

POLITECHNIKA ŚLĄSKA

WYDZIAŁ AUTOMATYKI, ELEKTRONIKI I INFORMATYKI

Projekt inżynierski

Aplikacja społecznościowa oparta na geolokalizacji

Autor: Tytus Dragon

Kierujący pracą: dr inż. Alina Momot

Gliwice, Styczeń 2016 r.

Spis treści

Wstęp.........................................................

1. Analiza tematu......................................
   1. Motywacje i opis tematu.................
   2. Założenia projektowe......................
2. Wybór narzędzi programistycznych.....
   1. System operacyjny..........................
   2. Języki programowania i technologie....................

Wstęp

Kilkanaście lat temu na rynku pojawiły się pierwsze smartfony. W ostatnich latach urządzenia te rozwijały się w błyskawicznym tempie. Producenci tworzą coraz lepsze i doskonalsze urządzenia, które służą do komunikacji, ułatwiają podróżowanie, zachowywanie bezpieczeństwa, dostarczają rozrywki oraz mają wiele różnych zastosowań. Dzięki rozwojowi technologii w ostatnich latach coraz bardziej stają się popularne aplikacje mobilne. Każdy może zarządzać swoim kontem bankowym przez aplikację, używać systemu nawigacji GPS podczas jazdy samochodem trafiajac do celu bez problemu, rozmawiać z przyjaciółmi i rodziną będąc setki kilometrów od nich, a także opisywać swoje życiowe wydarzenia, publikować zdjęcia oraz filmy w intenrecie.

Komunikacja międzyludzka przybrała wiele nowych form dzięki mobilnym technologiom. Wciąż powstają nowe sposoby i pomysły na realizację międzyludzkiej komunikacji takie jak blogi, sieci biznesowe, projekty ułatwiające pracę zespołową, fora dyskusyjne, portale umożliwiające udostępnianie zdjęć, recenzowanie produktów, gry społecznościowe i wiele innych . Niektóre z nich nie mają tylko celów rozrywkowych - możemy się spotkać z wieloma roziwązaniami biznesowymi usprawniającymi pracę, umożliwiającymi przesyłanie i archiwizację plików, dokumentów. Co więcej, istnieją także aplikacje i portale, które stanowią w średnich i dużych firmach systemy ewidencji czasu pracy. Ogólnie pojęte korzystanie z intenretowych i mobilnych technologii w celu komunikacji i interaktywnego dialogu niedawno przyjęło nazwę mediów społecznościowych (ang. *Social media*).

Celem tej pracy jest implementacja społecznościowej aplikacji mobilnej, służącej do komunikacji poprzez udostępnianie swojego położenia dzięki modułowi GPS, który obecnie posiada niemal każde urządzenie mobilne dostępne na rynku. Aplikacja będzie umożliwiała posiadanie własnego konta i profilu, możliwość udostępniania o sobie podstawowych informacji, takich jak imię i zdjęcie z wizerunkiem. Dane te mają posłużyć do indentyfikowania przyjaciół oraz wymiany danych między użytkownikami. Ponadto system ma pozwalać na zapraszanie użytkowników do listy znajomych, której celem jest łatwy i szybki dostęp do poszerzonych zasobów zwierających informacje o użytkownikach.

Niniejsza praca składa się z pięciu rozdziałów. Pierwszy przedstawia główne założenia projektu. Drugi zawiera opis wykorzystanych narzędzi, które były potrzebne podczas implementacji systemu oraz jego projektowania. Następnie opisane są specyfikacje, szczegóły implementacji, opisy użytkowania, instrukcje instalacji aplikacji, itd. Piąty rozdział obejmuje testowanie systemu i problemy jakie napotkano podczas realizacji projektu. Ostanie strony pracy to bibliografia i dodatki.

