|  |  |  |
| --- | --- | --- |
| wordml://75.png |  | wordml://76.png |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | | |  |  | | --- | --- | | Imię i nazwisko studenta: Artur Kąkol |  | | | |  | | --- | | Nr albumu: 143251 | | | |  | | --- | | Studia drugiego stopnia | | | |  | | --- | | Forma studiów: stacjonarne | | | |  | | --- | | Kierunek studiów: Informatyka | | | |  | | --- | | Specjalność/profil: Algorytmy i technologie internetowe | | |  |
|  |  |
| |  | | --- | | **PRACA DYPLOMOWA MAGISTERSKA** | | |
| |  | | --- | | Tytuł pracy w języku polskim: System internetowy wspierający marszrutyzację pojazdów | | Tytuł pracy w języku angielskim: Web-based system supporting Vehicle Routing Problems | |  | | |
| |  |  |  |  | | --- | --- | --- | --- | |  | |  | | --- | | Potwierdzenie przyjęcia pracy | | | |  | |  | | --- | | Opiekun pracy |   *podpis* | |  | | --- | | Kierownik Katedry/Zakładu |   *podpis* | |  | dr inż. Krzysztof Bruniecki |  | | |
| |  | | --- | | Data oddania pracy do dziekanatu: | | |

|  |  |  |  |
| --- | --- | --- | --- |
| wordml://75.png | |  | wordml://76.png |
|  | **OŚWIADCZENIE** | | |
|  | |  | | --- | | Imię i nazwisko: Artur Kąkol  Data i miejsce urodzenia: 27.05.1992, Kościerzyna  Nr albumu: 143251  Wydział: Wydział Elektroniki, Telekomunikacji i Informatyki  Kierunek: informatyka  Poziom studiów: II stopnia  Forma studiów: stacjonarne | | | |
|  | |  | | --- | | Ja, niżej podpisany(a), wyrażam zgodę/nie wyrażam zgody\* na korzystanie z mojej pracy dyplomowej zatytułowanej: Licznik uśmiechów  do celów naukowych lub dydaktycznych.1 | | | |
| |  |  | | --- | --- | | Gdańsk, dnia .................................. | .....................................................  *podpis studenta* | | | | |
|  | |  | | --- | | Świadomy(a) odpowiedzialności karnej z tytułu naruszenia przepisów ustawy z dnia 4 lutego 1994 r. o prawie autorskim i prawach pokrewnych (Dz. U. z 2006 r., nr 90, poz. 631) i konsekwencji dyscyplinarnych określonych w ustawie Prawo o szkolnictwie wyższym (Dz. U. z 2012 r., poz. 572 z późn. zm.),2 a także odpowiedzialności cywilno-prawnej oświadczam, że przedkładana praca dyplomowa została opracowana przeze mnie samodzielnie.  Niniejsza(y) praca dyplomowa nie była wcześniej podstawą żadnej innej urzędowej procedury związanej z nadaniem tytułu zawodowego.  Wszystkie informacje umieszczone w ww. pracy dyplomowej, uzyskane ze źródeł pisanych i elektronicznych, zostały udokumentowane w wykazie literatury odpowiednimi odnośnikami zgodnie z art. 34 ustawy o prawie autorskim i prawach pokrewnych.  Potwierdzam zgodność niniejszej wersji pracy dyplomowej z załączoną wersją elektroniczną. | | | |
| |  |  | | --- | --- | | Gdańsk, dnia .................................. | .....................................................  *podpis studenta* | | | | |
|  | Upoważniam Politechnikę Gdańską do umieszczenia ww. pracy dyplomowej w wersji elektronicznej w otwartym, cyfrowym repozytorium instytucjonalnym Politechniki Gdańskiej oraz poddawania jej procesom weryfikacji i ochrony przed przywłaszczaniem jej autorstwa. | | |
| |  |  | | --- | --- | | Gdańsk, dnia ................................. | .....................................................  *podpis studenta* | | | | |
|  | |  | | --- | | \*) niepotrzebne skreślić | | | |
| |  | | --- | |  | | | | |
| |  |  |  |  | | --- | --- | --- | --- | | |  | | --- | | 1 | | |  | | --- | | Zarządzenie Rektora Politechniki Gdańskiej nr 34/2009 z 9 listopada 2009 r., załącznik nr 8 do instrukcji archiwalnej PG. | | | |  | | --- | | 2 | | |  | | --- | | Ustawa z dnia 27 lipca 2005 r. Prawo o szkolnictwie wyższym: | | |  | |  | | --- | | Art. 214 ustęp 4. W razie podejrzenia popełnienia przez studenta czynu podlegającego na przypisaniu sobie autorstwa istotnego fragmentu lub innych elementów cudzego utworu rektor niezwłocznie poleca przeprowadzenie postępowania wyjaśniającego. | | |  | |  | | --- | | Art. 214 ustęp 6. Jeżeli w wyniku postępowania wyjaśniającego zebrany materiał potwierdza popełnienie czynu, o którym mowa w ust. 4, rektor wstrzymuje postępowanie o nadanie tytułu zawodowego do czasu wydania orzeczenia przez komisję dyscyplinarną oraz składa zawiadomienie o popełnieniu przestępstwa. | | | | | |

Streszczenie

Celem pracy jest utworzenie systemu pozwalającego na wspieranie marszrutyzacji pojazdów.

W ramach projektu powstała aplikacja pozwalająca na wyznaczanie tras dla pojazdów. Aplikację można podzielić na dwie części. Pierwszą z nich jest aplikacja kliencka oparta na frameworku Angular, pozwalająca na zarządzanie magazynami oraz odbiorcami. Pozwala także na wyświetlanie tras na mapie. Drugą częścią są usługi RESTowe pozwalające na obliczanie rozwiązania na podstawie zadanych parametrów.

**Słowa kluczowe:** marszrutyzacja pojazdów

**Dziedzina nauki i techniki, zgodnie z wymogami OECD:** Nauki inżynieryjne i techniczne. Elektrotechnika, elektronika, inżynieria informatyczna.

