POLSKO-JAPOŃSKA WYŻSZA SZKOŁA TECHNIK KOMPUTEROWYCH

Wydział Informatyki

Katedra Inżynierii Oprogramowania

Inżynieria Oprogramowania i Baz Danych

Jakub Milik
Nr albumu 4122

Geolokalizacja w portalach internetowych

Praca magisterska napisana
pod kierunkiem:

dr inż. Mariusz Trzaska

Warszawa, sierpień, 2010

Streszczenie

Niniejsza praca skupia się na temacie tworzenia spersonalizowanych map internetowych. Istnieje wiele rozwiązań umożliwiających zaznaczenie punktu na mapie i wprowadzenie jego opisu, lecz żadne z nich nie pozwala na zdefiniowanie rodzaju danych, jakie będą wprowadzane. Wynikiem pracy jest aplikacja, która próbuje wypełnić tę lukę.

Na początku pracy omówione zostały dostępne rodzaje map internetowych, oraz rozwiązania umożliwiające ich personalizowanie. W kolejnym rozdziale opisane są technologie wykorzystywane przy tworzeniu aplikacji. Następnie została zaproponowana funkcjonalność systemu umożliwiającego tworzenia spersonalizowanych map. Przedostatni rozdział traktuje o tym, jak system został zaimplementowany i jak z niego korzystać. Na końcu omówione zostały wady, zalety i plany rozwojowe.

W trakcie tworzenia aplikacji autor starał się dołożyć wszelkich starań, aby aplikacja cechowała się wysoką łatwością użytkowania (usability) i posiadała przyjemny interfejs graficzny. W wyniku tego powstała interaktywna aplikacja webowa wykorzystująca najnowsze technologie internetowe.

Spis treści

1 Wstęp 5
1.1 Cel pracy 5
1.2 Rozwiązania przyjęte w pracy 5
1.3 Rezultat pracy 6
1.4 Organizacja pracy 6
2 Mapy internetowe 8
2.1 Dostępne rodzaje interaktywnych mapy internetowych 8
2.1.1 Google Maps 8
2.1.2 Yahoo Maps 10
2.1.3 Bing Maps 11
2.2 Dostępne rozwiązania do tworzenia spersonalizowanych map 12
2.2.1 Google My Maps 12
2.2.2 MapVivo 13
2.2.3 EnTrip 14
2.2.4 Tagzania 14
2.3 Wady dostępnych rozwiązań 15
3 Narzędzia i technologie 17
3.1 Platforma .NET i język C# 17
3.2 Visual Studio 19
3.3 ReSharper 20
3.4 Framework ASP.NET MVC 20
3.5 Microsoft SQL Server 22
3.6 NHibernate 23
3.6.1 Fluent NHibernate 24
3.6.2 NHibernate LINQ 25
3.7 jQuery 26
3.8 NUnit 27
4 Propozycja nowego rozwiązania 29
4.1 Panel administracyjny 29
4.1.1 Ustawienia ogólne 29
4.1.2 Domyślna lokalizacja mapy 29
4.1.3 Struktura danych 30
4.1.4 Formularz dodawania miejsc 30
4.1.5 Zarządzanie miejscami 31
4.2 Spersonalizowana mapa 31
4.2.1 Filtry 31
4.2.2 Lista miejsc 32

4.2.3 Mapa miejsc 32
4.3 Algorytmy i wzory 32
4.3.1 Obliczanie odległości miedzy dwoma współrzędnymi geograficznymi 33
4.3.2 Obliczanie odległości na mapie między dwoma punktami. 34
4.3.3 Grupowanie miejsc na mapie 36
5 Prototyp 38
5.1 Wykorzystane wzorce projektowe 38
5.1.1 Domain Model 38
5.1.2 Unit of Work 39
5.1.3 Repository 39
5.1.4 Command-query Seperation 40
5.1.5 Dependency Injection 40
5.2 Implementacja 41
5.2.1 Opis architektury 41
5.2.2 Model domenowy 44
5.3 Przykład zastosowania 46
5.3.1 Tworzenie nowej mapy 46
5.3.2 Panel administracyjny 48
5.3.3 Zarządzanie ustawieniami 49
5.3.4 Zarządzanie strukturą danych 50
5.3.5 Dodawanie nowych miejsc 52
5.3.6 Zarządzanie miejscami 53
5.3.7 Przeglądanie mapy 54
6 Zalety, wady oraz plany rozwojowe 58
6.1 Zalety oraz wady przyjętych rozwiązań 58
6.2 Proponowane plany rozwoju 59
6.2.1 API 59
6.2.2 Import i eksport danych 59
6.2.3 Internacjonalizacja 59
6.2.4 Szablony interfejsu graficznego 60
6.2.5 Możliwość zagnieżdżenia utworzonej mapy na stronach internetowych 60
7 Podsumowanie 61
Spis rysunków 63
Spis przykładów 64

1 Wstęp

W czasach dzisiejszego Internetu coraz większą rolę odgrywają interaktywne usługi. Jedną z takich usług są mapy internetowe.

Prekursorem w tej dziedzinie jest znany wszystkim internetowy gigant Google, ze swoim Google Maps. Już na początku 2005 [1] roku Google udostępniło darmową usługę pozwalającą oglądać zdjęcia dużej części naszej planety w wysokiej rozdzielczości. Z biegiem czasu usługa rozrastała się oferując coraz większe pokrycie oraz API pozwalające programistom na integrowanie Google Maps ze swoimi stronami internetowymi. Google nie jest jedynym dostawcom interaktywnych map. W jego ślady podążyli inni giganci Internetowi, tj. Microsoft (Bing Maps), Yahoo (Yahoo Maps) czy choćby polski Onet (Zumi).

Większość dostawców map internetowych udostępnia API, dając jednocześnie programistom możliwość rozszerzenie ich usług o dodatkowe funkcjonalności. Istnieje bardzo dużo różnorodnych rozszerzeń, lecz nie ma takiego, które umożliwiłoby w prosty sposób niedoświadczonym użytkownikom Internetu stworzyć własną mapą ze sprecyzowanym przez siebie formatem danych z możliwością zarządzania i przeszukiwania oraz przyjaznym i łatwym w obsłudze interfejsem graficznym.

1.1 Cel pracy

Głównym celem niniejszej pracy jest zbadanie istniejących na rynku rodzajów map internetowych oraz rozwiązań do ich personalizacji. W wyniku przeprowadzonych badań opracowane zostaną cechy i wymagania funkcjonalne, jakimi powinna się charakteryzować nowoczesna aplikacja umożliwiająca personalizację map internetowych.

Celem pośrednim pracy jest zbudowanie aplikacji umożliwiającej tworzenie spersonalizowanych map internetowych. Program ten powinien się cechować:

- Łatwością tworzenia map dostosowanych do potrzeb użytkowników
- Prostotą precyzowania rodzaju danych wyświetlanych na mapach
- Wysoką łatwością użytkowania (usability)
- Przyjemnym i interaktywnym interfejsem użytkownika

1.2 Rozwiązania przyjęte w pracy

W pracy została wykorzystana technologia .NET 3.5. Struktura aplikacji została zbudowana przy użyciu frameworku ASP.NET MVC 2.0, który cechuje się wysoką

modułowością, daje duże możliwości konfiguracji i jest bardzo łatwo testowalny. Rdzeniem aplikacji jest bogaty model, który hermetyzuje jak największą ilość funkcjonalności. Model ten jednocześnie jest wystarczająco generyczny, aby sprostać potrzebie personalizacji przez użytkowników. Tam gdzie było to potrzebne, wykorzystane zostały znane wzorce i dobre praktyki programistyczne.

Interfejs użytkownika został wykonany z dużym naciskiem na interaktywność i łatwość użytkowania. Podstawową technologią wykorzystywaną po stronie klienta jest AJAX. Dla zwiększenia efektywności pracy z językiem JavaScript wykorzystana została biblioteka jQuery. Jako bazę dla tworzonych spersonalizowanych map zostało użyte Google Maps.

1.3 Rezultat pracy

Rezultatem pracy jest aplikacja webowa pozwalająca tworzyć spersonalizowane mapy internetowe. Za jej pomocą użytkownik może w łatwy sposób skonfigurować swoją mapę, ustalić jaki format danych będzie wprowadzał oraz zarządzać miejscami. Tak utworzoną i skonfigurowaną mapę, można przeglądać oraz filtrować obiekty na niej wyświetlanie. Filtrowanie odbywa się za pomocą generycznych kryteriów wygenerowanych na podstawie wcześniej ustalonego przez użytkownika formatu danych.

Wytworzona aplikacja posiada przyjazny interfejs użytkownika i cechuje się dużą łatwością użytkowania. Dzięki wykorzystaniu technologii AJAX oraz komponentów drag and drop aplikacja jest wysoce interaktywna.

1.4 Organizacja pracy

Praca rozpoczyna się od omówienia dostępnych na rynku rodzajów map internetowych. Opisuje możliwości, wady i zalety każdego z rozwiązań. Następnie przedstawione są aplikacje, które rozszerzają funkcjonalność map o możliwość ich personalizacji.

W trzecim rozdziale omówione są narzędzia i technologie, jakie były wykorzystywane przy tworzeniu tej pracy.

W czwartym rozdziale została zaproponowana funkcjonalność systemu, umożliwiającego tworzenie spersonalizowanych map oraz omówione niezbędne wzory i algorytmy potrzebne do realizacji założeń.

W piątym rozdziale omówiony został sposób implementacji systemu. Przedstawiona została zarówno ogólna architektura systemu, jak i implementacje najważniejszych algorytmów.

W szóstym rozdziale opisane zostały zauważone przez autora wady i zalety wytworzonego prototypu oraz proponowane plany rozwoju.

2 Mapy internetowe

Na rynku istnieje kilku większych dostawców interaktywnych map internetowych. W pierwszym podrozdziale zostaną opisani główni konkurenci w tej dziedzinie, w drugim podrozdziale przedstawione będą dostępne rozwiązania do tworzenie spersonalizowanych map, natomiast w ostatnim omówione zostaną ich wady.

2.1 Dostępne rodzaje interaktywnych mapy internetowych

2.1.1 Google Maps

Google Maps (pierwotnie Google Locale) to zbiór internetowych usług kartograficznych udostępnianych przez firmę Google, w skład których wchodzą m.in. Google Maps websites, Google Ride Finder, Google Transit oraz Google Maps API. Jak twierdzi jeden z twórców (Lars Rasmussen) misją Google Maps jest "uporządkowanie światowych zasobów informacji w sposób geograficzny" [2]. Mapy te oferują takie funkcjonalności jak:

- Mapy uliczne.
- Wyznaczanie trasy dla różnych środków transportu (pieszo, samochodem, komunikacją miejską).
- Geokoder – parsujący adresy podane w języku naturalnym.
- Steet View – panoramiczny widok miast z poziomu ulic.

Odzwierciedlenie powierzchni Ziemi w Google Maps jest dokonane za pomocą zmodyfikowanego odwzorowania Merkatora. Odwzorowanie to jest dokładne na równiku, ale wraz z oddalaniem się od niego w stronę biegunów rośnie deformacja. Z tych powodów Google Maps nie uwzględnia rejonów okołobiegunowych. Dokładniejsze odwzorowania (z uwzględnieniem całego globu) zastosowane zostało w spokrewnionym produkcie - Google Earth, który oferuje trójwymiarowy podgląd Ziemi.

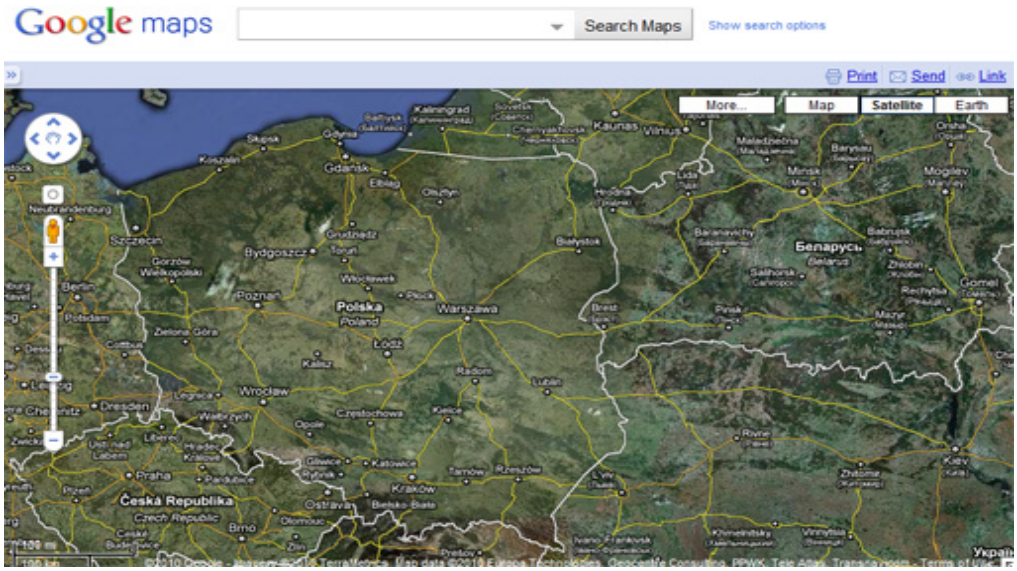
Zdjęcia udostępniane są w wysokiej rozdzielczości, w kilkunastu różnych skalach. Maksymalny poziom zbliżenia jest zależny od miejsca – w większych miastach da się bez problemu odróżnić nawet pojedyncze drzewa, samochody. Z drugiej strony dużą część świata można oglądać jedynie z dalszej perspektywy. Mając na względzie kwestie bezpieczeństwa i obawy przed atakami terrorystycznymi niektóre regiony (głównie w Stanach Zjednoczonych) zostały zamazane. Wbrew potocznemu myśleniu, jedynie mały odsetek zdjęć został wykonany za pomocą satelitów. Większość zdjęć o

wysokiej rozdzielczości i dużym zbliżeniu została wykonana z wykorzystaniem technik lotniczych.

Tak jak wszystkie aplikacje ze stajni Google również Google Maps kładzie duży nacisk na wykorzystywanie najnowszych technik internetowych. Zarówno wyszukiwanie, jak i przeglądanie map odbywa się bez konieczności przeładowywania strony. Zdjęcia są ładowane dynamicznie w momencie kiedy użytkownik zmienia pole widzenia. Cała komunikacja odbywa się za pomocą technologii AJAX oraz wykorzystuje JSON, jako format przesyłu danych między klientem a serwerem.

Google Maps udostępnia programistom API dając im możliwość integracji z własnymi stronami WWW oraz rozbudowy usługi o nowe funkcjonalności. Pierwotnie API było oferowane tylko w języku JavaScript, jednak z biegiem czasu zostało rozszerzone o technologie Adobe Flash, usługę umożliwiającą generowanie statycznych map, geokoder parsujący adresy w języku naturalnym oraz możliwość wyznaczania trasy podróży. Według niektórych źródeł [3] ponad 350000 stron aktywnie używa Google Maps API czyniąc je jednym z najintensywniej wykorzystywanych internetowym API.

Na rys. 1 przedstawiony został widok przykładowy widok mapy w Google Maps. Mapa ta składa się z pola do wyszukiwania oraz kilku prostych kontrolerów umożliwiających m.in. zmianę widocznego obszaru i zbliżenia oraz wybranie widoku satelitarnego lub ulicznego.



Rysunek 1 - Google Maps

2.1.2 Yahoo Maps

Yahoo Maps jest produktem trochę młodszym od Google Maps, dlatego też nie posiada aż tak bogatej funkcjonalności, jednakże główne funkcje są porównywalne. Do najistotniejszy możliwość należą:

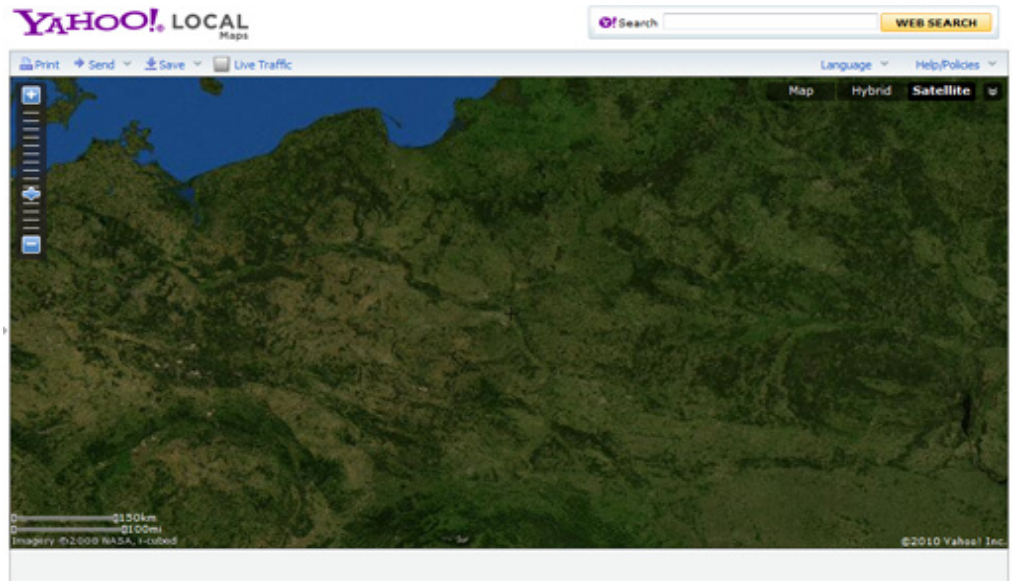
- Książka adresowa – umożliwiająca zarejestrowanym użytkownikom przechowywanie swoich często używanych adresów.
- Usługa „Live Traffic" – informująca na bieżąco o poziomie intensywności ruchu na autostradach oraz o wypadkach (dotyczy tylko Stanów Zjednoczonych).
- Point of Interest Finder – informująca o wybranych typach punktów zainteresowań w pobliżu zaznaczonej lokalizacji.
- Wyznaczanie trasy podróży – pozwalająca wyświetlić na mapie oraz w formie opisowej krok po kroku trasę od punktu A do B.

