# Pay With Friends

# Teoretyczne podstawy działania

Spis treści

[1. Oznaczenia 2](#_Toc353030707)

[1.1. Definicje 2](#_Toc353030708)

[1.2. Przykład transakcji 2](#_Toc353030709)

[2. Użycie oznaczeń w programie 3](#_Toc353030710)

[3. Możliwe akcje użytkowników 3](#_Toc353030711)

[3.1. Ogólnie 3](#_Toc353030712)

[3.2. Dodawanie nowej transakcji 4](#_Toc353030713)

[3.3. Optymalizacja transakcji grupy użytkowników 4](#_Toc353030714)

[3.4 Globalna optymalizacja transakcji 4](#_Toc353030715)

[4. Bezpieczeństwo użytkowników 5](#_Toc353030716)

[4.1. OAuth – autoryzacja użytkownika w aplikacji 5](#_Toc353030717)

[4.2. Szyfrowanie danych 5](#_Toc353030718)

[4.3. Bezpieczeństwo bazy danych 5](#_Toc353030719)

[5. Serwer HTTP/HTTPS 6](#_Toc353030720)

[5.1. Dynamiczny adres IP serwera 6](#_Toc353030721)

[5.2. Server www 6](#_Toc353030722)

[5.3. Serwer bazy danych 6](#_Toc353030723)

[6. Baza danych 7](#_Toc353030724)

[6.1. Tabele 7](#_Toc353030725)

[6.2. Opis tabel 7](#_Toc353030726)

[6.3. Użycie bazy danych w aplikacji 7](#_Toc353030727)

## Oznaczenia

### Definicje

– zbiór wierzchołków grafu (wszyscy użytkownicy)

– i-ty wierzchołek (i-tyużytkownik)

– liczba użytkowników

– bilans wierzchołku i (i-tego użytkownika),

– zbiór krawędzi grafu (wszystkie połączenia)

– krawędź pomiędzy wierzchołkami „i” i „j” (przyjaźń i-tego i j-tego użytkownika), kolejność i,j nie ma znaczenia, ponieważ graf jest symetryczny

– niewypełniona płatność od użytkownika „i” do użytkownika „j”,

### Przykład transakcji



Dla sytuacji przedstawionej na obrazku:

* wierzchołki (użytkownicy):
* krawędzie (przyjaźnie):
* bilansy wierzchołków (użytkowników):
* (po zaakceptowaniu transakcji przez wszystkich użytkowników)  
  niewypełnione płatności użytkowników:

Praktyczny sens sytuacji: „użytkownik 2” jest przyjacielem użytkowników „1” i „3”. Użytkownik 2 jest winien „użytkownikowi 1” 5 jednostek, a „użytkownikowi 3” 15 jednostek.

## Użycie oznaczeń w programie

**Wierzchołek grafu – użytkownik**. Każdy z użytkowników jest identyfikowany 15 cyfrowym numerem ID (który jest identyczny co Facebookowy numer ID użytkownika).

**Krawędź grafu – przyjaźń użytkowników.** Tylko użytkownicy będący przyjaciółmi (w Pay With Friends – nie w Facebooku, tak aby użytkownik mógł decydować z kim chce mieć powiązania finansowe).

**Bilans wierzchołka –** **bilans użytkownika w transakcji.** Po dodaniu nowej transakcji dla każdego użytkownika transakcji wyliczony jest jego bilans w tej transakcji, na tej podstawie będą potem wyliczone płatności pomiędzy użytkownikami.

## Możliwe akcje użytkowników

### 3.1. Ogólnie

**Stworzenie użytkownika** – stworzenie wierzchołka reprezentującego daną osobę w aplikacji. Zarówno złożoność obliczeniowa jak i pamięciowa tej operacji wynosi .

**Dodanie/potwierdzenia przyjaźni** – stworzenie krawędzi pomiędzy dwoma użytkownikami w aplikacji. Zarówno złożoność obliczeniowa jak i pamięciowa tej operacji wynosi .

**Dodanie nowej transakcji** – zmiana bilansów pewnej grupu użytkowników (która będzie miała miejce dopiero po zaakceptiowaniu transakcji przez wszystkich użytkowników). Bilansy zostaną od razu przekształcone w płatności pomiędzy użytkownikami (jeśli nie jest to możliwe(brak krawędzi pomiędzy użytkownikami) transakcja nie będzie mogła być zaakceptowana).