1. Analiza Tematu

W rozdziale tym znajduje się szerszy opis tematu pracy. Opisano tu także motywację tworzenia oprogramowania, które jest przedmiotem tej pracy. Drugą część tego rozdziału stanowią wymagania i założenia, które zdefinowano, by spełniał je implementowany system.

* 1. Motywacje i opis tamatu

Przedmiotem projektu jest mobilna aplikacja społecznościowa działająca na smartfonach z systemem operacyjnym Windows Phone 8.1 oraz Windows 10 Mobile. Aplikacja ma służyć do udostępniania danych o użytkownikach takich jak: login, zdjęcie i opis, krótką informacje o użytkowniku o dowolnej treści. Udostępniać ma również datę ostatniej wizyty, dane o lokalizacji użytkownika oraz, co bardzo istotne, mapę w celu przeglądania lokalizacji znajomych.

Inspiracją stworzenia systemu była chęć implemetacji aplikacji, która pozowoliłaby rodzicom na większą kontrolę swoich pociech. Dziecko, którego urządzenie mobilne miałoby zainstalowaną aplikację, która regularnie udostępnia dane GPS, było by bezpieczniejsze, a w przypadku zgubienia się szybko odnalezione. Drugą, choć wynikającą z pierwszej, motywacją była kontrola lokalizacji dzieci niepełnosprawnych, na przykład chorych umysłowo. Aplikacja mogłaby też znaleźć zastosowanie w krytycznych sytuacjach dla bliskich i być pomocniczo wkorzystywana wprocedurach stosowanych przez np. organy ścigania.

Ostatecznie zdecydowano się na pomysł nieco bardziej rozbudowanej aplikacji społecznościowej, skierowanej do dowolnej grupy użytkowników, nieograniczonej wiekowo. Zrezygnowano z pomysłu wyłącznej kontroli dzieci ze względów technicznych i ograniczeń darmowych rozwiązań techonlogii mobilnych.

* 1. Założenia projektowe

Podrozdział ten zawiera wymagania, które ma spełniać implmentowany

system. Wymagania te są podstawą do tworzenia kolejnych funkcji bazy danych, serwisu internetowego oraz aplikacji. Definują również ograniczenia, które autor pracy zakładał przy projektowaniu aplikacji.

Aplikacja będzie przeznaczona dla grup ludzi niezależnych wiekowo. Należy jednak wziąć pod uwagę, że żadne urządzenie, jakim jest smartfon, nie powinno stanowić zabawki dla dzieci poniżej trzeciego roku życia.

Ogólne założenia:

* działanie aplikacji na wielu urządzeniach dowolnej architektury z wszystkich dostępnych dla platformy,
* docelowe systemy operacyjne: Windows Phone 8.1, Windows 10 Mobile,
* prosty i przejrzysty interfejs użytkownika, zgodny ze standardami współczesnych technologii Microsoftu [1],
* dostęp do internetu z urządzenia z zainstalowaną aplikacją,
* dostępność aplikacji z Windows Store [2],

Konta i użytkownicy:

* jeden typ użytkownika w aplikacji – *user*, brak użytkowników z dodatkowym dostępem do danych,
* logowanie do aplikacji za pomocą loginu i hasła,
* jeden użytkownik posiadający tylko jedno konto,
* konto tworzone w aplikacji przy pierwszym logowaniu,
* brak możliwości zmiany loginu, możliwość zmiany hasła użytkownika

Dodatkowe możliwości użytkownika w aplikacji:

* dodawanie zdjęcia profilowego użytkownika,
* udostępnianie opisu, wiadomości dla innych użytkowników,
* udostępnianie danych GPS,
* logowanie ściśle związane z pozostawieniem w bazie informacji o lokalizacji, dacie i godzinie,
* możliwośc wyszukiwania osób w bazie danych na podstawie loginu,
* lista znajomych dla każdego użytkownika, system zaproszeń do grona znajomych,
* możliwość przeglądania mapy w celu odnalezienia lokalizacji osób znajdujących sie na liście znajomych.

