



Nazwa przedmiotu
"Projektowanie aplikacji biznesowych"

Informatyka
studia inżynierskie
semestr V

Dokumentacja projektu
„System przelewów międzybankowych”

Autorzy projektu
Stec Piotr
Szot Adam
Śliwa Katarzyna
Wojtowicz Aleksander
Wrona Wojciech
Zamorski Karol

Data sporządzenia 5.02.2021 r.

Spis treści:

1. Założenia projektu
2. Realizacja projektu
3. Schemat DDD
4. Przelewy ekspresowe
5. Przykładowe dane do obsługi banków
6. Sposób instalacji
7. Opis wyglądu aplikacji
8. Działanie aplikacji od strony backendowej
9. Widok weryfikacji ręcznej w jednostce rozliczeniowej
10. Baza danych w banku "Amethyst Holdings"
11. Baza danych w banku "Diamont Holdings"
12. Metody w jednostce rozliczeniowej
 - a) Tworzenie pliku json, który jest wysyłany do banku
 - b) Metoda do weryfikacji ręcznej, generowanie strony do weryfikacji ręcznej
 - c) Metoda przyjmująca obiekt xml z banków
 - d) Klasa odpowiadająca za połączenie z bazą danych
 - e) Klasa zawierająca metody do obsługi bazy danych
 - f) Funkcja do wyszukania transferu dla banku
 - g) Dodawanie transferu do bazy danych w jednostce rozliczeniowej
 - h) Metoda zwracająca status transferu
 - i) Metoda do automatycznej weryfikacji transakcji
 - j) Rozpoczęcie weryfikacji danych
13. Linki do źródeł i działających aplikacji

1. Założenia projektu

Założeniem projektu jest stworzenie dwóch działających aplikacji bankowych, pomiędzy którymi będzie można dokonywać operacji przelewów z użyciem specjalnej jednostki rozliczeniowej.

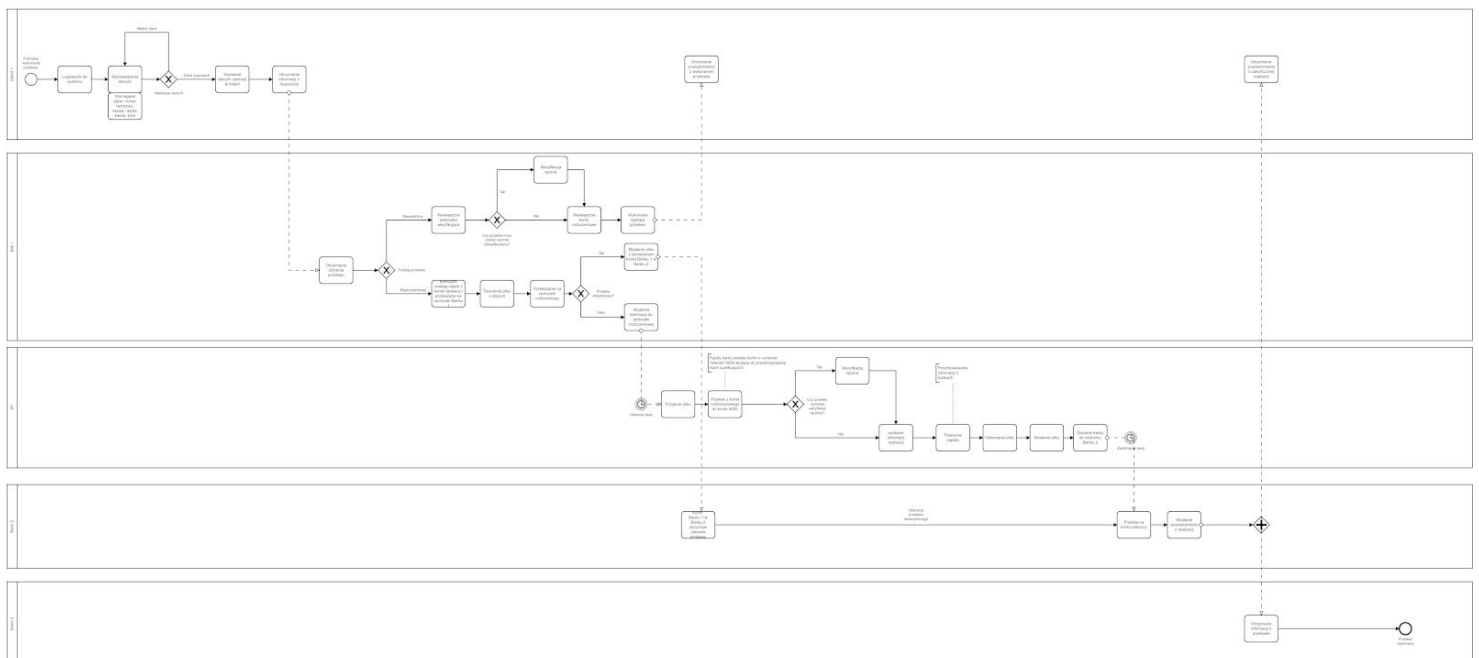
2. Realizacja projektu

- Zarówno pierwszy bank "Amethyst Holdings" jak i drugi bank "Diamond Holdings" zostały wykonane przy pomocy frameworku Laravel i technologii MySQL, PHP, JavaScript w tym JQuery.
- Jednostka rozliczeniowa oraz jednostka weryfikująca, zostały wykonane w języku Python, SQLAlchemy, Flask.

3. Schemat DDD

Załącznik

"DDD.png"



Elementem inicjującym działania wykonywane na schemacie jest potrzeba wykonania przelewu. Osoba, która chce wykonać tę operację, loguje się do systemu jednego z banków. Wybiera opcję realizacji przelewu, wprowadza niezbędne dane takie jak dane odbiorcy, nazwa przelewu, kwotę i rodzaj przelewu. Dane po wpisaniu są sprawdzane pod kątem poprawności (odpowiedniej liczby cyfr w rachunku, posiadanej kwoty). Po wypełnieniu pól przelew zostaje zapisany w historii oraz przekazany do realizacji.

W przypadku przelewu wewnętrznego przelew zostanie wykonany natychmiastowo. W przypadku większych kwot w tym procesie pośredniczy wewnętrzna jednostka rozliczeniowa banku, które

autonomicznie określa czy przelew może zostać wykonany, czy też wymaga autoryzacji manualnej przez pracownika banku.

W przypadku przelewów międzybankowych przelew po zatwierdzeniu danych przelew jest dodawany do puli przelewów oczekujących na wystanie do Jednostki Rozliczeniowej. Jednostka rozliczeniowa po przyjęciu pliku rozpoczyna od weryfikacji czy przelew przekracza wartość 1000 zł. Jeśli przesłana kwota jest większa, wtedy następuje wystanie wiadomości do banku nadawcy o potwierdzenie wykonania przelewu. Po pomyślnym procesie weryfikacji zostaje wysłana informacja o realizacji oraz wykonanie operacji dodania kwoty w drugim banku. Po zakończeniu operacji przelewu wysyłana jest wiadomość z potwierdzeniem wykonania przelewu.

