



Uniwersytet Rzeszowski

Mikrosystem bankowy

Cel projektu:

Stworzenie działającego hermetycznego systemu składającego się z dwóch banków i jednostki rozliczającej

Prowadzący:

mgr inż. Marcin Mrukowicz

Autorzy:

Damian Paluch

Adam Młynek

Łukasz Mączka

Patryk Paściak

Filip Papiernik

Mikrosystem bankowy	1
Projekt - koncepcja	4
Opis projektu	4
Bank A	4
Bank B	4
Jednostka rozliczeniowa	4
Baza danych	4
Funkcjonalności	5
Bank A i bank B	5
Jednostka rozliczeniowa	5
Bank A - frontend i backend	7
Struktura projektu	7
Interfejs	8
Rejestracja	8
Logowanie	10
Strona główna - po zalogowaniu	11
Przelewy	11
Historia	13
Backend	14
Config	14
Generowanie numeru konta	14
Funkcja do tworzenia przelewu	15
API - Wszystkie przelewy	15
API - Realizowanie przelewu	16
API - aktualizowanie statusu przelewu	16
Bank B - frontend i backend	17
Struktura projektu	17
Interfejs	19
Rejestracja	19
Logowanie	22
Strona główna - po zalogowaniu	22
Przelewy	23
Historia	24
Backend	25
Config	25
Generowanie numeru konta	25
Funkcja do tworzenia JSONA z przelewami	26
Jednostka rozliczeniowa	27
Struktura projektu	27
Interfejs	28

Panel logowania	28
Nieudane logowanie	28
Panel podsumowania	29
Akceptacja przelewów	29
Lista przelewów do automatycznego zrealizowania	30
Historia przelewów	30
Wyświetlanie poszczególnych podstron	31
Wylogowywanie	31
Algorytm	32

1. Projekt - koncepcja

1.1. Opis projektu

Celem projektu jest stworzenie działającego hermetycznego systemu składającego się z dwóch banków i jednostki rozliczającej. Dzięki jednostce systemu banków mogą się wzajemnie komunikować i wysyłać między sobą przelewy. Dodatkowo każdy z banków posiada funkcje przelewów wewnętrznych rozliczanych bez pośrednictwa jednostki rozliczeniowej.

1.2. Bank A

Implementacja Banku A została wykonana w języku PHP (backend), zaś wygląd aplikacji został stworzony w hipertekstowym języku znaczników HTML połączonym z JavaScriptem oraz kaskadowymi arkuszami stylów CSS. API potrzebne do komunikacji z jednostką rozliczeniową zostało również stworzone w języku PHP.

1.3. Bank B

Bank B został stworzony przy pomocy języka Java oraz frameworka Spring. Dla ułatwienia połączenia HTML oraz backendu został użyty dodatkowo Thymeleaf. Wygląd opiera się na kaskadowych arkuszach stylów oraz skryptach JavaScript.

1.4. Jednostka rozliczeniowa

Jednostka rozliczająca została zaimplementowana w języku PHP dla backendu (logika oraz API) oraz HTML i CSS dla frontendu.

1.5. Baza danych

Utworzona została baza danych MYSQL dla banku A, banku B oraz jednostki rozliczeniowej.

2. Funkcjonalności

2.1. Bank A i bank B

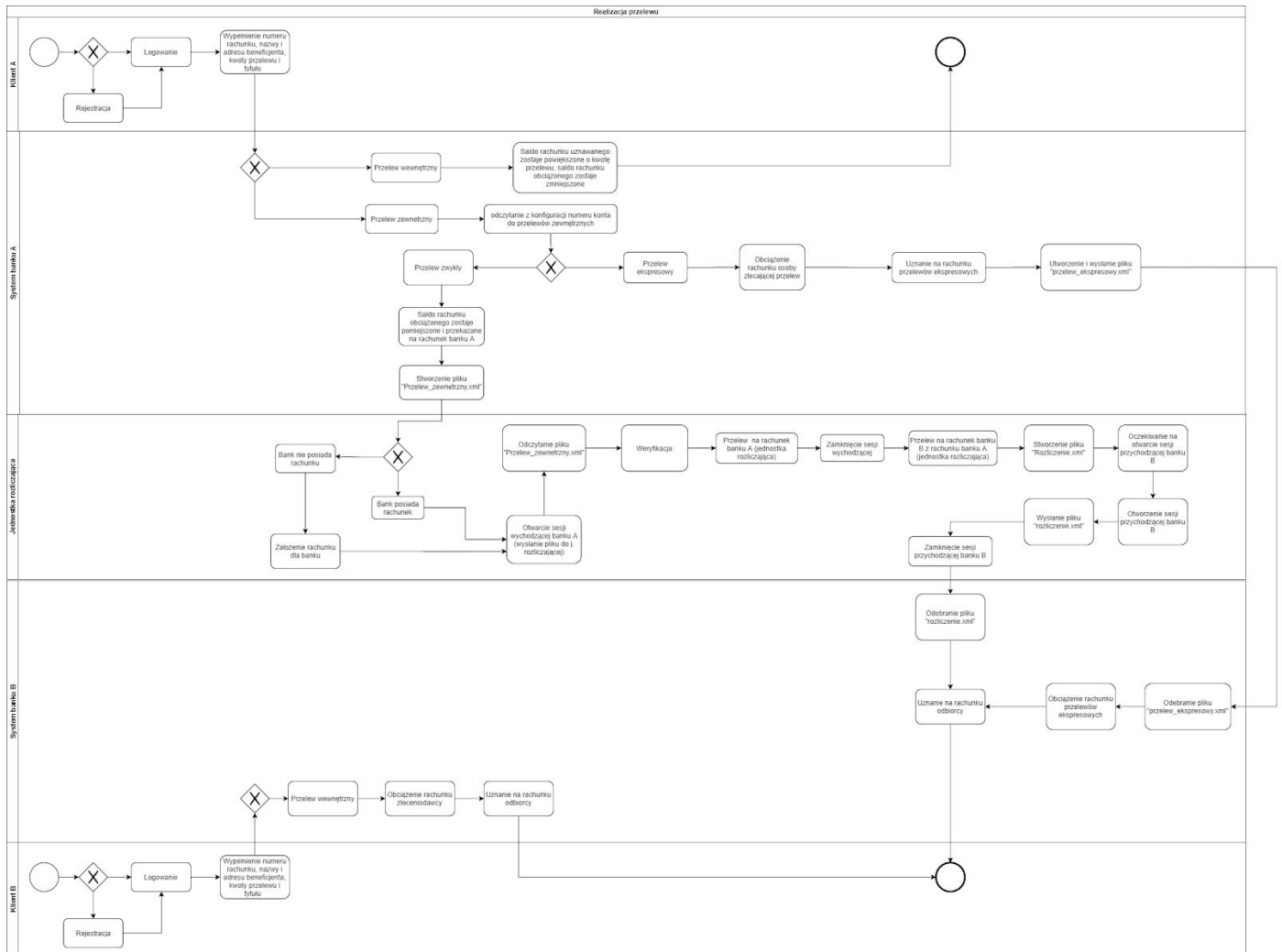
Każdy z banków może wykonać dwa rodzaje operacji - przelew wewnętrzny oraz przelew zewnętrzny który przechodzi przez jednostkę rozliczeniową. Dla przelewu wewnętrznego najpierw pobierana jest lista wszystkich kont bankowych, aby upewnić się, że konto odbiorcy istnieje. Jeśli taki numer bankowy istnieje odejmowana jest kwota z rachunku zleceniodawcy i jednocześnie dodawana jest ta sama kwota na rachunku odbiorcy. Żeby sprawdzić czy wykonywany przelew jest zewnętrzny sprawdzany jest numer banku, czyli osiem cyfr znajdujących się na pozycjach od czwartej do dwunastej w numerze rachunku. W przypadku przelewów zewnętrznych mechanizm sprawdzania czy dane konto istnieje jest podobny jak w wewnętrznych, z małą różnicą w postaci tego, że listę kont pobieramy od drugiego banku. Następnie odejmujemy pieniądze z konta zleceniodawcy oraz wysyłamy go do jednostki rozliczającej. W obu bankach jest możliwość rejestracji nowych użytkowników. Wtedy też jest tworzony nowy numer rachunku, który zawiera: dwie litery kraju, w tym przypadku "PL", dwie cyfry kontrolne obliczane według standardu IBAN, osiem cyfr banku oraz 16 losowych cyfr definiujących indywidualny numer. Użytkownik po zalogowaniu ma dodatkowo możliwość zobaczenia historii wszystkich przelewów przychodzących i wychodzących z rozwijanymi szczegółami oraz stanu swojego konta.

2.2. Jednostka rozliczeniowa

Jednostka rozliczeniowa odbiera przelewy zewnętrzne z podłączonych do niej banków, następnie przy użyciu zaimplementowanego algorytmu ocenia czy przelew powinien zostać automatycznie zaakceptowany oraz rozliczony. W przypadku gdy przelew nie zostanie automatycznie zaakceptowany ląduje w kolejce od zaakceptowania lub odrzucenia przez administratora jednostki rozliczeniowej. Przelewy realizowane są w trzech sesjach, które można ustawić z poziomu kodu programu. Dla nowo zarejestrowanych banków tworzone są konta wewnątrz jednostki, które zawierają: dwie litery kraju, w tym przypadku "PL", dwie cyfry kontrolne obliczane według standardu IBAN, osiem cyfr banku (jednostki rozliczeniowej) oraz 16 losowych cyfr definiujących indywidualny numer. Administrator po zalogowaniu ma dostęp do akceptacji

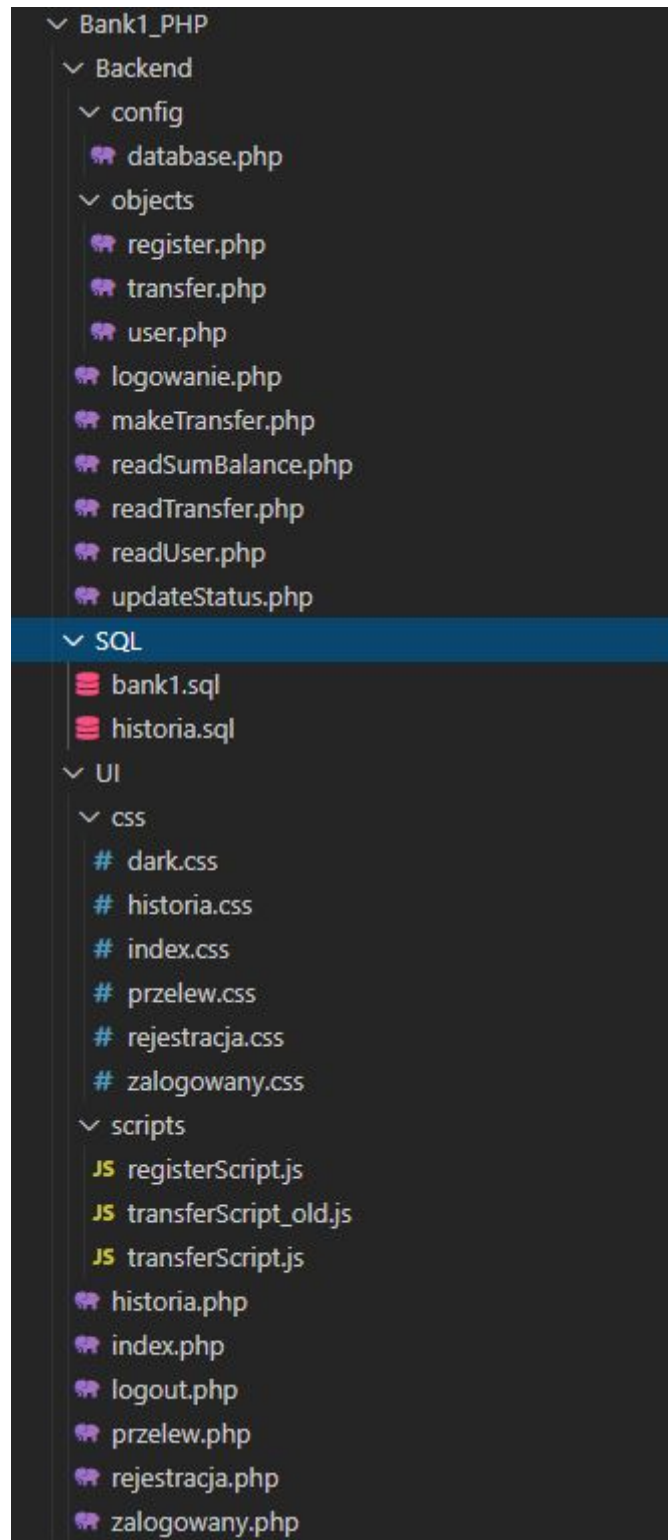
przelewów, przelewów które zostaną zaakceptowane po uruchomieniu sesji i historii przelewów.