Yahoo Maps, tak jak Google Maps, odzwierciedla powierzchnię Ziemi na płaszczyźnie za pomocą odwzorowania Merkatora. Zdjęcia również są wykonane z wykorzystaniem zarówno technologii satelitarnych, jak i lotniczych, jednakże w porównaniu do Google Maps, nie licząc obszaru Stanów Zjednoczonych, charakteryzują się gorszą jakością oraz mniejszym zbliżeniem.

Yahoo Maps intensywnie wykorzystuje najnowsze technologie internetowe. Całość wykonana jest z użyciem technologii AJAX, która zapewnia wysoką interaktywność rozwiązania. Duży nacisk został położony na różnorodność API. Do użytku programistów oddane są trzy rodzaje API:

- Flash API – API dla technologii Adobe Flash, umożliwiające wykorzystanie języków JavaScript, ActionScript i Adobe Flex 1.5
- AJAX API – API dla developerów piszących w JavaScript, niezamierzających wykorzystywać technologii Adobe Flash.
- „Simple API" – proste API do wyświetlania danych dostarczonych w formacie XML.

Na rys. 2 przedstawiony został widok satelitarny terytorium Polski w Yahoo Maps. Podobnie jak w Google Maps na mapie znajduje się pole do wyszukiwania i zestaw standardowych kontrolerów.



Rysunek 2 - Yahoo Maps

2.1.3 Bing Maps

Bing Maps (wcześniej Live Search Maps, Windows Live Maps i Windows Live Local) jest usługą interaktywnych map internetowych wchodzącą w skład pakietu usług Microsoft Bing. Funkcjonalność jest bardzo zbliżona do tej udostępnianej przez konkurencję i składa się z:

- Map ulicznych – zawierających punkty zainteresowań, tj. stacje metra, stadiony, szpitale i inne.
- Widoku satelitarnego – zawierającego zdjęcia wysokiej rozdzielczości zrobione zarówno za pomocą technologii satelitarnej, jak i lotniczej.
- Widoku „Bird's eye" – pozwalającego zobaczyć zdjęcia lotnicze ponad 100 lokalizacji w Stanach Zjednoczonych, Kanadzie, Japonii i około 80 w Europie z czterech różnych kątów.
- Streetside – udostępniającego panoramiczny widok miast z poziomu ulic.
- ClearFlow – które pokazuje informacje o ruchu na ulicach amerykańskich miast i pozwala z jego uwzględnieniem wyznaczyć optymalną trasę z punktu A do B.

Tak jak u konkurencji, również w przypadku Bing Maps do odzwierciedlenia powierzchni Ziemi na płaszczyźnie wykorzystane jest odwzorowanie Merkatora. Jakość zdjęć jest porównywalna do jakości w Yahoo Maps natomiast gorsza niż w Google Maps. Bing Maps również udostępnia API dla programistów oraz intensywnie wykorzystuje najnowsze technologie internetowe takie jak AJAX.

Rys. 3 przedstawia widok satelitarny terytorium Polski w Bing Maps. Tak jak we wcześniej omawianych mapach, również w tym przypadku dostępne jest pole do wyszukiwania oraz zestaw standardowych kontrolerów.

Rysunek 3 - Bing Maps

2.2 Dostępne rozwiązania do tworzenia spersonalizowanych map

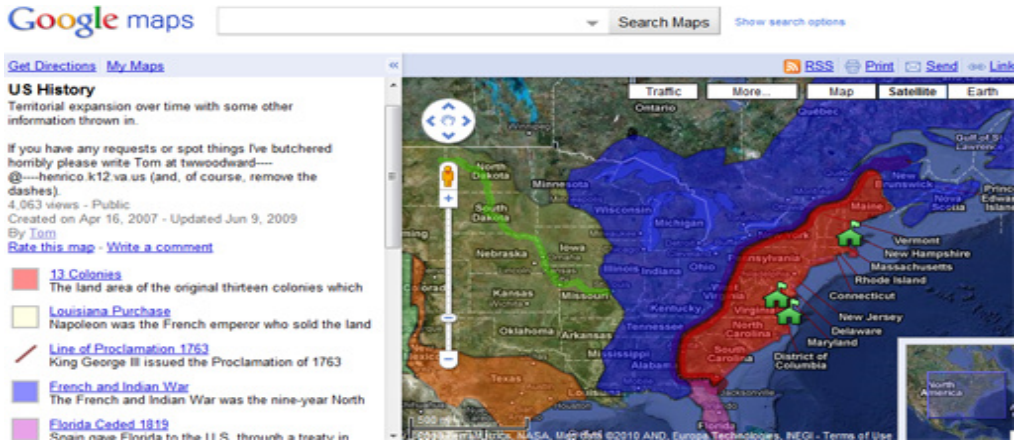
Większość dostawców interaktywnych map internetowych udostępnia API dla programistów umożliwiając im w ten sposób rozbudowywanie funkcjonalności map. Jednym z ciekawszych rozszerzeń jest możliwość personalizacji map. Poniżej zostanie przestawionych kilka ciekawszych rozwiązań umożliwiających użytkownikom tworzenie własnych spersonalizowanych map internetowych.

2.2.1 Google My Maps

W maju 2007 Google rozszerzyło możliwości swoich map o kolejną funkcjonalność. My Maps pozwala użytkownikom personalizować mapy poprzez umieszczanie na nich znaczników, ścieżek oraz wielokątów. Usługa ta udostępnia użytkownikom standardowy zestaw ikon dla znaczników oraz możliwość zmiany kolorów linii i wielokątów. Oprócz tego, utworzone mapy mogą być zapisane do późniejszej edycji oraz udostępniane innym użytkownikom. Każdy element dodany na mapę może mieć dodany opis w postaci tekstu albo kodu HTML.

Rozwiązanie to cechuje się sporą generycznością, ponieważ umożliwia umieszczenie na mapie dowolnego typu danych, którym da się przypisać lokalizacje geograficzną. Niestety Google My Maps uwzględnia tylko jedno pole (opis) dla wprowadzonych danych, co ogranicza możliwość przeszukiwania danych jedynie do

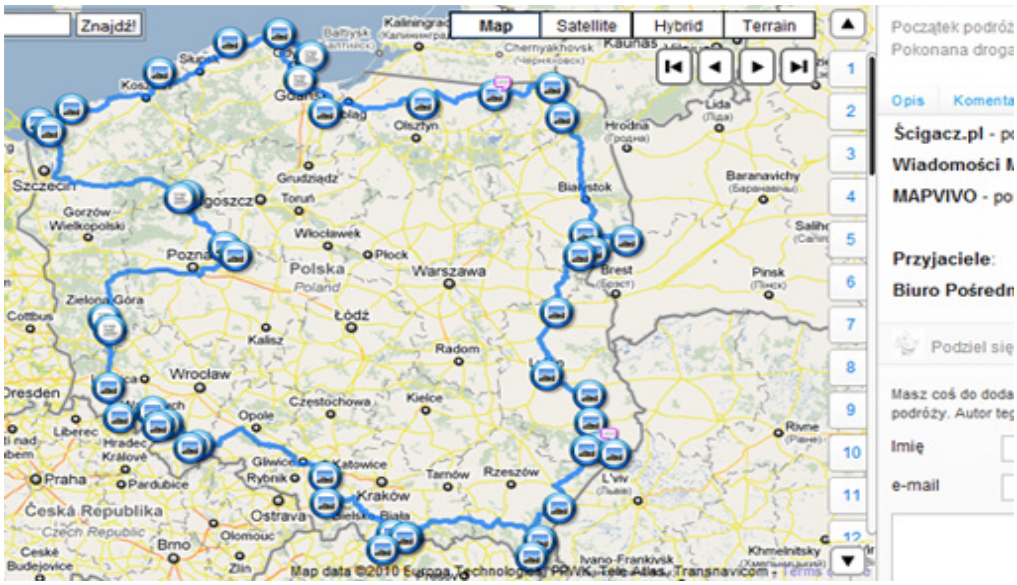
filtrowania po słowach kluczowych. Rys. 4 przedstawia przykładową mapę utworzoną za pomocą Google My Maps.



Rysunek 4. Google My Maps - przykładowa mapa

2.2.2 MapVivo

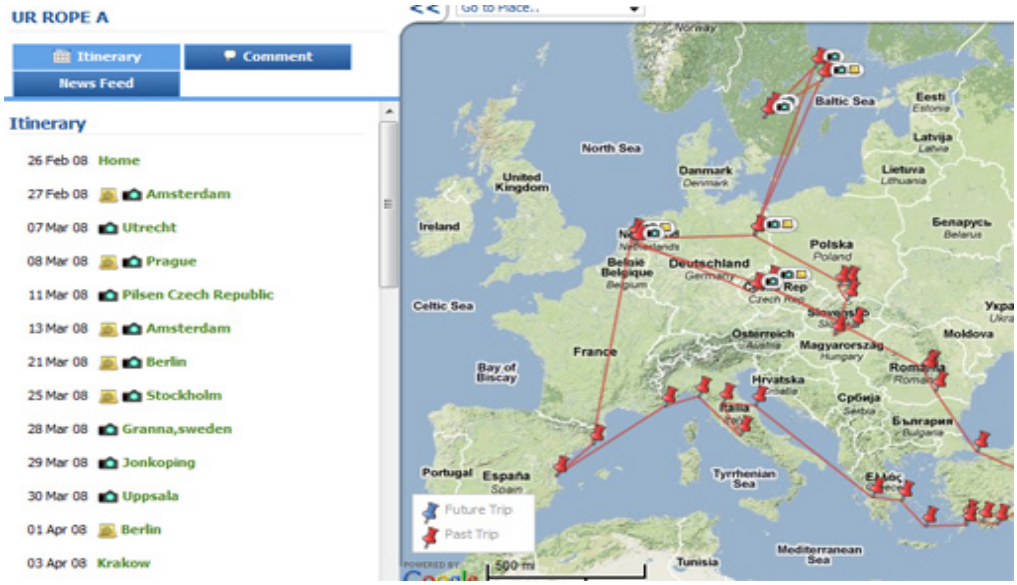
Autorzy MapVivo twierdzą, że ich portal to „Sposób by pokazać odwiedzone miejsca i kraje – dzielić się wrażeniami z bliskimi, pokazać im zdjęcia, filmiki i trasy swych podróży na mapie. Wirtualny dziennik podróży, który możesz w każdej chwili uzupełniać, nawet w drodze, a także zaplanować za jego pomocą nowe wyprawy i pokazać to wszystko, komu tylko zapragniesz.” [4]. Można powiedzieć, że jest to blog podróżnika z możliwością wizualizacji na mapie. Z powodu ścisłego powiązania z konkretną tematyką – podróżami – rozwiązanie cechuje się bardzo niskim poziomem generyczności. Na rys. 5 przedstawiona została mapa obrazująca przykładową podróży wokół Polski.



Rysunek 5. MapVivo - przykładowa mapa

2.2.3 EnTrip

EnTrip podobnie jak MapVivo jest portalem dla podróżników. Pozwala on swoim użytkownikom wizualizować wyprawy na mapie. Udostępniona funkcjonalność jest bardzo podobna do MapVivo. Dodatkowo odznacza się drobnymi dodatkami, tj. statystykami dotyczącymi m.in. łącznego przebytego dystansu. Zaproponowana przez autora funkcjonalność sprawia, że EnTrip jest portalem bardziej użytecznym. Ponadto EnTrip na tle swojego konkurenta wyróżnia się lepiej zrobionym interfejsem, który w całości wykonany jest z wykorzystaniem technologii AJAX. Rys. 6 przedstawia przykładową mapę podróży po Europie stworzoną w portalu EnTrip.



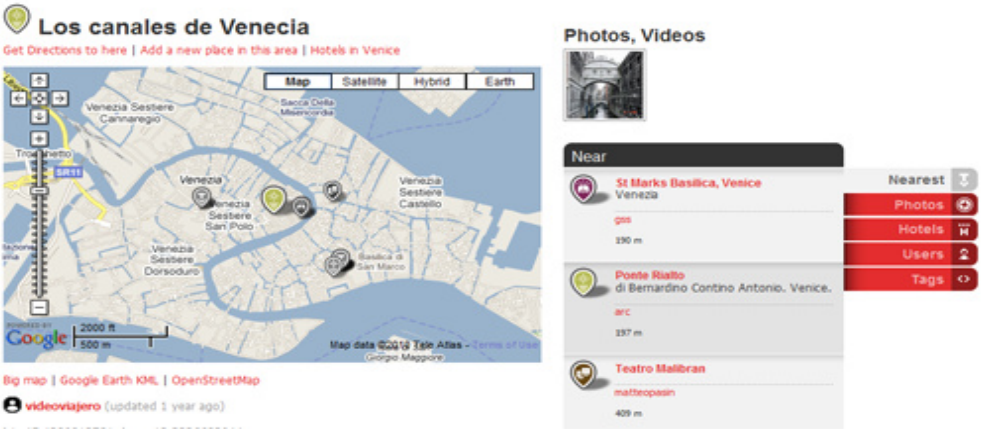
Rysunek 6. EnTrip - przykładowa mapa

2.2.4 Tagzania

Tagzania pozwala personalizować mapy za pomocą punktów, ścieżek i wielokątów. Funkcjonalność jest zbliżona do tej udostępnianej przez Google My Maps. Lokalizacje można opisywać, dodawać do nich zdjęcia, filmy oraz je otagowywać. Funkcja tagowania wydaje się być najciekawszą, ponieważ pozwala tworzyć mapy miejsc dodanych przez różnych użytkowników, które są ze sobą powiązane za pomocą słów kluczowych.

Funkcja tagowania oraz brak powiązania z konkretną tematyką sprawia, że rozwiązanie cechuje się wysokim poziomem generyczności. Niestety tak samo jak w Google My Maps oprócz możliwości dodania tytułu, opisu, zdjęć i filmów, również w przypadku Tagzanii nie ma możliwości dodania dodatkowych pól ze sprecyzowanym przez użytkownika rodzajem danych.

Rys. 7 przedstawia mapę stworzoną za pomocą Tagzanii, która zawiera informacje o ciekawych miejscach w Wenecji.



Rysunek 7. Tagzania - przykładowa mapa

2.3 Wady dostępnych rozwiązań

Wszyscy dostawcy interaktywnych map internetowych oferują podobną funkcjonalność i udostępniają rozbudowane API dające programistom możliwość tworzenia nowych rozszerzeń. Najbardziej znaczącą różnicą jest jakość zdjęć w poszczególnych rozwiązaniach. Google Maps oferuje bardzo dobrą jakość w miastach i zadowalającą na obszarach mniej zaludnionych na całym świecie. Zarówno Yahoo Maps, jak i Bing Maps oferuje podobną jakość, ale głównie na terenie Stanów Zjednoczonych. Poza USA zdjęcia są wyraźnie gorszej jakości oraz cechuje je mniejszy poziom zbliżenia.

Jeśli chodzi o rozwiązania umożliwiające personalizację map, wszystkie udostępniają możliwość dodawania miejsce na mapie wraz z ich opisem, zdjęciami i filmami. Tagzania dodatkowo umożliwia tagowanie lokalizacji pozwalając w ten sposób tworzyć mapy powiązanych miejsc dodanych przez różnych użytkowników. Z wymienionych rozwiązań EnTrip cechuje się najbardziej interaktywnym interfejsem graficznym, pozostałe rozwiązania nie dorównują mu pod tym względem. Rzeczą, której brakuje wszystkim jest możliwość personalizacji formatu wprowadzanych danych i zaawansowanego sposobu ich przeszukiwania. Żadne rozwiązanie nie daje możliwość dodania dodatkowych pól wraz z określeniem rodzaju wprowadzanych danych. W związku z tym nie umożliwiają filtrowania miejsc na mapie za pomocą innych kryteriów niż słowa kluczowe.

Autor pracy zamierza opracować rozwiązanie wolne od wyżej wymienionych wad. Wytworzona aplikacja będzie posiadała bogaty, łatwy w użyciu interfejs.

Użytkownik korzystając z tej aplikacji będzie miał dostęp do zaawansowanych opcji tj. filtrowanie wyświetlanych miejsc. Dzięki wysokiemu poziomowi generyczności, opracowane rozwiązanie będzie umożliwiało tworzenie spersonalizowanych map internetowych zawierających dowolnego rodzaju geolokalizowalne informacje.

3 Narzędzia i technologie

3.1 Platforma .NET i język C#

W lutym 2002 [5] roku Microsoft ogłosił wprowadzenie platformy .NET i wchodzącego w jej skład nowego języka programowanie nazwanego C#.

.NET to platforma programistyczna składająca się ze środowiska uruchomieniowego oraz bogatego zbioru bibliotek klas dostarczających standardową funkcjonalność. Wszystkie aplikacje napisane w technologii .NET uruchamiane są w środowisku uruchomieniowym zwanym CLR (ang. Common Runtime Enviroment). Spełnia ono rolę wirtualnej maszyny, dzięki której programiści nie muszą brać pod uwagę konfiguracji sprzętu, na którym aplikacja będzie uruchamiana. CLR dostarcza również inne ważne usługi, tj. zarządzanie bezpieczeństwem, pamięcią oraz obsługę wyjątków.

