

POLITECHNIKA WROCŁAWSKA
WYDZIAŁ ELEKTRONIKI

PROJEKT Z BAZ DANYCH

System zarządzania kontem bankowym

Termin zajęć: Poniedziałek, 13:15–15:00

AUTOR/AUTORZY:

Marek Morys

Indeks: 235034

Email: 235034@student.pwr.edu.pl

PROWADZĄCY ZAJĘCIA:

dr inż. Roman Ptak, W4/K9

Magdalena Olchawa

Indeks: 235061

Email: 235061@student.pwr.edu.pl

Mateusz Markowski

Indeks: 235605

Email: 235605@student.pwr.edu.pl

Wrocław, 2018 r.

Spis treści:

1. Wstęp.....	4
1.1. Cel projektu.....	4
1.2. Zakres projektu.....	4
2. Analiza wymagań.....	4
2.1. Opis działania i schemat logiczny systemu.....	4
2.2. Wymagania funkcjonalne.....	5
2.3. Wymagania нефункционалне.....	6
2.3.1. Wykorzystywane technologie i narzędzia.....	6
2.3.2. Wymagania dotyczące rozmiaru bazy danych.....	6
2.3.3. Wymagania dotyczące bezpieczeństwa systemu.....	6
3. Projekt systemu.....	7
3.1. Projekt bazy danych.....	7
3.1.1. Analiza rzeczywistości i uproszczony model konceptualny.....	7
3.1.2. Model logiczny i normalizacja.....	9
3.1.3. Model fizyczny i ograniczenia integralności danych.....	10
3.1.4. Inne elementy schematu – mechanizmy przetwarzania danych.....	11
3.1.5. Projekt mechanizmów bezpieczeństwa na poziomie bazy danych.....	11
3.2. Projekt aplikacji użytkownika.....	13
3.2.1. Architektura aplikacji i diagramy projektowe.....	13
3.2.2. Interfejs graficzny i struktura menu.....	19
3.2.3. Metoda podłączania do bazy danych – integracja z bazą danych.....	29
3.2.4. Projekt zabezpieczeń na poziomie aplikacji.....	29
4. Implementacja systemu baz danych.....	29
4.1. Tworzenie tabel i definiowanie ograniczeń.....	29
4.2. Implementacja mechanizmów przetwarzania danych.....	32
4.3. Implementacja uprawnień i innych zabezpieczeń.....	36
4.4. Testowanie bazy danych na przykładowych danych.....	38
5. Implementacja i testy aplikacji.....	41
5.1. Instalacja i konfigurowanie systemu.....	41
5.2. Instrukcja użytkowania aplikacji.....	41
5.3. Testowanie opracowanych funkcji systemu.....	41
5.4. Omówienie wybranych rozwiązań programistycznych.....	41

5.4.1. Implementacja interfejsu dostępu do bazy danych	42
5.4.2. Implementacja wybranych funkcjonalności systemu	47
5.4.3. Implementacja mechanizmów bezpieczeństwa	47
6. Podsumowanie i wnioski	47
Bibliografia.....	47

1. Wstęp

1.1. Cel projektu

Celem projektu jest zaprojektowanie oraz realizacja aplikacji systemu bankowego.

1.2. Zakres projektu

Projekt obejmuje identyfikację wymagań funkcjonalnych oraz нефункциональных systemu na podstawie rzeczywistego opisu sytuacji biznesowej. Następnie zamodelowana zostanie baza danych odzwierciedlająca strukturę przechowywania danych oraz zaprojektowana zostanie aplikacja dostępowa. Kolejnym krokiem będzie implementacja systemu (bazy danych oraz aplikacji dostępowej). Całość zostanie zakończona prezentacją systemu oraz przedstawieniem dokumentacji.

2. Analiza wymagań

2.1. Opis działania i schemat logiczny systemu

System ma posłużyć pracownikom lokalnego banku. Bank ten nie jest dużą placówką i pozwala klientom na najbardziej podstawowe działania, od założenia konta, poprzez uzyskiwanie informacji na temat wykonywanych transakcji, edycję swoich danych, wykonywanie płatności czy też zamknięcie rachunku. Dodatkowo bank umożliwia założenie lokaty czy wzięcie pożyczki. W banku wyróżnia się strukturę hierarchiczną, przy czym wszyscy pracownicy lokalnej placówki powinni mieć ten sam poziom dostępu do systemu, by móc łatwo zastępować siebie nawzajem w poszczególnych obowiązkach. Zamawiający system (czyli bank) postawił wysokie wymagania jeśli chodzi o bezpieczeństwo przechowywanych danych, z uwagi na ich newralgiczność.

Finalny system ma stanowić pomoc dla pracowników banku w zakresie tworzenia konta klienta. Jest to jedna z najbardziej podstawowych funkcjonalności systemu i musi ją cechować szybkość oraz intuicyjność (najlepiej, jeśli do tworzenia konta zostanie wykorzystany odpowiedni formularz z walidacją danych). W formularzu powinny się znaleźć pola dotyczące danych osobowych klienta (imię, nazwisko, PESEL, adres, NIP, numer telefonu). W wyniku rejestracji konta klient powinien otrzymywać indywidualny numer rachunku oraz dane dostępowe do logowania. Ponad to pracownik

powinien mieć możliwość księgowania przelewów oraz dokonywania wypłat z konta na życzenie klienta. Przelew i wypłata powinny być charakteryzowane przez tytuł, kwotę i datę wykonania. System musi także umożliwić pracownikowi uzyskanie informacji na temat założonego rachunku, a także edycję tych danych, jeśli klient wyrazi taką chęć. Pracownik będzie mógł również, korzystając z dostarczonego systemu, odnotować chęć wzięcia pożyczki przez klienta. Pożyczkę będą charakteryzowały: kwota, oprocentowanie, długość okresu oraz data utworzenia. Pracownik, na życzenie klienta, musi za pomocą systemu móc dostarczyć klientowi informacje na temat wykonanych transakcji z dowolnie wybranego okresu. Oprócz tego klient, korzystając z własnego konta w aplikacji systemu bankowego, powinien sam móc poznać historię swoich transakcji (z całego okresu lub z zadanego przedziału) oraz aktualny stan konta, a także wykonać przelew.

2.2. Wymagania funkcjonalne

- ✓ Obsługa logowania pracowników;
- ✓ Obsługa logowania klientów;
- ✓ Możliwość utworzenia konta bankowego przez pracownika, na życzenie klienta i po podaniu przez niego danych;
- ✓ Możliwość uzyskania danych na temat istniejącego konta bankowego przez pracownika, na życzenie klienta lub celem weryfikacji;
- ✓ Możliwość modyfikacji konta bankowego przez pracownika, na życzenie klienta;
- ✓ Możliwość usuwania konta bankowego przez pracownika, na wniosek klienta;
- ✓ Możliwość dokonywania przelewów między kontami bankowymi zarówno przez pracownika, jak i klienta;
- ✓ Możliwość sprawdzania historii transakcji na koncie, zarówno przez pracownika, jak i klienta;
- ✓ Możliwość udzielenia pożyczki dla danego konta przez pracownika, na podstawie oferty wynikającej ze zdolności kredytowej. Pożyczka zostaje udzielona na

konkretne konto przy założonym oprocentowaniu. Raty są miesięczne. Istnieje możliwość comiesięcznego potrącania z konta kwoty raty;

- ✓ Możliwość założenia konta oszczędnościowego dla konkretnego rachunku. Oprocentowanie ustalane w skali roku;
- ✓ Możliwość wpłacenia pieniędzy na lokatę przez pracownika.

2.3. Wymagania niefunkcjonalne

2.3.1. Wykorzystywane technologie i narzędzia

- ✓ Projektowanie: Visual Paradigm (system), Moqups (aplikacja)
- ✓ System Zarządzania Bazą Danych: MySQL
- ✓ Back-end: node.js
- ✓ Klient: Java

2.3.2. Wymagania dotyczące rozmiaru bazy danych

Baza danych powinna umożliwiać założenie i przechowywanie informacji na temat 20 000 kont. Dla każdego konta przyjmuje się możliwość zapisania 20 pożyczek oraz informacji na temat 1000 transakcji rocznie.

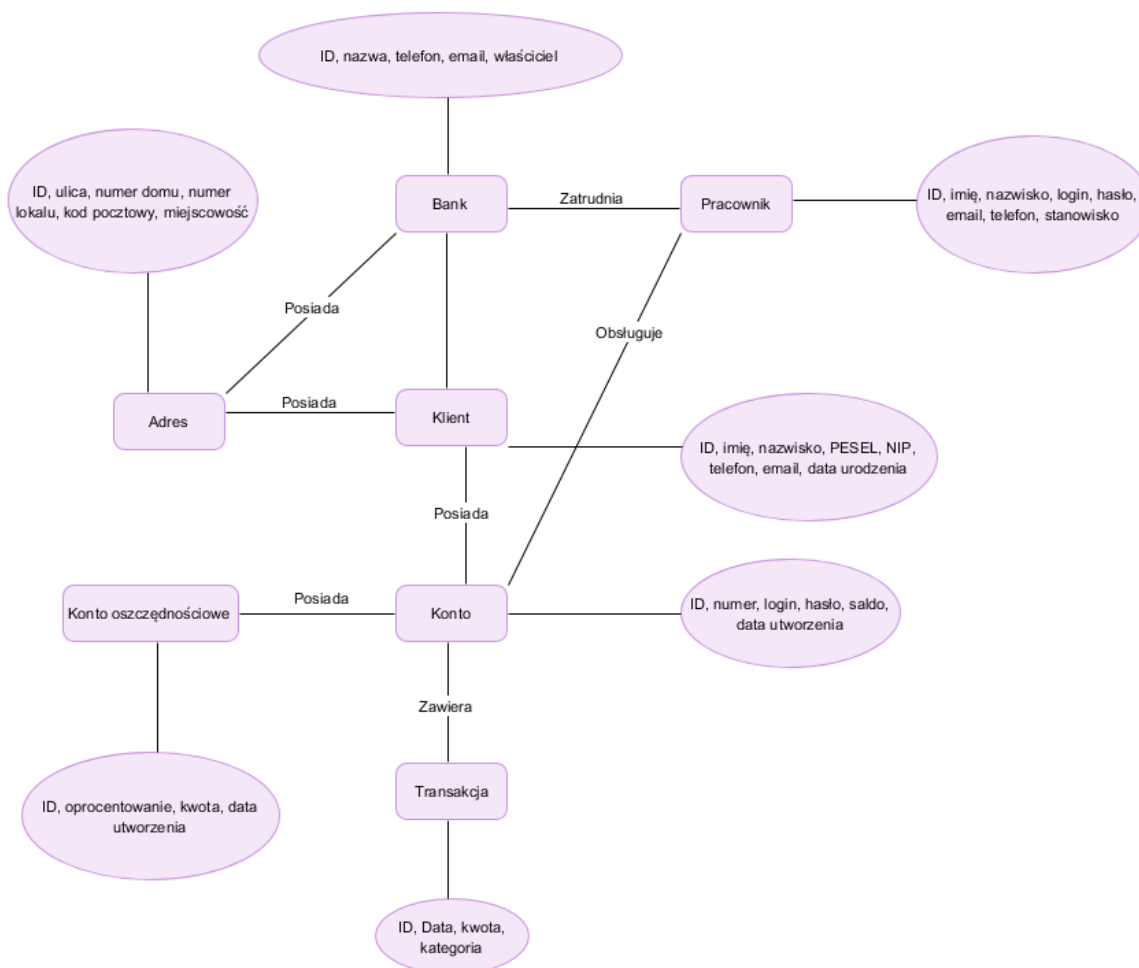
2.3.3. Wymagania dotyczące bezpieczeństwa systemu

System będzie korzystał z uwierzytelniania podczas logowania za pomocą loginu oraz hasła, a także autoryzację poprzez różne poziomy dostępności informacji i funkcji (pracownik/klient). Hasła będą przechowywane w bazie danych w postaci zaszyfrowanej (SHA-256). Komunikacja będzie odbywała się połączeniem szyfrowanym TLS 1.1/1.2.

