

SPDB Dokumentacja końcowa

Prawidłowa identyfikacja przystanków oraz obiektów znajdujących się wokół nich na obszarze miasta Białystok.

Kamil Bachanek

Karol Rogowski

Adam Zieliński

1. Aplikacja serwerowa

1.1. Wyznaczanie skorygowanych lokalizacji przystanków

Baza danych Systemów Dynamicznej Informacji Pasażerskiej zawiera historyczne dane dotyczące pozycji autobusów w postaci punktów w formacie WGS84[1]. Każdy punkt powiązany jest z konkretnym pojazdem oraz konkretnym dziennym kursem. Każdy punkt zawiera również informację o dacie oraz czasie, kiedy został zarejestrowany. Zazwyczaj *czas próbkowania* wynosił kilkanaście sekund. Oznacza to, że można je odnieść do faktycznego przejazdu autobusu z przeszłości oraz otrzymać informacje o tym, na jakim przystanku powinien się zatrzymywać. Dane te zawarte są w tabeli *csiptrace*. W konkretnej bazie, która była używana w trakcie projektu tabela zawierała ponad 7 milionów rekordów. Dzięki lokalizacjom punktów oraz czasom, w którym pojazd się znajdował w tych punktach, można wyznaczyć średnią prędkość pomiędzy punktami a w konsekwencji, z dość dużą pewnością, faktyczne punkty zatrzymania.

a) Algorytm wyznaczenia punktów zatrzymania

Posiadając dane dotyczące jednego przejazdu, średnią prędkość można obliczyć poprzez wyznaczenie odległości między dwoma punktami oraz czasu, który upłynął podczas przejazdu między nimi zgodnie ze wzorem:

$$V_i = \frac{S_{i,i-1}}{t_i - t_{i-1}}$$

gdzie:

- V_i – prędkość w punkcie i
- $S_{i,i-1}$ – odległość między punktami i oraz $i-1$
- t_i - czas w punkcie i
- t_{i-1} – czas w punkcie $i-1$

Odległość między dwoma punktami możemy wyznaczyć korzystając ze wzoru:

$$s_{i,i-1} = 2 * R * \sin \left(\frac{\sqrt{\cos\left(\frac{lat_{i-1} * \pi}{180}\right) * \cos\left(\frac{lat_i * \pi}{180}\right) * \left(1 - \cos\left(\frac{(lon_i - lon_{i-1}) * \pi}{180}\right)\right)}}{2} \right) + \zeta$$

Gdzie:

- R – średni promień Ziemi ($R = 6371 \text{ km}$)
- lat_i – szerokość geograficzna punktu i
- lon_i – długość geograficzna punktu i

Jak zostało wspomniane wcześniej przejazd dotyczy konkretnego kursu. Tabela *csiptrace* zawiera kolumnę *daycourseid* dzięki czemu wyszukując przejazd o konkretnym *daycourseid* jesteśmy w stanie odtworzyć trasę przejazdu autobusu a korzystając z dwóch powyższych wzorów również średnią prędkość na poszczególnych odcinkach. Wiąże się to oczywiście z pewną niedokładnością, ponieważ przy takim podejściu zakładamy, że autobus na poszczególnych odcinkach poruszał się po linii prostej, w rzeczywistości zaś mógł to być pewien łuk lub łamana. Jednak z perspektywy jednego z celów projektu, czyli znalezienia punktów zatrzymania pojazdu, nie ma to większego znaczenia. Większe znaczenie ma na pewno *czas próbkowania*. Mogła się bowiem przydarzyć sytuacja, w której autobus podczas czasu pomiędzy dwoma zarejestrowanymi punktami mógł zdążyć się zatrzymać i ruszyć pokonując jednocześnie krótki odcinek drogi. Wtedy nasza wyliczona średnia prędkość nie będzie równa zero. Jednak z perspektywy bardziej szczegółowego celu projektu, czyli wykrycie lokalizacji przystanku to również może mieć niewielkie znaczenie, ponieważ zazwyczaj autobus stoi na przystanku dłużej niż 10 sekund. Kolejną przesłanką mówiącą o tym, że ma to niewielkie znaczenie jest to, że posiadamy wystarczającą ilość przejazdów, aby te skrajne przypadki nie były bardzo istotne. Przyjęliśmy założenie, że pojazd zatrzymał się, jeżeli średnia prędkość na pewnym odcinku była mniejsza niż 0,1km/h. Jako punkt zatrzymania ustaliliśmy punkt znajdujący się w połowie odcinka pomiędzy dwoma punktami z bazy.