Główne cechy konstrukcyjne platformy .NET to:

- Interoperacyjność – interakcja między nowymi i starymi aplikacjami jest częstym wymaganiem. Platforma .NET za pomocą przestrzeni nazw System.Runtime.InteropServices, System.EnterpriseServices, oraz sposobu P/Invoke dostarcza sposoby dostępu do funkcjonalności zaimplementowanej poza nią.
- Środowisko uruchomieniowe – CLR pełni w .NET rolę wirtualnej maszyny. Wszystkie programy uruchamiane są pod jego nadzorem mając zapewnione zarządzanie pamięcią i bezpieczeństwem oraz obsługę wyjątków.
- Niezależność języka – platforma .NET wprowadziła CTS (ang. Common Type System). CTS definiuje wszystkie typy danych oraz konstrukcje programistyczne obsługiwane przez CLR i mówi o tym w jaki sposób mogą między sobą oddziaływać. Dzięki temu możliwa jest wymiana typów i instancji obiektów pomiędzy aplikacjami napisanymi w dowolnym języku obsługiwanym przez .NET.
- Standardowy zestaw bibliotek – BCL (ang. Base Class Library) to zestaw bibliotek udostępniających standardową funkcjonalność każdemu z języków platformy .NET. BCL dostarcza funkcjonalność m.in. zapisu i odczytu plików i strumieni, obsługi grafiki, interakcji z bazami danych oraz manipulacji dokumentami XML.

- Uproszczone wdrażanie – w skład platformy .NET wchodzi zestaw narzędzi ułatwiających zarządzanie procesem instalacji oprogramowania. Pozwalają one zapewnić brak konfliktów z wcześniej zainstalowanym oprogramowaniem oraz odpowiednie wymogi bezpieczeństwa.
- Bezpieczeństwo – platforma .NET zapewnia wspólny model bezpieczeństwa wszystkim aplikacjom oraz została zaprojektowana tak, aby sprostać wielu popularnym rodzajom zagrożeń.
- Przenośność – platforma .NET została zaprojektowana tak, aby być niezależną od konkretnej platformy systemowej. Oznacza to, że każda aplikacja napisana w .NET powinna działać na dowolnym systemie operacyjnym z zainstalowaną implementacją .NET. Mimo, że Microsoft nigdy nie zaimplementował .NET pod system inny niż Windows, istnieje pełna otwarta specyfikacja CLI (ang. Common Language Infrastructure). Daje to podmiotom trzecim możliwość napisania implementacji .NET pod inne systemy operacyjne.

C# jest mocno typowanym nowoczesnym wieloparadygmatowym językiem programowanie umożliwiającym programowanie imperatywne, deklaratywne, funkcyjne, generyczne oraz obiektowe. Z założenia jest łatwy w użyciu, cechuje się wysoką ekspresywnością i dużą wydajnością. Jest wzorowany na wielu językach programowanie, jednak najbardziej zauważalne są wpływy języków Java i C++. Był współtworzony przez Andersa Hejlsberga (twórcy języka Delphi) i Scotta Wiltamutha.

Główne cechy języka C#:

- Posiada hierarchię klas, w której istnieje obiekt nadrzędny System.Object. Dotyczy to zarówno klas, jak i typów prostych.
- Zarządzanie pamięcią należy do obowiązków CLR. Wiąże się to z tym, że nie ma potrzeby ręcznej alokacji oraz zwalniania pamięci.
- Posiada delegaty i zdarzenia, które pełnią podobną rolę jak wskaźniki na funkcje w języku C++.
- Posiada mechanizmy refleksji umożliwiające programistyczną analizę struktury kodu. Umożliwia to tworzenie wysoce generycznych rozwiązań operujących na kodzie nieznanym w czasie kompilacji.
- Posiada typy generyczne, dające funkcjonalność zbliżoną do szablonów z C++.

- Posiada możliwość dynamicznego tworzenia kodu w czasie działania aplikacji oraz dołączenia go do aktualnie wykonywanego kodu.
- Posiada bogaty zestaw bibliotek (BCL) umożliwiający tworzenie aplikacji konsolowych, okienkowych, bazodanowych, sieciowych, rozproszonych, internetowych oraz multimedialnych.

W kwietniu 2010 roku wyszła najnowsza wersja zarówno języka C# (4.0), jak i całej platformy .NET (4.0). Z racji tego, że niniejszą pracę rozpoczęto przed tym terminem została ona wykonana z wykorzystaniem starszych wersji: C# 3.0 i .NET 3.5.

3.2 Visual Studio

Visual Studio to zestaw narzędzi programistycznych (ang. IDE – Integrated Development Environment) umożliwiający tworzenie m.in. aplikacji konsolowych, z interfejsem graficznym, webowych oraz webserwisów.

Głównym komponentem Visual Studio jest edytor kodu z wbudowanym auto uzupełnianiem (ang. IntelliSense) oraz refaktoringiem kodu. Posiada zintegrowany debbuger działający zarówno na poziomie kodu źródłowego, jak i maszyny. Dodatkowo wbudowane zostały narzędzia do wizualnej budowy aplikacji z interfejsem graficznym, edytor klas, edytor schematu baz danych oraz wiele innych. Visual Studio posiada również możliwość zainstalowania specjalnych wtyczek, które mogą w sposób praktycznie nieograniczony rozszerzać jego funkcjonalność. Wtyczki takie mogą m.in. dodać możliwość wersjonowania kodu za pomocą repozytoriów, dostarczać nowych edytorów wizualnych lub wprowadzić nowe metody refaktoringu.

Visual Studio standardowo obsługuje wiele różnych języków programowania. Domyślnie obsługiwanymi językami są C/C++, VisualBasic .NET, C#, F# oraz XML/XSLT, HTML/XHTML, JavaScript i CSS. Obsługę dodatkowych języków można dodać poprzez tzw. usługi językowe.

Jak większość produktów ze stajni Microsoftu, które są przeznaczone dla programistów, Visual Studio jest dostępne za darmo w wersji Express. Pozostałe wersje są płatne, jednak studenci mają możliwość uzyskania darmowej licencji.

Najnowszą wersją Visual Studio jest wersja 2010, która została zaprezentowana wraz z wprowadzeniem platformy .NET 4.0. Niniejsza praca została wykonana z wykorzystaniem poprzednie wersji – Visual Studio 2008.

3.3 ReSharper

ReSharper jest wtyczką do Visual Studio wyprodukowaną przez firmę JetBrains, która rozszerza funkcjonalność tego IDE o dodatkowe możliwości, znacznie polepszające produktywność programistów.

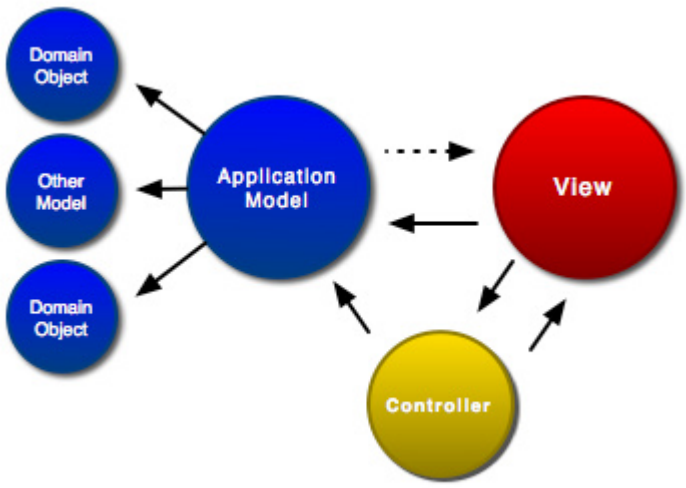
ReSharper dodaje taką funkcjonalność jak:

- wykrywanie błędów „w locie” (bez potrzeby kompilowania kodu)
- usprawnione sposoby korekcji błędów
- bardziej zaawansowane i wydajne uzupełnianie kodu
- usprawnioną nawigacje po strukturze projektu i przeszukiwanie plików w nim zawartych
- rozbudowane i konfigurowalne formatowanie kodu
- dodatkowy zestaw często wykorzystywanych szablonów
- możliwość automatycznej optymalizacji
- wbudowane narzędzie do testów jednostkowych obsługujące większość popularnych frameworków.

Oprócz bardzo bogatej funkcjonalności, ReSharper udostępnia również API, za pomocą którego programiści mogą jeszcze dodatkowo rozszerzyć jego możliwości. Przy pisaniu tej pracy autor wykorzystywał najnowszą wersję ReSharper oznaczoną numerem 5.0.

3.4 Framework ASP.NET MVC

ASP.NET MVC (patrz rys. 8) to technologia do tworzenia aplikacji webowych, która implementuje wzorzec MVC [6] (ang. Model-View-Controller).



Rysunek 8 - Framework ASP.NET MVC

ASP.NET MVC to alternatywa dla klasycznego ASP.NET, notabene na podstawie którego została zbudowana. Jest bardzo lekkim, łatwo testowalnym frameworkiem, który wykorzystuje najlepsze techniki z klasycznego ASP.NET. MVC jest standardowym wzorcem projektowym, który programiści bardzo dobrze znają. Pozwala budować aplikacje z podziałem na komponenty:

- Model – część aplikacji, która składa się z modelu danych wraz z implementacją operacji, które mogą być na nich wykonywane. Warstwa ta odpowiedzialna jest też za powiadamianie pozostałych warstw o zmianach w niej zachodzących. Stan modelu najczęściej zapisywany jest w trwały sposób np. za pomocą bazy danych.
- Widok – warstwa odpowiadająca za wyświetlanie interfejsu użytkownika. Widok wypełniany jest za pomocą danych z modelu.
- Kontroler – część odpowiedzialna za obsługę interakcji użytkownika, manipulowanie modelem i wybranie widoku do wyświetlenia.

Wzorzec ten dokładnie mówi, gdzie mają się znaleźć konkretne aspekty aplikacji, pozwala skupić się na każdym z nich z osobna, dzięki czemu pomaga zarządzać jej złożonością. Wynikiem tego jest aplikacja, która składa się z luźno ze sobą powiązanych komponentów, co z kolei czyni ją bardzo łatwo testowalną.

Główne cechy ASP.NET MVC to:

- Rozdzielenie zadań aplikacji – obsługa danych wprowadzonych przez użytkownika, logiki biznesowej i interfejsu użytkownika odbywa się w osobnych luźno powiązanych ze sobą warstwach.
- Łatwa testowalność – luźna budowa frameworku i intensywne wykorzystywanie interfejsów daje możliwość łatwego testowania np. poprzez wykorzystanie metodyki TDD (ang. Test Driven Developement).
- Rozszerzalność i wymienialność komponentów – większość komponentów w ASP.NET MVC została zaprojektowana w taki sposób, aby była możliwość ich rozszerzenia lub łatwej wymiany na inna implementacje. Można podłączyć np. własny silnik widoków lub politykę routingu URLi. Dodatkowo cały framework został zbudowany tak, aby z łatwością można było go zintegrować z kontenerem IoC (ang. Inversion of Control).
- Mapowanie URLi - ASP.NET MVC posiada rozbudowany silnik pozwalający mapować przyjazne URLe na akcje w kontrolerach.

- Wsparcie dla składni ASP.NET – znane z klasycznego ASP.NET strony .aspx, kontrolki .ascx są wykorzystywane w domyślnym silniku widoków. Umożliwia to korzystanie ze znanej programistom składni językowej.
- Wsparcie dla istniejących funkcjonalności klasycznego ASP.NET – ASP.NET MVC pozwala korzystać ze znanych mechanizmów, tj. autoryzacja, uwierzytelnianie, role, cachowanie czy sesji.

ASP.NET MVC był tworzony w ciekawy i odmienny dla Microsoftu sposób. W grudniu 2007 roku została wydana tzw. wersja CTP (ang. Community Technology Preview**)**. Idea polegała na tym, aby pokazać programistom prawie gotowy produkt, zebrać od nich jak największą liczbę opinii a następnie wprowadzić takie poprawki, aby jak najlepiej spełnić ich oczekiwania. Finalnie ASP.NET MVC 1.0 zostało zaprezentowane w marcu 2009 roku. Niniejsza praca zbudowana jest z wykorzystaniem wersja numer 2.0, która została wydana rok po premierze pierwszej.

3.5 Microsoft SQL Server

Microsoft SQL Server jest rozbudowaną platformą bazodanową typu klient-serwer charakteryzującą się dużą wydajnością, niezawodnością oraz możliwością skalowania. Wraz z Oracle, Microsoft SQL Server uważane jest za jedno z dwóch najbardziej kompleksowych rozwiązań dostępnych na rynku.

W skład platform Microsoft SQL Server wchodzą:

- Relacyjna baza danych – której silnik jest nowoczesny, bezpieczny, skalowany, niezawodny oraz cechuje się wysokim poziomem dostępności. Baza ta jest zgodna ze standardem ANSI/ISO, a jej głównym językiem zapytań jest Transact-SQL. W wersji 2005 została wprowadzona integracja z CLR umożliwiając tym samym pisanie procedur składowanych w językach C# i Visual Basic .NET.
- Usługi replikacji danych – używane przez MS SQL Server do replikacji i synchronizacji baz danych. W najnowszych wersjach MS SQL Server istnieją trzy rodzaje replikacji: migawkowa, transakcyjna i replikacja przez scalanie.
- Usługi powiadamiania – pozwalają dostarczać spersonalizowane wiadomości nawet do milionów odbiorców.
- Usługi integracyjne – służą do integrowania danych z różnych źródeł.
- Usługi analityczne – umożliwiają analityczne przetwarzanie danych w trybie on-line (ang. OLAP - On-Line Analytical Processing).

- Usługi raportowania – pozwalają generować zarówno tradycyjne drukowane raporty, jak i interaktywne, wyświetlane za pomocą przeglądarki internetowej.
- Narzędzia do zarządzania – umożliwiające zaawansowane zarządzanie bazami oraz precyzje ich dostrajanie.
- Narzędzia programistyczne – które są silnie zintegrowane z Visual Studio i pozwalają na programowanie zaawansowanych procesów.

Podobnie jak Visual Studio, Microsoft SQL Server dostępne jest za darmo w wersji Express. Nałożone na nią ograniczenia limitu jednego procesora i 1 GB pamięci RAM, sprawiają że jest przeznaczona do mniejszych rozwiązań tj. nieduże aplikacje internetowe czy bazy danych dla małej firmy. Aplikacja wytworzona na potrzeby niniejszej pracy korzysta z SQL Server 2008 w wersji Express.

3.6 NHibernate

NHibernate to bardzo popularny i rozbudowany obiektowo-relacyjny mapper (ang. ORM - Object-relational mapper) dla platform .NET. Jest to framework umożliwiający mapowanie obiektowo zorientowanego modelu w taki sposób, aby można go było zapisać w tradycyjnej relacyjnej bazy danych. Głównym celem NHibernate jest wyeliminowanie konieczności pisania dużych ilości kodu służących tylko i wyłącznie persystencji modelu.

NHibernate jest darmowym narzędziem, o otwartym kodzie udostępnionym na licencji GNU. Jest portem Javowego mappera Hibernate do środowiska .NET.

NHibernate dostarcza transparentną persystencję dla tzn. obiektów POCO (ang. Plain Old CLR Objects). POCO to obiekty, które zostały oprogramowane w taki sposób, aby nie były ściśle powiązane z żadnym frameworkiem.

Oprócz funkcji mapowania NHibernate posiada bardzo dobrze rozwinięte API, które umożliwia programistom dodawanie, usuwanie, aktualizowanie oraz w zaawansowany sposób wyszukiwanie obiektów w bazie danych. NHibernate domyślnie umożliwia wyszukiwanie za pomocą:

- HQL (Hibernate Query Language) – potężnego języka zapytań ze składnią przypominającą język SQL, w przeciwieństwie do którego jest w pełni obiektowo-zorientowany.
- Criteria API – które pozwala w przejrzysty sposób tworzyć zapytania za pomocą specjalnego zestawu metod.

Pierwsza wersja NHIbernate powstała na podstawie jego javowego odpowiednika – Hibernate 2.1. W niniejszym projekcie wykorzystywana jest wersja Hibernate 2.1, która jednocześnie jest najnowsza stabilną wersją tego ORMa.

3.6.1 Fluent NHibernate

Do mapowania danych relacyjnych na obiektowe, NHibernate używa specjalnych plików HBM, które są zapisywane w formacie XML. Każda klasa posiada odpowiadający jej plik HBM XML, który mapuje ją do struktury w bazie danych. Oddzielenie kodu klas od informacji o ich mapowaniu ma niewątpliwie zalety, jednakże prowadzi też do niepożądanych sytuacji. W związku z tym, że kompilator nie przetwarza plików XML, nie możliwe jest korzystanie z metod refaktoringu. Wiąże się to z tym, że w momencie wprowadzania zmian w klasach należy pamiętać o tym , żeby te same zmiany wprowadzić również w plikach mapujących. Dodatkowo z racji tego, że XML jest niekompilowany, o większości błędów w mapowaniu dowiadujemy się dopiero w czasie wykonywania programu.

Fluent NHibernate oferuje alternatywę dla standardowego mapowania. Zamiast pisać niekompilowalne dokumenty XML, Fluent NHibernate umożliwia mapowanie za pomocą mocno typowanego kompilowalnego kodu C#. Takie rozwiązanie umożliwia korzystanie z refaktoringu, poprawia czytelność mapowania oraz skraca jego długość. Oprócz alternatywnego sposobu mapowania, Fluent NHibernate daje możliwość:

- Automapowania za pomocą refleksji.
- Pełnej konfiguracji za pomocą wbudowanego API.
- Łatwego sprawdzenia prawidłowości mapowania z pomocą testów jednostkowych.