3. Projekt systemu

3.1. Projekt bazy danych

3.1.1. Analiza rzeczywistości i uproszczony model konceptualny



Rysunek 1 Model konceptualny bazy danych systemu bankowego

Opis relacji:

Bank-klient – jeden do wielu,

Bank-pracownik – jeden do wielu,

Klient-konto – jeden do wielu,

Konto-transakcja – jeden do wielu,

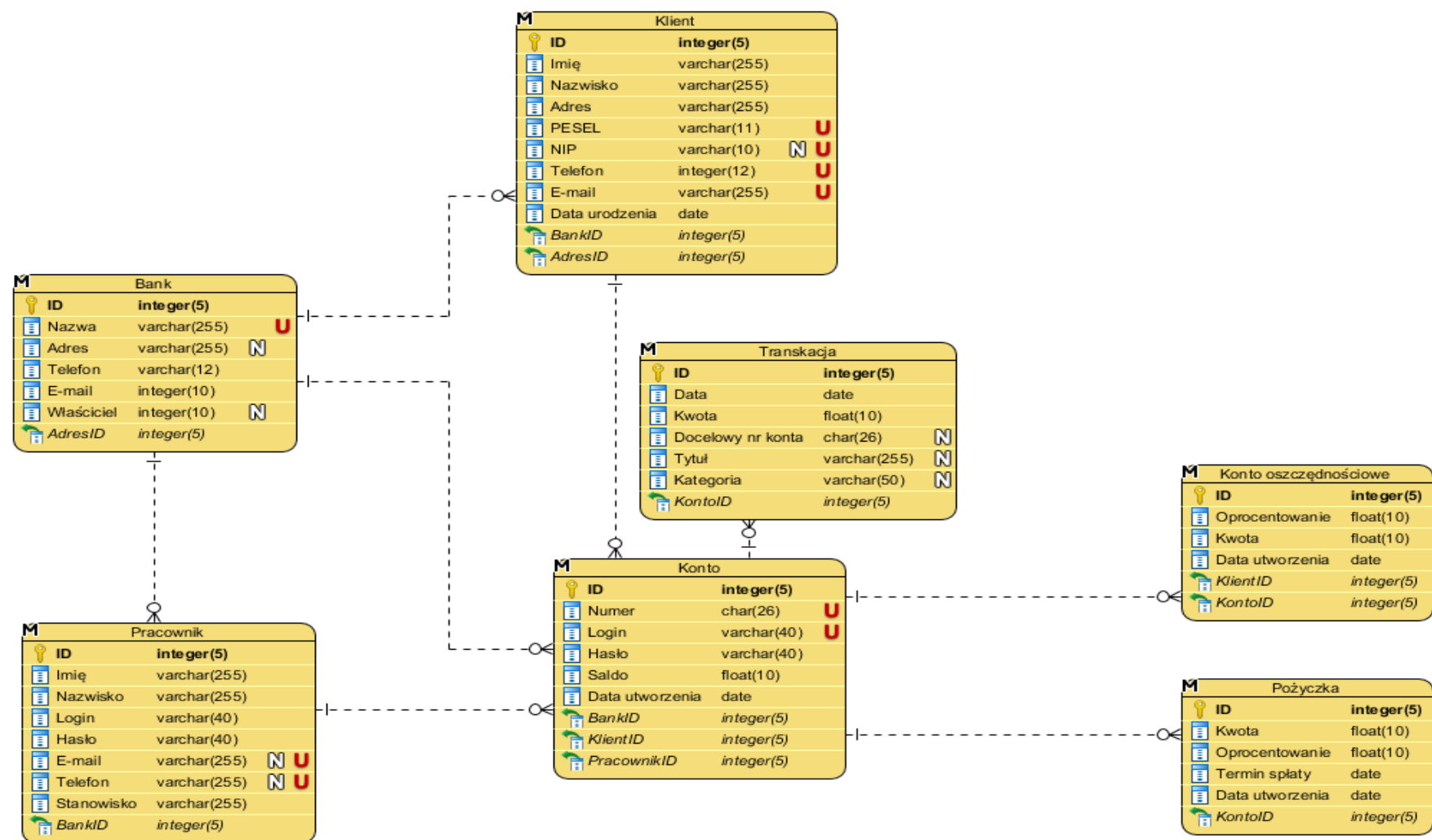
Adres-klient – jeden do wielu,

Bank-adres – jeden do jednego,

Konto-konto oszczędnościowe – jeden do wielu,

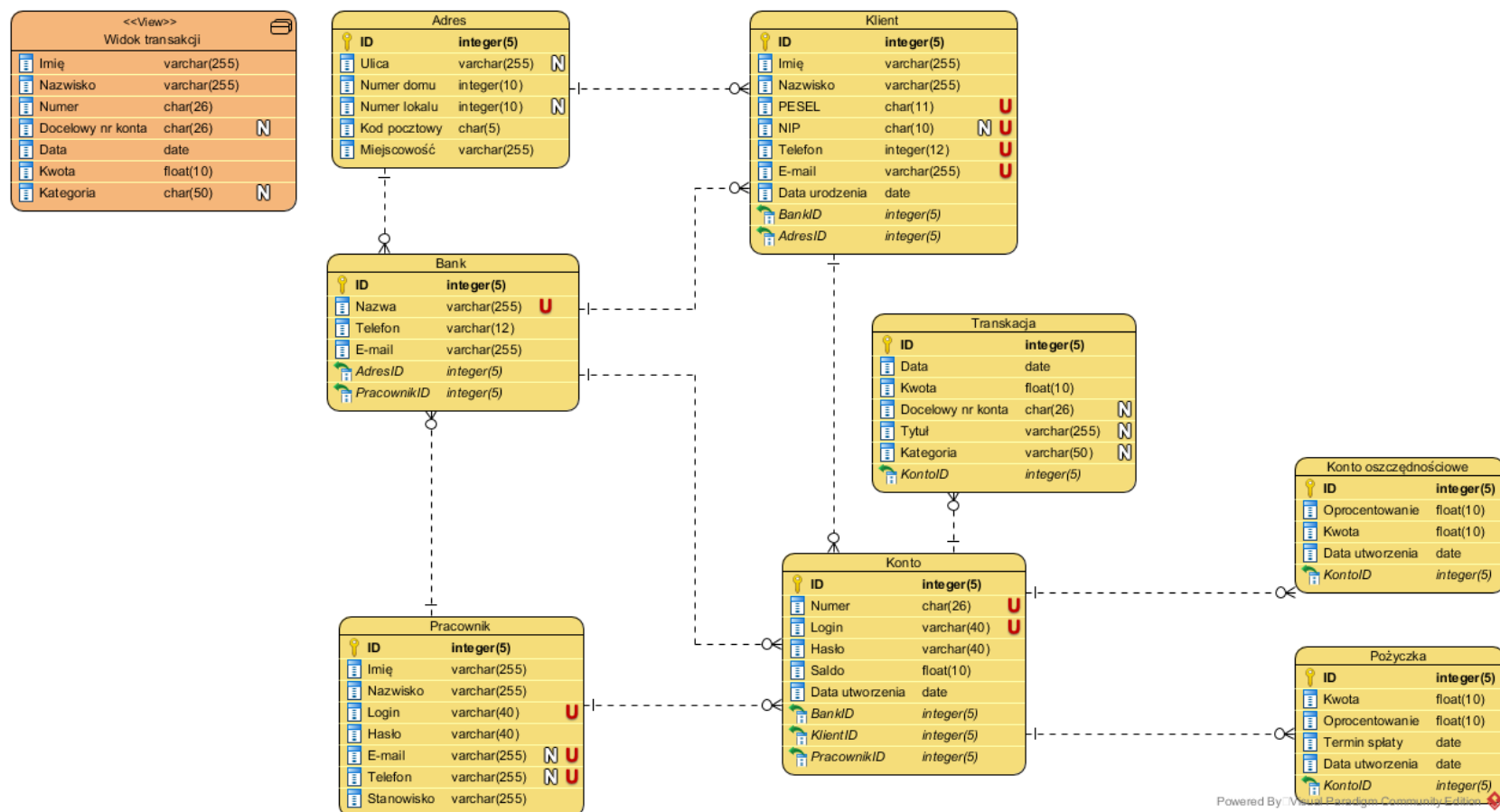
Pracownik-konto – jeden do wielu

3.1.2. Model logiczny i normalizacja



Rysunek 2 Model logiczny bazy danych systemu bankowego

3.1.3. Model fizyczny i ograniczenia integralności danych



Rysunek 3 Model fizyczny bazy danych systemu bankowego

3.1.4. Inne elementy schematu – mechanizmy przetwarzania danych

Trigger_transakcja – po utworzeniu nowej transakcji (przelew, wpłata, wypłata, spłata kredytu) zostaje dodany nowy wpis do logów transakcji, zawierający typ transakcji, datę, kwotę, oraz konto źródłowe i docelowe jeśli jest potrzebne.

Trigger_splata_kredytu – po każdej transakcji typu „spłata kredytu” pozostała kwota do spłaty zostaje pomniejszona o podaną sumę.

Trigger_okres_rozliczeniowy – co roku od dnia założenia konta oszczędnościowego i wpłaceniu na nie pieniędzy, stan konta jest powiększany o przyznany procent wpłaconej kwoty na to konto.

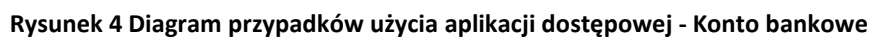
3.1.5. Projekt mechanizmów bezpieczeństwa na poziomie bazy danych