W przypadku utworzenia przelewu ekspresowego kwota nadawcy zostaje dodana do konta odbiorcy poprzez wykonanie przelewu wewnątrz bankowego. Zakładamy, że w każdy bank ma swoje konto w innych bankach. Z tego też konta zostanie wykonana operacja przelewu na konto odbiorcy wewnątrz banku odbiorcy.

4. Przelewy ekspresowe

Po wypełnieniu danych w formularzu utworzenia nowego przelewu tworzony jest przelew, którego dane są zapisywane, a później przekazywane w formacie json do drugiego banku. W drugim banku tworzony wpis w bazie przelewów o przychodzącej transakcji. Po pomyślnym dodaniu wpisu kwota jest przekazywana ze wskazanego konta, które należy do drugiego banku do konta docelowego. Następnie informacja o powodzeniu przelewu jest wysyłana do banku pierwszego, gdzie kwota jest transferowana z konta nadawcy do specjalnego konta służącego do wyrównania salda w koncie na drugiego banku.

5. Przykładowe dane do obsługi banków

Bank "Amethyst Holdings"

Użytkownik: 456789

Hasło: hasło

Użytkownik: 987654

Hasło: hasło

Pracownik: 111111

Hasło: pracownicze

Bank "Diamond Holdings"

Użytkownik: 123456

Hasło: haslo123

Pracownik: 111111

Hasło: pracownicze

6. Sposób instalacji

Do uruchomienia banków wymagane są:

- PHP (zalecana wersja 7.2)
- Narzędzie composer
- Pusta baza danych w MySQL

Aby uruchomić aplikację, należy wypakować jej zawartość. Następnie w konsoli z poziomu folderu głównego należy uruchomić polecenie "composer install". Po instalacji dependencji należy upewnić się, czy w strukturze znajduje się plik o nazwie ".env". Jeśli plik nie znajduje się w strukturze, należy skopiować plik ".env.example" i zmienić jego nazwę. W pliku ".env" należy ustalić połączenie z utworzoną bazą danych. Po utworzeniu połączenia należy wpisać polecenia: "php artisan key:generate" oraz "php artisan migrate" oraz na koniec "php artisan serve".

7. Opis wyglądu aplikacji

Bank Amethyst

Amethyst Holdings

➔ Zaloguj się



JKowalski



...



ZALOGUJ

Logowanie do banku Amethyst

Strona początkowa banku Amethyst zawiera okno logowania. Aby zalogować się, wpisujemy login oraz hasło do naszego konta, a następnie klikamy "ZALOGUJ".

Prezentuje się nam taka strona.

Witaj Jan Kowalski

Konto główne

79 1020 5226 0012 5478 9632 5416

Saldo: 652.42 zł

w tym debet w koncie: 0,00 zł

HISTORIA

PRZELEW

Ostatnie transakcje

#	Nadawca / Odbiorca	Tytuł	Data	Kwota	Saldo
1	23132421540000000000253876	79102052260012547896325416	Przelew ekspres	08-02-2021 14:58	+10.42zł
2	79102052260012547896325416	67102052269874587536985201	PrzelewWewn	08-02-2021 14:42	-33.00zł
3	79102052260012547896325416	23132421540000000000253876	za kino	08-02-2021 14:36	-55.00zł
4	79102052260012547896325416	23132421540000000000253876	za pizze	08-02-2021 14:35	-44.00zł
5	23132421540000000000253876	79102052260012547896325416	Wydatki	08-02-2021 14:35	+10.00zł

POKAŻ WIĘCEJ

Zalogowane konto użytkownika Jan Kowalski

Po poprawnym zalogowaniu mamy możliwość podglądu naszego konta - wyświetlenie numeru konta, ostatnie transakcje, możliwość podglądu historii jak i wykonanie przelewu.

Aby wykonać przelew, klikamy przycisk "PRZELEW", który przekierowuje nas na następującą stronę.

Przelew

Z konta
79 1020 5226 0012 5478 9632 5416 ▼

Odbiorca
Wpisz nazwę odbiorcy

Adres (opcjonalny)
Adres

Miejscowość (opcjonalna)
Miejscowość

Kod pocztowy (opcjonalny)
00-000

Numer konta
00 0000 0000 0000 0000 0000

Tytuł
Przelew środków

Kwota
0,00

Typ przelewu
☒ Standardowy ☐ Ekspresowy

POTWIERDŹ PRZELEW

Okno przelewu

Strona przelewu pokazuje nam, z jakiego konta odbywa się przelew - numer konta nadawcy, wraz z imieniem, nazwiskiem oraz adresem.

Użytkownik wprowadza, nazwę odbiorcy, numer konta, tytuł, kwotę, adres, kod pocztowy oraz miejscowość, oraz typ przelewu - Standardowy oraz ekspresowy. Po zweryfikowaniu danych możemy potwierdzić przelew, aby go wysłać do wskazanego przez nas odbiorcy.

Aby przeglądać historię przelewów, klikamy na przycisk "HISTORIA", bądź "POKAŹ WIĘCEJ" na stronie głównej.

Amethyst Holdings

Historia transakcji

#	Nadawca / Odbiorca	Tytuł	Data	Kwota	Saldo
1	23132421540000000000253876	79102052260012547896325416	Przelew ekspres	08-02-2021 14:58	+10.42zł
2	79102052260012547896325416	67102052269874587536985201	PrzelewWewn	08-02-2021 14:42	-33.00zł
3	79102052260012547896325416	23132421540000000000253876	za kino	08-02-2021 14:36	-55.00zł
4	79102052260012547896325416	23132421540000000000253876	za pizze	08-02-2021 14:35	-44.00zł
5	23132421540000000000253876	79102052260012547896325416	Wydatki	08-02-2021 14:35	+10.00zł
6	23132421540000000000253876	79102052260012547896325416	Rachunki 2	08-02-2021 14:34	+10.00zł
7	79102052260012547896325416	23132421540000000000253876	PrzelewAmethyst2	08-02-2021 14:26	-23.00zł

Historia transakcji użytkownika Jan Kowalski

W historii transakcji możemy podglądać przelewy przychodzące oraz wychodzące z naszego konta, tytuł przelewów, datę oraz czas wysłania przelewów, oraz kwota i saldo konta.

Amethyst Holdings

Przelewy do weryfikacji


#	Odbiorca	Nadawca	Tytuł	Data	Kwota	Potwierdzić?
1	79102052260012547896325416	67102052269874587536985201	PrzelewWewnAuth	08-02-2021 15:07	6000.00zł	<div>TAK</div> <div>NIE</div>

Transakcje do potwierdzenia przez pracownika

Po zalogowaniu się na konto pracownika mamy dostęp do listy przelewów oczekujących na ręczną weryfikację.