3. Diagram aplikacji



4. Bank A - frontend i backend

4.1. Struktura projektu




4.2. Interfejs

Interfejs użytkownika pozwala na tworzenie nowego konta, logowanie, przeglądanie historii operacji oraz wykonywanie przelewu wewnętrznego i zewnętrznego.

4.2.1. Rejestracja


BardzoDobry
Bank

 Dane logowania

Login

Hasło

Powtórz hasło

 Dane osobowe

Imię

Nazwisko

PESEL

Adres e-mail

Telefon komórkowy

 Adres zamieszkania

Miejscowość

Ulica

Numer domu

Kod pocztowy

The screenshot shows a registration panel with a dark background. On the left, there is a pink icon of a document with three horizontal lines, followed by the text "Oświadczenia". To the right of this, there are three stacked rectangular boxes, each containing a line of text. The first box says "Wyrażam zgodę na przetwarzanie przez Bank podanych przeze mnie danych kontaktowych". The second box says "Jestem świadomy(-ma) odpowiedzialności karnej za złożenie fałszywego oświadczenia". The third box says "Wyrażam zgodę na używanie przez Bank połączenia telefonicznego". At the bottom of the panel, there are two pink buttons: "Wróć" on the left and "Załącz konto" on the right. At the very bottom, in small text, it says "STOPKA COPYRAIT @ MOJE".

Oświadczenia

Wyrażam zgodę na przetwarzanie przez Bank podanych przeze mnie danych kontaktowych

Jestem świadomy(-ma) odpowiedzialności karnej za złożenie fałszywego oświadczenia

Wyrażam zgodę na używanie przez Bank połączenia telefonicznego

Wróć

Załącz konto

STOPKA COPYRAIT @ MOJE

Panel rejestracji pozwala na zarejestrowanie się w banku A, każde pole jest sprawdzane przez skrypt napisany w JavaScriptcie.

The screenshot shows a login and password registration form. It has a dark background. The first section is titled "Login" and contains a text input field with the value "sprawdz333". Below this field, there are two green bullet points with checkmarks: "• Przynajmniej sześć znaków ✓" and "• Musi zawierać tylko litery oraz cyfry ✓". The second section is titled "Hasło" and contains a text input field with four dots. Below this field, there are two red bullet points: "• Przynajmniej osiem znaków" and "• Musi zawierać symbol specjalny". The third section is titled "Powtórz hasło" and contains a text input field with four dots. The input fields for "Hasło" and "Powtórz hasło" are outlined in red, while the "Login" field is outlined in green.

Login

sprawdz333

- Przynajmniej sześć znaków ✓
- Musi zawierać tylko litery oraz cyfry ✓

Hasło

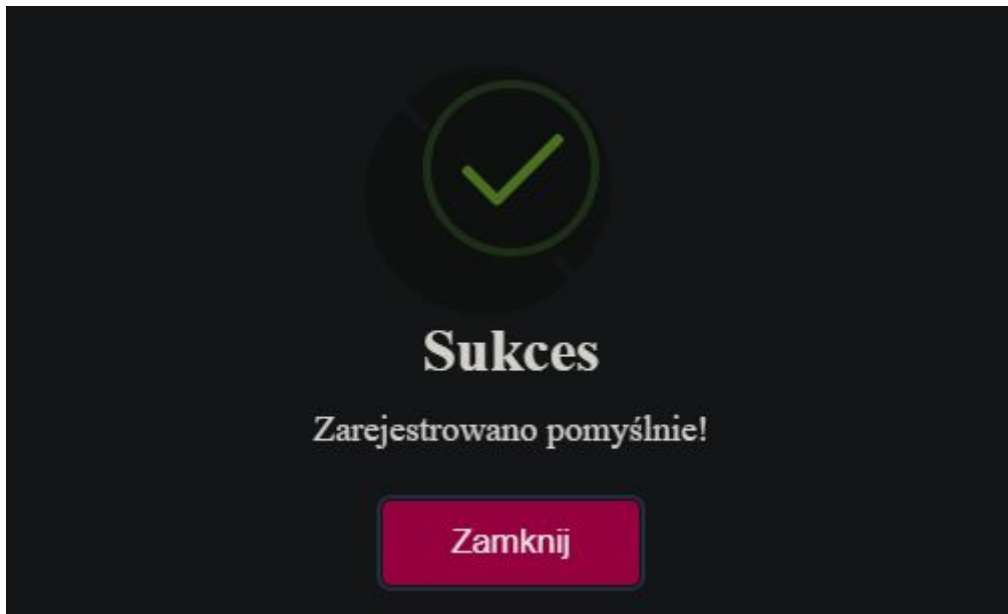
....

- Przynajmniej osiem znaków
- Musi zawierać symbol specjalny

Powtórz hasło

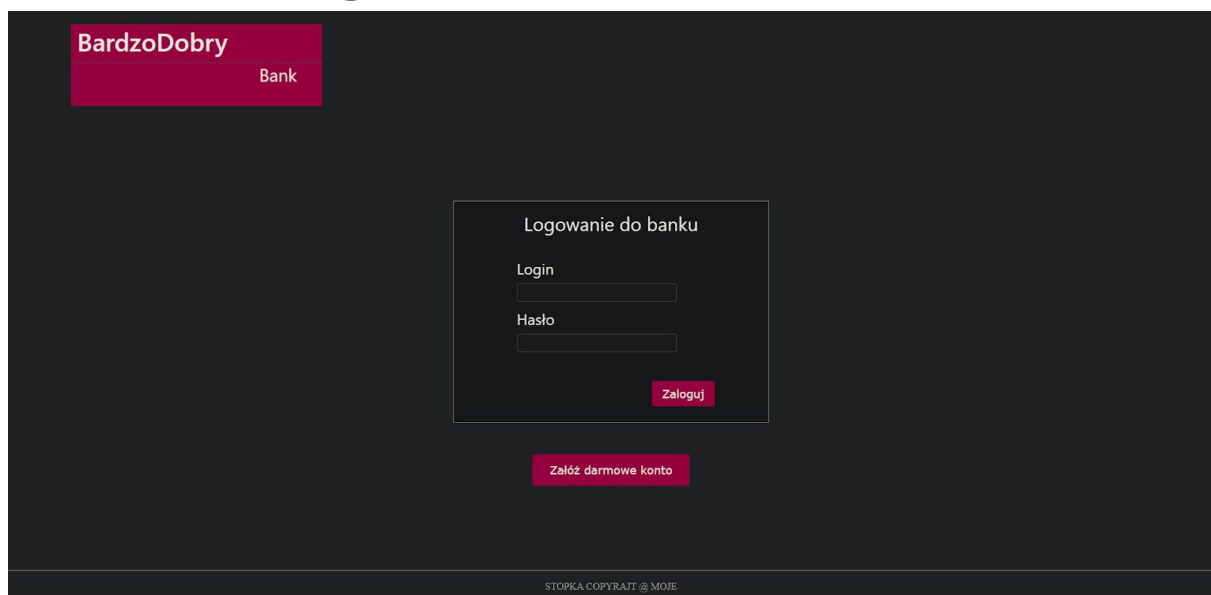
....

W przypadku poprawnego wpisania i spełnienia wszystkich warunków - pole zaświeci się na zielono. Wymogi znajdują się pod danym polem, zmieniają się i informują użytkownika o poprawności w czasie rzeczywistym. Gdy użytkownik nie spełnia wymogów pole zaświeci się na czerwono informując o błędzie. Jeżeli użytkownik naciśnie guzik "Zarejestruj" aplikacja przeniesie go do najwyższego błędnie wypełnionego pola.



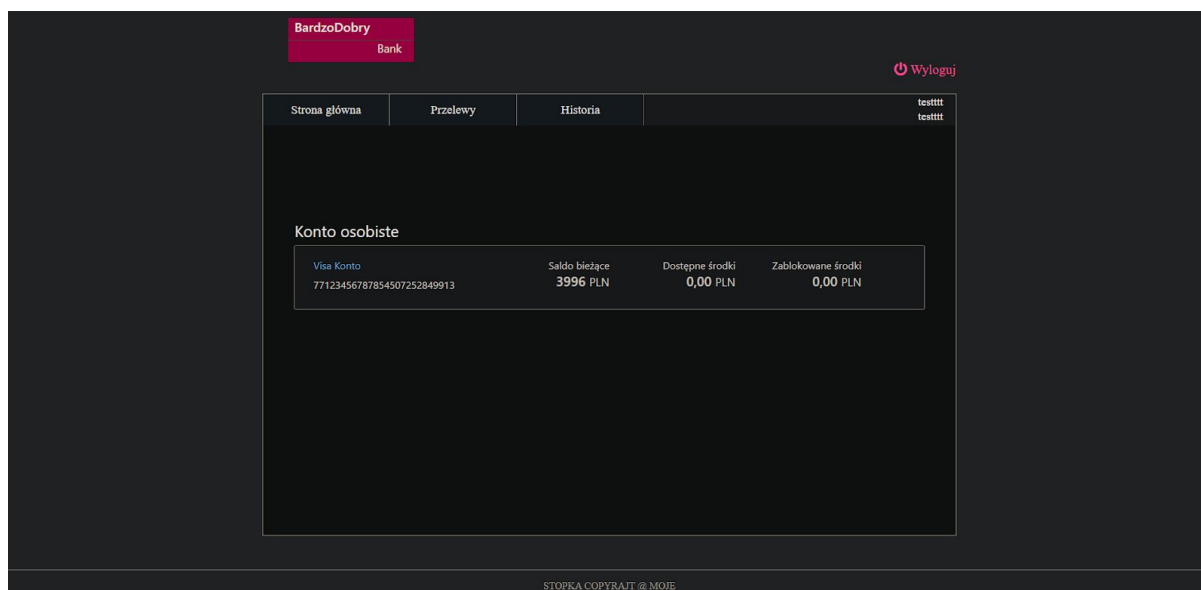
W przypadku pomyślnego zarejestrowania ukazuje się animowany alert z informacją o sukcesie. Stylizowany pod ciemno-różowy motyw banku.