b) Wyznaczanie faktycznych lokalizacji przystanków

Baza danych zawiera lokalizacje przystanków ustalone w niewiadomy nam sposób. Zakładając, że faktyczne lokalizacje przystanków znajdują się w dość bliskiej odległości lokalizacji przystanków (parametr algorytmu) z bazy można je wyznaczyć poprzez porównanie z wszystkimi wyznaczonymi punktami zatrzymania. Punkty zatrzymania należałoby zaś pogrupować w obszary zatrzymania oraz zliczyć ilość zatrzymań w danym obszarze. Za kryterium wyboru faktycznej lokalizacji przystanku można wtedy przyjąć pewien sztuczny wskaźnik wyznaczony na podstawie odległości przystanku z bazy od pewnego obszaru zatrzymania oraz ilości zatrzymań pojazdów w tym miejscu. Podejście takie wiąże się jednak z kilkoma mankamentami. Przede wszystkim w pobliżu (bliżej niż wcześniej ustalone 200m) może być przystanek po drugiej stronie i może to wpływać na błędy. Podejście to jest również bardzo kosztowne. Należałoby bowiem, dla każdego przystanku (jest ich niecałe 1000) wyznaczyć odległość od wszystkich punktów zatrzymania wyznaczonych z ponad 7 milionów punktów. Z tego względu zdecydowano się na inne podejście. Dla każdego przystanku istnieje możliwość wyznaczenia

wariantów, czyli tras kursów. Dla każdego wariantu można zaś wyznaczyć konkretny kurs. Możliwe jest zatem wyznaczenie wszystkich punktów zatrzymania dla kursów które musiały przebiegać przez dany przystanek. W tym podejściu nie będzie możliwości brania pod uwagę zatrzymań na przystanku po drugiej stronie ulicy. Jest ono również mniej kosztowne, ponieważ zawężona zostaje liczba potrzebnych odległości do policzenia. Rozważana również była możliwość jeszcze większego zawężenia niezbędnych odległości do policzenia, ponieważ możliwe jest wyznaczenie z rozkładu w okolicy którego przystanku powinien znajdować się autobus w danym czasie. Jednak ze względu na potencjalne odchylenia czasowe (spowodowane opóźnieniami, awariami, błędami w rozkładzie jazdy) opcja ta została odrzucona.

Używany algorytm do wyznaczania faktycznych lokalizacji przystanków:

1. Pobranie z bazy wszystkich lokalizacji przystanków.
2. Dla każdego przystanku:
 - a. Pobranie wszystkich możliwych ID wariantów(tras) przejazdów dla danego przystanku.
 - b. Pobranie wszystkich odbytych kursów dla wyznaczonych wariantów.
 - c. Dla każdego kursu wyznaczenie średniej prędkości w poszczególnych punktach oraz odrzucenie punktów gdzie średnia prędkość wynosiła mniej niż *speed_threshold* (parametr algorytmu). De facto jest to wyznaczenie punktów zatrzymania. Ponieważ operacje pobierania kursów z bazy oraz obliczenia średnich są dość kosztowne wszystkie kursy dla danych wariantów zapisywane są w pamięci, gdyż mogą zostać wykorzystane podczas analizy kolejnych przystanku. Przechowywane dane zawierają ID wariantu oraz listę z kursami należącymi do wariantu. Kurs zawiera lokalizacje punktów zatrzymania oraz prędkość.
 - d. Utworzenie listy mającej zawierać obszary zatrzymania. Element listy zawiera pierwszy zapisany w niej punkt zatrzymania, ilość zatrzymań w obszarze wokół tego punktu oraz listę z lokalizacjami wszystkich punktów zatrzymania wokół tego punktu.
 - e. Jeżeli punkt zatrzymania jest w odległości większej niż *search_distance* (parametr algorytmu) od lokalizacji przystanku z bazy odrzucić ten punkt (przejdź do analizy kolejnego punktu)
 - f. Jeżeli punkt zatrzymania znajduje się w odległości mniejszej niż *group_distance* (parametr algorytmu) od któregoś z punktów zapisanych w liście z punktu **d.** to dla elementu listy zawierającej ten punkt zwiększyć ilość zatrzymań w obszarze oraz dodać ten punktu do listy z lokalizacjami wszystkich punktów zatrzymania wokół tego punktu. W przeciwnym wypadku (jeżeli punkt znajduje się w odległości większej niż równej niż 20m od wszystkich punktów zapisanych w liście z punktu **d.**) należy stworzyć nowy element i dodać go do listy z punktu **d.**
 - g. Wprowadzony zostaje wskaźnik, na podstawie którego wyznaczamy obszar który najprawdopodobniej określa położenie przystanku:

$$w = \frac{N}{dist^2}$$

Gdzie:

- N – ilość zatrzymań w danym obszarze
 - $dist$ – odległość od lokalizacji przystanku z bazy (został podniesiony do kwadratu, aby zmniejszyć jego wpływ na wartość wskaźnika w , ponieważ istotniejsza jest liczba zatrzymań)
- h. Znalezienie obszaru, dla którego wskaźnik przyjmuje największą wartość
 - i. Wyznaczenie średniego punktu na podstawie punktów zatrzymania w tym obszarze.
Punkt ten jest wyznaczoną lokalizacją przystanku.

Celem zwiększenia szybkości pobierania danych do bazy zostały dodane odpowiednie indeksy.

2. Aplikacja kliencka

2.1. Uzyskanie informacji z serwera.

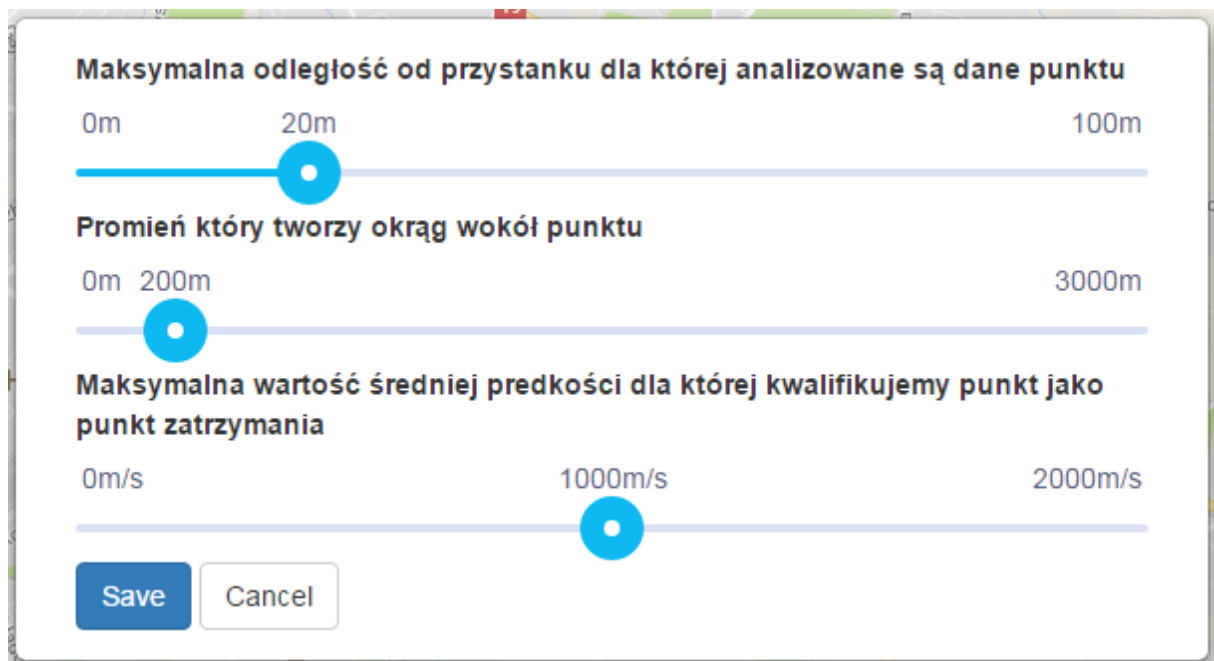
Klient podczas łączenia się z serwerem wywołuje funkcję dostępną pod adresem
<sa>/api/getBadBusStops/<int:grouping_distance>/<int:search_distance>/<int:speed_threshold>