Bank Diamond

DIAMOND
HOLDINGS



Login

Hasło

ZALOGUJ

Formularz logowania Banku Diamond

Stroną początkową banku Diamond jest formularz logowania. Danymi potrzebnymi do poprawnego zalogowania się użytkownika w tym banku są: minimum 6-znakowy, składający się z cyfr, login oraz hasło.

W przypadku błędnego podania hasła wyświetlany jest komunikat, iż nastąpił błąd logowania i należy sprawdzić poprawność wprowadzonego loginu bądź hasła. Po kliknięciu przycisku „zaloguj” wraz z poprawnymi danymi przekierowani zostajemy na stronę główną banku.



WITAJ ADAM ADAMSKI

KONTO BIEŻĄCE

23 1324 2154 0000 0000 0025 3876

SALDO: 268643.00 ZŁ

w tym debet: 0,00zł

HISTORIA

PRZELEW

OSTATNIE OPERACJE

	NADAWCA	ODBIORCA	TYTUŁ	DATA	KWOTA
1	23132421540000000000253876	23132421540000000000253876	Przelew	06-02-2021 21:01	-1.00zł
2	23132421540000000000253876	82102052260000610204177895	2253	06-02-2021 20:55	-2.00zł
3	23132421540000000000253876	82102052260000610204177895	Przelew	06-02-2021 20:55	-329.00zł

Strona główna

Na stronie głównej banku ukazane są podstawowe informacje dotyczące zalogowanego klienta takie jak jego imię nazwisko, saldo, numer konta.

Po prawej stronie znajdują się dwa przyciski przekierowujące na podstronę z historią oraz do formularza przelewowego.

Poniżej znajduje się historia 5 najnowszych przelewów klienta.

PRZELEW

Z konta

23 1324 2154 0000 0000 0025 3876



Saldo: 268643.00

Odbiorca

Imię nazwisko

Adres (opcjonalnie)

Adres

Miejscowość (opcjonalnie)

Miejscowość

Kod pocztowy (opcjonalnie)

00-000

Numer konta odbiorcy

00 0000 0000 0000 0000 0000

Tytuł przelewu

Przelew środków

Kwota

0,00

- ☒ Standardowy
☐ Ekspresowy

PRZELEJ

Formularz przelewu

W zakładce Przelew możemy wysyłać przelewy za pomocą formularza stworzonego do tej funkcjonalności. Możemy wybrać, z którego konta użytkownika wykonamy przelew oraz wypełniamy pola konieczne do realizacji przelewu zakończonej sukcesem, czyli: imię i nazwisko

odbiorcy, jego numer konta, tytuł przelewu oraz kwota, na jaką ma być przelew. Opcjonalnie wypełnić można też pola Adres, Miejscowość i Kod pocztowy. Wprowadzona została walidacja frontowa. Numery kont mogą być podawane ze spacjami, a także bez jednak muszą mieć odpowiedni format dla numeru konta, inaczej pojawi się stosowny komunikat. Nazwa odbiorcy musi posiadać zarówno imię jak i nazwisko, a jeżeli decydujemy podawać się kod pocztowy, musi on zostać podany w odpowiednim dla niego formacie. Kwota przelewu musi być różna od 0, nie może zawierać znaków takich jak - lub +, musi zawierać same cyfry oraz może być zapisana zarówno ze znakiem „,” jak i „.” lub bez tych znaków. Dostępne są także przyciski radio, dzięki którym możemy wybierać pomiędzy przelewem Standardowym a Ekspresowym. Domyślnie zaznaczona jest opcja przelewu Standardowego.

HISTORIA

Data początkowa

7 February, 2021

	NADAWCA	ODBIORCA	TYTUŁ	DATA	KWOTA
1	23132421540000000000253876	23132421540000000000253876	Przelew	06-02-2021 21:01	-1.00zł
2	23132421540000000000253876	82102052260000610204177895	2253	06-02-2021 20:55	-2.00zł
3	23132421540000000000253876	82102052260000610204177895	Przelew	06-02-2021 20:55	-329.00zł
4	23132421540000000000253876	82102052260000610204177895	Przelew1122211	06-02-2021 17:52	-52.00zł
10	23132421540000000000253876	82102052260000610204177895	Przelew	06-02-2021 16:26	-9.00zł
11	82102052260000610204177895	23132421540000000000253876	Przelew Środków	06-02-2021 15:00	+33.00zł
12	82102052260000610204177895	23132421540000000000253876	Przelew Środków	06-02-2021 14:13	+20.00zł
13	82102052260000610204177895	23132421540000000000253876	Przelew Środków	06-02-2021 14:11	+10.00zł
14	82102052260000610204177895	23132421540000000000253876	Przelew Środków	06-02-2021 14:06	+10.00zł
15	23132421540000000000253876	82102052260000610204177895	Przelew	05-02-2021 20:21	-200.00zł
16	23132421540000000000253876	82102052260000610204177895	Przelew-Express	05-02-2021 20:17	-150.00zł

Pełna historia przelewów

Historia przelewów zawiera dane o zrealizowanych przelewach.

Przelewy przesłane na konto klienta zaznaczone są kolorem zielonym oraz znakiem "+", natomiast te wysłane z konta klienta zaznaczone są na szaro wraz ze znakiem "-" przed kwotą.

Tabela ukazuje takie informacje jak: numer konta nadawcy, numer konta odbiorcy, tytuł przelewu, data przelewu oraz kwota, na jaką był przelew.

Istnieje funkcjonalność pozwalająca na wybranie interesującej nas daty za pomocą datepicker'a i do tego okresu czasu zostanie nam zawężona lista w historii.

DIAMOND HOLDINGS

STRONA GŁÓWNA

PRZELEW

HISTORIA TRANSAKcji

KONTA KLIENTA

WYLOGUJ

1. Saldo: 268643.00 zł
23 1324 2154 0000 0000 0025 3876

2. Saldo: 10880.00 zł
43 1324 2154 7327 4353 2393 7542

WITAJ ADAM ADAMSKI

KONTO BIEŻĄCE

43 1324 2154 7327 4353 2393 7542

SALDO: 10880.00 ZŁ

w tym debet: 0,00zł

HISTORIA

PRZELEW

OSTATNIE OPERACJE

NADAWCA	ODBIORCA	TYTUŁ	DATA	KWOTA
WIĘCEJ				

Tu ukazana jest możliwość wyboru konta przez użytkownika posiadającego więcej niż jedno konto w banku.

