Politechnika Śląska
Wydział Automatyki, Elektroniki i Informatyki

kierunek informatyka

Katarzyna Biernat

Wirtualny przewodnik dla urządzeń mobilnych

projekt inżynierski

kierujący pracą: dr inż. Krzysztof Simiński

Gliwice, 4 stycznia 2015

Spis treści

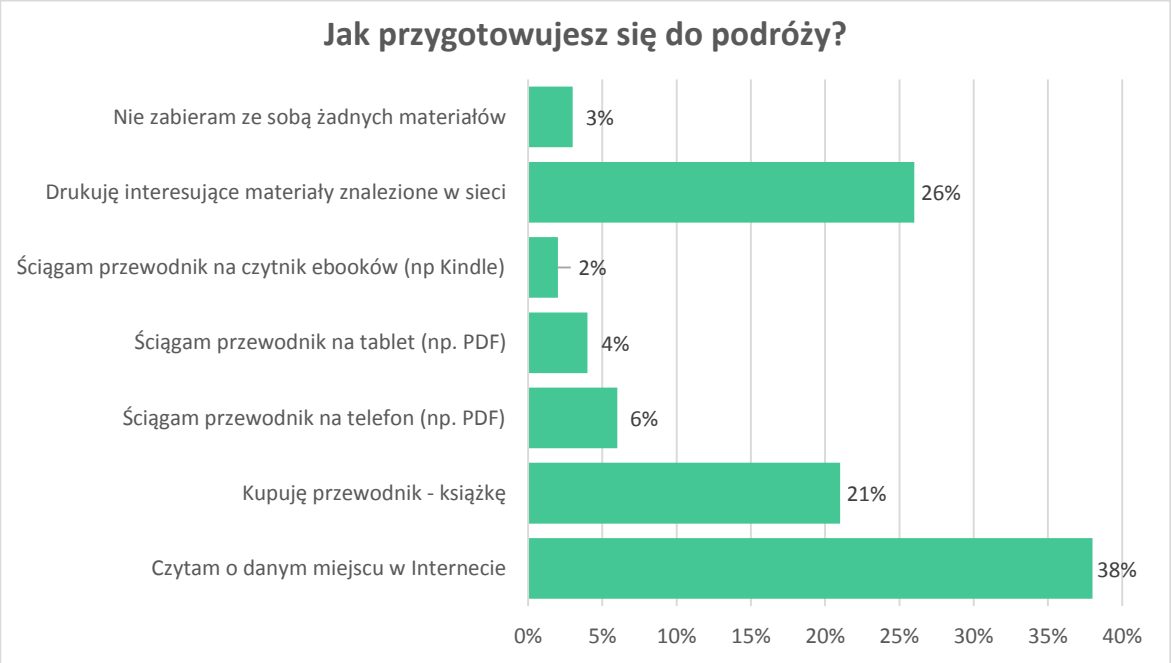
Rozdział 1

Wstęp

Ludzi od lat fascynowały dalekie podróże i nieznane lądy. Już starożytni zastanawiali się co znajduje się za wielką wodą. Chyba każdy ze szkoły zna opowieść o mitycznym Odyseuszu i jego wyprawie autorstwa Homera czy też *Anabazę* pióra Ksenofonta opisującą brawurowy powrót greckiego wojska najemnego spod Kunaksy w Babilonii do Grecji. Kiedy w czasach renesansu pojawiła się ku temu możliwość, ludzie wsiedli na statki i wypłynęli w szerokie morze w poszukiwaniu nowych, nieznanych terenów. I tak też w 1492 roku Krzysztof Kolumb dopłynął aż do Ameryki, a w 1519 Ferdynand Magellan wyruszył opłynąć cały świat. Warto jednak wspomnieć, że jeszcze do niedawna podróże zarezerwowane były dla nielicznych. Ograniczeniem były finanse i brak szybkich środków lokomocji, ale i ludzki strach przed nieznanym. Dopiero po 1945 roku [17], wraz z rozwojem motoryzacji, kolei i lotnictwa oraz ogólnym polepszeniem standardu życia, zagraniczne wakacje stały się popularne na skalę masową. Spowodowało to gwałtowną ekspansję branży turystycznej, na którą składają się nie tylko biura podróży czy hotele, ale i wydawnictwa podróżnicze wydające przewodniki turystyczne.

Branżą, która rozwija się jeszcze szybciej niż turystyka, jest branża IT. Jeszcze 10 lat temu nikomu nie śniło się, że dziś prawie każdy posiadać będzie telefon typu smartfon. Turystyka nie nadąża za rozwojem IT, więc w chwili obecnej na rynku istnieje nisza, jeśli chodzi o przewodniki elektroniczne.

Celem niniejszego projektu jest stworzenie przyjaznej platformy *TravelGuide* przeznaczonej dla osób podróżujących. Program ma ułatwić użytkownikowi zwiedzanie nowych miejsc oraz zupełnie zastąpić tradycyjny, książkowy przewodnik. Na całość składać się będzie aplikacja mobilna na platformę Windows Phone 8.1 oraz aplikacja działająca w przeglądarce internetowej. W Internecie można znaleźć bardzo wiele różnych informacji o ciekawych miejscach turystycznych, brakuje jednak przyjaznej formy przedstawienia ich podróżnikowi. Rysunek 1.1 przedstawia wyniki ankiety przeprowadzonej wśród 234 użytkowników forum Fly4Free.pl [14]. Jak się okazuje, zdecydowana większość podróżników zakupuje przewodnik o miejscu docelowym bądź pobiera stosowne informacje z sieci i drukuje je na swój użytek. Aplikacja *TravelGuide* ma na celu połączenie zalet obu tych rozwiązań. Prezentować



Rysunek 1.1: Wyniki ankiety przeprowadzonej wśród użytkowników forum fly4free.pl

będzie informacje w podobny sposób jak tradycyjny książkowy przewodnik przy jednoczesnej możliwości szybkiej aktualizacji informacji. Ponadto dzięki temu, że zawierać się będzie w smartfonie podróżnika, zwalnia go z noszenia ciężkich książek lub nieporęcznych drukowanych kartek. Dzięki wykorzystaniu nowoczesnych technologii, aplikacja będzie mogła podawać użytkownikowi najciekawsze informacje bazując na jego danych geolokalizacyjnych.

Praca jest zorganizowana w następujący sposób: rozdział 1 opisuje wprowadzenie do zagadnienia i uzasadnienie podjęcia się takiej tematyki. Rozdział 2 przedstawia użyte technologie, zarówno w części serwerowej (podrozdział 2.2), internetowej (podrozdział 2.3) jak i mobilnej (podrozdział 2.4). Rozdział 3 zawiera specyfikację zewnętrzną, w tym: wymagania systemowe (podrozdział 3.2), opis instalacji poszczególnych aplikacji (podrozdział 3.3) i instrukcję obsługi dla użytkowników końcowych aplikacji mobilnej i internetowej wraz ze zdjęciami systemu (podrozdział 3.4). Rozdział 4 zawiera specyfikację wewnętrzną, w tym: opis architektury (podrozdział 4.2), najważniejsze algorytmy (podrozdział 4.3), przypadki użycia (podrozdział 4.1) oraz opis komponentów (podrozdział 4.4). Rozdział 5 opisuje sposób, w jaki projekt był testowany, oraz prezentuje wybrane scenariusze testowe. Rozdział 6 przedstawia wnioski końcowe, w tym kierunek dalszych prac (podrozdział 6.1) i napotkane problemy (podrozdział 6.2). Zakończenie znajduje się w rozdziale 7.

Rozdział 2

Technologie i narzędzia

Rozdział opisuje technologie i narzędzia użyte do stworzenia części webowej, serwerowej i mobilnej aplikacji.

2.1 Narzędzia

2.1.1 Środowisko programistyczne

Całość projektu została zrealizowana za pomocą Visual Studio 2013. Jest to jedno z popularniejszych środowisk programistycznych, wyprodukowane przez firmę Microsoft. Mimo że nie jest to środowisko bezpłatne, projekt mógł zostać zrealizowany bez ponoszenia kosztów dzięki licencji akademickiej *DreamSpark*.

Visual Studio 2013 w pełni wspiera rozwijanie oprogramowania w platformie .NET; zarówno rozwiązania mobilne jak i internetowe. Poza tym, dzięki szeregowi dostępnych bibliotek, wspomaga pracę z HTML, LESS oraz JavaScript.

Środowisko to zostało wybrane ze względu na następujące zalety:

- bardzo dobra obsługa *IntelliSense* dla języka C#;
- wiele pomocnych rozszerzeń, np. *Web Essentials*;
- wsparcie dla składni języków HTML[1], CSS[2] i JavaScript;
- wbudowany emulator Windows Phone;
- gotowe szablony, które przyspieszają pracę z projektem.

2.1.2 System kontroli wersji

Pomimo że projekt realizowany był indywidualnie, zdecydowano się na wykorzystanie systemu kontroli wersji. Na rynku dostępnych jest wiele takich systemów. Do najpopularniej-

[1] *HyperText Markup Language* – hipertekstowy język znaczników wykorzystywany do tworzenia stron internetowych.

[2] *Cascading Style Sheets* – kaskadowe arkusze stylów służące do opisu formy prezentacji strony WWW.

szych należą: GIT [15], Mercurial [20], SVN [27], Bazaar [10] oraz TFS [28]. Podczas projektu zdecydowano się na wykorzystanie systemu GIT. Został on wybrany z wielu powodów. Jest to system rozproszony, więc jego repozytoria są relatywnie małe (np. w porównaniu do SVN). Ponadto GIT oferuje wiele funkcjonalności, które ułatwiają pracę z projektem, m.in. tzw. gałęzie (bardziej elastyczne w porównaniu do gałęzi SVN), etykiety czy lokalną przestrzeń roboczą. Nie bez znaczenia był także fakt, że autorka ma doświadczenie z tym systemem kontroli wersji.

Repozytorium zostało umieszczone w serwisie Github.com[3], który agreguje projekty z całego świata. Portal oferuje przejrzysty interfejs oraz możliwość graficznego przeglądania historii projektu.

2.1.3 Narzędzia projektowe UML

Aby lepiej wykonać projekt, przed przystąpieniem do programowania zostały wykonane pomocnicze diagramy UML. Na rynku istnieje wiele narzędzi do tego przeznaczonych, zarówno płatnych jak i darmowych. Wśród rozwiązań płatnych prym wiedzie program Enterprise Architect [12], który jest szeroko wykorzystywany przez profesjonalnych analityków przy komercyjnych projektach. W przypadku projektu *TravelGuide* zdecydowano się na wykorzystanie bezpłatnego programu StarUML [24]. Został on wybrany ze względu na szereg zalet, jakich nie znaleziono w innych podobnej klasy aplikacjach. Są to m. in. łatwa i intuicyjna obsługa, duży wybór dostępnych typów diagramów, możliwość eksportu do formatu wektorowego oraz wsparcie standardu UML 2.0.

2.2 Aplikacja serwerowa

Część serwerowa, realizowana jako WebService, wykonana została z wykorzystaniem platformy ASP.NET WebApi 2 [7]. Całość opiera się na platformie .NET 4.5 i napisana jest w języku C#.

W tej części wykorzystano szereg bibliotek. Aby przyspieszyć pracę i zapewnić większe bezpieczeństwo aplikacji, do celów identyfikacji i uwierzytelniania użytkowników została wykorzystana biblioteka AspNet.Identity [6].

Do pracy z kolekcjami wykorzystano bibliotekę LINQ. Jest to zestaw niezwykle przydatnych narzędzi, które ułatwiają pracę z takimi strukturami jak listy, kolejki i inne implementujące interfejs `IEnumerable`.

Nieodzowna okazała się być biblioteka Newtonsoft.Json [22], która wspomaga formatowanie obiektów na kod JSON[4]. Wszystkie akcje zwracające dane zwracają je właśnie w tym formacie. Komunikat w formacie JSON jest literałem obiektu języka JavaScript. Dane przekazywane są jako tablica asocjacyjna a wszystkie dane są zmiennymi.

[3] Serwis dostępny pod adresem `http://github.com`

[4] *JavaScript Object Notation* – tekstowy format wymiany danych

Jako że WebService działa w innej domenie niż aplikacja użytkownika, niezbędne było dodanie odpowiednich nagłówków HTTP, aby umożliwić zapytania z innej domeny (*Cross Origin Requests*). Bardzo ułatwia to biblioteka Microsoft.AspNet.Cors [5]. Dodaje ona szereg nagłówków wraz z odpowiednio dobranymi parametrami.

WebService komunikuje się z bazą danych, która działa w oparciu o silnik SQL Server 2014 [25]. Komunikacja odbywa się z wykorzystaniem biblioteki Entity Framework [13]. Jest to biblioteka typu ORM[5], która umożliwia mapowanie wyników pobranych z bazy danych na wcześniej przygotowane modele. Modele te są następnie mapowane na modele przejściowe (*ViewModels*) za pomocą biblioteki AutoMapper [8].

2.3 Aplikacja internetowa

Aplikacja internetowa napisana została w modelu *client-side*. Oznacza to, że cała aplikacja interpretowana jest po stronie przeglądarki i nie wymaga specjalistycznego serwera. Aby działała poprawnie, wystarczy dowolny serwer HTTP. Kod został napisany przy użyciu następujących języków: Javascript, HTML i LESS[6]. Aplikacja stworzona jest w oparciu o platformę AngularJS [3]. Jest to zestaw otwartych bibliotek języka JavaScript wspierany przez firmę Google. Umożliwia tworzenie aplikacji internetowych, z których można korzystać bez przeładowania strony (*Single Page Application*). Dane ładowane są w tle dzięki asynchronicznym zapytaniom do WebService realizowanym przez JavaScript.

Zostały również zastosowane poboczne biblioteki. Do najważniejszych zależą:

LeafletJS [19] – umożliwia sprawne dodawanie map do strony. Do wyświetlania map wybrany został silnik OpenStreetMap [23] ze względu na swoją otwartą licencję.

jQuery [18] – jest to zestaw bibliotek pomocniczych dla języka JavaScript. Rozszerza funkcjonalność i upraszcza składnię. W prostszy sposób pozwala na dostęp i manipulację elementami DOM.

Bootstrap [29] – zestaw bibliotek, który dołączony jest do Twitter Bootstrap. Dostarcza takie moduły jak np. interaktywne wyskakujące okna, animacje zamknięcie i in.

AngularTranslate.js [4] – biblioteka rozszerzająca możliwości platformy Angular o implementację wielojęzyczności.

Za część graficzną aplikacji odpowiada zmodyfikowany przez autorkę szablon AdminLTE [2] napisany w oparciu o Twitter Bootstrap. Szablon opisany jest językiem LESS, który jest następnie transformowany do CSS.

[5] *Object-Relational Mapping* – mapowanie obiektowo-relacyjne.
[6] *Leaner CSS* – dynamiczny język arkuszy stylów.

2.4 Aplikacja mobilna

Na część mobilną składa się aplikacja dedykowana dla platformy Windows Phone 8.1. Projekt zrealizowany został w modelu Universal Apps, więc cała logika wydzielona jest do osobnego projektu.

Aplikacja wykonana została przy wykorzystaniu platformy .NET, języka C# oraz XAML. Zaimplementowany został wzorzec architektoniczny MVVM (*Model View ViewModel*, *vide* rozdział 4.2.3). W tym celu do projektu dołączona została biblioteka MVVM Light [21].

