Streszczenie

Obecna dokumentacja zawiera zbiorcze dane, przedstawiające etapy wytwarzania oprogramowania, jak i dokumentuje rzeczy wdrożone w system. Efektem prac projektu dyplomowego inżynierskiego była aplikacja wspierająca generowanie tras w firmie logistyczne. Główną technologią wykorzystaną w projekcie jest ASP.NET MVC. Głównym zakresem pracy było stworzenie na podstawie istniejących algorytmów systemu, który umożliwi generowanie tras w firmie logistycznej oraz ich wizualizację za pomocą map online w przeglądarce internetowej. Podczas projektu powstała aplikacja posiadająca wyżej wymienione wymagania. Aby system był funkcjonalny dla potencjalnego klienta powstała również odrębna część systemu skupiająca się na zarządzaniu użytkownikami oraz część modelu przechowująca dane, aby system mógł być wdrożony w przyszłe przedsiębiorstwo. Ważnym aspektem naszego systemu jest fakt, że generuje on optymalne trasy przejazdu według zadanych parametrów, więc realistycznie może zmniejszyć koszty funkcjonowania firmy transportowej potencjalnego klienta. Dokument ten w głównej mierze przedstawia budowę systemu, jak i metodyki wykorzystane podczas wytwarzania. Przedstawiono dokładnie wizję systemu i założenia ustalone przed etapem implementacyjnym oraz dokładnie opisano poszczególne etapy wytwarzania oprogramowania.

Abstract

This documentation summarizes data showing the stages of software development, as well as documenting things implemented in the system. The outcome of the project was an engineering diploma application assisting the process of generation daily routes for couriers in a logistics company. The main technology used in the project is ASP.NET MVC. The main scope of this work was the creation, based on existing algorithms, of a system that will allow a logistics company to generate routes and visualize them using online maps systems (such as GoogleMaps) in web browser. During this project, the application meeting above-mentioned requirements was created. To make the system functional for a potential client a separate part of the system that focused on managing users, and part of the model that stores data for the system in a way that it could be implemented in the future potential client company were also created. An important aspect of our system is that it generates the optimal route according to preset parameters, so realistically can reduce the operating costs of transport company - potential customer. This document mainly shows the construction of the system and methodology used during development. The vision of the system and objectives set before the implementation stage are presented accurately and the various stages of software development were also presented accurately.

Spis treści

[1 Raport Końcowy 4](#_Toc405759377)

[1.1 Dane dotyczące projektu 4](#_Toc405759378)

[1.2 Kontekst projektu 4](#_Toc405759379)

[1.3 Osiągnięte rezultaty 5](#_Toc405759380)

[1.4 Proces realizacji projektu 6](#_Toc405759381)

[1.5 Zmiany w trakcie projektu 7](#_Toc405759382)

[1.6 Podsumowanie 8](#_Toc405759383)

[2 Dokumentacja techniczna 10](#_Toc405759384)

[2.1 Dokumentacja wizji technicznej 10](#_Toc405759385)

[2.2 Zawartość projektu 11](#_Toc405759386)

[2.2.1 Wykorzystane algorytmy 11](#_Toc405759387)

[2.2.2 Algorytm HeldKarp’a zaimplementowany w programie: 11](#_Toc405759388)

[2.2.3 Metoda obchodząca limit GoogleDirections API 12](#_Toc405759389)

[2.3 Instrukcja użytkowania 14](#_Toc405759390)

[3 Dokumentacja procesowa 29](#_Toc405759391)

[3.1 Product backlog i opis infrastruktury projektu 29](#_Toc405759392)

[3.1.1 Cel Systemu 29](#_Toc405759393)

[3.1.2 Kontekst systemu 29](#_Toc405759394)

[3.1.3 Wymagania funkcjonalne 29](#_Toc405759395)

[3.1.4 Wymagania poza funkcjonalne 30](#_Toc405759396)

[3.1.5 Scenariusze użycia 31](#_Toc405759397)

[3.1.6 Product Backlog 32](#_Toc405759398)

[3.1.7 Infrastruktura projektu 32](#_Toc405759399)

[3.2 Sprint pierwszy 36](#_Toc405759400)

[3.2.1 Funkcje i ograniczenia produktu, znane błędy 37](#_Toc405759401)

[3.2.2 Zmiany w stosunku do poprzedniego wydania 37](#_Toc405759402)

[3.2.3 Wymagania na środowisko i instrukcja uruchomienia 37](#_Toc405759403)

[3.2.4 Instrukcja użytkowania 37](#_Toc405759404)

[3.3 Drugi Sprint 38](#_Toc405759405)

[3.3.1 Funkcje i ograniczenia produktu, znane błędy 39](#_Toc405759406)

[3.3.2 Zmiany w stosunku do poprzedniego wydania 39](#_Toc405759407)

[3.3.3 Wymagania na środowisko i instrukcja uruchomienia 39](#_Toc405759408)

[3.3.4 Instrukcja użytkowania 39](#_Toc405759409)

[3.4 Trzeci Sprint 41](#_Toc405759410)

[3.4.1 Funkcje i ograniczenia produktu, znane błędy 42](#_Toc405759411)

[3.4.2 Zmiany w stosunku do poprzedniego wydania 42](#_Toc405759412)

[3.4.3 Wymagania na środowisko i instrukcja uruchomienia 43](#_Toc405759413)

[3.4.4 Instrukcja użytkowania 43](#_Toc405759414)

[3.5 Czwarty Sprint 45](#_Toc405759415)

[3.5.1 Funkcje i ograniczenia produktu, znane błędy 46](#_Toc405759416)

[3.5.2 Zmiany w stosunku do poprzedniego wydania 46](#_Toc405759417)

[3.5.3 Wymagania na środowisko i instrukcja uruchomienia 47](#_Toc405759418)

[3.5.4 Instrukcja użytkowania 47](#_Toc405759419)

[4 Słownik pojęć i skrótów 47](#_Toc405759420)

[5 Bibliografia 47](#_Toc405759421)

# Raport Końcowy

## Dane dotyczące projektu

Opiekun projektu: mgr inż. Tomasz Gawron

Klient projektu: mgr inż. Tomasz Gawron

Uczestnicy projektu:

* Paweł Troka, Katedra Algorytmów i Modelowania Systemów
* Łukasz Adrych, Katedra Inteligentnych Systemów Interaktywnych

Temat projektu:

* System wspierający generowanie tras w firmie logistycznej

## Kontekst projektu

* Charakterystyka projektu

Aplikacja realizowana w ramach projektu miała być aplikacją webową, która na późniejszym etapie wytwarzania mogła stać się systemem dla małej firmy logistycznej. Technologia oraz rozwiązanie problemów zostało pozostawione do wyboru zespołu wytwarzającego oprogramowanie. Została zasugerowana metodyka zwinna, która towarzyszy wytwarzaniu wszystkim projektom inżynierskim na Katedrze Inżynierii Oprogramowania.

* Cele projektu

Celem projektu było wytworzenie działającej aplikacji do generowania optymalnych tras przejazdu według wybranych kryteriów, oraz wizualizacji wyników w dowolnym systemie map online. W celu osiągnięcia wyznaczonego celu zostały zrealizowane poniższe zadania:

* Zapoznanie się z istniejącymi systemami do rozwiązywania problemu optymalizacji trasy przejazdu dostępnymi w Internecie (zarówno komercyjne jak i darmowe).
* Zapoznanie się z API map online za pomocą których można rozwiązać zadany problem.
* Przyrostowe wytwarzanie oprogramowanie, wzbogacane nowymi funkcjonalnościami.
* Potencjalny klient

Głównym klientem naszej aplikacji na etapie wytwarzania był mgr inż. Tomasz Gawron – opiekun projektu. W początkowym stadium planowania projektu planowaliśmy również współpracę z zewnętrzną firmą, ale ze względu na poufność niektórych danych, nie udało się ostatecznie sfinalizować potrzebnych wymagań od tej firmy, potrzebnych do wytworzenie systemu. Ostateczny format niektórych danych wyjściowych nie został skonsultowany z potencjalnym odbiorcą, i został określony przez zespół developerski.

* Przykład potencjalnego klienta

SPU jest niewielką firmą logistyczną, która zatrudnia ok. 50 osób i funkcjonuje na terenie Trójmiasta. SPU specjalizuje się w transporcie materiałów budowlanych i działa na rynku już od 10 lat. Firma otrzymuje zlecenia na transport materiałów budowlanych składowanych w magazynie do odbiorców końcowych. SPU posiada własny magazyn o powierzchni 10 tys m2, zlokalizowany poza miastem. Przez magazyn przechodzą wszystkie przesyłki obsługiwane przez SPU. W tej samej lokalizacji znajduje się również biuro firmy. Część kierowców jest zatrudniona bezpośrednio przez firmę, ale SPU dodatkowo współpracuje z kierowcami zewnętrznymi. Firma obecnie rozwija się i potrzebuje systemu wspomagającego generowanie tras dziennych dla kierowców. Obecna metoda planowania przejazdów jest nieefektywna, naraża firmę na ponoszenie dodatkowych, niepotrzebnych kosztów, jak również powoduje problemy z organizacją zadań.

## Podział pracy

Początkowy podział pracy był bardzo prosty i wynikał bezpośrednio z założonej technologii. Jako, że model programowania MVC składa się z trzech składowych oraz członków zespołu było trzech postanowiono każdemu członkowi zespołu przydzielić zależnie od umiejętności jedną część ze wzorca MVC.

* Model: Łukasz Adrych
* View: Daniel Milewski
* Controller: Paweł Troka

Jednak po odejściu Daniela Milewskiego z zespołu oraz z powodu niewielkich postępów wymagane były zmiany w podziale pracy. Zespół, z powodu bardzo niewielkich zasobów czasowych postanowił podzielić się możliwie najefektywniej, w taki sposób, aby członkowie zespołu sobie nie przeszkadzali w pracy, a wręcz praca jednego członka wymuszała pracę drugiego.

* Dokumentacja: Łukasz Adrych
* Aplikacja: Paweł Troka

Taki podział okazał się bardzo efektywny, każdy kolejny „commit” kodu wykonywany przez Pawła Troka w pewnym sensie wymuszał opisanie go w dokumentacji przez Łukasza Adrycha. Tak samo jak pytania Łukasza Adrycha o daną funkcjonalność (którą akurat opisywał) wymuszały często jej dopracowanie przez Pawła Troka i dogłębne przetestowanie przez cały zespół.

## Osiągnięte rezultaty

W ramach projektu dyplomowego inżynierskiego została wytworzona aplikacja internetowa umożliwiająca generowanie optymalnych tras dziennych, oraz ich wizualizację w systemie za pomocą map online. Produkt jest w pełni funkcjonalny, jak założono w podstawowych wymaganiach projektu. Jednak z uwagi na mniejszy skład w stosunku do założeń projekt oraz odejścia jednej osoby w trakcie pisania pracy nie udało nam się zrealizować żadnych dodatkowych funkcji, dzięki którym nasz produkt zyskałby na jakości. Obecna wersja posiada niżej wymienione funkcje:

* Logowanie oraz rejestracja do serwisu
* Zarządzanie prawami użytkownika
* Tworzenie listy kierowców dla pracownika
* Zarządzanie listą pracowników przez administratora/kierownika
* Generowanie trasy dziennej dla kierowcy
* Optymalizacja wyników na podstawie kilku parametrów i wyboru dostawców danych
* Usuwanie i edycja trasy dziennej.
* Wizualizacja trasy w system map online jak i tworzenie wersji do wydruku wygenerowanych danych

Obecna wersja systemu nie został jeszcze wdrożona na serwer i jej uruchomienie jest możliwe przez środowisko Visual Studio 2013. Jednak projekt zawiera wszystkie potrzebne pliki do migracji z serwera developerskiego na serwer produkcyjny. Powstały system był testowany za pomocą testów funkcjonalnych, wszystkie usterki znalezione podczas testów zostały wyeliminowane. Podczas prac nad projektem powstała również dokumentacja projektu, która zawiera wszystkie niezbędne dane opisujące rozwój projektu na przestrzeni przeprowadzonych sprintów. Głównymi punktami zawartymi są dokumentacja techniczna oaz dokumentacja procesowa.

