



UMCS

**UNIWERSYTET MARII CURIE-SKŁODOWSKIEJ
W LUBLINIE**
Wydział Matematyki, Fizyki i Informatyki

Kierunek: **Informatyka**

Paweł Małek

nr albumu: 236367

**Komponent GMap jako narzędzie
wyznaczania tras**

GMap Component as a Tool for Routing

Praca licencjacka
napisana w Zakładzie Informatyki Stosowanej Instytutu Informatyki
pod kierunkiem dr Barbary Gocłowskiej

Lublin rok 2016

Spis Treści

Wstęp.	5
Rozdział 1. GMap - Komponent mapy.	7
1.1 Google Maps API.	7
1.2 Umieszczenie komponentu w aplikacji internetowej.	7
1.3 Rodzaje map oraz warstwy.	9
Rozdział 2. Podstawowe obiekty overlay.	14
2.1 Markery	14
2.2 Circle, Rectangle i Polygon.	15
2.3 InfoWindow czyli wyskakujące okienko.	16
2.4 Stylizacja obiektów overlay.	18
2.5 Obsługa zdarzeń.	19
Rozdział 3 - Ścieżki podróży.	21
3.1 DirectionsService.	22
3.2 DirectionsRenderer.	24
Rozdział 4 – Aplikacja internetowa – Portal dla podróżników.	27
4.1 Proces dodawania nowej wycieczki.	28
4.2 Pierwsze spojrzenie na stronę kreowania tras.	30
4.3 Geocoding i Reverse Geocoding.	30
4.4 Główny panel zarządzania obiektami Overlay.	33
4.5 Wyszukiwanie i dołączanie do wycieczki.	35
4.6 Kilka słów na temat zapisu danych mapy.	36
4.6.1 JSON jako format zapisu danych.	37
4.6.2 Kodowanie współrzędnych.	38
Podsumowanie.	39
Bibliografia.	40

Wstęp

Wyznaczanie tras podróży na stronie lub aplikacji internetowej to bardzo ciekawa i przydatna opcja dla ludzi, którzy chcą zorganizować wycieczkę dla większego grona osób. Stąd też celem mojej pracy jest wykorzystanie komponentu mapy i możliwości jakie dostarcza do stworzenia graficznego interfejsu użytkownika umożliwiającego interaktywne wyznaczanie tras. W tym celu wykorzystuję komponent GMap z biblioteki PrimeFaces oraz Google Maps API, na podstawie którego zbudowany jest komponent. W moim przekonaniu komponent GMap w połączeniu z API od Google'a jest świetnym narzędziem do wyznaczania dokładnych tras z uwzględnieniem najmniejszych szczegółów. Główną część logiki komponentu mapy oraz obsługę wszelkich jego funkcjonalności napisałem w technologii JavaScript i jQuery. Sama aplikacja zbudowana jest w technologii JSF (Java Server Faces) z wykorzystaniem komponentów UI (User Interface) z biblioteki PrimeFaces. Do komunikacji z bazą danych używam technologii Hibernate, która jest bardzo dobrym i rozbudowanym narzędziem do tego typu celów.

W pierwszym rozdziale pracy przedstawiłem komponent GMap, podstawowe informacje jakie są potrzebne do rozpoczęcia pracy z mapą oraz najnowszą wersję Google Map API, bez którego korzystanie z map wydaje się niemożliwe. Rozdział drugi zawiera informacje na temat podstawowych elementów jakie można dodać do widoku mapy, ich konfiguracja wizualna oraz obsługa zdarzeń. W kolejnej części opisałem sposoby dodawania tras i ścieżek, DirectionService czyli serwisu, który automatycznie generuje trasę podróży. Ostatni rozdział poświęciłem na demonstrację Aplikacji – Portal dla podróżników, który umożliwia organizowanie wspólnych wycieczek, wyznaczanie tras podróży, wyszukiwanie i dołączanie do wycieczek utworzonych przez innych użytkowników.

JavaScript – jest językiem skryptowym stworzonym rzez firmę Netscape w 1995 roku. Wykorzystywany głównie do tworzenia interaktywnych interfejsów użytkownika na stronach internetowych. JS jest uniwersalnym silnie obiektowym językiem programowania co oznacza, że praca w JavaScript to praca głównie na obiektach [1].

jQuery – jest lekką biblioteką języka JavaScript, stworzoną aby ułatwić pisanie w tymże języku. Wspomaga kontrole obiektów DOM'u (Document Object Model) [26], ułatwia obsługę zdarzeń, tworzenie animacji, zarządzanie stylami CSS i wiele innych [2].

Hibernate – Szkielet aplikacji java odpowiadający za komunikacje z bazą danych. Główną zaletą technologii są odwzorowania obiektowo relacyjne (ORM), dzięki którym tabele bazy danych generowane są na podstawie stworzonych klas encyjnych i plików konfiguracyjnych. Wykorzystywany język zapytań to HQL (hibernate query language), którego składnia zbliżona jest do SQL [22].

PrimeFaces – Biblioteka komponentów UI (User Interface), dla aplikacji internetowych. Zawiera się w jednym pliku jar i nie wymaga żadnej konfiguracji. W bibliotece znajduje się także baza skórek zmieniających styl komponentów [3].

Rozdział 1 – GMap - Komponent mapy

GMap to komponent znajdujący się w bibliotece PrimeFaces, oparty jest na Google Maps API stąd też jego nazwa [4]. Służy do wstawiania widoku mapy na stronie lub aplikacji internetowej. Platforma PrimeFaces dostarcza własne API (Application Programming Interface) do obsługi mapy, dostępne z poziomu kodu Java [3]. Niestety nie jest ono wystarczająco rozbudowane i funkcjonalne, aby wykorzystać go do celów mojej aplikacji. Z tego względu używam natywnego API od Google opartego na języku JavaScript. W przypadku komponentu GMap technologia JS pozwala na stworzenie bardziej interaktywnego interfejsu zarządzania mapą i jej elementami.

1.1 Google Maps API

Jest to interfejs programistyczny aplikacji służący do korzystania z mapy oraz funkcjonalności i serwisów, jakie oferuje Google. Firma udostępnia swój interfejs na różne platformy między innymi Android, iOS czy też aplikacje internetowe. Na tę ostatnią API jest dostępne w języku JavaScript [4]. Google Maps API dostarcza takie możliwości jak: Geokodowanie, Geolokacja, DirectionService i wiele innych, o których napisałem w dalszej części pracy. Aktualnie dostępną wersja Google Maps API jest wersja trzecia (v3), która oferuje funkcjonalności zbliżone do tych, które znajdują się na oficjalnej stronie Google Maps [5].

1.2 Umieszczenie komponentu mapy w aplikacji internetowej

Pierwszym krokiem jaki trzeba wykonać, żeby zacząć korzystać z map, jest zainportowanie bibliotek JavaScript ze strony Google. Czynność tę należy wykonać bez względu na to czy, używa się komponentu mapy z PrimeFaces, czy natywnego obiektu mapy znajdującego się w Google Maps API.

```
1 | <script src="http://maps.google.com/maps/api/js?sensor=true/false"
2 |
3 |   type="text/javascript" />
4 | 
```

Listing 1.1. Dodanie biblioteki Google Maps API do projektu.

W poprzednich wersjach Google Maps API, do korzystania z map wymagany był specjalny klucz, tzw. API-KEY. Od czasu wprowadzenia wersji trzeciej, klucz nie jest wymagany, chyba że programista posiada dodatkowe przywileje i będzie używać map do celów komercyjnych. Klucz przydaje się również w sytuacjach, gdy aplikacja będzie monitorować statystyki używania map.

```
1 | <script src="http://maps.google.com/maps/api/js?sensor=true/false  
2 |   &key=API_KEY" type="text/javascript" />  
3 |  
4 |
```

Listing 1.2. Dodanie biblioteki Google Maps API z podaniem klucza API_KEY.

Import biblioteki z API jaki podałem w przykładach powyżej dostarcza podstawowych funkcjonalności i w większości przypadków użycia map to wystarczy. Jednakże zdarza się, że potrzebne są szczegółowe funkcje jakich domyślnie nie ma w zainstalowanej bibliotece, w takim przypadku wymagane jest dołączenie dodatkowych bibliotek.

```
1 | <script src="http://maps.google.com/maps/api/js?sensor=true/false  
2 |   &libraries=geometry, drawing" type="text/javascript" />  
3 |  
4 |
```

Listing 1.3. Przykład pobrania Google Maps API z dodatkowymi bibliotekami.

Samo wstawienie mapy do aplikacji jest czynnością bardzo prostą, wystarczy skorzystać z biblioteki PrimeFaces i zawartego w niej komponentu <p:gmap>. Mapę dodaje się tak jak każdy inny komponent, należy jednak pamiętać o trzech podstawowych atrybutach.

- center – współrzędne geograficzne punktu jaki ma się pojawić w centrum mapy,
- zoom – początkowa wartość przybliżenia,
- type – rodzaj mapy.

Bardzo ważnym atrybutem jest widgetVar, dzięki niemu mam możliwość dostania się do mapy z poziomu kodu JavaScript. Dokładniej mówiąc, widgetVar jest rodzajem wskaźnika na obiekt klasy google.maps.Map, tworzonego przez komponent PrimeFaces [27].

```
1 | <p:gmap id="Gmap"
2 |   widgetVar="mapVar"
3 |   center="51.246, 22.541"
4 |   zoom="10"
5 |   type="ROADMAP"
6 | </p:gmap>
```

Listing 1.4. Wstawienie komponentu mapy do aplikacji.

```
1 | var mapa = PF('mapVar').getMap();
2 |
3 |
```

Listing 1.5. Odwołanie się do obiektu mapy z poziomu kodu JS.

1.3 Rodzaje map oraz warstwy.

Jedną z podstawowych zmian jakich można dokonać w widoku mapy są typy map. Jest to rodzaj danych przestrzennych, jaki ukazuję się użytkownikowi np. mapa terenowa. Typy można zmieniać na dwa sposoby. Pierwszy – statyczny, oznacza przypisanie atrybutowi „type” w komponencie GMap odpowiednią wartość typu String. Sposób dynamiczny polega na bezpośredniej zmianie atrybutu obiektu mapy z poziomu kodu JavaScript za pomocą metody mapa.setMapTypeId() [6].

```
1 | <p:gmap id="mapa" center="51.246, 22.541" zoom="10" type="HYBRID" ></p:gmap>
2 |
3 |
```

Listing 1.6. Przykład zmiany typu mapy - statycznie.

```
1 | var mapa = PF('mapVar').getMap();
2 |
3 | mapa.setMapTypeId(google.maps.MapTypeId.ROADMAP);
4 |
5 |
```

Listing 1.7. Przykład zmiany typu mapy - dynamicznie.