Aplikacja pobiera dane z WebService i zapisuje je w swojej lokalnej bazie danych. Działa ona dzięki SQLight [26]. Do projektu dołączony jest adapter SQLite. Pobrane dane parsowane są do obiektów za pomocą biblioteki Newtonsoft.JSON.

Rozdział 3

Specyfikacja zewnętrzna

3.1 Słownik pojęć

W pracy pojawiają się sformułowania, które zdefiniowane są poniżej.

Użytkownik-twórca – jest to użytkownik, który posiada uprawnienia do tworzenia nowych treści. Może je dodawać za pomocą aplikacji internetowej.

Użytkownik-podróżnik – jest to użytkownik, który korzystając z aplikacji mobilnej korzysta z treści dodanych przez twórcę.

Przewodnik – w kontekście aplikacji *TravelGuide* jest to obiekt zawierający zbiór informacji o danym mieście, regionie lub kraju. Zawiera informacje ogólne, atrybuty i miejsca.

Informacje ogólne przewodnika – jest to część przewodnika. Zawiera fotografię, nazwę, opis i współrzędne geograficzne.

Atrybuty przewodnika – jest to zbiór dodatkowych informacji o przewodniku. Może zawierać takie informacje jak np. historia regionu czy wskazówki dojazdu, polecane restauracje, opis życia nocnego itp. Każdy atrybut zawiera tytuł, treść oraz ikonę wybraną z udostępnionego zbioru.

Miejsca przewodnika – jest to zbiór ciekawych turystycznie miejsc znajdujących się w okolicy opisywanej przez przewodnik. Miejsce zawiera nazwę, opis, fotografię oraz współrzędne geograficzne.

Elementy składowe przewodnika – jest to określenie agregujące opisane powyżej miejsca oraz atrybuty.

3.2 Wymagania

Aplikacja mobilna zaprojektowana jest z myślą o platformie Windows Phone 8.1. Obecnie nie jest dostępna dla użytkowników systemów Android czy iOS, jednakże docelowo zostanie

dodane wsparcie dla tych platform. Jeżeli użytkownik posiada telefon z systemem Windows Phone 8, powinien najpierw zaktualizować swój system. Następnie może pobrać i korzystać z aplikacji. System Windows Phone 7 nie jest wspierany.

Aplikacja internetowa do poprawnego działania wymaga nowoczesnej przeglądarki z włączoną obsługą JavaScript. Wspierane przeglądarki:

- Google Chrome w wersji 39 lub nowszy;
- Mozilla Firefox w wersji 32 lub nowszy;
- Opera w wersji 26 lub nowsza;
- Internet Explorer 11 lub nowszy.

Aplikacja internetowa działa także na urządzeniach o mniejszych rozdzielczościach takich jak tablety czy telefony.

3.3 Instalacja

Aplikacja mobilna *TravelGuide* w aktualnej wersji zainstalowana może być jedynie na telefonach z odblokowaną opcją deweloperską. Docelowo jednak program zostanie umieszczony w sklepie *Windows Store*, skąd będzie dostępny do pobrania dla użytkowników polsko- i angielskojęzycznych.

Aplikacja internetowa nie wymaga instalacji. Dostępna jest dla każdego, kto jest podłączony do Internetu, dysponuje kompatybilną przeglądarką internetową i posiada uprawnienia do dodawania nowych przewodników.

3.4 Instrukcja obsługi

Poniższa sekcja zawiera instrukcję obsługi aplikacji mobilnej i aplikacji internetowej dla użytkowników końcowych. Interfejs programów został zaprojektowany tak, by można było korzystać z nich w sposób jak najbardziej intuicyjny. W kolejnych podrozdziałach zostaną przedstawione scenariusze korzystania z aplikacji internetowej (podrozdział 3.4.1) i aplikacji mobilnej (podrozdział 3.4.2).

3.4.1 Instrukcja obsługi aplikacji internetowej

Niech przykładowym użytkownikiem systemu będzie Marian Pisarz. Marian jest użytkownikiem-twórcą i chce dodać nowy przewodnik o Lizbonie i udostępnić go podróżnikom. Chce również mieć możliwość edycji dodanego przez siebie przewodnika. Poniższy rozdział opisuje sposób dodawania, modyfikacji, przeglądania i publikacji przewodników przez twórców.

Tworzenie przewodnika

Marian uruchamia przeglądarkę internetową i wpisuje adres www aplikacji *TravelGuide*. Przeglądarka przenosi go do ekranu logowanie (rys. 3.1a). Marian wpisuje swoją nazwę użytkownika i hasło a następnie klika przycisk „Zaloguj". Zostaje przekierowany na stronę powitalną (rys. 3.1b). Następnie wybiera z menu podręcznego opcję „Przewodniki" → „Dodaj". Przeglądarka przekierowuje go do kreatora dodawania nowego przewodnika (rys. 3.1c).

W kroku pierwszym Marian powinien wymagane pola: wpisać nazwę przewodnika w polu *Nazwa* i opis w polu *Opis*. Następnie, klika na mapę aby zaznaczyć odpowiednie miejsce. Po kliknięciu na mapę pokazuje się na niej niebieski znacznik. Jeżeli Marian popełni błąd, może myszką przeciągnąć znacznik na odpowiednie miejsce. Finalnie, Marian dodaje zdjęcie główne dla przewodnika (okładkę). Może to zrobić na dwa sposoby: przeciągając zdjęcie na szare pole (metoda *Drag and Drop*[1]) lub klikając przycisk „Dodaj zdjęcie" i wybierając odpowiedni obrazek z dysku twardego komputera.

Jeżeli Marian nie wypełni któregoś z pól, przy próbie zapisu otrzyma informację, że dane pole jest wymagane. Jeśli formularz zostanie wypełniony poprawnie, Marian może kontynuować. Zostanie przeniesiony przez swoją przeglądarkę do kroku drugiego kreatora (rys. 3.1d).

W tym etapie kreator tworzenia przewodnika udostępnia możliwość dodania atrybutów, czyli ogólnych informacji o danym miejscu. Marian może dodać takie informacje jak historia, wskazówki dojazdu, ciekawostki, noclegi itp. Na tym etapie atrybuty można dowolnie dodawać i usuwać korzystając z przycisków „Dodaj atrybut" i „Usuń ten atrybut". Każdy atrybut traktowany jest niezależnie i przy każdym atrybucie pola *Tytuł*, *Opis* i *Symbol* są wymagane. Kliknięcie przycisku „Wybierz" pod etykietą *Symbol* powoduje wyświetlenie dodatkowego okna z wyborem możliwych symboli.

Gdy Marian kończy dodawanie atrybutów naciska przycisk „Zapisz i Kontynuuj" i przechodzi do trzeciego kroku kreatora (rys. 3.1e).

W tym etapie należy dodać zestaw interesujących obiektów w danym miejscu. Proces dodawania wygląda następująco: Marian klika na mapie wybierając lokalizację obiektu. Może także skorzystać z wyszukiwarki umieszczonej na górze mapy. Następnie uzupełnia pola *Nazwa*, *Opis* oraz *Kategoria*. Pole *Kategoria* jest polem typu lista rozwijalna i zawiera takie pozycje jak: *Zamki i Pałace*, *Pomniki*, *Kościoły i Cmentarze* itp. Wszystkie pola są wymagane i jest to sprawdzane podczas zapisu. Finalnie Marian wybiera obrazek reprezentujący dane miejsce. Proces dodawania obrazka jest taki sam jak w pierwszym kroku kreatora.

Kiedy miejsca zostaną dodane, Marian klika przycisk „Zapisz i Kontynuuj". Przewodnik zostaje zapisany a on zostaje przekierowany na stronę podsumowania (rys 3.1f).

Edycja przewodnika

Strona podsumowania jest jednocześnie stroną, z której możliwa jest edycja przewodnika. Aby edytować opis, nazwę lub zdjęcie danego miejsca, Marian klika zielony przycisk „Edytuj".

[1]Metoda przeciągnij i upuść

Statyczna sekcja z opisem zostaje dynamicznie zastąpiona formularzem (rys. 3.1g). Tutaj Marian może wprowadzić żądane zmiany. Ponownie jednak żadne pole nie może pozostać puste. Jeżeli na szare pole nie zostanie przeciągnięty żaden obrazek, poprzednia okładka zostaje zachowana. Zmiany można anulować klikając przycisk z symbolem X w prawym, górnym rogu formularza.

Analogicznie Marian może edytować atrybuty oraz miejsca. Może również usuwać bądź dodawać nowe korzystając z odpowiednich przycisków. Przycisk „Dodaj miejsce" (rys. 3.1h) aktywuje możliwość dodawania markera do mapy oraz dodaje formularz podobny do tego z kroku trzeciego kreatora.

Przy każdym z miejsc istnieją dwa przyciski: „Edytuj" oraz „Pokaż na mapie". Pierwszy przycisk zastępuje statyczną treść formularzem, podobnie jak ma to miejsce w przypadku informacji ogólnych. Drugi przycisk przenosi użytkownika z powrotem do mapy i oznacza wybrane miejsce chmurką z opisem.

Przegląd i publikacja

Korzystając z menu podręcznego Marian wybiera opcję „Przewodniki" → „Pokaż". Zostaje przekierowany do widoku tabeli, w której znajdują się wszystkie dodane przez niego przewodniki (rys. 3.1j). Oprócz okładki, nazwy i opisu widzi on także status każdego przewodnika. Nieopublikowane przewodniki nie są dostępne do pobrania przez użytkowników aplikacji mobilnej. Aby opublikować swój przewodnik Marian musi nacisnąć przycisk „Publikuj". Od teraz przewodnik może być pobrany na urządzenie mobilne a jego status oznaczony jest zieloną etykietą „Opublikowany".

3.4.2 Instrukcja obsługi aplikacji mobilnej

Niech przykładowym użytkownikiem systemu będzie Jacek Podróżnik. Jacek wybiera się na krótki urlop do Lizbony i chciałby zaopatrzyć się w aplikację na swój smarfon, która pomoże mu jak najlepiej poznać nowe miejsce.

Jacek pobiera aplikację z *Windows Store*. Program jest gotowy do użytku zaraz po instalacji. Po włączeniu widoczny jest ekran powitalny z logo (rys.3.2a). Chociaż aplikacja może działać w trybie *offline*, podczas pierwszego włączenia wymagane jest połączenie z Internetem.

Ekran powitalny gaśnie po kilku sekundach i użytkownik jest kierowany do strony logowania (rys. 3.2b). Jeżeli nie posiada on konta może je założyć lub pominąć logowanie i przejść od razu do użytkowania aplikacji. Jacek wybiera opcję rejestracji i jest przenoszony do widoku, w którym musi podać swój adres e-mail oraz hasło z potwierdzeniem (rys. 3.2c).

Jeżeli podczas rejestracji wystąpią błędy, użytkownik zostanie o tym powiadomiony za pomocą wyskakującego okienka z komunikatem. W przypadku Jacka proces przebiega pomyślnie, i zostaje on przekierowany na stronę z listą polecanych przewodników (3.2d). Tutaj może wybrać interesujący go przewodnik dotykając go na ekranie swojego urządzenia.

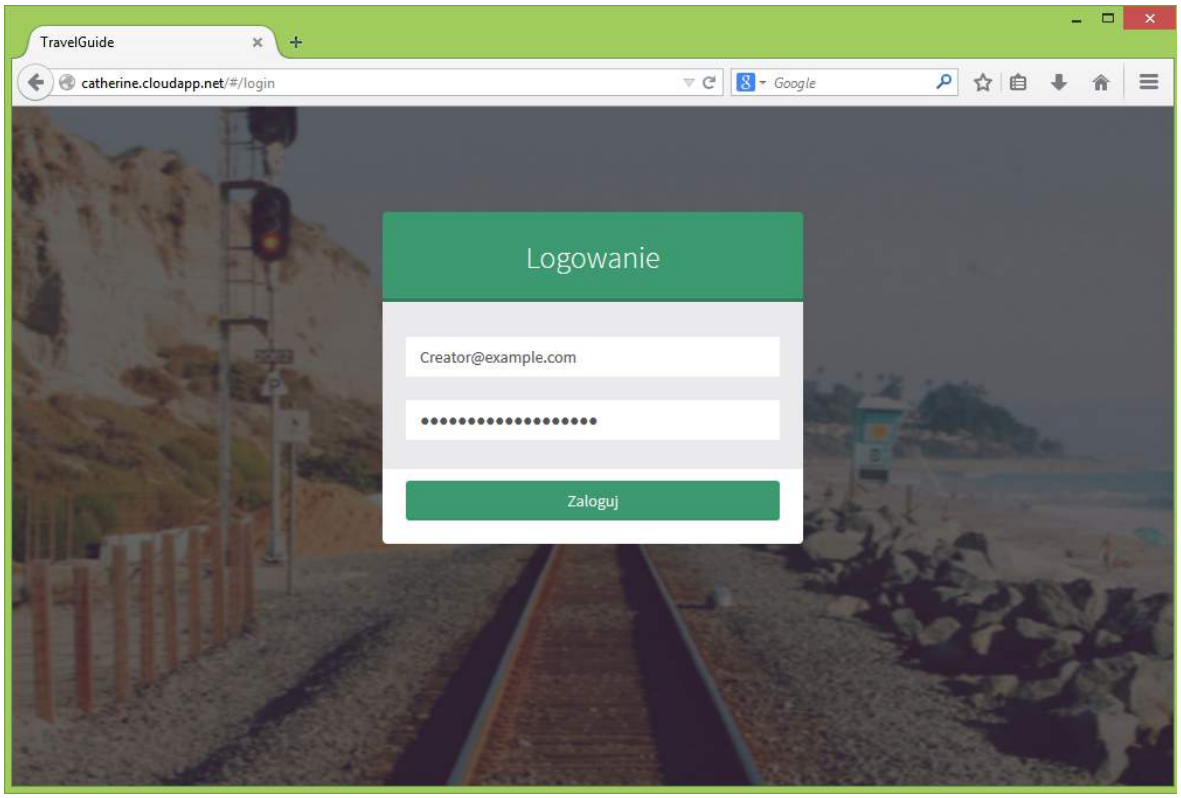
Jacek wybiera przewodnik po Lizbonie i zostaje przekierowany do widoku pobierania (rys. 3.2e). W momencie, kiedy naciska zielony przycisk „Pobierz", aplikacja rozpoczyna pobieranie danych w tle. W tym czasie użytkownik zostaje poinformowany, że aby w pełni używać programu w trybie *offline*, powinien również pobrać mapy dla danego miejsca. Po zamknięciu okienka, Jacek zostaje ponownie przekierowany do widoku przewodników. Tym razem na liście widać świeżo pobrany przewodnik o Lizbonie (rys. 3.2f).

Podróżnik dotyka miniaturki Lizbony i przechodzi do właściwej części przewodnika (rys. 3.2g). W tej sekcji Jacek może poczytać informacje dotyczące miasta (rys. 3.2h). Między kolejnymi elementami użytkownik może poruszać się przeciągając palcem po ekranie lub korzystając z przycisków pod mapą.