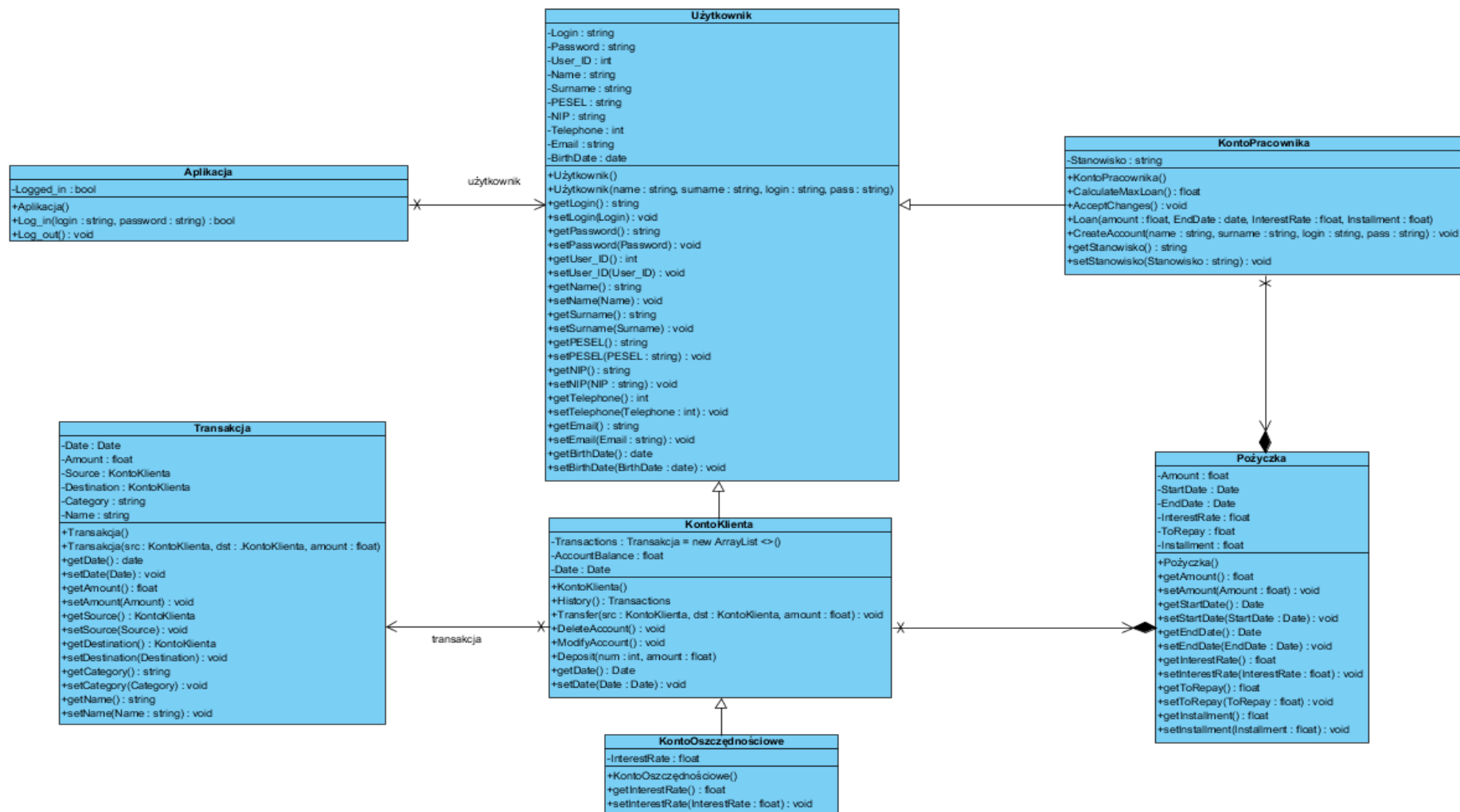
Dostępność do tabel

Tabela	Pracownik	Klient
Konto	Może zakładać i usuwać konta, a także ma do nich wgląd	Może dokonywać wpłat i wypłat na konto, do pozostałych kolumn ma dostęp read only
Transakcja	Ma wgląd do historii transakcji, a także może tworzyć transakcje	Może dokonywać operacji wpłaty, wypłaty i przelewu pomiędzy kontami
Konto oszczędnościowe	Może tworzyć i usuwać konta oszczędnościowe, a także ma do nich wgląd	Może dokonywać przelewów, a także ma wgląd do szczegółów konta
Pożyczka	Może tworzyć i usuwać pożyczki, a także ma do nich wgląd	Ma wgląd do szczegółów na temat pożyczki
Klient	Może tworzyć i usuwać klientów, a także ma do nich wgląd	Ma dostęp do swoich danych (do wglądu), plus może zmieniać dane
Adres	Może tworzyć i usuwać adresy, a także ma do nich wgląd	Ma wgląd do swoich adresów
Bank	Ma jedynie wgląd do danych o banku, pracownik będący dyrektorem banku może dodawać rekordy do tej tabeli, usuwać je i je	Nie ma dostępu do tej tabeli

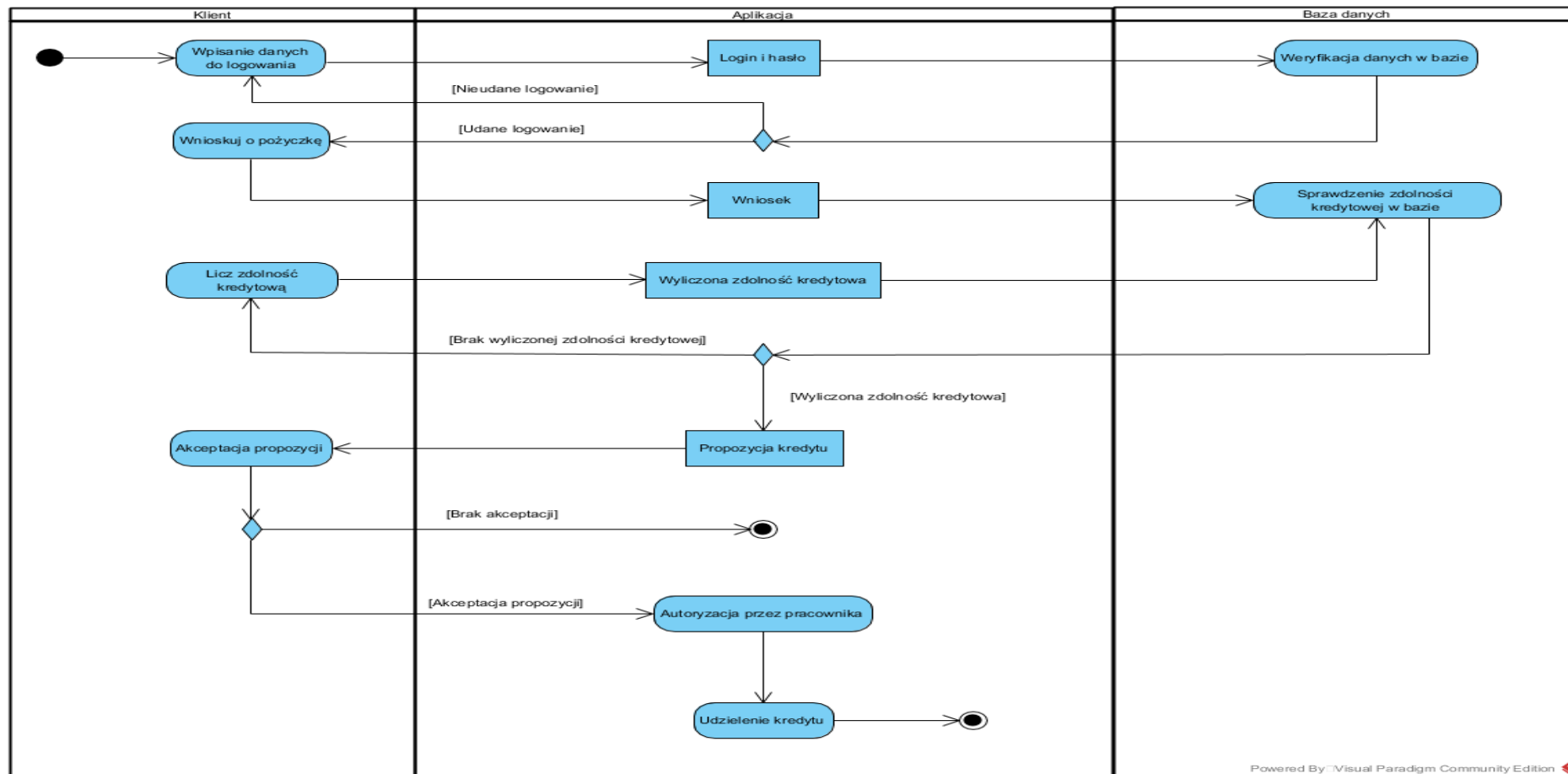
	zmieniać	
Pracownik	Ma jedynie wgląd do swoich danych, pracownik będący dyrektorem banku może dodawać rekordy do tej tabeli, usuwać je i je zmieniać	Nie ma dostępu do tej tabeli