4.2.2. Logowanie



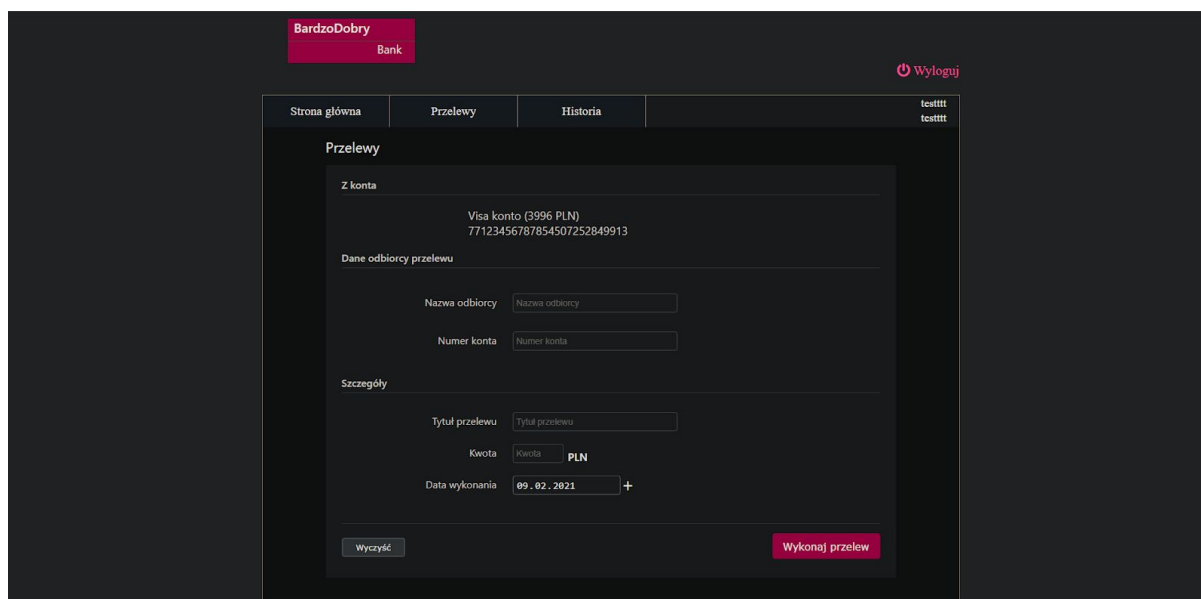
Użytkownik po uprzedniej rejestracji może się logować do banku gdzie będzie mógł wykonywać przelewy. W przypadku podania złych danych, aplikacja wyświetli stosowny komunikat.

4.2.3. Strona główna - po zalogowaniu



Po zalogowaniu użytkownik widzi swoje konto, w prawym rogu wyświetlane są informacje o użytkowniku (imię i nazwisko). Widać również aktualne saldo na koncie.

4.2.4. Przelewy



W zakładce “Przelewy” użytkownik może wykonywać przelewy zewnętrzne jak i wewnętrzne po uprzednim poprawnym uzupełnieniu wymaganych pól.

Strona główna	Przelewy	Historia	testttt testttt
---------------	----------	----------	--------------------

Przelewy

Z konta

Visa konto (3996 PLN)
77123456787854507252849913

Dane odbiorcy przelewu

Nazwa odbiorcy:

Numer konta:
Musi zawierać tylko cyfry ✓
Musi zawierać dwadzieścia sześć cyfr ✓
Musisz podać numer różny od swojego ✓

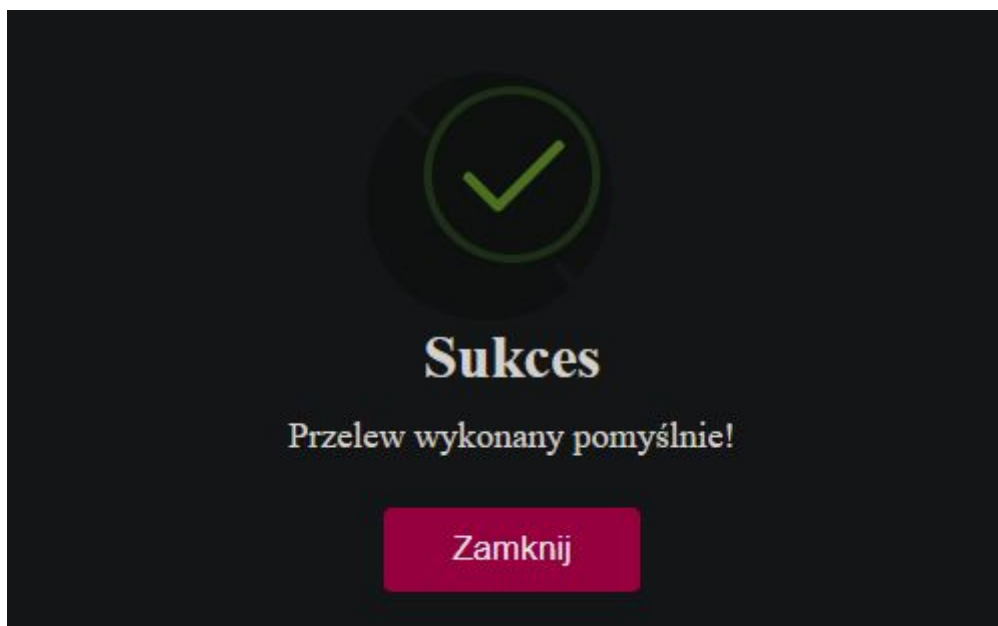
Szczegóły

Tytuł przelewu:

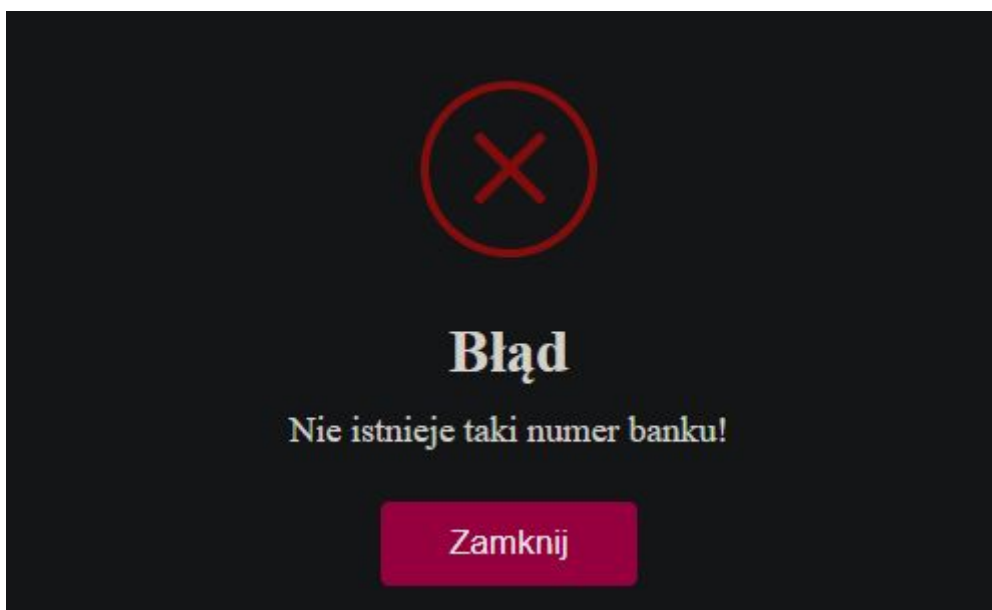
Kwota: **PLN**
Musi zawierać tylko cyfry ✓
Musisz mieć taką kwotę na koncie ✓
Musisz podać więcej niż zero ✓

Data wykonania: +

Błędy również sprawdzane są przez skrypt w JS.



Gdy wszystkie dane są poprawne wyskakuje alert z informacją o sukcesie.



W przypadku podania złego numeru banku który nie istnieje w systemie wyskoczy stosowny błąd z informacją.

4.2.5. Historia

Data przelewu	Tytuł przelewu	Odbiorca	Na numer konta	Kwota
2021-01-23	Za cos	Karol	77123456787854507252849913	999 zł
2021-01-23	Za cos	Karol	77123456787854507252849913	999 zł
2021-01-23	Za cos	Karol	77123456787854507252849913	999 zł
2021-01-23	Za cos	Karol	77123456787854507252849913	999 zł
2021-02-09	Testowy	Testowy Przelew	14876543216592903107435015	-382 zł

Tytuł przelewu:
Testowy

Odbiorca:
Testowy Przelew

Z numeru konta:
77123456787854507252849913

Na numer konta:
14876543216592903107435015

Kwota transakcji:
- 382 zł

W panelu “Historia” znajdują się przelewy wychodzące i przychodzące, każdy przelew można z osobna rozwinąć i uzyskać dodatkowe informacje i w przypadku gdy tytuł lub odbiorca jest za długi wyświetli się całość.

4.3. Backend

4.3.1. Config

```
<?php
class Database{
private $host = "localhost";
private $database_name = "bank1";
private $username = "root";
private $password = "";
public $connection;

public $bankNumberA = "12345678";
public $linkCheckNumber = "http://localhost/ProjektPAB/1_Rozliczajaca/BACKEND/readKonta.php";

public function getConnection(){
    $this->connection = null;
    try{
        $this->connection = new PDO("mysql:host=" . $this->host . ";dbname=" . $this->database_name, $this->username, $this->password);
        $this->connection->exec("set names utf8");
    }catch(PDOException $exception){
        echo "Błąd łączenia: " . $exception->getMessage();
    }
    return $this->connection;
}
}
?>
```

W tym pliku podajemy nazwę hosta, nazwę bazy danych, użytkownika i hasło, podajemy również numer banku.