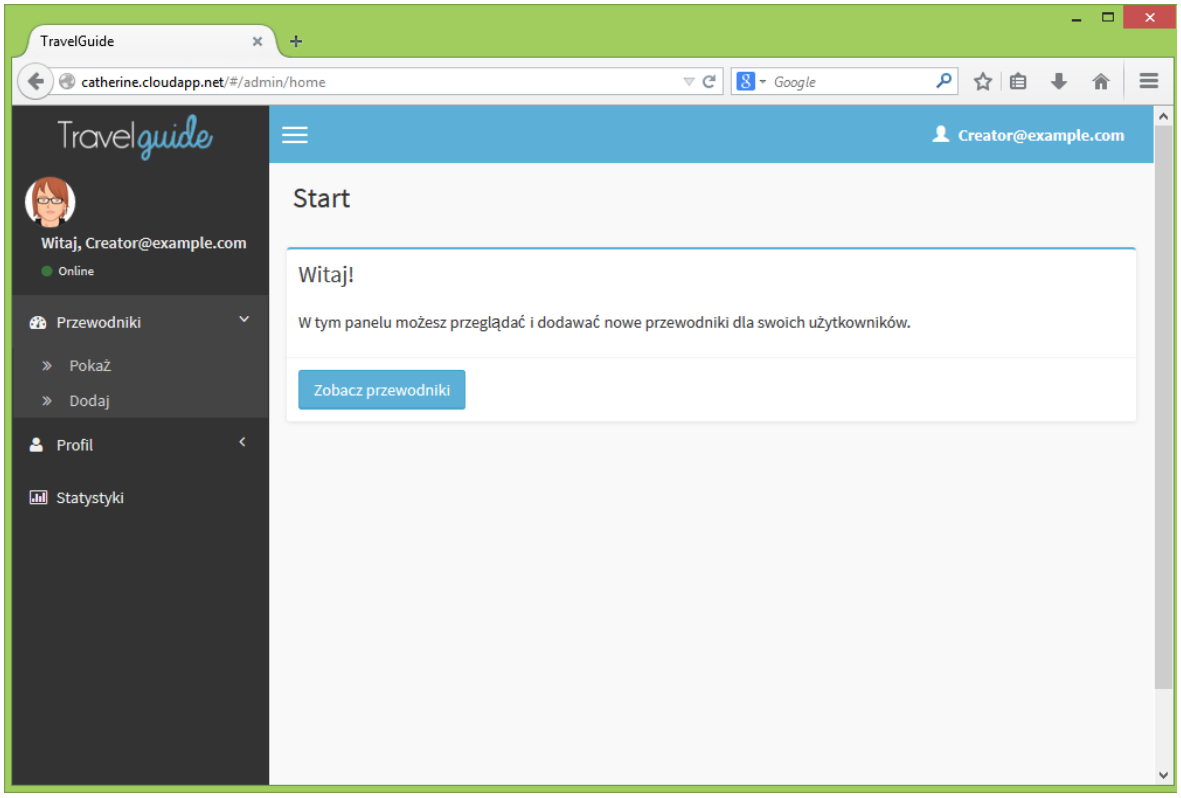
Zapoznawszy się z informacjami, Jacek chce dowiedzieć się o ciekawych miejscach w okolicy. Dotyka więc obrazek mapy i zostaje przeniesiony do prawdziwej, interaktywnej mapy z oznaczonymi punktami charakterystycznymi (rys. 3.2i). Po dotknięciu dowolnego z punktów wyświetlona zostaje chmurka z nazwą obiektu i odnośnikiem do szczegółów (rys. 3.2j).

Jacek dotyka zielony przycisk ze strzałką i zostaje przeniesiony do strony poświęconej danemu obiektowi (rys. 3.2k). Tu może dowiedzieć się więcej o miejscu oraz zobaczyć miniaturkę zdjęcia.

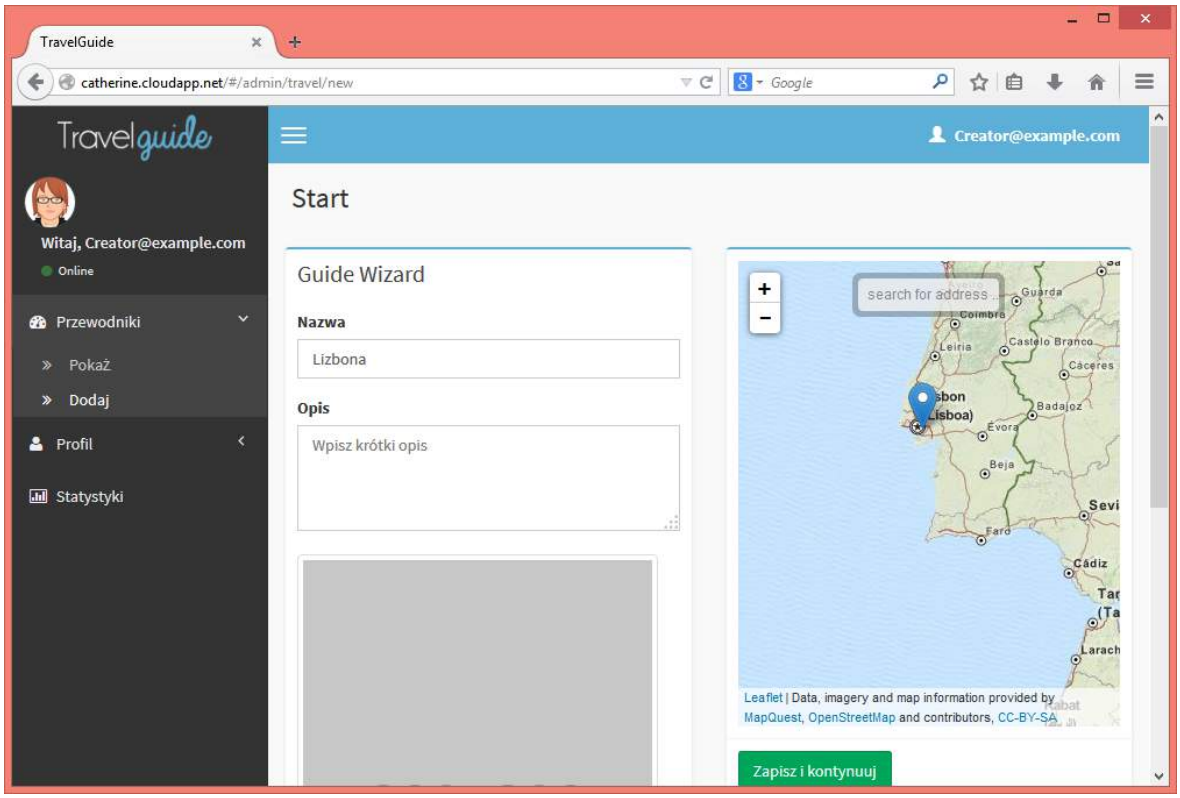
Powrót do poprzednich widoków odbywa się w sposób standardowy z obsługą systemu Windows Phone 8.1; za pomocą fizycznego przycisku wstecz.


TravelGuide
catherine.cloudapp.net/#/login
Google
Logowanie
Creator@example.com
Zaloguj

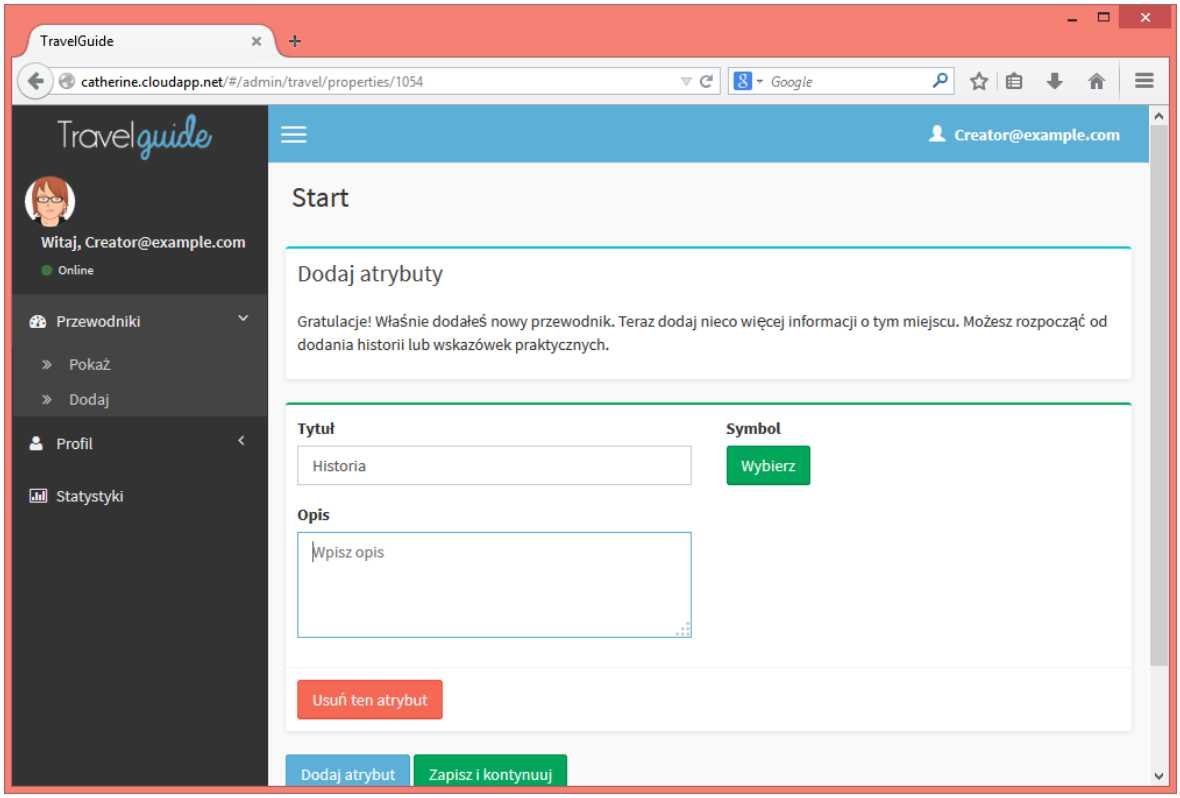
(a)


TravelGuide
catherine.cloudapp.net/#/admin/home
Google
Travelguide
Creator@example.com
Witaj, Creator@example.com
Online
Start
Przewodniki
Pokaż
Dodaj
Profil
Statystyki
Witaj!
W tym panelu możesz przeglądać i dodawać nowe przewodniki dla swoich użytkowników.
Zobacz przewodniki

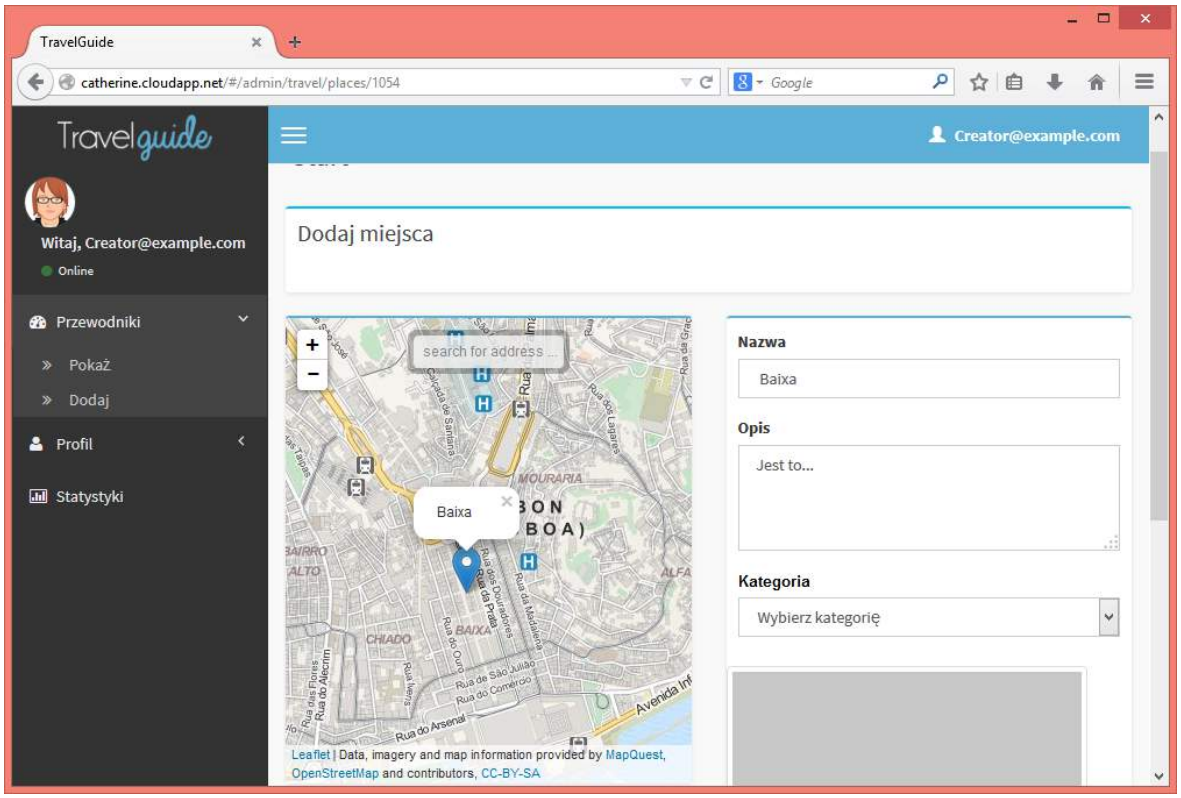
(b)


TravelGuide
catherine.cloudapp.net/#/admin/travel/new
Travelguide
Creator@example.com
Witaj, Creator@example.com
Online
Przewodniki
Pokaż
Dodaj
Profil
Statystyki
Start
Guide Wizard
Nazwa
Lizbona
Opis
Wpisz krótki opis
search for address
Leaflet | Data, imagery and map information provided by MapQuest, OpenStreetMap and contributors, CC-BY-SA
Zapisz i kontynuuj

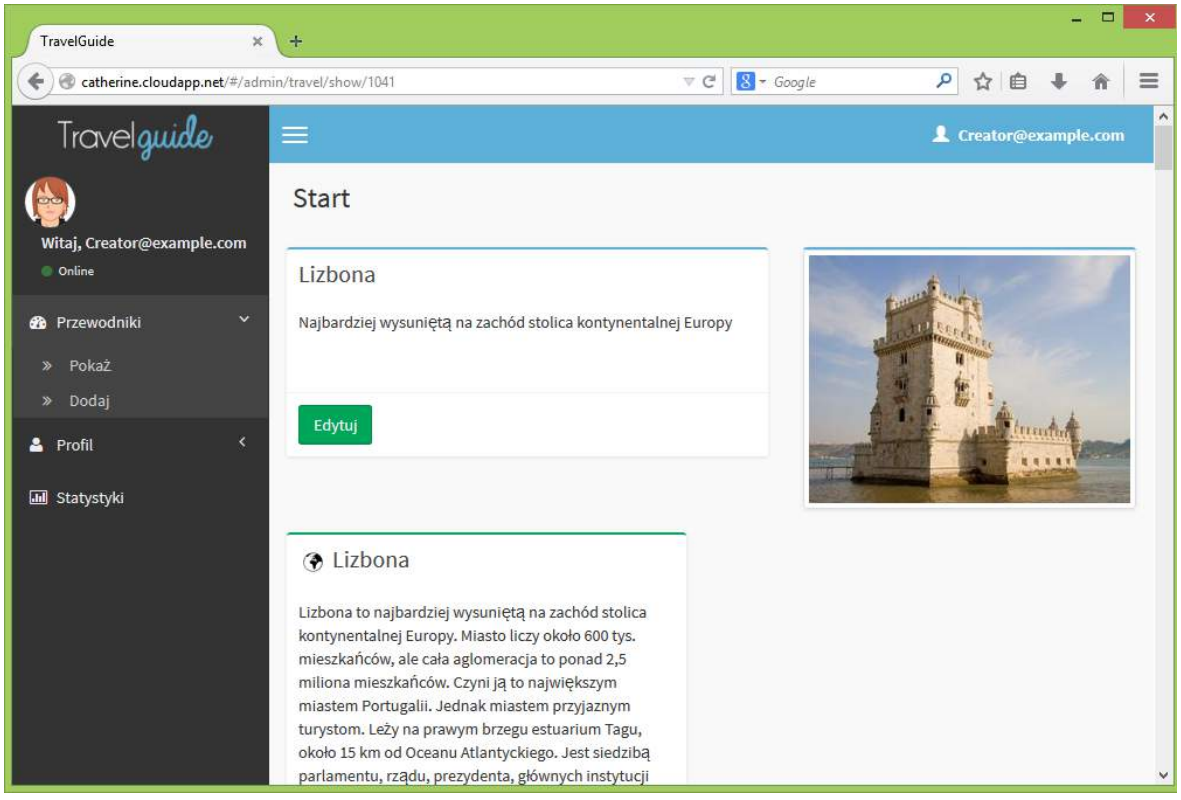
(c)