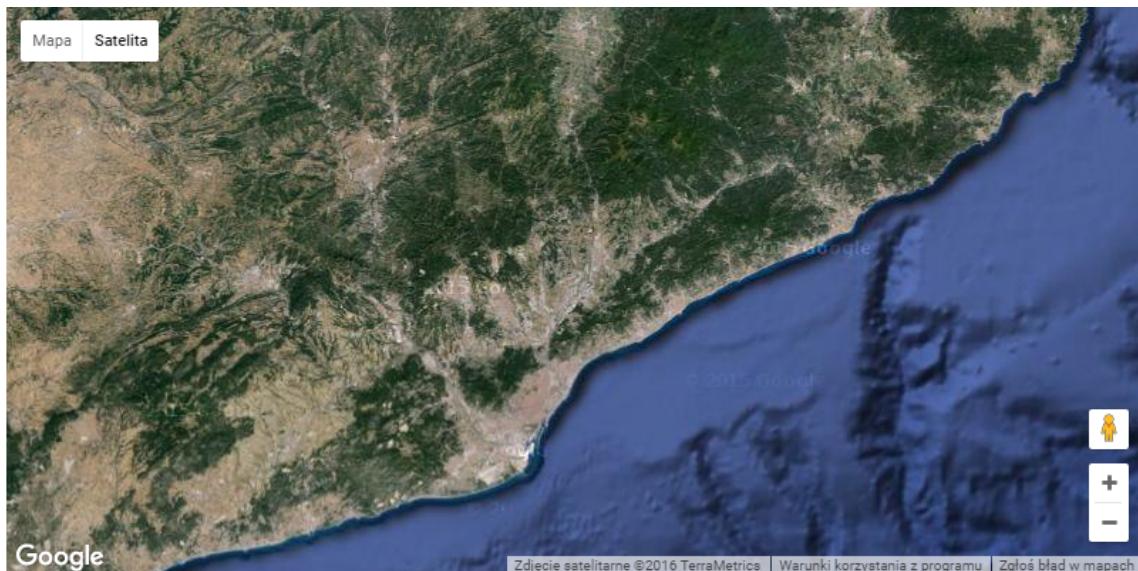
Do dyspozycji są cztery rodzaje map, które można ustawać według potrzeb i upodobań:

- ROADMAP – standardowa mapa z wyszczególnionym widokiem wszelkich dróg, ulic, autostrad itp. Typ reprezentowany przez klasę google.maps.MapTypeId.ROADMAP.



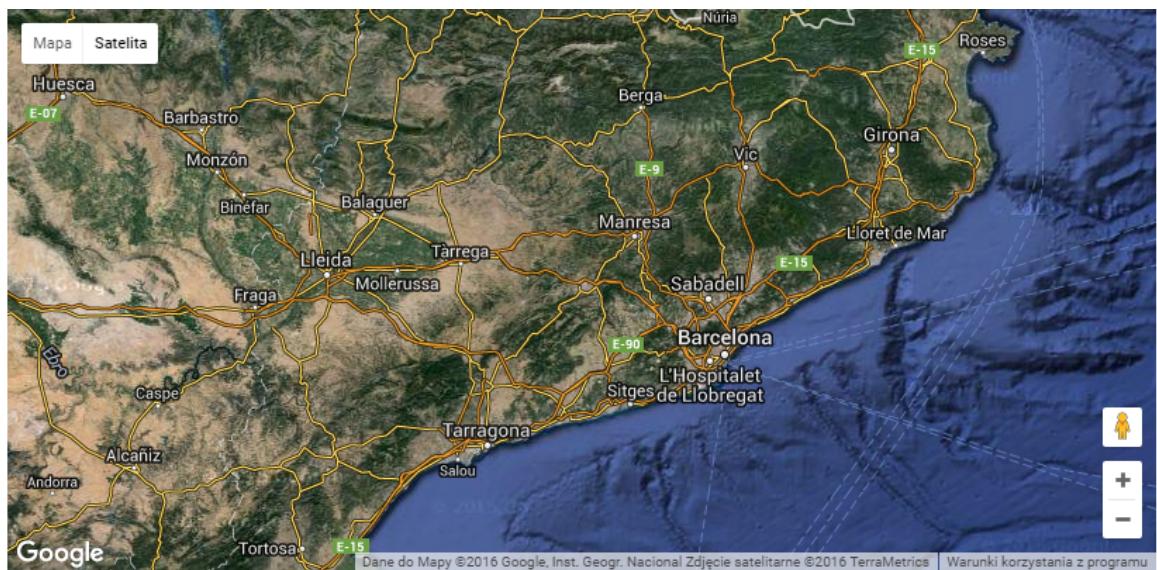
Rys 1.1 Widok mapy – Roadmap

- SATELLITE – widok mapy jako zdjęcia satelitarnego ziemi lub inaczej zdjęcia lotnicze. (google.maps.MapTypeId.SATELLITE)



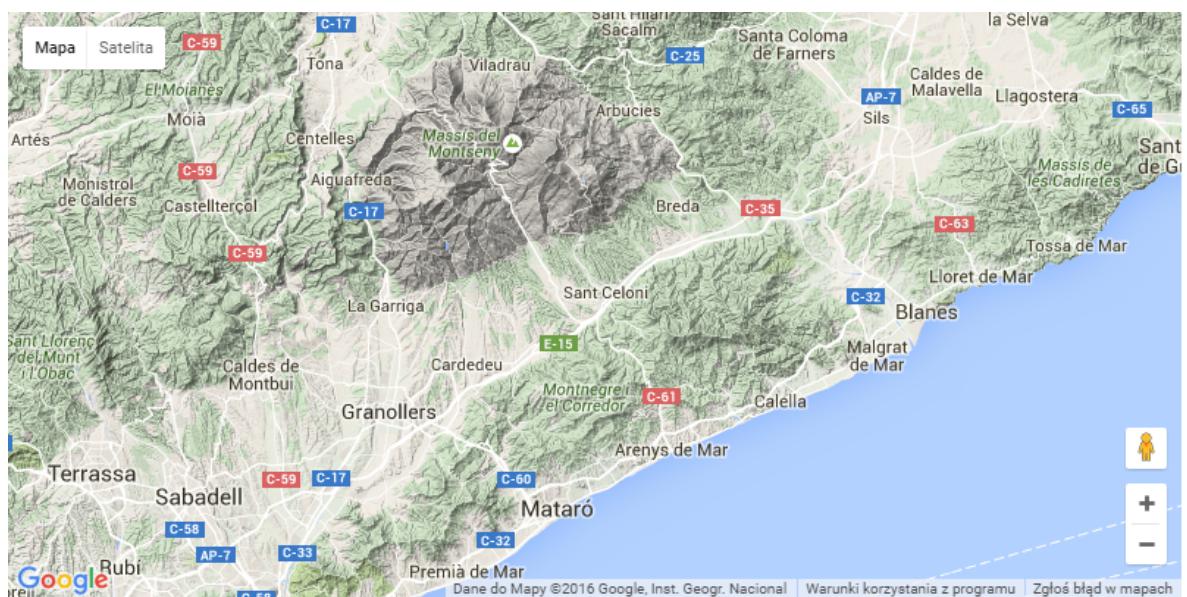
Rys 1.2. Widok mapy – Satellite

- HYBRID – połączenie widoku sieci dróg z mapą satelitarną (google.maps.MapTypeId.HYBRID)



Rys 1.3. Widok mapy – Hybrid

- TERRAIN – standardowa mapa z widokiem ukształtowania terenu. (google.maps.MapTypeId.TERRAIN)



Rys 1.4. Widok mapy – Terrain.

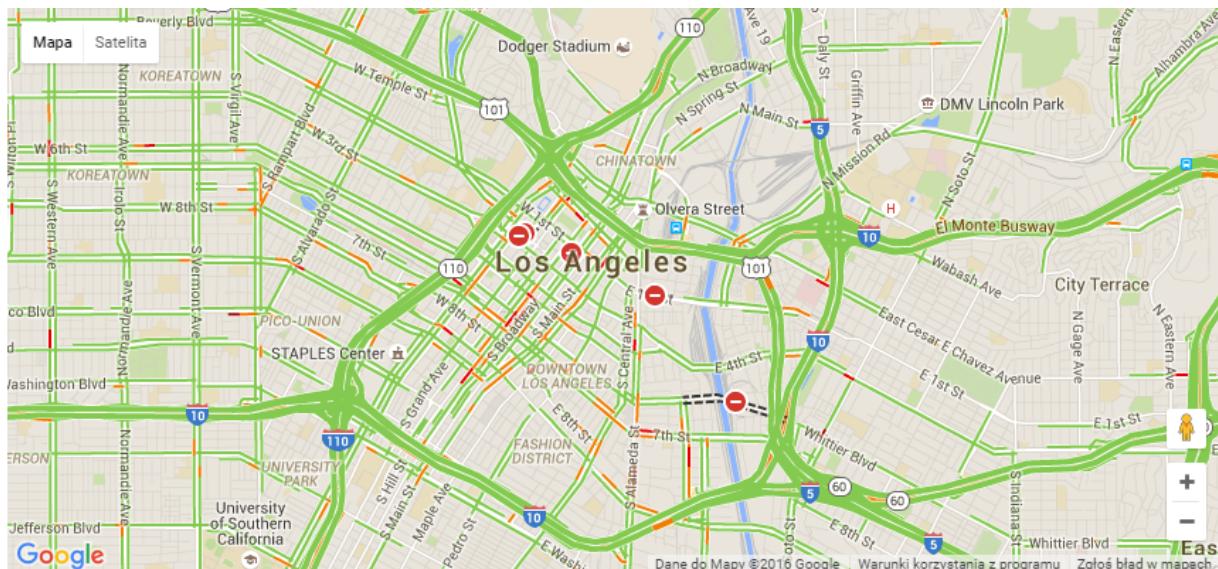
Podstawowy widok mapy można wzbogacić również poprzez dodanie warstw (Layer), które są obiektami składającymi się z jednego lub wielu graficznych elementów. Warstwy pokrywają mapę dodając do niej dodatkowe sieci dróg i różne dane co niewątpliwie może pomóc użytkownikowi w wyznaczaniu odpowiedniej trasy i dostarczyć potrzebnych informacji. Dodaje się je za pomocą metody setMap().

```
1 var mapa = PF('mapVar').getMap();
2 var warstwa = new google.maps.TrafficLayer();
3 warstwa.setMap(mapa);
4
5
6
7
```

Listing 1.8. Przykład dodania nowej warstwy.

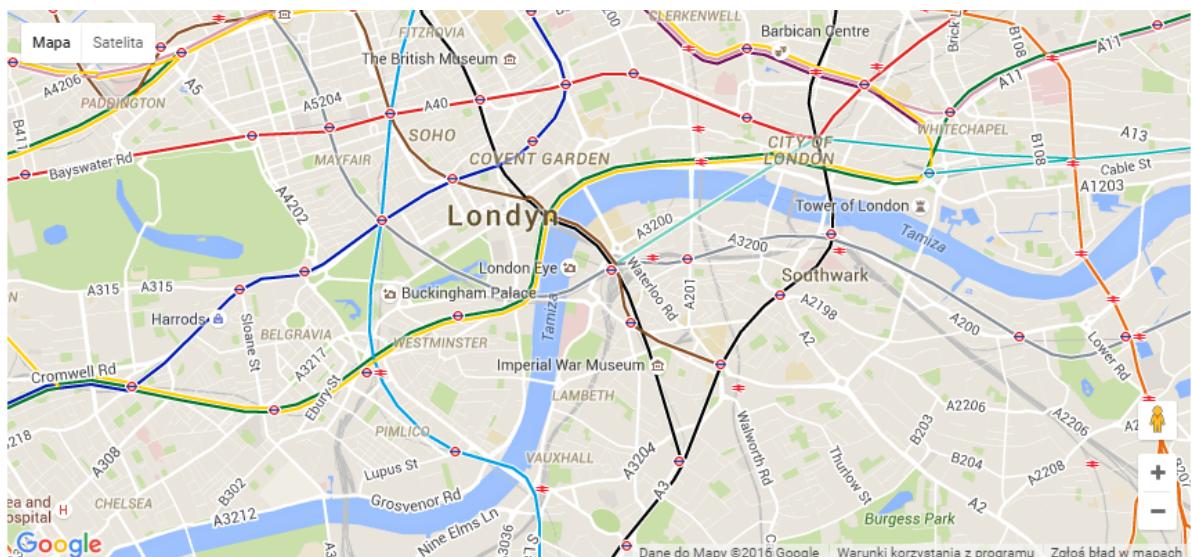
Główne warstwy, które użyłem w swojej aplikacji to:

- TrafficLayer – warstwa przedstawiająca mapę dróg z aktualnymi informacjami o natężeniu ruchu, przejezdności ulic, korkach, robotach drogowych itp.