DIAMOND HOLDINGS

WYLOGUJ

OPERACJE DO WERYFIKACJI

	NADAWCA	ODBIORCA	TYTUŁ	DATA	KWOTA	POTWIERDZENIE
1	23132421540000000000253876	86132421545870516397407055	Przelew6000	06-02-2021 23:06	6000.00zł	<div>TAK</div> <div>NIE</div>
2	23132421540000000000253876	86132421545870516397407055	Przelew3000	06-02-2021 23:07	3000.00zł	<div>TAK</div> <div>NIE</div>

Panel pracownika - tabela przelewów do weryfikacji

Po zalogowaniu się na konto pracownika banku za pomocą odpowiednich danych wprowadzonych w formularzu logowania możemy jako pracownik banku dokonać weryfikacji przelewów wewnątrz bankowych powyżej 1000 zł. Po kliknięciu przycisku "tak" przelew zostaje poddany realizacji a tabela zaktualizowana, po kliknięciu "nie" przelew zostaje odrzucony.

WITAJ ADAM ADAMSKI
KONTO BIEŻĄCE
23 1324 2154 0000 0000 0025 3876

SALDO: 259643.00 ZŁ
w tym debet: 0,00zł

HISTORIA
PRZELEW

OSTATNIE OPERACJE

	NADAWCA	ODBIORCA	TYTUL	DATA	KWOTA
1	23132421540000000000253876	86132421545870516397407055	Przelew 3000	06-02-2021 23:07	-3000.00zł
2	23132421540000000000253876	86132421545870516397407055	Przelew6000	06-02-2021 23:06	-6000.00zł

WITAJ BARTŁOMIEJ BABACKI
KONTO BIEŻĄCE
86 1324 2154 5870 5163 9740 7055

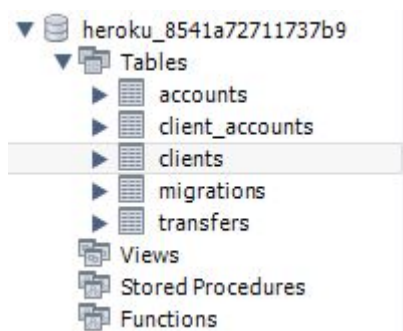
SALDO: 57715.00 ZŁ
w tym debet: 0,00zł

HISTORIA
PRZELEW

OSTATNIE OPERACJE

	NADAWCA	ODBIORCA	TYTUL	DATA	KWOTA
1	23132421540000000000253876	86132421545870516397407055	Przelew 3000	06-02-2021 23:07	+3000.00zł
2	23132421540000000000253876	86132421545870516397407055	Przelew6000	06-02-2021 23:06	+6000.00zł

8. Działanie aplikacji od strony backendowej



Banki posiadają swoją bazę danych, która przechowuje niezbędne informacje na temat klientów.

Realizacja przelewów wewnątrz bankowych.

“Diamonds Holdings”

5	23132421540000000000253876	19675.12	2021-02-03 12:22:49	2021-02-08 15:57:38
---	----------------------------	----------	---------------------	---------------------

Stan konta wysyłającego przelew, przed realizacją transakcji.

7	43132421547327435323937542	9980.00	2021-02-04 20:30:00	2021-02-08 14:41:07
---	----------------------------	---------	---------------------	---------------------

Stan konta odbiorcy przed realizacją przelewu.

	id	nadawca	nazwa_nad	adres_nad	kod_pocztowy_nad	miestowosc_nad	kwota	typ	tytul	odbiorca	nazwa_odb	adres_odb	kod_poc
▶	1991	23132421540000000000253876	Adam Adamski	ul. Krakowska 20	35123	Rzeszow	100.00	wewnetrzny	Wypłata	43132421547327435323937542	Piotr Stec	Rzeszów 123	12123

Utworzenie wiersza z przelewem.

5 23132421540000000000253876 19575.12 2021-02-03 12:22:49 2021-02-08 16:08:06

Stan konta nadawcy przelewu.

7 43132421547327435323937542 10080.00 2021-02-04 20:30:00 2021-02-08 16:08:06

Stan konta odbiorcy po realizacji przelewu.

“Amethyst Holdings”

5 79102052260012547896325416 59115.42 2021-02-08 09:28:44 2021-02-08 16:09:16

Stan konta wysyłającego przelew, przed realizacją transakcji.

6 67102052269874587536985201 25104.00 2021-02-08 09:28:44 2021-02-08 16:09:16

Stan konta odbiorcy przed realizacją przelewu.

161	79102052260012547896325416	Jani Kowalski	ul. Rzeszow 5	30123	Krakow	33.00	wewnetrzny	Przelew wewnetrzny	67102052269874587536985201	Bartłomiej Babacki
-----	----------------------------	---------------	---------------	-------	--------	-------	------------	--------------------	----------------------------	--------------------

Utworzenie wiersza z przelewem.

5 79102052260012547896325416 59115.42 2021-02-08 09:28:44 2021-02-08 16:09:16

Stan konta nadawcy przelewu.

6 67102052269874587536985201 25104.00 2021-02-08 09:28:44 2021-02-08 16:09:16

Stan konta odbiorcy po realizacji przelewu.

Realizacja przelewu międzybankowego

“Diamond Holdings”

5 23132421540000000000253876 19441.84 2021-02-03 12:22:49 2021-02-08 16:27:07

Stan konta nadawcy

“Amethyst Holdings”

5 79102052260012547896325416 59158.72 2021-02-08 09:28:44 2021-02-08 16:27:07

Stan konta odbiorcy.

2031	23132421540000000000253876	Adam Adamski	ul. Krakowska 20	35123	Rzeszow	36.10	ekspresowy	Pyszne.pl	79102052260012547896325416	Adam Szot
------	----------------------------	--------------	------------------	-------	---------	-------	------------	-----------	----------------------------	-----------

Tworzenie wierszy z przelewami. Widok banku "Diamond Holdings"

191	23132421540000000000253876	Adam Adamski	ul. Krakowska 20	35123	Rzeszow	36.10	ekspresowy	Pyszne.pl	79102052260012547896325416	Adam Szot
-----	----------------------------	--------------	------------------	-------	---------	-------	------------	-----------	----------------------------	-----------

Tworzenie wierszy z przelewami. Widok banku "Amethyst Holdings"

5	79102052260012547896325416	59194.82	2021-02-08 09:28:44	2021-02-08 16:39:05
---	----------------------------	----------	---------------------	---------------------

Stan konta odbiorcy po przelewach.

5	23132421540000000000253876	19405.74	2021-02-03 12:22:49	2021-02-08 16:39:05
---	----------------------------	----------	---------------------	---------------------

Stan konta nadawcy po przelewie.

9. Weryfikacja ręczna w jednostce weryfikującej

Przelewy do weryfikacji

Nadawca	Odbiorca	Kwota	Potwierdzić?
Diamond Holdings 89132421540000000000000003	Diamond Holdings 2 781020522600000000000001000	3000.00 zł	Wybierz opcję <div>Zatwierdź Zatwierdź Odrzuć</div>

POTWIERDŹ PRZELEWY

Po wejściu w link do weryfikacji ręcznej wyświetli nam się okno, w którym możemy zobaczyć oczekujące na realizację przelewy. Po wybraniu jednej z opcji możemy potwierdzić przelewy.