1. Wybór narzędzi programistycznych

Ten rozdział zawiera informacje dotyczące systemu operacyjnego dla ktorego implementowana jest aplikacja. Opisano także technologie, które wykorzystano i opisano krótko stosowane języki programowania.

* 1. System operacyjny

Zdecydowano, że aplikacja będzie stworzona w technologii Windows Phone. Platforma ta jest trzecią najbardziej popularną obecnie na rynku europejskim. W ostatnim czasie jednak traci na popularności i jest wypierana z rynku przez inne popularne systemy operacyjne. Uważa się jednak, że system dobrze zaprojektowany, a pomimo dużych ograniczeń, sprawdza się idealnie jako system dla celów biznesowych. Warto także wspomnieć tutaj o Windows Store, który jest witryną internetową lub aplikacją – sklepem, z którego pobierać można aplikację, wliczając w to niniejszy projekt. Docelowo projekt jest tworzony dla wersji systemu Windows Phone 8.1, jednak technologia jest kompatybilna wstecz, więc pliki projektu bądą również działać w systemie Windows 10 Mobile.

* 1. Języki programowania i technologie

Visual Studio 2015

Visual Studio 2015 Community jest zintegrowanym środowiskiem programistycznym umożliwiającym szeroko pojęte tworzenie oprogramowania. Wykorzystano go w pracy do implementacji zarówno serwisu internetowego w technologii ASP.NET MVC, ale również do generowania bazy danych. Jest podstawowym środowiskiem do tworzenia aplikacji dla systemu Windwos Phone. Posiada wiele narzędzi ułatwiających pracę programisty.

.NET Framework

Jest platformą programistyczną, którą opracowała firma Microsoft. Jej zadaniem jest zarządzanie elemantami systemu, pamięcią, oraz kodem. Dostarcza biblioteki klas, które zapewniają podstawowe funkcjonalności oprogramowaniu. Umożliwia implementację oprogramowania w wielu różnych językach. Są to między innymi C++, J#, F# oraz C#, który wykorzystano dla implemetacji niniejszego projektu.

C#

Jest wysokopoziomowym językiem obiektowym. Wykorzystywany od parunastu lat mieści się w czołówce najbardziej popularnych współcześnie języków programowania. Program, który napisano w tym języku, jest kompilowany do CIL – języka pośredniego, który uruchamia środowisko .NET. Do tworzenia projektu zdecydowano się na ten właśnie język ze względu na najdłuższe doświadczenie autora pracy w technologiach ziwązanych z tym językiem.

XAML

Wspomaga pracę nad oprogramowaniem w technologiach takich jak Windows Presentation Foundation, Silverlight oraz Windows Phone. Język opisu interfejsu użytkownika. Jest jedną z częsci .NET i tak jak C# jest zamieniany na język wspólny i interpretowany w locie. Rozdzielenie zdefiniowanego interfejsu od logiki pozwala na niezależność tworzenia pojedynczych elementów projektu.

GIT

Git jest narzędziem służącym do kontroli wersji. Jest narzędziem niezwykle przydatnym podczas rozwijania oprogramowania. Pozwala na szybki wgląd do kodu. łatwo w nim porównać dwa różne elementy, skasować zmiany i wrócić do wersji np. z powodu popełniena błędów. Pozwala także na tworzenie gałęzi projektu, co stanowi ogromne wsparcie w przypadku dużych projektów, przy których pracuje wiele osób.

Entity Framework

Narzędzie również powstało jako część platformy .NET. Jest narzedziem typu ORM (ang. Object Relational Mapping). Pozwala na wiele ułatwień przy implementacji baz danych. W projekcie służy jedynie do zdefiniowania i wygenerowania bazy danych, chociaż jest narzedziem wielu zastosowań. Przekształca stworzony schemat w plik bazy danych przy obecności platformy Azure.