Abstract

**Keywords:**

**OECD field of science and technology (FOS) classification:**.

Spis treści

[Streszczenie 3](#_Toc510552645)

[Abstract 4](#_Toc510552646)

[Spis treści 5](#_Toc510552647)

[1. Wstęp 6](#_Toc510552648)

[1.1. Cele pracy 7](#_Toc510552649)

[1.2. Tematyka pracy 7](#_Toc510552650)

[1.3. Planowanie tras pojazdów 7](#_Toc510552651)

[2. Opis systemu 9](#_Toc510552652)

[2.1. Architektura 9](#_Toc510552653)

[2.2. Część kliencka 9](#_Toc510552654)

[2.3. Część serwerowa 15](#_Toc510552655)

[2.4. Komunikacja 17](#_Toc510552656)

[2.5. Dostęp z urządzeń mobilnych 22](#_Toc510552657)

[3. Walidacja rozwiązania 23](#_Toc510552658)

[4. Podsumowanie 24](#_Toc510552659)

[Wykaz literatury 25](#_Toc510552660)

[Wykaz tabel 26](#_Toc510552661)

[Wykaz rysunków 27](#_Toc510552662)

# Wstęp

We współczesnym świecie transport odgrywa bardzo ważną rolę we współczesnej gospodarce. Zarówno w usługach(transport ludzi) jak i w przemyśle(dostarczanie towarów). Dynamiczny rozwój rynku wymusza na przedsiębiorstwach utrzymania konkurencyjności. Można ją uzyskać na kilka sposobów, między innymi poprzez niższą cenę. Aktualnie przyjmuje się, że koszty paliwa w przedsiębiorstwie zajmującym się transportem drogowym mieszczą się w przedziale 15%-20%[[1]](#endnote-1). Obniżając długości tras można w dużym stopniu przyczynić się do końcowej ceny towaru.

Kolejnymi ważnym aspektem jest też, jakość realizowanych usług. Dlatego warto zadbać o to, żeby czas realizacji był jak najkrótszy oraz sprawność transportu była jak najwyższa. Pomóc w tym może oraz powinien dedykowany system internetowy, dzięki któremu w łatwy oraz szybki sposób można by było wyznaczyć trasy dla poszczególnych pojazdów.

## Cele pracy

Celem pracy jest utworzenie systemu internetowego wspomagającego marszrutyzację pojazdów. Pozwalał on będzie na łatwe zarządzanie punktami docelowymi oraz położeniem magazynu. Każdy z punktów docelowych posiada wymagania, którym należy sprostać. System ten też wspierał będzie urządzenia mobilne.

Dodatkowo ocenione oraz porównane zostaną poszczególne sposoby obliczania tras dla poszczególnych pojazdów we flocie. Porównane zostaną algorytmy oraz zostanie pokazany wpływ, jaki wpływ na ostateczne rozwiązanie ma sposób obliczania odległości pomiędzy poszczególnymi odbiorcami lub magazynem.

## Tematyka pracy

Niniejszy dokument skupia się na zakresie marszrutyzacji pojazdów oraz planowaniu tras dla dostaw towarów. Zagadnienia w tym zakresie mogą być bardzo skomplikowane oraz złożone. Ze względu na rodzaj różnych zastosowań oraz uwarunkowań do rozważenia można wyróżnić wiele wariantów problemu. Najważniejszymi jednak są:

* Zagadnienie komiwojażera (ang. *Travelling Salesman Problem*)
* Klasyczny problem marszrutyzacji (ang. *Vehicle Routing Problem*)

Zagadnienie komiwojażera to najprostszy wariant planowania dostaw. Uwzględnia on tylko jeden środek transportu, bez dodatkowych ograniczeń. Polega na wyznaczenie najkrótszej trasy, która odwiedza wszystkich obiorców oraz wrócić z powrotem do magazynu.

Klasyczny problem marszrutyzacji odpowiada na pytanie: „Jaki jest optymalny zestaw tras dla floty pojazdów w celu dostarczenia dóbr dla określonych odbiorców?”. Po raz pierwszy został zaprezentowany przez G.B. Dantziga oraz R. H. Ramsera w 1959 roku[[2]](#endnote-2). Jako, że zadany problem, jest rozwinięciem problemu komiwojażera to należy do problemów NP-trudnych[[3]](#endnote-3). Istnieją liczne odmiany klasycznego zagadnienia marszrutyzacji.

Jednym z nich jest Capacitated Vehicle Routing Problem (CVRP), w którym flota ma ograniczoną pojemność, która nie może zostać przekroczona. Dodatkowo odbiorca może być odwiedzony tylko i wyłącznie przez jeden pojazd. Magazyn jest jeden dla wszystkich odbiorców. To głównie tą odmiana zostanie wykorzystana w danej pracy.

Kolejnym jest Vehicle Routing Problem with Time Windows (VRPTW). W tym rodzaju każdy z odbiorów przypisane ma ramy czasowe definiujące, kiedy dany odbiorca musi zostać obsłużony. Dopuszcza się, że pojazd może przybyć przed czasem do odbiorcy, jednak w takim wypadku musi czekać na rozładunek/załadunek.

Innymi przykładami mogą być: Vehicle Routing Problem with Multiple Trips (VRPMT), Vehicle Routing Problem with Pickup and Delivery (VRPPD) lub Open Vehicle Routing Problem (OVRP).

## Planowanie tras pojazdów

Planowanie tras pojazdów może być realizowane przez dwa różne rodzaje metod: dokładne oraz przybliżone. Wybór sposobu opiera się głównie na rozmiarze problemu, tj. na ilości odbiorców, ilości ograniczeń oraz typie floty. W przypadku problemów o mniejszym skomplikowaniu zasadnym jest używanie metod dokładnych, natomiast dla bardziej złożonych problemów ze względu na swoją złożoność obliczeniową są zdyskwalifikowane. Metody przybliżone w znacznym stopniu przyspieszają proces obliczeniowy, jednak ich rozwiązanie niekoniecznie jest optymalne (jednak może takie być).