TravelGuide
catherine.cloudapp.net/#/admin/travel/properties/1054
Travelguide
Creator@example.com
Witaj, Creator@example.com
Online
Przewodniki
Pokaż
Dodaj
Profil
Statystyki
Start
Dodaj atrybuty
Gratulacje! Właśnie dodałeś nowy przewodnik. Teraz dodaj nieco więcej informacji o tym miejscu. Możesz rozpocząć od dodania historii lub wskazówek praktycznych.
Tytuł
Historia
Symbol
Wybierz
Opis
Wpisz opis
Usuń ten atrybut
Dodaj atrybut
Zapisz i kontynuuj

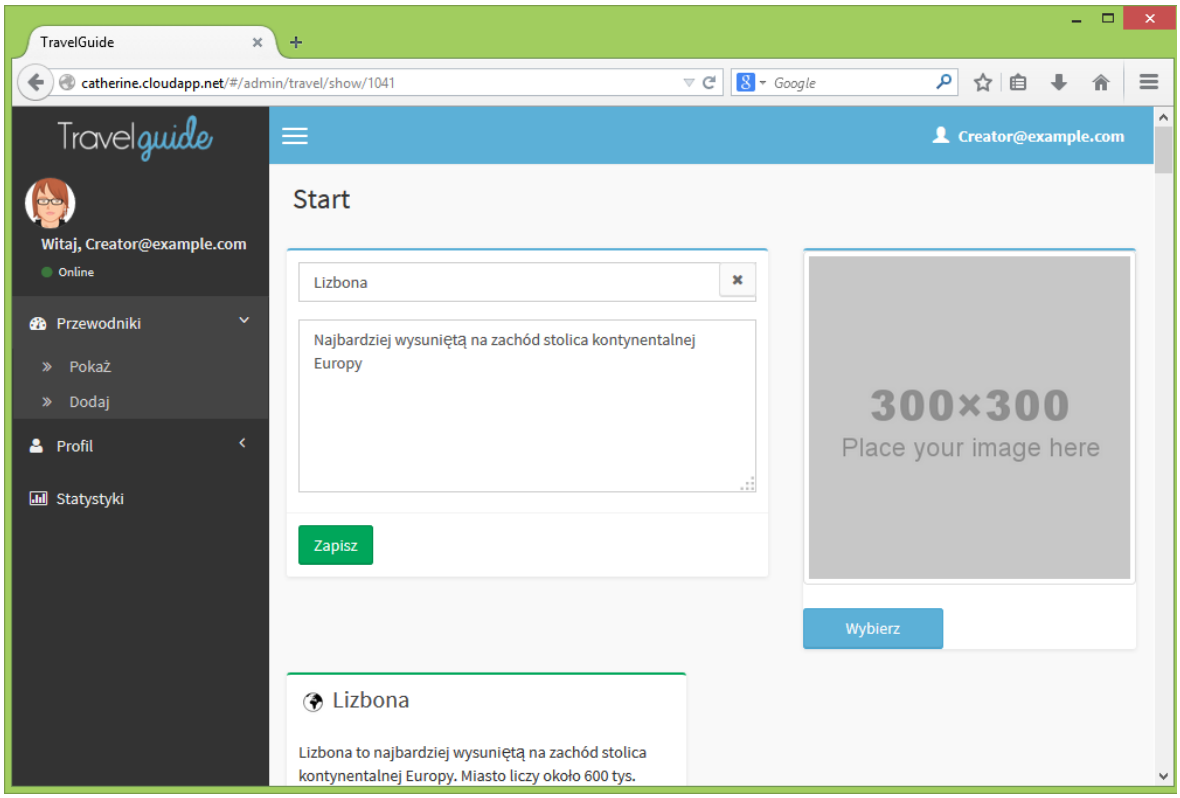
(d)


TravelGuide
catherine.cloudapp.net/#/admin/travel/places/1054
Travelguide
Creator@example.com
Witaj, Creator@example.com
Online
Przewodniki
Pokaż
Dodaj
Profil
Statystyki
Dodaj miejsca
search for address ...
Baixa
Nazwa
Baixa
Opis
Jest to...
Kategoria
Wybierz kategorię
Leaflet | Data, imagery and map information provided by MapQuest, OpenStreetMap and contributors, CC-BY-SA

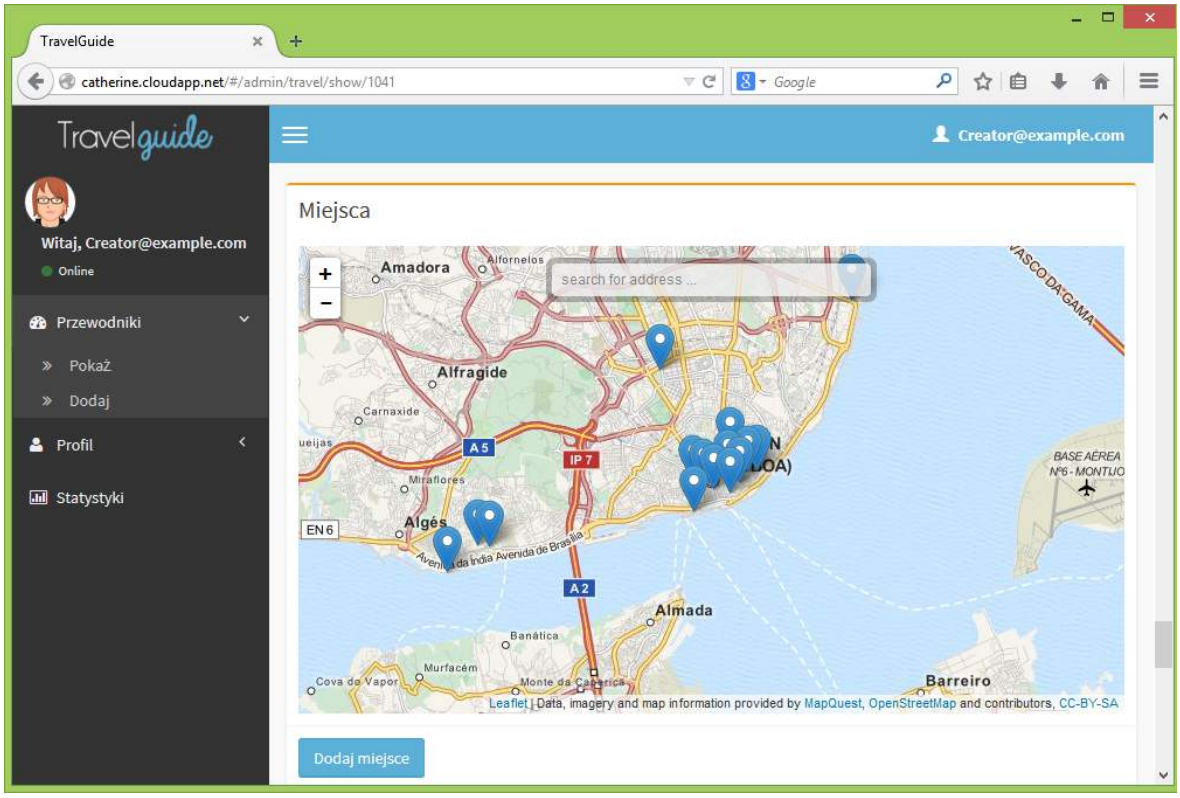
(e)


TravelGuide
catherine.cloudapp.net/#/admin/travel/show/1041
Travelguide
Creator@example.com
Witaj, Creator@example.com
Online
Przewodniki
Pokaż
Dodaj
Profil
Statystyki
Start
Lizbona
Najbardziej wysuniętą na zachód stolica kontynentalnej Europy
Edytuj
Lizbona
Lizbona to najbardziej wysuniętą na zachód stolica kontynentalnej Europy. Miasto liczy około 600 tys. mieszkańców, ale cała aglomeracja to ponad 2,5 miliona mieszkańców. Czyni ją to największym miastem Portugalii. Jednak miastem przyjaznym turystom. Leży na prawym brzegu estuarium Tagu, około 15 km od Oceanu Atlantyckiego. Jest siedzibą parlamentu, rządu, prezydenta, głównych instytucji

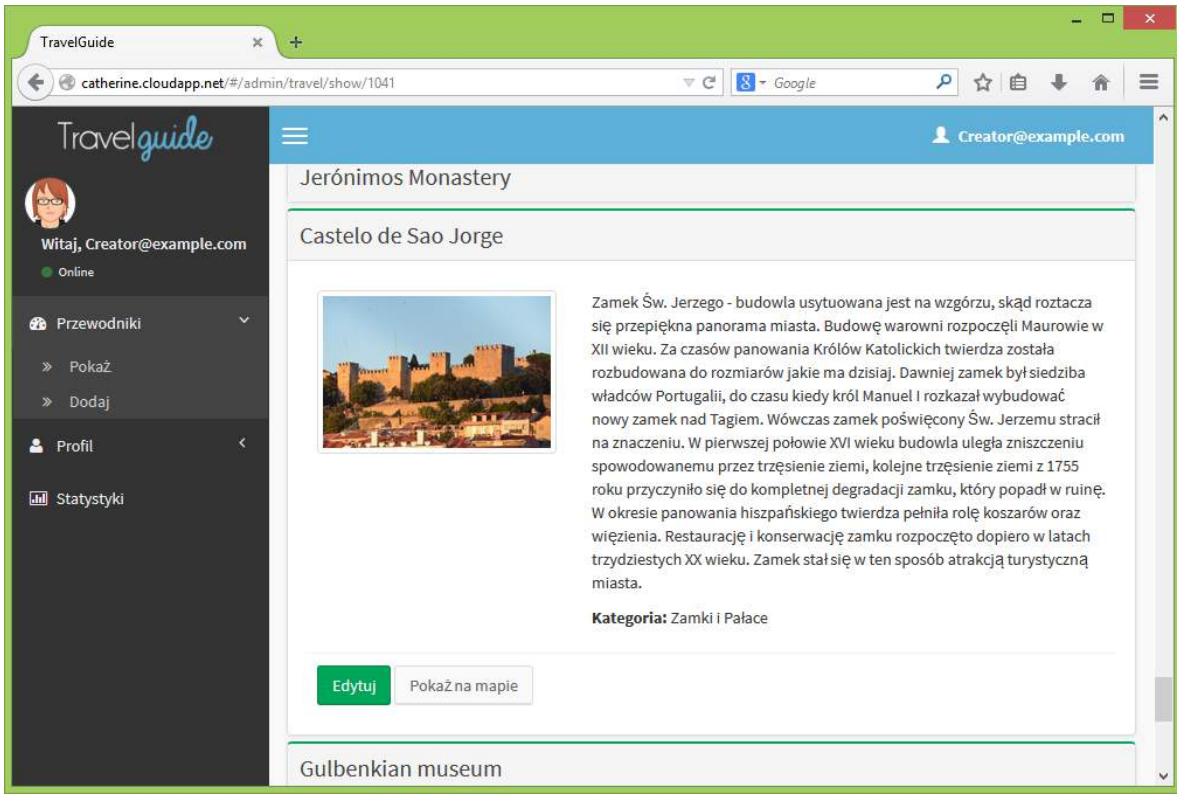
(f)


TravelGuide
catherine.cloudapp.net/#/admin/travel/show/1041
Travelguide
Creator@example.com
Witaj, Creator@example.com
Online
Przewodniki
Pokaż
Dodaj
Profil
Statystyki
Start
Lizbona
Najbardziej wysuniętą na zachód stolica kontynentalnej Europy
Zapisz
300×300
Place your image here
Wybierz
Lizbona
Lizbona to najbardziej wysuniętą na zachód stolica kontynentalnej Europy. Miasto liczy około 600 tys.

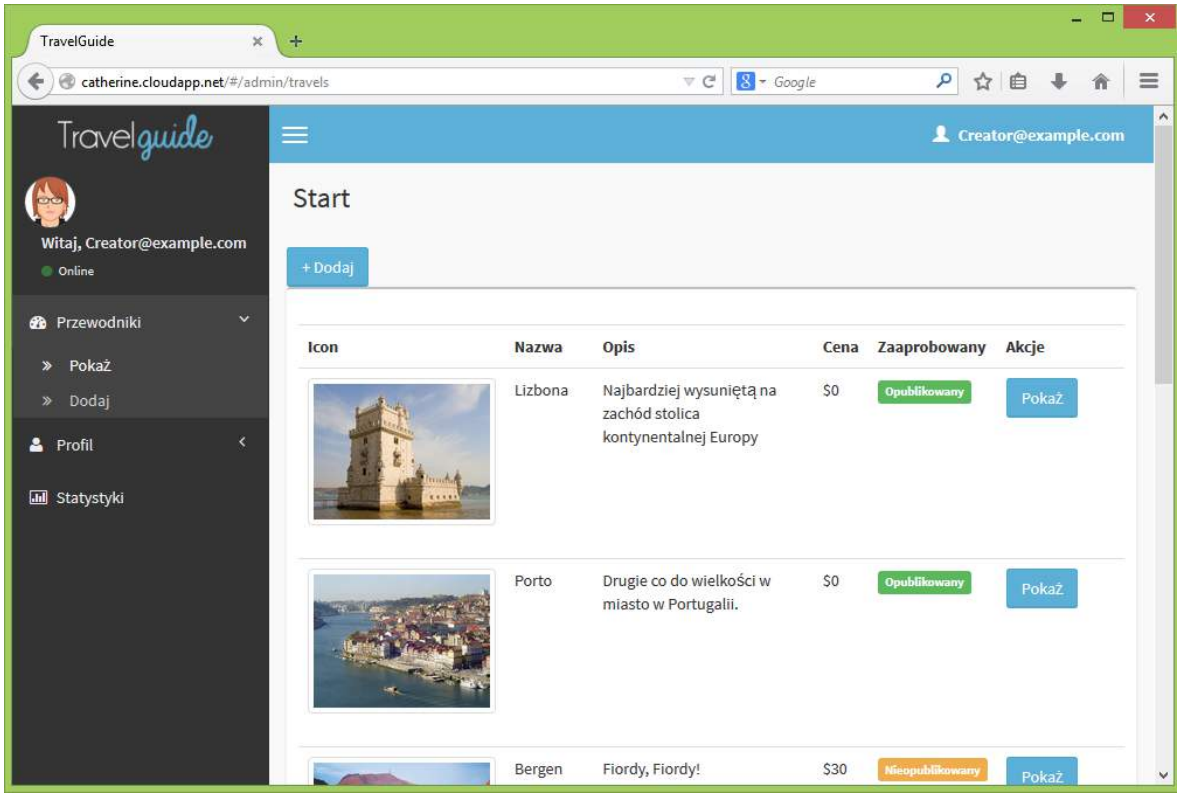
(g)