4.3.2. Generowanie numeru konta

```
function generateBankNumber($countryCode, $bankNumber, $accountNumber){
    if(!empty($countryCode) && !empty($bankNumber) && !empty($accountNumber)){
        $accountAndBankNumber = $bankNumber.str_pad($accountNumber,10,"0",STR_PAD_LEFT);
        $checkSum = str_pad(98 - bcmath($accountAndBankNumber, 97), 2, "0", STR_PAD_LEFT);
        $final = $countryCode.$checkSum.$accountAndBankNumber;
        return $final;
    }
    else
        return 0;
}

$accountNumber = rand(1000000000000000,999999999999999);
```

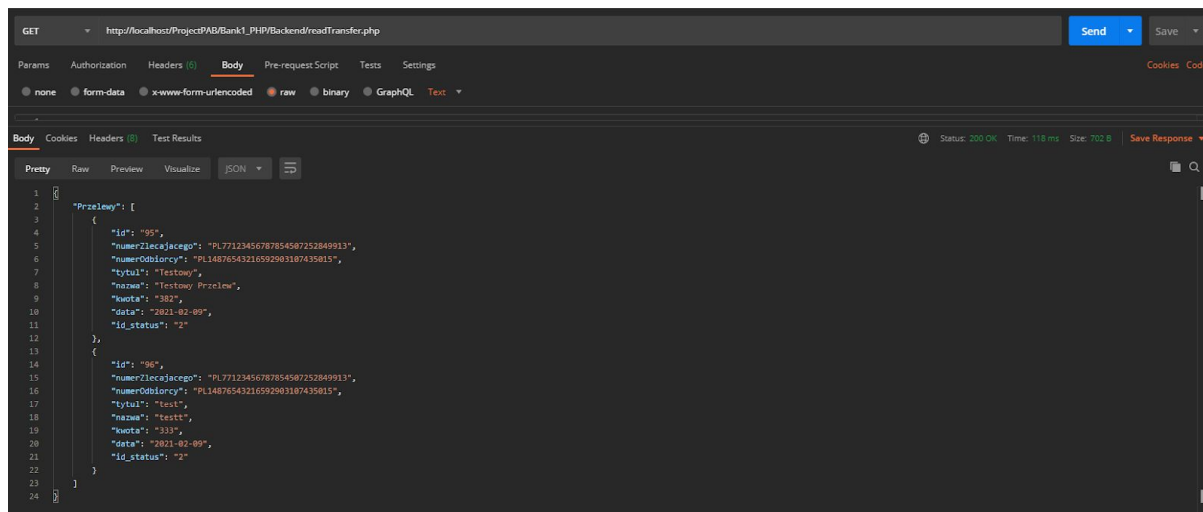
Podczas generowania numeru konta do funkcji podawany jest kod kraju (w tym przypadku "PL"). Następnie numer banku który podawany jest w configu i numer konta który jest generowany randomowo, wyliczana jest suma kontrolna. Następnie całość jest łączona. (W przypadku gdyby numer konta się powtórzył jest on sprawdzany czy nie istnieje już taki w bazie)

4.3.3. Funkcja do tworzenia przelewu

```
function makeTransfer(){
    // zapytanie do wstawiania rekordu
    $query = "INSERT INTO " . $this->tableHistory . "
    SET
    numerZlecajacego=:numerZlecajacego, numerOdbiorcy=:numerOdbiorcy, tytul=:tytul, nazwa=:nazwa, kwota=:kwota, data=:data, id_status=:id_status, type=:type;
    UPDATE user SET balance = (SELECT balance FROM user WHERE bankNumber=:numerOdbiorcy) + :kwota WHERE bankNumber=:numerOdbiorcy;";
    // przygotowanie zapytania
    $stmt = $this->connection->prepare($query);
    // zabezpieczenie
    $this->numerZlecajacego = htmlspecialchars(strip_tags($this->numerZlecajacego));
    $this->numerOdbiorcy = htmlspecialchars(strip_tags($this->numerOdbiorcy));
    $this->tytul = htmlspecialchars(strip_tags($this->tytul));
    $this->nazwa = htmlspecialchars(strip_tags($this->nazwa));
    $this->kwota = htmlspecialchars(strip_tags($this->kwota));
    $this->data = htmlspecialchars(strip_tags($this->data));
    $this->id_status = htmlspecialchars(strip_tags($this->id_status));
    $this->type = htmlspecialchars(strip_tags($this->type));
    // podłączenie wartości do zapytania
    $stmt->bindParam(":numerZlecajacego", $this->numerZlecajacego);
    $stmt->bindParam(":numerOdbiorcy", $this->numerOdbiorcy);
    $stmt->bindParam(":tytul", $this->tytul);
    $stmt->bindParam(":nazwa", $this->nazwa);
    $stmt->bindParam(":kwota", $this->kwota);
    $stmt->bindParam(":data", $this->data);
    $stmt->bindParam(":id_status", $this->id_status);
    $stmt->bindParam(":type", $this->type);
    // wykonanie zapytania
    if ($stmt->execute()) {
        return true;
    }
    return false;
}
```

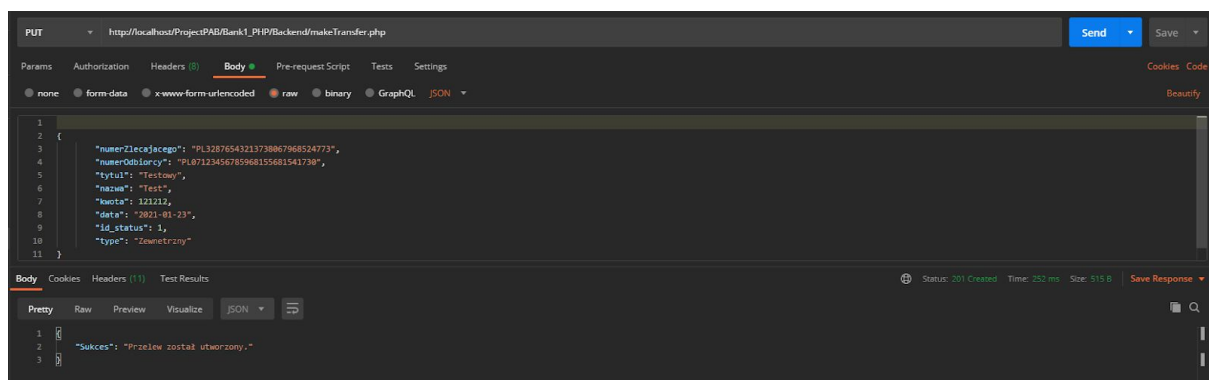
Funkcja makeTransfer służy do wykonania zapytania poprzez API. Wszystkie pola są zabezpieczone przed specjalnymi znakami HTML'a. Wywołanie zapytania powoduje dodanie przelewu do historii oraz odjęcie / dodanie kwoty od salda użytkownika. Na poziomie UI pola są zabezpieczone skryptem napisanym w JS. W API sprawdzana jest poprawność danych oraz czy podane dane nie są puste.

4.3.4. API - Wszystkie przelewy

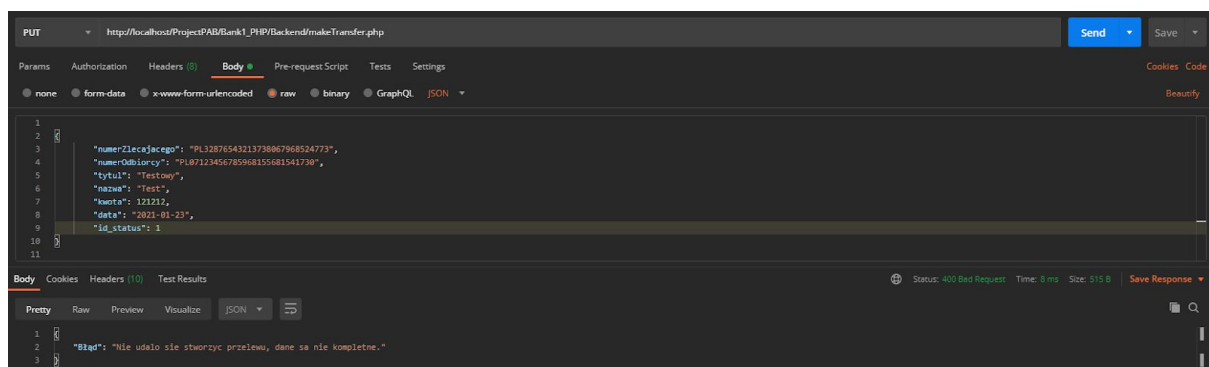


Zapytanie które zwróci wszystkie przelewy które znajdują się w bazie banku A.

4.3.5. API - Realizowanie przelewu

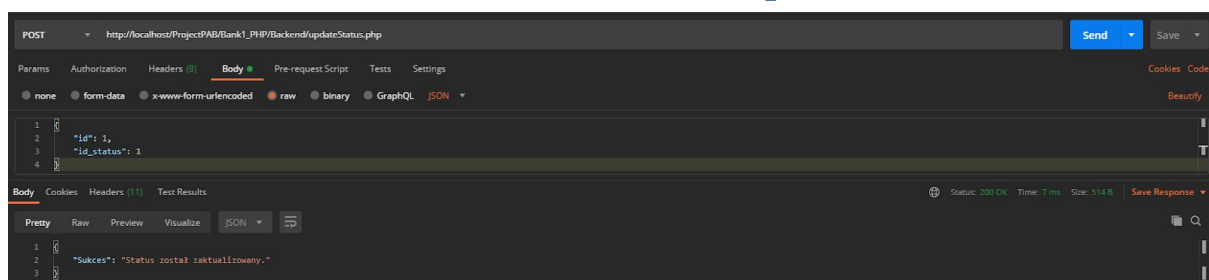


W przypadku gdy wszystkie dane są podane i są poprawne przelew zostanie zrealizowany, zostanie zwrócony status 201 oraz komunikat o sukcesie.



W przypadku błędnych danych lub braku niektórych z nich (w tym przypadku typu) zostanie zwrócony status 400 oraz komunikat o błędzie.

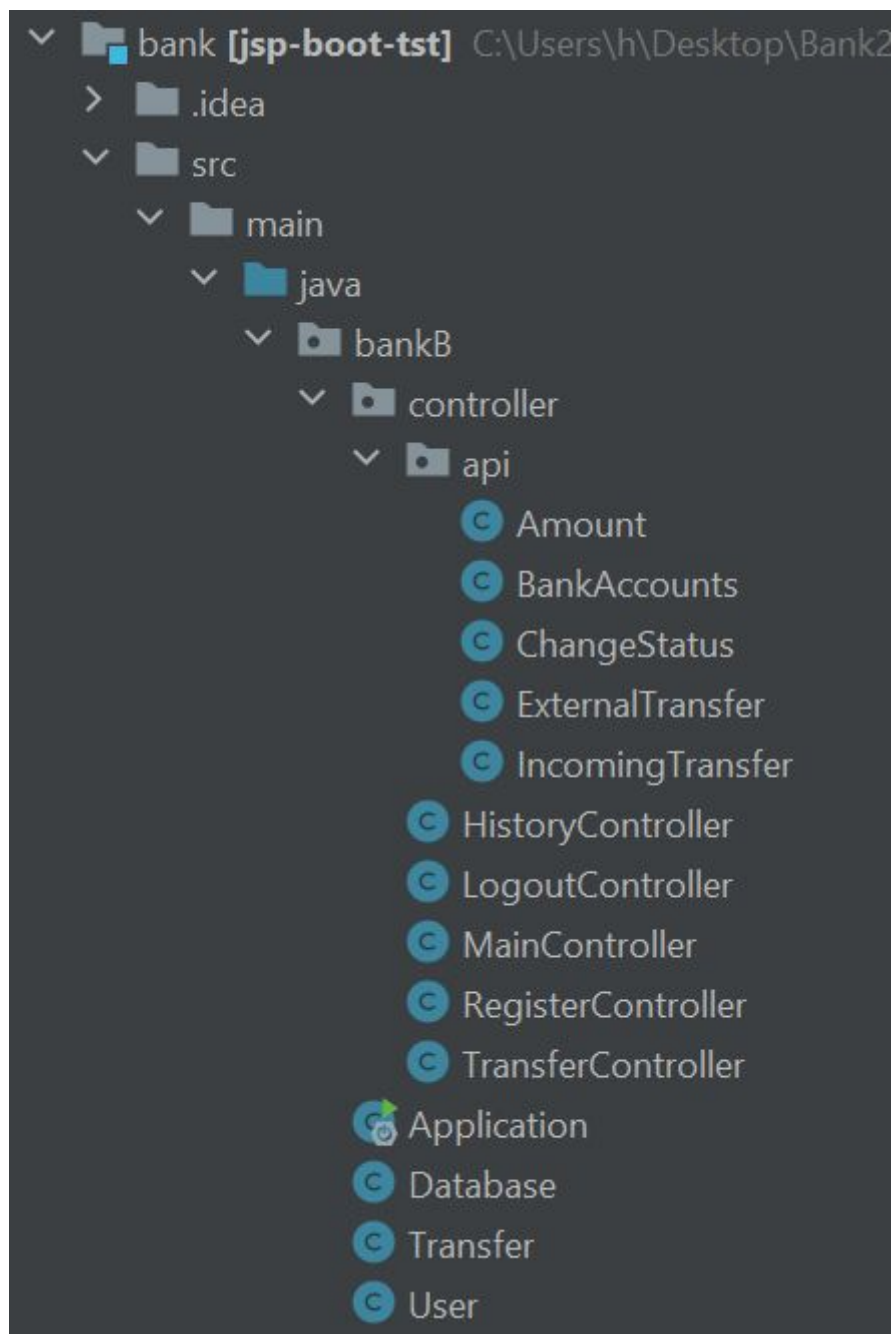
4.3.6. API - aktualizowanie statusu przelewu

