SPDB - Dokumentacja końcowa

Wojciech Przechodzeń

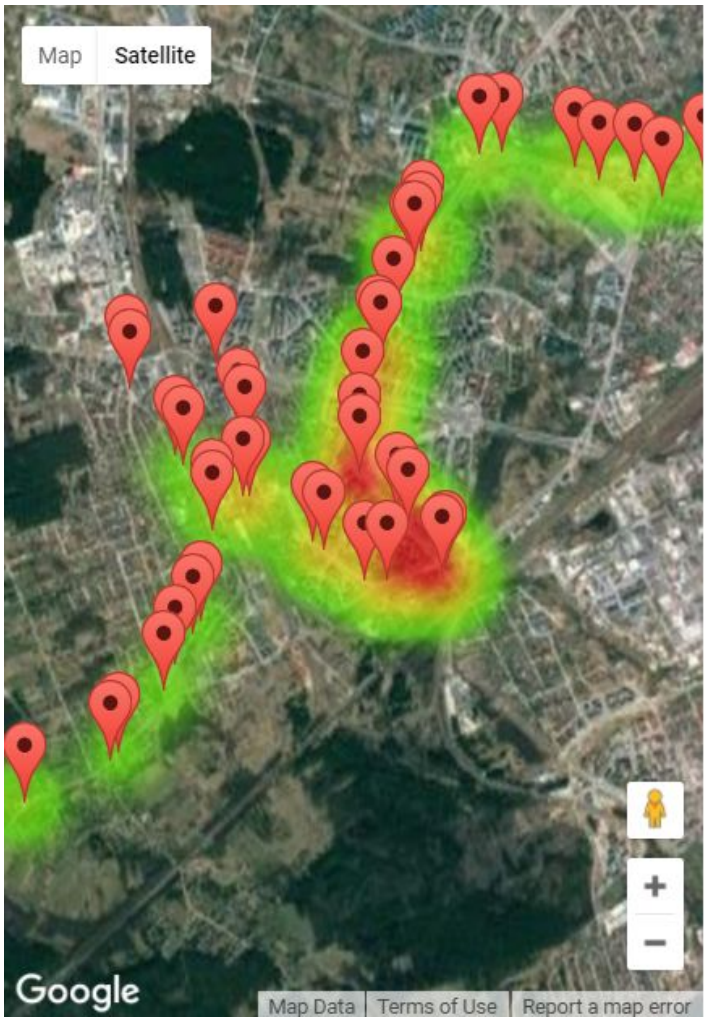
Radosław Nowak

1. Pseudokod algorytmu wyznaczania przyspieszeń / opóźnień.

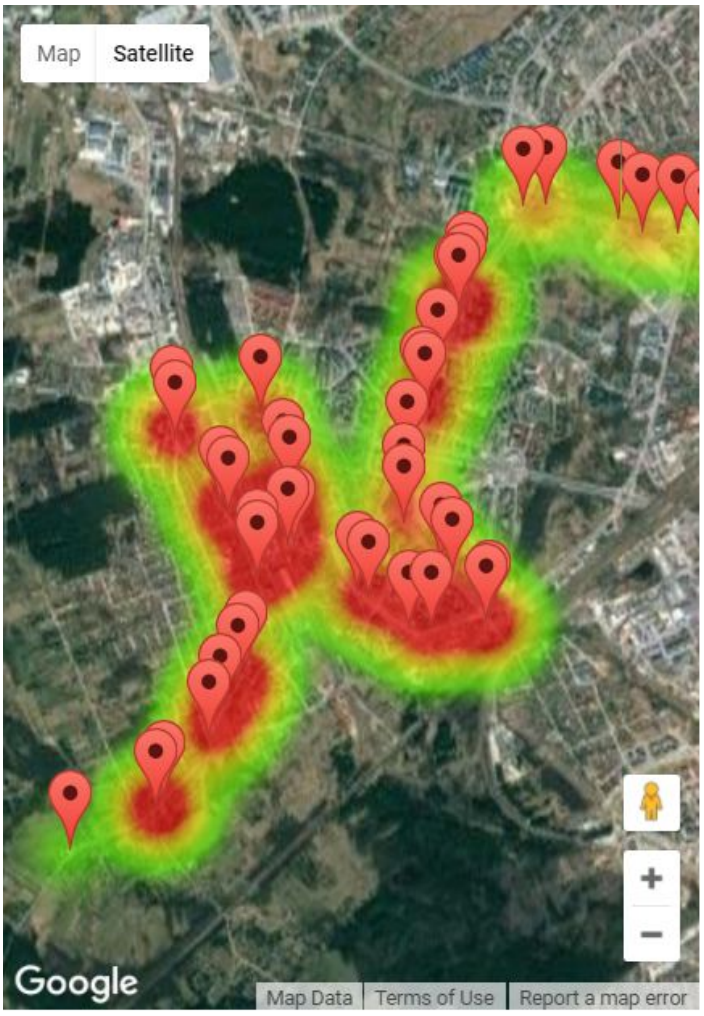
```
// lineName - numer linii, ktora nas interesuje. Np "127"
// startHour - początkowa godzina przedziału, który nas interesuje
// endHour - końcowa godzina przedziału, który nas interesuje
public AnalyzedData countDelays(lineName, startHour, endHour) {
        // definicja zmiennych.
        //  Dane pobrane z bazy danych
        List<DataFromDB> dataFromDB := dataBaseService.getData(lineName, startHour,
endHour);
        // Mapa, w której znajduja sie informacje o tym, z jakim opoznieniem ruszyl autobus
z danego przystanku
        Map<String, Long> delays := new HashMap<>();
        // Mapa wspolrzednych dla kazdego przystanku bioracego udzial w analizie
        Map<String, Cords> stops := new HashMap<>();
        // Głowna petla algorytmu, obliczajaca dla kazdego przystanku sume opoznien danej
linii.
        foreach (DataFromDB data : dataFromDB) {
            depDate := data.getScheduledDeparture().getTime();
            realDate := data.getRealDeparture().getTime();
            diff := realDate - depDate;
            stopId := data.getStopId();
            // Aktualizowanie opoznienia dla kazdego przystanku
            if (delays.contains(stopId) {
                currentDiff := delays.get(stopId);
                delays.put(stopId, currentDiff + diff);
            } else {
                delays.put(stopId, diff);
            }
            // aktualizowanie wspolrzednych dla analizowanego przystanku (jesli nie
istniaja w mapie)
            stops.putIfAbsent(stopId, new Cords(data.getX(), data.getY()));
        }
        // Łączenie danych z dwóch wcześniejszych map.
        return joinData(delays, stops);
    }
```