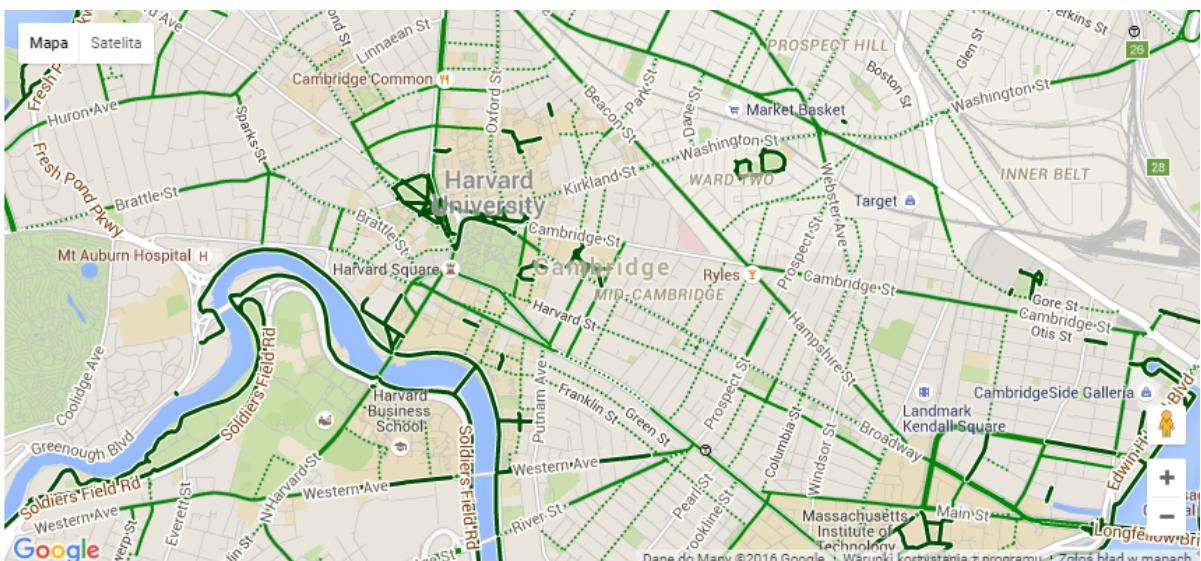
Rys 1.5. TrafficLayer

- TransitLayer – przedstawia mapę sieci metra, pociągów, autobusów i inne rodzaje komunikacji publicznej.



Rys 1.6. TransitLayer

- BicyclingLayer – nakładka przedstawiająca sieć dróg i ścieżek rowerowych.



Rys 1.7. BicyclingLayer

Rozdział 2 – Podstawowe obiekty overlay

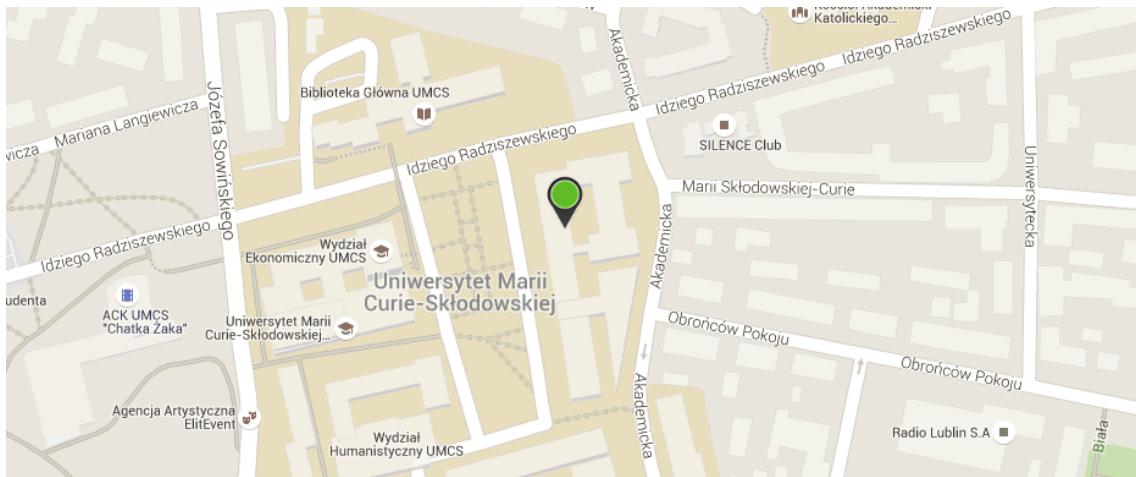
W odniesieniu do komponentu GMap są to wszelkiego rodzaju graficzne elementy mapy takie jak markery, zdjęcia, kształty i ścieżki, które można dodać do widoku mapy. Wszystkie te obiekty dziedziczą po klasie google.maps.OverlayView() [18]. Elementy overlay można wykorzystać do utworzenia własnej warstwy danych, którą umieszcza się na mapie, przykładem takiej warstwy może być niewątpliwie szczegółowa trasa podróży.

2.1 Markery

Marker jest znacznikiem, punktem, który można przypiąć do mapy pod konkretne współrzędne. Podstawowym atrybutem markera jest position - współrzędne geograficzne składające się z dwóch wartości latitude i longitude (lat, lng). Poza współrzędnymi geograficznymi markerowi można nadać tytuł, dodać opis lub zmienić domyślną ikonkę zastępując ją własną grafiką png lub gif. Marker jest podstawowym elementem wykorzystywany w wyznaczaniu trasy, pełni role punktów kontrolnych i służy do oznaczania szczególnych miejsc na mapie.

```
1 var marker = new google.maps.Marker({
2   position: new google.maps.LatLng(51.24605, 22.54191),
3   animation: new google.maps.Animation.DROP,
4   title: "Informatyka UMCS"
5   icon: {
6     url: './resources/images/markers/greenMarker.png',
7     size: new google.maps.Size(30, 44),
8     anchor: new google.maps.Point(14, 44)
9   },
10  customInfo: {
11    info: "Przykładowe informacje",
12    opis: "Przykładowy opis markera",
13  },
14  map: mapa
15 });
16
17
18 });
19
```

Listing 2.1. Utworzenie i dodanie markera do mapy.



Rys 2.1. Widok przykładowego markera na mapie.

2.2 Circle, Rectangle i Polygon.

Kolejny typ obiektów overlay, które chciałbym przedstawić to kształty (Shapes). Są to figury geometryczne jakie można wyrysować na mapie w celu zaznaczenia obszaru lub oznaczenia konkretnego budynku. W zależności od potrzeb w bibliotece Google Maps API do dyspozycji są:

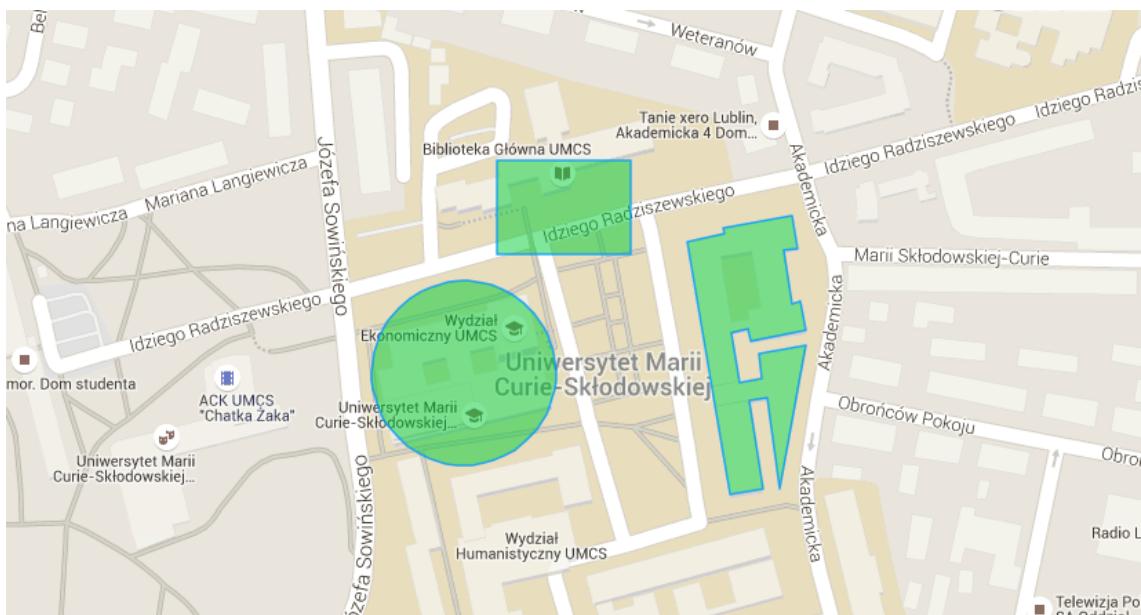
- Circle – Obszar w kształcie okręgu. Jego podstawowe atrybuty to: center czyli współrzędne środka okręgu oraz radius – długość promienia. Obiekt reprezentowany jest przez klasę google.maps.Circle [13].
- Rectangle – google.maps.Rectangle – prostokąt, do jego wyznaczenia potrzebne są dwie współrzędne przeciwnielegkich wierzchołków figury [15].
- Polygon – google.maps.Polygon – jest bardziej złożonym obiektem, który może odzwierciedlać dowolną figurę lub obszar na mapie. Jego główną składową jest „path” czyli ścieżka składająca się ze współrzędnych geograficznych kolejnych wierzchołków [14].
- Polyline – google.maps.Polyline – inaczej polilinia jest obiektem reprezentującym linię prostą lub zbiór połączonych ze sobą linii tworzących ścieżkę. Więcej o polilinii i o tym jak można ją wykorzystać przy wyznaczaniu tras podróży znajduje się w kolejnym rozdziale [12].

```

1
2
3
4
5
6
7
8
9
10
11
12
var circleTmp = new google.maps.Circle({
  strokeColor: '#0099FF',
  strokeOpacity: 0.8,
  strokeWeight: 1.5,
  fillColor: '#00CC33',
  fillOpacity: 0.5,
  map: mapa,
  center: new google.maps.LatLng(51.24605, 22.54191),
  radius: 150
});

```

Listing 2.2. Implementacja obiektu Circle.



Rys 2.2 Przykład użycia obiektów overlay do zaznaczania obszarów.

2.3 InfoWindow czyli wyskakujące okienko.

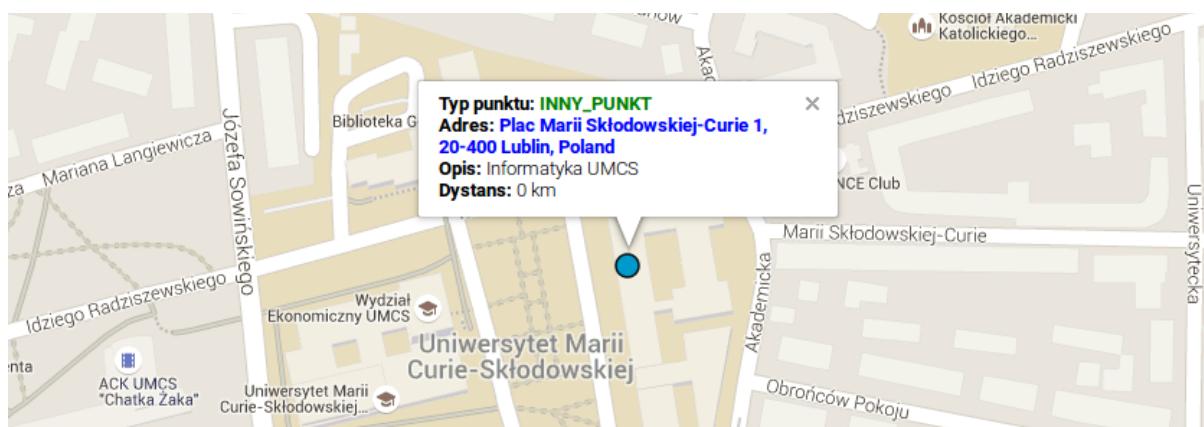
InfoWindow także jest obiektem overlay, jak sama nazwa podpowiada jest to okienko z informacjami [16]. Okno pojawia się i znika w zależności od zdefiniowanej obsługi zdarzeń na już dodanych elementach mapy. Przykładowo InfoWindow może pojawić się gdy użytkownik najedzie kursem myszy na marker lub inny obiekt, oraz znikać, gdy kurSOR myszy z niego zabierze. W swojej aplikacji, okienka InfoWindow używam do wyświetlania informacji dodanych do poszczególnych punktów, takich jak adres lub opis miejsca.