## Proces realizacji projektu

* Organizacja projektu

Harmonogram prac został ustalony samodzielnie, wspólnie ustaliliśmy, że przeprowadzimy cztery sprinty w których zostanie wytworzone oprogramowanie, równolegle z ostatnim sprintem powstała dokumentacja techniczna produktu:

|  |  |
| --- | --- |
| Etap | Termin |
| Sprint 1 | 13.10.2014 – 26.10.2014 |
| Sprint 2 | 27.10.2014 – 09.11.2014 |
| Sprint 3 | 10.11.2014 – 23.11.2014 |
| Sprint 4 | 24.11.2014 – 07.12.2014 |
| Sporządzenie dokumentacji końcowej | 01.12.2014 – 07.12.2014 |

Spotkania z opiekunem systemu planowane były co tydzień we wtorki o 17:15, w których przedstawiane były kolejne wersje systemu wytworzone w ramach konkretnego sprintu, a także konsultowane problemy które wystąpiły w ramach omawianego etapu.

W celu dobrej organizacji prac wykorzystaliśmy narzędzie acunote (<http://rga.acunote.com/projects/47324/sprints/192881/show>). Dokumentacja wytwarzana była przy pomocy Microsoft Word 2010. Zespół korzystał również z zdalnego repozytorium projektu umieszczonego na codeplex.com pod adresem: (<https://rga.codeplex.com>). Klientem wykorzystywanym do połączenia ze zdalnym repozytorium był Team Explorem wbudowany w Visual Studio 2013. W ramach tego repozytorium wersjonowany był kod projektu, dokumentacja nie podlegała wersjonowaniu podczas wytwarzania systemu.

* Metodyki wykorzystane w projekcie

Podstawą metodyką wykorzystaną przy wytwarzaniu aplikacji była metodyka Scrum. Jednak nie była wykorzystana czysta technika Scrum’a co wynikało z typu projektu oraz czasu poszczególnych osób na pracę przy projekcie. Scrum masterem i jednocześnie właścicielem produktu był mgr inż. Tomasz Gawron – opiekun projektu. Było to jedno z odstępstw od metodyki „czystego” Scrum’a w której właściciel produktu nie może być Scrum masterem – odstępstwo to wynikało z faktu, że opiekun projektu był jednocześnie jego klientem. Innym ważnym odstępstwem był fakt, że nie było codziennego Scrum’a co jest naturalne przy tego typu projekcie, który realizowany był głównie poza uczelnią i w różnych godzinach ze względu na różne plany zajęć osób biorących udział w projekcie. Nie odbywał się również daily scrum, przez co zmieniony musiał zostać sposób przydzielania zadań w ramach konkretnego sprintu. Wszystkie osoby w zespole były developerami. Zarządzanie ryzykiem odbywało się poprzez regulację zakresu prac narzuconych do wykonania w danych sprincie jak i w całym projekcie, ponieważ termin złożenia pracy był nieprzekraczalny oraz brak możliwości na rekrutację nowego członka w zespole. Na początku projektu został utworzony Product backlog na podstawie którego ustawiono zadania o najwyższym priorytecie do wykonania na początku Sprintów, oraz te o niższym priorytecie na koniec sprintów, jak i na ostatni Sprint, w którym było wytwarzane oprogramowanie.

## Zmiany w trakcie projektu

Podczas wytwarzania systemu nastąpiła jedna zmiana (która pociągnęła za sobą kolejne) –z zespołu odszedł p. Milewski, tłumacząc swoją decyzję faktem, że niektóre biblioteki które wykorzystamy w projekcie nie będą należeć do wolnego oprogramowania, co w jego przekonaniu było niedopuszczalne.

Przez zaistniałą sytuację na początku Sprintu trzeciego, zespół musiał zmienić priorytety części zadań, aby zrealizować projekt w wyznaczonym terminie. Część zadań został usunięta z Product Backlog gdyż wykonanie ich w mniejszym zespole było niemożliwe. Cała sytuacja sprawiła, że nasz projekt został okrojony z dużej ilości funkcjonalności przez co stracił na jakości. Główne funkcje, które zostały okrojone z systemu to:

* komunikacja wewnętrzna – wysyłanie, przeglądanie, zarządzanie wiadomościami
* filtrowanie, wyszukiwanie użytkowników
* nowy interfejs systemu
* statystyki wygenerowanych tras
* zarządzanie w postaci listy

Nastąpiły również zmiany w narzędziach do wytwarzania oprogramowania. Z kluczowych zmian zespół postanowił zmienić IDE na Visual Studio, gdyż wymóg aby całe używane oprogramowanie było OpenSource nie miał już racji bytu. Była to zmiana na plus, gdyż środowisko to jest bardziej przyjazne do tworzenia kodu niż Xamarin Studio, znacznie więcej rzeczy da się „wyklikać” przez co znacznie można oszczędzić czas przy pisaniu szablonowego kodu. Inna zmiana tyczyła się dostawcy bazy danych, przeszliśmy na MS Serwer, który jest zintegrowany z Visual Studio 2013. Zmiana co prawda nie zauważalna z poziomu kodu, ponieważ interakcja z bazą wykonywana jest przez ORM Entity Framerwok, ale zespół stracił czas na konfigurację serwera MySQL (początkowe rozwiązanie) .

## Podsumowanie

Na podstawie wykonanego projektu można wyciągnąć bardzo dużo wniosków, które pozwolą na lepszą organizację pracy przy innych nowych wyzwaniach oraz podniosą z pewnością jakość pracy uczestników w nowo rozwiązywanych problemach.

Pierwszym bardzo ważnym aspektem który można było zauważyć i który działał trochę na niekorzyść zespołu, to zbyt późne określenie zakresu projektu. Początkowo planowaliśmy pozyskać wymagania od zewnętrznej firmy, jednak z powodu niektórych przepisów działających w tej instytucji nie udało nam ostatecznie nawiązać współpracy. Przez to nieporozumienie straciliśmy około 2-3 tygodnie na pozyskanie wymagań i po części musieliśmy je definiować dopiero w trakcie realizacji projektu. Przy projekcie trwającym 10 tygodni jest to duża strata czasu, o czym zespół przekonał się w końcowym etapie wytwarzania.

Innym bardzo ważnym aspektem, który można wyciągnąć z realizacji tego projektu, to komunikacja w zespole. Zespół developerów przystępował do tego projektu jako nowy zespół, nigdy wcześniej ze sobą nie współpracując, co miało przełożenie w początkowych etapach wytwarzania oprogramowania. Z racji tego że każdy uczęszczał na inną katedrę oraz projekt był realizowany po godzinach developerom ciężko było zorganizować czas na wspólne spotkania i omówienie projektu. Kilka spraw związanych z wytwarzanym oprogramowanie pozostało nie do końca wyjaśnione w początkowym etapie, albo konsultowane za pomocą e-mail przez co zespół stracił sporo czasu na realizację zadania. Również w początkowym etapie wytwarzania nie każdy członek zespołu wiedział co ma dokładnie implementować, przez co prace wykonane w pierwszym i drugim sprincie są niewielkie.

Trzeci ważny aspekt, który można wyciągnąć z tego projektu to szacowanie ryzyka w projekcie. Głównym zagadnieniem związanym z ryzykiem tego projektu był fakt, że posiada on nieprzekraczalny termin oddania oraz skład zespołu w trakcie projektu musi być niezmienny. To właśnie mogło zadecydować o niepowodzeniu tego przedsięwzięcia, ponieważ ze współpracy wycofała się jedna osoba. W związku z tym zakres projektu musiał zostać zmniejszony do minimum, aby można było wydać użyteczną wersję oprogramowania spełniającą podstawowe wymagania zawarte w dokumencie projektu.

Reasumując powyższe wnioski i cały przebieg projektu, można powiedzieć że projekt zakończył się sukcesem, ponieważ pomimo wielu problemów udało nam się stworzyć funkcjonalną wersję oprogramowania która była założeniem projektu. Możemy jednak zauważyć gdzie popełniliśmy błędy i jak się ich wystrzegać na przyszłość. Dobrą praktyką jest również sporządzanie dokumentacji wykonanych prac na bieżąco, co nie zawsze nam się udawało i w końcowym etapie tworzenia dokumentacji programu sprawiło znacznie większy nakład prac nad wytworzeniem dokumentacji niż zakładaliśmy wcześniej.

# Dokumentacja techniczna

## Dokumentacja wizji technicznej

* Użyte technologie
  + ASP.NET MVC 5
  + Razor Engine 4
  + jQuery
  + jQuery UI
  + Bootstrap
  + Ajax
  + Google Directions API (Google Maps API Web Services)
  + Google Directions API (Google Maps Javascript API v3)
  + Google DistanceMatrix API (Google Maps API Web Services)
  + API MapQuest
  + EntityFramework
* Uzasadnienie wyboru

Wybór był stosunkowo jednoznaczny. Narzucone wymagania wskazywały aby powstały system był aplikacją webową, dlatego zespół zdecydował się na ASP.NET MVC. Głównym powodem wyboru tej technologii był fakt, że wszystkie osoby w zespole piszą lub pisały wcześniej w języku C# oraz korzystały z technologii .NET. Tak więc dzięki temu wyborowi nie była niezbędna dodatkowa faza polegająca na zapoznawaniu się z technologią/framework’iem dzięki czemu można było szybciej przystąpić do wykonywania prac związanych z tworzeniem systemu. Kolejną ważną zaletą która skłoniła nas do wyboru tej technologii jest fakt, że dużą część kodu można wygenerować na podstawie utworzonych modeli a następnie dostosować do własnych potrzeb. Fakt ten sprawia, że zespół nie musi pisać dużo szablonowego kodu, a robi to za nas Visual Studio. Przy wyborze ORM’a członkowie zespołu kierowali się prostotą użycia. Z przeglądu literatury jak i źródłem zawartych w Internecie deweloperzy dowiedzieli się, że Entity Framework nie jest zbyt wydajnym ORM’em, ale jest bardzo przyjemny w użyciu i prosty w poznaniu. Główne jego zalety to możliwość stworzenie modeli, na podstawie których zostanie wygenerowana baza danych ("Code First") jak i odwrotną sytuację – wygenerowanie modelu na podstawnie bazy danych ("Database First"). Oczywiście członkowie zespołu jako programiści wykorzystali podejście "Code First". Ostatnią i też ważną zaletą wybranego ORM’a jest fakt, że Entity Framework jest projektem typu open source, więc możemy go dobrowolnie wykorzystać w naszej aplikacji.

## Zawartość projektu

### Wykorzystane algorytmy

Do optymalizacji trasy dla problemu komiwojażera wykorzystano dwa następujące algorytmy:

* BruteForce

Podstawowy algorytm w wielu problemach, jego główną wadą jest złożoność obliczeniowa – O(n!). Zaletą jest taka, że daje zawsze optymalne rozwiązanie ponieważ sprawdza wszystkie permutacje adresów i stwierdza która z tych permutacji jest optymalną trasą po sumie wcześniej założonych kosztów.

* HeldKarp

Znacznie bardziej skomplikowany niż BruteForce, jednak posiada mniejszą złożoność obliczeniową, rzędu O(n22n), co w przypadku dużej liczby przesyłek jest bardzo pożądanym parametrem.