3.2.1. Architektura aplikacji i diagramy projektowe

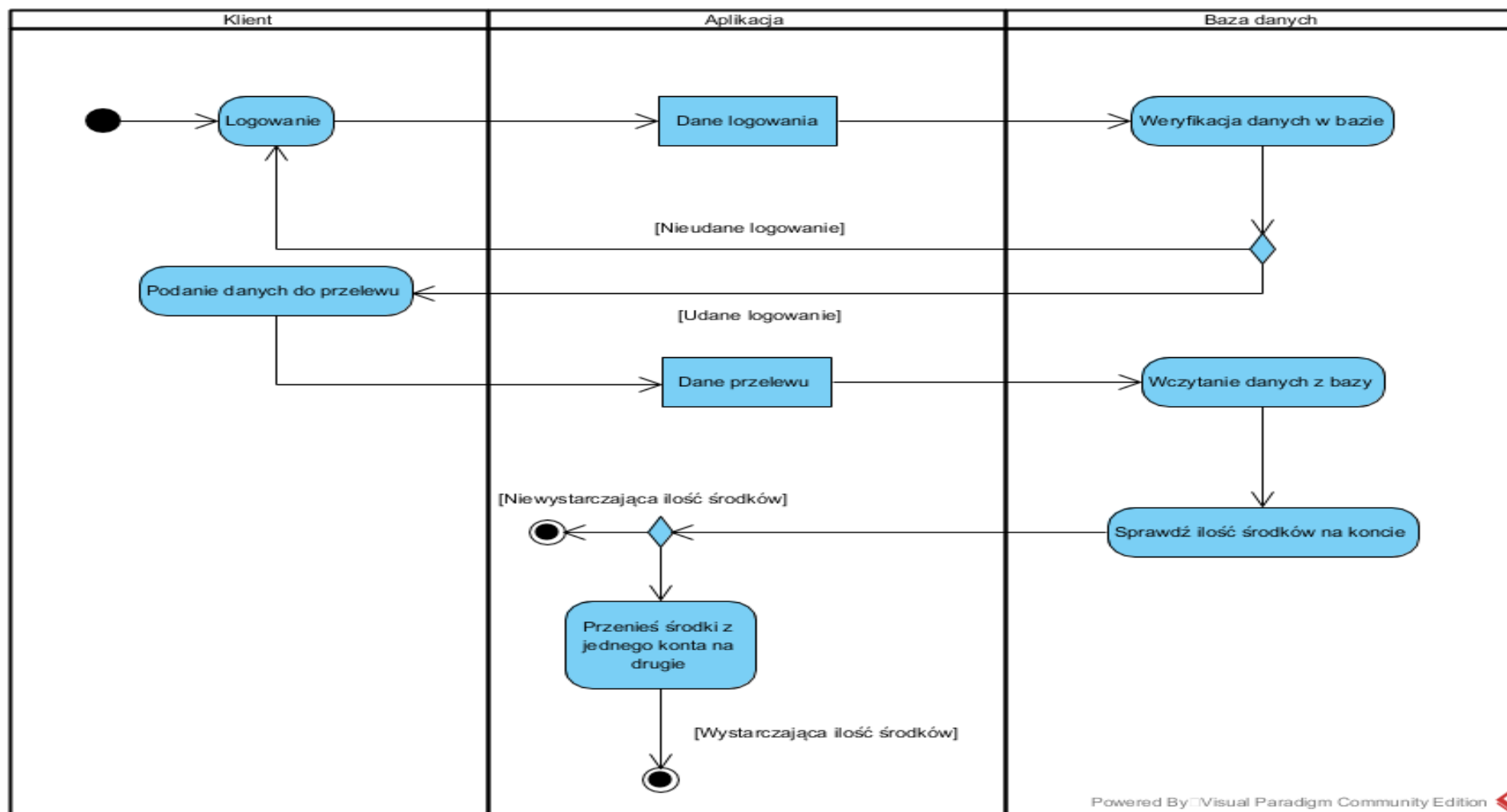




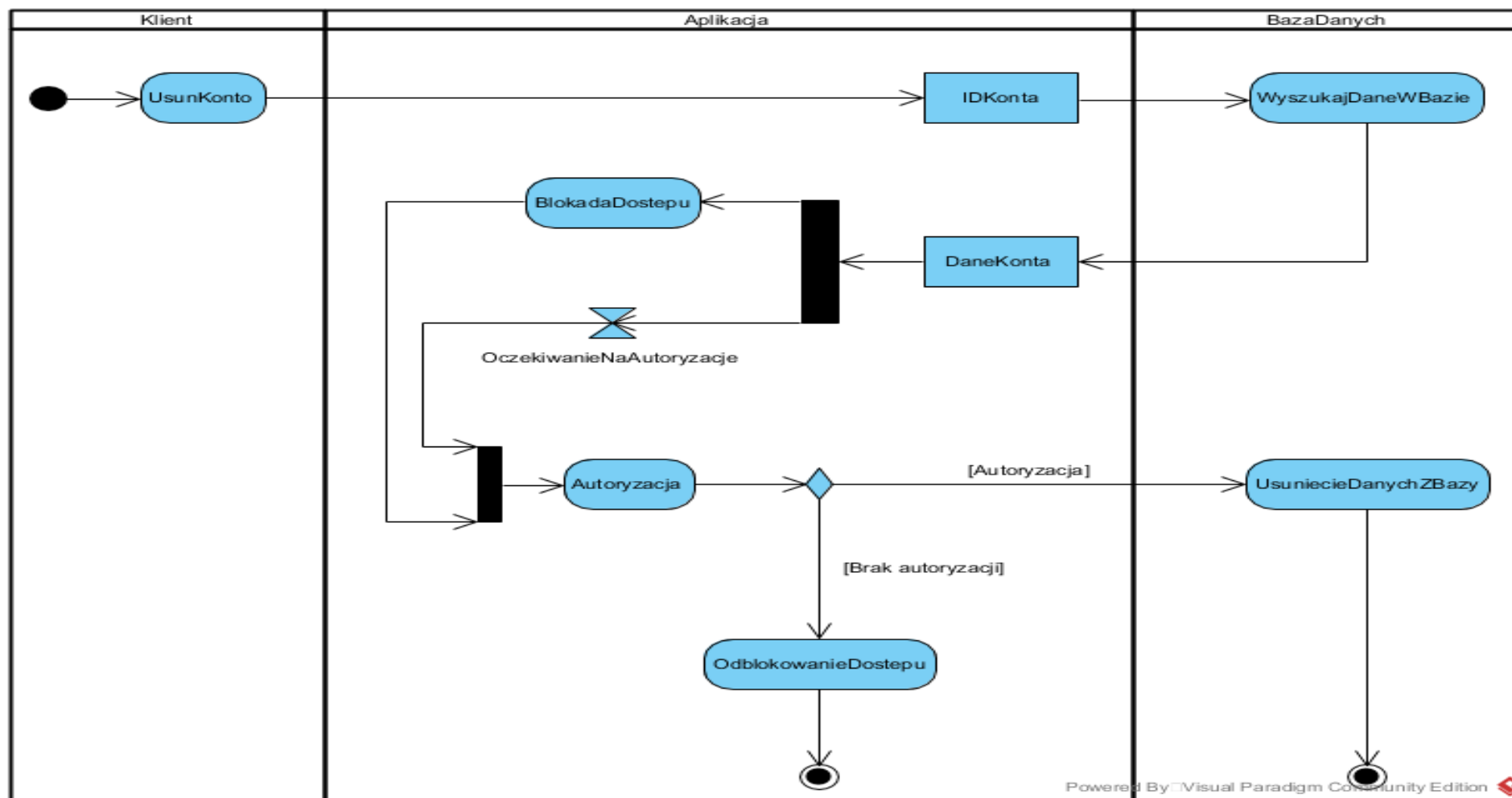
Rysunek 5 Diagram klas aplikacji dostępowej - Konto bankowe



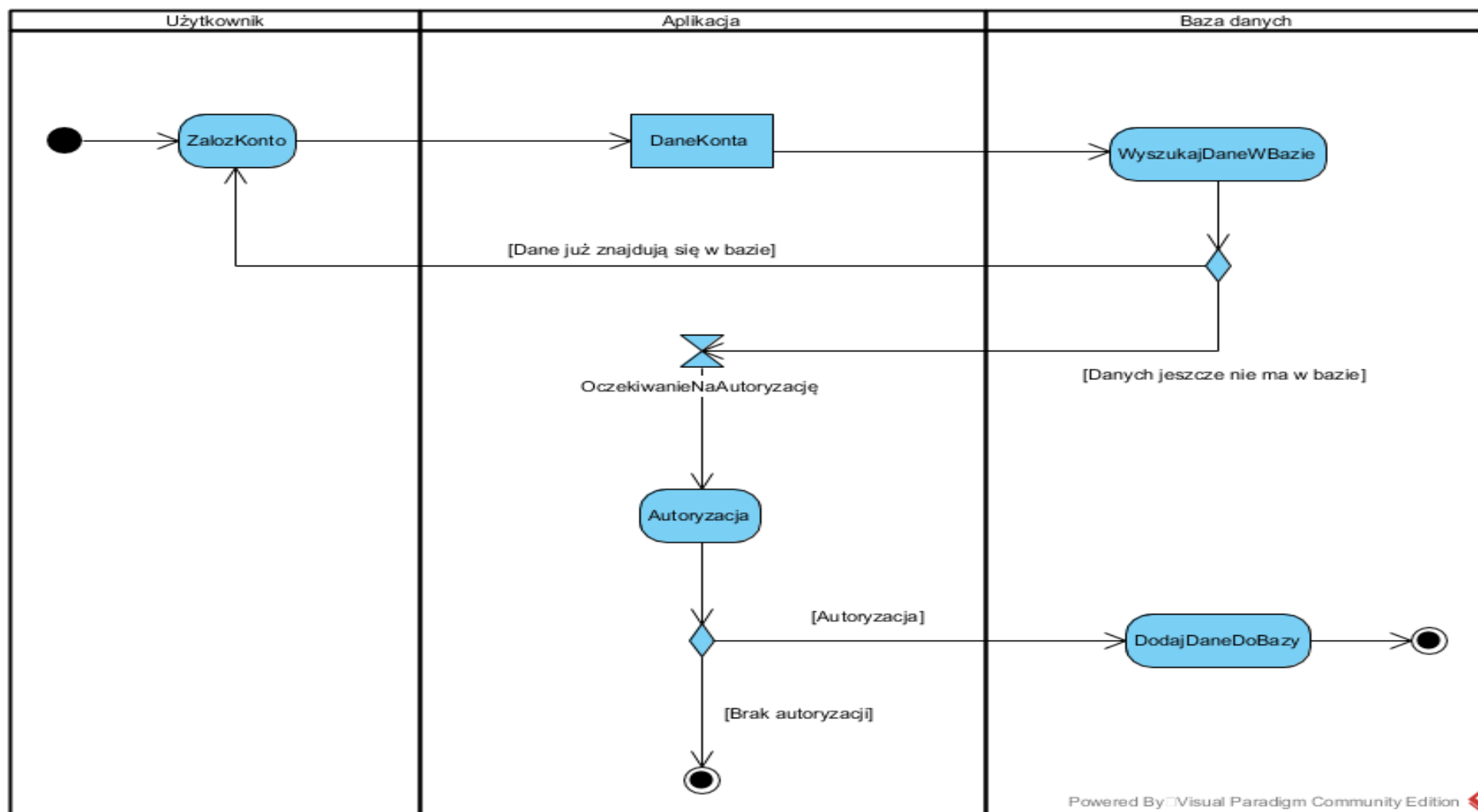
Rysunek 6 Diagram aktywności – PU Udzielenie pożyczki na wniosek klienta



Rysunek 7 Diagram aktywności - PU Wykonanie transakcji



Rysunek 8 Diagram aktywności - PU Usunięcie konta



Rysunek 9 Diagram aktywności - PU Założenie konta

3.2.2. Interfejs graficzny i struktura menu

Aplikacja bankowa

Interfejs wspólny dla pracownika oraz użytkownika

Login
jkwalski

Hasło

Niepoprawny login lub hasło!

Zaloguj

Komunikacja errora, jeśli nie powiedzie się logowanie

Interfejs pracownika

Aplikacja bankowa

Stwórz konto Akcja na istniejącym koncie Dodaj pracownika Dane placówki

Dane podstawowe	Adres zamieszkania	Dane kontaktowe
Imię	Ulica	Telefon *
Nazwisko	Numer domu	E-mail *
PESEL	Numer lokalu	
NIP *	Kod pocztowy	
Data urodzenia	Miejscowość	
Login		
Hasło		

Utwórz

Coś poszło nie tak, spróbuj ponownie

Przycisk nieaktywny do momentu, w którym wszystkie pola są wypełnione i są wypełnione poprawnie

Error, gdy nie udało się utworzyć konta

Aplikacja bankowa

Stwórz konto

Akcja na istniejącym koncie

Dodaj pracownika

Dane placówki

Wyszukaj konto

Imię

Nazwisko

PESEL

Opcja 3 i 4 dostępne tylko dla kierownika placówki

Komunikacja errora, jeśli nie powiedzie się wyszukanie

Nie znaleziono takiego konta!

Wyszukaj

Przycisk niekatywny do momentu, w którym wszystkie pola są wypełnione i są wypełnione poprawnie

Aplikacja bankowa

Stwórz konto

Akcja na istniejącym koncie

Dodaj pracownika

Dane placówki

Dane podstawowe

Imię

Nazwisko

Telefon*

E-mail*

Stanowisko

Login

Hasło

Przycisk niekatywny do momentu, w którym wszystkie pola są wypełnione i są wypełnione poprawnie

Error, gdy nie udało się dodać pracownika

Dodaj

Coś poszło nie tak, spróbuj ponownie

20

Aplikacja bankowa

Stwórz konto

Akcja na istniejącym koncie

Dodaj pracownika

Dane placówki

Dane podstawowe

Nazwa: Mój Bank
Telefon: 501 785 459
E - mail: moj.bank@poczta.pl
Właściciel : Jan Kowalski

Edytuj dane

Edycja danych powoduje,
że pola tekstowe
zamieniają się na typu
input, a przycisk "Edytuj
dane" zmienia się w
"Zapisz"

Aplikacja bankowa

Klient: Jan Kowalski Saldo: 15859.90 zł

Nowa transakcja

Historia

Dane konta

Pożyczki

Lokaty

Dane podstawowe

Adres zamieszkania

Dane kontaktowe

Imię: Jan
Nazwisko: Kowalski
PESEL: 90 11 26 33 567
NIP: b.d.
Data urodzenia: 26.11.1990

Ulica: Kwiatowa
Numer domu: 15
Numer lokalu: b.d.
Kod pocztowy: 55-123
Miejscowość: Wrocław

Telefon: 509 612 123
E-mail: jan.kowalski@poczta.pl

Numer konta: 25 0000 6985 7458 2361 1236 7852

Edytuj dane

Usuń konto

Edycja danych powoduje,
że pola tekstowe
zamieniają się na typu
input, a przycisk "Edytuj
dane" zmienia się w
"Zapisz"

Aplikacja bankowa

Klient: Jan Kowalski

Saldo: 15859.90 zł

Nowa transakcja

Historia

Dane konta

Pożyczki

Lokaty

Dane podstawowe

Adres zamieszkania

Dane kontaktowe

Imię: Jan

Nazwisko: Kowalski

PESEL: 90 11 26 33 567

NIP: b.d.

Data urodzenia: 26.11.1990

509 612 123

n.kowalski@poczta.pl

Czy na pewno chcesz usunąć konto?

Wszystkie dane przestaną być dostępne i konto stanie się nieaktywne.