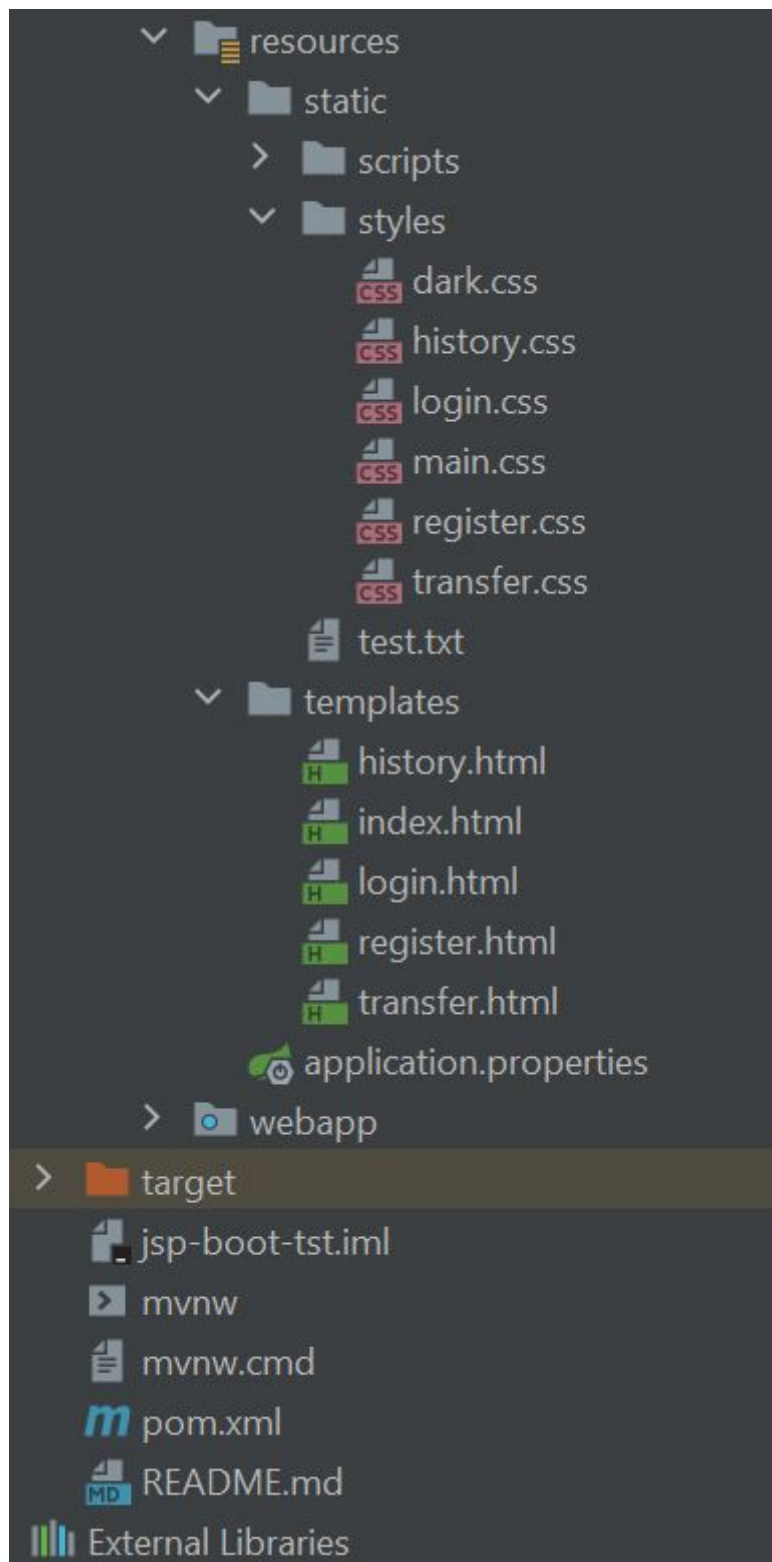


Gdy zostaną podane wszystkie dane potrzebne do zaktualizowania statusu zostanie zwrócony status 200 oraz komunikat o pomyślnej zmianie statusu. W przeciwnym wypadku zostanie zwrócony błąd.

5. Bank B - frontend i backend

5.1. Struktura projektu






5.2. Interfejs

5.2.1. Rejestracja

BardzoZły

Bank

 Dane logowania

Login


Login

Hasło

Hasło

Powtórz hasło

Powtórz hasło

 Dane osobowe

Imię

Imię

Nazwisko

Nazwisko

PESEL


PESEL

Adres e-mail

E-mail

Telefon komórkowy

Telefon

 Adres zamieszkania

Miejscowość

Miejscowość

Ulica


Ulica

Numer domu

Numer domu

Kod pocztowy

Kod pocztowy

 Oświadczenia

Wyrażam zgodę na przetwarzanie przez Bank podanych przeze mnie danych kontaktowych

Jestem świadomy(-ma) odpowiedzialności karnej za złożenie fałszywego oświadczenia

Wyrażam zgodę na używanie przez Bank połączenia telefonicznego

Wróć

Załącz konto

Panel rejestracji pozwala na założenie konta nowego użytkownika w Banku B.
Każde pole jest sprawdzane przez skrypty napisane w JavaScript.

Login

test123

- *Przynajmniej sześć znaków ✓*
- *Musi zawierać tylko litery oraz cyfry ✓*

Hasło

●●●●●●●●

- *Przynajmniej osiem znaków ✓*
- *Musi zawierać symbol specjalny*

Powtórz hasło

●●●●●●●● |

- *Hasła muszą być takie same*

W przypadku poprawnego wpisania i spełnienia wszystkich warunków - pole zaświeci się na zielono. Wymogi znajdują się pod danym polem, zmieniają się i informują użytkownika o poprawności w czasie rzeczywistym. Gdy użytkownik nie spełnia wymogów pole zaświeci się na czerwono informując o błędzie.

Jeżeli użytkownik naciśnie guzik “Zarejestruj” aplikacja przeniesie go do najwyższego błędnie wypełnionego pola.

5.2.2. Logowanie

BardzoZły
Bank

Logowanie do banku

Login

Hasło

Zaloguj

Załóż darmowe konto

STOPKA COPYRAIT @ NIE MOJE

Mamy tutaj dostęp do rejestracji konta oraz możliwość zalogowania się do banku.

5.2.3. Strona główna - po zalogowaniu

BardzoZły
Bank

Wyloguj

Strona główna Przelewy Historia

Robert Kubica

Konto osobiste

Mastercard Konto	Saldo bieżące	Dostępne środki	Zablokowane środki
876543216954738276055023	870521.0 PLN	0.0 PLN	0.0 PLN

STOPKA COPYRAIT @ NIE MOJE

Po zalogowaniu użytkownik widzi swoje konto, w prawym rogu wyświetlane są informacje o użytkowniku (imię i nazwisko). Widać również aktualne saldo na koncie.

5.2.4. Przelewy

BardzoZły Bank

Wyloguj

Strona główna Przelewy Historia Robert Kubica

Przelewy

Z konta

Mastercard konto (870488.0 PLN)
876543216954738276055023

Dane odbiorcy przelewu

Nazwa odbiorcy

Numer konta

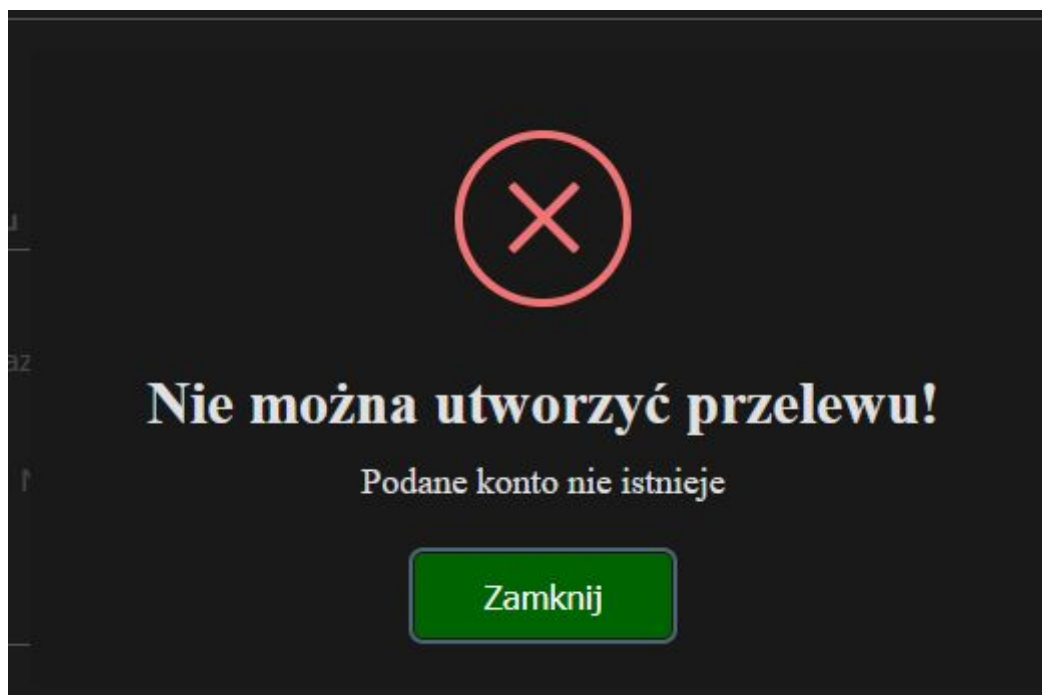
Szczegóły

Tytuł przelewu

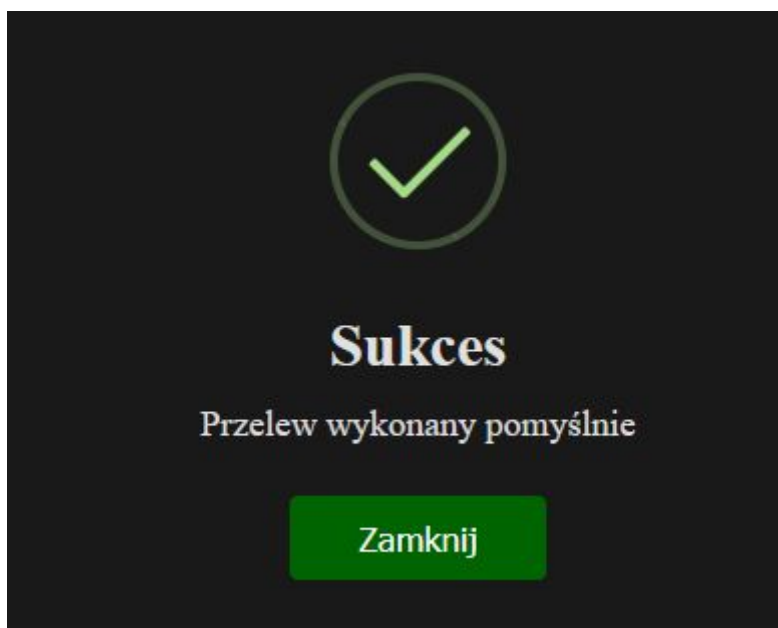
Kwota PLN

Musi zawierać tylko cyfry✓
Musisz mieć taką kwotę na koncie✓
Więcej niż zero✓

W zakładce “Przelewy” użytkownik może wykonywać przelewy zewnętrzne jak i wewnętrzne po uprzednim poprawnym uzupełnieniu wymaganych pól.