# Opis systemu

## Architektura

Stworzony system został oparty na architekturze klient-serwer. Pozwoliło to na łatwe rozdzielenie obsługi mapy, niezależnego wyświetlania wyników obliczeń oraz części odpowiedzialnej za obliczenia. Dało to możliwość ewentualnych implementacji innych aplikacji korzystających z części serwerowej – na przykład dedykowanej aplikacji mobilnej. Poniższy diagram przestawia wykorzystane technologie w poszczególnych modułach oraz schemat komunikacji między nimi. Poszczególne części będą opisane w kolejnych podrozdziałach.

## Część kliencka

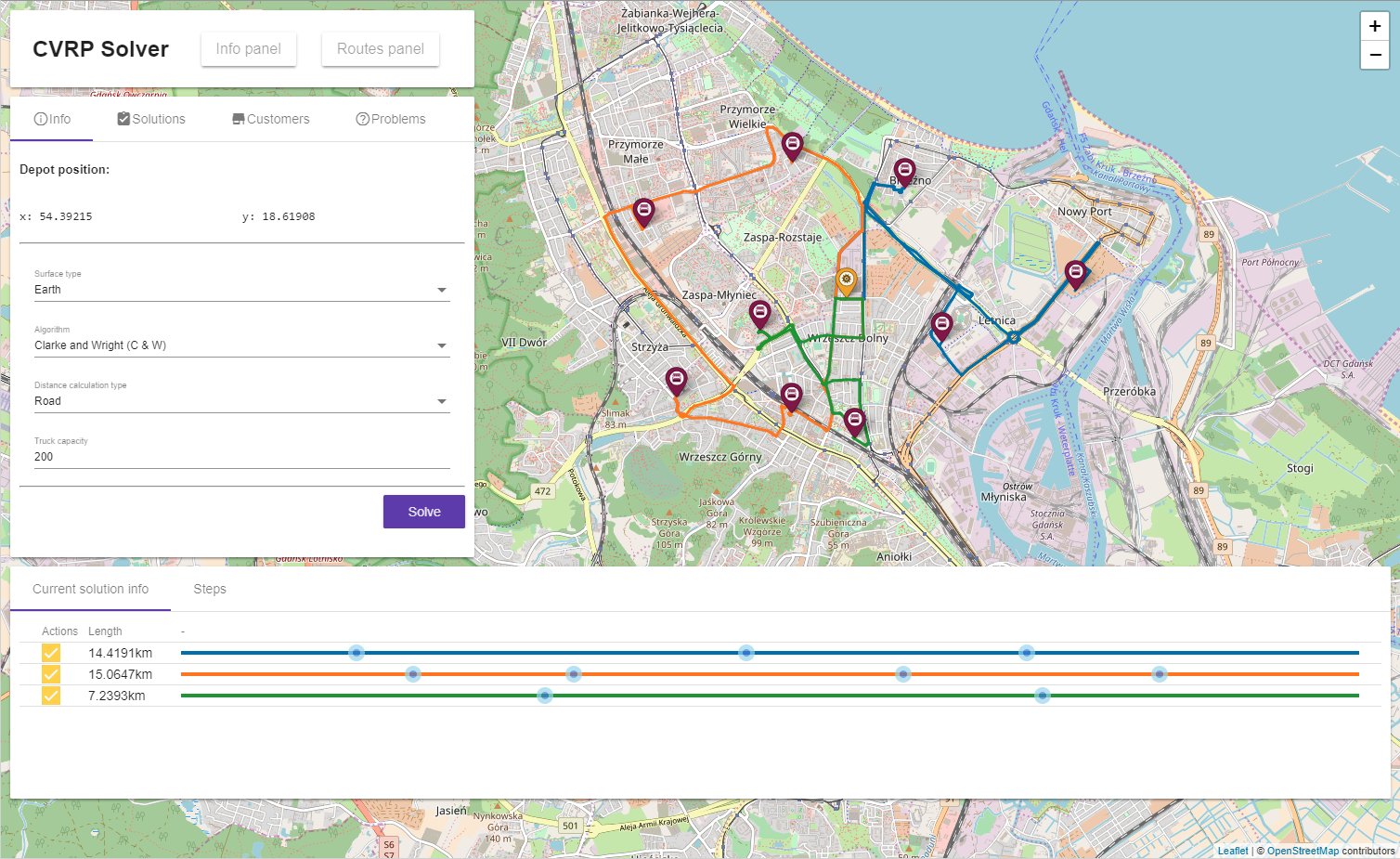
Front-end jest odpowiedzialny za pobieranie danych od użytkownika oraz przekazywaniu ich do back-endu w celu opracowania wyników. Z tego powodu powinien być on jak najprostszy oraz w intuicyjny sposób dawać możliwość na edycję danych. Część ta została zbudowana w podejściu SPA (Single Page Application). Oznacza to, że strona, która została załadowana na początku, po interakcji użytkownika nie odświeża całej strony z serwera tylko dynamicznie zmienia aktualną stronę. Dzięki temu użytkownik ma wrażenie jakby korzystał z natywnej aplikacji.

### Wykorzystane technologie

Aplikacja kliencka została napisana przy użyciu platformy programistycznej Angular[[4]](#endnote-4) w wersji 4.2.4, która jest wspierana oraz rozwijana przez firmę Google. Obsługa mapy wpierana jest dzięki bibliotece leaflet.js. Same mapy wyświetlane są dzięki OpenStreetMap – aktywnie utrzymywanemu projektowi tworzącemu darmową mapę.

### Budowa interfejsu WWW

Aplikacja składa się głównie z jednego widoku, na którym odbywa się cała kontrola aplikacji widoczny na Rys x.