Azure

Jest produktem Microsoftu. Stanowi platformę chmurową umożliwijącą uruchamianie aplikacji, stron internetowych i wiele innych. Serwis, który implementowano w niniejszym projekcie korzysta właśnie z usług Azure. Pozwala na zasadniczo prosty proces wdrażania aplikacji do interentu. W praktyce proces ten jest dość skomplikowany i łatwo wyłączyć zasoby nieświadomie lub zburzyć ich wspólpracę, którą potem bardzo trudno przywrócić w środowisku Visual Studio.

1. Specyfikacja wewnętrzna

Niniejszy rozdział zawiera informacje związane ze specyfikacją wewnętrzną implementowanego projektu. Omówiono w nim bazę danych, wraz z jej elementami, wykorzystane biblioteki, strukturę projektu oraz zawarto niektóre ważne, zaimplementowane algorytmy.

* 1. Struktura projektu

Implementowany projekt stanową trzy warstwy. Pierwsza warstwa jest bazą danych SQL osadzoną na serwerach platformy Azure. Jest osobną warstwą pod względem logicznym, jednak jej struktura została wygenerowna przy pomocy narzędzia Entity Framework równocześnie z implementacją serwisu. Serwis stanowi następną warstę logiczną. Jest napisany w technologii ASP. NET MVC. Na potrzeby aplikacji jest jedynie mostem między aplikacją a bazą danych wydobywającym, filtrującym oraz przetwarzającym dane. Wykorzystano w nim także niektóre zapytania pochodzące z biblioteki Linq, która jest oparata na języku SQL. Technologia ta pozwala na stworzenie uniwersalnego serwisu typu RESTfull, dlatego chcąc na przykład rozszerzyć projekt, nie powinna stanowić problemu implementacja niniejszej aplikacji na inną platformę niż Windows Phone.

By stworzyć API, czyli interfejs komunikacyjny dla bazy danych oraz aplikacji Windows Phone skorzystano z technologii ASP.NET MVC. Nie wykorzystano całego potencjału owej technologii, gdzyż tworzenie API to tylko jedna z wielu jej możliwość. Projekt stanowią klasy w dużej części wygenerowane przy tworzeniu projektu. Najważniejsze w implementacji tego serwisu było utworzenie połączenia z zewnętrzynm zasobem, jakim jest baza danych oraz implementacja Kontrolerów.

MVC stanowi wzorzec architektoniczny dla aplikacji w techonlogii ASP.NET. W projektowaniu serwisu wykorzystano właściwie tylko dwie części wzorca. Wygenerowany widok służy wyłącznie do ręcznego przeglądania danych i jest zbędny w przypadku tworzenia tego typu API. W projekcie znajdziemy folder o nazwie Controllers. To w nim znajdują się najważniejsze klas. Część z nich została również wygenerowana przy pomocy narzędzi Entitty Framework.

Najbardziej polcanym wzorcem architektonicznym dla Windows Phone jest MVVM. Stotuje się go w celu rozwiązania najczęstszych problemów związanych z implementacją programu. Model-View-ViewModel ma wiele zalet i jest prosty w działaniu. Z powodu większej ilośći plików i klas łatwo pomyśleć, że stosowanie jakichkolwiek wzorców projektowych to utrudnianie pracy i zbędna strata czasu. Zarówno Model-View-Controller, jak i Model-View-ViewModel pozwala na oszczędność czasu w pózniejszych fazach projektu. Jest niezwykle przydatny pod względem modyfikowalności ze wzgledu na podział na odrębne moduły odpowiadajace za widok oraz logikę. Pozwala na łatwe przenoszenie oprogramowania na inne technologie oraz umożliwia przeprowadzanie zautomatyzowanych testów.