Główne atrybuty InfoWindow:

- content – może to być zwykły tekst, opis markera, miejsca itp. lub bardziej skomplikowana struktura w formacie html, w której można zawrzeć linki, zdjęcia pola tekstowe itp.
- position – współrzędne na mapie, w których ma pojawić się okno. Przeważnie lokalizacje deklaruje się w momencie wywołania okna InfoWindow za pomocą metody open(), której parametrami jest mapa oraz pozycja na mapie.
- maxWith – maksymalna szerokość okna. Jest to opcjonalne ustawienie, jednakże należy o nim pamiętać jeżeli dane umieszczone w okienku będą bardzo rozbudowane.

```
1 google.maps.event.addListener(marker, 'click', function() {  
2     var infoWindow = new google.maps.InfoWindow({maxWidth: 240});  
3  
4     var content = '<strong><p style="color: black;">Typ punktu: </p></strong>+'  
5     content += '<p style="color: green;">' + this.customInfo.typ + '</p><br/>+'  
6     content += '<strong><p style="color: black;">Adres: </p></strong>+'  
7     content += '<p style="color: blue;">' + this.customInfo.adres + '</p><br/>+'  
8     content += '<strong><p style="color: black;">Opis: </p></strong>+'  
9     content += '<p style="color: black;">' + this.customInfo.opis + '</p><br/>+'  
10    content += '<strong><p style="color: black;">Dystans: </p></strong>+'  
11    content += '<p style="color: black;">' + this.customInfo.dystans + ' km</p><br/>';  
12  
13    infoWindow.setContent(content);  
14    infoWindow.open(GmapOdwzorowanie, this);  
15};  
16});  
17  
18
```

Listing 2.3. Przykładowa implementacja infoWindow.



Rys 2.3 Przykład użycia InfoWindow.

2.4 Stylizacja obiektów overlay.

Większość obiektów dodanych do mapy można również dowolnie skonfigurować pod względem wizualnym. Sposób zmiany właściwości poszczególnych elementów zbliżony jest do technologii CSS (Cascading Style Sheets) [7]. Każdy z obiektów overlay posiada atrybuty, które modyfikują jego wygląd.

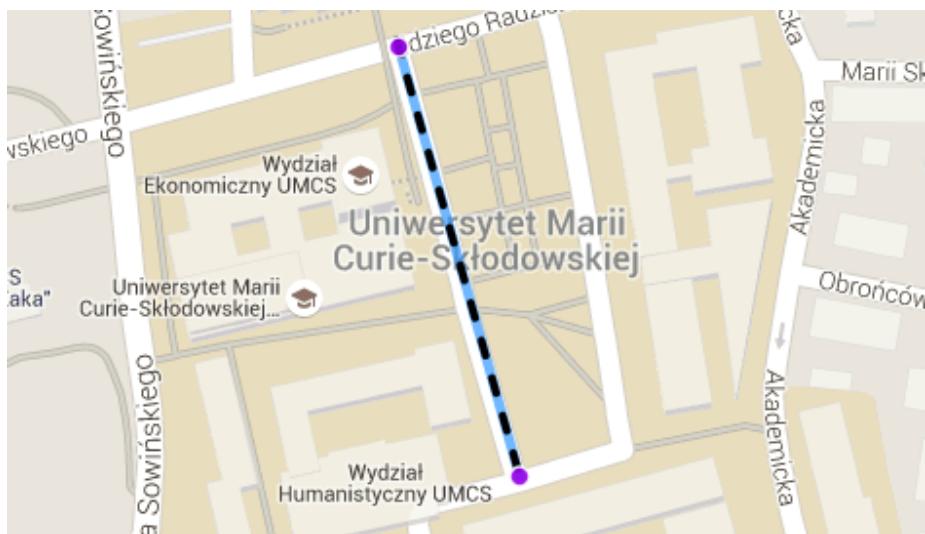
Podstawowe atrybuty:

- strokeColor – kolor krawędzi. Do definiowania koloru używana jest notacja szesnastkowa (HEX), która jest kombinacją kolorów RGB(Red Green Blue), np. kolor czarny to #000000 -> rgb(0,0,0).
- StrokeWeight – grubość krawędzi (wartość zmiennoprzecinkowa).
- StrokeOpacity – przeźroczość krawędzi (wartość zmienno-przecinkowa od 0 do 1).
- fillColor – kolor wypełnienia.
- fillOpacity – przeźroczość wypełnienia (od 0 do 1).

Żeby wzbogacić nieco utworzony obiekt można również użyć SVG – (Scalable Vector Graphic) [8]. Jest to standard modelowania grafiki wektorowej w formacie XML, zatwierdzony przez organizacje W3C (World Wide Web Consortium) [9].

```
1  strokeColor: '#3399FF',
2  strokeopacity: 0.7,
3  strokeWeight: 5,
4  icons:[{
5    icon:{ //przerywana linia
6      path: 'M 0, -1 0,1',
7      strokeOpacity: 0.1,
8      strokeColor: black,
9      scale: 4
10     },
11     offset: '0',
12     repeat: '20px'
13   }],
14
15
```

Listing. 2.4. Przykład stylizacji obiektu Polyline z wykorzystaniem SVG do stworzenia efektu przerywanej Lini.



Rys 2.4. Przykład obiektu Polyline z wykorzystaniem SVG.

2.5 Obsługa zdarzeń.

Jednym z najważniejszych zagadnień dotyczących obiektów jest obsługa zdarzeń. Dzięki niej definiuje się sposób, w jaki elementy znajdujące się na mapie będą się zachowywać względem siebie oraz w jaki sposób będzie z nich korzystał użytkownik. Przykładem obsługi zdarzeń jest reakcja markera na kliknięcie i wyświetlenie informacji w nim zawartych w postaci okienka infoWindow lub dodanie współrzędnych klikniętego punktu do ścieżki. Głównym i podstawowym zdarzeniem jest bezpośrednie kliknięcie w mapę. Zdarzenie to obsługuje za pomocą atrybutu onPointClick, który umieszcza się w komponencie GMap. W przypadku kliknięcia w mapę wykonywana jest funkcja w kodzie skryptu JavaScript, podana jako argument atrybutu onPointClick. Pozostałe zdarzenia dotyczące zarówno samej mapy jak i wszelkich obiektów znajdujących się na niej, takie jak markery czy linie, można obsłużyć bezpośrednio z poziomu kodu JS. Żeby konkretny obiekt reagował na wybraną akcję, którą wykonuje użytkownik, wystarczy przypisać ją do tego obiektu w sposób, który pokazałem na listingu 2.6.

Przykładowe zdarzenia:

- click – reakcja na kliknięcie w obiekt.
- rightclick - reakcja na kliknięcie prawym przyciskiem myszy.
- dblclick - podwójne kliknięcie.

- center_changed – zdarzenie przesunięcia widoku mapy.
 - mouseout - najechanie wskaźnikiem myszy na obiekt.
 - mouseover - zdjęcie wskaźnika myszy z obiektu.

```
1 <p:gmap id="Gmap"
2     widgetVar="mapVar"
3     center="51.246, 22.541"
4     zoom="10"
5     type="ROADMAP"
6     onPointClick="GmapClickEventHandler(event);"
7
8 </p:gmap>
```

```
1 function GmapClickEventHandler(event){  
2     window.alert("lat: "+event.latLng.lat());  
3     window.alert("lng: "+event.latLng.lng());  
4     //dalsze instrukcje  
5 }  
6  
7  
8
```

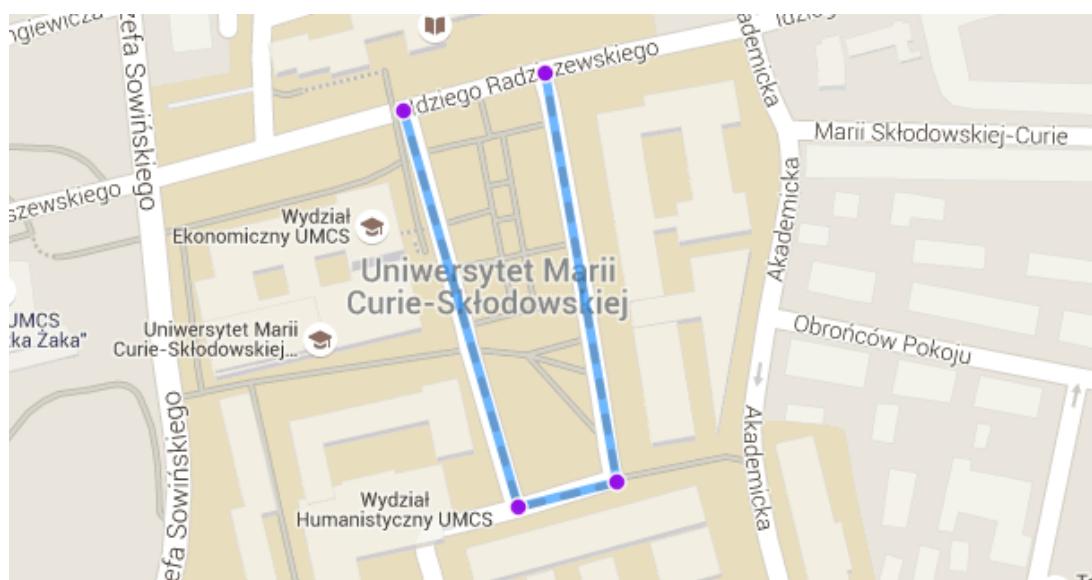
Listing 2.5. Przykład obsługi zdarzenia kliknięcia w mapie.

```
1 google.maps.event.addListener(marker, 'click', function(mouseEvent){  
2     window.alert("Tytuł markera: "+this.getTitle());  
3     //instrukcje...  
4 }  
5  
6 }  
7 }
```

Listing 2.6. Przykład obsługi kliknięcia w marker.

Rozdział 3 - Ścieżki podróży

Wszystkie opisane do tej pory przez mnie elementy i obiekty biorą mniejszy lub większy udział w procesie wyznaczania trasy podróży. Nie umniejszając ich wartości, głównym elementem tras są po prostu ścieżki ciągnące się pomiędzy poszczególnymi przystankami utworzonej wycieczki. Bez względu na sposób ich wygenerowania ścieżki reprezentowane są przez obiekt linii prostej – Polyline [10]. O polilini wspomniałem przy omawianiu obiektów overlay. Główną właściwością polilinii jest Path. Jest to zbiór współrzędnych punktów na mapie, które definiują bieg ścieżki. Ścieżkę polyline można wyznaczyć ustawiając kolejne współrzędne punktów np. klikając w mapę, oczywiście wymaga to zdefiniowania konkretnej obsługi zdarzeń dla takich kliknięć.



Rys. 3.1 Przykład polilinii składającej się z czterech punktów.

Wyznaczanie trasy w sposób manualny, czyli określanie pojedynczo kolejnych współrzędnych ścieżki daje pewne rezultaty i w niektórych przypadkach bardzo się przydaje. Aczkolwiek w przypadku bardzo długich tras, które mogą rozciągać się na tysiące kilometrów, w których ścieżka powinna być dokładna i wyznaczona zgodnie z pewnymi zasadami i preferencjami użytkownika trzeba skorzystać z innego rozwiązania jakim jest DirectionService.