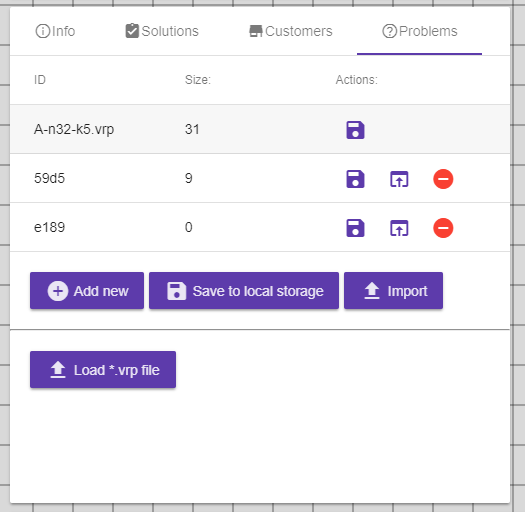


Rys. 1

Można go podzielić na 4 główne komponenty:

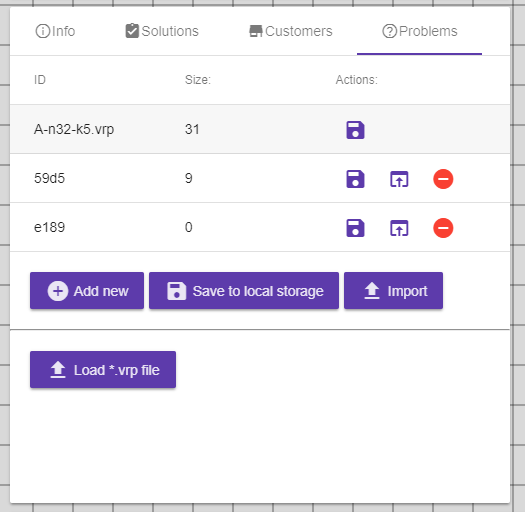
* Górny pasek wyświetlający nazwę aplikacji oraz nazwę zalogowanego użytkownika.
* Komponent, który pozwala na centralne zarządzanie całym systemem.
* Mapę, która wyświetla rozmieszczenie oraz obliczone rozwiązanie.
* Dolne komponent wyświetlający szczegółowe informacje na temat rozwiązania.