Anuluj

Usuń

Numer konta: 25 0000 6985 7458 2361 1236 7852

Edytuj dane

Usuń konto

Po usunięciu konta następuje powrót do widoku głównego

Aplikacja bankowa

Klient: Jan Kowalski

Saldo: 15859.90 zł

Nowa transakcja

Historia

Dane konta

Pożyczki

Lokaty

Dane podstawowe

Przelew

Wyplata

Wplata

Na konto

Tytułem

0,00 zł

Kategoria

25/11/2018

Wykonaj

Coś poszło nie tak, spróbuj ponownie

Przycisk nieaktywny do momentu, w którym wszystkie pola są wypełnione i są wypełnione poprawnie

Error, gdy nie udało się dokonać transakcji

Aplikacja bankowa

Klient: Jan Kowalski

Saldo: 15859.90 zł

Nova transakcja

Historia

Dane konta

Pozyczki

Lokaty

Dane podstawowe

0,00

zł

25/11/2018

Wykonaj

Przelew

Wyplata

Wplata

Przycisk niekatywny do momentu, w którym wszystkie pola są wypełnione i są wypełnione poprawnie

Error, gdy nie udało się dokonać transakcji

Coś poszło nie tak, spróbuj ponownie

Aplikacja bankowa

Klient: Jan Kowalski

Saldo: 15859.90 zł

Nova transakcja

Historia

Dane konta

Pozyczki

Lokaty

Dane podstawowe

0,00

zł

25/11/2018

Wykonaj

Przelew

Wyplata

Wplata

Przycisk niekatywny do momentu, w którym wszystkie pola są wypełnione i są wypełnione poprawnie

Error, gdy nie udało się dokonać transakcji

Coś poszło nie tak, spróbuj ponownie

Aplikacja bankowa

Klient: Jan Kowalski

Saldo: 15859.90 zł

Nowa transakcja

Historia

Dane konta

Pożyczki

Lokaty

Wybierz okres

od: 1/11/2018

do: 25/11/2018

Wybierz kategorię

Kategoria

Filtruj

Szukaj po tytule

▼ Data	▼ Typ	▼ Na konto	▼ Tytułem	▼ Kategoria	▼ Kwota
25/11/2018	przelew	22 3451 2345 5678 1234 4321 9087	XYZ	Brak	125.00 zł
12/11/2018	wypłata	Brak	Brak	Brak	50.00 zł
03/11/2018	wpłata	Brak	Brak	Brak	1000.00 zł
25/10/2018	przelew	22 3451 0000 5678 1234 1111 9087	ABC	Transport	111.00 zł
23/04/2018	splata pożyczki	Brak	Brak	Brak	1312.50 zł

Aplikacja bankowa

Klient: Jan Kowalski

Saldo: 15859.90 zł

Nowa transakcja

Historia

Dane konta

Pożyczki

Lokaty

Udzielone pożyczki

▼ Data udzielenia	▼ Data spłaty	▼ Kwota	▼ Oprocentowanie	▼ Wysokość raty	▼ Pozostała część
25/11/2018	25/11/2019	15000.00 zł	5.00 %	1312.50 zł	15000.00 zł

Nowa pożyczka

Aplikacja bankowa

Klient: Jan KowalskiSaldo: 15859.90 zł

Nowa transakcja

Historia

Dane konta

Pożyczki

Lokaty

Utwórz pożyczkę

Kwota: 0,00 zł

Data spłaty 25/11/2018

Oprocentowanie: 0,00 %

Wysokość raty: 0,00 zł

Udziel

Coś poszło nie tak, spróbuj ponownie

Przycisk nieaktywny do momentu, w którym wszystkie pola są wypełnione i są wypełnione poprawnie

Error, gdy nie udało się udzielić pożyczki

Aplikacja bankowa

Klient: Jan KowalskiSaldo: 15859.90 zł

Nowa transakcja

Historia

Dane konta

Pożyczki

Lokaty

Posiadane lokaty

▼ Data utworzenia	▼ Kwota	▼ Oprocentowanie	▼ Obecna kwota
25/11/2018	100000.00 zł	1.00 %	100000.00 zł

Nowa lokata

Aplikacja bankowa

Klient: Jan Kowalski

Saldo: 15859.90 zł

Nowa transakcja

Historia

Dane konta

Pożyczki

Lokaty

Utwórz lokatę

Kwota: 0,00 zł

Oprocentowanie: 0,00 %

Otwórz

Coś poszło nie tak, spróbuj ponownie

Przycisk nieaktywny do momentu, w którym wszystkie pola są wypełnione i są wypełnione poprawnie

Error, gdy nie udało się otworzyć lokaty

Interfejs klienta

Aplikacja bankowa

Klient: Jan Kowalski

Saldo: 15859.90 zł

Nowa transakcja

Historia

Dane konta

Pożyczki

Lokaty

Dane podstawowe

Imię: Jan

Nazwisko: Kowalski

PESEL: 90 11 26 33 567

NIP: b.d.

Data urodzenia: 26.11.1990

Adres zamieszkania

Ulica: Kwiatowa

Numer domu: 15

Numer lokalu: b.d.

Kod pocztowy: 55-123

Miejscowość: Wrocław

Dane kontaktowe

Telefon: 509 612 123

E-mail: jan.kowalski@poczta.pl

Numer konta: 25 0000 6985 7458 2361 1236 7852

Edytuj dane

Edycja danych powoduje, że pola tekstowe zamieniają się na typu input, a przycisk "Edytuj dane" zmienia się w "Zapisz"

Aplikacja bankowa

Klient: Jan Kowalski

Saldo: 15859.90 zł

Nowa transakcja

Historia

Dane konta

Pożyczki

Lokaty

Wybierz okres

od: 1/11/2018

▼

do: 25/11/2018

▼

Wybierz kategorię

Kategoria ▼

Filtruj

Q Szukaj po tytule

▼ Data	▼ Typ	▼ Na konto	▼ Tytułem	▼ Kategoria	▼ Kwota
25/11/2018	przelew	22 3451 2345 5678 1234 4321 9087	XYZ	Brak	125.00 zł
12/11/2018	wypłata	Brak	Brak	Brak	50.00 zł
03/11/2018	wpłata	Brak	Brak	Brak	1000.00 zł
25/10/2018	przelew	22 3451 0000 5678 1234 1111 9087	ABC	Transport	111.00 zł
23/04/2018	splata pożyczki	Brak	Brak	Brak	1312.50 zł

Aplikacja bankowa

Klient: Jan Kowalski

Saldo: 15859.90 zł

Nowa transakcja

Historia

Dane konta

Pożyczki

Lokaty

Dane podstawowe

Na konto

Tytułem

0,00 zł

Kategoria ▼

25/11/2018 ▼

Wykonaj

Przycisk nieaktywny do momentu, w którym wszystkie pola są wypełnione i są wypełnione poprawnie

Error, gdy nie udało się dokonać transakcji

Coś poszło nie tak, spróbuj ponownie

27

Aplikacja bankowa

Klient: Jan Kowalski

Saldo: 15859.90 zł

Nowa transakcja

Historia

Dane konta

Pożyczki

Lokaty

Moje pożyczki

▼ Data udzielenia	▼ Data spłaty	▼ Kwota	▼ Oprocentowanie	▼ Wysokość raty	▼ Pozostała część
25/11/2018	25/11/2019	15000.00 zł	5.00 %	1312.50 zł	15000.00 zł

Aplikacja bankowa

Klient: Jan Kowalski

Saldo: 15859.90 zł

Nowa transakcja

Historia

Dane konta

Pożyczki

Lokaty

Posiadane lokaty

▼ Data utworzenia	▼ Kwota	▼ Oprocentowanie	▼ Obecna kwota
25/11/2018	100000.00 zł	1.00 %	100000.00 zł

3.2.3. Metoda podłączania do bazy danych – integracja z bazą danych

Połączenie z bazą danych będzie nawiązywane za pomocą sieci Internet z wykorzystaniem framework Retrofit 2.0.

3.2.4. Projekt zabezpieczeń na poziomie aplikacji

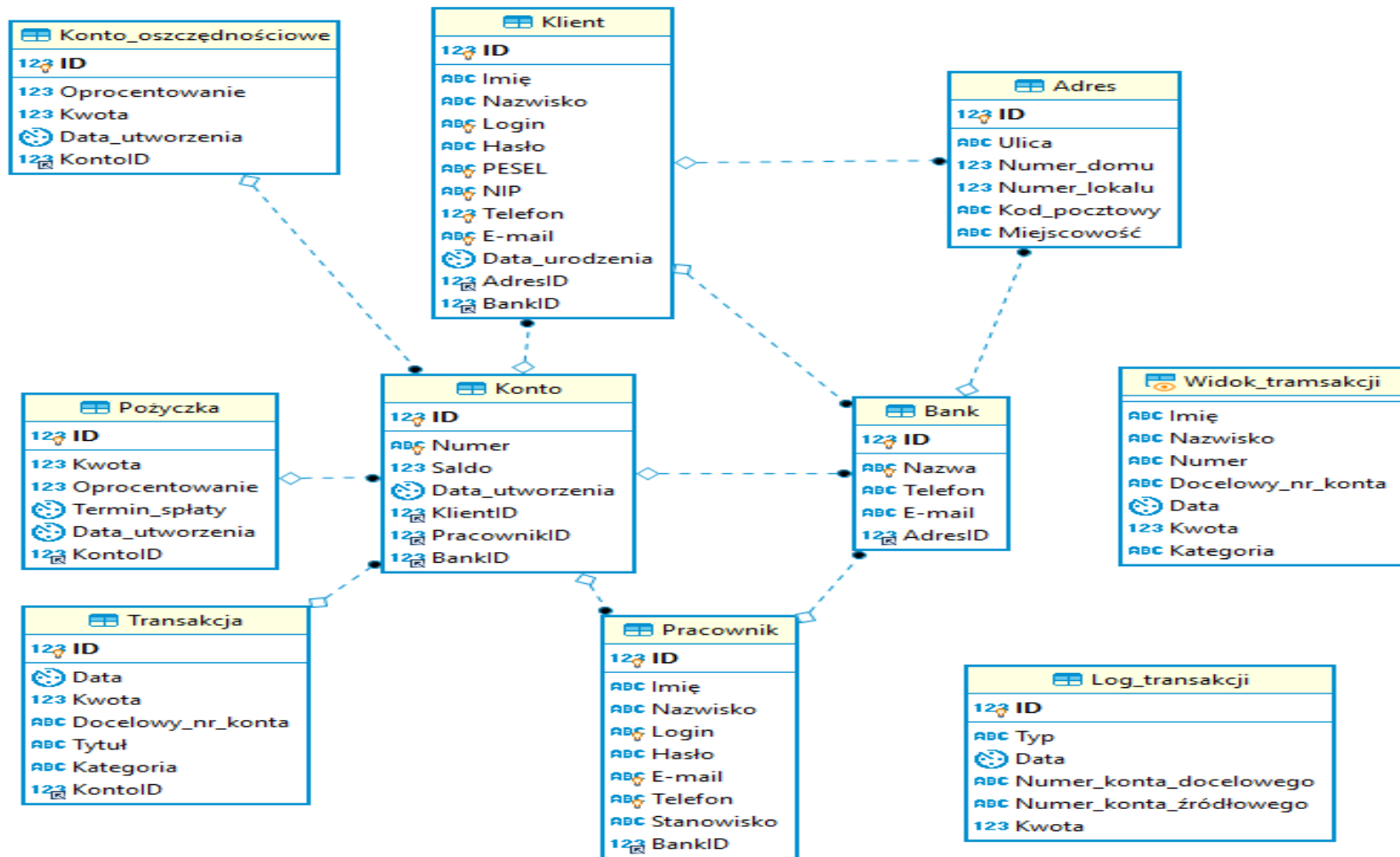
Dane przed wysłaniem do bazy danych zostaną sprawdzone pod kątem zgodności typu. Interfejs w zależności od poziomu dostępu się różni, dzięki czemu użytkownicy mają dostęp do przewidzianych dla siebie funkcji. Logowanie odbywa się za pomocą formularza, gdzie hasło wpisywane jest w trybie zakrytym.

4. Implementacja systemu baz danych

4.1. Tworzenie tabel i definiowanie ograniczeń

Baza powstała w systemie model first, co oznacza, że najpierw został wprowadzony do programu My SQL Workbench zaprojektowany model bazy danych, a następnie został stworzony skrypt tworzący uzupełniony o dodatkowe elementy (jak triggery czy procedury).