- <sa> - adres serwera, na którym znajduje się aplikacja obliczająca lokalizację przystanków,
- <int:grouping_distance> - odległość grupowania, odległość dla której analizowane są wokół istniejących przystanków punkty,
- <int:search_distance> - promień wyszukiwania, promień tworzący okrąg wokół punktu, jeżeli punkty znajdują się w obrębie wspomnianego okręgu, są razem grupowane,
- <int:speed_threshold> - próg prędkości, maksymalna wartość średniej prędkości dla której punkt kwalifikowany jest jako punkt zatrzymania

Przy starcie aplikacja ładuje domyślnie utworzonego JSON z pliku, dla parametrów

- <int:grouping_distance> - 20
- <int:search_distance> - 200,
- <int:speed_threshold> - 100

Możliwe jest przeładowanie wartości przy wyborze istotnych parametrów wywołania za pomocą przycisku „Parametry obliczeń”



The image shows a web-based configuration interface for bus stop calculations. It features three horizontal sliders, each with a blue circular handle. The first slider is titled 'Maksymalna odległość od przystanku dla której analizowane są dane punktu' and has tick marks at 0m, 20m, and 100m. The second slider is titled 'Promień który tworzy okrąg wokół punktu' and has tick marks at 0m, 200m, and 3000m. The third slider is titled 'Maksymalna wartość średniej prędkości dla której kwalifikujemy punkt jako punkt zatrzymania' and has tick marks at 0m/s, 1000m/s, and 2000m/s. Below the sliders are two buttons: a blue 'Save' button and a white 'Cancel' button with a grey border.

Rysunek 1 Wybór parametrów obliczeń położenia przystanków

Na podstawie wywołania wspomnianej funkcji zwracany jest JSON w następującym formacie:

```
[...
{
  "id": 2,
  "name": "Popiełuszki/Hetmańska",
  "original_latitude": 53.124832,
  "original_longitude": 23.114172,
  "calculated_latitude": 53.1248,
  "calculated_longitude": 23.114215,
  "how_many": 144,
  "distance": 0.0074445056076164985,
  "lines": [
    "12",
    "20",
    "21",
    "8"
  ]
},
{
  "id": 3,
  "name": "Popiełuszki/Upalna",
  "original_latitude": 53.125888,
  "original_longitude": 23.105895,
  "calculated_latitude": 53.12588,
  "calculated_longitude": 23.10591,
  "how_many": 139,
  "distance": 0.0013401079585095871,
  "lines": [
    "12",
    "20",
    "21",
    "8"
  ]
}, ...
]
```