10. Baza danych banku "Amethyst Holdings"

MySQL Workbench

Query 1: `SELECT * FROM heroku_8541a7271173769.transfers;`

Result Grid:

#	id	nadawca	nazwa_nad	adres_nad	kod_pocztowy_nad	miejscowosc_nad	kwota	typ	tytuł	odbiorca	nazwa odb
111	67302526867468753686501	Bank 2		ul. Lwowska 15	30512	Rzeszow	1500.00	medybankowy	Skadka auto	8613242154000000000023376	2313242154000000000000000000000
121	310252260012547896325416	Jan Kowalski		ul. Szkolna 5	30500	Rzeszow	20.00	medybankowy	Wyromanie	310252260012547896325416	Bank 2
126	67302526867468753686501	Bank 2		ul. Lwowska 15	30512	Rzeszow	100.00	medybankowy	Palno	2313242154000000000023376	Adam Adam
141	67302526867468753686501	Bank 2		ul. Lwowska 15	30512	Rzeszow	100.00	medybankowy	Cogrik	2313242154000000000023376	Adam Adam
151	791025260012547896325416	Jan Kowalski					15.35	medybankowy	Jedzenie	791025260012547896325416	Jan Kowalski
161	791025260012547896325416	Jan Kowalski					25.00	medybankowy	Kino	791025260012547896325416	Jan Kowalski
171	791025260012547896325416	Jan Kowalski					88.00	medybankowy	PrzebiegMedybank	791025260012547896325416	Jan Kowalski
181	791025260012547896325416	Jan Kowalski					66.00	medybankowy	Za lino i kawę	791025260012547896325416	Jan Kowalski
191	791025260012547896325416	Jan Kowalski					11.00	medybankowy	Za daninę	791025260012547896325416	Jan Kowalski
201	791025260012547896325416	Jan Kowalski					15.00	medybankowy	Wyromanie	791025260012547896325416	Diamond Holdings
211	791025260012547896325416	Jan Kowalski					3.00	medybankowy	Za pizzę	791025260012547896325416	Jan Kowalski
221	791025260012547896325416	Jan Kowalski					7.00	medybankowy	Za kulur ydze	791025260012547896325416	Jan Kowalski
231	67302526867468753686501	Bank 2		ul. Lwowska 15	30512	Rzeszow	100.00	medybankowy	Lokalizacja	2313242154000000000023376	Adam Adam
241	67302526867468753686501	Bank 2		ul. Lwowska 15	30512	Rzeszow	100.00	medybankowy	Jedzenie	2313242154000000000023376	Adam Adam
251	791025260012547896325416	Jan Kowalski					15.35	medybankowy	Jedzenie	791025260012547896325416	Jan Kowalski
261	791025260012547896325416	Jan Kowalski					25.00	medybankowy	Kino	791025260012547896325416	Jan Kowalski
271	791025260012547896325416	Jan Kowalski					88.00	medybankowy	PrzebiegMedybank	791025260012547896325416	Jan Kowalski
281	791025260012547896325416	Jan Kowalski					66.00	medybankowy	Za lino i kawę	791025260012547896325416	Jan Kowalski
291	791025260012547896325416	Jan Kowalski					11.00	medybankowy	Za daninę	791025260012547896325416	Jan Kowalski
301	791025260012547896325416	Jan Kowalski					15.00	medybankowy	Wyromanie	791025260012547896325416	Diamond Holdings
311	791025260012547896325416	Jan Kowalski					3.00	medybankowy	Za pizzę	791025260012547896325416	Jan Kowalski
321	791025260012547896325416	Jan Kowalski					7.00	medybankowy	Za kulur ydze	791025260012547896325416	Jan Kowalski
331	791025260012547896325416	Jan Kowalski					23.00	medybankowy	Za czynsz	791025260012547896325416	Jan Kowalski
341	791025260012547896325416	Jan Kowalski					100.00	medybankowy	PrzebiegMedybank	791025260012547896325416	Jan Kowalski
351	8613242154000000000023376	Jan Kowalski		ul. Rzeszowska 15	30012	Krakow	50.00	ekspresowy	Ekspresowy	791025260012547896325416	Jan Kowalski

transfers 1 x

Output:

Action Output:

1 17.32.12 SELECT * FROM heroku_8541a7271173769.transfers LIMIT 0. 1000

Message: 35 row(s) returned

Duration / Fetch: 0.047 sec / 0.000 sec

11. Baza danych "Diamond Holdings"

MySQL Workbench

Query 1: `SELECT * FROM heroku_04927649a57ff.transfers;`

Result Grid:

#	id	nadawca	nazwa_nad	adres_nad	kod_pocztowy_nad	miejscowosc_nad	kwota	typ	tytuł	odbiorca	nazwa odb
1	2313242154000000000023376	Adam Adamski		ul. Krakowska 20	30523	Rzeszow	15.35	medybankowy	Jedzenie	791025260012547896325416	Jan Kowalski
11	2313242154000000000023376	Adam Adamski		ul. Krakowska 20	30523	Rzeszow	25.00	medybankowy	Kino	791025260012547896325416	Jan Kowalski
21	2313242154000000000023376	Adam Adamski		ul. Krakowska 20	30523	Rzeszow	12.00	wonczony	Przebieg	8613242154000000000023376	Jan Kowalski
31	2313242154000000000023376	Adam Adamski		ul. Krakowska 20	30523	Rzeszow	15.00	ekspresowy	PrzebiegMedybank	791025260012547896325416	Jan Kowalski
41	791025260012547896325416	Jan Kowalski		ul. Rzeszowska 5	30523	Krakow	20.00	ekspresowy	Wyrobki	2313242154000000000023376	Adam Adamski
51	2313242154000000000023376	Adam Adamski		ul. Krakowska 20	30523	Rzeszow	88.00	medybankowy	PrzebiegMedybank	791025260012547896325416	Jan Kowalski
61	2313242154000000000023376	Adam Adamski		ul. Krakowska 20	30523	Rzeszow	66.00	medybankowy	Za lino i kawę	791025260012547896325416	Jan Kowalski
71	2313242154000000000023376	Adam Adamski		ul. Krakowska 20	30523	Rzeszow	333.00	medybankowy	Za lino i kawę	791025260012547896325416	Jan Kowalski
81	2313242154000000000023376	Adam Adamski		ul. Krakowska 20	30523	Rzeszow	11.00	medybankowy	Za daninę	791025260012547896325416	Jan Kowalski
91	8913242154000000000000000	Diamond Holdings		ul. Szkolna 5	30500	Rzeszow	15.00	medybankowy	Wyromanie	791025260012547896325416	Diamond Holdings
101	2313242154000000000023376	Adam Adamski					50.00	medybankowy	Woda	2313242154000000000023376	Adam Adamski
111	2313242154000000000023376	Adam Adamski					10.00	medybankowy	Prad	2313242154000000000023376	Adam Adamski
121	2313242154000000000023376	Adam Adamski		ul. Krakowska 20	30523	Rzeszow	3.00	medybankowy	Za pizzę	791025260012547896325416	Jan Kowalski
131	2313242154000000000023376	Adam Adamski		ul. Krakowska 20	30523	Rzeszow	7.00	medybankowy	Za kulur ydze	791025260012547896325416	Jan Kowalski
141	2313242154000000000023376	Adam Adamski					10.00	medybankowy	Woda	2313242154000000000023376	Adam Adamski
151	2313242154000000000023376	Adam Adamski					50.00	medybankowy	Prad	2313242154000000000023376	Adam Adamski
161	2313242154000000000023376	Adam Adamski		ul. Krakowska 20	30523	Rzeszow	23.00	medybankowy	Za czynsz	791025260012547896325416	Jan Kowalski
171	2313242154000000000023376	Adam Adamski		ul. Krakowska 20	30523	Krakow	100.00	wonczony	PrzebiegMedybank	8613242154000000000023376	Jan Kowalski
181	8613242154000000000023376	Jan Kowalski		ul. Rzeszowska 15	30012	Krakow	100.00	medybankowy	PrzebiegMedybank	791025260012547896325416	Jan Kowalski
191	2313242154000000000023376	Adam Adamski					10.00	medybankowy	Woda	2313242154000000000023376	Adam Adamski
201	2313242154000000000023376	Adam Adamski					10.00	medybankowy	Prad	2313242154000000000023376	Adam Adamski
211	2313242154000000000023376	Adam Adamski					100.00	medybankowy	Palno	2313242154000000000023376	Adam Adamski
221	2313242154000000000023376	Adam Adamski					100.00	medybankowy	Cogrik	2313242154000000000023376	Adam Adamski
231	8613242154000000000023376	Jan Kowalski		ul. Rzeszowska 15	30012	Krakow	50.00	ekspresowy	Ekspresowy	791025260012547896325416	Jan Kowalski