3.1 DirectionService

DirectionService jest usługą dostępną w Google Maps API, która na podstawie odpowiednio zbudowanego zapytania (request), zwraca szczegółowe informacje na temat trasy przejazdu z punktu „a” do punktu „b” (response) [11]. Request przesyłany do usługi początkowo jest obiektem JavaScript zamienianym na tekstowy format JSON [12]. Zapytanie składa się z kilku atrybutów określających preferencje użytkownika i szczegółów ścieżki. Serwis w odpowiedzi zwraca status zapytania oraz dane, które również są w postaci JSON'a. Zawiera on takie informacje jak: lista współrzędnych trasy podróży, informacje o długości ścieżki, średnim czasie przejazdu, czy nawet szczegółowe instrukcje dojazdu. Czasem zdarza się, że DirectionService zwróci status negatywny. Oznacza to, że nie udało się dopasować tras do przesłanych danych. Najczęstszym błędem są niepoprawne współrzędne (np. lokalizacja na środku oceanu), lub przekroczenie ilości zapytań do serwisu tak zwany OVER_QUERY_LIMIT.

Podstawowe parametry zapytania:

- origin – współrzędne startowego punktu podróży.
- destination – punkt docelowy.
- unitSystem – preferowana jednostka miary w jakiej mają być przedstawione informacje.
- travelMode - preferowany rodzaj transportu jakim podróżuje użytkownik [21], do dyspozycji są: DRIVING (domyślny tryb), BICYCLING, TRANSIT oraz WALKING.
- provideRouteAlternatives – możliwość generowania tras alternatywnych.
- avoidHighways - unikanie autostrad.
- avoidTolls – unikanie płatnych dróg.

```
1 var request = {  
2   origin: startPoint,  
3   destination: endPoint,  
4   avoidTools: true,  
5   unitSystem: google.maps.UnitSystem.METRIC,  
6   travelMode: google.maps.TravelMode.DRIVING  
7 }  
8 }
```

Listing 3.1. Przykład zapytania do serwisu DirectionService.

```

1 var directionService = new google.maps.DirectionService();
2
3 directionService.route(request, function (response, status){
4
5     if(status === google.maps.DirectionService.OK){
6         //instrukcje
7     }
8     else{
9         window.alert("Status błędu: "+status);
10    }
11 }

```

Listing 3.2. Implementacja i przykład użycia DirectionService.

```

"routes":[{
  "bounds": {
    "south": 51.246170000000006,
    "west": 22.54091,
    "north": 51.24649,
    "east": 22.54303},
    "copyrights": "Dane do Mapy ©2016 Google",
    "legs": [
      {
        "distance": { "text": "0,2 km", "value": 154},
        "duration": { "text": "1 min", "value": 23},
        "end_address": "Akademicka 4, 20-400 Lublin, Polska",
        "end_location": { "lat": 51.2464551, "lng": 22.543026800000007 },
        "start_address": "Idziego Radziszewskiego 11, 20-400 Lublin, Polska",
        "start_location": { "lat": 51.2461749, "lng": 22.54091440000002 },
        "steps": [
          {
            "distance": { "text": "0,2 km", "value": 150},
            "duration": { "text": "1 min", "value": 22 },
            "end_location": { "lat": 51.2464904, "lng": 22.54301099999999 },
            "polyline": { "points": "q_xwHuoqhCAIMuAMiAa@yF" },
            "start_location": { "lat": 51.2461749, "lng": 22.54091440000002 },
            "travel_mode": "DRIVING",
            "encoded_lat_lngs": "q_xwHuoqhCAIMuAMiAa@yF",
            "path": [
              { "lat": 51.24617000000006, "lng": 22.54091 },
              { "lat": 51.24618, "lng": 22.540960000000002 },
              { "lat": 51.24625, "lng": 22.541390000000003 },
              { "lat": 51.246320000000004, "lng": 22.541760000000004 },
              { "lat": 51.24649, "lng": 22.543010000000002 }
            ],
            "lat_lngs": [
              { "lat": 51.24617000000006, "lng": 22.54091 },
              { "lat": 51.24618, "lng": 22.540960000000002 },
              { "lat": 51.24625, "lng": 22.541390000000003 },
              { "lat": 51.246320000000004, "lng": 22.541760000000004 },
              { "lat": 51.24649, "lng": 22.543010000000002 }
            ],
            "instructions": "Kieruj się <b>Idziego Radziszewskiego</b> na <b>plac Marii Curie-Skłodowskiej</b>",
            "maneuver": "",
            "start_point": { "lat": 51.2461749, "lng": 22.54091440000002 },
            "end_point": { "lat": 51.2464904, "lng": 22.543010999999998 }
          }
        ]
      }
    ]
  }
}

```

```

{
  "distance": {"text": "4 m", "value": 4},
  "duration": {"text": "1 min", "value": 1},
  "end_location": {"lat": 51.2464551, "lng": 22.543026800000007},
  "maneuver": "turn-right",
  "polyline": {"points": "qaxwHy/qhCDC"},
  "start_location": {"lat": 51.2464904, "lng": 22.54301099999998},
  "travel_mode": "DRIVING",
  "encoded_lat_lngs": "qaxwHy/qhCDC",
  "path": [
    {"lat": 51.24649, "lng": 22.543010000000002},
    {"lat": 51.246460000000006, "lng": 22.54303}
  ],
  "lat_lngs": [
    {"lat": 51.24649, "lng": 22.543010000000002},
    {"lat": 51.246460000000006, "lng": 22.54303}
  ],
  "instructions": "Skręć <b>w prawo</b> w <b>Akademicka</b>",
  "start_point": {"lat": 51.2464904, "lng": 22.54301099999998},
  "end_point": {"lat": 51.2464551, "lng": 22.543026800000007}
},
],
"traffic_speed_entry": [],
"via_waypoint": [],
"via_waypoints": []
},
],
"overview_polyline": "q xwHuqhqC0 BMiAa@yFDC",
"summary": "Idziego Radziszewskiego",
"warnings": [],
"waypoint_order": [],
"overview_path": [
  {"lat": 51.246170000000006, "lng": 22.54091},
  {"lat": 51.24625, "lng": 22.541390000000003},
  {"lat": 51.246320000000004, "lng": 22.541760000000004},
  {"lat": 51.24649, "lng": 22.543010000000002},
  {"lat": 51.246460000000006, "lng": 22.54303}
]
],
"status": "OK",
"request": {
  "origin": {"lat": 51.24613875111497, "lng": 22.54093050956726},
  "destination": {"lat": 51.24645440175848, "lng": 22.543022632598877},
  "unitSystem": 0,
  "travelMode": "DRIVING"
}
}

```

Listing 3.3 Przykładowy JSON zwrotny z serwisu DirectionService.

3.2 DirectionsRenderer

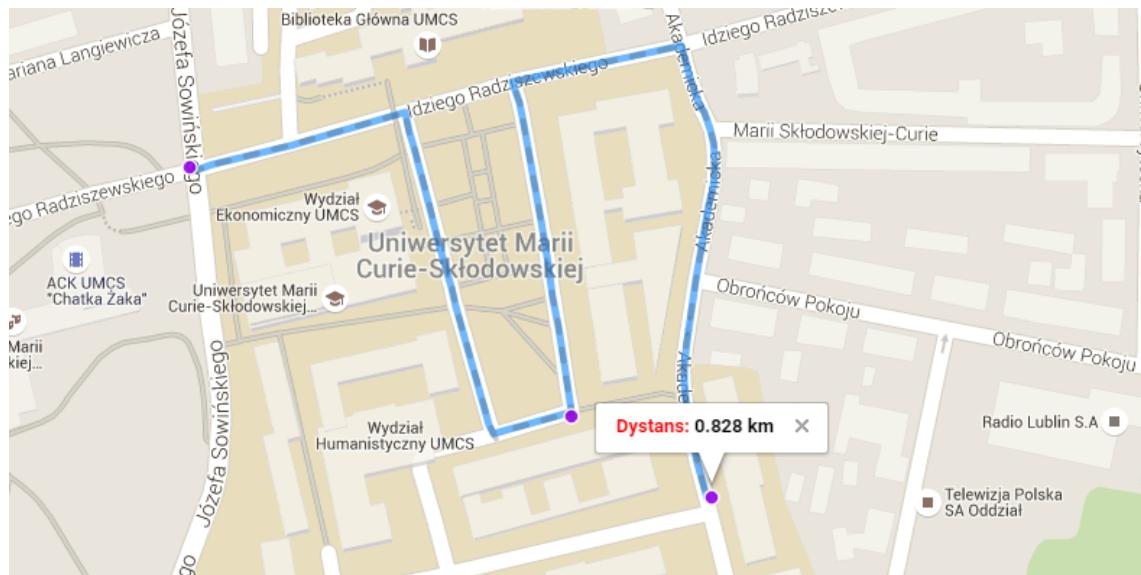
Jak można było zauważyć, JSON z informacjami otrzymanymi jako odpowiedź z DirectionService, jest bardzo rozbudowany i przy większej ilości danych trudno jest przetworzyć wszystkie zawarte w nim informacje. Całe szczęście biblioteka Google Maps API udostępnia klasę DirectionsRenderer [20]. Wykonanie metody setDirections() i podanie jako parametr, danych zwrotnych z serwisu DirectionService, powoduje wyrysowanie na mapie rozbudowanej polilini, której wygląd i atrybuty można określić w konstruktorze. Dodatkowo DirectionsRenderer ma możliwość wyświetlenia w określonym wcześniej bloku <div> </div> szczegółowych wskazówek dojazdu w postaci listy.

```

1 var directionsDisplay = new google.maps.DirectionsRenderer({
2
3     map: mapa,
4     polylineOptions:{
5         strokeColor: '#3399FF',
6         strokeOpacity: 0.7,
7         strokeWeight: 5,
8         icons: [
9             {
10                icon: {
11                    path: 'M 0,-1 0,1',
12                    strokeColor: 'black',
13                    scale: 4
14                },
15                offset: '0',
16                repeat: '20px'
17            }
18        },
19        draggable: false,
20        suppressMarkers: true
21    },
22    panel: document.getElementById("directionsDIV")
23);
24
25 directionsService.route(request, function (response, status) {
26
27     if (status === google.maps.DirectionsStatus.OK) {
28
29         directionsDisplay.setDirections(response);
30     }
31 });
32 });

```

Listing 3.4. Przykład użycia DirectionRenderer.



Rys 3.2. Przykładowa ścieżka składająca się z 3 punktów. Wygenerowana za pomocą DirectionService i wyrysowana na mapie przez DirectionsRenderer.