Głównie przy wyborze tych algorytmów zespół kierował się faktem, że dają one optymalne rozwiązania, co było założeniem naszego projektu – „generowanie optymalnych tras w firmie logistycznej”. Rozwiązania problemu komiwojażera dostarczane przez takich dostawców jak MapQuest czy GoogleMaps używają heurystyk do generowania tras – tak więc ich rozwiązania nie są optymalne (chociaż często dosyć bliskie rozwiązaniu optymalnego, dlatego też zespół zdecydował pozostawić w systemie opcję wyboru innego dostawcy optymalizacji trasy niż nasza aplikacja).

### Algorytm HeldKarp’a zaimplementowany w programie:

private double GetMinimumCostRoute(int startVertex, HashSet<int> set, Node root)

{

if (!set.Any())

{

return \_adjacencyMatrix[startVertex, 0];

}

double minCostTotal = double.MaxValue;

int i = 0;

int selectedIdx = i;

root.ChildNodes = new Node[set.Count()];

foreach (int destinationVertex in set)

{

root.ChildNodes[i] = new Node {Vertex = destinationVertex};

double currentVertexCost = \_adjacencyMatrix[startVertex, destinationVertex];

var newSet = new HashSet<int>(set);

newSet.Remove(destinationVertex);

double costFromHere = GetMinimumCostRoute(destinationVertex, newSet, root.ChildNodes[i]);

double newC = currentVertexCost + costFromHere;

if (minCostTotal > newC)

{

minCostTotal = newC;

selectedIdx = i;

}

i++;

}

root.ChildNodes[selectedIdx].Selected = true;

return minCostTotal;

}

*Implementacja algorytmu wykorzystana ze strony: http://www.codeproject.com/Articles/762581/Held-Karp-algorithm-implementation-in-Csharp*

Metoda ta wylicza rekurencyjnie koszty poszczególnych permutacji odcinków tras. Po wyjściu z metody zwracany jest najmniejszy wyliczony koszt według zadanego kryterium, to znaczy czasu lub odległości. Metoda na wejściu otrzymuje dwa parametry: wierzchołek z którego zaczynamy oraz listę wierzchołków do odwiedzenia. Następnie w przypadku jeżeli lista wierzchołków nie jest pusta szukamy optymalnej trasy według zadanych kryteriów w tym punkcie, w którym się aktualnie znajdujemy. Jeżeli nie mamy już punktów do odwiedzenia, to zwracamy odległość między punktem początkowym a wybranym wierzchołkiem na obecnym poziomie zagłębienia rekurencji.

### Metoda obchodząca limit GoogleDirections API

private GoogleMapsApi.Entities.Directions.Response.Route getRoute()

{

GoogleMapsApi.Entities.Directions.Response.Route responseRoute = null;

DirectionsRequest directionsRequest = null;

DirectionsResponse directions = null;

var summaries = "";

var legs = new List<Leg>();

var waypointsOrder = new List<int>();

var warnings = new List<string>();

var allAddresses = new List<string>() { BaseAddress };

allAddresses.AddRange(Addresses);

allAddresses.Add(BaseAddress);

for (int i = 0; i < allAddresses.Count - 1; i += 9) //because free version of GoogleMaps Directions API is limited to maximum 8 waypoints per request

{

directionsRequest = new DirectionsRequest

{

Origin = allAddresses[i],

Destination = (i + 9 < allAddresses.Count) ? allAddresses[i + 9] : allAddresses.Last(),

Waypoints = allAddresses.GetRange(i + 1, Math.Min(8, allAddresses.Count - i - 2)).ToArray(),

Language = "pl",

Avoid = AvoidWay.Tolls,

};

if (routeOptimizationProvider == RouteOptimizationProvider.GoogleMaps)

directionsRequest.OptimizeWaypoints = true;//solves TSP for us, but is limited to 8 points

directions = GoogleMaps.Directions.Query(directionsRequest);

if (directions.Status != DirectionsStatusCodes.OK)

throw new Exception("Nie udało się wygenerować trasy!\nPowód: " + directions.StatusStr);

else

{

legs.AddRange(directions.Routes.First().Legs);

summaries += directions.Routes.First().Summary + Environment.NewLine;

if (routeOptimizationProvider == RouteOptimizationProvider.GoogleMaps)

waypointsOrder.AddRange(directions.Routes.First().WaypointOrder);

warnings.AddRange(directions.Routes.First().Warnings);

}

}

responseRoute = new GoogleMapsApi.Entities.Directions.Response.Route()

{

Legs = legs,

Summary = summaries,

Warnings = warnings.ToArray()

};

if (routeOptimizationProvider == RouteOptimizationProvider.GoogleMaps)

responseRoute.WaypointOrder = waypointsOrder.ToArray();

return responseRoute;

}

Powyższy kod ma za zadanie obchodzenie limitów jakie narzuca GoogleDirections API. Są to limity na liczbę wprowadzanych adresów, jakie przetwarza algorytm, limit wynosi 10. W przypadku naszej aplikacji i do celów wizualizacji taka liczba jest nie wystarczająca ponieważ pozwala na wyznaczenie trasy wyłącznie dla 8 adresów lub mniej ( 1 – początkowy adres bazy, 2-9 adresy dostawy, 10 adres końcowy baza) co w przypadku firmy logistycznej może być dużym ograniczeniem. Cała sytuacja została rozwiązana w pętli powyższego kodu: for (int i = 0; i < allAddresses.Count - 1; i += 9).

Jeżeli nasza trasa ma więcej punktów niż 10, to dzielimy naszą trasę na takie 10 elementowe „podtrasy” i po kolei odpytujemy o te podtrasy w tej pętli, a później łączymy to w aplikacji w jedną trasę.

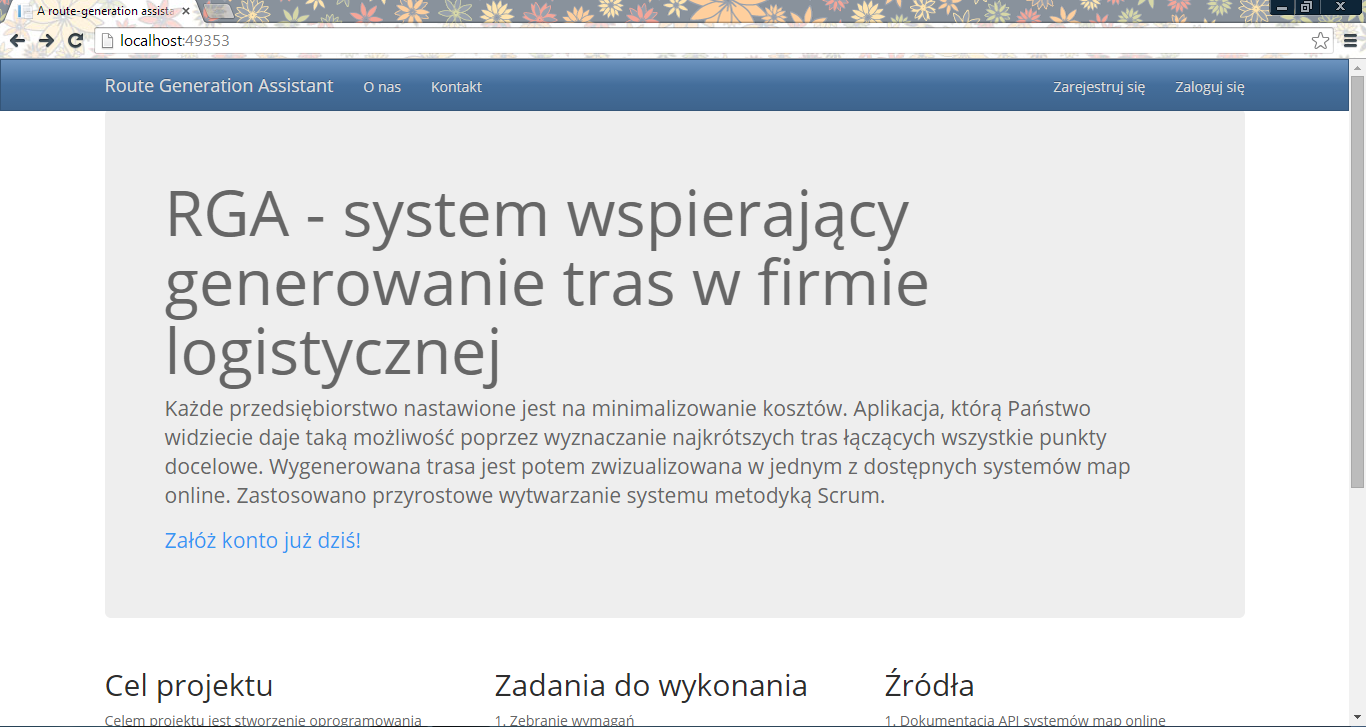
## Instrukcja użytkowania

* Wymagania sprzętowe

Aplikacja wspierająca generowanie tras w firmie logistycznej jest napisana w technologii ASP.NET MVC, w celu zapewnienia poprawności działania aplikacji należy spełnić poniższe kryteria:

* system operacyjny Microsoft Windows Vista lub nowszy
* serwer IIS Express lub lepszy
* środowisko programistyczne Visual Studio 2013
* posiadanie Microsoft .NET Framework 4.5 lub nowszy
* dowolna przeglądarka internetowa spośród (Internet Explorer 10, Mozilla Firefox 17, Opera 17, Chrome 28) lub nowsza
* Uruchomienie aplikacji

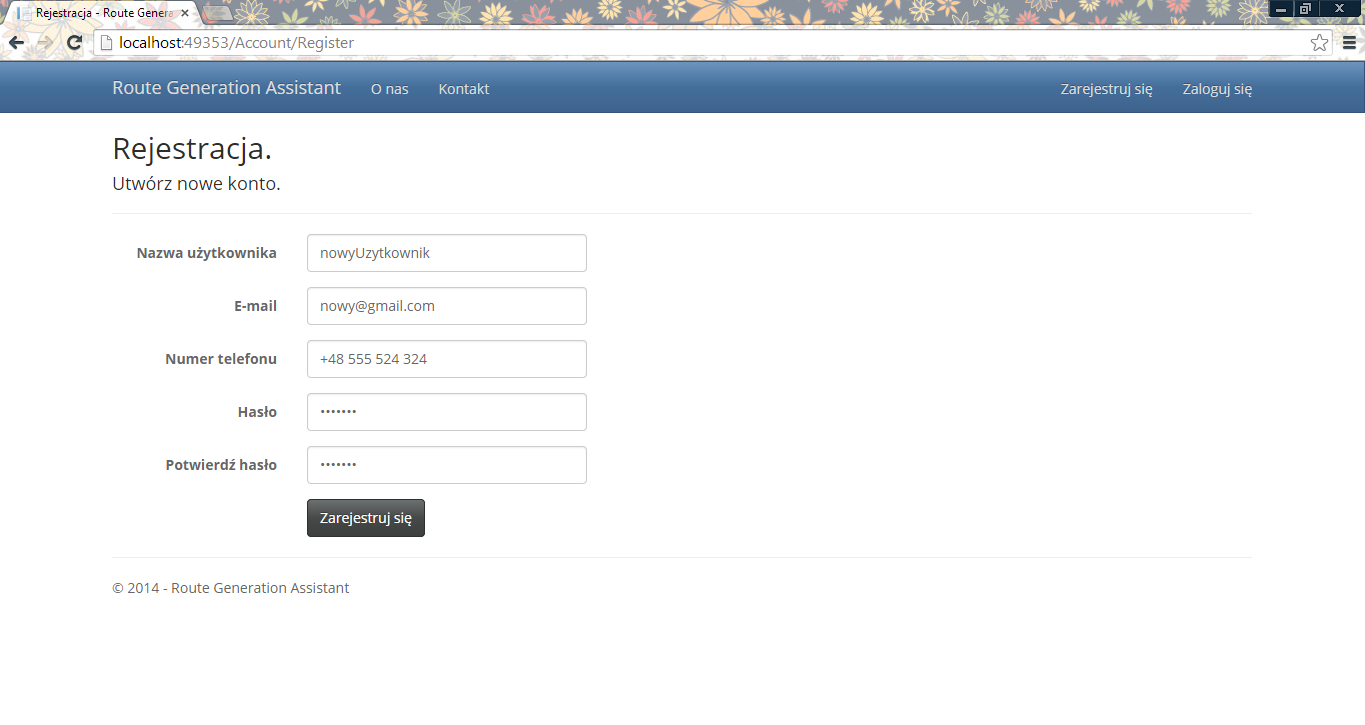
W celu uruchomienia systemu należy uruchomić projekt w Visual Studio i otworzyć projekt Generowania optymalnych tras w firmie logistycznej (RGA.sln) – skrótowa nazwa. Następnie należy kliknąć zieloną strzałkę w programie, przy której mamy ustawioną domyślną przeglądarkę i nastąpi przeniesienie do strony tytułowej systemu.