**Optymalizacja transakcji grupy użytkowników** – zmiana płatności pomiędzy pewną grupą wierzchołków tak, aby zminimalizować pewien wskaźnik transakcji (np. ilość niezerowych płatności, suma wartości płatności).

**Globalna optymalizacja transakcji** – analogiczna optymalizacja przeprowadzona pomiędzy wszystkimi wierzchołkami.

### 3.2. Dodawanie nowej transakcji

Akcje potrzebne do dodania nowej transakcji (n – liczba uczestników transakcji):

* obliczenie bilansu każdego użytkownika (proste działania artymetyczne) – złożoność
* stworzenie SP (Spanning Tree) - drzewa spinającego wszystkich użytkowników po dostępnych krawędziach (przy zastosowaniu algorytmu BFS (Bread First Search – przeszukiwania w szerz) ma złożoność )– jeśli nie jest możlwe stworzenie SP, oznacza to, że transakcja nie może zostać wprowadzona do systemu – potrzebne są dodatkowe krawędzie – przyjaźnie pomiędzy użytkownikami
* dla każdego wierzchołka mającego 1 krawędź stworzenie transakcji zmniejszającej bilans tego użytkownika do 0 i usunięcie krawędzi i wierzchołka z MSP (złożoność

Cały algorytm ma więc złożoność obliczeniową równą

### 3.3. Optymalizacja transakcji grupy użytkowników

W celu optymalizacja transakcji w grupie użytkowników korzystam z identycznego algorytmu co przy dodawaniu nowej transakcji, lecz jako bilans wierzchołka obliczam jego bilans względem grupy wierzchołków w obrębie której odbywa się transakcja.  
Złożoność algorytmu wynosi więc

### 3.4 Globalna optymalizacja transakcji

W celu optymalizacji globalnej można zastosować identyczny algorytm co przy lokalnej optymalizacji transakcji – wtedy algorytm ma złożoność . Algorytm ma więc liniową złożoność Przy bardzo dużej liczbie wierzchołków. Nawet dla liczby wierzchołków rzędu , kiedy każdy z nich ma 100 krawędzi,a łączna liczba krawędzi to optymalizacja globalna nie będzie wielkim problemem dla aplikacji. Znacznie większy koszt obliczeniowy będzie w takim przypadku miało samo zczytanie mapy połączeń użytkowników z bazy danych. Pojawiające się pomysły optymalizacji algorytmu przez podział użytkowników na podgrupy i wykonywanie optymalizacji na tych grupach, przy łączeniu grup tylko za pomocą jednego użytkownika nie mają w tym przypadku większego sensu, ponieważ krawędzie w praktyce i tak będą występować głównie właśnie w takich podgrupach, a sam podział użytkowników zająłby wiele czasu procesora.

## Bezpieczeństwo użytkowników

### OAuth – autoryzacja użytkownika w aplikacji

W celu ułatwienia użytkownikom korzystania z aplikacji zdecydowałem się na wykorzystanie otwartego standardu autoryzacji Oauth korzystając z Facebooka jako z serwisu gwarantującego bezpieczeństwo autoryzacji. W celu zalogowania do serwisu wystarczy być zalogowanym na swoje konto na Facebook’u, a następnie wejść na stronę aplikacji („app.ezyd.pl” lub „facebook.com/ezyd\_app”).

### Szyfrowanie danych

Aplikacja korzysta z szyfrownia danych za pomocą SSL przy wykorzystaniu 128-bitowego klucza. Niestety nie posiadam certyfikatu SSL, przez co gdy chce się korzystać z serwisu poprzez HTTPS wiele przeglądarek internetowych wyświetla komunikaty o zagrożeniu. Jeśli liczba użytkowników serwisu będzie się zwiększać zamierzam wykupić certyfikat SSL.

### Bezpieczeństwo bazy danych

Baza danych MySQL, z której korzysta serwis jest postawiona na moim domowym komputerze tak samo jak reszta serwisu. Router zza którego łączę się z internetem jest zabezpieczony firewallem i połączenia z zewnątrz nie mogą połączyć się z bazą danych. Do tego dostęp do bazy danych jest na wszelki wypadek zabezpieczony hasłem.