transfers 6 x

Output:

Action Output:

6 17.28.55 SELECT * FROM heroku_04927649a57ff.transfers LIMIT 0. 1000

7 17.31.12 SELECT * FROM heroku_04927649a57ff.transfers LIMIT 0. 1000

8 17.31.38 SELECT * FROM heroku_04927649a57ff.transfers LIMIT 0. 1000

9 17.31.47 SELECT * FROM heroku_04927649a57ff.transfers LIMIT 0. 1000

10 17.32.03 SELECT * FROM heroku_04927649a57ff.transfers LIMIT 0. 1000

11 17.32.40 SELECT * FROM heroku_04927649a57ff.transfers LIMIT 0. 1000

Message: 24 row(s) returned

Duration / Fetch: 0.047 sec / 0.000 sec

12. Metody obsługujące jednostkę rozliczeniową

a) Tworzenie pliku json, który jest wysyłany do banku

```
def make_json(obj):  
    return {  
        'nadawca': {  
            'imie_nazwisko': obj.sender.name,  
            'adres': obj.sender.address,  
            'kod_pocztowy': obj.sender.zip_code,  
            'miestowosc': obj.sender.city,  
            'numer_konta': obj.sender.account_number,  
            'nazwa_banku': obj.sender.bank.name  
        },  
        'odbiorca': {  
            'imie_nazwisko': obj.receiver.name,  
            'adres': obj.receiver.address,  
            'kod_pocztowy': obj.receiver.zip_code,  
            'miestowosc': obj.receiver.city,  
            'numer_konta': obj.receiver.account_number,  
            'nazwa_banku': obj.receiver.bank.name  
        },  
        'kwota': obj.money,  
        'typ': obj.type,  
        'czas': obj.time,  
        'zweryfikowany': obj.verified,  
        'status': obj.status  
    }
```

b)Metoda do weryfikacji ręcznej, generowanie strony do weryfikacji ręcznej

```
@app.route("/manual_verification", methods=['GET', 'POST'])
def manual_verification():
    transfers = DBMethods().get_query(Transfers).filter_by(status=TransferStatusEnum.AWAITS_MANUAL_VERIFICATION.name)

    if request.method == 'POST':
        checkedBoxes = request.form.getlist('verify')
        checkedBoxesIds = request.form.getlist('verifyId')
        for i, j in zip(checkedBoxesIds, checkedBoxes):
            transfer = DBMethods().get(Transfers, int(i))
            if j == '1':
                update_transfer = Transfers(money=transfer.money,
                                              id_sender=transfer.id_sender,
                                              id_receiver=transfer.id_receiver,
                                              time=transfer.time,
                                              verified=True,
                                              status=TransferStatusEnum.VERIFIED.name,
                                              title=transfer.title)
                DBMethods().update_transfer(int(i), update_transfer)
            if j == '2':
                update_transfer = Transfers(money=transfer.money,
                                              id_sender=transfer.id_sender,
                                              id_receiver=transfer.id_receiver,
                                              time=transfer.time,
                                              verified=True,
                                              status=TransferStatusEnum.REJECTED.name,
                                              title=transfer.title)
                DBMethods().update_transfer(int(i), update_transfer)

        print(checkedBoxes)
        print(checkedBoxesIds)
        post_transfers = DBMethods().get_query(Transfers).filter_by(status=TransferStatusEnum.AWAITS_MANUAL_VERIFICATION.name)
        after_manual_verification_rejected(checkedBoxesIds)
        return render_template("potwierdz_przelew.html", obj=post_transfers)

    return render_template("potwierdz_przelew.html", obj=transfers)
```

c) Metoda przyjmująca obiekt xml z banków

```
@app.route("/", methods=['POST'])
def incoming():
    obj = xmltodict.parse(request.data)

    obj_json = obj
    print(json.dumps(obj_json, indent=4, default=str).encode('utf8'))

    bank = obj_json['Data']['BankData']

    if 'ReturnTransfer' not in obj_json['Data']:
        transfers = obj_json['Data']['Transfer']
        return_transfers = None
        json_return = verification_get_data(transfers, bank, return_transfers)
        print(json_return)
        return json_return, 200
    elif obj_json['Data']['ReturnTransfer'] is None:
        transfers = obj_json['Data']['Transfer']
        return_transfers = obj_json['Data']['ReturnTransfer']
        json_return = verification_get_data(transfers, bank, return_transfers)
        print(json_return)
        return json_return, 200
    elif obj_json['Data']['Transfer'] is not None and 'Transfer' in obj_json['Data']:
        transfers = obj_json['Data']['Transfer']
        return_transfers = obj_json['Data']['ReturnTransfer']
        json_return = verification_get_data(transfers, bank, return_transfers)
        print(json_return)
        return json_return, 200
    return jsonify(obj_json), 404
```


