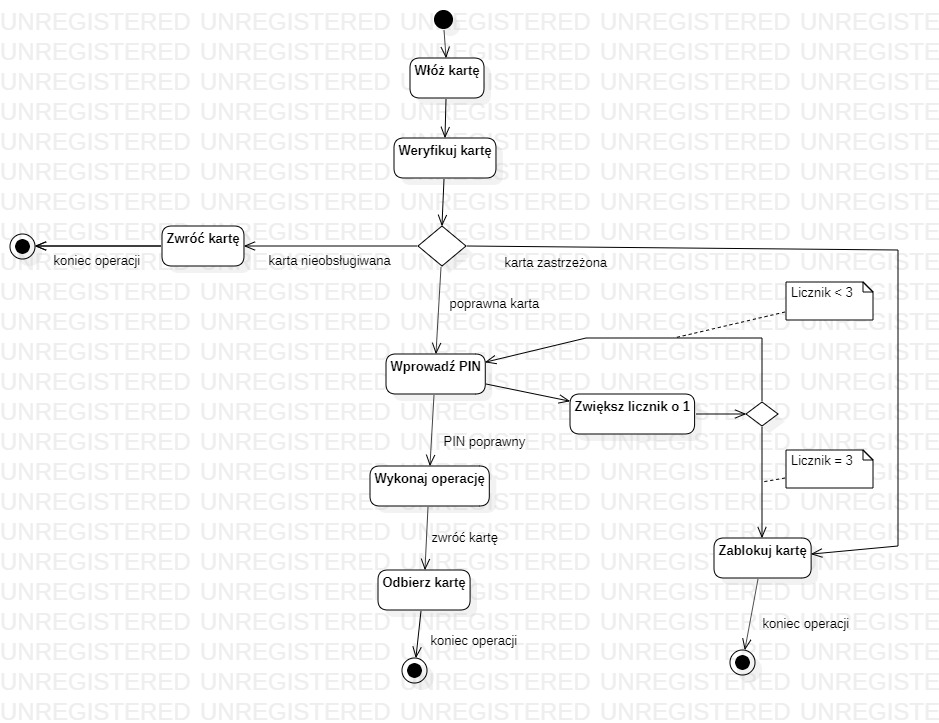
### Use Case Diagram

Diagram przypadków użycia przedstawia funkcjonalność systemu wraz z jego otoczeniem. Pozwala na graficzne zaprezentowanie własności systemu tak, jak są one widziane po stronie użytkownika oraz służy do zobrazowania usług, które są widoczne z zewnątrz systemu.



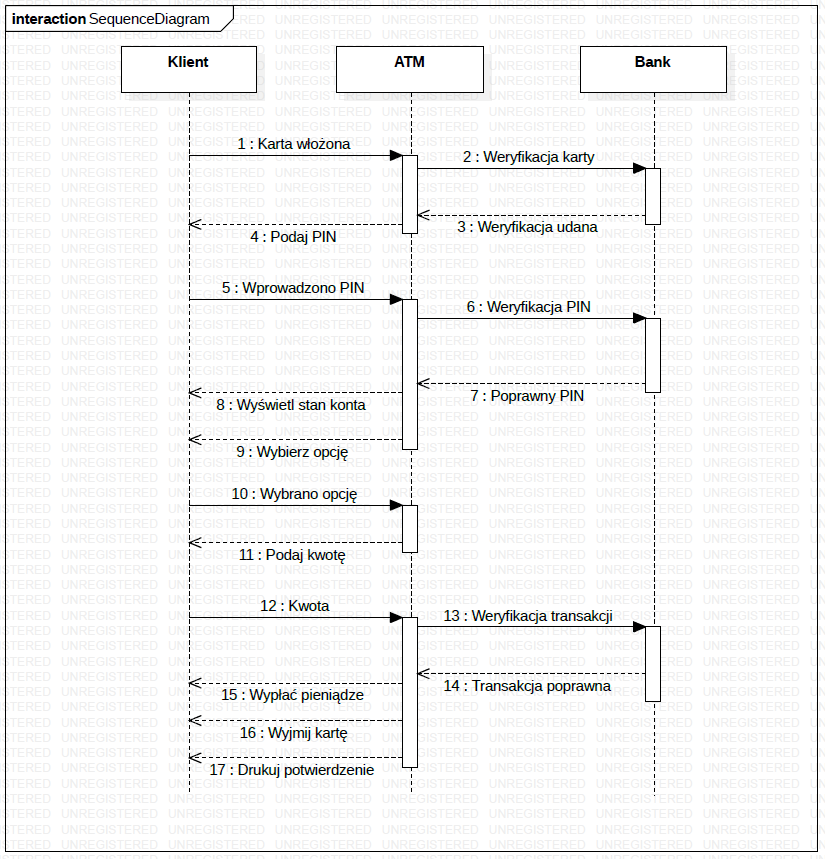
### Activity Diagram

Diagram aktywności jest diagramem interakcji, który służy do modelowania dynamicznych aspektów systemu. Na diagramie widoczne są kolejne czynności, które wykonuje Klient i Bankomat, uwzględniając wszystkie możliwe przypadki, od włożenia karty do bankomatu, poprzez weryfikację i wprowadzenie kodu PIN, aż po wykonanie operacji i odbiór karty. Również opisane zostały przypadki, w których wykonanie operacji jest niemożliwe, jak włożenie nieobsługiwanej karty, karty zastrzeżonej, czy trzykrotne wpisanie błędnego PINu.



### Sequence Diagram

Diagram sekwencji służy do prezentowania interakcji pomiędzy obiektami wraz z uwzględnieniem w czasie komunikatów, jakie są przesyłane pomiędzy nimi. Zasadniczym zastosowaniem diagramów sekwencji jest modelowanie zachowania systemu w kontekście scenariuszy przypadków użycia. Diagramy sekwencji pozwalają uzyskać odpowiedź na pytanie, jak w czasie przebiega komunikacja pomiędzy obiektami. Dodatkowo diagramy sekwencji stanowią podstawową technikę modelowania zachowania systemu, które składa się na realizację przypadku użycia. Poniższy diagram prezentuje poprawnie wykonaną operację wypłaty gotówki.



### Class Diagram

Diagram klas przedstawia strukturę systemu w modelach obiektowych przez ilustrację struktury klas i zależności między nimi.



## Podsumowanie

Z punktu widzenia Klienta projekt symuluje działanie aplikacji bankomatu w wersji podstawowej, pozwalając na wyłaty i wpłaty gotówki. Po stronie Pracownika aplikacja pozwala na obsługę kilku dodatkowych usług, jak przegląd klientów, kont i kart kredytowych, oraz wypłaty i wpłaty gotówki, symulując w ten sposób działanie okienka kasowego banku. Projekt można z powodzeniem rozbudować o funkcje i moduły do obsługi wykonywania przelewów i wyświetlania historii transakcji z wykorzystaniem dodatkowej klasy **Transaction** oraz wykorzystać do tego celu bazę **Transactions**, która została już zdefiniowana w bazie danych.