Idziego Radziszewskiego 14, 20-400 Lublin, Polska

0,3 km. Około 2 min

1. Kieruj się Idziego Radziszewskiego na wschód w stronę plac Marii Curie-Skłodowskiej 0,1 km
- ▶ 2. Skręć w prawo w plac Marii Curie-Skłodowskiej 0,2 km
- ◀ 3. Skręć w lewo, pozostając na plac Marii Curie-Skłodowskiej 42 m
Miejsce docelowe będzie po prawej stronie



plac Marii Curie-Skłodowskiej 1, 20-400 Lublin, Polska

Dane do Mapy ©2016 Google



plac Marii Curie-Skłodowskiej 1, 20-400 Lublin, Polska

0,5 km. Około 2 min

1. Kieruj się plac Marii Curie-Skłodowskiej na północ 0,2 km
- ▶ 2. Skręć w prawo w Idziego Radziszewskiego 89 m
- ▶ 3. Skręć w prawo na 1 skrzyżowaniu w: Akademicka 0,2 km
Miejsce docelowe będzie po lewej stronie

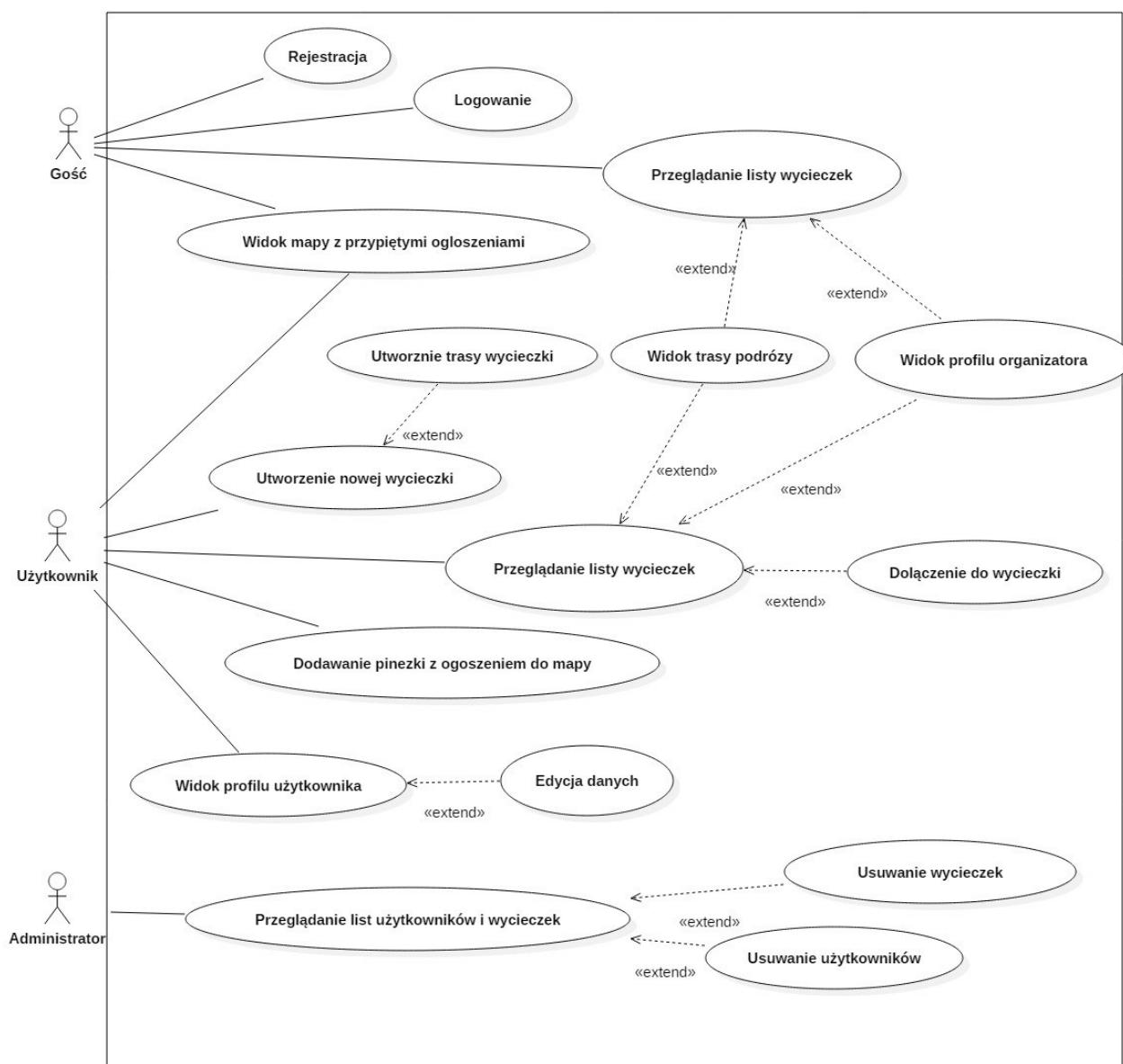


Akademicka 11, 20-033 Lublin, Polska

Rys. 3.3. Wskazówki dojazdu wyświetcone za pomocą DirectionsRenderer.

Rozdział 4 – Aplikacja internetowa – Portal dla podróżników

Żeby pokazać praktyczne wykorzystanie komponentu mapy oraz możliwości jakie dostarcza Google Maps API, stworzyłem aplikację internetową skierowaną dla podróżników oraz osób, które chcieliby wyruszyć w podróż ale nie mają z kim. Głównym celem aplikacji jest organizowanie nowych podróży oraz wyznaczanie szczegółowych tras i planów wycieczek. Użytkownicy mają także możliwość wyszukiwania i dołączania do wycieczek utworzonych przez innych.



Rys 4.1. Diagram przypadków użycia.

4.1 Proces dodawania nowej wycieczki.

Wybierając w głównym menu zakładkę „Kreator podróży” użytkownik zostaje przeniesiony do sekcji, w której ma możliwość utworzenia swojej własnej wycieczki. Proces dodawania nowej podróży podzielił się na trzy etapy składające się z prostych formularzy. W pierwszej części użytkownik musi podać podstawowe informacje takie jak:

- Unikatowa nazwa wycieczki.
- Miejsce docelowe - kraj lub miasto.
- Główny typ transportu jakim będzie podróżować grupa.
- Cel podróży np. Zwiedzanie, wypoczynek czy uprawianie sportów.

The screenshot shows the 'SamNieJade.pl' website interface. At the top, there's a navigation bar with links: STRONA STARTOWA, TWÓJ PROFIL, KREATOR PODRÓŻY, OZNACZ SIĘ NA MAPIE, LISTA WYCIECZEK, MAPA OZNACZONYCH OSÓB, and PANEL ADMINISTRATORA. The 'KREATOR PODRÓŻY' tab is active. On the left, there's a sidebar with a 'Cofnij' (Cancel) button. The main content area is titled 'Kreator nowej wycieczki'. It has three tabs: 'Podstawowe informacje' (selected), 'Wymogi i zastrzeżenia', and 'Czas i miejsce'. Under 'Podstawowe informacje', there are input fields for 'Nazwa' (Name) and 'Miejsce docelowe' (Destination). Below these is a dropdown menu for 'Główny rodzaj Transportu' (Main type of transport), which is currently set to 'Rodzaj transportu'. A list of other transport types is visible in the dropdown: Piesza wędrówka, Rowerem, Samochodem, Motocyklem, Samolotem, Statkiem, Komunikacja publiczna, "Na Stopa", and Inne. At the bottom right of the dropdown menu, there's a 'Dalej' (Next) button.

Rys. 4.2. Etap pierwszy – Podstawowe informacje.

Etap drugi ma na celu określenie pewnych wymogów od uczestników podróży. Znajduję się tu:

- Średni koszt podróży.
- Cechy, umiejętności uczestników np. jazda na nartach.
- Uwagi i zastrzeżenia, np. jeśli organizator nie chce podróżować z osobami palącymi, może to wpisać w tym miejscu.
- Maksymalna ilość osób w grupie.

- Przedział wiekowy.
- Preferowana płeć uczestników.

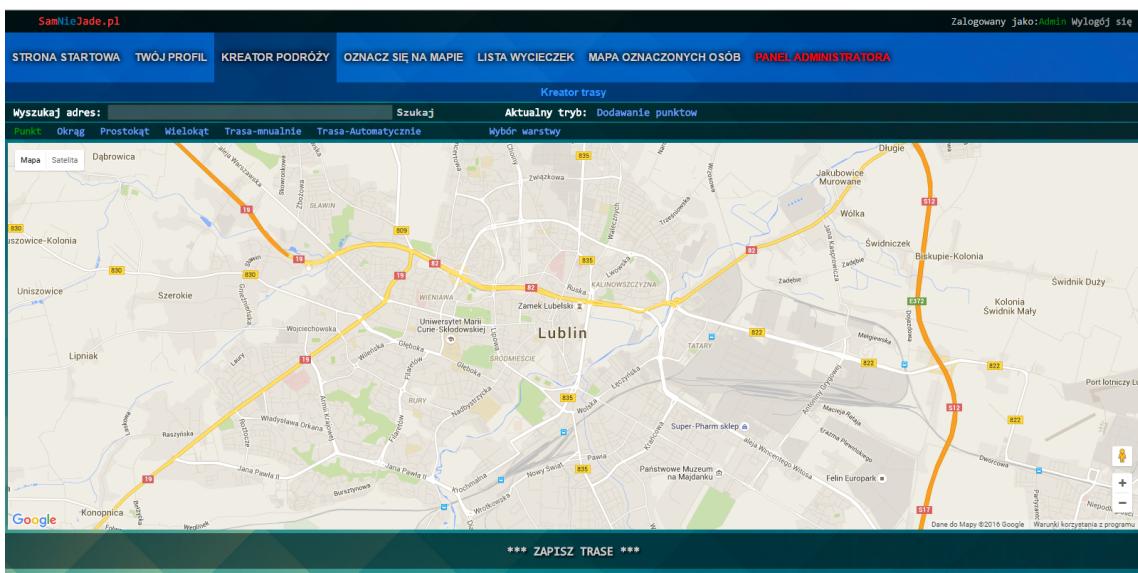
Rys 4.3. Etap drugi – Wymogi i zastrzeżenia.

W ostatnim etapie trzeba podać takie informacje jak: data podróży, czas trwania, lub przybliżony dzienny dystans do przebycia np. jeżeli będzie to podróż samochodem. W tym miejscu można wykreować własną trasę podróży lub zakończyć tworzenie wycieczki na tym etapie.

Rys 4.4. Trzeci etap tworzenia wycieczki.

4.2 Pierwsze spojrzenie na stronę kreowania tras.

Wyznaczanie trasy jest głównym celem aplikacji. Żeby przejść do tego modułu trzeba mieć już za sobą proces dodawania wycieczki i wybrać opcję kreowania trasy. Głównym elementem strony jest oczywiście komponent mapy GMap, na którym zostaną wykonane wszystkie operacje.



Rys 4.5. Widok strony kreatora trasy.

4.3 Geocoding i Reverse Geocoding.

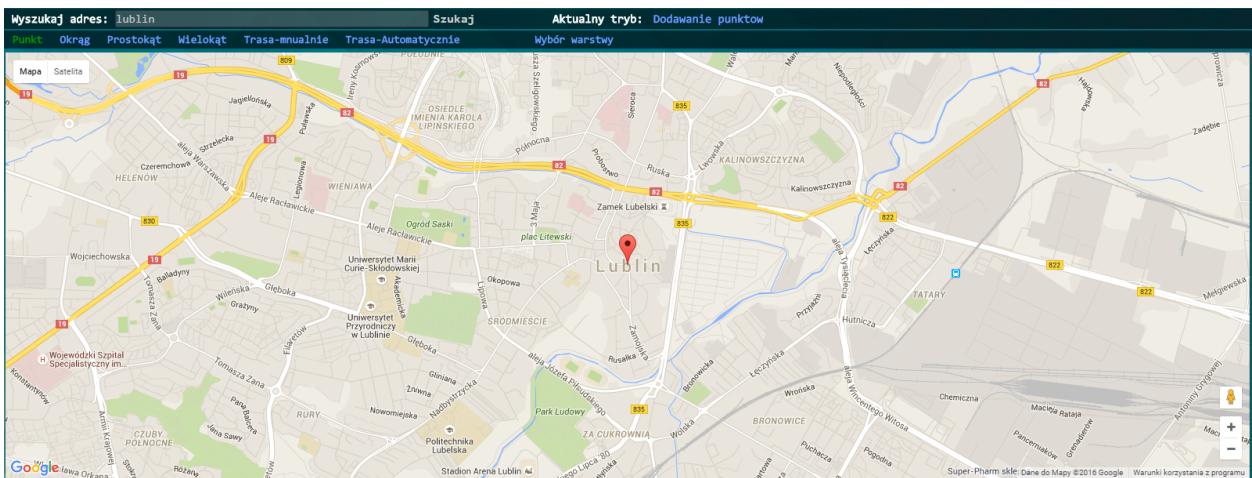
Pierwszym od góry panelem jaki znajduje się na stronie jest wyszukiwarka adresów. W pole wejściowe wystarczy wpisać poszukiwany adres, może to być państwo, miasto, ulica z numerem domu lub nawet nazwa instytucji np. Lublin, UMCS. W rezultacie szukana lokalizacja pojawi się w centrum mapy, zaznaczona markerem. Wyszukiwarka działa w oparciu o mechanizm Geokodowania (Geocoding), który dostępny jest z bibliotek Google Maps [19].