d) Klasa odpowiadająca za połączenie z bazą danych

```
class SQLUtil:
    __engine__ = engine
    __session__ = Session
    __connection__ = engine
    __transaction__ = engine

    # creates new engine for orm
    def create_engine(self):
        string = 'mysql+pymysql://freedbtech_jrPABur:1234567890@freedb.tech/freedbtech_jednostkaRozliczajaca'
        self.__engine__ = create_engine(string)

    # return current engine for orm
    def get_engine(self):
        return self.__engine__

    # creates new session for orm
    def open_session(self):
        session = sessionmaker()
        session.configure(bind=self.__engine__)
        self.__session__ = session()

    def get_session(self):
        return self.__session__

    def close_session(self):
        self.__session__.close()

    def open_connection(self):
        self.__connection__ = self.get_engine().connect()

    def get_connection(self):
        return self.__connection__

    def close_connection(self):
        return self.__connection__.close()

    def transaction(self):
        self.__transaction__ = self.__connection__.begin()

    def get_transaction(self):
        return self.__transaction__

    def transaction_rollback(self):
        return self.__transaction__.rollback()

    def session_rollback(self):
        return self.__session__.rollback()
```

e) Klasa zawierająca metody do obsługi bazy danych

```
class DBMethods:

    def __init__(self):
        self.util = SQLUtil()
        self.util.create_engine()
        self.util.open_session()

    def get_all(self, entity):
        try:
            return self.util.get_session().query(entity).all()
        except:
            self.util.session_rollback()
            raise

    def get(self, entity, id):
        try:
            return self.util.get_session().query(entity).get(id)
        except:
            self.util.session_rollback()
            raise

    def get_query(self, entity):
        try:
            return self.util.get_session().query(entity)
        except:
            self.util.session_rollback()
            raise

    def add(self, entity):
        try:
            self.util.get_session().add(entity)
            self.util.get_session().commit()
            return True
        except:
            self.util.session_rollback()
            raise

    def delete_id(self, entity, e_id):
        try:
            a = self.util.get_session().query(entity).get(e_id)
            self.util.get_session().delete(a)
            self.util.get_session().commit()
        except:
            self.util.session_rollback()
            raise

    def update_banks(self, e_id, new_entity):
        try:
            bank = self.util.get_session().query(Banks).get(e_id)
            bank.name = new_entity.name
            bank.account_number = new_entity.account_number
            bank.balance = new_entity.balance
            self.util.get_session().commit()
        except:
            self.util.session_rollback()
            raise

    def update_transfer(self, e_id, new_entity):
        try:
            sel = self.util.get_session().query(Transfers).get(e_id)
            sel.money = new_entity.money
            sel.id_sender = new_entity.id_sender
            sel.id_receiver = new_entity.id_receiver
            sel.time = new_entity.time
            sel.verified = new_entity.verified
            sel.status = new_entity.status
            sel.title = new_entity.title
            self.util.get_session().commit()
        except:
            self.util.session_rollback()
            raise
```

f) Funkcja do wyszukania transferu dla banku

```
def find_transfers_for_bank(bank):
    db = DBMethods()
    transfers = db.get_all(Transfers)
    transfers_list = []
    for transfer in transfers:
        recipient = db.get_query(Client).filter_by(id=transfer.id_receiver).first()
        sender = db.get_query(Client).filter_by(id=transfer.id_sender).first()
        re_bank = db.get_query(Banks).filter_by(name=recipient.bank.name).first()
        if bank == re_bank.name:
            if transfer.status.name == 'VERIFIED' or transfer.status.name == 'REJECTED' \
                or transfer.status.name == 'EXECUTED_TRUE' or transfer.status.name == 'EXECUTED_FALSE':
                transfers_list.append(make_json_transfers(transfer, sender, recipient))
    return transfers_list
```

g) Dodawanie transferu do bazy danych w jednostce rozliczeniowej

```
def add_transfer_to_db(obj):
    db = DBMethods()
    for i in obj:
        bank = DBMethods().get_query(Banks).filter_by(name=i['Sender']['BankName']).first()
        check_if_sender_exists = db.get_query(Client).filter_by(name=(i['Recipient']['Name']),
                                                                account_number=(i['Recipient']['Account']),
                                                                bank_id=bank.id).first()

        if check_if_sender_exists is None:
            client = Client(name=i['Recipient']['Name'], account_number=i['Recipient']['Account'], bank_id=bank.id)
            db.add(client)
            sender = db.get_query(Client).filter_by(name=(i['Recipient']['Name']),
                                                    account_number=(i['Recipient']['Account']),
                                                    bank_id=bank.id).first()
        else:
            sender = check_if_sender_exists

        bank_id = -1
        for j in db.get_all(Banks):
            acc_num = j.account_number[3:10]
            recip_acc_num = str(i['Recipient']['Account'])[3:10]
            if acc_num == recip_acc_num:
                bank_id = j.id
        if bank_id != -1 and bank_id >= 0:
            bankRe = DBMethods().get_query(Banks).filter_by(id=bank_id).first()
            check_if_recipient_exists = db.get_query(Client).filter_by(name=(i['Recipient']['Name']),
                                                                        account_number=(i['Recipient']['Account']),
                                                                        bank_id=bankRe.id).first()

            if check_if_recipient_exists is None:
                client2 = Client(name=i['Recipient']['Name'], account_number=i['Recipient']['Account'], bank_id=bankRe.id)
                db.add(client2)
                recipient = db.get_query(Client).filter_by(name=(i['Recipient']['Name']),
                                                            account_number=(i['Recipient']['Account']),
                                                            bank_id=bankRe.id).first()
            else:
                recipient = check_if_recipient_exists

        transfer = Transfers(money=i['Details']['Amount'], id_sender=sender.id, id_receiver=recipient.id,
                             time=i['Details']['Timestamp'], status=i['Details']['Status'],
                             verified=i['Details']['Verified'], title=i['Details']['Title'])
        db.add(transfer)
```