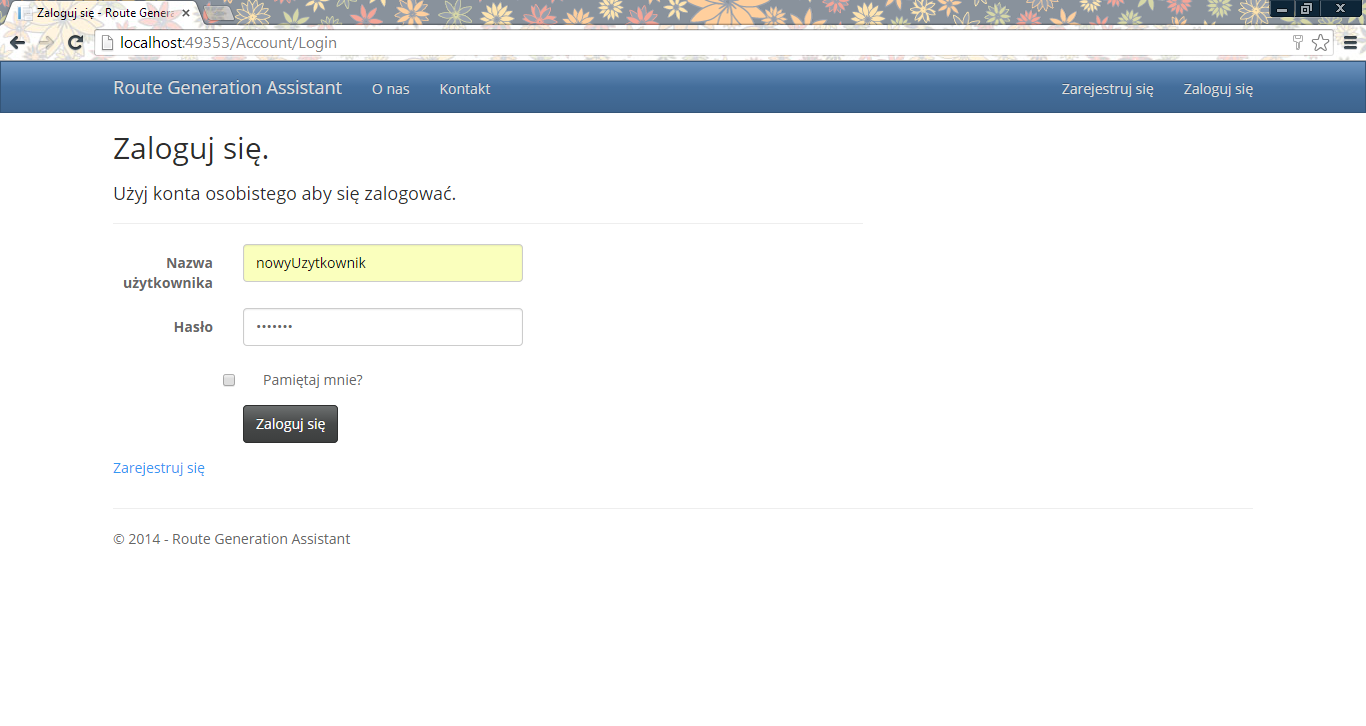
Główne funkcjonalności jakie oferuje nam strona główna (bez zalogowania) to:

* Zarejestruj się
* Zaloguj się
* O nas – informacje ogólne o projekcie
* Route Generation Assistant – odnośnik do strony głównej
* Kontakt – dane twórców serwisu

W celu skorzystania z serwisu wymagane jest posiadanie konta, więc kolejnym krokiem będzie zaprezentowanie widoku do rejestracji:

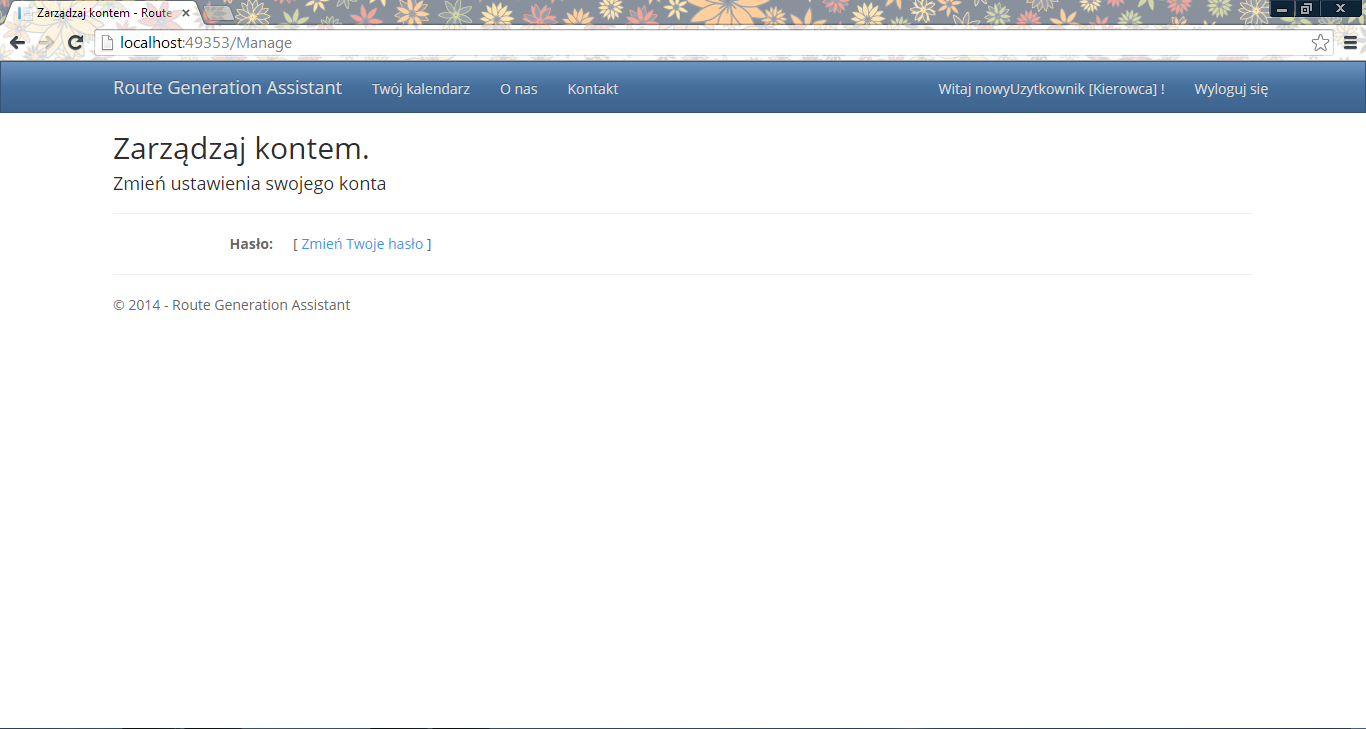


Jeżeli Rejestracja przebiegła prawidłowo zostaniemy automatycznie uwierzytelnieni w systemie. Każda nowa osoba jest domyślnie Kierowcą. Poniżej ekran logowania do systemu:

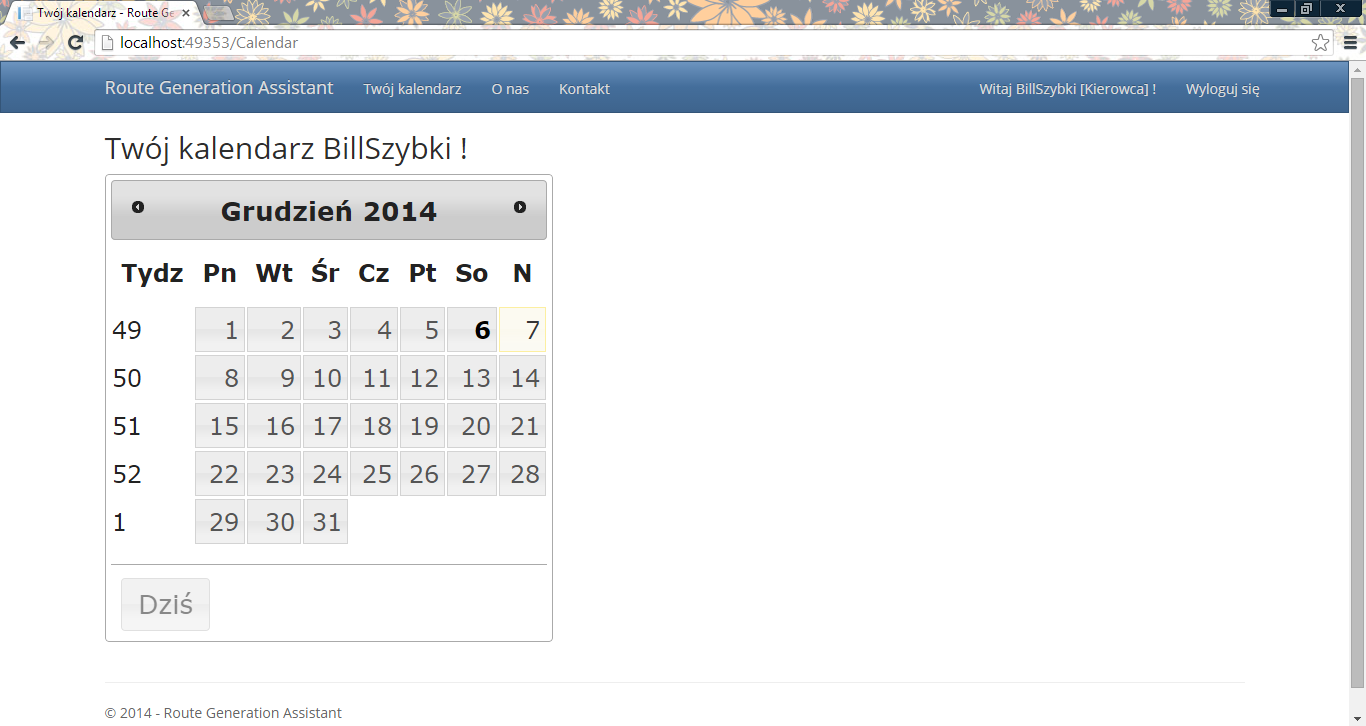


W przypadku poprawnego wprowadzenia danych zostanie przekierowanie do tej samej strony, co w przypadku rejestracji, czyli do strony głównej serwisu jednak otrzymamy dodatkowe uprawnienia i będziemy mogli wykonywać funkcje zależne od posiadanego statusu.