### Zakładka „Info”



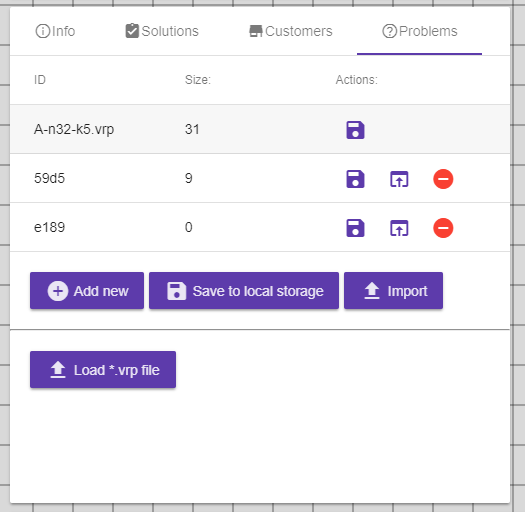
Rys. 2

### Zakładka „Solutions”



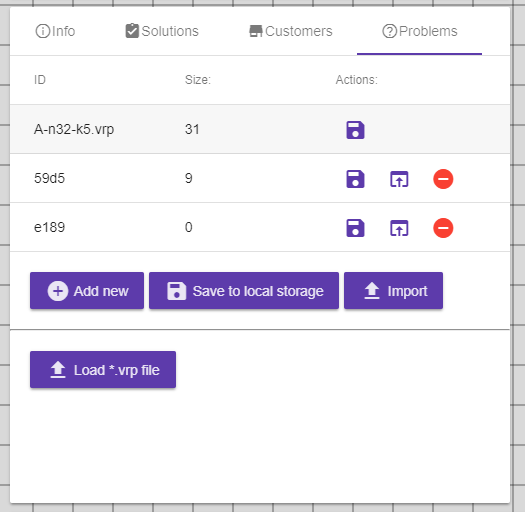
Rys. 3

### Zakładka „Customers”



Rys. 4

### Zakładka „Problems”

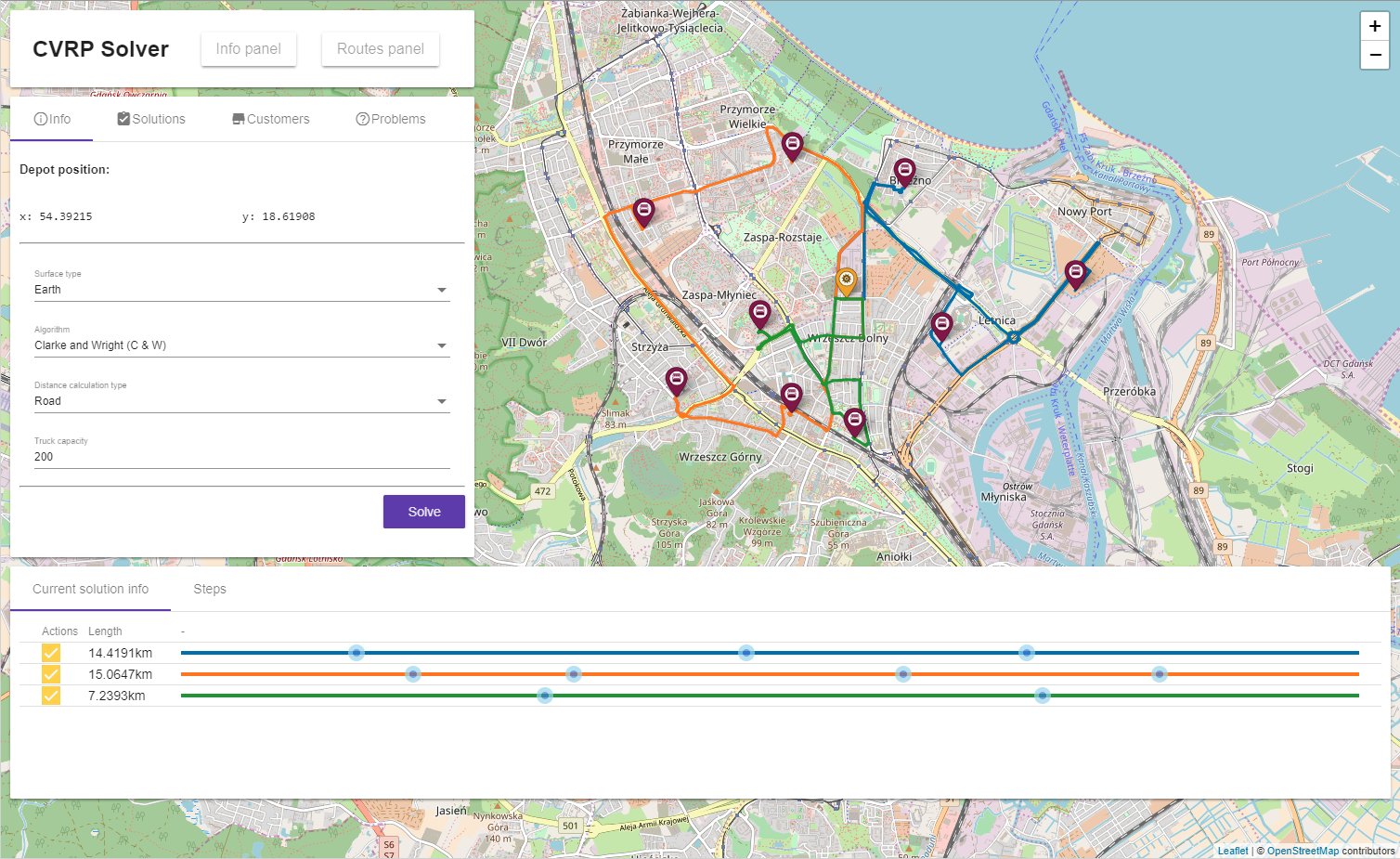


Rys. 5

### Mapa

### Komponent wyświetlający szczegóły rozwiązania

### Ekran logowania, rejestracji oraz zmiany hasła użytkownika

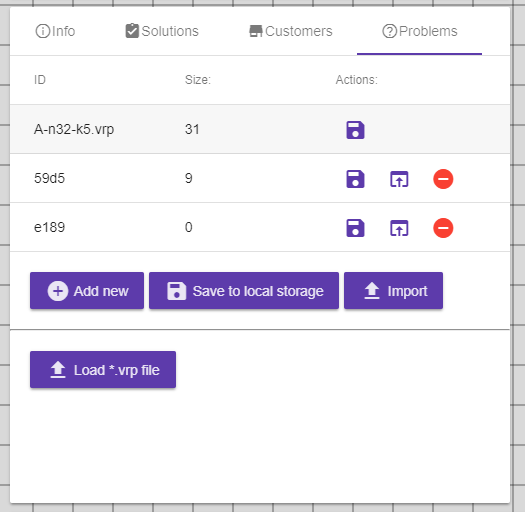


Rys. 6

Jak można zauważyć, większą część aplikacji zajmuje mapa, która aktualnie wyświetla wyliczone trasy. Po lewej stronie znajduje się menu odpowiedzialne na ustawienie parametrów wybranego elementu lub ogólnych ustawień, takich jak wybór algorytmu czy sposób obliczania odległości pomiędzy punktami. Natomiast dolny panel wyświetla aktualnie wyliczone rozwiązanie oraz pośrednie stany działania algorytmu, jeżeli takie istnieją.

### Dodawanie problemu

Dodawanie oraz przełączanie się pomiędzy problemami dostępne jest na zakładce „Problems” dostępnej na panelu dostępnym po lewej stronie ekranu.



Rys. 7

Aplikacja w tym samym momencie może pracować na kilku problemach. Można się dowolnie pomiędzy nimi przełączać. Na powyższym rysunku można zobaczyć 3 problemy wyświetlone w tabeli. Pierwsza kolumna w tej tabeli oznacza unikalne ID problemu. Następna zatytułowana „*Size*” oznacza liczbę odbiorców w danym problemie. Ostatnią kolumną są akcje, które są dostępne na podanym problemie.

Pierwszą z nich jest export. Problem exportowany jest w formacie *json*. W wygenerowanym pliku znajdują się wszystkie ustawienia oraz wcześniej obliczone rozwiązania.

Kolejną akcją jest załadowanie problemu. Opcja ta nie jest dostępna dla pierwszego z nich, ponieważ jest ona aktualnie wczytany.

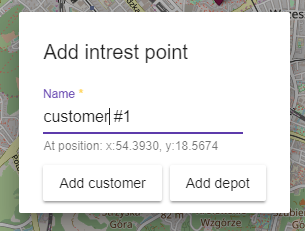
Ostatnią akcją jest usunięcie problemu.

Pod tabelą znajdują się cztery przyciski. Przycisk o nazwie „*Add new*” odpowiedziany jest za dodawanie nowego problemu z domyślnymi ustawieniami. Kolejnym jest „*Save to local storage*”, który zapisuje wszystkie problemy do pamięci lokalnej przeglądarki, dzięki czemu po ponownym otwarciu strony możemy edytować te problemu, nad którymi wcześniej pracowaliśmy. Kolejny przycisk - „*Import*”. Pozwala na wczytanie wcześniej wyeksportowanych problemów w formacie \*.json. Ostatnim przyciskiem dostępnym na tej zakładce jest „*Load \*.vrp file*”, który pozwala na wczytanie problemu z pliku \*.vrp.

Plik ten możemy podzielić na dwie sekcje, część specyfikacji oraz część z danymi. Pierwsza nich składa się z informacji na temat formatu danych natomiast druga zawiera konkretne dane na temat odbiorców oraz magazynu.