## Serwer HTTP/HTTPS

### Dynamiczny adres IP serwera

Serwer na którym postawiona ma dynamicznie przydzielany adres IP. Aby połączenie się z serwerem było możliwe z zewnątrz po zmianie IP serwera wykorzystałem serwis „no-ip.org”.

Po wejściu na stronę „app.ezyd.pl” lub „facebook.com/ezyd\_app” zostajemy przekierowani na stronę „ezydapplication.no-ip.org”, która z kolei przekierowuje na adres serwera, który automatycznie co każde 5 minut aktualizuje swój adres IP na serwisie „no-ip.org” (przez aplikację „No-IP DUC”). Taki czas aktualizacji adresu sprawia, że każda zmiana adresu IP serwera powoduje średnio 2.5 minutową przerwę w działaniu aplikacji, jednak nie wydaje się, żeby na obecnym etapie rozwoju serwisu był to jakiś problem.

### Server www

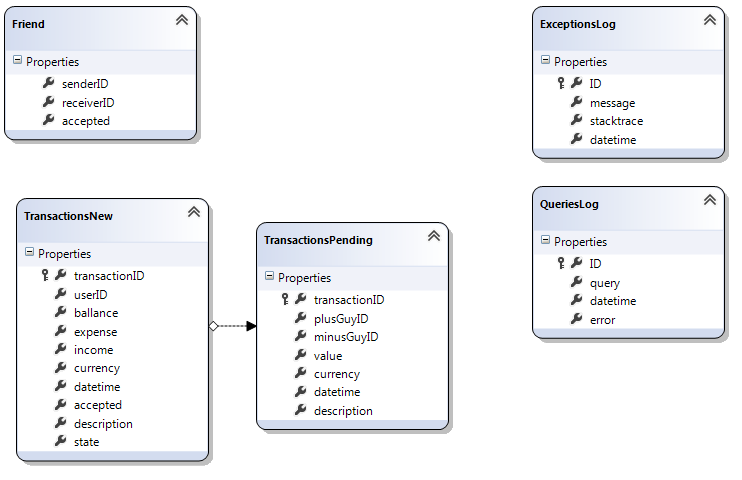
Serwer HTTP/HTTPS postawiony jest na moim domowym komputerze na „Internet Information Services 7” ( „IIS7”). Jest to oprogramowanie dostępne wraz z Windowsem 7 i pozwala stworzyć bardzo niezawodny serwer www.

### Serwer bazy danych

Jako serwera bazy danych użyłem WAMP’a (Windows-Apache-MySQL-PHP). Zdecydowałem się na niego, ponieważ jest to darmowe i bardzo stabilne środowisko. Bałem się korzystania z komercyjnych baz danych takich jak MS SQL, czy baz Oracle, ponieważ są to bardzo drogie środowiska, a w przyszłości zamierzam rozwijać „Pay With Friends” także komercyjnie. Niestety przy korzystaniu z MySQL nie mogłem korzystać z LINQ, które bardzo uprościłoby pisanie aplikacji (słyszałem o możliwości skorzystania z „ADO.NET Entity Framework” w celu mapowania obiektów bazy danych w kodzie, lecz podobno sprawia to wiele problemów i nie zdecydowałem się na użycie tego frameworka).

## Baza danych

### Tabele



### Opis tabel

Opis poszczególnych tabel w bazie danych:

* Friend – informacje na temat krawędzi grafu
* TransactionsNew – informacje na temat niezaakceptowanych transakcji
* TransactionsPending – informację na temat oczekujących transakcji
* [opcjonalnie] ExceptionsLog – debugowa tabela przechowywująca nieosbłużone wyjątki, które wyrzucił program
* [opcjonalnie] QueriesLog – debugowa tabela przechowująca wszystkie zapytania wysłane do bazy danych przez serwis (poza tymi które dotyczą debugowych tabel)

### Użycie bazy danych w aplikacji

.NET nie daje wsparcia dla LINQ dla baz MySQL, więc zastosowałem bibliotekę MySql.Data.MySqlClient. Niestety łączenie z bazą danych, jak i wszystkie zapytania musiały zostać przeze mnie napisane odręcznie (oczywiście zostały one owrappowane funkcjami).