TravelGuide
catherine.cloudapp.net/#/admin/travel/show/1041
Travelguide
Creator@example.com
Witaj, Creator@example.com
Online
Przewodniki
Pokaż
Dodaj
Profil
Statystyki
Miejsca
search for address ...
Amadora
Alfragide
Carnaxide
Miraflores
Algés
Almada
Barreiro
Banática
Murfacém
Cova do Vapor
A5
IP 7
A2
EN 6
Leaflet | Data, imagery and map information provided by MapQuest, OpenStreetMap and contributors, CC-BY-SA
Dodaj miejsce

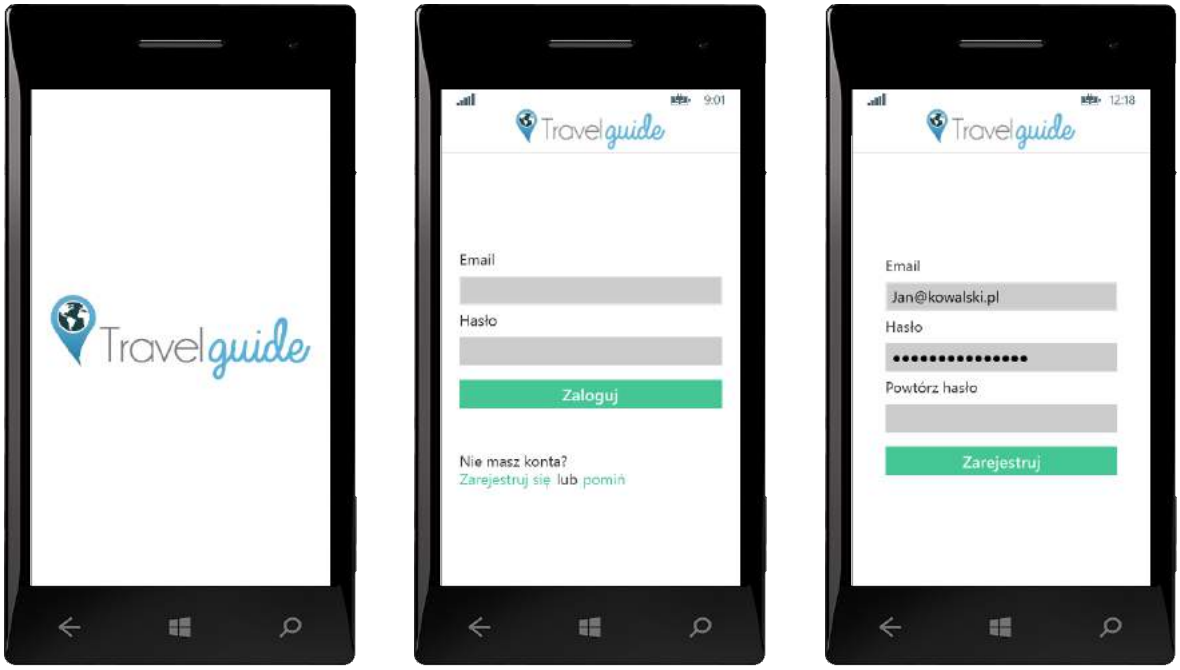
(h)


TravelGuide
catherine.cloudapp.net/#/admin/travel/show/1041
Google
Travelguide
Creator@example.com
Witaj, Creator@example.com
Online
Przewodniki
Pokaż
Dodaj
Profil
Statystyki
Jerónimos Monastery
Castelo de Sao Jorge
Zamek Św. Jerzego - budowla usytuowana jest na wzgórzu, skąd roztacza się przepiękna panorama miasta. Budowę warowni rozpoczęli Maurowie w XII wieku. Za czasów panowania Królów Katolickich twierdza została rozbudowana do rozmiarów jakie ma dzisiaj. Dawniej zamek był siedziba władców Portugalii, do czasu kiedy król Manuel I rozkazał wybudować nowy zamek nad Tagiem. Wówczas zamek poświęcony Św. Jerzemu stracił na znaczeniu. W pierwszej połowie XVI wieku budowla uległa zniszczeniu spowodowanemu przez trzęsienie ziemi, kolejne trzęsienie ziemi z 1755 roku przyczyniło się do kompletnej degradacji zamku, który popadł w ruinę. W okresie panowania hiszpańskiego twierdza pełniła rolę koszarów oraz więzienia. Restaurację i konserwację zamku rozpoczęto dopiero w latach trzydziestych XX wieku. Zamek stał się w ten sposób atrakcją turystyczną miasta.
Kategoria: Zamki i Pałace
Edytuj
Pokaż na mapie
Gulbenkian museum

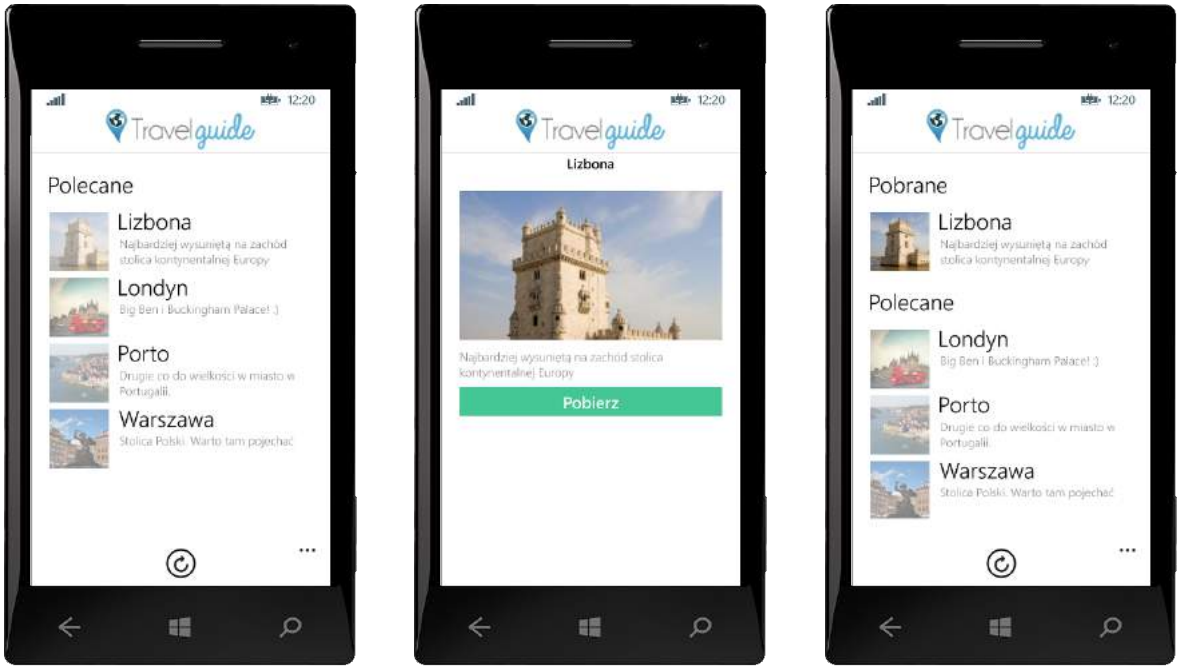
(i)


TravelGuide
catherine.cloudapp.net/#/admin/travels
Google
Travelguide
Creator@example.com
Witaj, Creator@example.com
Online
Przewodniki
Pokaż
Dodaj
Profil
Statystyki
Start
+ Dodaj
Icon
Nazwa
Opis
Cena
Zaaprobowany
Akcje
Lizbona
Najbardziej wysuniętą na zachód stolica kontynentalnej Europy
$0
Opublikowany
Pokaż
Porto
Drugie co do wielkości w miasto w Portugalii.
$0
Opublikowany
Pokaż
Bergen
Fiordy, Fiordy!
$30
Nieopublikowany
Pokaż

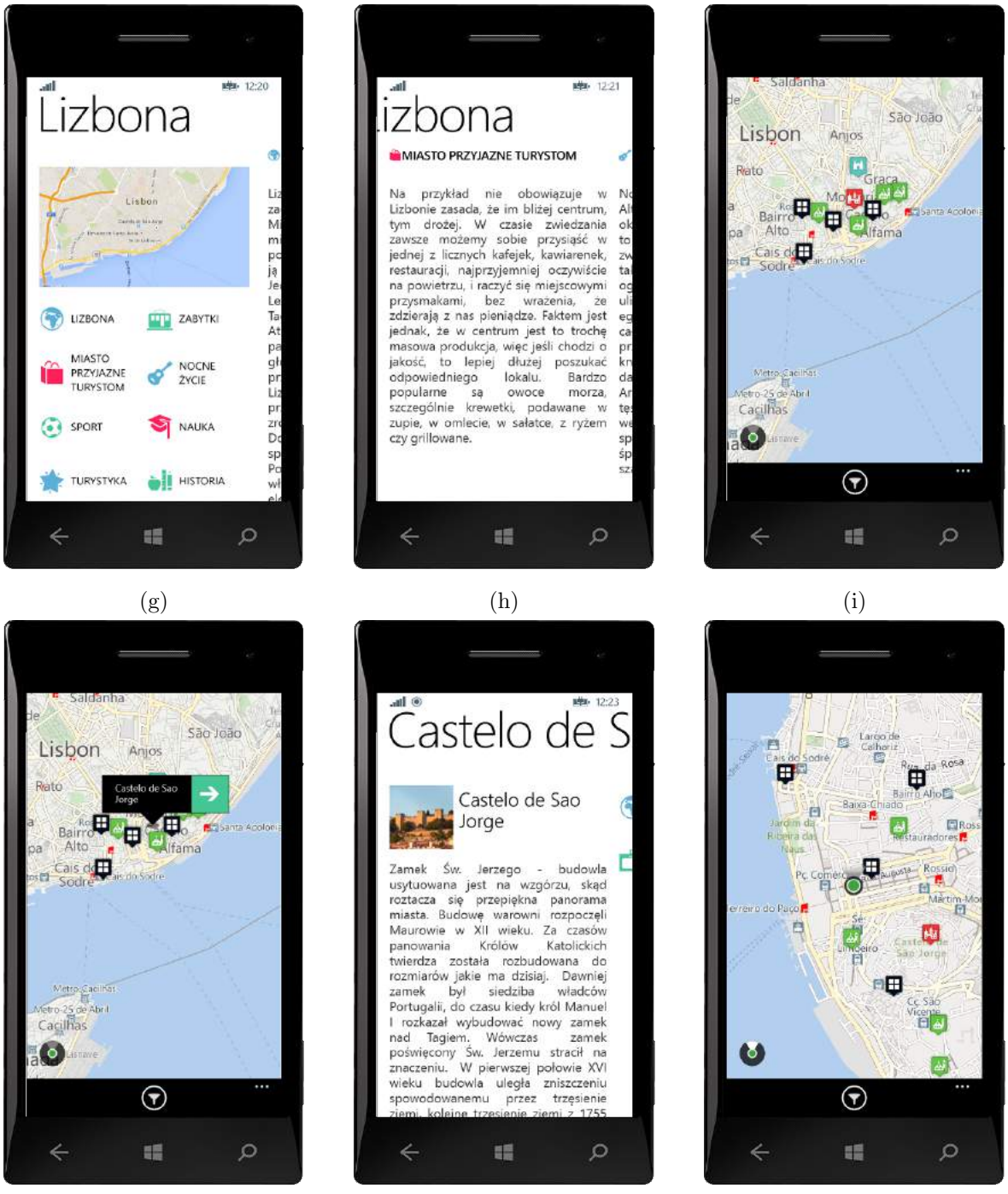
(j)


Travelguide
Email
Hasło
Zaloguj
Nie masz konta?
Zarejestruj się lub pomiń
Email
Jan@kowalski.pl
Hasło
Powtórz hasło
Zarejestruj

(a) (b) (c)


Travelguide
Polecane
Lizbona
Londyn
Porto
Warszawa
Lizbona
Najbardziej wysuniętą na zachód stolica kontynentalnej Europy
Pobierz
Pobrane
Lizbona
Polecane
Londyn
Porto
Warszawa

(d) (e) (f)



(g) (h) (i)

(j) (k) (l)

Rysunek 3.2

Rozdział 4

Specyfikacja wewnętrzna

4.1 Przypadki użycia

Zarówno aplikacja internetowa jak i mobilna przeznaczona jest dla użytkownika końcowego, jednak ta pierwsza służy użytkownikowi-twórcy, a ta druga użytkownikowi-podróżnikowi. Twórca używa aplikacji w następujący sposób: dodaje przewodnik oraz dodaje jego części składowe: podstawowe informacje, miejsca, atrybuty. Następnie może opublikować przewodnik. Twórca ma możliwość także edycji stworzonego przez siebie przewodnika; może edytować jego podstawowe informacje, może edytować, dodawać i usuwać miejsca, atrybuty.

Użytkownik-podróżnik może pobierać przewodniki, a następie je odczytywać i wykorzystywać. W skład tych akcji wchodzi odczyt atrybutów i miejsc, a także lokalizowanie położenia tych miejsc i swojego położenia dzięki wbudowanemu w telefon modułowi GPS.

Diagram 4.1) przedstawia najważniejsze przypadki użycia aplikacji klienckich. Pominięta została m.in. funkcja logowania dostępna dla obu aplikacji oraz funkcja rejestracji użytkownika dostępna z poziomu aplikacji mobilnej.