### Dodawanie magazynu lub odbiorcy

Aby dodać magazyn lub odbiorcę należy nacisnąć lewym przyciskiem myszy na mapę a następnie z menu wybrać, co chcemy dodać.



Rys. 8 Menu pozwalające na dodawanie magazynu lub odbiorcy

### Obliczanie wyniku

Po poprawnym uzupełnieniu danych oraz wybraniu odpowiednich proces obliczania można rozpocząć poprzez naciśnięcie przycisku „*Solve*”. Jako, że proces ten może trwać długo, pokazywany jest ekran ładowania przysłaniający mapę oraz ustawienia. W tle można jednak zobaczyć etapy pośrednie dla wyliczania końcowego rozwiązania, inne w zależności od użytego algorytmu lub ustawień.

### Wykorzystane technologie

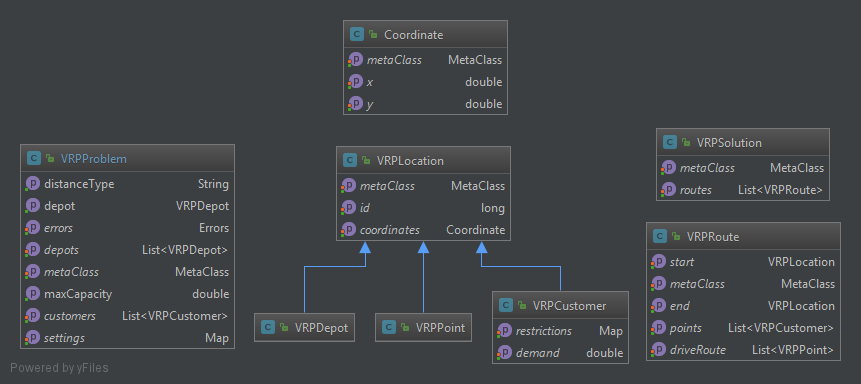
//TODO://

## Część serwerowa

Część serwerowa odpowiedzialna jest za obliczanie rozwiązania dla podanego problemu. Aplikacja została napisana w języku Groovy przy pomocy frameworka Grails w wersji 3.2.4.

### Schemat klas

//TODO:// Dokonczyc



Rys. 9

### Przykładowe zapytanie do serwera

{

"depots":[

{

"coordinates":{

"x":54.374258435906924,

"y":18.617620468139652

},

"id":0

}

],

"customers":[

{

"coordinates":{

"x":54.37955760256642,

"y":18.58242988586426

},

"demand":50,

"id":0

}

],

"settings":{

"algorithm":"savings",

"distance":"air",

"geo\_distance":"spherical",

"capacity":200

}

}

### Proces przetwarzania żądania

Po otrzymaniu przez serwer na podstawie podanych danych tworzy obiekt klasy *VRPProblem*. Przekazywany jest on do serwisu odpowiedzialnego za obsługę problemu. Ten wyszukuje wszystkie zarejestrowane

TODO://

### Obliczanie odległości pomiędzy punktami

Istnieje kilka możliwości obliczania odległości pomiędzy poszczególnymi punktami na mapie. W aplikacji zostały wybrane dwie z nich, odległość drogowa (obliczanie najkrótszej trasy samochodowej) oraz lotnicza (bezpośrednia).

W celu obliczenia odległości drogowej potrzebne są dane na temat dróg. Tutaj podobnie jak w wypadku części klienckiej wykorzystano mapy OpenStreetMap[[5]](#endnote-5). Dane na temat dróg można pobrać ze strony [http://download.geofabrik.de].

## Komunikacja

Proces obliczania rozwiązania może być czasochłonny. Istnieje kilka sposobów komunikacji z serwerem w celu ustalenia czy rozwiązanie zostało już obliczone lub na jakim etapie właśnie się proces obliczający znajduje. Sposoby te zostaną poniżej opisane oraz porównane pod względem szybkości działania oraz ilości przesyłania danych. Zostaną także przedstawione zalety oraz wady każdego z użytych rozwiązań.

### XHR

Protokół XMLHttpRequest(XHR)[[6]](#endnote-6) jest interfejsem API pozwalającym przesyłać informację do i z serwera za pomocą JavaScript. Dzięki XHR nie jest konieczne odświeżenie strony WWW w celu jej aktualizacji jednak można to zrobić asynchronicznie pod kontrolą kodu JavaScitpt. Protokół ten nie tylko udostępnia komunikację, ale również ją znacząco upraszcza. Przeglądarka za programistę zajmuje się wieloma aspektami: między innymi zarządzaniem połączenia, negocjacją protokołów lub formatowaniem zapytań HTTP. Za pomocą XHR możemy przesyłać dane zarówno binarne jak i tekstowe.

var xhr = new XMLHttpRequest();

xhr.open('GET', 'url');

xhr.responseType = 'arraybuffer';

xhr.onload = function() {

if (this.status == 200) {

var arraybuffer = this.response;

/\* \*/

}

};

xhr.send();

Jak pokazuje powyższy przykład, na początku tworzymy obiekt XMLHttpRequest a następnie ustawiamy typ zwracanych danych, jako ‘arraybuffer’.

Protokół XHR potrafi skutecznie pobierać aktualizacje z serwera. Klient wysyła żądanie, a następnie serwer na nie odpowiada. Nie istnieje jednak możliwość inicjowania połączenia w odwrotnym kierunku.

### Odpytywanie w protokole XHR

Jedną ze strategii odbierania aktualizacji z serwera jest okresowe wysyłanie zapytania do serwera w regularnych odstępach czasu. Jeżeli na serwerze pojawi się odpowiedz to jest ona wysyłana, w przeciwnym wypadku odsyłana jest pusta wiadomość. Jest to bardzo proste w implementacji. Ważną kwestią jest ustawienie odpowiedniego interwału, zbyt krótki generowałby zbędny ruch orz obciążenie klienta oraz serwera, natomiast zbyt długi opóźnione aktualizacje.

function getUpdates(url){

var xhr = new XMLHttpRequest();

xhr.open(‘GET’, url);

xhr.onload = function(){ … };

xhr.send();

}

setInterval(getUpdates(‘/vrp/solver/12345’), 500);

Na powyższym przykładzie widać okresowe wysyłanie zapytania do serwera, co pół sekundy.

