## Imprezy Wokół Nas – Projekt aplikacji w technologii ASP.NET MVC

**Wprowadzenie**

Przedmiotem projektu jest aplikacja, której zadaniem jest umożliwienie użytkownikom znajdującym się w pobliżu (lub zgodnie z innym kryterium) wymiany powiadomień pomiędzy sobą. Ponadto projekt ma umożliwiać przeszukiwanie bazy użytkowników w celu identyfikacji potencjalnych znajomych i zarządzanie tymi znajomymi. Dodatkowymi funkcjonalnościami jest system uwierzytelniania i autoryzacji oraz możliwość zarządzania własnym profilem.

Aplikacja została zaprojektowana w taki sposób aby większość komunikacji pomiędzy serwerem, a klientem odbywała się w sposób asynchroniczny dzięki wykorzystaniu zapytań AJAX. Ponadto całość została zbudowana w taki sposób, aby spełniać standardy RWD. Responsywność uzyskana została między innymi dzięki technologii Bootstrap oraz rozszerzeniom napisanym w języku JavaScript. Dane wymagające stosunkowo dużej retencji przechowywane są w bazie danych MySQL, która w naszym wypadku obsługiwana jest głównie przez technologię Entity Framework. Technicznym założeniem naszej aplikacji jest aktualizacja interfejsu w czasie rzeczywistym bez konieczności przeładowywania całej strony w żądaniu *postback*.

Pierwszą z zaimplementowanych funkcjonalności jest asynchronicznie działający system autoryzacji użytkowników. System ten umożliwia rejestrację, uwierzytelnienie i autoryzację, przypomnienie hasła oraz powtórne wysyłanie linków aktywacyjnych. Dodatkowymi cechami systemu jest możliwość edycji profilu włącznie z awatarem przy użyciu mechanizmu drag and drop, geolokalizacja użytkowników na podstawie adresu IP, dwustopniowa walidacja oraz blokowanie użytkowników przy niepoprawnym wprowadzaniu danych.

Drugą funkcjonalnością jest silnik wyszukiwania znajomych, który pozwala przeszukać użytkowników zgodnie z zadanymi kryteriami, wyświetlić wyniki w czasie rzeczywistym oraz umożliwia dodawanie znajomych bezpośrednio z wyświetlonej listy bez jej przeładowywania.

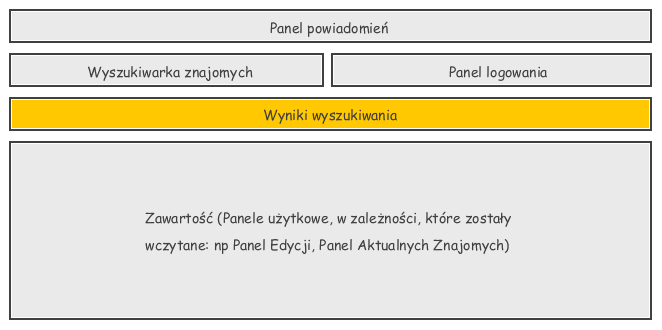
Trzecia funkcjonalność pozwala zarządzać użytkownikami będącymi w relacji z zalogowanym użytkownikiem. W naszej aplikacji znajomy jest relacją wzajemną, która skłąda się z dodania użytkownika do listy znajomych oraz bycia dodanym jako znajomy przez tego użytkownika (bądź akceptacji zaproszenia). System umożliwia wysyłanie i anulowanie zaproszeń, ich odrzucanie bądź akceptację i usuwanie ze znajomych, a także prezentację obecnego statusu.

Ostatnia funkcjonalność pozwala wysyłać powiadomienia, które po otrzymaniu są zaprezentowane w górnej części interfejsu całej aplikacji. Powiadomienia na obecny stan prac nad projektem wysyłane mogą być zgodnie z trzema kryteriami: do wszystkich, do wybranych znajomych z listy, bądź w zależności od odległości pomiędzy zalogowanym użytkownikiem, a jego znajomymi. Panel otrzymanych powiadomień jest rozwijanym zestawem nagłówków wszystkich otrzymanych powiadomień. Nagłówki z kolei pozwalają rozwinąć treść poszczególnych wiadomości. Dodatkowo każda z nich może być asynchronicznie usunięta z listy.