Zastosowanie MVVM wiąże się z podziałem aplikacji na trzy moduły:

* Model – zawiera logikę biznesową aplkacji, językiem jest C#,
* View – stanowi interfejs użytkownika i implemetuje się go w XAML oraz C#
* ViewModel – jest łącznikiem dla modelu oraz widoku, stosuje mechanizm Bindowania [3] oraz Komendy, językiem implementacji jest C#



Rysunek 1. Schemat wzorca architektonicznego Model-View-ViewModel

Stosowanie takiego modelu prowadzi także do możliwości podziału projektu na trzy odrębne moduły, podprojekty które mają swoje funkcje i stanowią odrębne części oprogramowania. Solucja Aplikacji mobilnej składa się z następujących podprojektów:

* Tillsammens.WindowsPhone.App
* Tillsammens.WindowsPhone.Domain
* Tillsammens.WindowsPhone.WebServices

Pierwszy podprojekt stanowi zbiór klas charakterystycznych dla Windows Phone, zawiera całą logikę oprogramowania. Znajdują się w nim widoki oraz klasy dziedziczące po klasie *ViewModelBase*. Podprojekt ten zawiera także pliki typu *resource dictionares*, pisane w XAML, które odpowiadają za wygląd aplikacji i są odpowiednikiem plików w formacie *.css* dla aplikacji pisanych w HTML. Znajduje się także folder z niezbędnymi klasami konwerterów, które wspomagają XAML, np. służą do interpretacji właściwośći boolowskiej na właściwość *Visibility*, która odpowiada, czy dany element interfejsu użytkownika jest widoczny, bądź nie. Innym przykładem konwertera jest zmiana obiektu *DateTime* na odpowiedni, skrócony format bez koniecznosci robienia tego w *ViewModel*. Co ważne, w folderze *ViewModels* znajduje się klasa, króra jest klasą podstawową dla klas typu *ViewModel.* Jako jdyna dziedziczy po klasie podstawowej dla tego wzorca aritektonicznego. Zawiera w sobie te właściwości i metody, które uznano za wspólne dla wszystkich klas.

Domain oraz WebServices to biblioteki klas, które zawierają fukncje i metody uniwersalne dla całej technologii .NET. Zawierają klasy pomocnicze do implementacji projektu dla Windows Phone, ale same stanowią odrębną część. Pozwala to na łatwe przeniesienie aplikacji na inną technologie, głównie przez jedynie zmianę interfejsu użytkownika, a w przypadku tej pracy, szczególnie na przejrzystość implementowanego kodu i możliwośc zachownia porządku w całej solucji.

W WebServices znajdziemy klasy, które odpowiadają za wysyłanie zapytań do serwisu typu REST, który stanowi odrębną warstwę projektu. Znajduje się tutaj także moduł MVVM, jakim jest model.

* 1. Baza danych

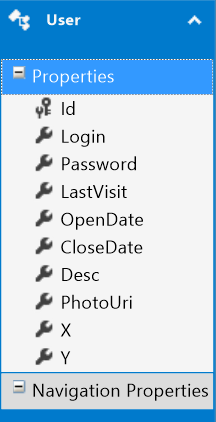
Podrozdział ten zawiera wszelkie informacje dotyczące bazy danych wykorzystanej w niniejszym projekcie. Znajduje się tu również opis tabel.

Platforma Azure umożliwia tworzenie własnych baz danych i osadzenie ich uruchomionych na serwerach plaformy. Pustą bazę danych można utowrzyć korzystając z portalu Azure [4]. Należy jednak najpierw zdefiniować serwer, na którym baza ta ma być osadzona. W celu utworzenia bazy danych dla niniejszego projektu zdefiniowano serwer o nazwie *project18server.database.windows.net*. Pierwsza część nazwy jest nazwą własną. W trakcie implementacji owego projektu tworzono te zasoby wielokrotnie w celach testowych, dlatego nazwa jest tak ogólna. Niemniej jednak baza danych nosi nazwę: *project18dataBase*, co stanowi nazwę podobną do serwera. W portalu można na wiele sposobów monitorować swoje zasoby. Przydatne okazuje się monitorowanie blędów lub czy zasób jest w trybie on-line oraz ile miejsca zajmują dane w bazie danych.