Po implementacji systemu bazodanowego okazało się, że wymagane są drobne zmiany modelu, zatem ostateczny skrypt tworzący odnosił się do ostatecznej wersji. Poniżej zaprezentowany zaktualizowany model bazy danych.



Rysunek 10 Zweryfikowany model bazy danych

Poniższy fragment kodu prezentuje tworzenie tabeli klientów. Tworzone są poszczególne kolumny oraz nadawane są na nie ograniczenia (np. NOT NULL, czyli które pola są obowiązkowe). Ponad to tworzone są indeksy, które potem ułatwią wyszukiwanie.

```
-- -----  
-- Table `BankAccountDatabase`.`Klient`  
-- -----  
DROP TABLE IF EXISTS `BankAccountDatabase`.`Klient` ;  
  
CREATE TABLE IF NOT EXISTS `BankAccountDatabase`.`Klient` (  
  `ID` INT(11) NOT NULL AUTO_INCREMENT,  
  `Imię` VARCHAR(255) NOT NULL,  
  `Nazwisko` VARCHAR(255) NOT NULL,  
  `Login` VARCHAR(40) NOT NULL,  
  `Hasło` VARCHAR(255) NOT NULL,  
  `PESEL` CHAR(11) NOT NULL,  
  `NIP` CHAR(10) NULL DEFAULT NULL,  
  `Telefon` INT(12) NOT NULL,  
  `E-mail` VARCHAR(255) NOT NULL,  
  `Data_urodzenia` DATE NOT NULL,  
  `AdresID` INT(11) NOT NULL,  
  `BankID` INT(11) NOT NULL,  
  PRIMARY KEY (`ID`),  
  CONSTRAINT `fk_Klient_Adres1`  
    FOREIGN KEY (`AdresID`)  
    REFERENCES `BankAccountDatabase`.`Adres` (`ID`)  
    ON DELETE NO ACTION  
    ON UPDATE NO ACTION,  
  CONSTRAINT `fk_Klient_Bank1`  
    FOREIGN KEY (`BankID`)  
    REFERENCES `BankAccountDatabase`.`Bank` (`ID`)  
    ON DELETE NO ACTION  
    ON UPDATE NO ACTION)  
ENGINE = InnoDB  
AUTO_INCREMENT = 4  
DEFAULT CHARACTER SET = utf8;  
  
CREATE UNIQUE INDEX `PESEL_UNIQUE` ON `BankAccountDatabase`.`Klient` (`PESEL`  
ASC);  
  
CREATE UNIQUE INDEX `Telefon_UNIQUE` ON `BankAccountDatabase`.`Klient` (`Telefon`  
ASC);  
  
CREATE UNIQUE INDEX `E-mail_UNIQUE` ON `BankAccountDatabase`.`Klient` (`E-mail`  
ASC);  
  
CREATE UNIQUE INDEX `Login_UNIQUE` ON `BankAccountDatabase`.`Klient` (`Login`  
ASC);
```

```
CREATE UNIQUE INDEX `ID_UNIQUE` ON `BankAccountDatabase`.`Klient` (`ID` ASC);

CREATE UNIQUE INDEX `NIP_UNIQUE` ON `BankAccountDatabase`.`Klient` (`NIP` ASC);

CREATE INDEX `fk_Klient_Adres1_idx` ON `BankAccountDatabase`.`Klient` (`AdresID`
ASC);

CREATE INDEX `fk_Klient_Bank1_idx` ON `BankAccountDatabase`.`Klient` (`BankID`
ASC);
```

4.2. Implementacja mechanizmów przetwarzania danych

Komunikacja z bazą danych odbywa się za pomocą serwera z wystawionym REST API. Część mechanizmów jest zawarta w procedurach stworzonych po stronie bazy danych. Niektóre zmiany są realizowane przez triggery, wyzwalane na zaistniałe zmiany w bazie danych (np. trigger logujący oraz trigger na spłatę kredytu). Doliczenie lokaty po roku odbywa się wewnątrz Cron-taska, który uruchamia się codziennie o północy i aktualizuje wymagane pola.

Poniżej prezentowany jest kod triggera, który uruchamia się na akcję stworzenia nowego rekordu o kategorii `Spłata_kredytu` w tabeli `Transakcje`. Jest to trigger, który uruchamia się po akcji dodania rekordu i zmniejsza kwotę pożyczki o wartość spłaty.

```
USE `BankAccountDatabase` $$
DROP TRIGGER IF EXISTS `BankAccountDatabase`.`Transakcja_AFTER_INSERT` $$
USE `BankAccountDatabase` $$
CREATE
TRIGGER `BankAccountDatabase`.`Transakcja_AFTER_INSERT`
AFTER INSERT ON `BankAccountDatabase`.`Transakcja`
FOR EACH ROW
BEGIN
    IF NEW.Kategoria = 'Spłata_kredytu' THEN
        UPDATE Pożyczka SET Kwota = Kwota - NEW.Kwota WHERE KontoID =
NEW.KontoID;
    END IF;
END $$
```

Procedury pozwalają na wykonywanie nawet kilku działań wewnątrz bazy danych za pomocą pojedynczego wywołania z kodu po stronie serwera. Poniżej kilka spośród zaimplementowanych procedur.

Procedura odpowiadająca za powiększenie salda na koncie, na które udzielana jest pożyczka:

```
-- -----
-- procedure nowaPozyczka
```



```

-- -----
USE `BankAccountDatabase`;
DROP procedure IF EXISTS `BankAccountDatabase`.`nowaPozyczka`;

DELIMITER $$
USE `BankAccountDatabase`$$
CREATE PROCEDURE `nowaPozyczka`(IN kwota INT, IN kontoId INT)
UPDATE Konto SET Saldo = Saldo + kwota WHERE id = kontoId$$

DELIMITER ;

```

Procedura odpowiadająca za aktualizację salda na koncie z którego wykonywany jest przelew, oraz na koncie, na który przelew trafia:

```

-- -----
-- procedure przelew
-- -----

USE `BankAccountDatabase`;
DROP procedure IF EXISTS `BankAccountDatabase`.`przelew`;

DELIMITER $$
USE `BankAccountDatabase`$$
CREATE PROCEDURE `przelew`(IN kwota INT, IN kontoIdSource INT, IN
kontoIdTarget INT)
BEGIN
    UPDATE Konto SET Saldo = Saldo + kwota WHERE id = kontoIdTarget;
    UPDATE Konto SET Saldo = Saldo - kwota WHERE id = kontoIdSource;
END$$

DELIMITER ;

```

Procedura odpowiadająca za aktualizację salda na koncie oszczędnościowym w momencie, gdy mija rok od założenia lokaty:

```

-- -----
-- procedure przeliczLokaty
-- -----

USE `BankAccountDatabase`;
DROP procedure IF EXISTS `BankAccountDatabase`.`przeliczLokaty`;

DELIMITER $$
USE `BankAccountDatabase`$$
CREATE PROCEDURE `przeliczLokaty`()

```

```

UPDATE Konto_oszczędnościowe k SET Kwota = Kwota + (Oprocentowanie/100 *
Kwota) WHERE timestampdiff(YEAR, k.Data_utworzenia, NOW()) > 0 AND
MONTH(k.Data_utworzenia) = MONTH(NOW()) AND DAY(k.Data_utworzenia) = DAY(NOW())$$

DELIMITER ;

```

Część mechanizmów realizowana jest przez REST API. Poniżej kilka ciekawszych metod.

Metoda odpowiedzialna za wykonywanie przelewu przez klienta. Widoczne jest wywołanie procedury odpowiedzialnej za aktualizację odpowiednich pól w bazie danych.

```

router.post('/transakcje/przelew', authenticate, checkKlientPermission,
(req, res, next) => {
  return konto.findOne({
    where: {
      klientId: req.user.id
    }
  }).then(kontoObj => {
    return transakcja.create({
      docelowyNrKonta: req.body.docelowyNrKonta,
      tytul: req.body.tytul,
      kwota: req.body.kwota,
      kategoria: 'Przelew',
      data: req.body.data,
      kontoId: kontoObj.id
    }).then(traObj => {
      return konto.findOne({
        where: {
          numer: req.body.docelowyNrKonta
        }
      }).then(kontoTarget => {
        return sequelize.query(`CALL przelew(${req.body.kwota},
${kontoObj.id}, ${kontoTarget.id})`).then(x => {
          return res.json({
            success: 1
          })
        })
      })
    }).catch(err => {
      console.log(err)
      return res.json({
        success: 0
      })
    })
  })
})

```

Metoda odpowiedzialna za wyszukanie klienta przez pracownika. Zwracany jest JSON z danymi wyszukiwanego klienta.

```
router.get('/konto/wyszukaj/klient', authenticate, checkPracownikPermission, (req, res, next) => {
  return klient.findOne({
    where: {
      imie: req.query.imie,
      nazwisko: req.query.nazwisko,
      pesel: req.query.pesel,
      bankId: req.user.bankId
    },
    include: [{
      model: konto
    },
    {
      model: adres
    }],
    subQuery: false
  }).then(klientObj => {
    return res.json(klientObj)
  })
})
```

Metoda odpowiedzialna za edycję profilu przez klienta.

```
router.patch('/profil', authenticate, checkKlientPermission, (req, res, next) => {
  return klient.findOne({
    where: {
      id: req.user.id
    }
  }).then(profil => {
    return klient.update({
      imie: req.body.imie,
      nazwisko: req.body.nazwisko,
      telefon: req.body.telefon,
      email: req.body.email
    }, {
      where: {
        id: req.user.id
      }
    })
  }).then(obj => {
    return adres.update({
      ulica: req.body.ulica,
      numerDomu: req.body.numerDomu,
      numerLokalu: req.body.numerLokalu,
      kodPocztowy: req.body.kodPocztowy,
      miejscowosc: req.body.miejscowosc
    }, {
      where: {

```

```

        id: profil.adresId
      }
    }).then(obj => {
      return res.json({
        success: 1
      })
    }).catch(err => {
      console.log(err)
      return res.json({
        success: 0
      })
    })
  })
  }).catch(err => {
    console.log(err)
    return res.json({
      success: 0
    })
  })
})
})
})

```

4.3. Implementacja uprawnień i innych zabezpieczeń

Autoryzacja użytkownika opiera się o mechanizm tokenów. W tym celu wykorzystano tzw. Tokeny JWT, które są zgodne ze standardem RFC 7519. W headerze wysyłany jest token JWT, na podstawie którego autoryzowany jest użytkownik. Zarówno pracownicy, jak i klienci mają odpowiadający swoim uprawnieniom zestaw endpointów do komunikacji z bazą danych.

Token generowany jest na podstawie danych użytkownika oraz *secret*.

```

const generateJwt = (user) => {
  return jwt.sign({
    id: user.id,
    login: user.login,
    email: user.email,
    imie: user.imie,
    nazwisko: user.nazwisko
  }, secret)
}

```

Przykładowy token dla klienta:

```

"authToken":
"eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJpZCI6NCwibG9naW4iOiJtYWdkYWxlbmEiLCJlbWFPbCI6Im1hZ2RhbGVuYU9nbWVpYyB5b20iLCJpbWl1IjoiTWFnZGFsZW5hIiwibmF6d2lza28iOiJPbGNvYXdhIiwiaWF0IjoxNTQ2ODE0NDM1fQ.sjli40Vbf_lwNz1IG7u5E12hz3gqmU1jorVunNxcrR8"

```

Po zdekodowaniu tokena na stronie JWT mamy widoczne podstawowe dane o użytkowniku, zgodnie z definicją generowania tokena.