**Odnośniki do projektu:**

* Github: *https://github.com/rvnlord/EventsAroundUs*
* Działająca aplikacja: http://eventsaroundus.azurewebsites.net/
* Konto testowe: **użytkownik**: *test*, **hasło**: *test123* (można też utworzyć własne konto testowe)

**Schemat interfejsu użytkownika:**

****

**Zasady nawigacji:**

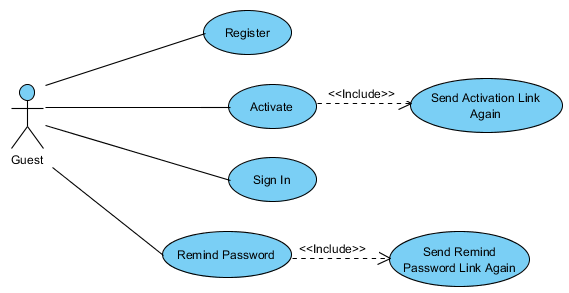
Poniżej przedstawiono przejścia pomiędzy kolejnymi interfejsami graficznymi, jako punkt wyjścia wzięta została strona główna:

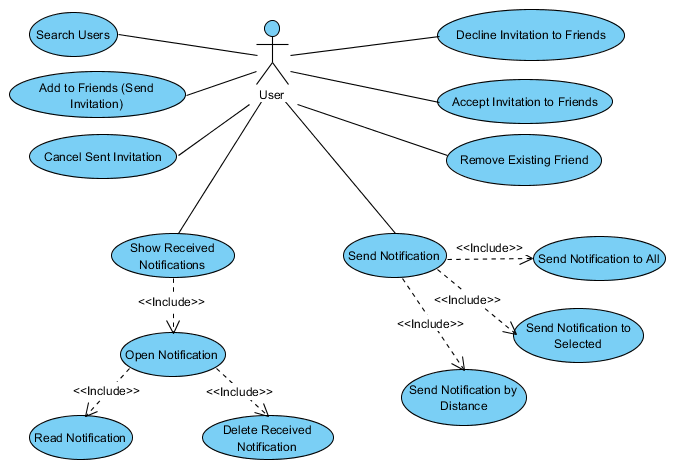
* <Zarejestruj Się> lub <Przypomnij> (kliknięcie) 🡪 User/Register (Panel Rejestracji, Panel Aktywacji, Panel Przypomnienia Hasła)
* <Szukaj> (kliknięcie lub wpisanie frazy) 🡪 Panel Wyników
* <Zaloguj> (kliknięcie) 🡪 Panel Zalogowanego Użytkownika
* <Nazwa Użytkownika> (kliknięcie) 🡪 User/Edit (Panel Edycji Profilu, Panel Wysyłania Powiadomień)
* <Powiadomienia> (kliknięcie) 🡪 Panel Powiadomień

**Zasady interakcji z danymi:**

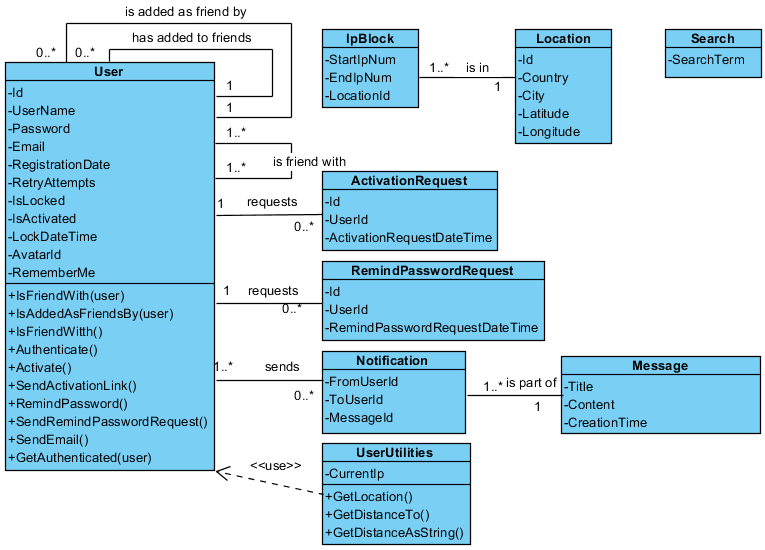
Większość danych aplikacji przechowywana jest w bazie danych MySQL, wyjątek stanowi informacja o zalogowaniu. Informacja ta znajduje się w zależności od wyboru użytkownika w pliku cookie lub w sesji. Interakcja z danymi tymczasowymi odbywa się przy użyciu metod języka c# dedykowanych do tego celu, natomiast interakcja z bazą danych odbywa się przy użyciu systemu relacyjnego mapowania obiektów (ORM), którym jest Struktura Encji (Entity Framework).