Rys. 4.6 Panel wyszukiwania

Geokodowanie jest mechanizmem zamieniającym podane dane adresowe na współrzędne - wysokość i szerokość geograficzną (lat, lng), które zazwyczaj znajdują się odnalezione na mapie. Czasami zdarzy się, że system nie odnajdzie poszukiwanej lokalizacji ze względu na błędne współrzędne, lub brak adresu w bazie, podobnie jak w przypadku DirectionService. Istnieje również mechanizm przeciwny do geokodowania czyli ReverseGeocoding, który na podstawie wprowadzonych współrzędnych geograficznych generuje dokładne informacje adresowe. W mojej aplikacji wykorzystuję oba rodzaje geokodowania. Pierwszy we wspomnianej wyszukiwarce, natomiast odwrotne geokodowanie znajduje zastosowanie w wyszukiwaniu danych adresowych, które dodaje kolejno jako informacje w markerach i innych punktach na mapie.

Żeby skorzystać z dobrodziejstw geokodowania trzeba przede wszystkim utworzyć obiekt klasy google.maps.Geocoder() i wywołać metodę geocode() z odpowiednimi parametrami. Metodę można użyć na dwa sposoby, pierwszy z nich to przekazanie w parametrach atrybutu „address” i jako wartość dowolne informacje adresowe w postaci zwykłego tekstu (Geocoding), natomiast drugi sposób to użycie atrybutu „location”, którego wartością są współrzędne geograficzne jako obiekt klasy google.maps.LatLng (Reverse Geocoding).



Rys. 4.7. Wyszukanie adresu – Lublin

```

1 function codeAddress() {
2     var address = $("#searchPanel\\:wyszukajAdresId").val();
3     var Geocoder = new google.maps.Geocoder();
4
5     Geocoder.geocode( { 'address': address}, function(results, status) {
6         if (status === google.maps.GeocoderStatus.OK){
7             var Marker = new google.maps.Marker({
8                 position: results[0].geometry.location,
9                 map: mapa
10            });
11
12            google.maps.event.addListener(Marker, 'mouseover', function() {
13                Marker.setMap(null);
14            });
15
16            Gmap.setCenter(results[0].geometry.location); //centruje mapę
17        }
18        else {
19            if(status === "OVER_QUERY_LIMIT"){ //OVER_QUERY_LIMIT
20                window.alert('LIMIT USŁUGI GEOKODINGU' + status);
21            }
22            else{
23                window.alert('WYSTĄPIŁ BŁAD: ' + status);
24            }
25        }
26    });
27 }
28
29 }
30
31

```

Listing 4.1. Implementacja funkcji obsługującej geocoding.

```

1 function geocodeLatLng(SzukanyAdreslatlng) {
2     var Geocoder = new google.maps.Geocoder();
3
4     Geocoder.geocode({ 'location': SzukanyAdreslatlng}, function(results, status) {
5         if (status === google.maps.GeocoderStatus.OK) {
6             if (results[1]) {
7                 marker.customInfo.adres = results[0].formatted_address;
8             }
9             else {
10                 window.alert('NIE ZNALEZIONO ADRESU');
11             }
12         }
13         else {
14             if(status === "OVER_QUERY_LIMIT"){ //OVER_QUERY_LIMIT
15                 window.alert('LIMIT USŁUGI GEOKODINGU' + status);
16             }
17             else{
18                 window.alert('WYSTĄPIŁ BŁAD: ' + status);
19             }
20         }
21     });
22 }
23
24
25
26

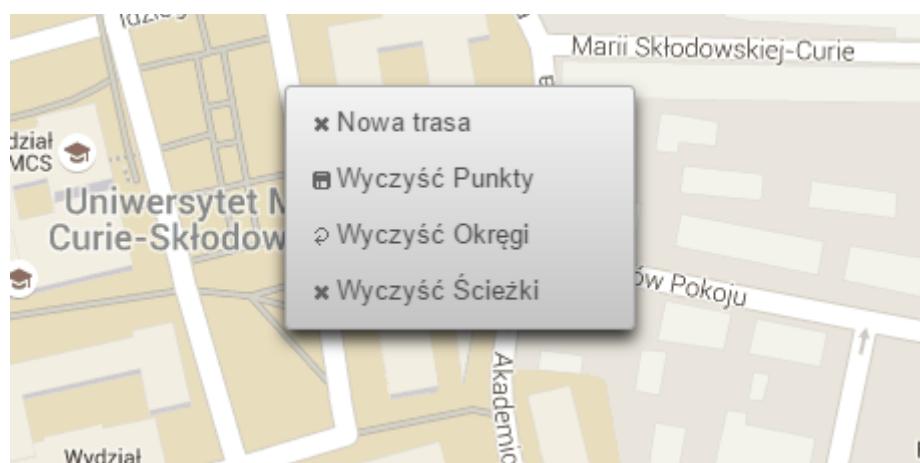
```

Listing 4.2. Implementacja Reverse Geocoding.

4.4 Główny panel zarządzania obiektami Overlay.

Zaraz pod panelem wyszukiwania znajduję się główny panel, umożliwiający użytkownikowi przełączanie się pomiędzy trybami „rysowania” obiektów overlay. Wystarczy kliknąć w wybrany tryb i od razu można przystąpić do umieszczania obiektów na mapie. Do dyspozycji są:

- Markery czyli punkty określające szczególne miejsca takie jak początek i koniec trasy, miejsca postoju, konkretne budynki itp.
- Obiekty zaznaczania obszarów (Circle, Rectangle i Polygon). Służą one do zaznaczania granic budynków, parków i innych miejsc w zależności od potrzeb użytkownika.
- Trasa-manulanie – jest to opcja, pozwalająca na rysowanie ścieżki składającej się z pojedynczych elementów Polyline. Kolejne składowe ścieżki można dodawać klikając w mapę w już dodane markery, wtedy ścieżka dołączy się do tego punktu. Podczas rysowania trasy, do każdego punktu pośredniego zapisują się informacje o długości trasy(w kilometrach). Żeby przerwać tworzenie jednej ścieżki i zacząć rysowanie kolejnej niezależnej, można wybrać opcje „Nowa trasa” w menu kontekstowym mapy (wywołanie poprzez kliknięcie prawym przyciskiem myszy), lub można usunąć wszystkie dodane ścieżki wybierając opcje „Wyczyszc ścieżki”.



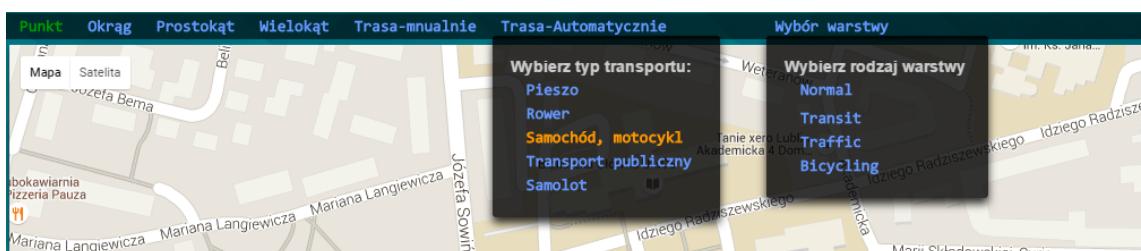
Rys. 4.8. Menu kontekstowe mapy.

- Trasa-Automatyczna – Ten tryb rysowania ścieżki, korzysta z DirectionService. Trasa dodaje się automatycznie, jedyne co musi zrobić użytkownik to określenie punktu startowego i docelowego. Można to wykonać w sposób identyczny jak to było w przypadku trasy-manulanej, wystarczy kliknąć w odpowiedni punkt na mapie lub marker czy inny obiekt. Tryb trasy-automatycznej rozbudowany jest o wybór rodzaju transportu, (domyślnym trybem jest DRIVE). W zależności od wybranego typu, ścieżka będzie rysowana w odpowiedni sposób np. wybierając tylko ścieżki rowerowe,



Rys. 4.9. Menu wyboru typu transportu.

Na panelu głównym znajduje się również menu wyboru warstw mapy. Warstwy zostały opisane w rozdziale pierwszym. W zależności od wyboru, mapa zostanie pokryta odpowiednią warstwą co w moim przekonaniu pomoże użytkownikowi w kreowaniu odpowiedniej trasy np. omijając najbardziej ruchliwe drogi.



Rys. 4.10. Widok panelu z opcjami Obiektów overlay.

4.5 Wyszukiwanie i dołączanie do wycieczki.

Na głównym panelu nawigacyjnym aplikacji znajduje się zakładka „Lista Wycieczek”. Przenosi ona do sekcji, w której użytkownik ma możliwość przeglądania listy wycieczek utworzonych przez innych. Widok pojedynczej wycieczki podzieliłem na trzy bloki, żeby dane wycieczki były lepiej widoczne dla przeglądającego. Znajdują się tu wszystkie informacje, które organizator wprowadził w kreatorze podróży. Najbardziej wyeksponowaną częścią jest informacja o ilości wolnych miejsc w wycieczce. Poniżej znajduje się możliwość dołączenia do listy uczestników. Zapisać się mogą jedynie zalogowani użytkownicy, natomiast przeglądać wycieczki może każdy. Nad listą wycieczek znajduje się panel z opcjami umożliwiającymi posortowanie listy, tak żeby użytkownik mógł szybciej wyszukać wycieczkę dopasowaną do swoich potrzeb. Istnieje również możliwość pokazania tylko tych wycieczek, w których są wolne miejsca. Obok menu sortowania znajduje się panel wyszukiwania wycieczki względem nazwy, miejsca docelowego i celu podróży.