Poniżej porównany został sposób mapowania tej samej klasy za pomocą HBM XML i Fluent NHibernate. Przykład 1 przedstawia mapowanie za pomocą HBM XML. Przykład 2 pokazuje mapowanie za pomocą Fluent NHibernate.

Przykład 1 - Mapowanie za pomocą HBM XML [7]

```
<?xml version="1.0" encoding="utf-8" ?>
<hibernate-mapping xmlns="urn:nhibernate-mapping-2.2"
  namespace="QuickStart" assembly="QuickStart">

  <class name="Cat" table="Cat">
    <id name="Id">
      <generator class="identity" />
    </id>

    <property name="Name">
      <column name="Name" length="16" not-null="true" />
    </property>
    <property name="Sex" />
    <many-to-one name="Mate" />
    <bag name="Kittens">
      <key column="mother_id" />
        <one-to-many class="Cat" />
      </bag>
  </class>
</hibernate-mapping>
```

Przykład 2 - Mapowanie za pomocą Fluent NHibernate [7]

```
public class CatMap : ClassMap<Cat>
{
  public CatMap()
  {
    Id(x => x.Id);
    Map(x => x.Name)
      .Length(16)
      .Not.Nullable();
    Map(x => x.Sex);
    References(x => x.Mate);
    HasMany(x => x.Kittens);
  }
}
```

3.6.2 NHibernate LINQ

Gdy w 2006 roku Microsoft wprowadził na rynek framework .NET 3.0 z obsługą nowego, w pełni zintegrowanego języka zapytań LINQ (Language Integrated Query), kwestią czasu było, kiedy zacznie on być obsługiwany przez NHibernate. NHibernate LINQ umożliwia programistom pisanie natywnych zapytań, które są tłumaczone przez framework na zapytania do bazy danych. Dzięki temu, że zapytania te są silnie zintegrowane z językiem programowania, możliwa jest walidacja ich poprawności już w czasie kompilacji. Obecna wersja NHibernate LINQ jeszcze nie obsługuje w pełni języka LINQ, jednakże trwają prace nad nową wersją, której premiera jest planowana wraz z premierą NHibernate 3.0.

Poniżej zaprezentowane zostało to samo zapytanie napisane na trzy różne sposoby. Przykład 3 przedstawia zapytanie w języku HQL. Przykład 4 wykorzystuje Criteria API. Przykład 5 pokazuje zapytanie w języku LINQ.

Przykład 3 - Przykładowe zapytanie w HQL

```
session.CreateQuery("from Person p where p.Name == ‘Stefan’ and p.Age >
25").List<Person>();
```

Przykład 4 - Przykładowe zapytanie z wykorzytaniem Criteria API

```
session.CreateCriteria<Person>()
    .Add(Restrictions.Eq("Name","Stefan"))
    .Add(Restrictions.Gt("Age", 25))
    .List<Blog>();
```

Przykład 5 - Przykładowe zapytanie w LINQ

```
from p in session.Linq<Person>() where p.Name == "Stefan" and p.Age > 25
```

3.7 jQuery

jQuery jest biblioteką, której głównym celem jest uproszczenie sposobu pisania skryptów w języku JavaScript. Dzięki temu, że jest bardzo łatwa w użyciu i obsługuje wszystkie popularne przeglądarki internetowe, jest obecnie najczęściej używaną biblioteką JavaScriptową. Jest to biblioteka darmowa, z otwartym kodem źródłowym, udostępniona zarówno na licencji MIT License, jak i GNU General Public License.

Składnia jQuery jest zaprojektowana tak, aby uczynić łatwiejszym nawigowanie po dokumencie, zaznaczanie elementów DOM, tworzenie animacji i obsługę zdarzeń. Dodatkowo biblioteka udostępnia programistom możliwość tworzenia specjalnych wtyczek rozszerzających jej funkcjonalność. Używając wszystkich tych funkcjonalności programiści są w stanie z dużą łatwością stworzyć bardzo zaawansowane, dynamiczne strony internetowe.

jQuery jest obecnie tak popularne, że giganci tj. Microsoft czy Nokia zdecydowali się zintegrować z nim niektóre swoje produkty. Microsoft włączył jQuery do Visual Studio integrując je tym samym z istniejącym już Microsoft ASP.NET AJAX Framework i ASP.NET MVC Framework. Nokia natomiast zintegrowała jQuery ze swoją platformą Web Run-Time.

Najważniejsze cechy jQuery to:

- Silnik Sizzle – który przeźroczyście obsługuje każdą przeglądarkę i umożliwia efektywne przeszukiwanie i filtrowanie elementów DOM.

- Łatwa manipulacja elementami DOM – element można dodać lub usunąć za pomocą pojedynczych instrukcji.
- Obsługa zaawansowany efektów – tj. animacja większości atrybutów CSS tj. rozmiar czy kolor.
- Obsługa zdarzeń – jest identyczna dla każdej przeglądarki i cechuje się dużą prostotą używania.
- Uproszczona obsługa AJAX – dzięki której można wysłać dynamiczne zapytanie do serwera HTTP za pomocą paru linii kodu.
- Zbiór użytecznych narzędzi – np. funkcja `each`
- Obsługa wszystkich popularnych przeglądarek – pozwalająca nie zwracać uwagi na różnice występujące między każdą z nich.

jQuery dostarczane jest w postaci pojedynczego pliku JavaScript, w którym zawarta jest cała funkcjonalność. jQuery można wywoływać na dwa sposoby: za pomocą funkcji `$`, która jest fabryką obiektów i za pomocą zbioru funkcji poprzedzonych prefiksem `$.` , które udostępniają dodatkową, użyteczną funkcjonalność. Funkcja fabryki obiektów jako parametr przyjmuje selektor CSS, a jej rezultatem są obiekty, które pasują do tego selektora. Na tak zwróconym obiekcie można łańcuchowo wywoływać kolejne operacje, np.:

```
$("div.main").add("p ").addClass("yellow").slideUp("fast");
```

Metody z prefiksem `$.` udostępniają dodatkową użyteczną funkcjonalność, np. w poniższym przykładzie metoda `each` spowoduje wypisanie po kolei numerów 3, 5, 7.

```
$.each([3,5,7], function()
{
  document.write(this);
});
```

Pierwsza wersja jQuery została wydana w 2006 roku. W niniejszej pracy wykorzystywana jest najnowsza wersja zaprezentowana w roku 2010 – jQuery 1.4.

3.8 NUnit

NUnit jest frameworkiem do przeprowadzania testów jednostkowych we wszystkich językach platformy .NET. Pierwotnie był to port znanego z języka Java JUnit, jednakże obecna wersja, w całości napisana w C#, została kompletnie przebudowana i wykorzystuje najlepsze cechy platformy .NET.

Pierwsza wersja NUnit była aplikacją konsolową - interfejs graficzny został dodany dopiero w wersji numer 2.0. Wersja 3.0 ma być bardziej kompleksową

platformą, dodatkowo umożliwiającą testowanie integracyjne, akceptacyjnych oraz wydajnościowe.

Ponieważ języki z platformy .NET nie dają możliwości wielodziedziczenia, NUnit do testowania wykorzystuje mechanizm atrybutów. Za ich pomocą dodaje się do klasy dodatkowe meta dane, które informują o tym, czy dana metoda jest np. testem, czy służy do przygotowania środowiska. Pakiet NUnit składa się z dwóch komponentów: biblioteki NUnit.Framework.dll, którą dołączamy do projektów z testami jednostkowymi oraz aplikacji uruchamiającej testy.

NUnit jest obecnie najczęściej używanym frameworkiem testowym dla platformy .NET. W połączeniu z metodyką TDD daje programistom możliwość pisania stabilniejszego i działającego zgodnie z oczekiwanie oprogramowania.

4 Propozycja nowego rozwiązania

Wytworzona aplikacja będzie umożliwiała użytkownikom tworzenie spersonalizowanych map internetowych. Rozwiązanie będzie charakteryzowało się łatwością użytkowania tak, aby nawet niedoświadczeni użytkownicy komputerów nie mieli problemów z jego obsługą. Aplikacja będzie wykonana w postaci strony internetowej, która nie wymaga od użytkowników instalowania dodatkowego oprogramowania.

System do tworzenia spersonalizowanych map będzie się składał z dwóch głównych modułów: modułu do administracji mapą oraz modułu do jej przeglądania.

4.1 Panel administracyjny

Panel administracyjny jest miejscem, z którego będzie odbywało się zarządzanie wszystkimi ustawieniami aplikacji oraz miejscami wyświetlanymi na mapie. Dostęp do panelu będzie możliwy po wcześniejszej rejestracji i zalogowaniu się. Panel administracyjny będzie się składał z następujących modułów:

- Ustawienia ogólne.
- Domyślna lokalizacja mapy.
- Struktura danych.
- Formularz dodawania miejsc.
- Zarządzanie miejscami.

4.1.1 Ustawienia ogólne

W module ustawień ogólnych użytkownik będzie miał możliwość konfiguracji podstawowych parametrów swojej mapy. Każda nowo utworzona mapa ma ustawioną domyślną nazwę wyświetlaną, której zmiana będzie możliwa za pomocą tej części panelu.

4.1.2 Domyślna lokalizacja mapy

Każda nowo stworzona mapa posiada domyślną lokalizację. Dla tej lokalizacji ustawiony jest również pewien poziom zbliżenia. Za pomocą tej części panelu użytkownik będzie miał możliwość zmiany zarówno lokalizacji, jak i poziomu zbliżenia. Będzie dostępny podgląd interaktywnej mapy, na którym za pomocą myszki można będzie ustawić wyżej wymienione parametry. Interfejs będzie się cechował

wysokim poziomem interaktywności, osiągniętym m.in. poprzez wykorzystywanie techniki drag-and-drop.

4.1.3 Struktura danych

Użytkownik będzie miał możliwość określenia jaki format mają informacje opisujące dodawane przez niego miejsca. Definiowanie formatu będzie odbywało się za pomocą edytora wizualnego i przypominało budowę formularz do wprowadzania danych. Dostępne będą następujących typy pól:

- Małe pole tekstowe.
- Duże pole tekstowe typu RichText.
- Pole numeryczne.
- Pole typu data.
- Pole wyboru wraz z możliwością określeniem dostępnych opcji.
- Pole typu URL.

Każde pole będzie miało możliwość edycji etykiety opisującej jego zawartość. Kolejność pól będzie można zmienić poprzez ich przeciąganie i upuszczanie za pomocą myszki. Pierwsze pole zawsze będzie małym polem tekstowym i nie będzie możliwości zmiany jego kolejności. Podyktowane jest to tym, że musi istnieć jakaś wartość tekstowa tj. tytuł lub nazwa za pomocą, której będzie można w skrócie opisać miejsce. Modyfikacja struktury danych będzie możliwa, nawet w sytuacji gdy już będą dodane miejsca na mapie.

4.1.4 Formularz dodawania miejsc

Formularz dodawania miejsc będzie umożliwiał użytkownikowi dodanie nowych miejsc do jego mapy. Formularz ten będzie składał się z trzech części:

- Interaktywnej mapki – na której użytkownik będzie mógł klikając zaznaczyć lokalizację dodawanego miejsca.
- Formularza do wprowadzania danych – wygenerowanego na podstawie zdefiniowanego w module „Struktura danych” formatu.
- Modułu do wgrywania zdjęć.

Dane wprowadzone w formularzu będą walidowane, w taki sposób aby były poprawne w zależności od typu pola. Jedynym obowiązkowym polem jest pierwsze pole będące tytułem, nazwą lub innym elementem tekstowym w skrócie opisującym dane miejsce. Jest to konieczne, ponieważ musi istnieć jakaś wartość, która będzie

wyświetlana np. na liście miejsc. Pozostałe pola muszą być opcjonalne, ponieważ istnieje możliwość edycji struktury danych.

Moduł wgrywania zdjęć służy do przypisywania ich do wprowadzanego miejsca. Będzie udostępniał możliwość wgrywania zdjęć bez konieczności przeładowywania strony w przeglądarce. Wgrane zdjęcia będą mogły być usuwane oraz sortowane za pomocą techniki drag-and-drop.

4.1.5 Zarządzanie miejscami

W panelu administracyjnym będzie dostępna lista wszystkich dodanych miejsc. Lista ta dla większej ilości elementów będzie podzielona na podstrony. Użytkownik będzie miał możliwość prostego filtrowania elementów za pomocą słów kluczowych oraz przedziału daty. Lista będzie udostępniała możliwość usuwania poszczególnych miejsc oraz przejścia do formularza edycji. Formularz edycji miejsc będzie tożsamy z formularzem dodawania.

4.2 Spersonalizowana mapa

Spersonalizowana mapa będzie miejscem, gdzie będą wyświetlane wszystkie miejsca dodane przez użytkownika w panelu administracyjnym. Będzie się ona składała z trzech głównych części:

- Filtrów.
- Listy miejsc.
- Mapy miejsc.

4.2.1 Filtry

Za pomocą filtrów będą ograniczane miejsca wyświetlane zarówno na liście, jak i mapie. Wyświetlane będą tylko te spełniające wszystkie podane kryteria. Filtry będą generowane na podstawie zdefiniowanego w panelu administracyjnym formatu danych. Dla poszczególnych typów pól będą zdefiniowane następujące filtry:

- Dla pól tekstowych – jeden wspólny filtr ograniczający wyniki do tych zawierających podane słowa kluczowe.
- Dla pól numerycznych – filtr od-do.
- Dla pól typu data – filtr od-do.
- Dla pól wyboru – filtr z listą dostępnych opcji.

Oprócz standardowych filtrów będzie zdefiniowany globalny filtr pozwalający wyświetlać jedynie miejsca znajdujące się w pewnym promieniu od podanej lokalizacji. Do podania lokalizacji będzie służyło pole tekstowe wykorzystujące Google API z opcją automatycznego uzupełniania nazwy. Do obliczania odległości między dwoma punktami na mapie zostanie wykorzystany specjalny algorytm (patrz rozdział 4.3.1).

W momencie dodawania lub usuwania wartości filtrowania, zarówno lista miejsc, jak i mapa będą ulegały automatycznej aktualizacji, tak aby spełniać nowe kryteria.

4.2.2 Lista miejsc

Obok mapy będzie wyświetlana lista miejsc dodanych do mapy. Przy większej ilości elementów lista będzie stronicowana. Wyświetlane będą jedynie elementy, które spełniają wszystkie kryteria zdefiniowane w filtrach. Po kliknięciu na wybrane miejsce, mapa ulegnie wyśrodkowaniu do współrzędnych geograficznych opisujących je.

4.2.3 Mapa miejsc

Miejsca na mapie będą wyświetlane w postaci punktów. Dla zwiększenia wydajności wyświetlane będą jedynie miejsca, które się znajdują w polu widzenia oraz spełniają wszystkie kryteria zdefiniowane za pomocą filtrów.

Punkty na mapie nie powinny nachodzić na siebie. Punkty, które znajdują się zbyt blisko siebie będą ulegały grupowaniu za pomocą specjalnego algorytmu (patrz rozdział 4.3.3). Każda grupa będzie miała widoczną etykietę z liczbą informującą o tym, ile znajduje się w niej punktów.

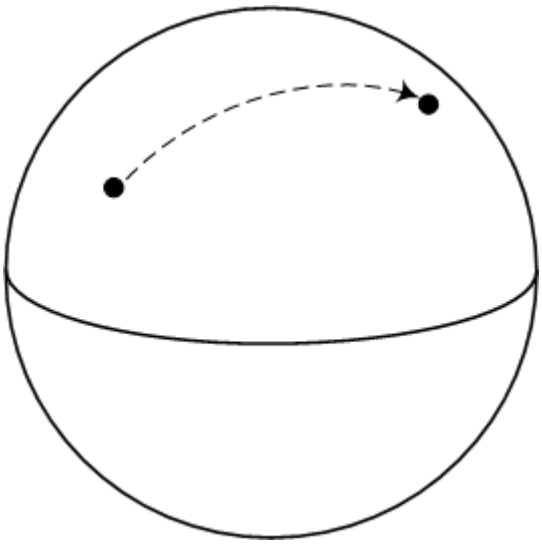
Po kliknięciu na pojedynczy punkt na mapie ukaże się podsumowanie danego miejsca, zawierające zdjęcia oraz szczegółowe informacje.

4.3 Algorytmy i wzory

Do realizacji wyżej wymienionej funkcjonalności konieczne jest zastosowanie wzorów na obliczanie odległości między dwoma współrzędnymi geograficznymi, odległości między dwoma punktami na mapie oraz algorytmu grupowania wyświetlanych obiektów.

4.3.1 Obliczanie odległości miedzy dwoma współrzędnymi geograficznymi

Ponieważ Ziemia swoim kształtem przypomina kulę, liczenie odległości między dwoma punktami znajdującymi się na jej powierzchni nie jest zadaniem banalnym (patrz rys. 9).



Rysunek 9 - Najkrótsza ścieżka między dwoma punktami na powierzchni kuli.

Niech:

- $\phi_s, \lambda_s;\ \phi_f, \lambda_f$ - będzie współrzędnymi geograficznymi dwóch punktów.
- $\Delta\phi, \Delta\lambda$ - będzie różnicą długości i szerokości geograficznej obu punktów.
- $\Delta\widehat{\sigma}$ - będzie różnicą kątową obu punktów.