Kolejną możliwością jest długotrwałe odpytywanie w protokole XHR. Modyfikując tak ostatni sposób, aby nie zwracać pustej odpowiedzi a zamiast tego utrzymywać otwarte połączenie aż do pojawienia się aktualizacji. Dzięki temu klient może otrzymać odpowiedź tak szybko jak pojawi się aktualizacja. Dzięki zastosowaniu takiej techniki zmniejsza się narzut wprowadzany przez odpytywanie.

function getUpdates(url){

var xhr = new XMLHttpRequest();

xhr.open(‘GET’, url);

xhr.onload = function () {

…

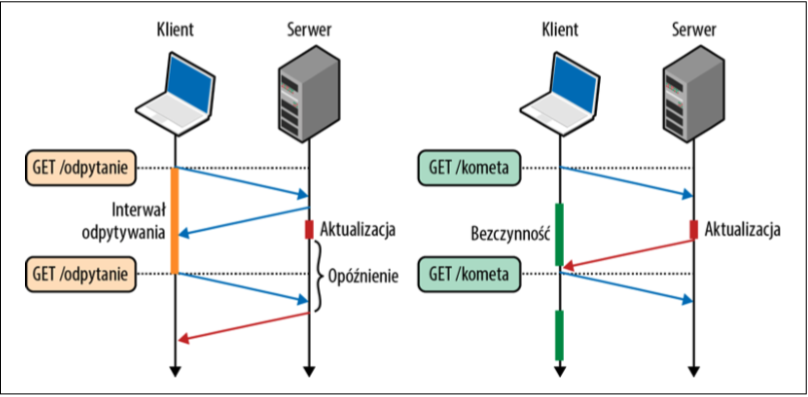
getUpdates(url);

}

}

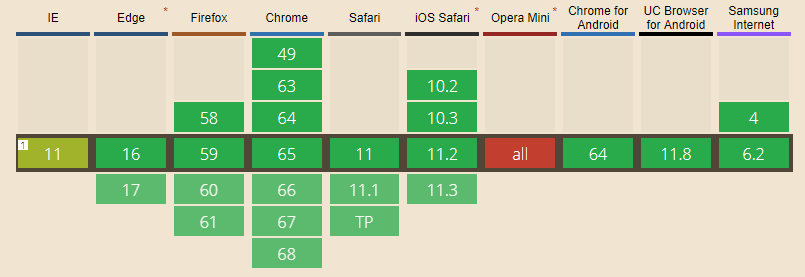
getUpdates(‘/vrp/solver/1234’);

Jak widać powyżej, po przetworzeniu odpowiedzi od serwera jest wysyłane nowe żądanie. Takie podejście posiada też wady, może generować większy ruch niż okresowe, jeżeli częstotliwość odpowiedzi z serwera jest bardzo duża. Dodatkowo utrzymanie każdego połączenia na serwerze jest kosztowne oraz wykorzystuje zasoby.



Rys. 10

Na powyższym rysunku po lewej stronie możemy zobaczyć odpytywanie zwykłe oraz długotrwałe z prawej strony. Na poniższym zrzucie ze strony: <https://caniuse.com/#feat=xhr2> możemy zobaczyć, że prawie wszystkie przeglądarki obsługują w pełni protokół XHR2(oznaczone na zielono). Jedynym wyjątkiem jest Opera Mini oraz Internet Explorer w wersji, 11 który nie obsługuje typy zwracanych danych, jako ‘*json*’.



Rys. 11

### SSE

Server-Sent Events jest standardem, który umożliwia pozwala na wysyłanie powiadomień z serwera do klienta w czasie rzeczywistym. Składa się z dwóch komponentów: typu danych ‘*text/event-stream*’ oraz interfejs *EventSource* w przeglądarce.

var source = new EventSource("/path/to/stream-url");

source.onopen = function () { ... };

source.onerror = function () { ... };

source.addEventListener("foo", function (event) {

processFoo(event.data);

});

source.onmessage = function (event) {

log\_message(event.id, event.data);

if (event.id == "CLOSE") {

source.close();

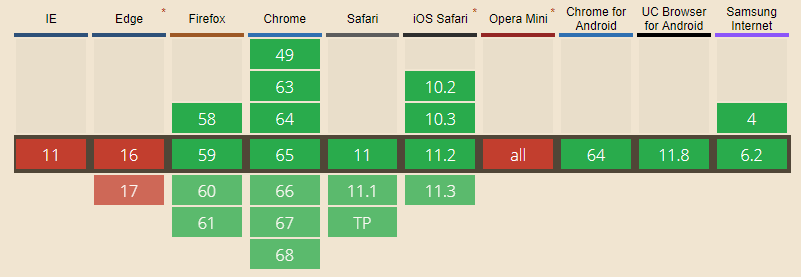
}

}

Na powyższym przykładzie widać na początku otworzenie połączenia do punktu końcowego połączenia. W kolejnych linijkach mamy obsługę zdarzeń ‘onload’ oraz ‘onerror’. Kolejna linijka przedstawia subskrypcję zdarzenia typu ‘foo’. Ostatnia metota jest wywoływana zawsze, niezależnie od zdarzenia, jakie wystąpiło. Identyfikator, typ oraz granice każdego komunikatu są definiowane przez protokół strumieniowania zdarzeń. Składa on się z predefiniowanych nazw pól:

* **event**: nazwa zdarzenia, takie jak ‘message’ lub ‘foo’
* **data:** pole zawierające dane
* **retry:** liczba definiująca czas ponownego połączenia w przypadku utracenia połączenia
* **id**: identyfikator wiadomości

Protokół SSE oferuje wbudowaną obsługę utraty przerwanych połączeń oraz ponowne wysłanie utraconych komunikatów. Jak widać na poniższym zżucie ekranu[[7]](#endnote-7) interfejs *EventSource* wspierany jest przez większość nowoczesnych przeglądarek. Wyjątkiem tutaj są przeglądarki Edge oraz Internet Explorer.