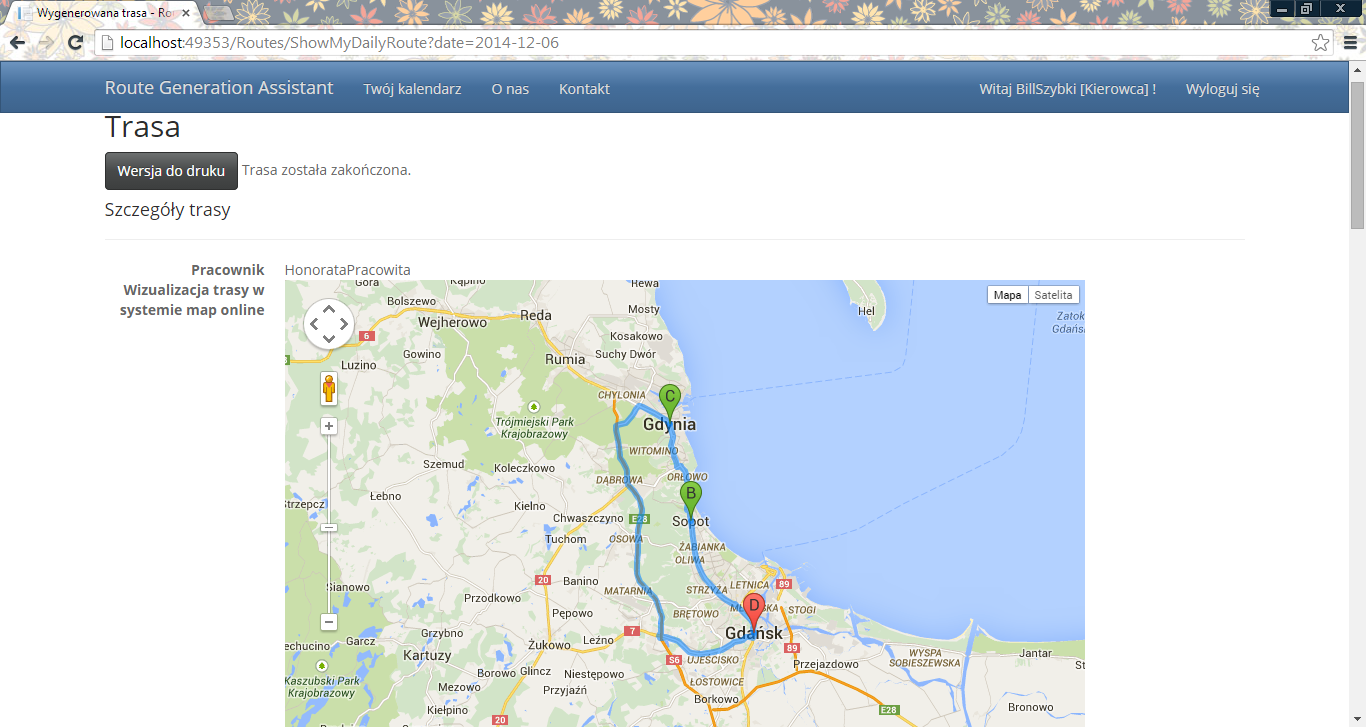
Po poprawnym zalogowaniu znajdujemy się na stronie głównej systemu, w celu zmiany hasła należy kliknąć w swoją nazwę na górnym pasku systemu a zostaniemy przekierowani na stronę z formularzem do zmiany hasła:



Kolejną funkcją udostępnianą dla kierowcy jest kalendarz. Aby przejść do widoku kalendarza należy kliknąć w „Twój Kalendarz” na górnym pasku systemu, a system przekieruje nas do widoku kalendarza. W kalendarzu mamy zaznaczone poszczególne dni tygodnia, dni podświetlone na czarno mają wygenerowaną trasę dzienną:

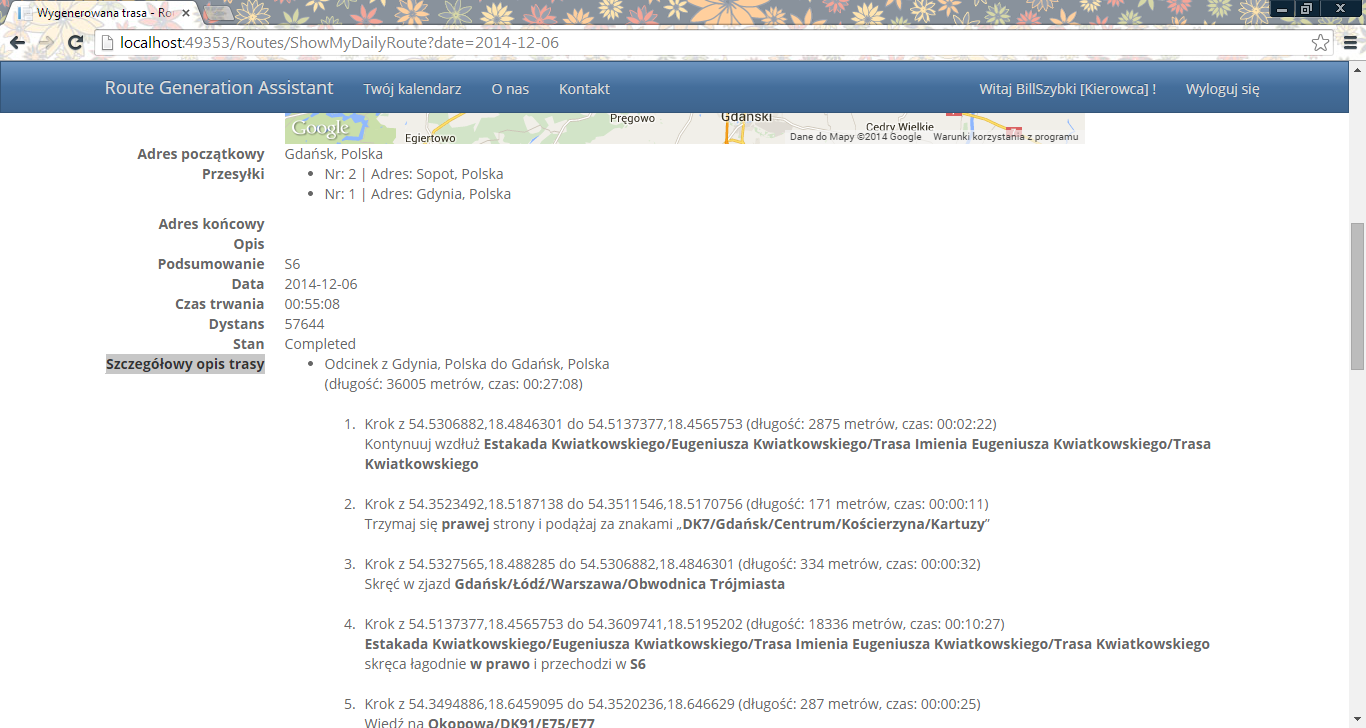


W celu przejścia do podglądu trasy dziennej należy wybrać dzień dla którego wygenerowana jest trasa i kliknąć w przycisk „Zobacz trasę dzienną”. Po wykonaniu tej akcji zostaniemy przeniesieni do widoku wybranej trasy dziennej:

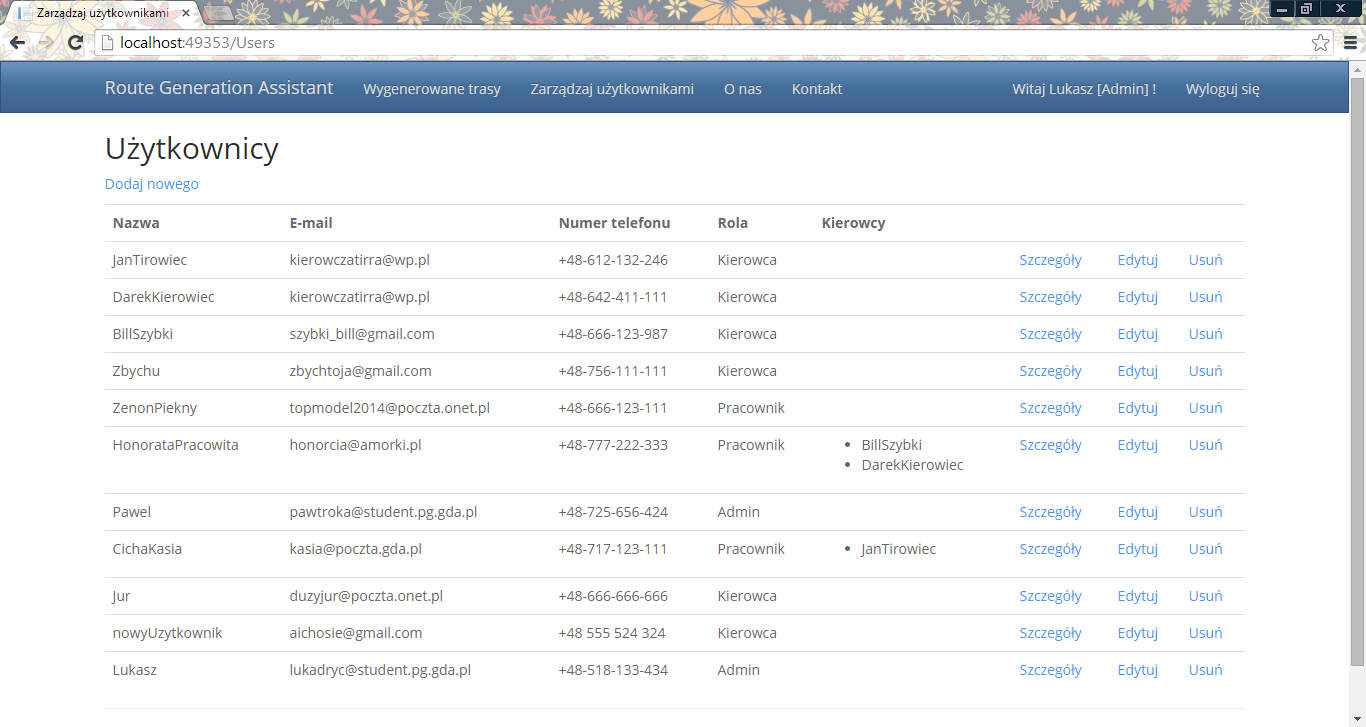


Jak widać na rysunku powyżej mamy zwizualizowaną trasę, wygenerowaną przez naszego pracownika. Poniżej znajduje się szczegółowy opis wygenerowanej trasy, który zawiera takie dane jak:

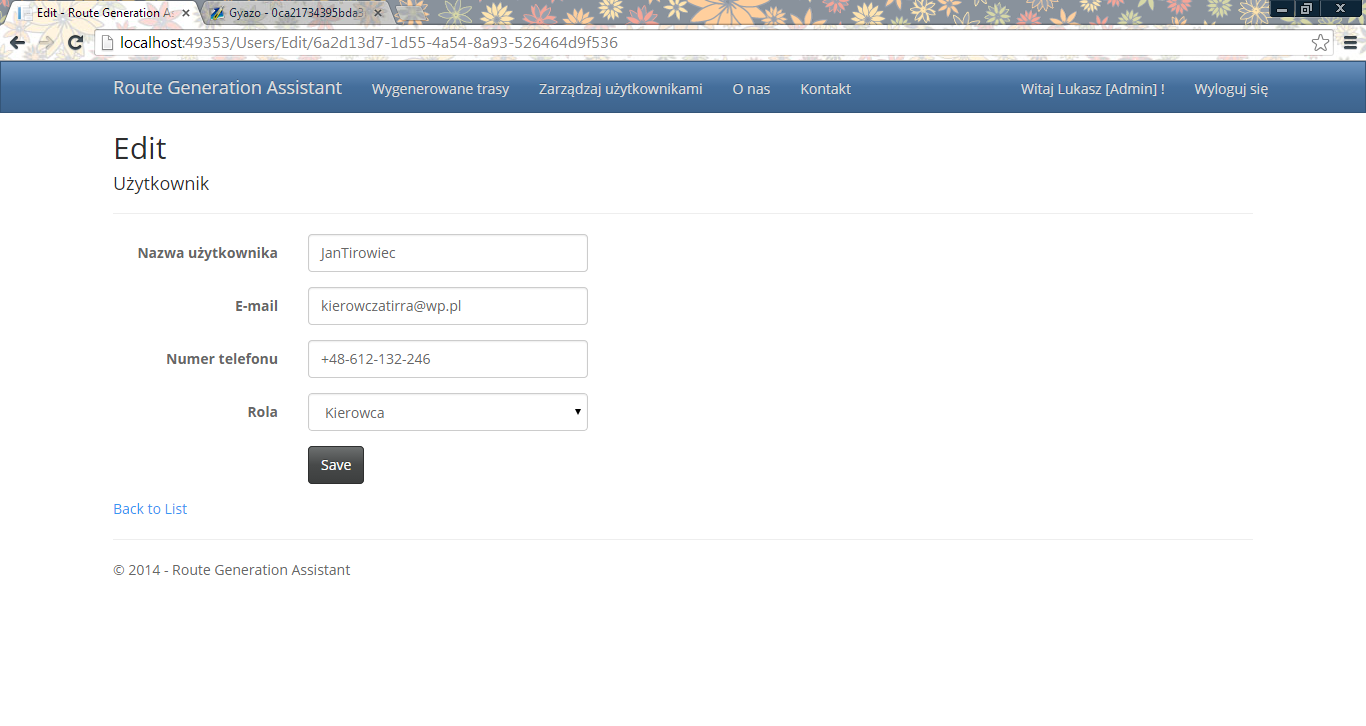
* Adres początkowy przesyłki
* Adres końcowy
* Data
* Szacowany czas trwania
* Długość trasy
* Szczegółowy opis trasy