Baza danych składa się z dwóch tabel. Początkowo miała się składać z czterech elementów, jednak zoptymalizowano ją i uproszczono, gdyż proste rozwiązanie okazało się być łatwiejszym i wystarczającym na potrzeby ninejszego proejktu.

Aby utworzyć baze danych, użyto narzędzia EntityFramework. Konieczne jest zaprojektowanie bazy danych w podprogramie dostępnym w środowisku programowania, a następnie wygenerowanie jej przy pomocy owego narzędzia. Podczas tworzenia, ważnym jest, by w projekcie zdefiniować server i bazę danych , w których chcemy stworzyć zaprojektowany model. W niniejszym projekcie posłużono się pustą bazą danych, o której wcześniej mowa. W ten sposób w baza danych posiada ten sam schemat modeli, co implementowany projekt ASP.NET, co ułatwia pracę z wydobywaniem z bazy danych tablic i ich elementów.

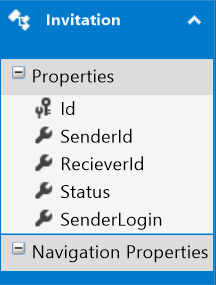
Tabela UserSet jest pierwszym elementem bazy danych. Reprezentuje jednego użytkownika aplikacji. Stanowi ją 10 pól róznych typów danych:



Rysunek 2 Baza danych – schemat elementu tablicy UserSet

* **Id –** Pole typu *Int32*, unikalny identyfikator każdego obiektu typu *User*, klucz główny, wartości są autonumerowane.
* **Login –** Łańcuch znaków, ograniczony do 25 elementów. Również unikaly, jednak unikalnością zajmuje się serwis, który przy tworzeniu użytkownika zgłasza błąd w przypadku próby utworzenia konta o juz istniejącym loginie.
* **Password –** Zaszyfrowane hasło potrzebne przy logowaniu do aplikacji, nieograniczony łańcuch znaków.
* **LastVisit –** Pole typu *DateTime*, przechowuje datę i godzine ostatniej wizyty użytkownika w aplikacji. Przy uruchomieniu aplikacji jest aktualizowane razem z polami dotyczącymi lokalizacji użytkownika.
* **OpenDate –** Pole typu *DateTime*. W tym polu znajduje się Data i godzina w której założono konto danego użytkownika. Pole to nie jest modyfikowane przez żadne zapytanie w API.
* **CloseDate –** Pole typu *DateTime*. Przechowuje Date i godzinę, w której użytkownik usunął konto. Konta nigdy nie są usuwane trwale – jedynie aktualizowane jest to pole mówiące o tym, czy użytkownik jest aktywny. Pole służy do filtrowania kont przy logowaniu, pobieraniu znajomych oraz przy wyszukiwaniu użytkowników w serwisie.
* **Desc –** Łańcuch znakowy – pełni funkcję opisu danego użytkownika. Zakładający konto może napisać w aplikacji informacje o sobie, a ten opis jest przechowywany w tym polu. Początkowo każdy użytkownik ma domyślny opis, jednak w aplikacji jednym kliknięciem można go zmienić.
* **Photo Uri –** Łańcuch znaków. Przechowywuje adres do obrazka, zdjęcia, które pełni funkcje zdjęcia profilowego danego użytkownika. Pobierane jest przy logowaniu do aplikacji, przy szukaniu znajomych oraz przy wyświetlaniu listy znajomych
* **X –** Pole typu *double*. Stanowi pierwszą wartość definiującą lokalizację użytkownika – szerokość geograficzną. Aktualizowana przy każdym zalogowaniu do aplikacji przez użytkownika oraz w trakcie działania podczas odświeżania stanu aplikacji.
* **Y –** Pole typu *double*. Jest drugą częścią lokalizacji użytkownika – stanowi długość geograficzną. Aktualizowana tak często jak powyższe pole *X* tabeli UserSet.