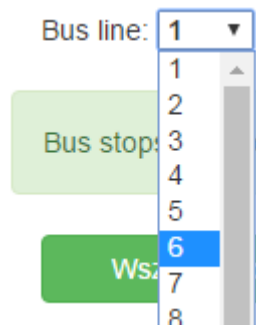
Gdzie:

- Id – id przystanku,
- Name – nazwa przystanku,
- Original_latitude – szerokość geograficzna przystanku zapisanego w bazie,
- Original_longitude – długość geograficzna przystanku zapisanego w bazie,
- Calculated_latitude – szerokość geograficzna obliczona,
- Calculated_longitude – długość geograficzna obliczona,
- How_many – ilość punktów, na podstawie których został wyznaczony przystanek,

- Distance – odległość od lokalizacji przystanku w bazie,
- Lines – tablica linii autobusowych, dla numerów, które zatrzymują się na danym przystanku.

2.2. Naniesienie przystanków autobusowych na mapę.

Na podstawie przekazanych parametrów, użytkownik wybiera linię autobusową z dostępnych:



Rysunek 2 Wybór linii autobusowej

Dla wybranej linii autobusowej następuje wybór interesujących przystanków

List of available bus stops

Gen.E.Fieldorfa "Nila"
Sienkiewicza / Białówny
KZK Zajezdnia
Branickiego/Świętojańska
Branickiego/Miłosza
Branickiego/Nowowarszawska
Baranowicka/Ciołkowskiego
Baranowicka/Sowlańska
Baranowicka/Chłodnia
Baranowicka / Opolska
Branickiego / Baranowicka
Geodetów

List of taken bus stops

KPK Zajezdnia
Malmeda/Zamenhofa
Malmeda/Lipowa
Branickiego/Teatr
Geodetów

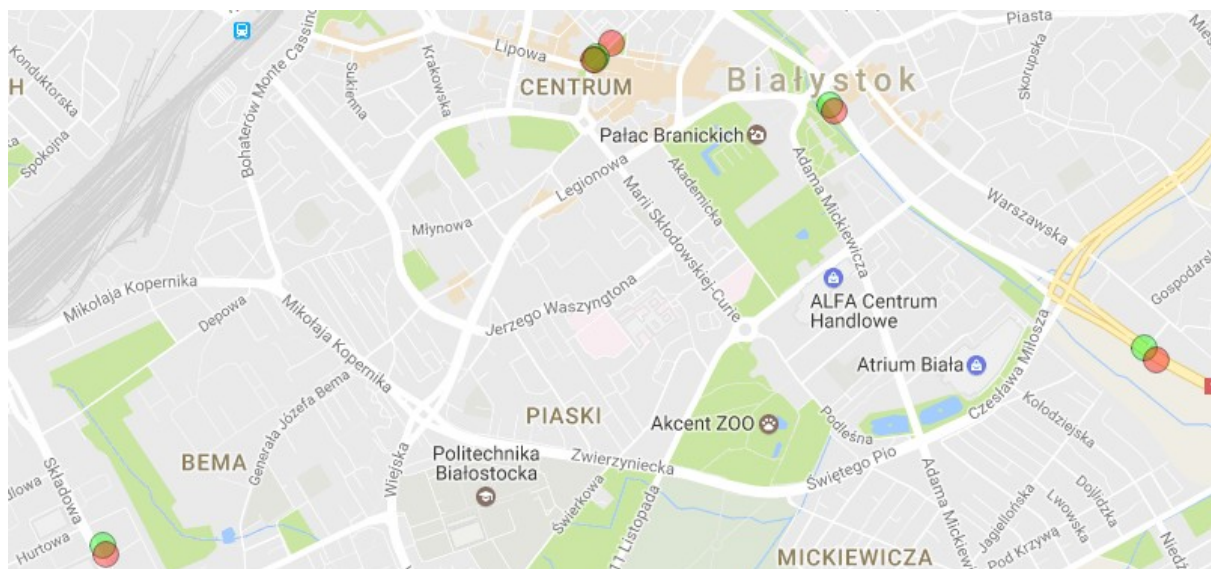
Rysunek 3 Wybór interesujących przystanków