2. Wyniki testów

Testy wykazały to, czego się spodziewaliśmy, to znaczy, że największe opóźnienia dotyczą przedziałów godzinowych popołudniowych, kiedy ludzie wracają pracy oraz przystanków mieszczących się w centrum oraz na końcu trasy przejazdu. Dla przykładu zamieszczona zostanie poniżej analiza przyspieszeń / opóźnień dla linii ‘107’ w godzinach porannych (6:00 - 7:00) oraz wieczornych, typowych dla powrotów z pracy do domu (17:00 - 18:00).



Rys 1. Opóźnienia dla godzin porannych



Rys 2. Opóźnienia dla godzin popołudniowych

3. Architektura aplikacji

Aplikacja zaimplementowana została jako aplikacja webowa z podziałem na back-end oraz front-end.

a. **Back-end** - zaimplementowany został z wykorzystaniem framework'a Spring Boot w języku Java. Serwer wystawia api REST-owe pozwalające klientowi front-endowemu na pobranie informacji o opóźnieniach / przyspieszeniach danej linii komunikacyjnej w zadanym przedziale czasowym.

Aplikacja serwera składa się z modułów: *Kontrolera*, Serwisów oraz *wzorców danych*. *Kontroler* odpowiedzialny jest za komunikację z aplikacją klienta - przyjmuje zapytanie HTTP i odpowiednio je interpretuje. *Serwisy* odpowiedzialne są za dokonywanie obliczeń na podstawie danych przekazanych im przez kontroler oraz na podstawie danych pobranych z bazy

danych. *Wzorce danych* są pomocniczymi klasami, które odpowiadają logicznie danym, które są obliczane i interpretowane.

b. **Front-end** - aplikacja klienta zaimplementowana została wykorzystaniem framework'a dedykowanego językowi JavaScript - AngularJS. Do wizualizacji opóźnień użyto technologię Google Maps. Aplikacja klienta podzielona jest na warstwy zgodnie ze wzorcem MVC(Model-View-Controller).

4. Zastosowany model danych

Model danych, które cache'owane są w pamięci programu serwera wygląda następująco:

```
public class OutputData {
    private String stopId;
    private String stopName;
    private String x,y;
    // seconds
    private long delaySum;
    private long stopCount;
}
```

Model ten przechowuje wszystkie potrzebne klientowi informacje i dzięki zastosowaniu techniki cache'owania danych dostęp do tych danych jest zdecydowanie szybszy i nie obciąża bazy danych. Zadanie wykonywujące cache wykonuje się raz na dobę, ponieważ zakładamy, że w rzeczywistym systemie dane dotyczące odjazdów / przyjazdów pojazdów komunikacji miejskiej mogą napływać non stop.