Tabela InvitationSet jest następnym elementem bazy danych. Reprezentuje relację dwóch użytkowników aplikacji. Składa się z czterech pól:



Rysunek 3 Baza danych – schemat elementu tablicy InvitationSet

* + **Id –** Pole typu *Int32*, unikalny identyfikator każdego obiektu typu *Invitation*, klucz główny, wartości są autonumerowane.
  + **SenderId –** Pole typu *Int32*. Zawiera w sobie identyfikator użytkownika, który tworzył zaproszenie do grona znajomych dla innego użytkownika.
  + **RecieverId –** Pole typu *Int32*. Stanowi identyfikator użytkownika, do którego zaproszenie do grona znajomych jest adresowane.
  + **Status –** Łańcuch znaków. Prezentuje status danego zaproszenia do grona znajomych. Na podstawie tego pola serwis filtruje użytkowników w zależności od rodzaju danego zapytania.
  + **SenderLogin –** Łańcuch znaków. Stanowi Login użytkownika, który stworzył dane zaproszenie w aplikacji. Pole to umożliwia wyświetlenie komunikatu zaproszenia do grona znajomych adresatowi, bez dodatkowego pobierana całego obiektu uzytkownika zapraszającego.
  1. Biblioteki

W celu ułatwienia zadania projekt jest wsparty zewnętrzymi bibliotekami. Ten krótki podrozdział jest poświęcony dodatkowym bibliotekom wykorzystanym do implementacji aplikacji dla systemu Windows Phone.

Ważną bilbioteką, którą użyto w niniejszym projekcie jest *RestSharp*. Wszystkie uzyte biblioteki dołączono do solucji przy pomocy narzędzia NuGet, dostępnym w Visual Studio. Dzięki niemu można sprawnie i szybko dodawać biblioteki zewnętrzne do swojego oporgramowania. Biblioteka *RestSharp* jest zbiorem klas które ułatwiają realizację aplikacji korzystających z internetu. Przy użyciu tej biblioteki stworzono w podprojekcie *WebServices* uniwersalną dla wszysttkich zapytań klasę RestClientBase, która odpowiada za komunikację aplikacji z serwisem. Skorzystano głównie z trzech klas tej biblioteki:

* RestClient – klasa definiująca połączenie z danym serwisem typu RESTfull. W jej konstruktorze podaje się główny adres serwisu, do którego aplikacja ma wysyłać zapytania.
* RestRequest – klasa, która odpowiada za pojedyncze zapytanie. Instancje tej klasy są tworzone przy każdym zapytaniu. Pozwala na zdefinowanie zapytania w formie łańcucha znaków oraz wybór rodzaju metody sposród dostępnych takich jak: GET, POST, DELETE, czy PUT.
* RestResponse – klasa, która jest wykorzystywana w chwili zapytania. No niej przekazywana jest cała odpowiedź danego zapytania. W przypadku pomyślnej odpowiedzi dane są serializowane i wysyłane w inne moduły programu.

Drugą ważną, zastosowaną biblioteką jest *GalaSoft.MvvmLight*. Instalacja tej biblioteki zaopatruje projekt w dodatkowe funkcje oraz klasy, które stanowią podstawę wzorca projektowego *Model-View-ViewModel,* o którym mowa wcześniej w pierwszej części tego rozdziału. W projekcie wykorzystano między innymi takie klasy jak *ViewModelLocator*, który ułatwia dostęp do klas, które dziedziczą po klasie *ViewModelBase*. Użyto także *RelayCommands* – należące do tej biblioteki klasa implementująca *ICommand*, z której korzysta się w mechanizmie bindowania komend np. do przycisków w widoku.