The screenshot shows the JWT Debugger interface. At the top, there's a navigation bar with links: Debugger, Libraries, Introduction, Ask, Get a T-shirt!, and a note 'Crafted by Auth0'. The main heading is 'Debugger'. A warning message states: 'Warning: JWTs are credentials, which can grant access to resources. Be careful where you paste them! We do not record tokens, all validation and debugging is done on the client side.' Below this, there's a dropdown menu for 'ALGORITHM' set to 'HS256'. The interface is split into two main sections: 'Encoded' and 'Decoded'. The 'Encoded' section has a text area with a long JWT token. The 'Decoded' section shows the token's structure: 'HEADER: ALGORITHM & TOKEN TYPE' with a JSON object {'alg': 'HS256', 'typ': 'JWT'}, 'PAYLOAD: DATA' with a JSON object containing user information, and 'VERIFY SIGNATURE' with a function call to HMACSHA256. A 'SHARE JWT' button is at the bottom right.

Debugger

Warning: JWTs are credentials, which can grant access to resources. Be careful where you paste them! We do not record tokens, all validation and debugging is done on the client side.

ALGORITHM HS256

Encoded PASTE A TOKEN HERE

Decoded EDIT THE PAYLOAD AND SECRET

HEADER: ALGORITHM & TOKEN TYPE

```
{
  "alg": "HS256",
  "typ": "JWT"
}
```

PAYLOAD: DATA

```
{
  "id": 4,
  "login": "magdalena",
  "email": "magdalena@gmail.com",
  "imie": "Magdalena",
  "nazwisko": "Olchawa",
  "iat": 1546814435
}
```

VERIFY SIGNATURE

```
HMACSHA256(
  base64UrlEncode(header) + "." +
  base64UrlEncode(payload),
  your-256-bit-secret
) ☐ secret base64 encoded
```

SHARE JWT

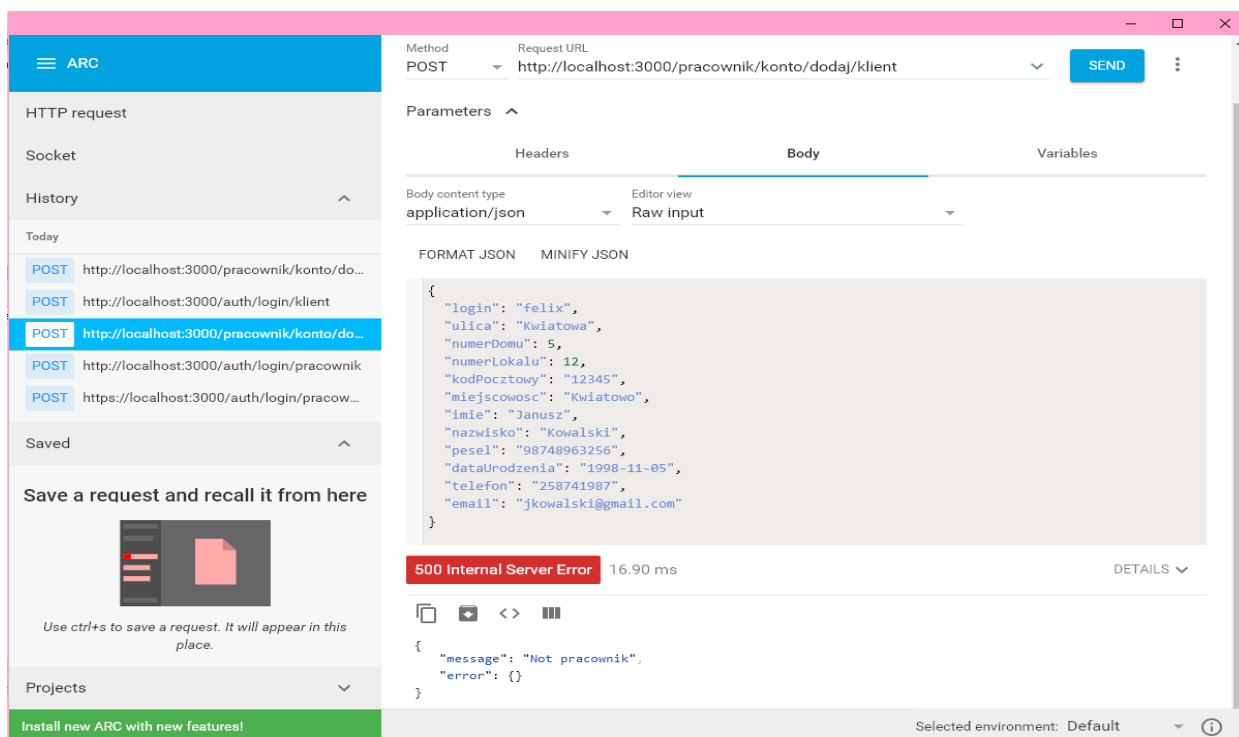
Rysunek 11 Dekodowanie tokena autoryzacyjnego na stronie biblioteki JWT

Ponadto, hasło jest także szyfrowane SH-256 z *secret*:

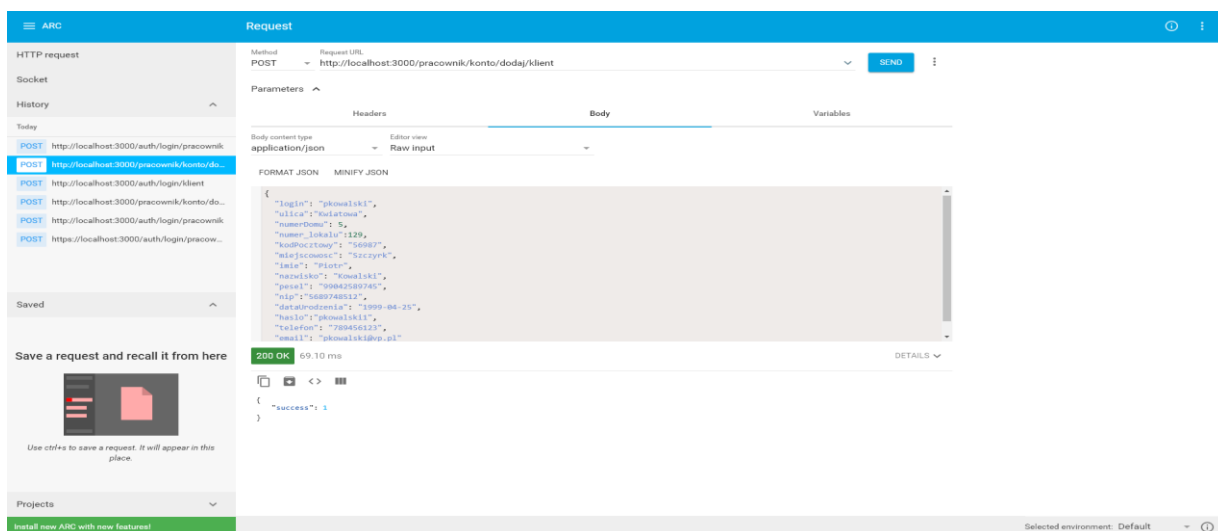
```
const encryptPwd = (pwd) => {
  return crypto.createHash('sha256').update(secret + '|' + pwd, 'utf-8').digest('hex')
}
```

4.4. Testowanie bazy danych na przykładowych danych

Testowanie autoryzacji na przykładzie dodawania nowego użytkownika.



Rysunek 12 Próba dodania klienta z wykorzystaniem tokena klienta, nie pracownika



Rysunek 13 Dodanie klienta - powodzenie

Testowanie procedury podczas wykonywania przelewu.

	ID	Numer	Saldo	Data_utworzenia	KlientID	PracownikID	BankID
▶	3	69266901048092719099627509	899.55	2019-01-06 21:24:49	4	3	2
	4	10168131743090895857963562	59.36	2019-01-07 00:09:47	5	3	2
	5	59232054616642479075505237	14.99	2019-01-07 00:26:46	6	3	2
	6	59527896292333283617980977	1269	2019-01-07 00:36:15	8	3	2
	7	76405765579360353731445464	125.25	2019-01-07 00:40:19	9	3	2
	8	22405712579360358931445464	100.45	2019-01-07 00:40:19	4	3	2

Rysunek 14 Stan bazy danych przed wykonaniem przelewu

The screenshot shows a REST client interface with a sidebar on the left containing a 'History' list of requests. The main area displays a 'Request' tab for a POST method to the URL 'http://localhost:3000/klient/transakcje/przelew'. The 'Body' tab is active, showing a JSON payload:

```
{
  "docelowylekanta": "22405712579360358931445464",
  "tytul": "Zarzut za sanki",
  "kwota": 12.99,
  "data": "2019-01-06"
}
```

. The response status is '200 OK' with a response time of '33.70 ms'. The response body shows

```
{
  "success": 1
}
```

. A green banner at the bottom indicates 'Install new ARC with new features!'.

Rysunek 15 Wykonanie przelewu

	ID	Numer	Saldo	Data_utworzenia	KlientID	PracownikID	BankID
▶	3	69266901048092719099627509	886.56	2019-01-06 21:24:49	4	3	2
	4	10168131743090895857963562	59.36	2019-01-07 00:09:47	5	3	2
	5	59232054616642479075505237	14.99	2019-01-07 00:26:46	6	3	2
	6	59527896292333283617980977	1269	2019-01-07 00:36:15	8	3	2
	7	76405765579360353731445464	125.25	2019-01-07 00:40:19	9	3	2
	8	22405712579360358931445464	113.44	2019-01-07 00:40:19	4	3	2

Rysunek 16 Stan bazy danych po wykonaniu przelewu

Testowanie spłaty pożyczki (trigger) oraz select na bazie danych.

```
1 SELECT pożyczka.KontoID, pożyczka.Kwota, konto.Saldo
2 FROM pożyczka
3 INNER JOIN konto ON pożyczka.KontoID=konto.ID;
4
```

<

Result Grid | Filter Rows: | Export: | Wrap Cell Content: |

	KontoID	Kwota	Saldo
▶	4	850	1009.36

Rysunek 17 Stan bazy danych przed transakcją będącą spłatą pożyczki

```
INSERT INTO `bankaccountdatabase`.`transakcja` (`ID`, `Data`, `Kwota`, `Tytuł`,  
`Kategoria`, `KontoID`)  
VALUES ('12', '2018-01-06', '50', 'Spłata 4', 'Spłata_kredytu', '4');
```

```
1 SELECT pożyczka.KontoID, pożyczka.Kwota, konto.Saldo
2 FROM pożyczka
3 INNER JOIN konto ON pożyczka.KontoID=konto.ID;
```

<

Result Grid | Filter Rows: | Export: | Wrap Cell Content: |

	KontoID	Kwota	Saldo
▶	4	800	959.36

Rysunek 18 Stan bazy danych po dokonaniu częściowej spłaty pożyczki

5. Implementacja i testy aplikacji

5.1. Instalacja i konfigurowanie systemu

Do działania naszego systemu będzie wymagane środowisko Java w wersji 8 do obsługi klienta/pracownika oraz aplikacja PM2 do działania serwera.

5.2. Instrukcja użytkowania aplikacji

Po uruchomieniu aplikacji pierwszym dostępnym oknem jest ekran logowania (rysunek 19), w nim należy podać swój login oraz pasujące do niego hasło (dane te muszą znajdować się w naszej bazie) i wybrać odpowiednio czy logujemy się jako klient czy jako pracownik od czego będą zależały kolejne kroki. Jako pracownik mamy dostęp do: Tworzenia nowych kont klientom oraz pracownikom (rysunki 21 i 22), przeglądania danych placówki do której jest przypisany zalogowany pracownik (rysunek 20) ponadto wyszukania istniejącego już konta klienta i wykonywania operacji na nim (rysunek 23). Gdy zalogujemy się jako klient mamy dostęp do przeglądania danych swojego konta (rysunek 24), wykonywania przelewów (rysunek 28), przeglądania historii wykonywanych przelewów (rysunek 25) oraz aktualnie zaciągniętych pożyczek i posiadanych lokat (rysunki 26 i 27).