Następną bardzo ważną funkcjonalnością naszego serwisu jest panel administratora. Po zalogowaniu mamy dostępne dwie główne opcje do wyboru: „Wygenerowane trasy” oraz „Zarządzaj użytkownikami”, wybieramy tą drugą i otrzymujemy listę użytkowników w systemie.

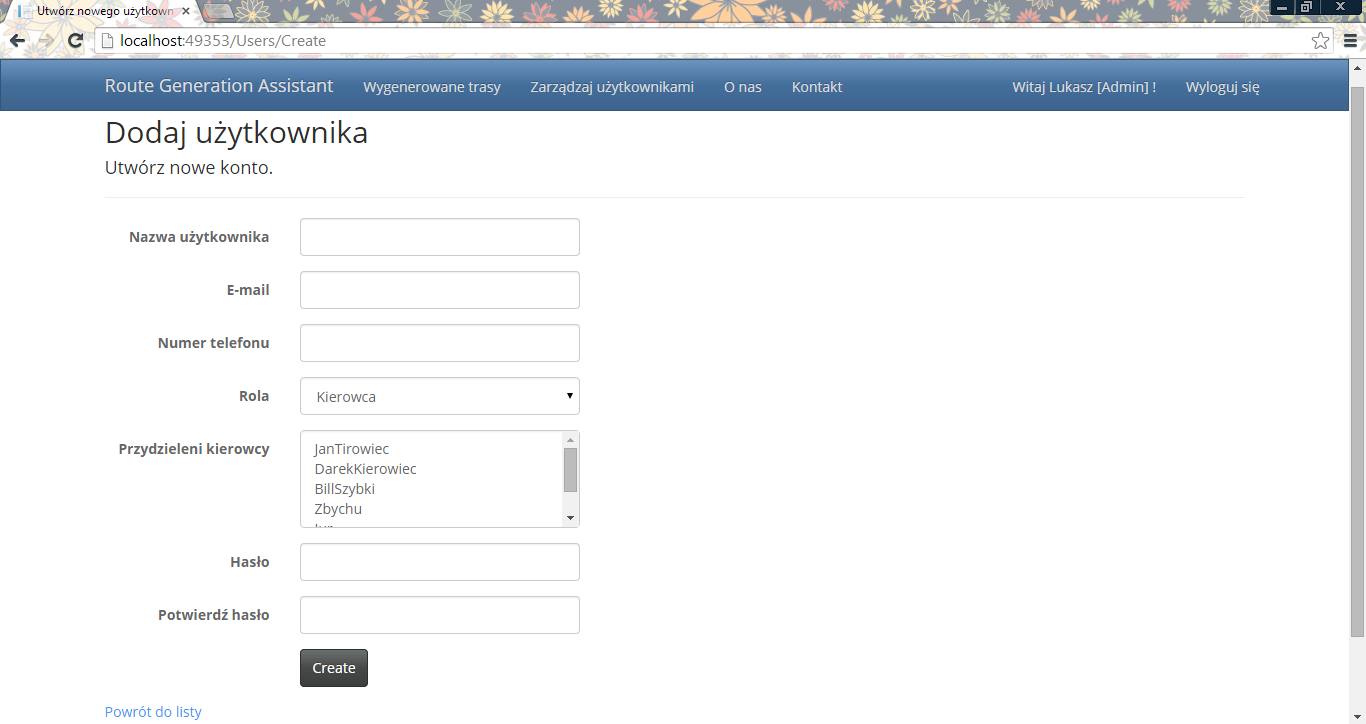


Jak widać na powyższym rysunku każdy użytkownik ma przypisaną rolę, jeżeli ktoś jest pracownikiem może mieć przypisanych kierowców, dla których może generować trasy. Możemy dokonać edycji danych każdej osoby, usunąć osobę z systemu jak również podejrzeć informacje o osobie dzięki opcji szczegóły. Po wybraniu opcji edytuj zostaniemy przeniesieni do widoku edycji użytkownika przedstawionego poniżej:

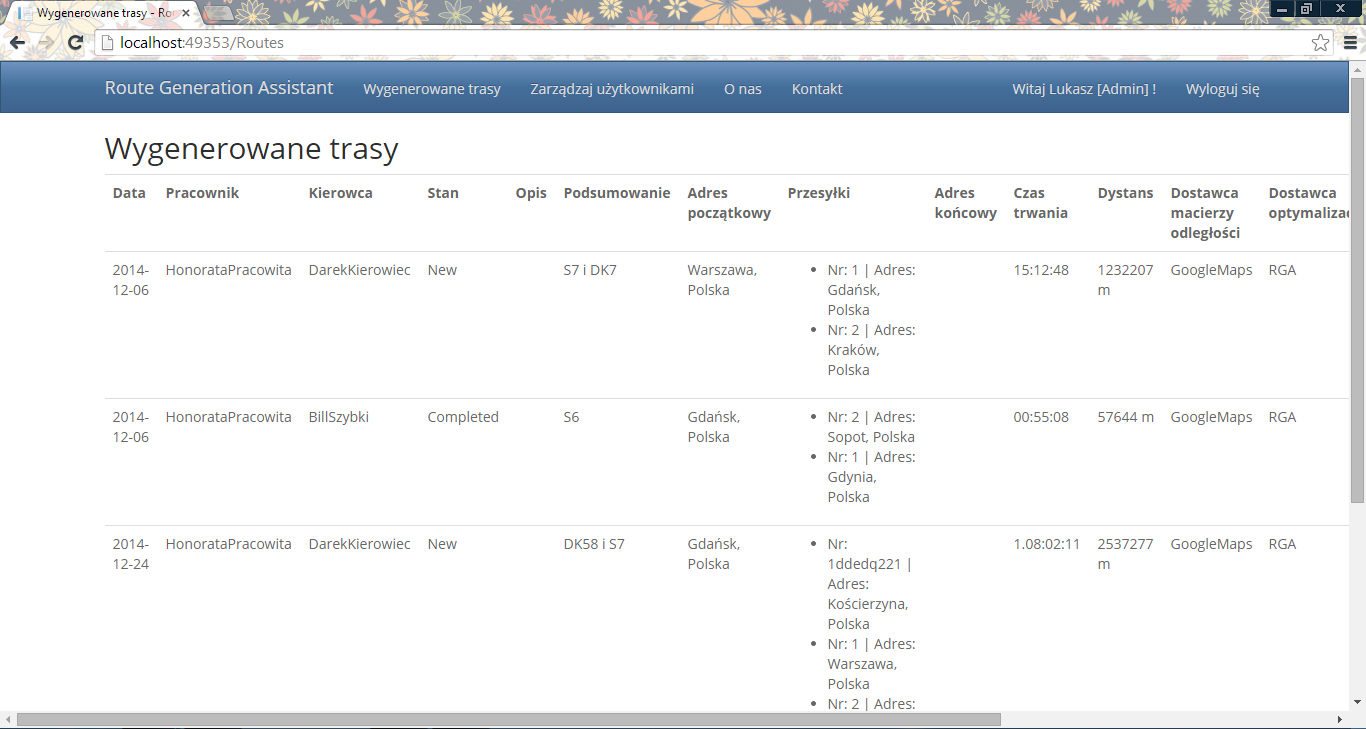


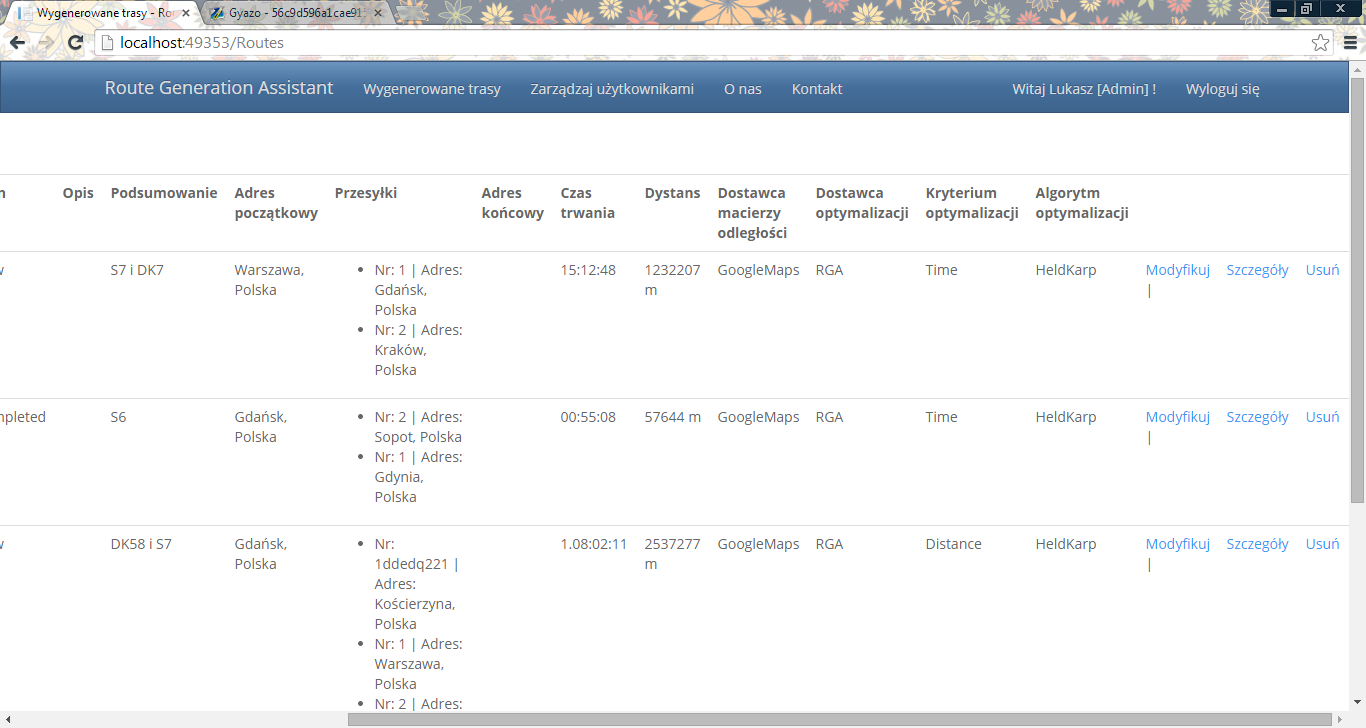
Podczas edycji możemy zmienić główne dane konta użytkownika, jednak zmiana hasła możliwa jest jedynie dla użytkownika systemu.