Korzystając z geometrii sferycznej, możemy otrzymać wzór:

$$\Delta\widehat{\sigma} = \arccos\left(\sin\phi_s \sin\phi_f + \cos\phi_s \cos\phi_f \cos\Delta\lambda\right).$$

(1)

Mając kulę o promieniu *r* i $\Delta\widehat{\sigma}$ wyrażone w radianach, do obliczenia długość łuku będącego odległością między dwoma punktami można zastosować wzór:

$$d = r\,\Delta\widehat{\sigma}.$$

(2)

Powyżej przedstawiona metoda ma jednak poważna wadę. Przy małych odległościach w wyniku błędów zaokrąglania, rezultat potrafi być mocno przekłamany. Dużo częściej wykorzystywaną metodą, charakteryzującą się lepszą dokładnością jest formuła, która wykorzystuje tzw. twierdzenie haversinów:

$$d = R\,\mathrm{haversin}^{-1}(h) = 2R\arcsin\left(\sqrt{h}\right) =$$
$$2R\arcsin\left(\sqrt{\sin^2\left(\frac{\Delta\phi}{2}\right) + \cos\phi_s\cos\phi_f\sin^2\left(\frac{\Delta\lambda}{2}\right)}\right).$$

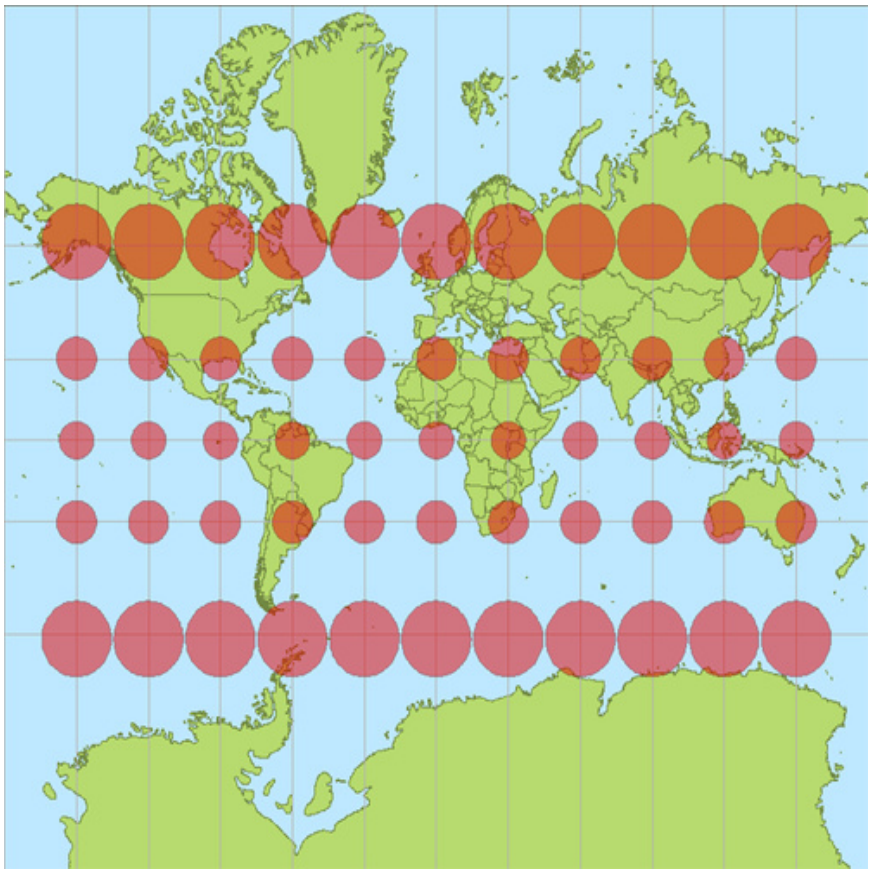
(3)

4.3.2 Obliczanie odległości na mapie między dwoma punktami.

Wszystkie rodzaje map internetowych, które były omawiane w niniejszej pracy, do odzwierciedlenia powierzchni Ziemi na płaszczyznę używają odwzorowania walcowego wielokątnego, znanego jako odwzorowanie Merkatora (patrz rys. 10).

Rysunek 10 - Ziemia w odwzorowaniu Mercatora [8]

Odwzorowanie to jest dokładne na równiku, ale wraz z oddalaniem się od niego w stronę biegunów rośnie deformacja (patrz rys. 11). Jest to jedno z najstarszych znanych odwzorowań, wynalezione w XVI w. było głównie używane w czasach wielkich odkryć geograficznych.



Rysunek 11 - Zniekształcenia w odwzorowaniu Mercatora

Mając długość (φ) i szerokość (λ) geograficzną, aby otrzymać współrzędne x i y punktu znajdującego się na mapie wykorzystującej odwzorowanie Merkatora (gdzie λ_0 to szerokość geograficzna środka) należy zastosować równanie:

$$
\begin{aligned}
x &= \lambda - \lambda_0 \\
y &= \ln\left(\tan\left(\frac{\pi}{4} + \frac{\varphi}{2}\right)\right) \\
&= \frac{1}{2}\ln\left(\frac{1+\sin(\varphi)}{1-\sin(\varphi)}\right) \\
&= \sinh^{-1}\left(\tan(\varphi)\right) \\
&= \tanh^{-1}\left(\sin(\varphi)\right) \\
&= \ln\left(\tan(\varphi) + \sec(\varphi)\right).
\end{aligned}
$$

(4)

Następnie mając dwa punktu na płaszczyźnie kartezjańskiej i wykorzystując twierdzenie Pitagorasa można obliczyć długość odcinka, który je łączy. Otrzymana długość będzie odległością na mapie między dwoma punktami.

4.3.3 Grupowanie miejsc na mapie

Wyświetlane na mapie miejsca nie powinny nachodzić na siebie. W momencie, gdy znajdują się zbyt blisko powinny być łączone w grupy. W związku z tym, że dystans między punktami jest postrzegany inaczej na każdym poziomie zbliżenia, grupowanie powinno się odbywać dla każdego z tych poziomów z osobna. Możliwymi rozwiązaniami są:

- Grupowanie oparte na wirtualnej siatce.
- Grupowanie oparte na podstawie dystansu między punktami.

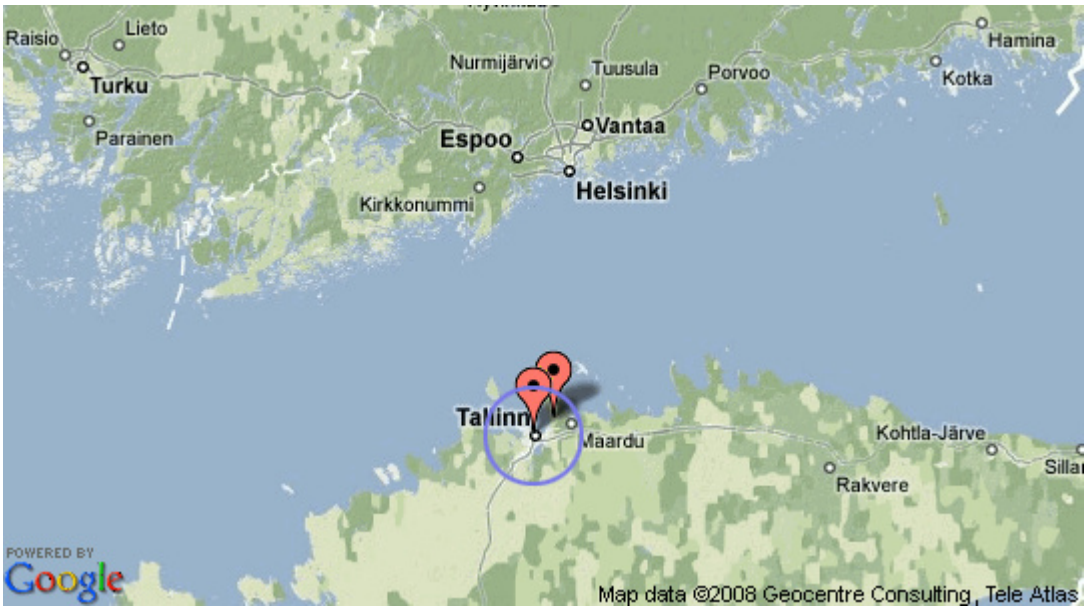
Grupowanie oparte na wirtualnej siatce dzieli mapę na kwadratowe regiony. Niezależnie od poziomu zbliżenia, rozmiar kwadratów w pikselach jest stały, natomiast wraz ze wzrostem zbliżenia wyznaczany przez nie rzeczywisty obszar maleje. Ta technika ma jednak swoje ograniczenia (patrz rys. 12).

Rysunek 12 - Grupowanie oparte na wirtualnej siatce [9]

Dwa punkty są tak blisko, że nachodzą na siebie. Jednocześnie nie są zgrupowane, ponieważ należą do innych regionów wyznaczonych przez wirtualną siatkę.

Drugim podejściem jest grupowanie oparte na dystansie między punktami. Można wykorzystać algorytm obliczania dystansu między dwiema współrzędnymi geograficznymi (patrz rozdział 4.3.1) i grupować obiekty znajdujące się w promieniu pewnej ilości kilometrów od siebie. Rozwiązanie to posiada jednak jedną wadę. Odległość w kilometrach jest inaczej postrzegana w zależności od zbliżenia. Na jednym poziomie zbliżenia kilometr może być równoważny 10 a na innym 100 pikselom.

Rozwiązaniem tego problemu jest grupowanie miejsc z wykorzystaniem algorytmu obliczania odległości na mapie między dwoma punktami (patrz rozdział 4.3.2) wyrażanej w pikselach. Jeden piksel zawsze będzie jednym pikselem niezależnie od poziomu zbliżenia. Dla przykładu chcemy zgrupować miejsca znajdujące się w odległości 20 pikseli od siebie (patrz rys. 13).



Rysunek 13 - Grupowanie oparte na podstawie odległości między punktami. [9]

W tym przypadku oba punkty zostaną zgrupowane, ponieważ znajdują się w promieniu 20 pikseli.

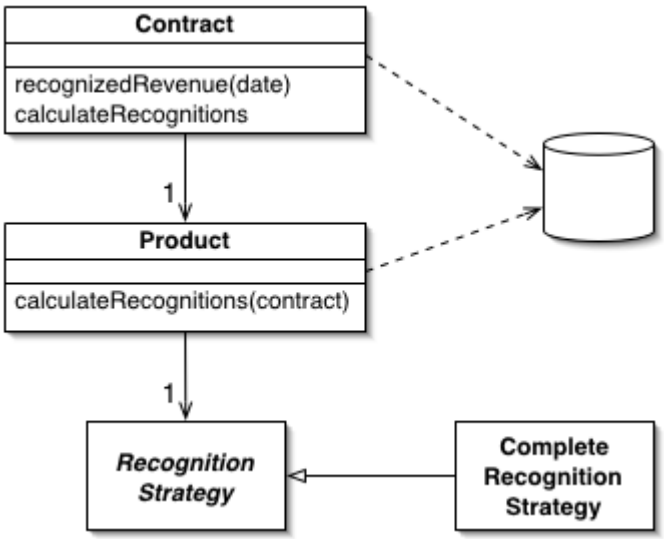
5 Prototyp

W niniejszej części przedstawiony został prototyp do tworzenia spersonalizowanych map internetowych. Powstał on z wykorzystaniem wymienionych wcześniej narzędzi i technologii (patrz rozdział 0) oraz interaktywnych map Google Maps. Na początku omówione zostaną wykorzystane wzorce projektowe, następnie ogólna architektura oraz model domenowy. Na końcu przedstawiony zostanie zaimplementowany prototyp wraz z przykładem zastosowania.

5.1 Wykorzystane wzorce projektowe

5.1.1 Domain Model

Martin Fowler w swojej książce [10] określa Domain Model (model domenowy) jako sieć połączonych ze sobą obiektów, które mają odzwierciedlenie w realnym obszarze biznesowym, zawierających dane i reguły biznesowe. Model domenowy jest tworzony, aby jak najlepiej opisać elementy budowanego systemu. Model taki oprócz definiowania atrybutów i akcji, jakie można wykonać na każdym obiekcie określa, w jaki sposób poszczególne obiekty są ze sobą połączone. Za pomocą takiego sposobu opisu projektowanego systemu, można w bardzo łatwy sposób zweryfikować, czy dziedzina problemowa została odpowiednio zrozumiana. Dzięki temu, że model domenowy jest stosunkowo łatwy do zrozumienia, jest przydatny w komunikacji między zespołami technicznymi a biznesowymi. Rys. 14 przedstawia przykładowy model domenowy.

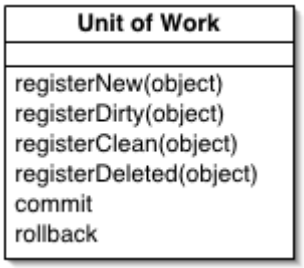


Rysunek 14 - Przykładowy model domenowy

5.1.2 Unit of Work

Według Fowlera wzorzec Unit of Work [10] służy do zarządzania listą obiektów biznesowych i koordynuje zapisywaniem zmian w nich wprowadzonych oraz rozwiązuje problemy synchronizacyjne z tym związane.

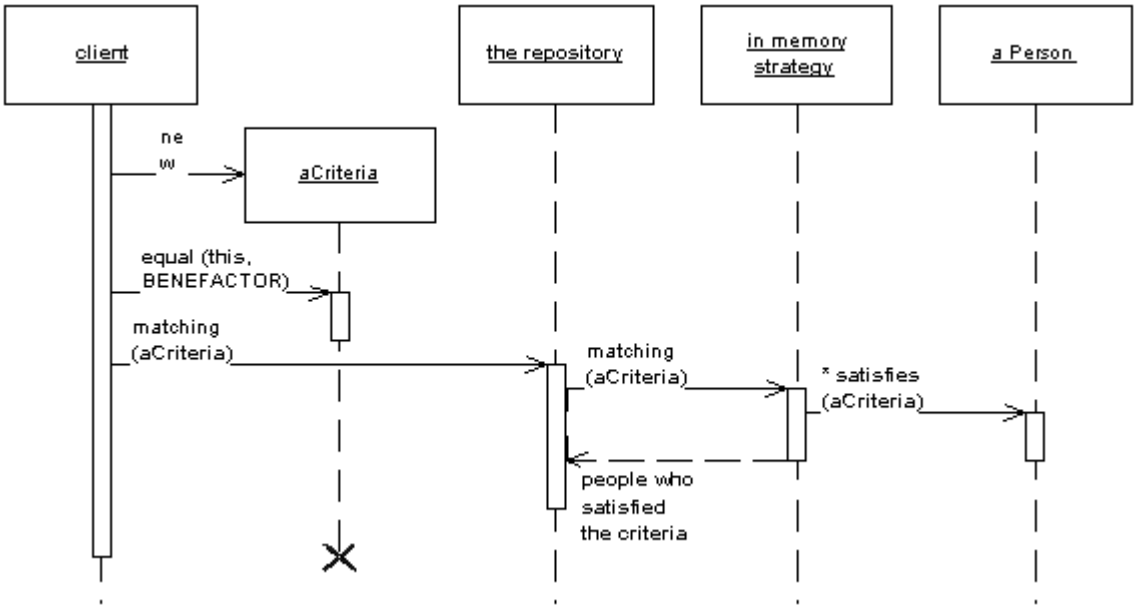
Kiedy obiekty są pobierane z bazy danych konieczne jest śledzenie zmian w nich zachodzących tak, aby zostały one później utrwalone. Podobnie, gdy obiekty są tworzone lub usuwane trzeba pamiętać, aby wykonać operację na bazie. Przy każdej zmianie można wykonywać odwołanie do bazy danych, lecz takie rozwiązanie jest niepraktyczne i mało wydajne. Wzorzec Unit of Work rozwiązuje ten problem poprzez śledzenie wszystkich zmian za nas. W momencie kiedy wszystkie operacje zostaną ukończone, Unit of Work zajmuje się utrwaleniem aktualnego stanu w bazie danych. Rys. 15 przedstawia klasę Unit of Work.



Rysunek 15 - Klasa Unit of Work

5.1.3 Repository

Martin Fowler opisuje wzorzec Repository [10] jako mediator pomiędzy obiektami modelu domenowego a warstwą dostępu do bazy danych. Mediator ten umożliwia dostęp do obiektów modelu domenowego za pomocą interfejsu przypominającego kolekcję. Repository udostępnia metody do dodawania, usuwania oraz wyszukiwania obiektów. Wyszukiwanie może być zrealizowane za pomocą wielu konkretnych metod lub poprzez generyczną metodę przyjmującą specyfikację jako parametr. Rys. 16 przedstawia diagram sekwencji obrazujący wyszukiwanie realizowane za pomocą Repository.



Rysunek 16 - Wyszukiwanie za pomocą Repository

5.1.4 Command-query Seperation

Command-query Seperation [11] jest paradygmatem zaprezentowanym przez Bertranda Meyera. Meyer twierdzi, że metody obiektu powinny być podzielone na dwie kategorie:

- Komendy – które zmieniają stan obiektu.
- Zapytania – które zwracają wartość lecz nie zmieniają stanu obiektu.

Paradygmat ten zakłada, że obiekt jest albo odpytywany albo wysyłana jest do niego komenda – nigdy obie czynności nie powinny zachodzić naraz. Command-query Seperation nie koniecznie musi być stosowany na poziomie obiektu. Równie skutecznie sprawdza się na poziomie moduły czy całego systemu.