h) Metoda zwracająca status transferu

```
def return_transfers_status(obj):
    db = DBMethods()
    for i in obj:
        bank_id = -1
        for j in db.get_all(Banks):
            acc_num = j.account_number[3:10]
            sender_acc_num = str(i['Recipient']['Account'])[3:10]
            if acc_num == sender_acc_num:
                bank_id = j.id
        if bank_id != -1 and bank_id >= 0:
            bankRe1 = DBMethods().get_query(Banks).filter_by(id=bank_id).first()

            sender = db.get_query(Client).filter_by(name=(i['Sender']['Name']),
                                                    account_number=(i['Sender']['Account']),
                                                    bank_id=bankRe1.id).first()

            bank_id1 = -1
            for k in db.get_all(Banks):
                acc_num = k.account_number[3:10]
                recip_acc_num = str(i['Recipient']['Account'])[3:10]
                if acc_num == recip_acc_num:
                    bank_id1 = k.id
            if bank_id1 != -1 and bank_id1 >= 0:
                bankRe = DBMethods().get_query(Banks).filter_by(id=bank_id1).first()
                recipient = db.get_query(Client).filter_by(name=(i['Recipient']['Name']),
                                                            account_number=(i['Recipient']['Account']),
                                                            bank_id=bankRe.id).first()

                print(sender.id)
                print(recipient.id)

                transfer_cur = db.get_query(Transfers).filter_by(money=i['Details']['Amount'],
                                                                id_sender=sender.id,
                                                                id_receiver=recipient.id,
                                                                title=i['Details']['Title']).first()

                print(transfer_cur)
                if i['Details']['Executed'] == 'True' or i['Details']['Executed'] == 'true':
                    transfer = Transfers(money=i['Details']['Amount'],
                                          id_sender=sender.id,
                                          id_receiver=recipient.id,
                                          status=TransferStatusEnum.EXECUTED_TRUE,
                                          verified=True, title=i['Details']['Title'])

                    db.update_transfer(transfer_cur.id, transfer)
                else:
                    transfer = Transfers(money=i['Details']['Amount'],
                                          id_sender=sender.id,
                                          id_receiver=recipient.id,
                                          status=TransferStatusEnum.EXECUTED_FALSE,
                                          verified=True, title=i['Details']['Title'])

                    db.update_transfer(transfer_cur.id, transfer)
```

i) Metoda do automatycznej weryfikacji transakcji

```
def auto_verification(listt, bank):
    amount_sum = 0.0
    for i in listt:
        i['Details']['Status'] = TransferStatusEnum.VERIFIED.name
        i['Details']['Verified'] = True
        d = DBMethods()
        bank_id = -1
        for j in d.get_all(Banks):
            acc_num = j.account_number[3:10]
            recip_acc_num = str(i['Recipient']['Account'])[3:10]
            if acc_num == recip_acc_num:
                bank_id = j.id
        if bank_id != -1 and bank_id >= 0:
            db = d.get_query(Banks).filter_by(id=bank_id).first()
            bal = float(db.balance) - float(i['Details']['Amount'])
            up_ba = Banks(name=db.name, account_number=db.account_number, balance=bal)
            DBMethods().update_banks(db.id, up_ba)
            amount_sum += float(i['Details']['Amount'])
            # print(i['Details']['Amount'])
        else:
            i['Details']['Status'] = TransferStatusEnum.REJECTED.name
            i['Details']['Verified'] = True
    db = DBMethods()
    sum_in_db = db.get_query(Banks).filter_by(account_number=bank['AccountNumber']).first()
    # print(sum_in_db.balance)
    new_sum = float(sum_in_db.balance)-float(amount_sum)
    # print('amount sum: ', amount_sum)
    # print(new_sum)
    ba = Banks(name=sum_in_db.name, account_number=sum_in_db.account_number, balance=new_sum)
    # print(ba.balance)
    db.update_banks(sum_in_db.id, ba)
    return listt, amount_sum
```

j) Rozpoczęcie weryfikacji danych

```
def verification_get_data(obj, bank, return_transfers):
    db = DBMethods()
    current_bank = db.get_query(Banks).filter_by(account_number=bank['AccountNumber']).first()
    if current_bank is None:
        b = Banks(name=bank['Name'], account_number=bank['AccountNumber'], balance=10000.00)
        DBMethods().add(b)

    # if return_transfers is not None:
    #     return_transfers_status(return_transfers)

    if obj is not None:
        to_be_verified_manually = []
        to_be_verified_automatically = []

        for i in obj:
            i['Details']['Status'] = TransferStatusEnum.UNVERIFIED.name

            if float(i['Details']['Amount']) > 1000.00:
                i['Details']['Status'] = TransferStatusEnum.AWAITS_MANUAL_VERIFICATION.name
                i['Details']['Verified'] = False
                to_be_verified_manually.append(i)
            else:
                to_be_verified_automatically.append(i)

        sum_ver_man = get_sum(to_be_verified_manually)
        sum_ver_auto = get_sum(to_be_verified_automatically)
        sum_all = round(sum_ver_auto+sum_ver_man, 2)

        # print('Sum of the auto verified transactions:\t%.2f' % sum_ver_auto)
        # print('Sum of the manually verified transactions:\t%.2f' % sum_ver_man)
        # print('Total sum of the transfers:\t%.2f' % sum_all)
        # print(float(bank['Amount']))
        if sum_all == float(bank['Amount']):
            if sum_ver_auto > float(current_bank.balance) \
                or sum_ver_man > float(current_bank.balance) \
                or sum_all > float(current_bank.balance):
                for j in obj:
                    j['Details']['Status'] = TransferStatusEnum.UNVERIFIED.name
                    j['Details']['Verified'] = False
                # print('Transactions can not be executed, bank balance is to low')
                js = {
                    "error": 'Transactions can not be executed, bank balance is to low'
                }
                return json.dumps(js, ensure_ascii=False, indent=4).encode('utf8')
            else:
                # print('Transactions can be executed, bank balance is ok')
```

```

list1, amount_sum1 = auto_verification(to_be_verified_automatically, bank)
# print("Auto auth:\t%.2f" % amount_sum1)
add_transfer_to_db(list1)

list2, amount_sum2 = manual_verification(to_be_verified_manually)
# print("Manual auth:\t%.2f" % amount_sum2)
add_transfer_to_db(list2)

sum1 = amount_sum1+amount_sum2
# print('Total sum of the transfers:\t%.2f' % sum1)

l = []
for i in to_be_verified_manually:
    l.append(i)
for j in to_be_verified_automatically:
    l.append(j)

cur_bank = DBMethods().get_query(Banks).filter_by(account_number=bank['AccountNumber']).first()
js = {
    "BankData": {
        "AccountNumber": bank['AccountNumber'],
        "Balance": cur_bank.balance
    },
    "ReturnTransfers": 1,
    "Transfers": find_transfers_for_bank(bank['Name'])
}
return jsonify(js)
else:
    for j in obj:
        j['Details']['Status'] = TransferStatusEnum.UNVERIFIED.name
    # print('Transactions can not be executed, sum of transfers in not equal to stated sum in the file')
    js = {
        "error": 'Transactions can not be executed, sum of transfers in not equal to stated sum in the file'
    }
    return json.dumps(js, ensure_ascii=False, indent=4).encode('utf8')

```

13. Linki do źródeł i działających aplikacji

a. GitHub

<https://github.com/wojtekwrona232/fuzzy-winner>

b. Jednostka rozliczeniowa

i. pab-jr.herokuapp.com

ii. weryfikacja ręczna

pab-jr.herokuapp.com/manual_verification

c. Bank - Diamond

i. bank-diamond.herokuapp.com

d. Bank - Amethyst

i. bankamethyst.herokuapp.com

e. Załączniki

a. "DDD.png" [DDD.png](#)

b. "ERD_JR.png" [ERD_JR.png](#)

c. "ERD_Bank.png" [ERD_Bank.png](#)