Następnie należy zaznaczyć który rodzaj przystanków autobusowych ma zostać naniesiony na mapę



Rysunek 4 Wybór jaki typ przystanków ma zostać naniesiony na mapę

Dla wybranych przystanków, po wyborze przycisku „Dodaj przystanki” zostaną one naniesione na mapę



Rysunek 5 Przykładowe przystanki naniesione na mapę

W celu umożliwienia rozrysowania trasy autobusu na mapie, na podstawie przystanków, został wprowadzony przycisk „Wszystkie przystanki”. Dla wybranej linii, następuje przeniesienie wszystkich przystanków do sekcji wybranych. Po naniesieniu ich na mapę ukazuje się obraz przebiegu linii autobusowej.

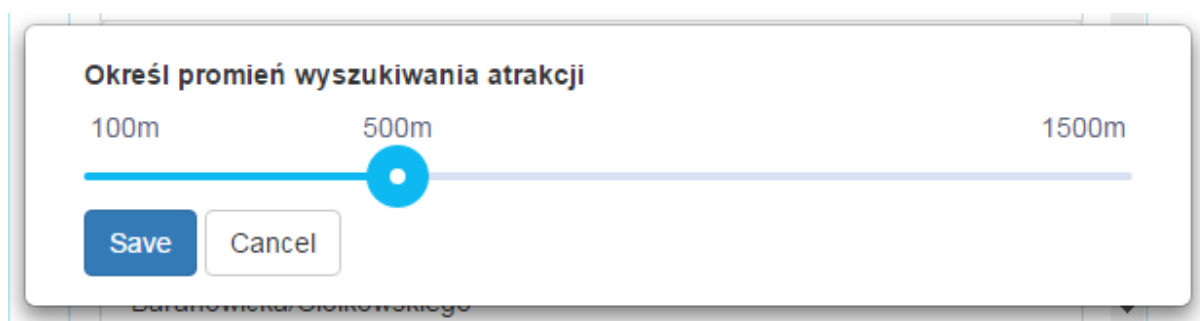


Rysunek 6 Linia przebiegu dla autobusu linii numer 1

Anomalie w przebiegu linii autobusowej, oznaczają najczęściej zajezdnię bądź też przystanki na trasie do zajezdni.