5.1.5 Dependency Injection

Dependency Injection [10] jest wzorcem projektowym, który promuje luźne powiązanie komponentów. W klasycznym podejściu komponent A wykorzystujący funkcjonalność udostępnianą przez komponent B, odpowiada za stworzenie i skonfigurowanie instancji B. W wyniku tego oba obiektu są ściśle ze sobą powiązane.

Wzorzec Dependency Injection stara się rozwiązać ten problem poprzez odciążenie komponentu A z obowiązku tworzenia i konfigurowania komponentu B. Zamiast tego obiekt B jest inicjowany zewnętrznie (najczęściej przez specjalny kontener) i następnie przekazywany za pomocą konstruktora lub atrybutu do komponentu A. Dzięki temu, że obiekty są tworzone i konfigurowane przez zewnętrzny

byt istnieje lepsza możliwość zarządzania ich cyklem życiowym. Stosując wzorzec Dependency Injection, najluźniejsze powiązanie między komponentami osiąga się poprzez definiowanie zależności między komponentami jako interfejsy a nie jako klasy.

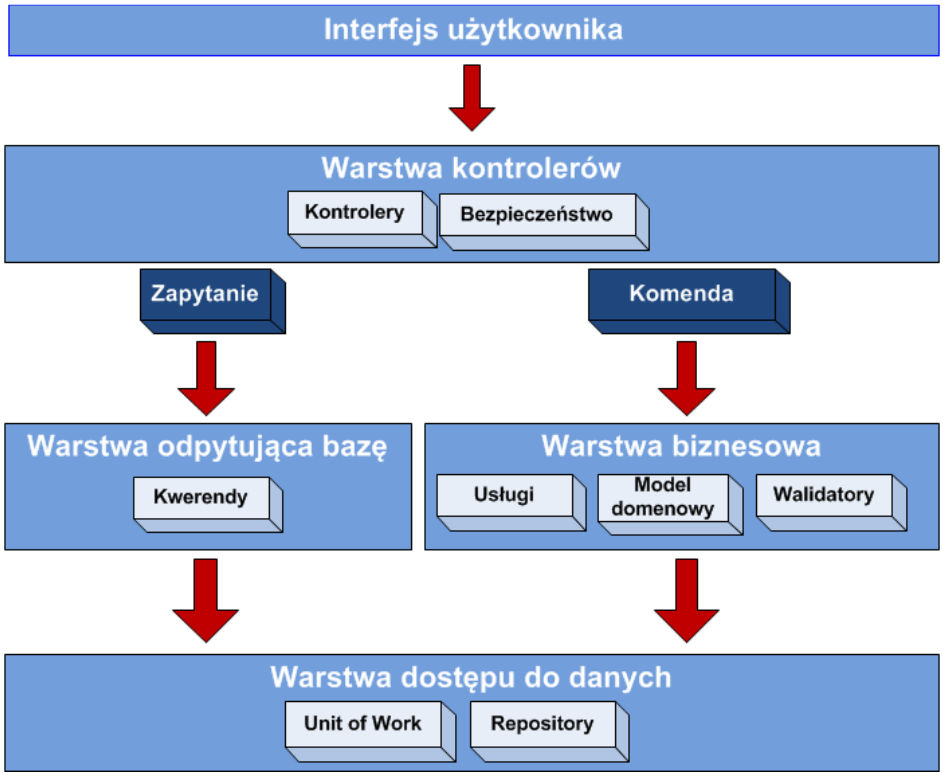
5.2 Implementacja

Przy implementacji aplikacji zostały wykorzystane ogólnie znane wzorce projektowe (przedstawione w rozdziale 5.1) oraz dobre praktyki programistyczne. Konieczna była również implementacja wzorów obliczających odległość na mapie oraz powierzchni kuli ziemskiej oraz algorytmu do grupowania miejsc znajdujących się na mapie (patrz rozdział 4.3). W niniejszym rozdziale autor skupił się na przedstawieniu ogólnej architektury aplikacji oraz opisie implementacji modelu pozwalającego zapisać oraz przeszukiwać dane o zdefiniowanej przez użytkownika strukturze.

5.2.1 Opis architektury

Wytworzony w ramach niniejszej pracy prototyp jest aplikacją webową, składającą się z interfejsu graficznego utworzonego z wykorzystaniem technologii HTML, CSS i AJAX oraz backendu wykorzystującego framework ASP.NET MVC.

Na rys. 17 pokazana została ogólna architektura aplikacji z podziałem na warstwy.



Rysunek 17 - Ogólna architektura aplikacji

W większości przypadków interfejs graficzny komunikuje się z serwerem za pomocą asynchronicznych zapytań http. Cała asynchroniczna komunikacja odbywa się z wykorzystaniem formatu JSON.

Część serwerowa jest wielowarstwowa. Tak jak każdy webowy framework MVC, ASP.NET MVC obsługuje zapytania http w kontrolerach. W wytworzonym prototypie kontrolery tworzą pierwszą warstwę aplikacji. Akcje w kontrolerach zaimplementowane są z wykorzystaniem wzorca CQS (patrz rozdział 5.1.4). Oznacza to, że zapytania przychodzące do aplikacji są podzielone na dwie kategorie: zapytania modyfikujące i dodające dane oraz zapytania odpytujące. W zależności od rodzaju akcji sterowanie przekazywane jest dalej do jednej z dwóch warstw: warstwy biznesowej lub warstwy odpytującej bazę danych.

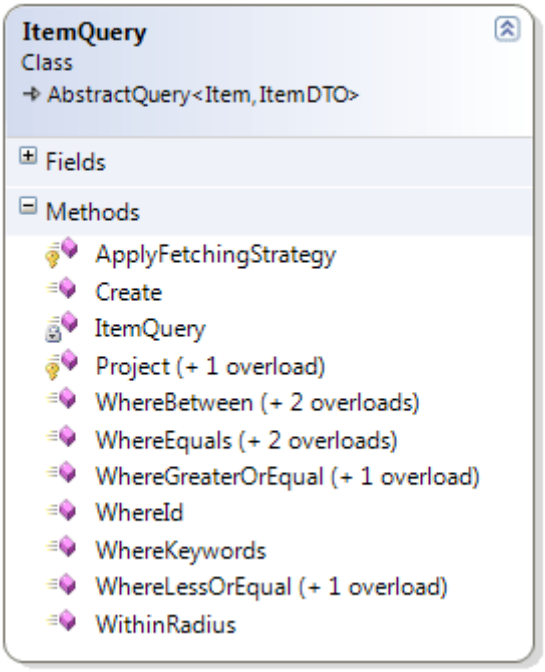
Warstwa odpytująca bazę danych składa się z klas implementujących kwerendy. Wszystkie klasy kwerend implementują specjalny interfejs IQuery (patrz przykład 6). Interfejs ten udostępnia m.in. metody do pobrania wszystkich wyników, pojedynczego wyniku, pojedynczej strony oraz do zliczenia liczby wyników i stron. Dodatkowo bazową implementację IQuery można znaleźć w klasie AbstractQuery.

Przykład 6 - Interfejs IQuery

```
public interface IQuery<T, TResultT> where T : EntityBase
{
    IQuery<T, TResultT> WithItemsPerPage(int itemsPerPage);

    IList<TResultT> All();
    IList<TResultT> Page(int page);
    TResultT First();
    int PageCount();
    int Count();
    bool Any();
}
```

Wszystkie klasy kwerend są napisane z wykorzystaniem tzw. Fluent Interface [10]. Jest to styl tworzenia interfejsów, w taki sposób aby umożliwiały wywoływanie metod w sposób ciągły, a ciąg wywołanych metod nierzadko sprawiał wrażenie napisanego w języku naturalnym.



Rysunek 18 - Klasa kwerendy napisana w stylu fluent

W przykładzie 7 pokazane zostało wywołanie zaimplementowanej w aplikacji klasy ItemQuery (patrz rys. 18), która została napisana w stylu fluent.

Przykład 7 - Wywołanie w stylu fluent

```
ItemQuery.Create()
    .WithinRadius(new Coordinates(19.0, 17.2), 50)
    .WhereLessOrEqual("Price", 20000)
    .WhereKeywords("Honda")
    .All();
```

Metody wewnątrz klas opakowujących kwerendy do odpytywania bazy danych używają biblioteki NHibernate LINQ, która umożliwia pisanie zapytań w języku LINQ oraz za pomocą wyrażeń lambda (patrz przykład 8).

Przykład 8 - Odpytywanie bazy za pomocą wyrażeń lambda

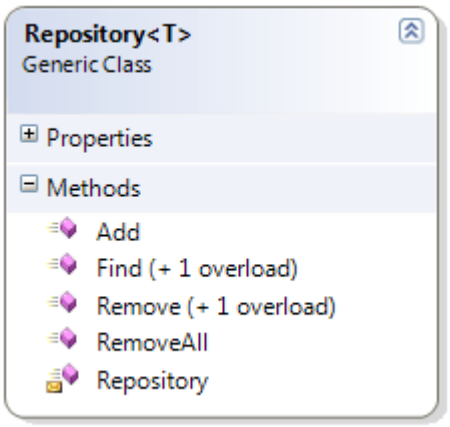
```
     public ItemQuery WhereBetween(string key, double fromValue,
double toValue)
    {
        if (string.IsNullOrEmpty(key))
            throw new ArgumentException("`key` is null or empty");

        Items = Items.Where(
            x => x.NumberValues.Any(
                y => y.StructureElement.Key == key &&
                    y.Value >= fromValue &&
                    y.Value <= toValue));

        return this;
    }
```

Warstwa biznesowa jest jedną z najważniejszych części aplikacji. Jej podstawowym składnikiem jest bogaty model biznesowy, który implementuje jak największą funkcjonalność. Funkcjonalność nie mieszcząca się w modelu biznesowym została umieszczona w klasach usług. Dodatkowo zaimplementowany został zestaw walidatorów, służących do weryfikacji danych wejściowych komend wysyłanych do systemu. Warstwa ta również posiada zaimplementowane wszystkie algorytmy omawiane w rozdziale 4.3. Do wczytywania i utrwalania stanu modelu wykorzystywana jest funkcjonalność udostępniana przez warstwę dostępu do bazy danych.

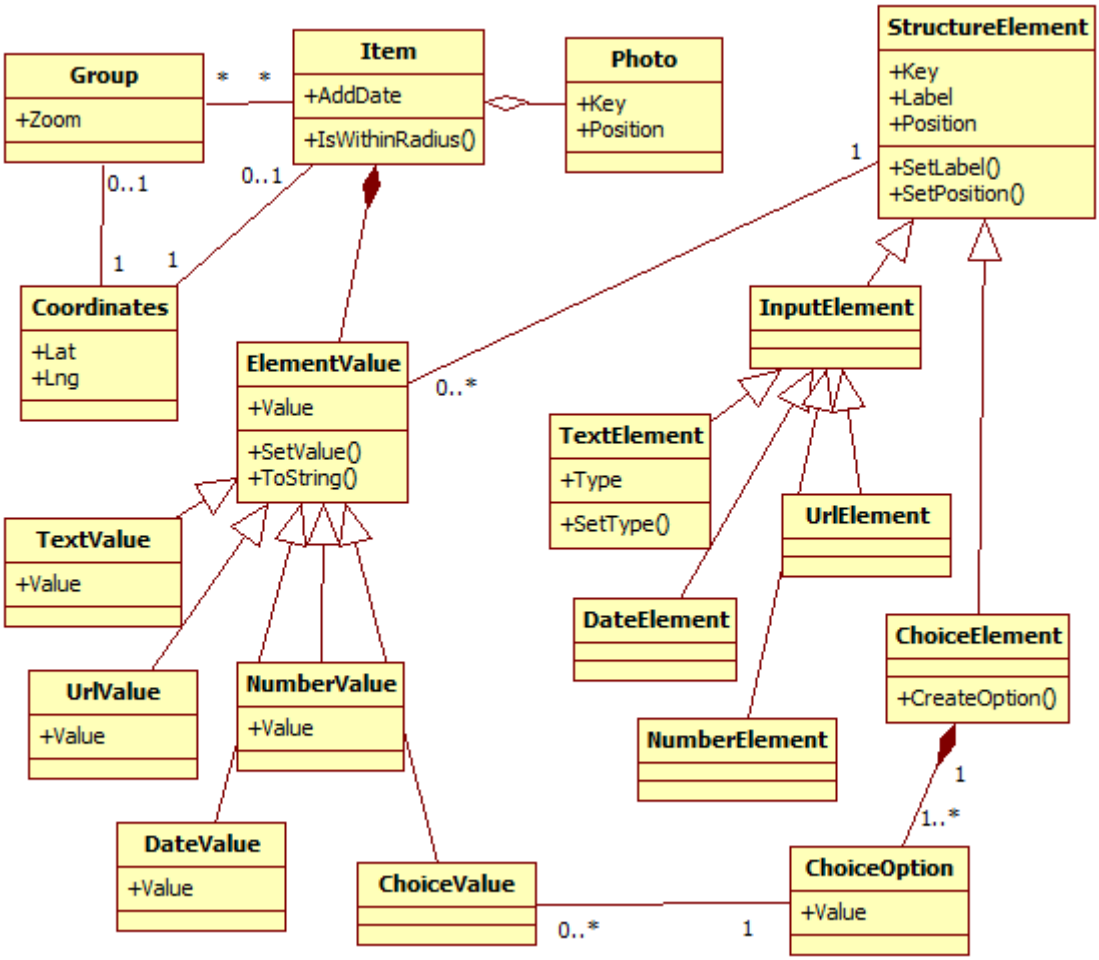
Warstwa dostępu do danych implementuje omówione wcześniej wzorce Unit of Work i Repository (patrz rozdział 5.1.2 i 5.1.3). Dostęp do bazy danych jest zrealizowany za pomocą biblioteki NHibernate. Klasa Repository (patrz rys. 19) jest typu generycznego i udostępnia metody do dodawania, wyszukiwania i usuwania obiektów. Funkcja wyszukująca jako parametr przyjmuje specyfikację w postaci wyrażenia lambda, a zwraca interfejs IQueryable, dzięki czemu istnieje możliwość dalszego zaawansowanego przeszukiwania. Repository jest jedynym udostępnianym innym warstwom sposobem dostępu do bazy danych.



Rysunek 19 - Klasa Repository

5.2.2 Model domenowy

Model domenowy jest sercem aplikacji. Służy do przechowywania wszystkich danych przez nią przetwarzanych oraz umożliwia wykonanie na nich różnych operacji. Na rys. 20 przedstawiony został model zaimplementowanego prototypu.



Rysunek 20 - Model domenowy

Stworzony model domenowy spełnia założone wymagania i umożliwia przechowywanie miejsc wraz ze sprecyzowanym przez użytkownika złożonym rodzajem informacji. Zaproponowana struktura jest na tyle elastyczna, że można ją w przyszłości rozbudować o dodatkowe typy.

Dzięki zastosowaniu tak generycznego podejścia istnieje możliwość zbudowania odpowiedniego zapytania i filtrowania wyszukiwanych miejsc już w bazie danych a nie pamięci operacyjnej. Daje to możliwość tworzenia map z bardzo dużą ilością punktów bez obaw o wydajność. Dodatkowo dzięki wykorzystaniu biblioteki NHibernate LINQ istnieje możliwość pisania mocno typowanych sprawdzanych w czasie kompilacji kwerend w języku LINQ.

5.3 Przykład zastosowania

W niniejszym rozdziale na przykładzie tworzenia i obsługi mapy zostanie zaprezentowana funkcjonalność zaimplementowanej aplikacja do tworzenia spersonalizowanych map internetowych.

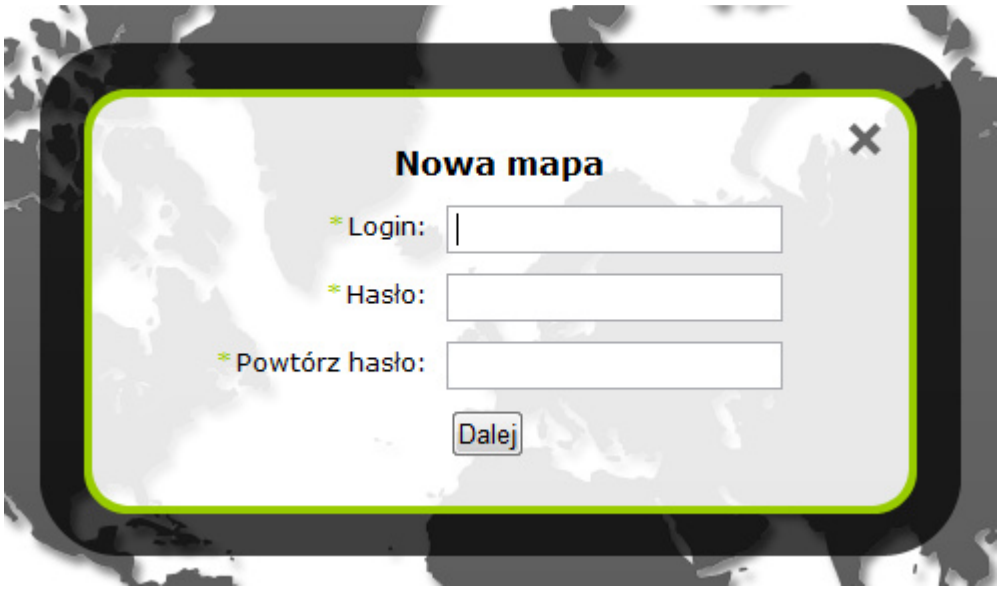
Za przykład posłuży wyimaginowana agencja nieruchomości, która chce zrobić mapę aktualnie sprzedawanych przez siebie domów i mieszkań oraz dać użytkownikom zaawansowane możliwości ich wyszukiwania. Wszystkie wprowadzane dane zostały wymyślone przez autora pracy, a użyte zdjęcia pochodzą z darmowych zasobów strony http://www.sxc.hu.