W przypadku próby wysłania przelewu na nieistniejące konto zostaniemy o tym powiadomieni i przelew się nie wykona



Gdy wszystkie dane są poprawne wyskakuje alert z informacją o sukcesie.

5.2.5. Historia

BardzoZły

Bank

Wyloguj

Strona główna

Przelewy

Historia

Robert Kubica

Data przelewu	Tytuł przelewu	Odbiorca	Na numer konta	Kwota
2021-02-07	Test przelewu zewnętrznego...	UzytkownikInnego...	50123456787156635103312696	-999.0
<div>Tytuł przelewu: Test przelewu zewnętrznego bez akceptacji</div> <div>Odbiorca: UzytkownikInnegoBanku</div> <div>Z numeru konta: 77876543216680888028561231</div> <div>Na numer konta: 50123456787156635103312696</div> <div>Kwota transakcji: - 999.0</div>				
2021-02-07	Test przelewu zewnętrznego...	Stefan	50123456787156635103312696	-1234.0
2021-02-07	dsada	asdasd	45123456787998223613645938	-999.0
2021-02-07	sdada	dasda	45123456787998223613645938	-1001.0
2021-02-07	dada	adasd	45123456787998223613645938	-999.0
2021-02-07	dsada	dasd	45123456787998223613645938	-999.0

W panelu “Historia” znajdują się przelewy wychodzące i przychodzące, każdy przelew można z osobna rozwinąć i uzyskać dodatkowe informacje i w przypadku gdy tytuł lub odbiorca jest za długi wyświetli się całość.

5.3. Backend

5.3.1. Config

```
public class Database {  
  
    public static Connection getConnection() {  
        try {  
            Class.forName("com.mysql.jdbc.Driver");  
        } catch (ClassNotFoundException ex) {  
            ex.printStackTrace();  
        }  
        Connection connection = null;  
        try {  
            connection = DriverManager.getConnection(  
                "jdbc:mysql://localhost:3306/bank2", "s1: \"root\", \"s2: \"");  
        } catch (SQLException e) {  
            e.printStackTrace();  
        }  
        return connection;  
    }  
  
    private static final String bankCode = "87654321";  
    private static final String bankCountry = "PL";  
    private static final String incomingTransferURL = "http://localhost/J_Rozliczajaca/BACKEND/readZrealizo";  
    private static final String getIncomingTransferURL = "http://localhost/J_Rozliczajaca/UI/html/zmianasta";  
    private static final String accountsURL = "http://localhost/J_Rozliczajaca/BACKEND/readKonta.php";  
}
```

Znajduje się tutaj funkcja łącząca z bazą MYSQL, która zwraca Connection. Dodatkowo znajdują się tutaj wszystkie zmienne, które są możliwe do edycji.

5.3.2. Generowanie numeru konta

```
int kodKraju = 252100; //PL00  
String numerKonta0 = Database.getBankCode() + ThreadLocalRandom.current().nextLong(1, 1000000000000000L, 11: 999999999999999L) + kodKraju;  
String czescNumeru = numerKonta0.substring(1, 5);  
int moduloCzesci = 0;  
for(int i = 5; i < 30; i += 5){  
    {  
        moduloCzesci = Integer.parseInt(czescNumeru) % 97;  
        czescNumeru = moduloCzesci + numerKonta0.substring(i, i + 5);  
    }  
}  
moduloCzesci = 98 - Integer.parseInt(czescNumeru) % 97;  
if(moduloCzesci < 10) czescNumeru = "0" + moduloCzesci;  
else czescNumeru = String.valueOf(moduloCzesci);  
String numerKonta = Database.getBankCountry() + czescNumeru + numerKonta0.substring(1, 24);
```

Funkcja ta liczy sumę kontrolną numeru rachunku według standardu IBAN. Działanie algorytmu jest następujące: tworzymy numer konta bez sumy kontrolnej: “PL” + 8 cyfr banku + 16 losowych cyfr rachunku, a następnie przenosimy litery “PL” na koniec numeru i zamieniamy je na wartość liczbową

(w tym przypadku 2521) oraz dodajemy “00” jako numer kontrolny. Z tak otrzymanej liczby obliczamy resztę z dzielenia przez 97. Aby uniknąć działań na ogromnych liczbach dzielimy liczbę na pięciocyfrowe kawałki i z nich obliczamy resztę z dzielenia. Po otrzymaniu wyniku odejmujemy go od liczby 98. Jeśli wynik wyszedł mniejszy niż 10 dodajemy przed nim “0”. Tak otrzymaną cyfrę kontrolną możemy wpisać pomiędzy kod kraju a kod banku.

5.3.3. Funkcja do tworzenia JSONA z przelewami

```
String json = "";
try {
    Connection conn = Database.getConnection();
    Statement stmt = conn.createStatement();
    ResultSet rs = stmt.executeQuery("SELECT * FROM historia WHERE id_status=2 AND type='Zewnetrzny'");

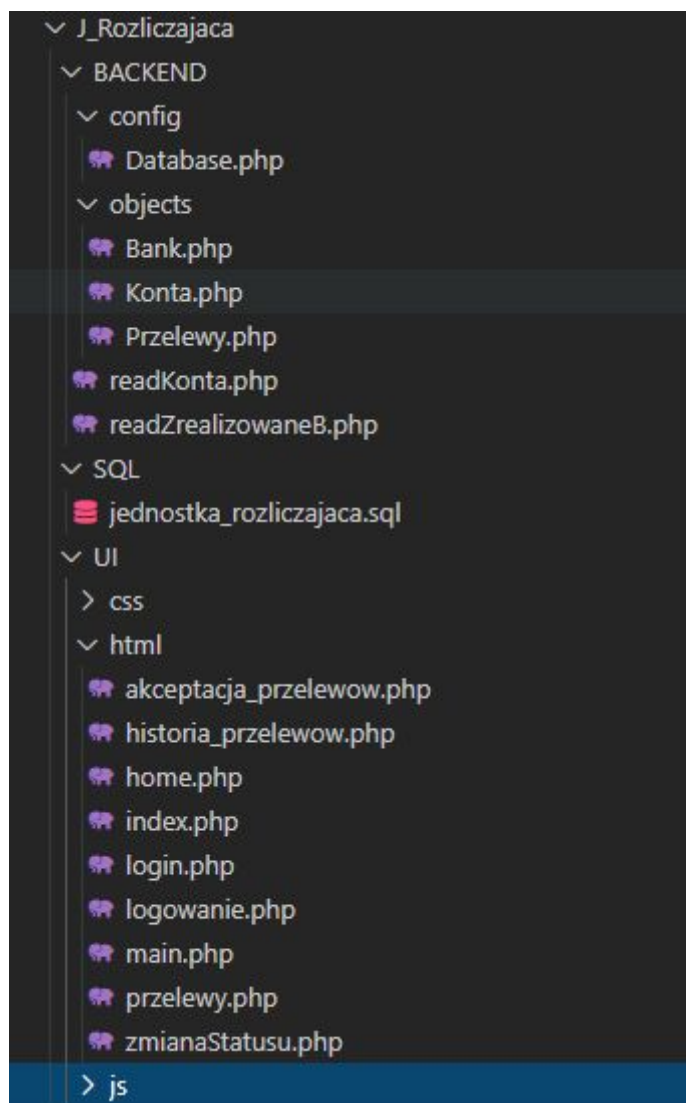
    JSONObject jsonObject = new JSONObject();
    JSONArray array = new JSONArray();
    while (rs.next()) {
        JSONObject transfer = new JSONObject();

        transfer.put("id", rs.getInt(1));
        transfer.put("numerZlecajacego", rs.getString(2));
        transfer.put("numerOdbiorcy", rs.getString(3));
        transfer.put("tytul", rs.getString(4));
        transfer.put("nazwa", rs.getString(5));
        transfer.put("kwota", rs.getDouble(6));
        transfer.put("data", rs.getDate(7).toString());
        transfer.put("id_status", rs.getInt(8));
        array.put(transfer);
    }
    jsonObject.put("Przelewy", array);
    json = jsonObject.toString();
} catch (SQLException e) {
    e.printStackTrace();
}
```

Funkcja pobiera wszystkie oczekujące przelewy zewnętrzne z bazy i tworzy JSON o tytule “Przelewy”, który następnie jest obsługiwany przez jednostkę rozliczającą.

6. Jednostka rozliczeniowa

6.1. Struktura projektu



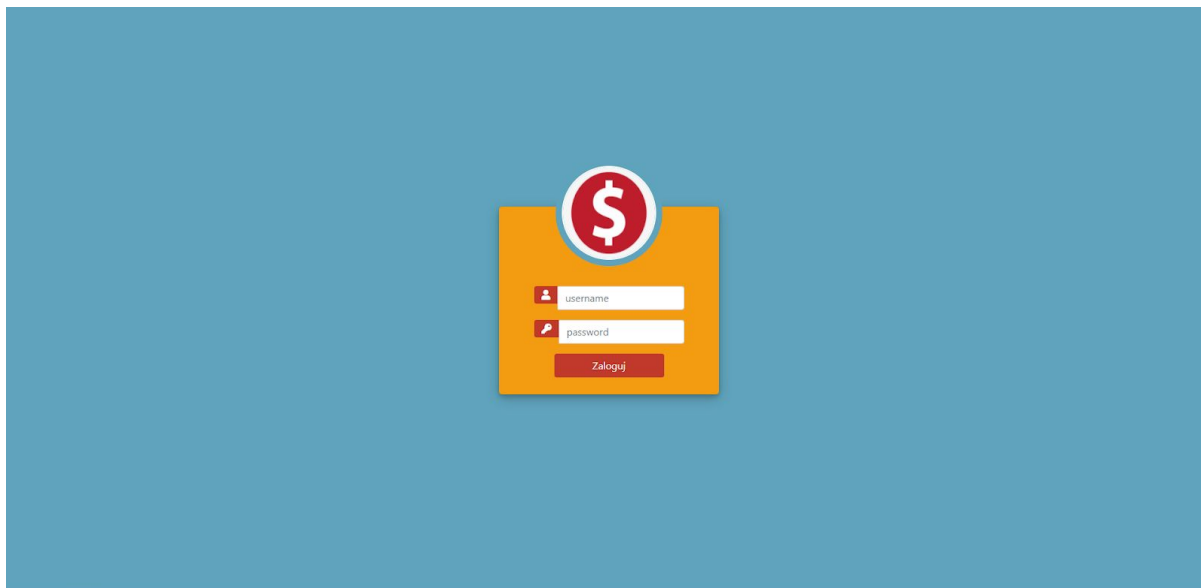
Struktura jednostki składa się z trzech głównych katalogów:

- BACKEND - pliki konfiguracyjne bazy danych, obiekty klas, endpointy API jednostki rozliczającej
- SQL - wyeksportowana baza danych JR
- UI - zawiera pliki php witryn internetowych oraz katalog css

W katalogach css oraz js znajdują się pliki Bootstrapa, ale ze względu na obszerność, nie zostały one rozwinięte.