4.2 Architektura

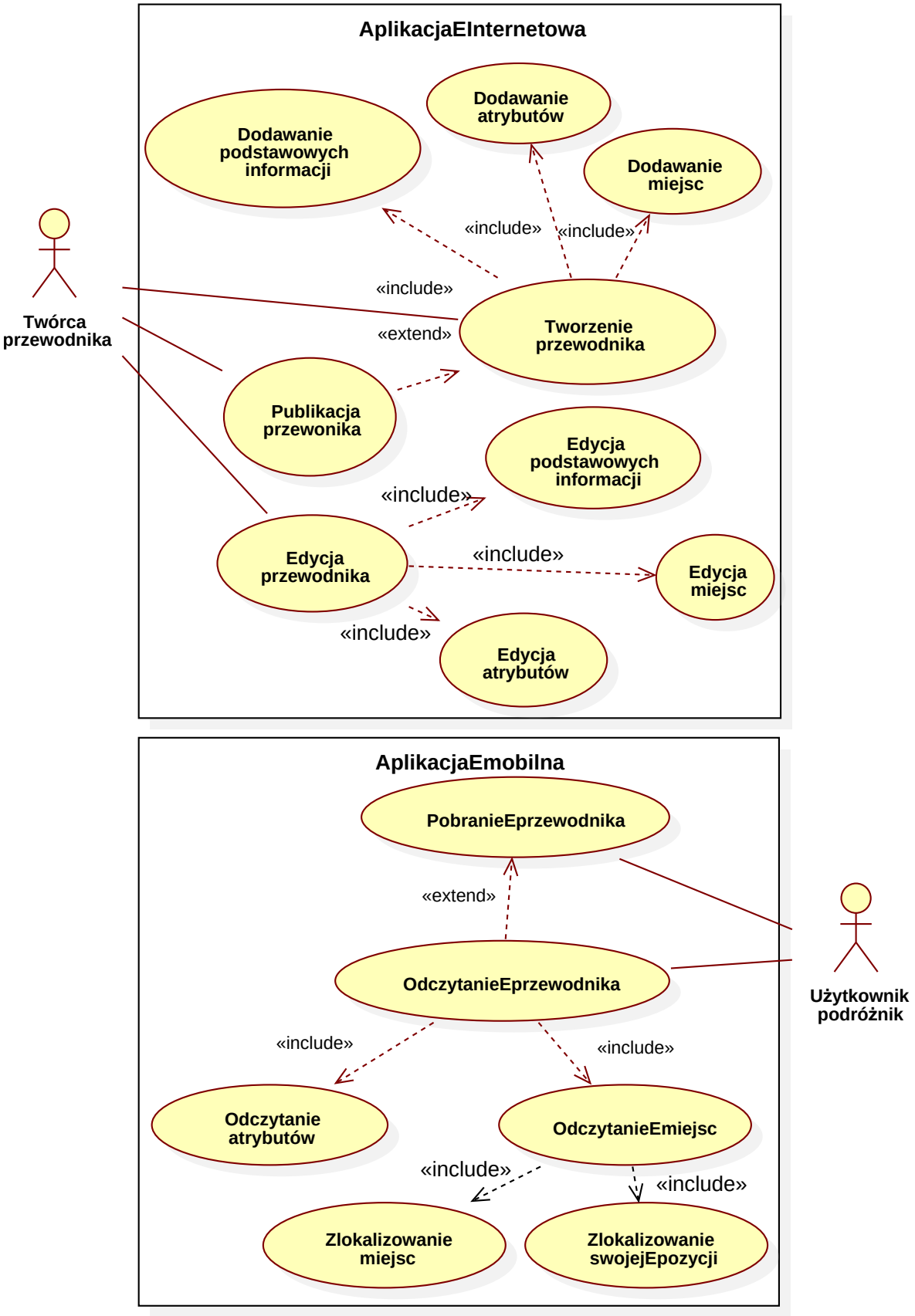
Na cały system składają się trzy aplikacje: aplikacja mobilna (*Mobile app*), aplikacja internetowa (*Website*) oraz aplikacja serwerowa (*WebService*). Aplikacja serwerowa jest odpowiedzialna za komunikację pomiędzy główną bazą danych a aplikacjami klienckimi. Dostarcza ona API[1]. Związek pomiędzy tymi aplikacjami przedstawiony jest na rysunku 4.2.

4.2.1 Architektura aplikacji serwerowej

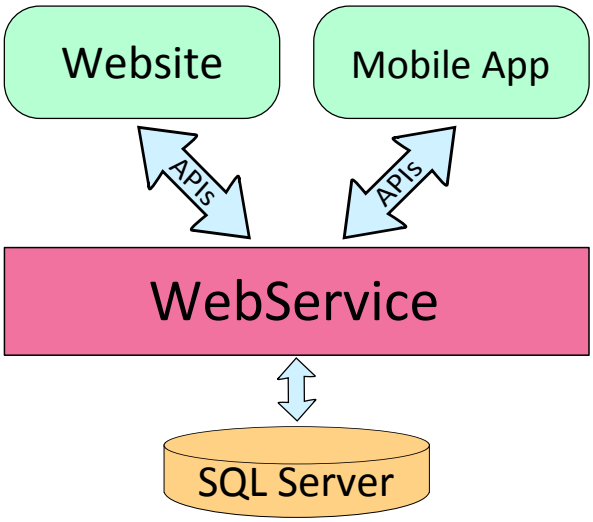
Aplikacja serwerowa stworzona jest przy wykorzystaniu platformy ASP.NET WebApi 2 i zgodnie ze wzorcem architektonicznym REST[2]. Wzorzec ten zakłada kilka warunków:

[1] Interfejs programistyczny aplikacji (ang. *Application Programming Interface*) – sposób, w jaki programy komunikują się między sobą.

[2] *Representational State Transfer*



Rysunek 4.1: Diagram przypadków użycia



Rysunek 4.2: Architektura systemu

Klient-serwer – aplikacja powinna składać się z dwóch odrębnych części: klienta i serwera. Klient nie powinien mieć dostępu do magazynu danych. Serwer z kolei nie powinien zawierać informacji na temat interfejsu użytkownika. I tak też aplikacja serwerowa systemu *TravelGuide* jest jedyną, która ma dostęp do bazy danych. Nie zawiera ona żadnych widoków, a ze światem komunikuje się za pomocą zapytań HTTP[3].

Bezstanowość – każde żądanie HTTP (*request*) wysyłane do serwera jest niezależne od pozostałych. I tak też w aplikacji serwerowej systemu *TravelGuide* za odbiór żądań odpowiada warstwa kontrolera. Każda akcja kontrolera obsługuje jedno żądanie. Akcje są od siebie niezależne.

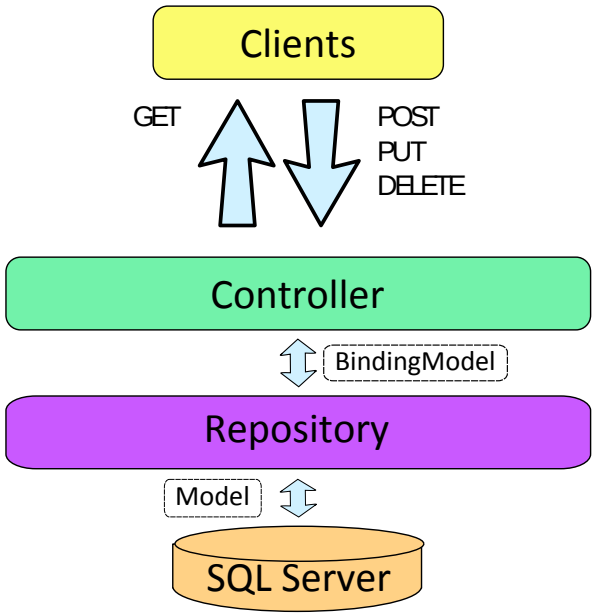
Wielowarstwowość – klient nie wie, czy jest podłączony bezpośrednio do serwera, czy też otrzymuje informacje przez systemy pośrednie. Dzięki takiej skalowalności, systemy pośrednie mogą zostać wykorzystane, aby zwiększyć wydajność całości. Aplikacja serwerowa systemu *TravelGuide* przyjmuje żądania HTTP a odpowiedź wysyła w formacie JSON. Można więc dołożyć kolejne systemu pośrednie zgodnie z zapotrzebowaniem.

Lokalność [4] – serwer powinien zwracać informację, które z odpowiedzi mogą zostać zapisane w pamięci podręcznej (*cache*) przeglądarki. W przypadku aplikacji serwerowej systemu *TravelGuide* nie jest to zaimplementowane, gdyż żadna z odpowiedzi nie powinna być zapisywana.

Wewnętrznie aplikacja składa się z wielu warstw, które komunikują się ze sobą. Każde odebrane zapytanie HTTP jest obsługiwane przez odpowiednią akcję kontrolera. Zapytania realizowane są czterema standardowymi metodami HTTP: *GET*, *POST*, *PUT*, *DELETE*.

Zazwyczaj akcja wymaga interakcji z bazą danych. Kontroler nie komunikuje się z nią jednak bezpośrednio. Zamiast tego składa odpowiednio sparametryzowane zapytanie do war-

[3] *Hypertext Transfer Protocol* – protokół przesyłania dokumentów hipertekstowych
[4] ang. *cachable*



Rysunek 4.3: Architektura aplikacji serwerowej

stwy repozytorium. Parametrem może być cyfra (`Integer`), ciąg znaków (`String`) lub obiekt typu modelu wiążącego (*Binding Model*). Parametr może być także pusty.

Repozytorium przetwarza żądanie. Jeżeli parametrem był obiekt typu modelu wiążącego mapuje go na model, a następnie za pomocą *Entity Framework* komunikuje się z bazą danych. Rezultat jest następnie ponownie mapowany na model wiążący i przekazywany do akcji kontrolera. Kontroler zwraca wynik w postaci odpowiedzi HTTP (*HTTP Response*).

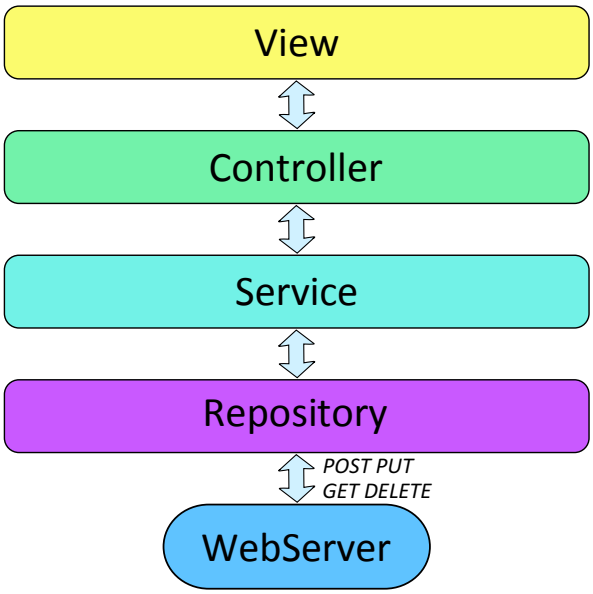
Schemat architektury aplikacji serwerowej przedstawiony jest na rysunku 4.3.

4.2.2 Architektura aplikacji internetowej

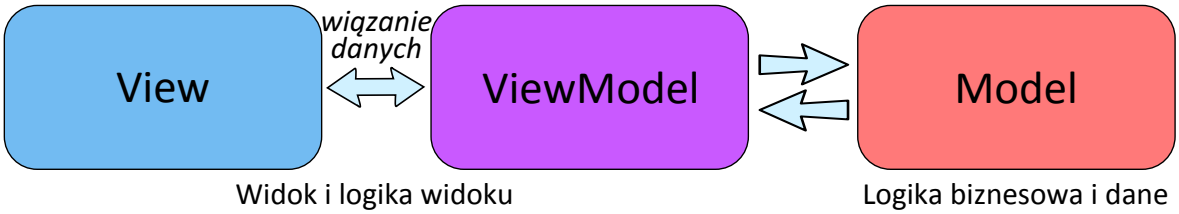
Aplikacja internetowa została stworzona przy użyciu platformy AngularJS i w całości działa bez przeładowania strony (*Single Page Application*). Zawiera główny widok napisany w języku HTML, do którego w zależności od kontekstu załączane są widoki cząstkowe (*partial views*). Z każdym widokiem powiązany jest kontroler.

Kiedy użytkownik wchodzi w interakcję z aplikacją, jego akcje są wysyłane do odpowiednich akcji kontrolera. Kontroler następnie obsługuje daną operację lub, jeżeli jest ona bardziej skomplikowana, odwołuje się do odpowiedniej metody z warstwy serwisu. Jeżeli wymagane jest pobranie danych z WebService, warstwa serwisu komunikuje się z warstwą repozytorium, która następnie wysyła odpowiednio skonstruowane zapytanie. Otrzymany wynik jest przekazywany z powrotem do warstwy serwisu, przetwarzany, przekazywany do kontrolera i wiązany (*Bind*) z widokiem.

Schemat architektury aplikacji internetowej przedstawiony jest na rysunku 4.4.



Rysunek 4.4: Architektura aplikacji internetowej



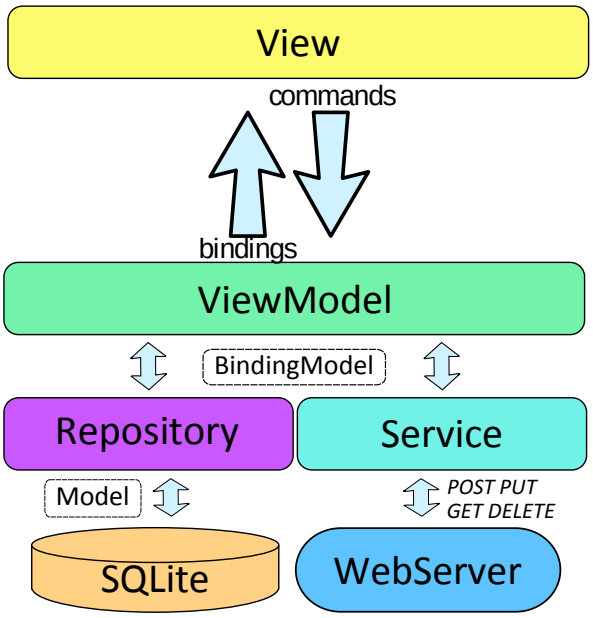
Rysunek 4.5: Wzorzec MVVM

4.2.3 Architektura aplikacji mobilnej

Aplikacja mobilna dedykowana jest dla platformy Windows Phone 8.1. Została stworzona przy użyciu wzorca MVVM (*Model View ViewModel*). Wzorzec ten polega na rozdzieleniu warstwy widoku (*View*) od warstwy danych (*Model*). Są one połączone za pomocą warstwy pośredniej (*ViewModel*), która zawiera większą część logiki związanej z prezentacją danych (rys. 4.5). Chociaż w początkowym etapie projektu wprowadzenie go wymaga większego nakładu pracy to ma on bardzo wiele korzyści. W przeciwieństwie do utrzymywania kodu w *Code Behind*, gwarantuje niezależność logiki od sposobu wyświetlania danych, brak sztywnych powiązań między widokiem a logiką oraz niezależność kodu od technologii, w której wykonana jest warstwa prezentacji [31].

Program przy pierwszym uruchomieniu łączy się z aplikacją serwerową i pobiera dane, do których uprawniony jest zalogowany użytkownik. Dane te są zapisywane w lokalnej bazie danych obsługiwanej przez silnik SQLite.

Użytkownik wchodzi w interakcję z aplikacją przy pomocy zachowań typowych dla obsługi aplikacji mobilnych. Są to gesty takie jak dotknięcie ekranu, dłuższe przytrzymanie ekranu czy posuwisty ruch dłonią po ekranie smartfona. Po stronie aplikacji odbiorcą takich poleceń jest warstwa widoku (*View*). Komunikuje się ona następnie z warstwą pośrednią (*ViewModel*)



Rysunek 4.6: Architektura aplikacji mobilnej

wprost lub za pomocą konwerterów (*Converter*). Komunikacja odbywa się dzięki wykorzystaniu poleceń (*Commands*.

Warstwa pośrednia realizuje zadanie zadane przez użytkownika. Jeżeli wymaga to komunikacji z bazą danych, łączy się ona z warstwą repozytorium (*Repository*). Repozytorium operuje na dwóch typach obiektów: na modelach (*Model*) i na modelach wiążących (*Binding model*). Modele odwzorowują schemat bazy danych w postaci obiektów języka C# i są wykorzystywane w komunikacji z nią. Modele wiążące to modele zmodyfikowane na potrzeby danego zadania. Są one zmodyfikowane o dodatkowe pola, zbędne pola nie są zaś uwzględnione. Przykładowo, model wiążący wykorzystywany do rejestracji użytkownika stworzony jest na bazie modelu użytkownika. Nie uwzględnia on takich informacji jak „Data rejestracji" bądź „Konto Aktywne". Z drugiej strony jest uzupełniony o takie pole jak „Powtórz hasło".

Za komunikację z aplikacją serwerową odpowiada warstwa serwisów (*Service*), która wysyła do niej odpowiednie zapytania w formacie RESTful.

Warstwa pośrednia nie wie o istnieniu modeli. Od repozytorium otrzymuje ona jedynie kolekcje modeli wiążących. Finalnie są one przetwarzane i przekazywane do warstwy widoku za pomocą mechanizmu wiązania (*binding*).

Schemat architektury aplikacji mobilnej przedstawiony jest na rysunku 4.6.