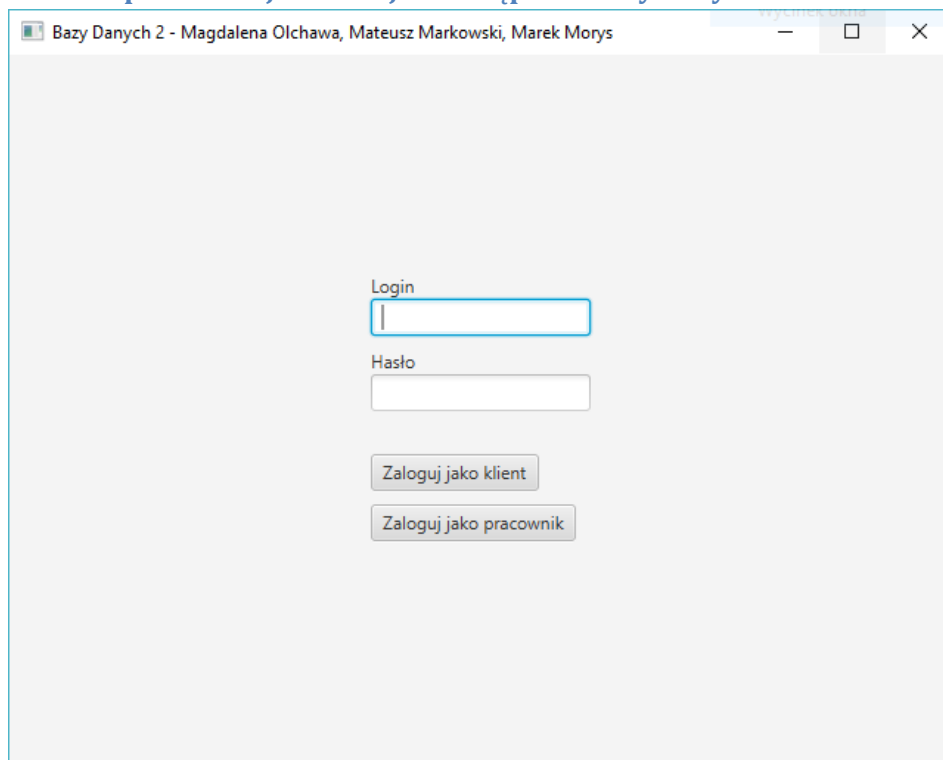
5.3. Testowanie opracowanych funkcji systemu

Testy bazy danych były wykonywane manualnie w MySQL Workbench oraz przez aplikację Postman poprzez wprowadzanie odpowiednich wyrażeń języka SQL. Testy aplikacji oraz połączenia jej z bazą danych również były wykonywane manualnie poprzez debugowanie w IntelliJ, przeglądanie logów serwera i aktualnego stanu bazy danych po wprowadzonych zmianach.

5.4. Omówienie wybranych rozwiązań programistycznych

W naszym projekcie wykorzystaliśmy technologie: MySQL, Node.js, Java w tym użyte biblioteki: JavaFX 8.0.181, retrofit-2.5.0, adapter-rxjava2-2.5.0, annotations-13.0, converter-gson-2.5.0, gson-2.8.5, kotlin-stdlib-1.3.11, kotlin-stdlib-common-1.3.11, okhttp-3.12.1, okio-2.2.1, reactive-streams-1.0.2, rxjava-2.2.5.

5.4.1. Implementacja interfejsu dostępu do bazy danych



Bazy Danych 2 - Magdalena Olchawa, Mateusz Markowski, Marek Morys

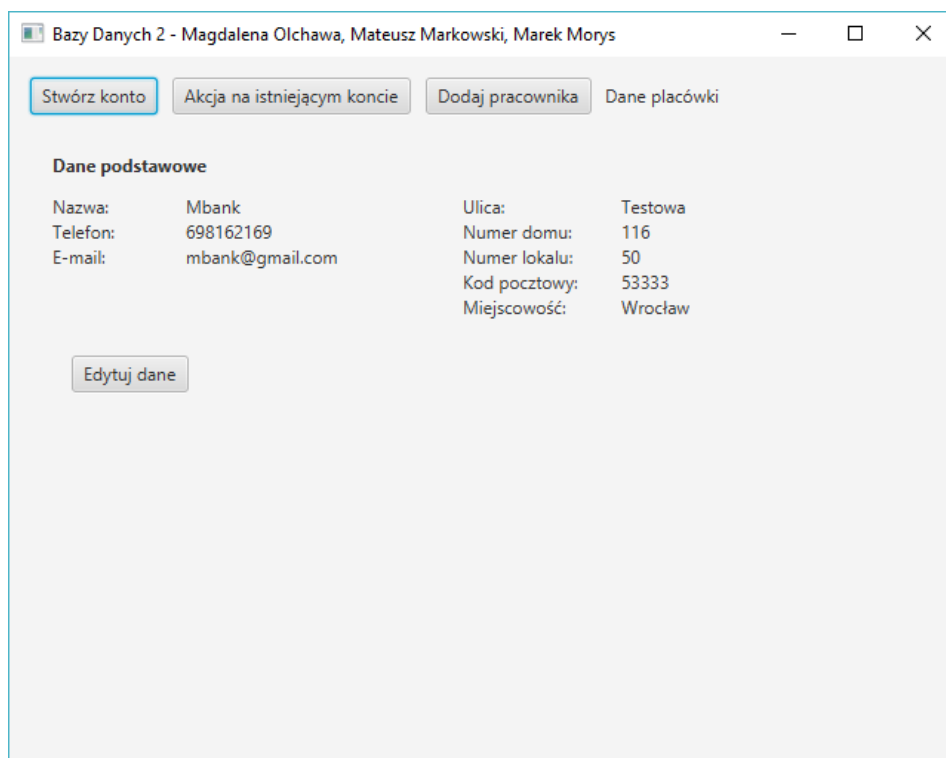
Login

Hasło

Zaloguj jako klient

Zaloguj jako pracownik

Rysunek 19 Ekran logowania



Bazy Danych 2 - Magdalena Olchawa, Mateusz Markowski, Marek Morys

Stwórz konto Akcja na istniejącym koncie Dodaj pracownika Dane placówki

Dane podstawowe

Nazwa:	Mbank	Ulica:	Testowa
Telefon:	698162169	Numer domu:	116
E-mail:	mbank@gmail.com	Numer lokalu:	50
		Kod pocztowy:	53333
		Miejscowość:	Wrocław

Edytuj dane

Rysunek 20 Ekran dane placówki

Bazy Danych 2 - Magdalena Olchawa, Mateusz Markowski, Marek Morys

Stwórz konto **Akcja na istniejącym koncie** Dodaj pracownika Dane placówki

Dane podstawowe	Adres zamieszkania	Dane kontaktowe
Imię	Ulica	Telefon
Nazwisko	Numer domu	E-mail
PESEL	Numer lokalu	
NIP	Kod pocztowy	
Data urodzenia	Miejscowość	
Login		
Hasło		

Utwórz

Rysunek 21 Ekran stwórz konto

Bazy Danych 2 - Magdalena Olchawa, Mateusz Markowski, Marek Morys

Stwórz konto Akcja na istniejącym koncie Dodaj pracownika Dane placówki

Dane podstawowe

Imię
Nazwisko
Telefon
E-mail
Stanowisko
Login
Hasło

Dodaj

Rysunek 22 Ekran dodaj pracownika

Bazy Danych 2 - Magdalena Olchawa, Mateusz Markowski, Marek Morys

Stwórz konto Akcja na istniejącym koncie Dodaj pracownika Dane placówki

Wyszukaj konto

Imię

Nazwisko

PESEL

Wyszukaj

Rysunek 23 Ekran akcja na koncie

Bazy Danych 2 - Magdalena Olchawa, Mateusz Markowski, Marek Morys

Klient: Marek Morys Saldo: 1700.77

Nowa transakcja Historia Dane konta Pożyczki Lokaty

Dane podstawowe		Adres zamieszkania		Dane kontaktowe	
Imię:	Marek	Ulica:	Morysowa	Telefon:	123123123
Nazwisko:	Morys	Numer domu:	11	E-mail:	morys@mor.com
PESEL:	9998877766	Numer lokalu:	22		
NIP:	124	Kod pocztowy:	33444		
Data urodzenia:	1997-01-01	Miejscowość:	Morysowo		

Numer konta: 38358802318133294798691190

Edytuj dane

Rysunek 24 Ekran dane konta

Bazy Danych 2 - Magdalena Olchawa, Mateusz Markowski, Marek Morys

Klient: Marek Morys Saldo: 1700.77

Nowa transakcja Historia Dane konta Pożyczki Lokaty

Wybierz okres

od: do: Wybierz kategorię:

Filtruj

Szukaj po tytule

Rysunek 25 Ekran historia transakcji

Bazy Danych 2 - Magdalena Olchawa, Mateusz Markowski, Marek Morys

Klient: Marek Morys Saldo: 1700.77

Nowa transakcja Historia Dane konta Pożyczki Lokaty

Posiadane lokaty

Rysunek 26 Ekran lokaty

Bazy Danych 2 - Magdalena Olchawa, Mateusz Markowski, Marek Morys

Klient: Marek Morys Saldo: 1700.77

Moje pożyczki

Rysunek 27 Ekran pożyczki

Bazy Danych 2 - Magdalena Olchawa, Mateusz Markowski, Marek Morys

Klient: Marek Morys Saldo: 1700.77

Dane podstawowe

zł

Rysunek 28 Ekran nowa transakcja

5.4.2. Implementacja wybranych funkcjonalności systemu

W systemie zostało zaimplementowane działanie wszystkich zakładanych przez nas funkcji, lecz w aplikacji zostały dodane wyłącznie następujące funkcje: logowanie do systemu przez pracowników i klientów, dodawanie nowych pracowników i klientów, wyświetlanie danych placówki dla aktywnego pracownika, wyszukanie klientów, wykonywanie przelewów oraz wyświetlanie danych zalogowanego użytkownika.

5.4.3. Implementacja mechanizmów bezpieczeństwa

System korzysta z uwierzytelniania podczas logowania za pomocą loginu oraz hasła, a także z autoryzacji poprzez różne poziomy dostępu informacji i funkcji (pracownik/klient). Interfejs w zależności od poziomu dostępu się różni, dzięki czemu użytkownicy mają dostęp do przewidzianych dla siebie funkcji. Dane przed wysłaniem do bazy danych zostają sprawdzone pod kątem zgodności typu. Logowanie odbywa się za pomocą formularza, gdzie hasło wpisywane jest w trybie zakrytym. Hasła są przechowywane w bazie danych w postaci zaszyfrowanej (SHA-256). Komunikacja odbywa się połączeniem szyfrowanym TLS 1.1/1.2.

6. Podsumowanie i wnioski

Zarówno system bazodanowy, jak i aplikacje dostępne stanowią duże uproszczenie świata rzeczywistego. Projekt pozwolił nam na zapoznanie się z nowoczesnymi technologiami, m.in. SQL, Java, Node.js i przeprowadzenie implementacji systemu od modelowania oraz projektu, poprzez implementację bazy danych, a także aplikację dostępową. Pozwoliło to skupić się na integralnych rozwiązaniach, a także na kompleksowych testach.

Bibliografia

1. Dokumentacja PM2, Disqus
<http://pm2.keymetrics.io/docs/usage/cluster-mode/> [20.01.2019]
2. Dokumentacja JavaFX, Oracle
<https://docs.oracle.com/javase/8/javafx/api/toc.htm> [20.01.2019]
3. Dokumentacja Retrofit, Square
<https://square.github.io/retrofit/> [20.01.2019]
4. Dokumentacja Node.js, Ryan Dahl
<https://nodejs.org/api/> [20.01.2019]