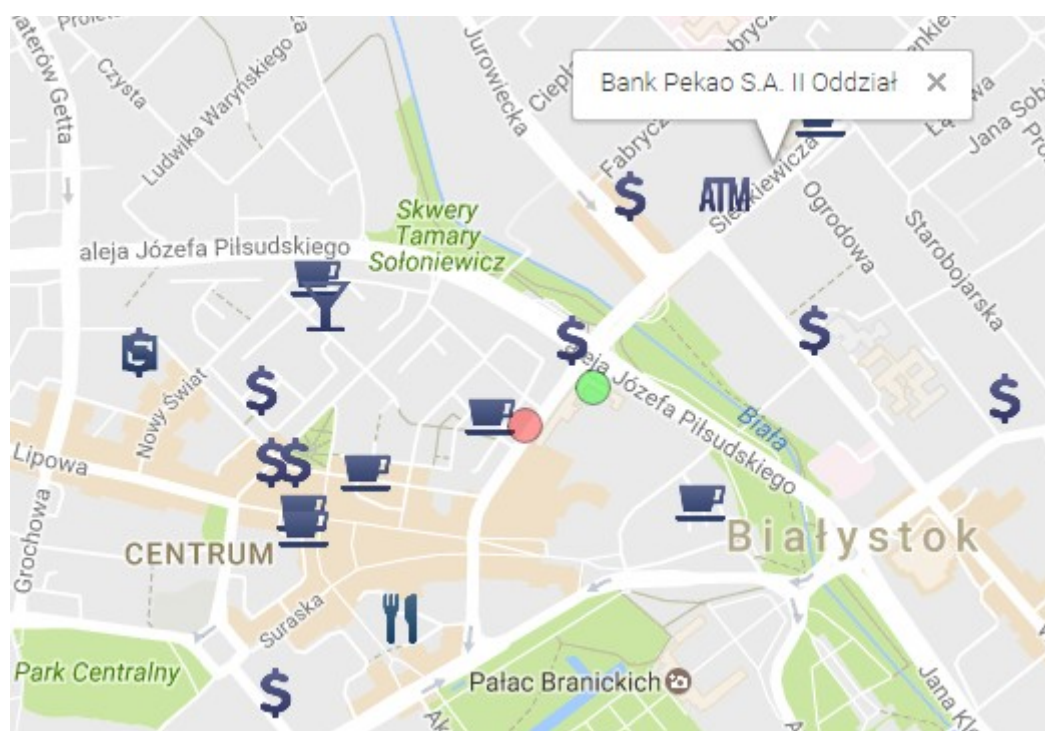
2.3. Naniesienie atrakcji na mapę

Po naniesieniu interesujących przystanków autobusowych na mapę możliwe jest wyszukanie atrakcji jakie są dostępne w określonej odległości od przystanku. Należy określić promień wyszukiwania dla każdego z przystanków



Rysunek 7 Ustalenie promienia wyszukiwania atrakcji

Po ustaleniu promienia wyszukiwania należy zdefiniować określony typ atrakcji, np. piekarnia, bankomat, sklep.



Rysunek 8 Wynik wyszukiwania banków i barów

2.4. Wyszukanie najbliższej atrakcji dla przystanków

Po dokonaniu każdego z poprzednich kroków, możliwe jest również określenie najbliższej atrakcji danego typu dla każdego z naniesionych przystanków. Należy w tym celu przejść poprzednio określoną ścieżkę a następnie wybrać przycisk „Najbliższa atrakcja”.

W celu znalezienia najbliższej atrakcji wykorzystano Algorytm Haversine’a. Określa on najkrótszą drogę pomiędzy dwoma punktami na powierzchni kuli biegnąca po jej powierzchni z uwzględnieniem ich długości i szerokości geograficznych. W formule Haversine’a wyszczególniony jest przypadek bardziej ogólnej formuły w trygonometrii sferycznej, która dotyczy boków i kątów trójkątów sferycznych.

$$a = \sin^2(\Delta\phi/2) + \cos \phi_1 \cdot \cos \phi_2 \cdot \sin^2(\Delta\lambda/2)$$

$$c = 2 \cdot \operatorname{atan2}(\sqrt{a}, \sqrt{1-a})$$

$$d = R \cdot c$$

Gdzie:

- Φ jest szerokość geograficzna,
- λ jest długością geograficzną,
- R jest promieniem Ziemi (średni promień = 6.371 km);

Dla ciekawości, c jest odległością kątową w radianach, natomiast a jest kwadratem połowy długości cięciwy pomiędzy punktami.

Formuła Haversine'a pozostaje szczególnie dobrze przystosowana do obliczeń liczbowych nawet w niewielkich odległościach - w przeciwieństwie do obliczeń opartych na kulistym prawie cosinusów.

Powyższa formuła jest tylko aproksymacją stosowaną na Ziemi, która nie jest idealną kulą. Promień Ziemi R jest zróżnicowany na biegunach względem długości na równiku. Co ważniejsze, promień krzywizny linii północ-południe na powierzchni Ziemi jest o 1% większy w biegunach ($\approx 6399,594$ km) niż w równiku (≈ 6335.439 km). Wynika stąd, że poprawność wyników może zawierać błąd pomiarowy wielkości 0.5%. Dla wielkości geograficznych, których odległość jest dość duża, jest to zakres niepewności pomiarowej, jednak na potrzeby projektu oraz małe zróżnicowanie położenia geograficznego jest to akceptowalne.