4.3 Najważniejsze algorytmy

4.3.1 Wyznaczanie odległości między punktami na mapie

W celu wyznaczenia odległości pomiędzy miejscami A i B o współrzędnych geograficznych odpowiednio $A(\varphi_a, \lambda_a)$ i $B(\varphi_b, \lambda_b)$ obliczono długość ortodromy łączącej te punkty. Algorytm taki jest dość prosty: należy skonstruować trójkąt sferyczny biorąc jeden z biegunów. Z prawa

kosinusów dla trójkąta sferycznego wiemy [1]:

$$\cos l = \cos(\pi/2 - \varphi_a)\cos(\pi/2 - \varphi_b) + \sin(\pi/2 - \varphi_a)\sin(\pi/2 - \varphi_b)\cos(\lambda_a - \lambda_b), \tag{4.1}$$

gdzie l to kąt ortodromy, wzór można przekształcić

$$\cos l = \sin\varphi_a \sin\varphi_b + \cos\varphi_a \cos\varphi_b \cos(\lambda_a - \lambda_b). \tag{4.2}$$

Wyznaczenie długości d ortodromy jest proste:

$$d = R_{\oplus} l, \tag{4.3}$$

gdzie $R_{\oplus} \approx 6371{,}0\,\text{km}$ to średni promień Ziemi. Uwzględniając równanie (4.2) otrzymujemy

$$d = R_{\oplus} \arccos\left(\sin\varphi_a \sin\varphi_b + \cos\varphi_a \cos\varphi_b \cos(\lambda_a - \lambda_b)\right), \tag{4.4}$$

Z tego względu, że dla blisko położonych punktów mogą pojawić się problemy numeryczne (co w dzisiejszych czasach jest już rzadsze), można wzór (4.4) przekształcić do łatwiejszej obliczeniowo postaci

$$d = 2R_{\oplus} \arcsin\sqrt{\sin^2\left(\frac{\varphi_a - \varphi_b}{2}\right) + \cos\varphi_a \cos\varphi_b \sin^2\left(\frac{\lambda_a - \lambda_b}{2}\right)}. \tag{4.5}$$

Oczywiście trzeba pamiętać, że rachunki powyższe nie uwzględniają tego, że Ziemia nie jest kulą, dlatego są obarczone pewnym błędem. Jednakże taki błąd jest dopuszczalny w przypadku aplikacji *TravelGuide*.

Implementacja funkcji w projekcie w języku C# została przedstawiona na listingu 4.1.

Listing 4.1: Funkcja obliczająca dystans pomiędzy pozycją użytkownika a punktem na mapie

```
1
2     public void SetDistance(Geoposition currentPosition)
3     {
4       const double degreesToRadians = (Math.PI / 180.0);
5       const double earthRadius = 6371; // kilometers
6
7       // convert latitude and longitude values to radians
8       double prevRadLat = _placeLatitude * degreesToRadians;
9       double prevRadLong = _placeLongitude * degreesToRadians;
10      double currRadLat = currentPosition.Coordinate.Point.Position.
            Latitude * degreesToRadians;
11      double currRadLong = currentPosition.Coordinate.Point.Position.
            Longitude * degreesToRadians;
12
13      // calculate ortodroma
14      var exp1 = Math.Pow((Math.Sin((prevRadLat - currRadLat) / 2), 2);
15      var exp2 = Math.Cos(prevRadLat)*Math.Cos(currRadLat)*Math.Pow(Math.
            Sin((prevRadLong-currRadLong)/2) ,2);
```

```
16        var ortodroma = 2 * earthRadius * Math.Asin(Math.Sqrt(exp1+exp2));
17
18        // calculate radian delta between each position.
19        this.Distance = ortodroma * 1000;  // return results as meters
20    }
```

4.3.2 Pobieranie danych w aplikacji mobilnej

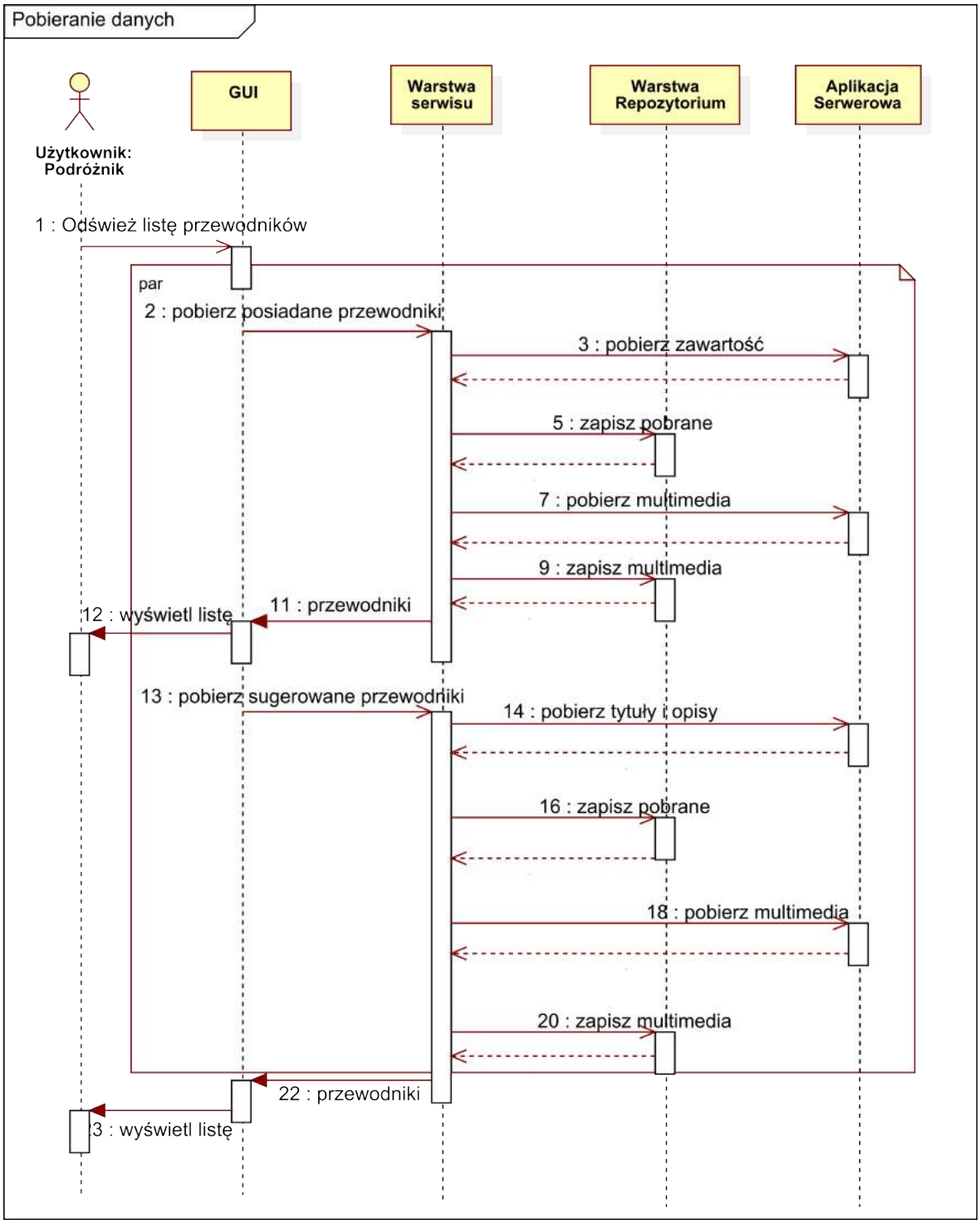
W celu zoptymalizowania pobierania danych zastosowano algorytmy wykonywane równolegle. Gdy użytkownik zażąda pobrania danych bądź też aplikacja jest uruchomiana po raz pierwszy, do warstwy serwisów przekazane jest polecenie pobrania danych. Wysyła ona równolegle dwa zapytania do aplikacji serwerowej: o przewodniki posiadane przez użytkownika oraz o listę elementów rekomendowanych do pobrania. Gdy zapytanie zwróci odpowiednie dane, są one od razu zapisywane do lokalnej bazy danych za pośrednictwem warstwy repozytorium oraz wyświetlane jako lista użytkownikowi.

W tym samym czasie aplikacja mobilna odpytuje aplikację serwerową raz jeszcze, tym razem w celu pobrania odpowiednich multimediów. Gdy pobieranie zostanie zakończone, obrazki są zapisane do lokalnej pamięci i wyświetlone w programie.

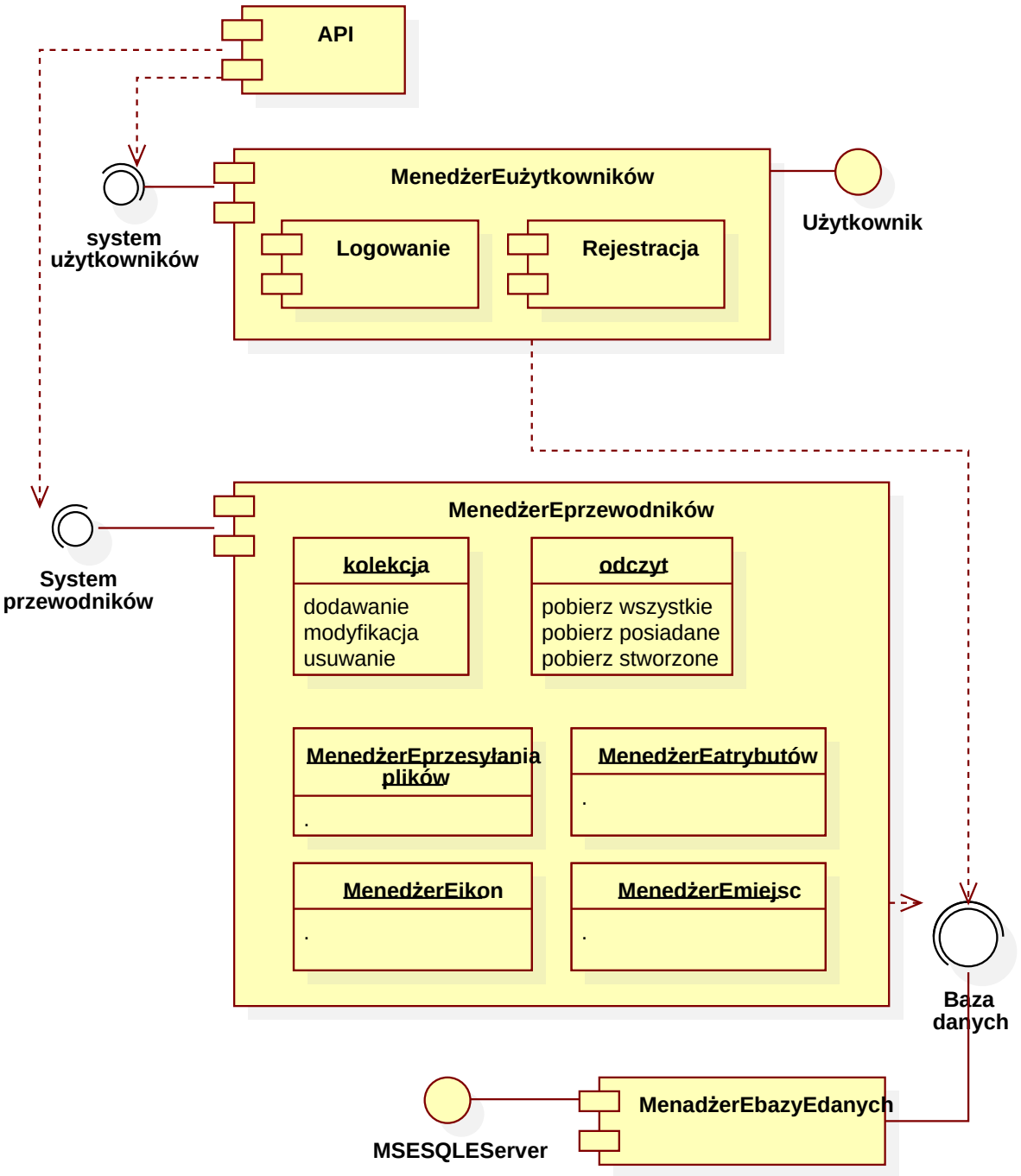
Diagram 4.7 przedstawia interakcje, jakie zachodzą pomiędzy poszczególnymi komponentami.

4.4 Komponenty

Aplikacja serwerowa składa się z kilku komponentów. Do tych głównych należą menedżer użytkowników oraz menedżer przewodników. Menedżer użytkowników odpowiedzialny jest za obsługę logowania oraz rejestracji. Kontroluje on również, czy żądanie HTTP jest odpowiednio autoryzowane i czy użytkownik jest uprawniony do wykonania zadanej akcji. Menedżer przewodników służy do zarządzania przewodnikami oraz ich częściami składowymi (miejscami, atrybutami, ikonami etc.). Koordynuje on zarówno proces dodawania, modyfikacji i usuwania, ale także pobieranie przewodników, do których użytkownik jest uprawniony. Diagram 4.8 przedstawia komponenty aplikacji serwerowej.



Rysunek 4.7: Diagram interakcji



Rysunek 4.8: Diagram komponentów aplikacji serwerowej.

Rozdział 5

Testowanie

Testowanie aplikacji odbywało się w kilku etapach. Na etapie projektowania zostały sprecyzowane wymagania funkcjonalne, jakie każda z aplikacji powinna spełniać. Następnie na podstawie tych wymagań sformułowane zostały scenariusze testowe. W czasie rozwijania kodu, moduły testowane były manualnie. Pod koniec każdego etapu, przeprowadzane były testy akceptacyjne na podstawie scenariuszy testowych.

Poniższy rozdział zawiera wybrane scenariusze testowe:

- testowanie aplikacji internetowej:
  - tworzenie przewodnika (tab. 5.1)
  - tworzenie przewodnika z błędami (tab. 5.2)
  - modyfikacja atrybutu przewodnika (tab. 5.3)
- testowanie aplikacji mobilnej:
  - odczytanie informacji o obiekcie turystycznym z przewodnika (tab. 5.4)
  - nawigacja z kompasem (tab. 5.5)
  - pobranie danych (tab. 5.6)
  - nieudana próba pobrania danych (tab. 5.7)

Tabela 5.1: Scenariusz testowy dla akcji «Tworzenie przewodnika».

Cel testu	Testowanie poprawnego dodawania przewodnika
Sposób dostępu	Widok wywołany z poziomu menu „Przewodniki” → „Dodaj”. Użytkownik powinien posiadać uprawnienia do tworzenia przewodników.
Scenariusz (kroki testowe)	
Akcja użytkownika	**Odpowiedź systemu**
1. Wpisanie nazwy przewodnika „Lizbona” w pole „Nazwa”.	2. –
3. Wpisanie opisu „Lorem ipsum dolor sit amet” w pole „Opis”.	4. –
5. Odnalezienie na mapie miasta Lizbona i kliknięcie w to miejsce.	6. Wyświetlenie niebieskiego znacznika w miejscu kliknięcia.
7. Przeciągnięcie w szare pole zdjęcia z dysku twardego w formacie png o wymiarach większych niż 500 × 500 px.	8. Wyświetlenie wybranego zdjęcia w szarym polu.
9. Kliknięcie przycisku „Zapisz i kontynuuj”.	10. Przekierowanie do strony „Dodaj atrybuty” i informacja o sukcesie.

Tabela 5.2: Scenariusz testowy dla akcji «Tworzenie przewodnika z błędami».