5.3.1 Tworzenie nowej mapy

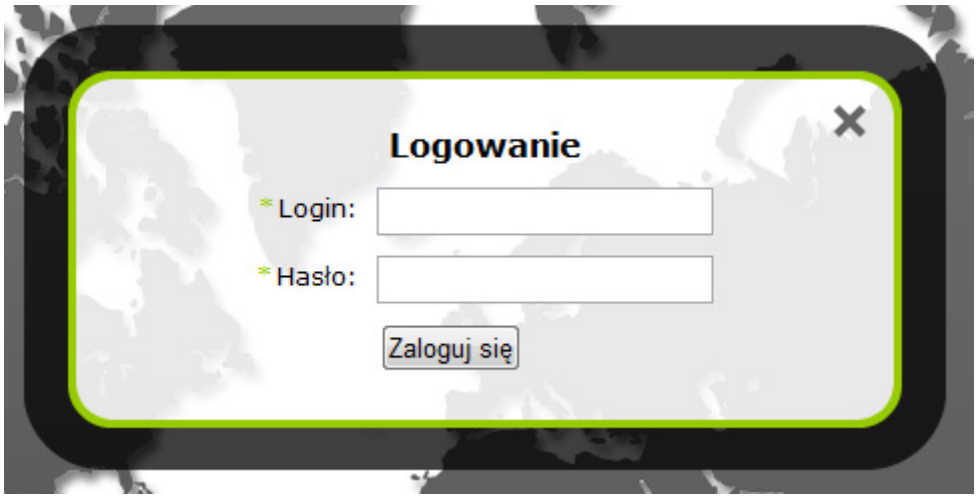
Po uruchomieniu aplikacji w przeglądarce internetowej pojawia się pierwsza strona umożliwiająca utworzenie nowej mapy oraz zalogowanie do już istniejącej (patrz rys. 21).

Rysunek 21 - Strona główna

Po wybraniu opcji, pojawia się jeden z dwóch formularzy: formularz tworzenia nowej mapy (patrz rys. 22) lub formularz logowania (patrz rys. 23).



Rysunek 22 - Formularz tworzenia nowej mapy



Rysunek 23 - Formularz logowania

Ponieważ cała aplikacja została zbudowana z naciskiem na dużą interaktywność, wszystkie operacje wykonywane na stronie głównej nie wymagają jej przeładowywania oraz są realizowane z intensywnym wykorzystaniem animacji.

Wszystkie formularze w aplikacji wykorzystują ten sam styl graficzny i obsługiwane są asynchronicznie. W momencie zatwierdzenia formularza pojawia się animacja informująca o tym, że należy czekać, po czym w zależności od okoliczności wykonywana jest jedna z następujących czynności:

- Pojawia się informacja o błędach (patrz rys. 24), jeśli jakieś wystąpiły. Informacja ta jest wysuwana oraz animowany jest kolor tworząc efekt błysku, który ma na celu przykuć uwagę użytkownika. Dodatkowo pola

zwierające błędy są oznaczone za pomocą grubego kolorowego obramowania.

- Pojawia się komunikat informujący o sukcesie zakończonej operacji.
- Następuje przekierowanie do innej strony.

Rysunek 24 – Informacja o błędach w formularzu

Dla celów niniejszego przykładu została stworzona mapa z loginem *nieruchomosci*. Po zatwierdzeniu prawidłowo wypełnionego formularza rejestracji nastąpiło przekierowanie do panelu administracyjnego.

5.3.2 Panel administracyjny

Panel administracyjny jest miejscem, które umożliwia zarządzanie wszystkimi ustawieniami aplikacji, wprowadzanie, edytowanie i usuwanie miejsc na mapie oraz definiowanie formatu danych do nich przypisanych.

Na rys. 25 przestawiony został główny widok panelu administracyjnego. Z lewej strony znajduje się menu, które jest stałe dla wszystkich podstron, natomiast z prawej strony jest zawartość aktualnej podstrony.

Z poziomu menu można przejść do:

- Widoku utworzonej mapy.
- Ustawień ogólnych.
- Ustawień struktury danych.

- Ustawień domyślnej lokalizacji mapy.
- Formularza dodawania nowego miejsca.
- Listy wszystkich dodanych miejsc.

Rysunek 25 - Panel administracyjny

5.3.3 Zarządzanie ustawieniami

Na rys. 26 przedstawiony został formularz ustawień ogólnych. Umożliwia on zmianę nazwy wyświetlanej mapy.

Rysunek 26 - Ustawienia ogólne

Rys. 27 przedstawia sekcję panelu administracyjnego umożliwiającą ustawienie domyślnej lokalizacji mapy. Zmian dokonuje się za pomocą myszki. Zmiana wyświetlanego obszaru następuje poprzez klikanie i przeciąganie, natomiast poziom zbliżenia ustawia się za pomocą scrolla. Nie ma konieczności ręcznego zatwierdzania wprowadzonych ustawień, ponieważ zaznaczony region i zbliżenie jest automatycznie, w sposób asynchroniczny wysyłany do serwera, gdzie następuje ich zapis.

Rysunek 27 - Ustawienia domyślnej lokalizacji mapy

Dla celów niniejszego przykładu jako nazwa mapy została wprowadzona *Agencja nieruchomości*, a okolice Warszawy zostały ustawione jako domyślnie wyświetlany region.

5.3.4 Zarządzanie strukturą danych

Ustalenie struktury wprowadzanych danych jest jedną z najważniejszych funkcjonalności wytworzonej aplikacji. Zarządzanie strukturą danych zgodnie z wymaganiami zostało zrobione w postaci wizualnego edytora. Edytor ten cechuje się wysoką interaktywnością oraz wykorzystuje technikę drag-and-drop.

Rysunek 28 - Zarządzanie strukturą danych

Na rys. 28 przestawiony został widok, który użytkownik zastanie, gdy po raz pierwszy wejdzie do tej sekcji panelu administracyjnego. Moduł ten składa się z dwóch części: z lewej widoczny jest tworzony formularz do wprowadzania danych, z prawej lista rodzajów pól do wyboru. Po kliknięciu na wybrane z prawej strony pole jest ono

automatycznie dodawane do formularza. Na rys. 29 przedstawiony został widok formularza z dodanymi paroma polami. Można zmienić kolejność wszystkich pól oprócz pierwszego. Aby tego dokonać należy złapać wybrane pole za ikonkę ze strzałkami i wykorzystując technikę drag-and-drop umieścić je w wybranym miejscu.

Rysunek 29 - Zarządzanie strukturą danych. Opcje pola

Każde pole ma możliwość konfiguracji – po kliknięciu na ikonkę klucza z prawej strony pojawia się dodatkowa sekcja z formularzem umożliwiającym zmianę ustawień. Wszystkie pola mają możliwość zmiany etykiety oraz usunięcia. Pola tj. pole wyboru mają dodatkowe możliwości konfiguracji np. listę opcji (patrz rys. 29).

Dla celów niniejszego przykładu została stworzona struktura (patrz rys. 30) umożliwiająca przechowywanie następujących informacji:

- Nazwa – pole tekstowe dla nazwy oferty
- Rodzaj – pole wyboru z opcjami: Dom, Mieszkanie
- Rynek – pole wyboru z opcjami: Pierwotny, Wtórny
- Powierzchnia – pole numeryczne
- Liczba pokoi – pole numeryczne
- Cena – pole numeryczne
- Opis – wzbogacone pole tekstowe

Rysunek 30 - Struktura danych dla agencji nieruchomości

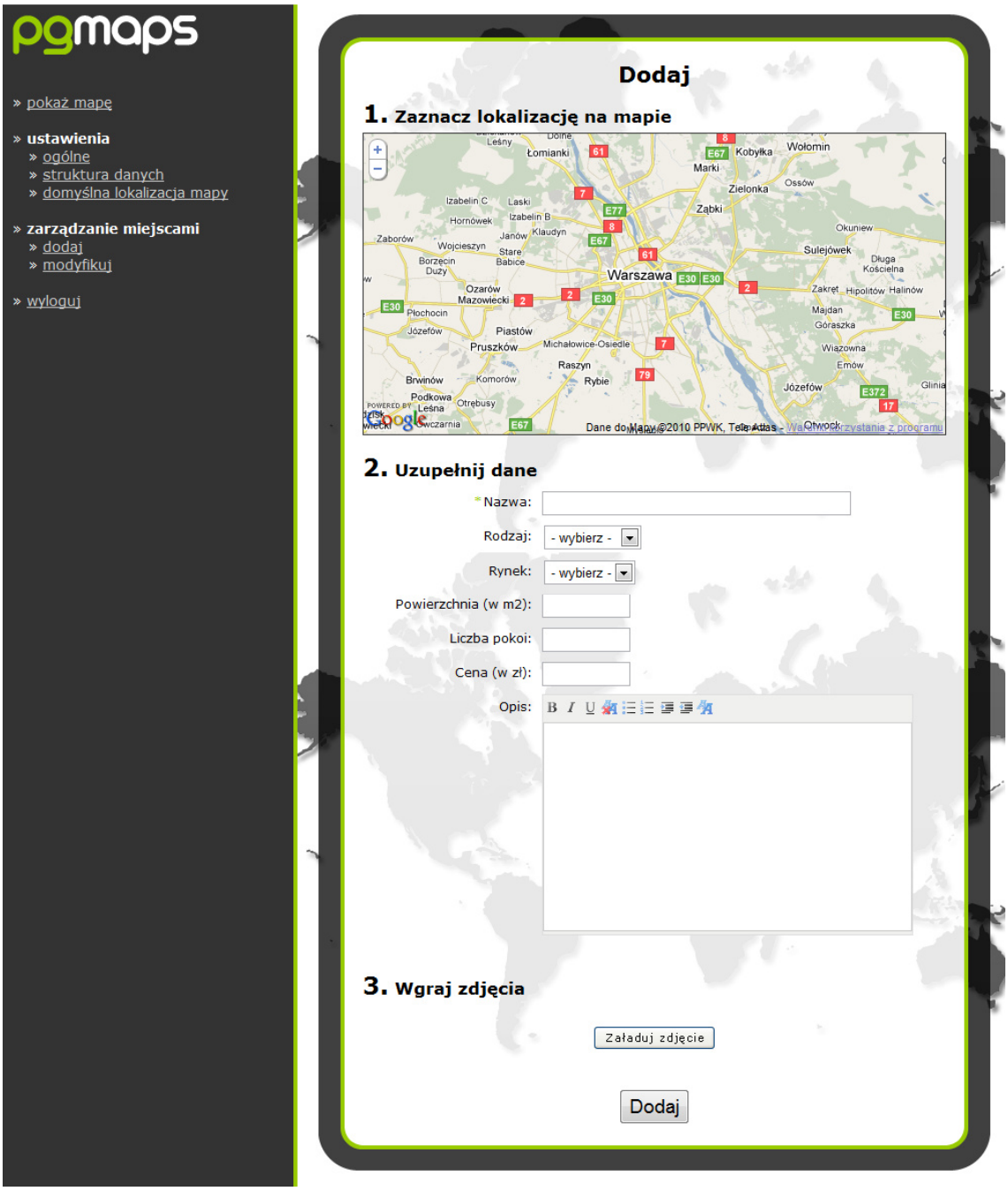
5.3.5 Dodawanie nowych miejsc

Przedstawiony na rys. 31 moduł do dodawania miejsc składa się z trzech części:

- Mapki – na której za pomocą kliknięcia można zaznaczyć lokalizację dodawanego miejsca.
- Formularza – wygenerowanego na podstawie ustalonej wcześniej struktury danych.
- Sekcji do wgrywania zdjęć.

Po zatwierdzeniu, wszystkie dane z formularza są walidowane w taki sposób, aby odpowiadały poszczególnym typom pól. W razie wystąpienia błędów użytkownik zostaje o tym stosownie poinformowany. Wgrywanie zdjęć odbywa się asynchronicznie. Wgrane zdjęcia pojawiają się w postaci miniaturek, które można zamieniać kolejnością poprzez stosowanie techniki drag-and-drop.

Dla celów niniejszego przykładu zostało dodanych około 20 przykładowych nieruchomości na terenie Warszawy i okolic.



Rysunek 31 - Formularz dodawania miejsc dla agencji nieruchomości

5.3.6 Zarządzanie miejscami

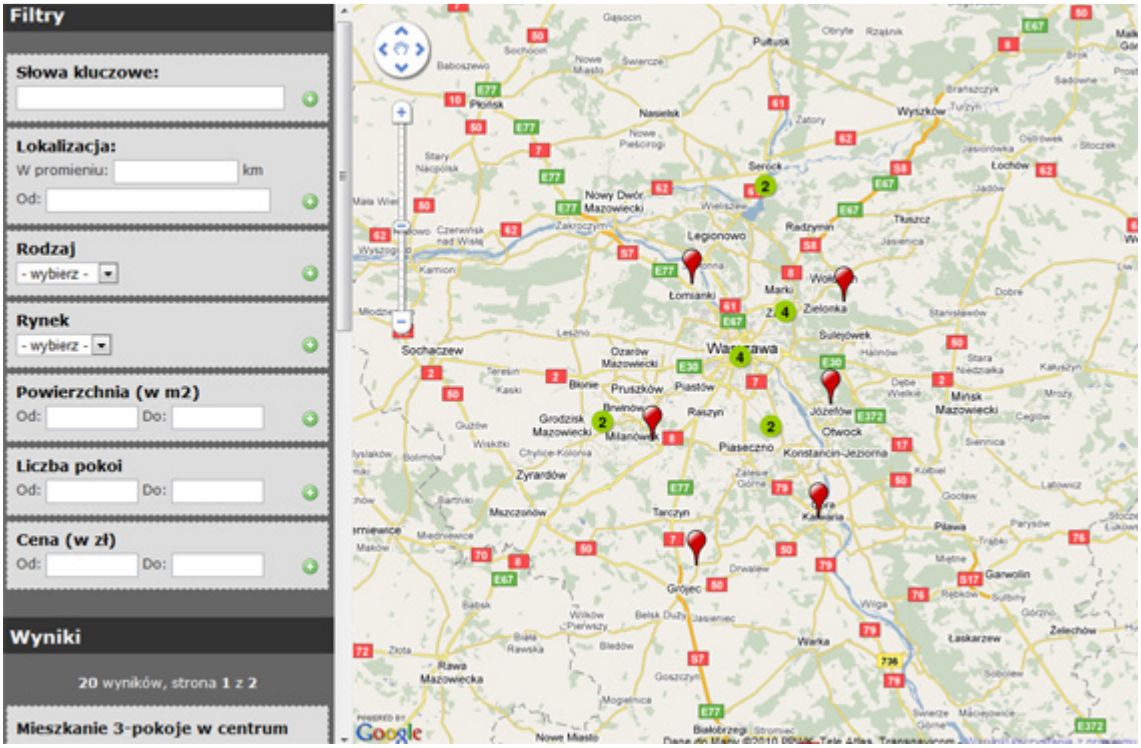
Rys. 32 przedstawia sekcję panelu administracyjnego umożliwiającą zarządzanie dodanymi miejscami. Wyświetlana jest stronicowana lista, którą można filtrować za pomocą słów kluczowych oraz przedziału daty. Ikony po prawej stronie każdego elementu umożliwiają usunięcie elementu oraz jego edycji. Moduł edycji miejsc jest tożsamy z modułem dodawania.

Na rys. 32 widać pierwszą stronę listy nieruchomości dodanych w ramach niniejszego przykładu.

Rysunek 32 - Lista dodanych miejsc

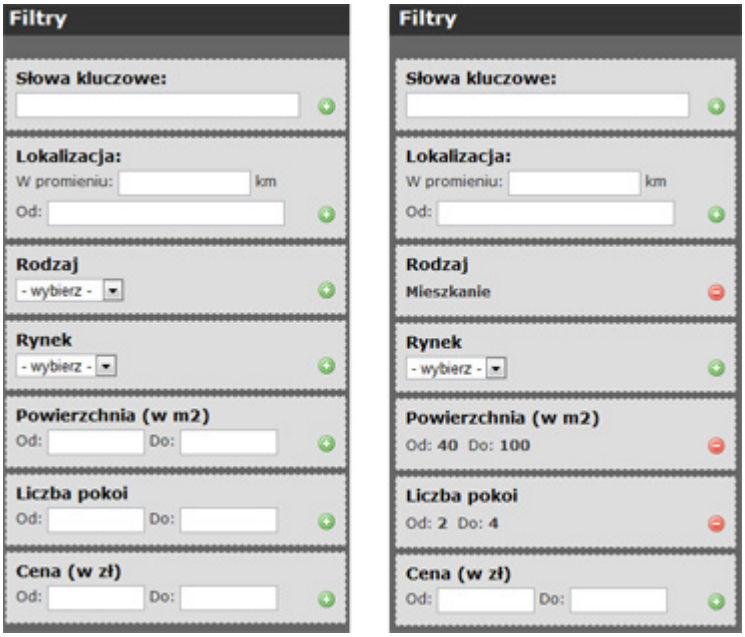
5.3.7 Przeglądanie mapy

Rys. 33 przedstawia skonfigurowaną i gotową do użycia mapę. Widok mapy podzielony jest na trzy części. Z lewej strony znajdują się filtry i lista wyników, natomiast z prawej strony znajduje się właściwa mapa.