**Diagram przypadków użycia:**





**Diagram klas:**



**Model danych:**

Poniżej zamieszczony SQL został kod definiujący dane naszej bazy danych, kod ten generuje tabele analogiczne do klas zaprezentowanych na diagramie klas:

SET GLOBAL general\_log = 'ON';

DROP DATABASE IF EXISTS acsm\_d1e5fb607a11ffe;

CREATE DATABASE acsm\_d1e5fb607a11ffe;

ALTER DATABASE acsm\_d1e5fb607a11ffe DEFAULT COLLATE utf8\_general\_ci;

CREATE TABLE tblImages

(

Id CHAR(36) PRIMARY KEY,

ImageData LONGBLOB,

ImagePath VARCHAR(256)

);

CREATE TABLE tblUsers

(

Id CHAR(36),

UserName VARCHAR(100),

Password VARCHAR(200),

Email VARCHAR(200),

RegistrationDate DATETIME,

RetryAttempts INT,

IsLocked INT,

LockedDateTime DATETIME,

IsActivated INT,

AccessLevel INT,

AvatarId CHAR(36),

CurrentIp INT UNSIGNED,

CONSTRAINT pk\_users\_id PRIMARY KEY (Id),

CONSTRAINT ck\_users\_id CHECK (Id REGEXP '[a-fA-F0-9]{8}-[a-fA-F0-9]{4}-[a-fA-F0-9]{4}-[a-fA-F0-9]{4}-[a-fA-F0-9]{12}'),

CONSTRAINT fk\_users\_avatarid FOREIGN KEY (AvatarId) REFERENCES tblImages(Id)

);

CREATE TABLE tblActivationRequests

(

Id CHAR(36) PRIMARY KEY,

UserId CHAR(36),

ActivationRequestDateTime DATETIME,

FOREIGN KEY (UserId) REFERENCES tblUsers(Id)

);

CREATE TABLE tblRemindPasswordRequests

(

Id CHAR(36) PRIMARY KEY,

UserId CHAR(36),

RemindPasswordRequestDateTime DATETIME,

FOREIGN KEY (UserId) REFERENCES tblUsers(Id)

);

CREATE TABLE tblKeys

(

Id VARCHAR(30) PRIMARY KEY,

Value VARCHAR(1500)

);

CREATE TABLE tblFriendRelationships

(

UserId CHAR(36),

FriendId CHAR(36),

PRIMARY KEY (UserId, FriendId),

FOREIGN KEY (UserId) REFERENCES tblUsers(Id),

FOREIGN KEY (FriendId) REFERENCES tblUsers(Id)

);

DROP TABLE IF EXISTS tblMessages;

CREATE TABLE tblMessages

(

Id CHAR(36),

Title TEXT,

Content LONGTEXT,

CreationTime DATETIME,

PRIMARY KEY (Id)

);

DROP TABLE IF EXISTS tblNotifications;

CREATE TABLE tblNotifications

(

FromUserId CHAR(36),

ToUserId CHAR(36),

MessageId CHAR(36),

PRIMARY KEY (FromUserId, ToUserId, MessageId),

FOREIGN KEY (FromUserId) REFERENCES tblUsers(Id),

FOREIGN KEY (ToUserId) REFERENCES tblUsers(Id),

FOREIGN KEY (MessageId) REFERENCES tblMessages(Id)

);

CREATE TABLE tbllocations

(

Id INT PRIMARY KEY NOT NULL,

country CHAR(2),

city LONGTEXT,

latitude DOUBLE,

longitude DOUBLE

);

CREATE TABLE tblips

(

startIpNum INT UNSIGNED DEFAULT 0 NOT NULL,

endIpNum INT UNSIGNED DEFAULT 0 NOT NULL,

LocationId INT,

CONSTRAINT pk\_ips PRIMARY KEY (startIpNum, endIpNum),

CONSTRAINT fk\_locationid FOREIGN KEY (LocationId) REFERENCES tbllocations(Id)

);

ALTER DATABASE acsm\_d1e5fb607a11ffe DEFAULT COLLATE utf8\_general\_ci;

SET foreign\_key\_checks = 0;

ALTER TABLE tblUsers CONVERT TO CHARACTER SET utf8 COLLATE utf8\_general\_ci;

ALTER TABLE tblActivationRequests CONVERT TO CHARACTER SET utf8 COLLATE utf8\_general\_ci;

ALTER TABLE tblremindpasswordrequests CONVERT TO CHARACTER SET utf8 COLLATE utf8\_general\_ci;

ALTER TABLE tblKeys CONVERT TO CHARACTER SET utf8 COLLATE utf8\_general\_ci;

ALTER TABLE tblImages CONVERT TO CHARACTER SET utf8 COLLATE utf8\_general\_ci;

ALTER TABLE tblFriendRelationships CONVERT TO CHARACTER SET utf8 COLLATE utf8\_general\_ci;

ALTER TABLE tblIps CONVERT TO CHARACTER SET utf8 COLLATE utf8\_general\_ci;

ALTER TABLE tblLocations CONVERT TO CHARACTER SET utf8 COLLATE utf8\_general\_ci;

ALTER TABLE tblNotifications CONVERT TO CHARACTER SET utf8 COLLATE utf8\_general\_ci;

ALTER TABLE tblMessages CONVERT TO CHARACTER SET utf8 COLLATE utf8\_general\_ci;

SET foreign\_key\_checks = 1;