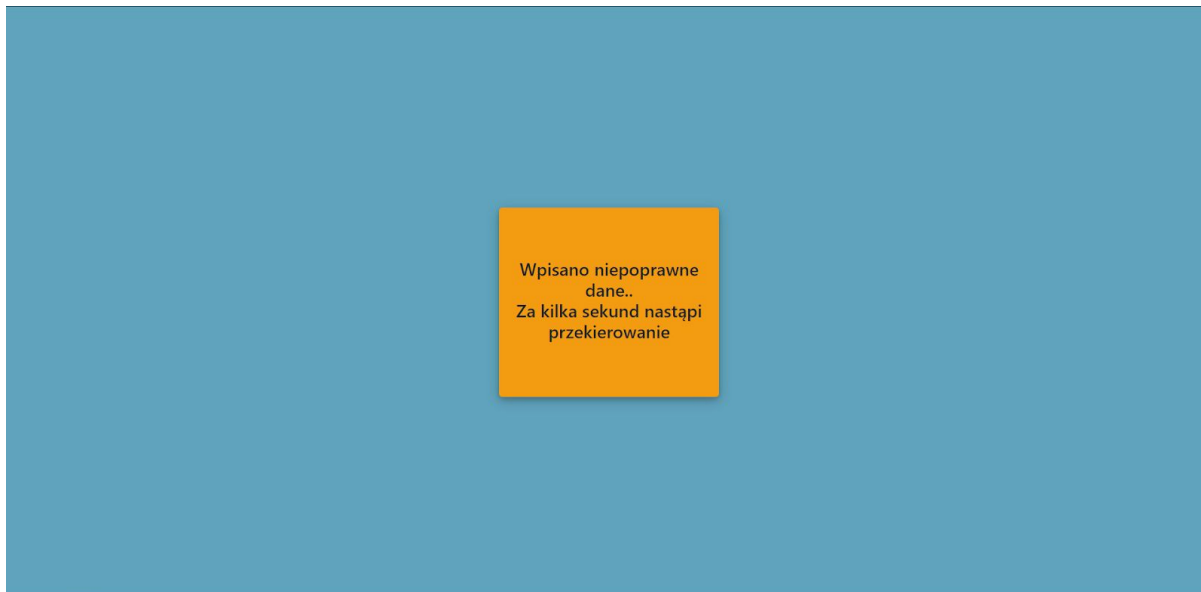
6.2. Interfejs

6.2.1. Panel logowania



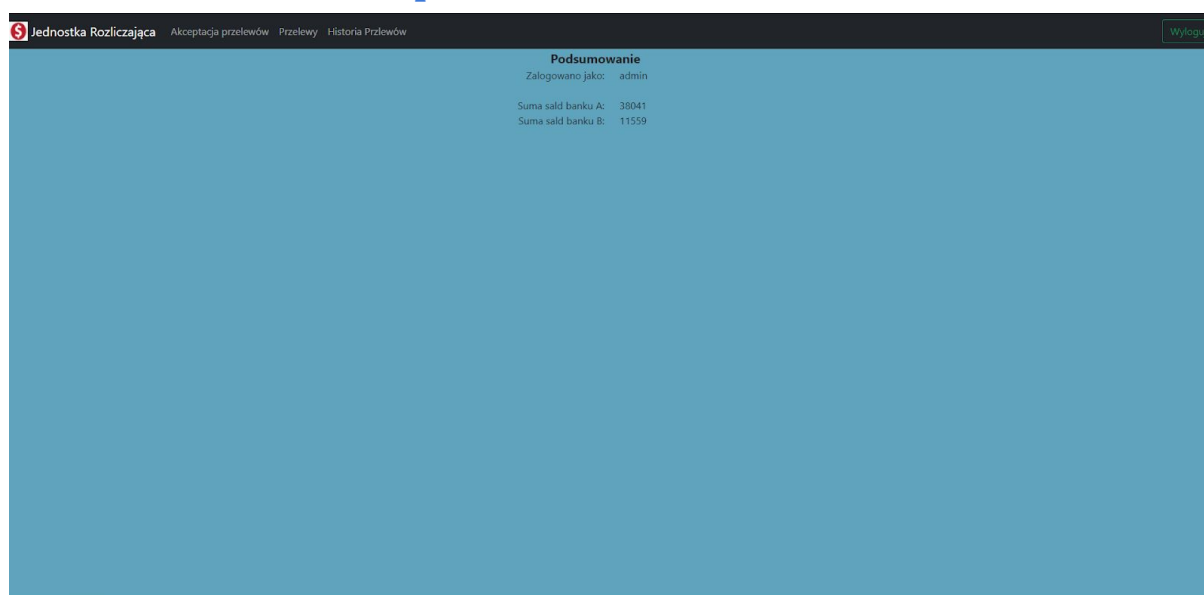
Aby administrator jednostki się zalogował musi podać poprawne dane logowania. W przypadku podania niepoprawnych danych wypisany zostanie odpowiedni komunikat.

6.2.2. Nieudane logowanie



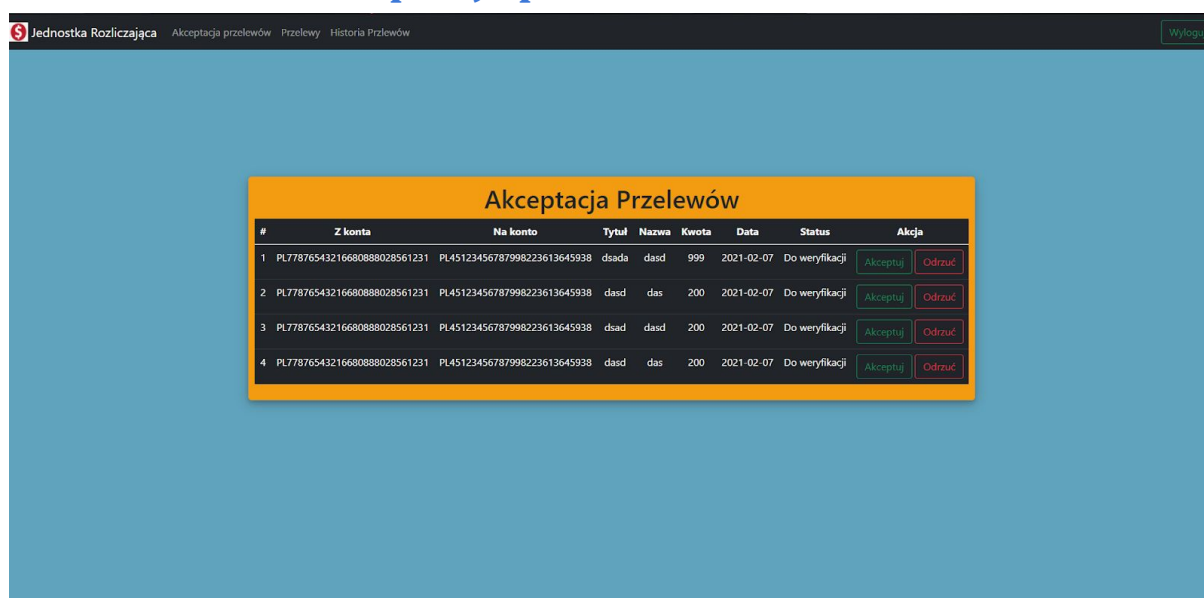
Wcześniej wspomniany komunikat o niepoprawnych danych logowania.

6.2.3. Panel podsumowania



W tym miejscu mamy informację o stanie konta podłączonych do jednostki banków oraz nazwa zalogowanego administratora.

6.2.4. Akceptacja przelewów



Na tej stronie administrator jednostki może zaakceptować lub odrzucić przelew który został przydzielony do kolejki przelewów do zaakceptowania

6.2.5. Lista przelewów do automatycznego zrealizowania

#	Bankowe ID	Z konta	Na konto	Tytuł	Nazwa	Kwota	Data
Brak przelewów do wyświetlenia.							

#	Bankowe ID	Z konta	Na konto	Tytuł	Nazwa	Kwota	Data
1	299	PL7787654321668088028561231	PL45123456787998223613645938	dasd	das	200	2021-02-07

[Odpal sesję](#)

W tym miejscu znajdują się dwie tabele przelewów, które wykonają się przy następnym uruchomieniu sesji.

6.2.6. Historia przelewów

#	Z konta	Na konto	Tytuł	Nazwa	Kwota	Data	Status
BANK A							
1	PL45123456787998223613645938	PL7787654321668088028561231	dasdasd	dasdas	1001	2021-02-07	Niezrealizowany
2	PL50123456787156635103312696	PL7787654321668088028561231	Test przelewu do odrzucenia	Uzytkownik	1234	2021-02-07	Niezrealizowany
BANK B							
3	PL7787654321668088028561231	PL50123456787156635103312696	Test przelewu zewnętrznego bez akceptacji	UzytkownikInnegoBanku	999	2021-02-07	Zrealizowany
4	PL7787654321668088028561231	PL50123456787156635103312696	Test przelewu zewnętrznego zweryfikacja	Stefan	1234	2021-02-07	Zrealizowany
5	PL7787654321668088028561231	PL45123456787998223613645938	dsada	asdasd	999	2021-02-07	Zrealizowany
6	PL7787654321668088028561231	PL45123456787998223613645938	sdada	dasda	1001	2021-02-07	Zrealizowany
7	PL7787654321668088028561231	PL45123456787998223613645938	dada	adasd	999	2021-02-07	Zrealizowany
8	PL7787654321668088028561231	PL45123456787998223613645938	dasd	dsad	200	2021-02-07	Zrealizowany
9	PL7787654321668088028561231	PL45123456787998223613645938	dsad	dasd	220	2021-02-07	Zrealizowany
10	PL7787654321668088028561231	PL45123456787998223613645938	dasd	dsa	300	2021-02-07	Zrealizowany
11	PL7787654321668088028561231	PL45123456787998223613645938	dsad	dsad	200	2021-02-07	Zrealizowany
12	PL7787654321668088028561231	PL45123456787998223613645938	dasd	dasd	250	2021-02-07	Zrealizowany
13	PL7787654321668088028561231	PL45123456787998223613645938	dsad	dasd	240	2021-02-07	Zrealizowany

Na tej stronie znajdują się przelewy które zostały dotychczas rozliczone przez jednostkę (zaakceptowane lub odrzucone).