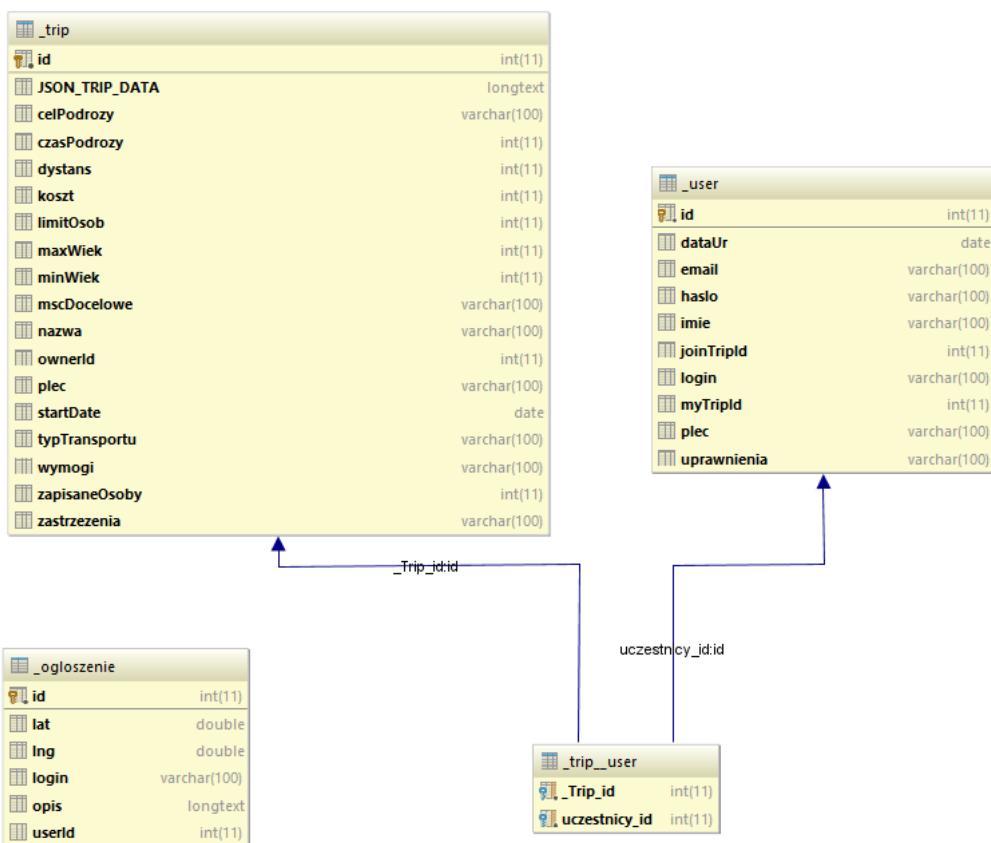
Lista wycieczek		
Klucz sortowania		
Wyszukaj wycieczkę		
Wycieczka: Nazwa: Wycieczka na narty Zalożyciel: Admin Miejsce docelowe: Niemcy, Główny cel: Sporty śnieżne Transport: Samochodem Czas trwania: 7 dni Szacowany koszt: 3000 zł Zobacz profil założyciela: Admin Pokaż trase wycieczki	Uczestnicy: Wiek od 18 do 35 Płeć: Bez znaczenia Wymogi: Jazd na na nartach lub snowboardzie Zastrzeżenia: Brak Zapisane osoby: kinga,	Data rozpoczęcia: 2016-06-30 Wolne miejsca: 9/10 Dołącz jako: pawel
Wycieczka: Nazwa: Wyjazd do ameryki Zalożyciel: pawel Miejsce docelowe: USA, Nowy York Główny cel: Zwiedzanie Transport: Samolotem Czas trwania: 0 dni Szacowany koszt: 5000 zł Zobacz profil założyciela: pawel Pokaż trase wycieczki	Uczestnicy: Wiek od 18 do 40 Płeć: Bez znaczenia Wymogi: Brak Zastrzeżenia: Brak Zapisane osoby: Brak użytkowników	Data rozpoczęcia: 2016-08-02 Wolne miejsca: 40/40 Dołącz jako: pawel
Wycieczka: Nazwa: Wyjazdówka Zalożyciel: kinga Miejsce docelowe: Wrocław Główny cel: zwiedzanie, odpoczynek, rejs statkiem Transport: Samochodem Czas trwania: 4 dni Szacowany koszt: 500 zł Zobacz profil założyciela: kinga Pokaż trase wycieczki	Uczestnicy: Wiek od 20 do 30 Płeć: Bez znaczenia Wymogi: Jedna osoba z prawkiem kat. B Zastrzeżenia: Osoby niepalące Zapisane osoby: Admin, pawel,	Data rozpoczęcia: 2016-06-22 Wolne miejsca: 2/4 Dołącz jako: pawel

Rys. 4.11. Widok listy wycieczek.



Rys. 4.12. Widok panelu wyboru klucza sortowania listy i wyszukiwania.

4.6 Kilka słów na temat zapisu danych mapy.



Rys. 4.13. Widok schematu bazy danych.

Wszystkie dane aplikacji przechowywane są w bazie MySQL. Natomiast do komunikacji z bazą wykorzystuję szkielet aplikacji java - Hibernate. Obie te technologie współpracują ze sobą znakomicie co bardzo ułatwia pracę z danymi. Odwzorowanie klas encyjnych dokonywane jest w za pomocą adnotacji JPA (Java Persistence API)

bezpośrednio w ciele tych klas, bez konieczności tworzenia kolejnych plików mapujących [23].

4.6.1 JSON jako format zapisu danych.

Budowanie trasy wiąże się z dodawaniem bardzo dużej ilości różnych elementów i informacji do mapy. Warstwa danych mapy musi być elastyczna, tak żeby mogła być z łatwością naniesiona z powrotem na mapę, gdyż jakakolwiek chęć odtworzenia trasy wiąże się koniecznością wyrysowania wszystkich obiektów. Jako, że logika działania komponentu GMap napisana jest w technologii JavaScript, bardzo dobrym formatem do zapisu i odczytu danych jest wspomniany w poprzednich rozdziałach format JSON. Większość elementów mapy przechowywanych jest początkowo w listach obiektów JavaScript, więc transformacja do formatu JSON jest bardzo prosta, wystarczy użyć wbudowanej metody `JSON.stringify()`.

```
1 | var textJSON = JSON.stringify(ListaMarkerow);  
2 |  
3 |
```

Listing 4.3. Przykład transformacji listy markerów do pliku tekstowego w formacie JSON.

JSON - (JavaScript Object Notation) – tekstowy format zapisu danych, który odzwierciedla strukturę obiektów tworzonych w JS, jednakże jest to format niezależny od technologii i może być równie dobrze używany np. w Java, czy C++. W tym formacie można zapisać zarówno pojedyncze obiekty, jak i całe listy bardzo rozbudowanych obiektów.

```
1 | {  
2 |   "dane": {  
3 |     "user": [  
4 |       {  
5 |         "imie": "Jan",  
6 |         "nazwisko": "Kowalski"  
7 |       },  
8 |       {  
9 |         "imie": "Piotr",  
10 |        "nazwisko": "Nowak"  
11 |      }  
12 |    }  
13 | }  
14 | }  
15 | }
```

Listing 4.4. Przykład danych użytkownika w formacie JSON [17].

4.6.2 Kodowanie współrzędnych

W rozdziale o obiektach overlay wspomniałem, że dane ścieżek, a dokładniej zbiór punktów określających bieg trasy przechowywany jest jako zmienna Path. Path może być zwykłą listą ArrayList lub google.maps.MVCArray, przechowującą poszczególne współrzędne jako obiekty reprezentowane przez klasę google.maps.LatLng(). W przypadku gdy ścieżka jest bardzo dłuża, lista współrzędnych również znacząco się rozrasta zajmując dużo miejsca w pamięci. W bibliotece Google Maps API znajduje się funkcjonalność, która bardzo ułatwiła mi prace z „rozrośniętymi” listami i znacznie ograniczyła zajmowane przez nie zasoby pamięci. Mowa o funkcji kodującej współrzędne [24]. Funkcja otrzymuje jako parametr listę współrzędnych i zwraca zakodowany ciąg znaków ASCII [25]. Wówczas otrzymany tekst można o wiele łatwiej zapisać w bazie danych. W bibliotece Google Maps API znajduje się również funkcja odwrotna, zamieniająca zakodowany tekst na listę współrzędnych.

```
1 function encodePath(path){  
2     return google.maps.geometry.encoding.encodePath(path);  
3 }  
4  
5
```

Listing 4.5. Implementacja funkcji kodującej.

```
1 function decodePath(path){  
2     return google.maps.geometry.encoding.decodePath(path);  
3 }  
4  
5
```

Listing 4.6. Implementacja funkcji dekodującej.

```
1 "path": [  
2     {"lat": 51.24617000000006, "lng": 22.54091},  
3     {"lat": 51.24625, "lng": 22.54139000000003},  
4     {"lat": 51.24632000000004, "lng": 22.54176000000004},  
5     {"lat": 51.24649, "lng": 22.54301000000002},  
6     {"lat": 51.24646000000006, "lng": 22.54303}  
7 ]  
8
```

```
1 "q_xwHuoqhCO_BMiAa@yFDC"  
2  
3
```

Listing 4.7. Trasa ścieżki przed i po zakodowaniu.

Podsumowanie

Aplikacja, którą przedstawiłem, pomimo wielu niedoskonałości i drobnych błędów, spełnia większość początkowych założeń. W rezultacie użytkownicy mają możliwość tworzenia nowych wycieczek oraz co najważniejsze mogą wyznaczać szczegółowe trasy i plany podróży. Osoby, które obawiają się lub po prostu nie chcą organizować własnych podróży, mogą skorzystać z wyszukiwarki wycieczek i dołączać do tych, które najbardziej spełniają ich wymogi.

Myślę że przedstawiona aplikacja spodobałby się wielu osobom lubiącym podróżować, ze względu na możliwość kreowania własnych tras oraz brak podobnych projektów w internecie. W przyszłości planuję również rozbudować aplikację o kilka nowych funkcjonalności. Między innymi chciałbym dodać panel zarządzania wycieczkami, który umożliwiłby organizatorom kontrolę nad przebiegiem podróży i polepszyć komunikację pomiędzy uczestnikami wycieczek a organizatorami. Planuję również udostępnić aplikację, umieszczając ją na publicznym serwerze.

Bibliografia

1. David Sawyer McFarland, „JavaScript i jQuery: nieoficjalny podręcznik”. Helion, Gliwice 2012.
2. jQuery - <https://pl.wikipedia.org/wiki/JQuery> [dostęp 30.05.2016]
3. PrimeFaces – <http://primefaces.org/> [dostęp 30.05.2016]
4. Gabriel Svennerberg, „Beginning Google Maps API 3”, 2010
Google Maps API v3 - <https://developers.google.com/maps/> [dostęp 30.05.2016]
5. Mapy Google - <https://maps.google.pl/>. [dostęp 30.05.2016]
6. Typy Map - <https://developers.google.com/maps/documentation/javascript/maptypes>
7. CSS – <https://www.w3.org/Style/CSS/> [dostęp 30.05.2016],
https://pl.wikipedia.org/wiki/Kaskadowe_arkusze_styl%C3%B3w [dostęp 30.05.2016]
8. SVG - <https://www.w3.org/Graphics/SVG/> [dostęp 30.05.2016]
9. W3C - <https://www.w3.org/> [dostęp 30.05.2016]
10. <https://developers.google.com/maps/documentation/javascript/reference#Polyline> [dostęp 30.05.2016]
11. <https://developers.google.com/maps/documentation/javascript/reference#DirectionsService> [dostęp 30.05.2016]
12. <http://www.json.org/json-pl.html> [dostęp 30.05.2016]
13. <https://developers.google.com/maps/documentation/javascript/reference#Polyline> [dostęp 30.05.2016]
14. <https://developers.google.com/maps/documentation/javascript/reference#Circle> [dostęp 30.05.2016]
15. <https://developers.google.com/maps/documentation/javascript/reference#Polygon> [dostęp 30.05.2016]
- 15 .<https://developers.google.com/maps/documentation/javascript/reference#Rectangle> [dostęp 30.05.2016]
16. <https://developers.google.com/maps/documentation/javascript/infowindows> [dostęp 30.05.2016]
17. zdjécie <http://www.yarpo.pl/2011/03/06/json-jako-format-wymiany-danych/> [dostęp 30.05.2016]

18. <https://developers.google.com/maps/documentation/javascript/reference#OverlayView> [dostęp 30.05.2016]
19. <https://developers.google.com/maps/documentation/javascript/reference#Geocoder> [dostęp 30.05.2016]
20. <https://developers.google.com/maps/documentation/javascript/reference#DirectionsRenderer> [dostęp 30.05.2016]
21. <https://developers.google.com/maps/documentation/javascript/reference#TravelMode> [dostęp 30.05.2016]
22. Hibernate – <http://iprogramujesz.pl/hibernate-dla-poczatkujacych/>,
<https://pl.wikipedia.org/wiki/Hibernate> [dostęp 30.05.2016]
23. JPA - <http://www.oracle.com/technetwork/java/javaee/tech/persistence-jsp-140049.html> [dostęp 30.05.2016]
24. EncodePath - <https://developers.google.com/maps/documentation/javascript/examples/geometry-encodings>
25. ASCII - <http://www.asciiitable.com/>
26. DOM - http://www.w3schools.com/js/js_htmldom.asp
27. Map class - <https://developers.google.com/maps/documentation/javascript/reference#release-version>