Rysunek 33 - Gotowa mapa

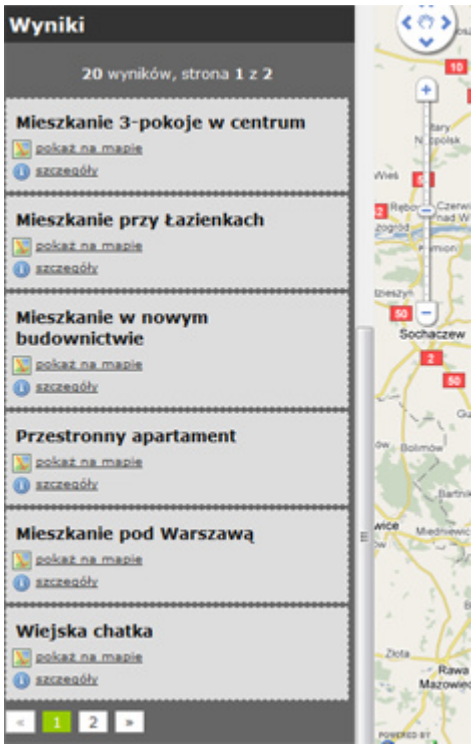
Filtry wygenerowane są automatycznie na podstawie struktury danych zdefiniowanej w panelu administracyjnym. Aby uaktywnić filtr należy wypełnić jego pola i kliknąć na ikonkę znajdującą się z jego prawej strony. Po kliknięciu, dane zarówno na mapie, jak i liście wyników zostaną w sposób asynchroniczny zaktualizowane. Specjalnymi rodzajów filtrów są filtry słów kluczowych i lokalizacji. Po zaaplikowaniu filtra słów kluczowych zwracane są wszystkie miejsca, które zawierają te słowa w jakimkolwiek polu tekstowym. Natomiast filtr lokalizacji ogranicza wyświetlane miejsca tylko do tych, znajdujących się w pewnym promieniu od podanej lokalizacji. Na rys. 34 przedstawiony jest widok przed i po zastosowaniu filtrów. Zastosowane filtry to: rodzaj – mieszkanie; powierzchnia od 40 do 100 m2; liczba pokoi – od 2 do 4.



Rysunek 34 - Filtry - z lewej wszystkie nieaktywne. Z prawej - część aktywna

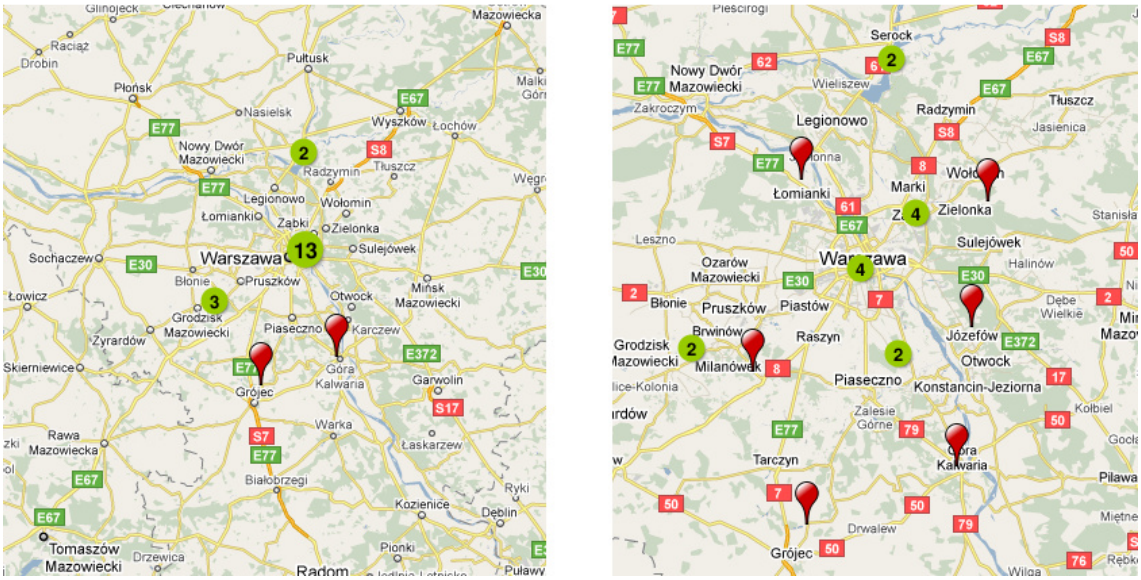
Na rys. 35 pokazana jest lista wyników. Na tej liście znajdują się jedynie miejsca, które spełniają wszystkie kryteria zdefiniowane w filtrach. Każdy element listy posiada dwie opcje:

- pokaż na mapie – wyśrodkowuje miejsce na mapie oraz zwiększa poziom zbliżenia do takiego poziomu, aby nie było zgrupowane.
- Szczegóły – wyświetla podsumowanie miejsca zawierające szczegółowe informacje wraz ze zdjęciami.



Rysunek 35 - Lista wyników

Mapa stworzona jest w taki sposób, aby pobierała z serwera tylko punkty znajdujące się w jej widocznym obszarze. W momencie kiedy zmieniany jest region lub poziom zbliżenia, lista punktów jest asynchronicznie aktualizowana. Jak widać na rys. 33 punkty różnią się od siebie ikonką oraz niektóre z nich są dodatkowo opisane liczbą. Punkty bez etykiety liczbowej to pojedyncze miejsca, natomiast punkty z liczbą są grupą punktów. Etykieta liczbowa informuje o tym, ile punktów zawiera w sobie dana grupa. W zależności od poziomu zbliżenia miejsca są inaczej grupowane.



Rysunek 36 - Wpływ zbliżenia na grupowanie

Rys. 36 obrazuje jaki sposób zbliżenie wpływa na grupowanie obiektów. Z lewej strony przedstawiona jest okolica Warszawy w zbliżeniu o jeden poziom mniejszym niż z prawej strony.

Po kliknięciu na punkt będący pojedynczym miejscem lub wybraniu z listy wyników opcji *szczegóły* wyświetlone zostanie podsumowanie miejsca, zawierające szczegółowe informacje wraz ze zdjęciami. Na rys. 37 zostało pokazane przykładowe podsumowanie miejsca.

Rysunek 37 - Przykładowe podsumowanie miejsca

6 Zalety, wady oraz plany rozwojowe

Wytworzony prototyp do tworzenia spersonalizowanych map internetowych, realizuje zaproponowane przez autora założenia. Mimo, że aplikacja jest w pełni funkcjonalna, nie jest produktem doskonałym. Istnieją pewne wady oraz rozwiązania, które mogłyby uczynić ją bardziej atrakcyjną, lecz nie zostały one zastosowane.

6.1 Zalety oraz wady przyjętych rozwiązań

Niewątpliwą zaletą powstałego prototypu jest to, że cechuje się wysokim poziomem interaktywności osiągniętym dzięki zastosowaniu technologii AJAX oraz technik tj. drag-and-drop. Aplikacja została stworzona w taki sposób, aby jej obsługa nie wymagała żadnej specjalistycznej wiedzy komputerowej. Kolejną zaletą jest rozwiązanie nie spotkane przez autora u konkurencji – możliwość definiowania własnej struktury danych opisujących dodawane lokalizacje. Rozwiązanie to daje użytkownikom możliwość dodawania miejsc z dużą ilością rozbudowanych informacji. Dzięki temu możliwe było również skonstruowanie wyszukiwarki, umożliwiającej zaawansowane filtrowanie miejsc wyświetlanych na mapie.

Wytworzona aplikacja nie jest pozbawiona wad. Z racji tego, że jest to prototyp autor nie przykładał wystarczającej uwagi do kwestii tj. bezpieczeństwo czy zaawansowana obsługa błędów. Nie udało się również uniknąć błędnych decyzji dotyczących wyboru technologii, braków w specyfikacji funkcjonalności oraz drobnych błędów programistycznych.

Błędną decyzją było wybranie jQuery jako frameworku dla języka JavaScript. jQuery oferuje znakomitą możliwość manipulacji elementami DOM, posiada bogaty zbiór zaawansowanych efektów oraz niezliczoną ilość wtyczek rozszerzających jego funkcjonalność. Niestety programowanie w jQuery okazało się zbyt proceduralne. Wraz ze wzrostem ilości napisanego kodu, zarządzanie nim stawało się coraz trudniejsze. Programiści preferujący styl obiektowy zamiast jQuery powinni wybrać np. framework MooTools. MooTools w przeciwieństwie do jQuery nie jest biblioteka służącą głównie do manipulacji elementami DOM. MooTools zmienia sposób pisania kodu JavaScript i czyni go znacznie bardziej obiektowym i lepiej zorganizowanym.

Prototyp nie posiada znacznych braków w funkcjonalności. Jedynymi brakami zauważonymi przez autora są zbyt mała ilość typów pól możliwych do wybrania przy definiowaniu struktury danych oraz brak pola z auto-uzupełnianiem wspomagającego zaznaczenie miejsca na mapie przy dodawaniu nowych miejsc.

6.2 Proponowane plany rozwoju

Istnieją pewne dodatkowe funkcjonalności nie zaimplementowanej w prototypie, o które można by rozszerzyć aplikację. Prototyp jest zbudowany z wykorzystaniem architektury, która kładzie duży nacisk na luźne powiązanie komponentów, więc nie powinno być problemu z jego rozszerzeniem. Proponowane funkcjonalności, które mogły by być rozwinięte:

- API.
- Import i eksport danych.
- Internacjonalizacja.
- Szablony interfejsu graficznego.
- Możliwość zagnieżdżenia utworzonej mapy na stronach internetowych.

6.2.1 API

Wprowadzenie API umożliwiłoby programistom łatwą integrację swoich istniejących aplikacji z systemem do personalizacji map internetowych. API dałoby możliwość programistycznego zarządzania wszystkimi ustawieniami oraz dodawanie, usuwanie i edytowanie miejsc na mapie.

6.2.2 Import i eksport danych

Wprowadzanie możliwości importu i eksportu danych mogłoby znacznie usprawnić korzystanie ze spersonalizowany map. Nie byłoby konieczności ręcznego wprowadzanie dużej ilości danych. Użytkownik mógłby importować dane za pomocą utworzonego przez siebie pliku w formacie XML, JSON lub CSV. Dodatkowo udostępnione API umożliwiłoby programistom w łatwy sposób zautomatyzować eksportowanie i importowanie danych.

6.2.3 Internacjonalizacja

Kolejną ważną zmianą mogłaby być internacjonalizacja aplikacji. Aplikacja mogłaby udostępniać łatwy sposób dodawania nowych języków. Domyślnie język byłby ustalany na podstawie informacji zawartych w zapytaniu http, z możliwością zmiany w panelu administracyjnym. Mógłby być również zaimplementowany sposób ustawiania odpowiedniego formatu daty, czasu i liczb dla regionu, z którego użytkownik pochodzi.

6.2.4 Szablony interfejsu graficznego

Użytkownik mógłby mieć możliwość zmiany wyglądu swojej mapy. Mogłaby być możliwość edytowania specjalnych szablonów przypisanych do poszczególnych elementów interfejsu graficznego. Szablony wykorzystywałyby język HTML z dodatkowymi znacznikami identyfikującymi pewne sekcje. Oprócz edycji szablonów możliwa byłaby edycja styli CSS.

6.2.5 Możliwość zagnieżdżenia utworzonej mapy na stronach internetowych

Funkcjonalność ta po utworzeniu spersonalizowanej mapy, umożliwiłaby zagnieżdżenie jej w swojej stronie internetowej. Odbywałoby się to za pomocą specjalnego skrypty JavaScript lub ramki IFrame. Mapa dodatkowo mogłaby przyjmować dodatkowe parametry, które filtrowałyby wyświetlane miejsca.

7 Podsumowanie

Interaktywne mapy internetowe posiadają bardzo szerokie zastosowanie. Jednym z nich jest możliwość tworzenie spersonalizowanych map, które były tematem rozważań niniejszej pracy.

Autor pracy dogłębnie zbadał zagadnienie wraz dostępnymi na rynku rodzajami map internetowych oraz rozwiązaniami umożliwiającymi ich personalizację. Tym sposobem zidentyfikował, czym powinno się charakteryzować dobre rozwiązanie oraz zanalizował braki występujące w już istniejących rozwiązaniach. Autor pracy doszedł do wniosku, że aplikacja powinna posiadać łatwy, intuicyjny i wysoce interaktywny interfejs użytkownika, nie sprawiać problemu w obsłudze nawet niedoświadczonym użytkownikom komputerów oraz udostępniać możliwość precyzowania rodzaju danych opisujących dodawane na mapie miejsca. Dodatkowo mapa powinna mieć możliwość zaawansowanego filtrowania wyświetlanych miejsc.

Opracowany w wyniku tej pracy, w pełni funkcjonalny prototyp pokazuje, że możliwe jest stworzenie rozwiązania spełniającego wszystkie wymienione wyżej kryteria. Wytworzona aplikacja dzięki wysokiej generyczności umożliwia stworzenie spersonalizowanej mapy internetowej zawierającej dowolnego rodzaju geolokalizowalne informacje.

Bibliografia

[1] Taylor, Bret. Mapping your way.
http://googleblog.blogspot.com/2005/02/mapping-your-way.html.

[2] Lars Rasmussen on Inventing Google Maps.
http://www.abc.net.au/tv/bigideas/stories/2009/07/10/2620279.htm.

[3] Pegg, Mike. Big Birthday... Google Maps API Turns 5!
http://googlegeodevelopers.blogspot.com/2010/06/big-birthday-google-maps-api-turns-5.html.

[4] Informacje o MapVivo.
http://mapvivo.com/about.

[5] Microsoft Launches XML Web Services Revolution With Visual Studio .NET and .NET Framework.
http://www.microsoft.com/presspass/press/2002/feb02/02-13revolutionpr.mspx.

[6] Trygve Reenskaug. MVC XEROX PARC 1978-79.
http://heim.ifi.uio.no/~trygver/themes/mvc/mvc-index.html.

[7] Fluent NHibernate wiki - Getting started.
http://wiki.fluentnhibernate.org/Getting_started.

[8] Wikipedia - Odwzorowanie walcowe równokątne.
http://pl.wikipedia.org/wiki/Odwzorowanie_walcowe_r%C3%B3wnok%C4%85tne.

[9] Mika Tuupola. Introduction to Marker Clustering With Google Maps.
http://www.appelsiini.net/2008/11/introduction-to-marker-clustering-with-google-maps.

[10] Fowler, Martin. *Patterns of Enterprise Application Architecture.* Addison-Wesley Professional, 2002. ISBN-13: 978-0321127426.

[11] Construction, Object-Oriented Software. *Bertrand Meyer.* Prentice Hall, 1988. ISBN-10: 0136291554.

Spis rysunków

Rysunek 1 - Google Maps 9
Rysunek 2 - Yahoo Maps 11
Rysunek 3 - Bing Maps 12
Rysunek 4. Google My Maps - przykładowa mapa 13
Rysunek 5. MapVivo - przykładowa mapa 13
Rysunek 6. EnTrip - przykładowa mapa 14
Rysunek 7. Tagzania - przykładowa mapa 15
Rysunek 8 - Framework ASP.NET MVC 20
Rysunek 9 - Najkrótsza ścieżka między dwoma punktami na powierzchni kuli 33
Rysunek 10 - Ziemia w odwzorowaniu Mercatora [8] 34
Rysunek 11 - Zniekształcenia w odwzorowaniu Mercatora 35
Rysunek 12 - Grupowanie oparte na wirtualnej siatce [9] 36
Rysunek 13 - Grupowanie oparte na podstawie odległości między punktami. [9] 37
Rysunek 14 - Przykładowy model domenowy [9] 38
Rysunek 15 - Klasa Unit of Work [10] 39
Rysunek 16 - Wyszukiwanie za pomocą Repository [11] 40
Rysunek 17 - Ogólna architektura aplikacji 41
Rysunek 18 - Klasa kwerendy napisana w stylu fluent 43
Rysunek 19 - Klasa Repository 44
Rysunek 20 - Model domenowy 45
Rysunek 21 - Strona główna 46
Rysunek 22 - Formularz tworzenia nowej mapy 47
Rysunek 23 - Formularz logowania 47
Rysunek 24 – Informacja o błędach w formularzu 48
Rysunek 25 - Panel administracyjny 49
Rysunek 26 - Ustawienia ogólne 49
Rysunek 27 - Ustawienia domyślnej lokalizacji mapy 50
Rysunek 28 - Zarządzanie strukturą danych 50
Rysunek 29 - Zarządzanie strukturą danych. Opcje pola 51
Rysunek 30 - Struktura danych dla agencji nieruchomości 52
Rysunek 31 - Formularz dodawania miejsc dla agencji nieruchomości 53
Rysunek 32 - Lista dodanych miejsc 54
Rysunek 33 - Gotowa mapa 54
Rysunek 34 - Filtry - z lewej wszystkie nieaktywne. Z prawej - część aktywna 55
Rysunek 35 - Lista wyników 56
Rysunek 36 - Wpływ zbliżenia na grupowanie 56
Rysunek 37 - Przykładowe podsumowanie miejsca 57

Spis przykładów

Przykład 1 - Mapowanie za pomocą HBM XML [7] 25
Przykład 2 - Mapowanie za pomocą Fluent NHibernate [7] 25
Przykład 3 - Przykładowe zapytanie w HQL..... 26
Przykład 4 - Przykładowe zapytanie z wykorzytaniem Criteria API 26
Przykład 5 - Przykładowe zapytanie w LINQ 26
Przykład 6 - Interfejs IQuery 42
Przykład 7 - Wywołanie w stylu fluent 43
Przykład 8 - Odpytywanie bazy za pomocą wyrażeń lambda 43