6.3. Wyświetlanie poszczególnych podstron

```
<?php
$current_page = isset($_GET['page']) ? $_GET['page'] : null;

switch ($current_page){

    case 'home':
        default;
        include $_SERVER['DOCUMENT_ROOT'].'/J_rozliczajaca/UI/html/home.php';
        break;

    case 'akceptacja_przelewow':
        include $_SERVER['DOCUMENT_ROOT'].'/J_rozliczajaca/UI/html/akceptacja_przelewow.php';
        break;

    case 'przelewy':
        include $_SERVER['DOCUMENT_ROOT'].'/J_rozliczajaca/UI/html/przelewy.php';
        break;

    case 'historia_przelewow':
        include $_SERVER['DOCUMENT_ROOT'].'/J_rozliczajaca/UI/html/historia_przelewow.php';
        break;

}
?>
```

6.4. Wylogowywanie

```
<?php
if($_SERVER['REQUEST_METHOD'] == "POST" and isset($_POST['logout']))
{
    func();
}
function func()
{
    session_destroy();
    redirect('index.php?page=login');
    exit;
}
function redirect($url)
{
    if (!headers_sent())
    {
        header('Location: '.$url);
        exit;
    }
    else
    {
        echo '<script type="text/javascript">';
        echo 'window.location.href="'.$url.'";';
        echo '</script>';
        echo '<noscript>';
        echo '<meta http-equiv="refresh" content="0;url='.$url.'" />';
        echo '</noscript>'; exit;
    }
}
?>
```


6.5. Algorytm

Zadaniem algorytmu jest rozdzielenie przelewów przychodzących, na przelewy rozliczane automatycznie w przypadku wystąpienia sesji rozliczeniowej oraz na przelewy wymagające ręcznej akceptacji.

```
<?php
//PRZELEWY Z BANK A
$curl_handle=curl_init();
curl_setopt($curl_handle, CURLOPT_URL, 'http://localhost/Bank1_PHP/Backend/readTransfer.php');
curl_setopt($curl_handle, CURLOPT_CONNECTTIMEOUT, 2);
curl_setopt($curl_handle, CURLOPT_RETURNTRANSFER, 1);
$json = curl_exec($curl_handle);

if($json[3] != '\\'){

    $json = json_decode($json);

    $kwota_checkPowtorka = 0;
    $data_checkPowtorka = '';
    $numerZlecajacego_checkPowtorka = '';
    $numerOdbiorcy_checkPowtorka = '';

    $iterator_tabeli = 0;
    foreach ($json->Przelewy as $key => $val) {
        if($kwota_checkPowtorka == $val->kwota && $data_checkPowtorka == $val->data && $numerZlecajacego_checkPowtorka == $val->numerZlecajacego
        && $numerOdbiorcy_checkPowtorka == $val->numerOdbiorcy){
            $przelewy->registerPrzelewDoWeryfikacji($val->id, $val->numerZlecajacego, $val->numerOdbiorcy, $val->tytul, $val->nazwa, $val->kwota, $val->data);
            continue;
        }else{
            $kwota_checkPowtorka = $val->kwota;
            $data_checkPowtorka = $val->data;
            $numerZlecajacego_checkPowtorka = $val->numerZlecajacego;
            $numerOdbiorcy_checkPowtorka = $val->numerOdbiorcy;
        }
    }
    $result = $przelewy->readPoNumerzeZrealizowanych($val->numerZlecajacego);
    $num = $result->rowCount();
    if($num > 0){
        if($val->kwota > 1000){
            if($num >= 5){
                $przelewy_arr = array();
                $przelewy_arr['data'] = array();
                $suma_przelewow = 0;
                $licznik = 0;
                $najwiekszy_przelew = 0;
                while($row = $result->fetch(PDO::FETCH_ASSOC)){
                    extract($row);
                    if($kwota>$najwiekszy_przelew){
                        $najwiekszy_przelew=$kwota;
                    }
                }
                $licznik++;
            }
        }
    }
}
```



```

        $suma_przelewow = $suma_przelewow + $kwota;
    }
    $srednia = $suma_przelewow / $licznik;
    $kwota_graniczna = $srednia + $najwiekszy_przelew;
    if($kwota_graniczna < 1000){
        $kwota_graniczna = 1000;
    }
    if($kwota_graniczna < $val->kwota){
        $przelewy->registerPrzelewDoWeryfikacji($val->id, $val->numerZlecajacego, $val->numerOdbiorcy, $val->tytul, $val->nazwa, $val->kwota, $val->data);
        //echo "Wyląduje ze statusem 4 do weryfikacji ręcznej bo kwota przekracza algorytm<br>";
        continue;
    }else{
        $przelewy->registerPrzelew($val->id, $val->numerZlecajacego, $val->numerOdbiorcy, $val->tytul, $val->nazwa, $val->kwota, $val->data);
        //echo "Przejdzie<br>";

        $url = 'http://localhost/Bank1_PHP/Backend/updateStatus.php';

        $status = '1';
        $type = 'Zewnetrzny';

        $data = array(
            'id' => $val->id,
            'id_status' => $status
        );

        $content = json_encode($data);

        $curl = curl_init($url);

        curl_setopt($curl, CURLOPT_HEADER, false);
        curl_setopt($curl, CURLOPT_RETURNTRANSFER, true);
        curl_setopt($curl, CURLOPT_HTTPHEADER,
            array("Content-type: application/json"));
        curl_setopt($curl, CURLOPT_POST, true);
        curl_setopt($curl, CURLOPT_POSTFIELDS, $content);

        $json_response = curl_exec($curl);
        curl_close($curl);

        $urlPrzelewDoB = 'http://localhost:8080/incomingTransfers';

        $curl = curl_init($urlPrzelewDoB);

        $json_response = curl_exec($curl);
        curl_close($curl);
    }
}

```

```

    }else{
        $przelewy->registerPrzelewDoWeryfikacji($val->id, $val->numerZlecajacego, $val->numerOdbiorcy, $val->tytul, $val->nazwa, $val->kwota, $val->data);
        //echo "Wyląduje ze statusem 4 do weryfikacji ręcznej bo za mała historia<br>";
        continue;
    }
}
}else{
    $przelewy->registerPrzelew($val->id, $val->numerZlecajacego, $val->numerOdbiorcy, $val->tytul, $val->nazwa, $val->kwota, $val->data);

    $url = 'http://localhost/Bank1_PHP/Backend/updateStatus.php';

    $status = '1';
    $type = 'Zewnetrzny';

    $data = array(
        'id' => $val->id,
        'id_status' => $status
    );

    $content = json_encode($data);

    $curl = curl_init($url);

    curl_setopt($curl, CURLOPT_HEADER, false);
    curl_setopt($curl, CURLOPT_RETURNTRANSFER, true);
    curl_setopt($curl, CURLOPT_HTTPHEADER,
        array("Content-type: application/json"));
    curl_setopt($curl, CURLOPT_POST, true);
    curl_setopt($curl, CURLOPT_POSTFIELDS, $content);

    $json_response = curl_exec($curl);
    curl_close($curl);

    $urlPrzelewDoB = 'http://localhost:8080/incomingTransfers';

    $curl = curl_init($urlPrzelewDoB);

    $json_response = curl_exec($curl);
    curl_close($curl);
}
}
}else{

```

```

if($val->kwota > 1000){
    $przelewy->registerPrzelewDoWeryfikacji($val->id, $val->numerZlecajacego, $val->numerOdbiorcy, $val->tytul, $val->nazwa, $val->kwota, $val->data);
    //echo "Wyladuje ze statusem 4 do weryfikacji ręcznej bo brak historii<br>";
    continue;
}else{
    $przelewy->registerPrzelew($val->id, $val->numerZlecajacego, $val->numerOdbiorcy, $val->tytul, $val->nazwa, $val->kwota, $val->data);

    $url = 'http://localhost/Bank1_PHP/Backend/updateStatus.php';

    $status = '1';
    $type = 'Zewnetrzny';

    $data = array(
        'id' => $val->id,
        'id_status' => $status
    );

    $content = json_encode($data);

    $curl = curl_init($url);

    curl_setopt($curl, CURLOPT_HEADER, false);
    curl_setopt($curl, CURLOPT_RETURNTRANSFER, true);
    curl_setopt($curl, CURLOPT_HTTPHEADER,
        array("Content-type: application/json"));
    curl_setopt($curl, CURLOPT_POST, true);
    curl_setopt($curl, CURLOPT_POSTFIELDS, $content);

    $json_response = curl_exec($curl);
    curl_close($curl);

    $urlPrzelewDoB = 'http://localhost:8080/incomingTransfers';

    $curl = curl_init($urlPrzelewDoB);

    $json_response = curl_exec($curl);
    curl_close($curl);
}
}

$iterator_tabeli++;
?>
<tr>
<td><?php echo $iterator_tabeli; ?></td>
<td><?php echo $val->id; ?></td>
<td><?php echo $val->numerZlecajacego; ?></td>
<td><?php echo $val->numerOdbiorcy; ?></td>
<td><?php echo $val->tytul; ?></td>
<td><?php echo $val->nazwa; ?></td>
<td><?php echo $val->kwota; ?></td>
<td><?php echo $val->data; ?></td>
</tr>
<?php
}
}
}else{
?>
<tr>
<td colspan=8>Brak przelewów do wyświetlenia.</td>
</tr>
<?php
curl_close($curl_handle);
}
?>

```

Zasada działania algorytmu:

Odbierane są przelewy wychodzące z banku A, następnie sprawdzane jest, czy jakiegokolwiek przelewy zewnętrzne zostały wykonane. Jeżeli nie to wyświetlany jest odpowiedni komunikat. Jeżeli natomiast odebrano przelewy zewnętrzne to rozpoczyna się działanie algorytmu sortującego przelewy.

Jesteśmy w pętli która wykonuje się dla każdego przelewu wychodzącego z banku A. Sprawdzamy czy przelew nie ma takich samych danych jak poprzednio wysłany przelew, aby zminimalizować ryzyko wystąpienia powtórki. Jeżeli algorytm wykryje taką sytuację, następuje wysłanie potencjalnie zduplikowanego przelewu do kolejki przelewów do ręcznej weryfikacji (inaczej akceptacji przelewów). Jeżeli nie to przechodzimy dalej.

Pobieramy listę wszystkich dotychczas wykonanych przelewów użytkownika banku A. Jeżeli ilość tych przelewów jest mniejsza niż 5, to każdy przelew powyżej 1000 zł, zostanie oznaczony jako przelew wymagający ręcznej

weryfikacji. Jeżeli w historii użytkownika widnieje co najmniej 5 przelewów, to algorytm wybiera największy przelew z historii użytkownika oraz przypisuje go do zmiennej “największy przelew”, następnie oblicza średnią wszystkich przelewów oraz tworzy zmienną “kwota graniczna” która jest sumą średniej wszystkich przelewów oraz kwoty największego przelewu. Kolejno sprawdzane jest czy wcześniej wyliczona kwota graniczna jest mniejsza niż 1000 zł. Jeżeli tak to do zmiennej “kwota graniczna” przypisana jest wartość 1000 odpowiadająca 1000 zł. Ostatecznie jeżeli kwota przelewu nie przekracza wyliczonej kwoty granicznej to wybrany przelew przechodzi do kolejki przelewów realizowanych automatycznie, w przeciwnym wypadku przelew przechodzi do kolejki przelewów wymagających ręcznej weryfikacji.