3. Architektura i technologie

Aplikacja została zrealizowana w postaci aplikacji Webowej. Architektura wyróżnia dwie warstwy: klient oraz serwer. Warstwa klienta wykorzystywała technologie HTML5, angularJS, Bootstrap oraz Google Maps Api.

AngularJS to framework JavaScript stworzony przez inżynierów z Google. Służy on do szybkiego i łatwego budowania aplikacji internetowych, tak zwanych – single app. Model oparty o MVW (Model – View – Whatever) pozwala pogodzić idee JavaScript i modelu MVC. AngularJS daje możliwość modułowej budowy aplikacji, która może być testowana w łatwy sposób.

Do zaprojektowania interfejsu graficznego wykorzystano narzędzie Bootstrap. Jest to framework CSS, rozwijany przez programistów Twittera. Zawiera zestaw przydatnych narzędzi ułatwiających tworzenie interfejsu graficznego stron oraz aplikacji internetowych. Bazuje głównie na gotowych rozwiązaniach

HTML oraz CSS i może być stosowany m.in. do stylizacji takich elementów jak teksty, formularze, przyciski, wykresy, nawigacje i innych komponentów wyświetlanych na stronie. Bootstrap korzysta także z języka JavaScript.

Google Maps to jeden z serwisów wyszukiwarki internetowej Google, który umożliwia wyświetlanie szczegółowych zdjęć powierzchni Ziemi oraz map kartograficznych dróg i miast. Dzięki Google Maps możemy także wyznaczać trasy, obliczać odległości pomiędzy punktem początkowym i końcowym oraz czas przebycia wyznaczonej trasy, wybranym środkiem transportu, poprzez określone punkty pośrednie. Dodatkowo pozwala uzyskać informacje dotyczące komunikacji miejskiej, spisu firm i usług, atrakcji turystycznych, miejscowych restauracji i miejsc noclegowych włącznie z opiniami klientów. Aby umożliwić korzystanie z usług Google Maps został stworzony i udostępniony przez firmę Google Interfejs Programowania Aplikacji (ang. Application Programming Interface) w skrócie API, który umożliwia wstawienie własnej mapy na dowolną stronę internetową oraz zarządzanie mapą za pomocą API przy użyciu języka Google Maps JavaScript API V3. Google Maps API zawiera szereg interfejsów API, które pozwalają na osadzanie i zarządzanie Google Maps na stronie internetowej. Na potrzeby projektu wykorzystaliśmy interfejs Maps JavaScript API oraz Web Services, w którym zawarte były usługi Places API - umożliwia wyszukiwanie w określonym regionie miejsc spełniających przekazane w parametrach kryteria. Można m.in. wyszukiwać: hotele, restauracje, stacje benzynowe, banki, urzędy pocztowe, kina, sklepy, parki, kościoły, muzea itp.

Część serwerowa została zrealizowana przy wykorzystaniu języka Python w wersji 3.6 wraz z wykorzystaniem biblioteki Flask zajmującej się zarządzaniem zapytaniami, wywoływaniem odpowiednich funkcji oraz wysyłaniem odpowiedzi. W celu komunikacji z bazą danych został wykorzystany framework SQLAlchemy wraz ze sterownikiem Psycopg2. Architektura zakładała położenie nacisku na optymalizację czasu przetwarzania zapytań bazodanowych. W tym celu dane były przetrzymywane w pamięci RAM. Oznaczało to, że w przypadku gdy użytkownik zdecyduje się wykorzystać te same parametry, nie było wysyłane zapytanie do bazy danych, gdyż informacje zostały wcześniej zapisane w cache.