Cel testu	Testowanie walidacji pól podczas dodawania przewodnika
Sposób dostępu	Widok wywołany z poziomu menu „Przewodniki" → „Dodaj". Użytkownik powinien posiadać uprawnienia do tworzenia przewodników.
Scenariusz (kroki testowe)	
Akcja użytkownika	**Odpowiedź systemu**
1. Pozostawienie pole „Nazwa" i pole „Opis" pustym.	2. –
3. Przeciągnięcie w szare pole zdjęcia z dysku twardego w formacie png o wymiarach większych niż 500x500 px.	4. Wyświetlenie wybranego zdjęcia w szarym polu.
5. Kliknięcie przycisku „Zapisz i kontynuuj".	6. Wyświetlenie informacji błędach: „Nazwa nie może być pusta", „Opis nie może być pusty", „Proszę wybrać miejsce".

Tabela 5.3: Scenariusz testowy dla akcji «Modyfikacja atrybutu przewodnika».

Cel testu	Testowanie edycji atrybutu przewodnika
Sposób dostępu	Widok wywołany z poziomu menu „Przewodniki" → „Pokaż". Wybrany przewodnik: „Lizbona". Użytkownik powinien posiadać uprawnienia do tworzenia przewodników i być twórcą przewodnika „Lizbona".
Scenariusz (kroki testowe)	
Akcja użytkownika	**Odpowiedź systemu**
1. Odnalezienie atrybutu „Historia" i kliknięcie „Edytuj".	2. Podmiana tekstów wybranego atrybutu na pola tekstowe. Podmiana obrazku symbolu na przycisk.
3. Zmiana tytułu atrybutu na „Lizboński renesans". Dodanie „Lorem ipsum" do opisu.	4. –
5. Kliknięcie przycisku z ikoną symbolu.	6. Wyświetlenie wyskakującego okna z dostępnymi symbolami.
7. Wybranie symbolu z 3 rzędu, 2 kolumny.	8. Wyświetlenie nowo wybranego symbolu w miejscu starego.
9. Kliknięcie przycisku „Zapisz"	10. Podmiana pól tekstowych i przycisku symbolu na statyczny tekst z uwzględnieniem zmian.

Tabela 5.4: Scenariusz testowy dla akcji «Odczytanie informacji o obiekcie turystycznym z przewodnika».

Cel testu	Testowanie poprawnego odczytu informacji o danym miejscu
Sposób dostępu	Widok wywołany poprzez włączenie aplikacji. Użytkownik powinien być zalogowany i mieć przewodnik „Lizbona" już pobrany na swój telefon.
Scenariusz (kroki testowe)	
Akcja użytkownika	**Odpowiedź systemu**
1. Wybranie listy pobranych przewodników pozycję „Lizbona".	2 Wyświetlenie widoku przewodnika.
3. Dotknięcie obrazku mapy.	4. Wyświetlenie widoku mapy wraz z markerami miejsc.
5. Wybranie markera opisanego jako „Castelo de Sao Jorge".	6. Wyświetlenie chmurki z nazwą miejsca.
7. Dotknięcie przycisku z białą strzałką na zielonym tle.	8. Wyświetlenie widoku miejsca wraz z jego opisem.

Tabela 5.5: Scenariusz testowy dla akcji «Nawigacja z kompasem».

Cel testu	Testowanie poprawnego działania nawigacji z kompasem
Sposób dostępu	Widok wywołany poprzez wybranie z pobranych przewodników pozycji „Lizbona" i przejście do mapy. Użytkownik powinien być zalogowany i mieć ten przewodnik już pobrany na swój telefon. GPS powinien być aktywny.
Scenariusz (kroki testowe)	
Akcja użytkownika	**Odpowiedź systemu**
1. Dotknięcie zielonego markera GPS w prawym, dolnym roku.	2. Wycentrowanie mapy do aktualnej pozycji użytkownika.
3. Ponowne dotknięcie markera GPS.	4. Wyświetlenie szarego pola możliwych pozycji i obrót mapy tak, aby północ wskazywała faktyczną północ telefonu.
5. Obrót telefonem	6. Obrót mapy zgodnie z obrotem telefonu.

Tabela 5.6: Scenariusz testowy dla akcji «Pobieranie danych».

Cel testu	Testowanie poprawnego działania pobierania danych
Sposób dostępu	Widok wywołany poprzez włączenie aplikacji. Użytkownik powinien być zalogowany. Transfer danych powinien być aktywny.
Scenariusz (kroki testowe)	
Akcja użytkownika	**Odpowiedź systemu**
1. Dotknięcie ikony „odśwież" znajdującej się na pasku aplikacji.	2. Wyświetlenie kółka postępu.
3. Oczekiwanie	4. Wyświetlenie pobranych przewodników w dwóch kategoriach: „Pobrane" i „Polecane"

Tabela 5.7: Scenariusz testowy dla akcji «Nieudana próba pobrania danych».

Cel testu	Testowanie obsługi błędów podczas pobierania danych
Sposób dostępu	Widok wywołany poprzez włączenie aplikacji. Użytkownik powinien być zalogowany. Transfer danych powinien być wyłączony.
Scenariusz (kroki testowe)	
Akcja użytkownika	**Odpowiedź systemu**
1. Dotknięcie ikony „odśwież" znajdującej się na pasku aplikacji.	2. Wyświetlenie kółka postępu.
3. Oczekiwanie	4. Wyświetlenie komunikatu o braku połączeniu z Internetem. Wyświetlenie poprzednich pozycji na listach.

Rozdział 6

Wnioski końcowe

6.1 Kierunek dalszych prac

Realizacja projektu przebiegła pomyślnie i udało się osiągnąć założone cele. Powstała w pełni funkcjonalna aplikacja mobilna, która gotowa jest do użytku przez użytkowników końcowych. Aplikacja internetowa wymaga jeszcze kilku dodatkowych funkcjonalności przed wdrożeniem jej do publicznego użytku. Są to: funkcja edycji konta użytkownika, funkcja przywracania i zmiany hasła, rola administratora i możliwość zarządzania użytkownikami, tworzenie nowych kont dla twórców. W przyszłości warto byłoby wdrożyć także panel administracyjny dla użytkowników-podróżników, gdzie mieliby oni wgląd do pobranych przez siebie przewodników oraz mogliby pobierać kolejne. Jedną z dodatkowych możliwości rozwoju jest umożliwienie podróżnikom tworzenie i publikowanie własnych przewodników.

Aplikacja mobilna także posiada bardzo wiele perspektyw rozwoju. W pierwszej kolejności warto byłoby stworzyć wersję dla systemów iOS oraz Android, gdyż są one bardziej popularne niż platforma Windows Phone. Ponadto, do aplikacji można wdrożyć funkcję wirtualnego przewodnika. Polegałaby ona na tym, że użytkownik wybierałby miejsca, które chce odwiedzić a aplikacja obliczałaby optymalną drogę pomiędzy nimi. Następnie tryb mapy przełączałby się w tryb nawigacji i użytkownik byłby prowadzony do kolejnych punktów i informowany na bieżąco o tym, co znajduje się wokół niego. Można by tu także pomyśleć o wykorzystaniu rozszerzonej rzeczywistości.

6.2 Napotkane problemy

Podczas realizacji projektu pojawiło się wiele nieoczekiwanych problemów. Po pierwszych testach aplikacji mobilnej okazało się, że pobieranie danych jest bardzo niewydajne. Algorytmy zostały więc zoptymalizowane i pobieranie znacznie przyśpieszyło. Problematyczne było także pierwsze zetknięcie z Bing Maps API. Dostosowanie zachowania mapy wymagało więcej czasu niż zakładano.

Również aplikacja serwerowa przeniosła kilka niespodziewanych problemów. Pierwotnie zaczęła być ona rozwijana w oparciu o platformę Groovy on Grails[16], szybko jednak została

przeniesiona do technologii .NET. Zanim zdecydowano się na wykorzystanie platformy Web API poczyniono podejście do technologii WCF[30]. Web Api zostało wybrane, gdyż jest ono znacznie prostsze niż WCF, nowsze lecz dobrze udokumentowane.

Początkowo problematyczne było odpowiednie podzielenie solucji na projekty, co skutkowało wielokrotną refaktoryzacją kodu. Finalnie projekt nabrał prawidłowych kształtów zgodnych ze wzorcami projektowymi opisanymi w sekcji 4.2.

Niespodziewanym problemem, który ujawnił się dopiero po instalacji aplikacji na zdalnym serwerze, była blokada zapytań pochodzącej z innej domeny. Problem oraz jego rozwiązanie zostały szczegółowo opisane w sekcji 2.2.

Aplikacja internetowa nie od razu została napisana przy użyciu platformy Angular. Przeprowadzono eksperymenty z EmberJS[11] oraz BackboneJS[9]. EmberJS okazał się być zbyt młodą biblioteką, której zabrakło wsparcia dla serwisów Web API; Backbone okazał się zbyt ciężki oraz dość trudny w skonfigurowaniu w środowisku IIS. Finalnie wybrano więc AngularJS. Jako, że ta biblioteka była nowością dla autorki, sporo czasu zostało poświęcone na naukę i zrozumienie zasady działania platformy. Problematyczne było pierwsze podejście do konfiguracji, jednakże dzięki bogatym zasobom w Internecie udało się ten problem sprawnie pokonać.

Zdecydowana większość problemów wynikała z tego, że było to pierwsze podejście autorki do danej technologii. Wszystkie zostały rozwiązane dzięki poszukiwaniom odpowiedzi w Internecie, literaturze technicznej oraz dzięki zasięganiu porady u bardziej doświadczonych kolegów i koleżanek.

Rozdział 7

Zakończenie

W wyniku przeprowadzonego projektu powstała w pełni użyteczna platforma *TravelGuide* gotowa do wdrożenia do użytku przez użytkowników końcowych. Program ułatwia podróżnikom zwiedzanie nowych miejsc i może zastąpić tradycyjne przewodniki.

Na całość składa się aplikacja mobilna na platformę Windows Phone 8.1, aplikacja działająca w przeglądarce internetowej oraz aplikacja serwerowa odpowiedzialna za pośredniczenie w komunikacji pomiędzy aplikacjami klienckimi i bazą danych.

Aplikacja internetowa przeznaczona jest dla użytkowników-twórców, którzy mają możliwość dodawania nowych przewodników za pomocą przyjaznego i dynamicznego kreatora. Te przewodniki są następnie publikowane i stają się dostępne dla użytkowników aplikacji mobilnej.

Użytkownicy-podróżnicy mają możliwość pobrania przewodników na swoje telefony i wykorzystania ich w trakcie zwiedzania nowych miejsc (także w trybie offline). Aplikacja przedstawi im najciekawsze informacje, przydatne wskazówki a także wyświetli mapę z miejscami wartymi zobaczenia posortowanymi po kategoriach. Ponadto, gdy użytkownik aktywuje urządzenie GPS w swoim smartfonie, aplikacja wyświetli mu jego pozycję (co nie uniemożliwi zgubienie się w nowym miejscu) oraz pokaże ciekawe obiekty w najbliższej okolicy.

Funkcja wykorzystująca dane geolokalizacyjne jest największą przewagą aplikacji *TravelGuide* nad książkowymi przewodnikami. Dodatkową korzyścią jest także możliwość aktualizacji treści tak, że nie ma ryzyka iż ulegną one przedawnieniu. No i wreszcie – użytkownik-podróżnik nie musi już martwić się wizytą w księgarni przed wyjazdem na urlop. W swój przewodnik zaopatrzy się nawet na lotnisku czy w innym dowolnym miejscu z dostępem do Internetu.

Bibliografia

[1] Jan Mietelski. *Astronomia w geografii.* Państwowe Wydawnictwo Naukowe, 1989.

[2] AdminLTE. http://almsaeedstudio.com/AdminLTE/. [data dostępu: 21 grudnia 2014].

[3] AngularJS. https://angularjs.org/. [data dostępu: 21 grudnia 2014].

[4] AngularTranslateJS. http://angular-translate.github.io/. [data dostępu: 21 grudnia 2014].

[5] ASP.NET Cors. https://www.nuget.org/packages/Microsoft.AspNet.Cors/. [data dostępu: 21 grudnia 2014].

[6] ASP.NET Identity. https://aspnetidentity.codeplex.com/. [data dostępu: 21 grudnia 2014].

[7] ASP.NET WebApi 2. http://msdn.microsoft.com/en-us/library/dn448365%28v=vs.118%29.aspx. [data dostępu: 21 grudnia 2014].

[8] Automapper. https://github.com/AutoMapper/AutoMapper. [data dostępu: 21 grudnia 2014].

[9] Backbone. http://backbonejs.org/. [data dostępu: 23 grudnia 2014].

[10] Bazaar. http://bazaar.canonical.com. [data dostępu: 21 grudnia 2014].

[11] EmberJS. http://emberjs.com/. [data dostępu: 23 grudnia 2014].

[12] Enterprise Architect http://www.sparxsystems.com/products/ea/. [data dostępu: 31 grudnia 2014].

[13] Entity Framework. http://msdn.microsoft.com/en-us/data/ef.aspx. [data dostępu: 21 grudnia 2014].

[14] Fly4Free.pl. http://www.fly4free.pl/forum/wasze-sposoby-na-planowanie-zwiedzania-nowych-miejsc,18,59116. [data dostępu: 6 grudnia 2014].

[15] GIT. http://git-scm.com. [data dostępu: 21 grudnia 2014].

[16] Groovy on Grails. https://grails.org/. [data dostępu: 23 grudnia 2014].

[17] Historia turystyki na świecie. http://pl.wikibooks.org/wiki/Podstawy_turystyki/Historia_turystyki_na_%C5%9Bwiecie. [data dostępu: 28 grudnia 2014].

[18] jQuery. http://jquery.com/. [data dostępu: 21 grudnia 2014].

[19] LeafletJS. http://leafletjs.com/. [data dostępu: 21 grudnia 2014].

[20] Mecurial. http://mercurial.selenic.com. [data dostępu: 21 grudnia 2014].

[21] MVVM light. https://mvvmlight.codeplex.com/. [data dostępu: 21 grudnia 2014].

[22] Newtonsoft JSON. http://james.newtonking.com/json. [data dostępu: 21 grudnia 2014].

[23] OpenStreetMaps. http://www.openstreetmap.org/. [data dostępu: 21 grudnia 2014].

[24] StarUML http://staruml.io/. [data dostępu: 31 grudnia 2014].

[25] SQL Server 2014. http://www.microsoft.com/pl-pl/server-cloud/products/sql-server/. [data dostępu: 21 grudnia 2014].

[26] SQLite. http://www.sqlite.org/. [data dostępu: 21 grudnia 2014].

[27] SVN. https://subversion.apache.org. [data dostępu: 21 grudnia 2014].

[28] TFS. http://www.visualstudio.com/en-us/products/tfs-overview-vs.aspx. [data dostępu: 21 grudnia 2014].

[29] Twitter Bootstrap. http://getbootstrap.com/. [data dostępu: 21 grudnia 2014].

[30] WCF Data Services. http://msdn.microsoft.com/pl-pl/library/ff843374.aspx. [data dostępu: 23 grudnia 2014].

[31] Wzorzec MVVM. http://msdn.microsoft.com/pl-pl/library/wprowadzenie-do-wzorca-projektowego-model-view-viewmodel-na-przykladzie-aplikacji-wpf.aspx. [data dostępu: 27 grudnia 2014].

Załączniki

Do pracy dołączona jest płyta CD zawierająca ten dokument w wersji elektronicznej oraz kod źródłowy aplikacji *TravelGuide*.