Rys. 12

Dodatkowo protokół ten nie jest przystosowany do przesyłania danych binarnych. Ograniczenie to jednak można ominąć przy użyciu metody base64 do zakodowania obiektu binarnego. Dzięki temu jednak jest wprowadzany znaczny narzut(około 33%[[8]](#endnote-8)).

### WebSocket

WebSocket jest protokołem wpierającym dwukierunkową transmisję danych binarnych lub tekstowych. Składa się z dwóch elementów[[9]](#endnote-9). Pierwszą z nich jest ustalenie parametrów transmisji danych na poziomie komunikatów HTTP. Kolejną zaś jest wymiana danych z użyciem niewielkiej wielkości ramek.

Do nawiązania połączenia WebSocket początkowo klient wysyła żądanie do serwera. Jest to zwykłe żądanie HTTP z dodatkowo ustawionymi nagłówkami. Przykładowe żądanie pokazane jest poniżej.

GET /chat HTTP/1.1

Host: server.example.com

Upgrade: websocket

Connection: Upgrade

Sec-WebSocket-Key: x3JJHMbDL1EzLkh9GBhXDw==

Sec-WebSocket-Protocol: chat, superchat

Sec-WebSocket-Version: 13

Origin: <http://example.com>

Natomiast odpowiedź serwera wygląda następująco:

HTTP/1.1 101 Switching Protocols

Upgrade: websocket

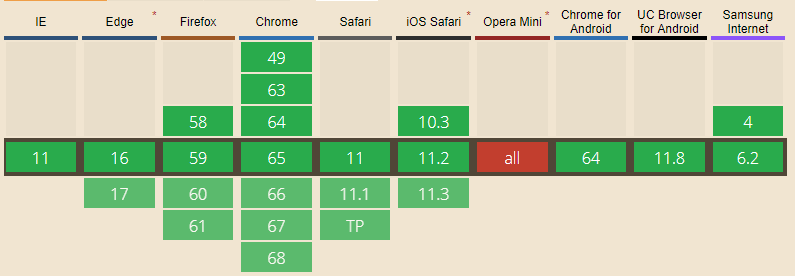
Connection: Upgrade

Sec-WebSocket-Accept: HSmrc0sMlYUkAGmm5OPpG2HaGWk=

Sec-WebSocket-Protocol: chat

Dzięki takiemu podejściu oba protokoły wykorzystują ten sam port TCP 80. Przykładowymi nagłówkami, które możemy użyć do negocjacji połączenia są:

* Sec-WebSocket-Version – definiuje wersję protokołu WebSocket
* Sec-WebSocket-Key – gnerowany przez klienta klucz, upewniający się, że serwer obsługuje daną wersję protokołu
* Sec-WebSocket-Accept – odpowiedź serwera na wartość klucza Sec-WebSocket-Key
* Sec-WebSocket-Protocol – negocjacja protokołu aplikacji
* Sec-WebSocket-Extensions – nagłówek wykorzystywany do negocjacji rozszerzeń protokołu WebSocket



Rys. 13

Protokół WebSocket jest wspierany przez wszystkie nowoczesne przeglądarki (z wyłączeniem Opery Mini)

### Porównanie

W ramach pracy zostały zaimplementowane wszystkie z wymienionych wyżej sposobów wymiany informacji z serwerem. Każde z nich posiada pewne wady. Tabela poniżej próbuje w

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **XHR** | | **SSE** | **WebSocket** |
| **Odpytywanie** | **Kometa** |
| Kompresja |  |  |  |  |
| Wsparcie |  |  |  |  |
| Łatwość w implementacji |  |  |  |  |
| Multiplex |  |  |  |  |
| Duplex |  |  |  |  |
| Pamięć podręczna |  |  |  |  |
| Dane binarne |  |  |  |  |
| Wznawianie połączenia |  |  |  |  |
| Narzut komunikatów |  |  |  |  |
| Łatwość w implementacji |  |  |  |  |
|  |  |  |  |  |

Kompresja, wsparcie, szyfrowanie, multiplex, duplex, pamięć podręczna, przesyłanie danych binarnych, wznawianie połączenia, ponowne wysyłanie wiadomości po utracie połączenia, łatwość w implementacji, wsprarcie firewalli

- napisac ze xhr long poll wydaje się być dobrym sposobem w tym przypadku i tutaj link dlaczego tak nie jest

Wykres ile pobiera danych, wyłączona kompresja, protoków http w wersji 1.1 a nie 2.0.

## Dostęp z urządzeń mobilnych

# Walidacja rozwiązania

# Podsumowanie

Wykaz literatury

Wykaz tabel

**Nie można odnaleźć pozycji dla spisu ilustracji.**

Wykaz rysunków

**Nie można odnaleźć pozycji dla spisu ilustracji.**

1. http://www.wneiz.pl/nauka\_wneiz/frfu/77-2015/FRFU-77-349.pdf [↑](#endnote-ref-1)
2. https://andresjaquep.files.wordpress.com/2008/10/2627477-clasico-dantzig.pdf [↑](#endnote-ref-2)
3. http://www.sciencedirect.com/science/article/pii/S0166218X04002860 [↑](#endnote-ref-3)
4. https://angular.io/ [↑](#endnote-ref-4)
5. openStreetMap [↑](#endnote-ref-5)
6. https://www.w3.org/TR/XMLHttpRequest/ [↑](#endnote-ref-6)
7. Zrzut wykonany ze strony: https://caniuse.com/#feat=eventsource [↑](#endnote-ref-7)
8. https://kb.iu.edu/d/aepm [↑](#endnote-ref-8)
9. https://tools.ietf.org/html/rfc6455 [↑](#endnote-ref-9)