* 1. Implementacja – wybrane algorytmy

Podczas implementacji systemu zmierzono się z kilkoma zagadnieniami , które pod względem programistycznym należałoby omówić.

Pobranie znajomych z bazy danych przez serwis. W kotrolerze *FriendsConrtoller* znajduje się metoda typu *GET*, której zadaniem jest zrwrócić wszystkich znajomych uzytkownika o identyfikatorze podanym jako prametr wejściowy.

[ResponseType(typeof(IEnumerable<User>))]

public IHttpActionResult GetFriends(int id)

{

try

{

using (var db = new DataModelContainer())

{

var invitations =

db.InvitationSet.Where(invitation =>

(invitation.RecieverId == id || invitation.SenderId == id)

&& invitation.Status.Contains("Accepted"))

.ToList();

var friends = new List<User>();

foreach (var i in invitations)

{

if (i.SenderId == id) riends.Add(db.UserSet.Find(i.RecieverId));

else friends.Add(db.UserSet.Find(i.SenderId));

}

var sortedFriends = (from f in friends

where f.CloseDate > DateTime.Now

select new User

{

Id = f.Id,

Login = f.Login,

PhotoUri = f.PhotoUri,

LastVisit = f.LastVisit,

Desc = f.Desc,

X = f.X,

Y = f.Y,

}).ToList();

return Ok(sortedFriends);

}

}

catch(Exception ex)

{

return InternalServerError(ex);

}

}

Metoda pobiera z bazy danych obiekty typu *Invitation*. Przy pomocy metody *Where()* pochodzącej z biblioteki *Linq* wydobywa zaproszenia, które dotyczą użytkownika o danym identyfikatorze oraz mają status zaakceptowanych. Tylko na podstawie tych obiektów tworzy się obecna lista znajomych dla danego użytkownika. Następnie odbywa się wydobycie wszystkich potrzebnych obiektów typu *User* poprzez zawarte w liście identyfikatory. Lista filtruje uzytkowników i zwraca potrzebne dane.

1. Specyfikacja zewnętrzna

Ten rozdział zawiera opis specyfikacji zewnętrzej niniejszego projektu. Omówiony w nim został szczgółowo interfejs użytkownika oraz podane zostały niezbędne informacje dotyczące uruchomienia aplikacji i korzystania z niej.

* 1. Wymagania sprzętowe

Podrozdział ten zawiera niezbędne informacje do uruchomienia aplikacji. Informacje dotyczą wymagań sprzętowych oraz dostepnych systemów operacyjnych, na których można uruchomić aplikację.

System Windwos Phone 8.1 oraz Windows 10 Mobile obecnie można znaleźć jedynie na urządzeniach marki Nokia oraz Microsoft z drobnymi wyjątkami marki HTC, czy Alcatel. To spore ograniczenie, ale w przypadku braku dostępnego urządzenia można skorzystać z emulatora – maszyny wirtualnej instalowanej przy dodawaniu Windows Phone SDK do środowiska Visual Studio 2015 Community.

Program uruchamiano wielokrotnie na różnych urzadzeniach z systemem operacyjnym Windows Phone 8.1 oraz Windows 10 Mobile. Najsłabszy sprzętowo smartfon, na którym testowano oprogramowanie uznano za telefon z minimalną konfiguracją sprzętową. Oto minimalne paramtry sprzętu:

* Procesor: Qualcomm MSM8227, 1,00 GHz
* RAM: 512MB
* Wolne miejsce w pamięci: 4MB
* System Operacyjny: Windows Phone 8.1
  1. Instalacja
  2. Interfejs użytkownika

Podrozdział ten stanowi omówienie tej części oprogramowania, z którą uzytkownik ma bezpośredni kontakt. Omówiono w nim funkcjonalności oraz uzytkowanie programu.

* Tworzenie konta.

Jeśli smartfon pomyślnie zainstalował aplikację, możemy ją wybrać z listy aplikacji w celu uruchomienia