Każdy administrator posiada opcję tworzenia nowych konto, poprzez specjalnie przygotowany do tego celu formularz. W celu przejścia do formularza dodawania nowego użytkownika należy kliknąć w link „Dodaj nowego”. Dużą zaletą tego formularza jest możliwość tworzenia nowych użytkowników o innych rolach niż kierowca i możliwości jednoczesnego zdefiniowania podwładnych kierowców dla pracownika:



Następnym bardzo ważnym narzędziem jakie posiada administrator jest widok, pozwalający na przeglądanie wygenerowanych tras dziennych. Trasy przedstawione są w tabeli. Administrator posiada opcje przeglądu, modyfikacji i usuwania tras:

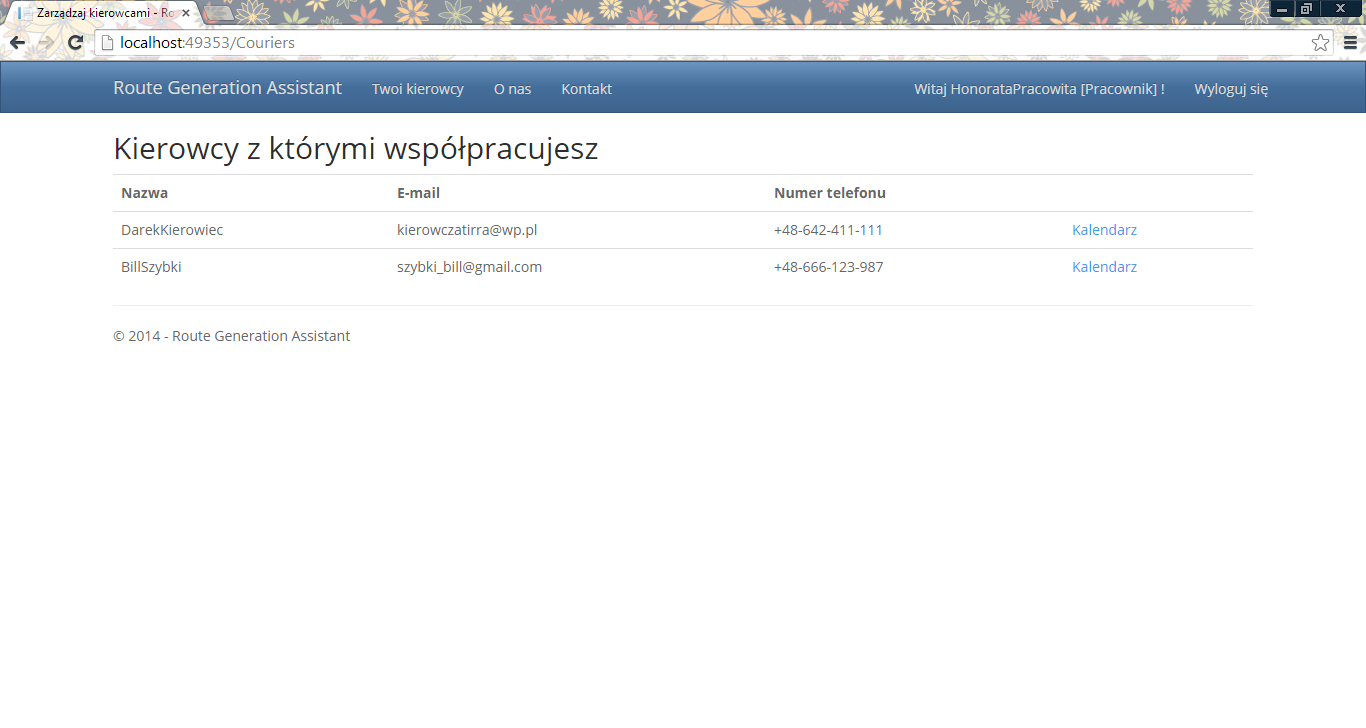




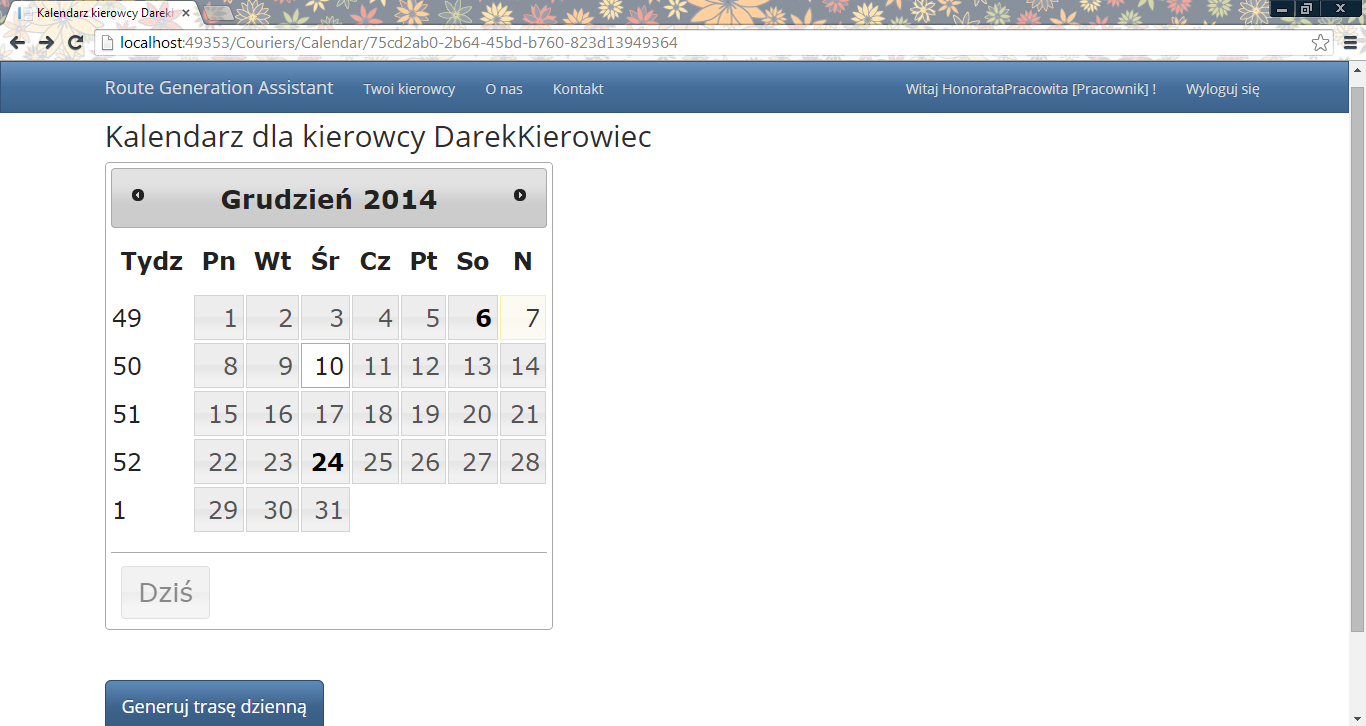
Przedstawione powyżej dwa zrzuty ekranu przedstawiają w sposób tabelaryczny informacje o wygenerowanych trasach, główne dane jakie można odczytać z tabeli to:

* Data na kiedy została wygenerowana trasa
* Pracownik który wygenerował trasę
* Kierowca który realizował/będzie realizował trasę
* Stan
* Opis
* Podsumowanie – główne drogi którymi będzie podróżował kierowca
* Adres początkowy – miejsce z którego wyjeżdża kierowca
* Przesyłki – wszystkie punkty w którem musi dojechać kierowca
* Czas trwania – szacowany czas przejazdu
* Dystans – odległość do pokonania w metrach
* Dostawca macierzy długości – wybór mapy z której były pobrane dane do obliczeń
* Dostawca optymalizacji – wybrany dostawca optymalizacji
* Kryterium optymalizacji – po czasie lub odległości
* Algorytm optymalizacji – HeldKarp lub BruteForce

Inną ważną funkcjonalnością systemu jest zarządzanie pracownikami z poziomu konta Pracownika – osoba która ustala trasy dla swoich kierowców. Po zalogowaniu przechodzimy do zakładki „Twoi kierowcy” która przekierowuje nas do widoku kierowców z którymi współpracujemy:

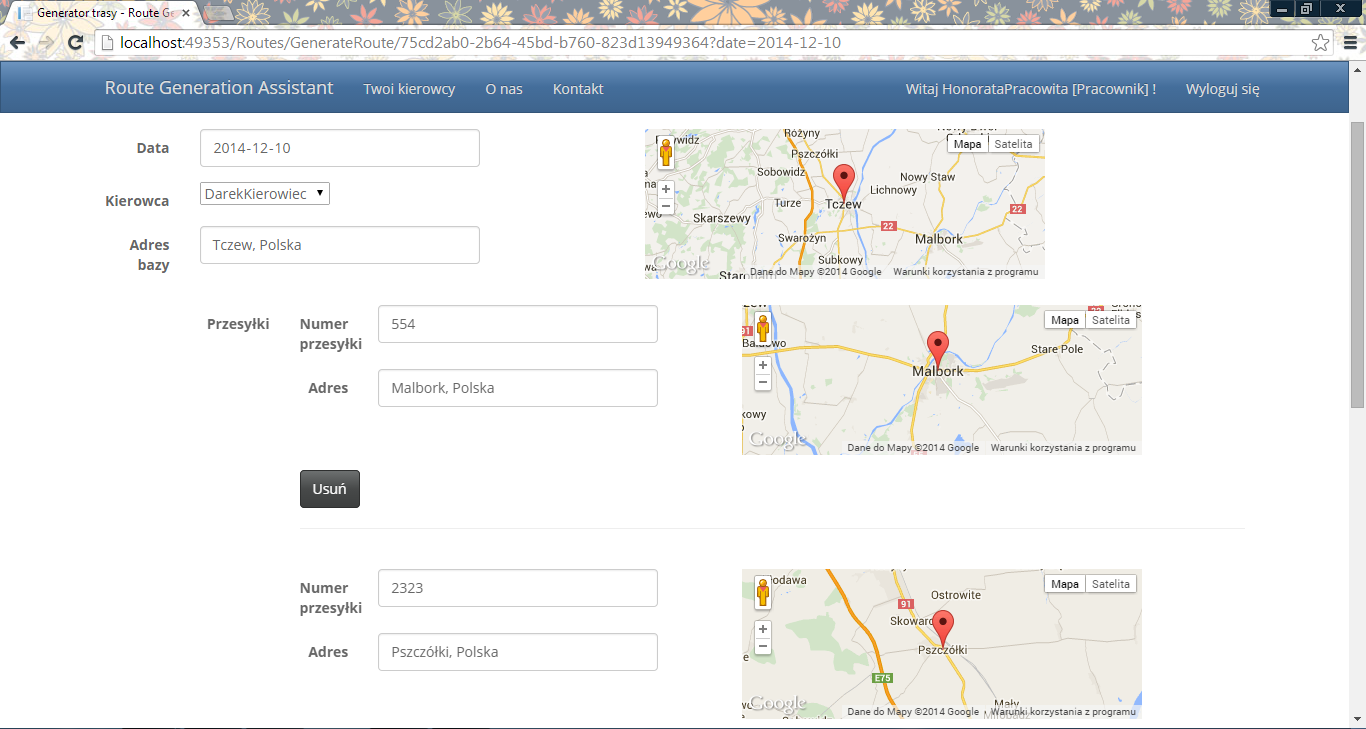


W tym oknie mamy wszystkie dane związane z kierowcą, w celu zobaczenia jego planu musimy wybrać kalendarz – jest on bardzo podobny do kalendarza kierowcy ale posiada dodatkową opcję – generowanie trasy w dni wolne:

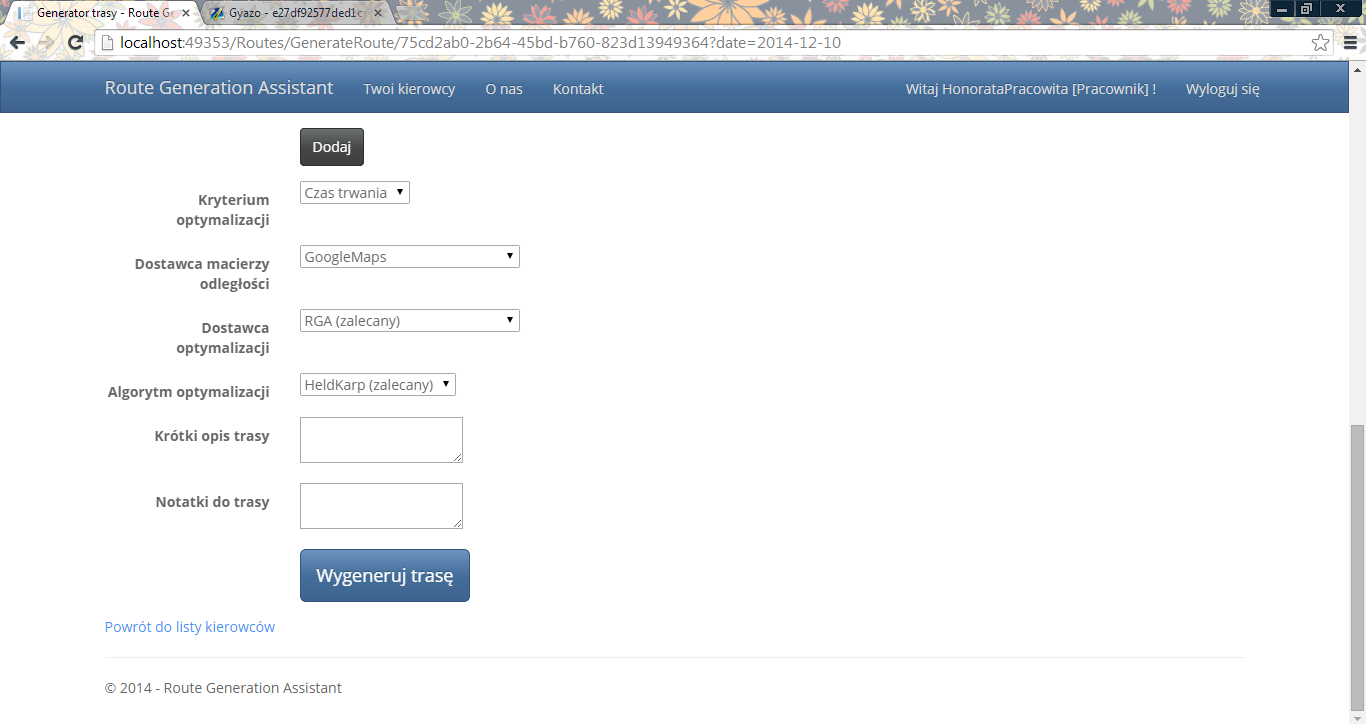


Po kliknięciu w przycisk "Generuj trasę dzienną" zostajemy przeniesieni do formularza generowania tras. Jest to jeden z ważniejszych aspektów aplikacji, ponieważ dzięki niemu możemy określać bazę jak i miejsca w która mają dotrzeć przesyłki. Budowa formularza, jak i opis podstawowych parametrów pól został przedstawiony na kolejnej stronie.

Ustawienie bazy i punktów dostarczenia przesyłek:



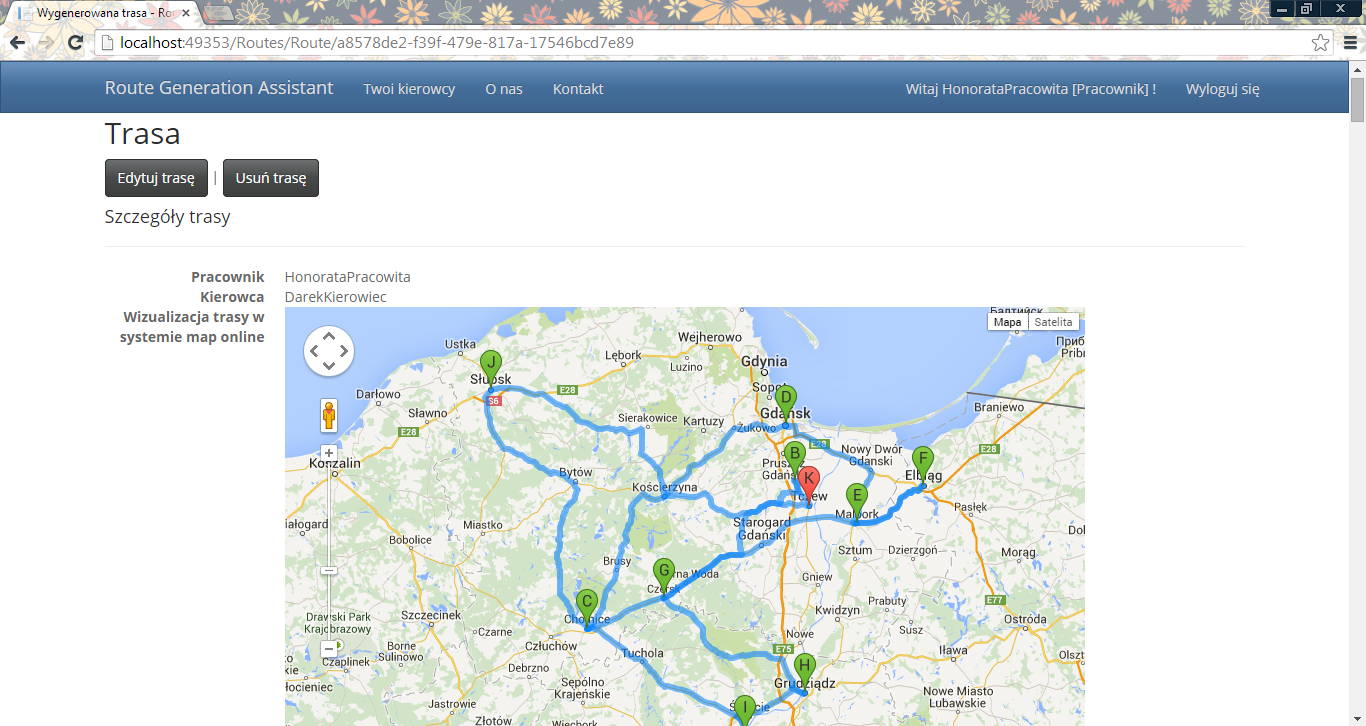
Ustawienie parametrów algorytmu:



Najważniejsze cechy powyższego formularza:

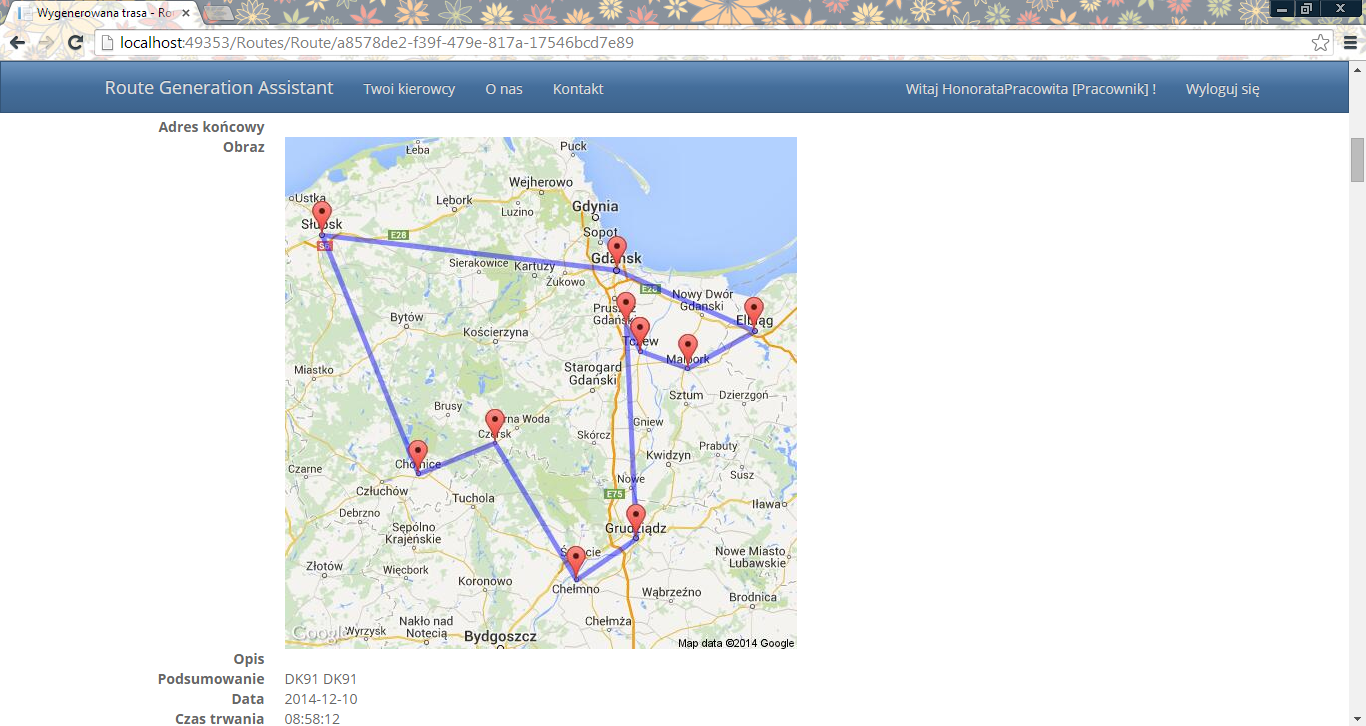
* Pola data i kierowca pozwalają zmienić wcześniej wybrane dane
* Adres bazy określa miejsce naszego magazynu
* Przesyłki identyfikowane na podstawie numeru jak i adresu. Adres pobierany z GoogleMaps API w przypadku poprawnego załadowania adresu, wizualizacja miejsca na małej mapie
* Przyciski dodaj i usuń za pomocą których można regulować ilość przesyłek
* Parametry algorytmu na podstawie których zostanie wygenerowana trasa
* Przycisk „Wygeneruj trasę” który generuje trasę i przenosi nas do widoku trasy.

Widok wygenerowanej trasy prezentuje się następująco :



Mapa przedstawia wygenerowaną trasę dla pewnej ilości punktów. Mamy możliwość edycji trasy, oraz usunięcia wygenerowanej trasy. Kolejne punkty przejazdu są oznaczane kolejnymi literami alfabetu. Punkt startowy A zawsze się pokrywa z ostatnim punktem na mapie w tym przypadku K (ponieważ musimy wrócić do bazy, co jest raczej oczywistym powodem).

Widok wszystkich punktów na mapie, wygenerowanych w postaci obrazka:



Widok ten nie zawiera trasy przejazdu, ale wyświetla wszystkie punkty na mapie, spinając je w wielobok. Wielobok ten obrazuje dosyć dobrze optymalną kolejność odwiedzania kolejnych adresów punktów doręczenia przesyłek i jest uboższą alternatywą wizualizacji trasy dla tych użytkowników którzy mają wyłączony JavaScript w przeglądarce. Więc alternatywnym rozwiązaniem rozwozu przesyłek mogło by być dostarczenie ich według obwodu wieloboku – tak jak przedstawiono to na rysunku powyżej.

# Dokumentacja procesowa

Poniższy rozdział przedstawia wszystkie procesy wykonane w ramach powstawania owego systemu. W tym rozdziale przedstawiono dokładny harmonogram prac wykonanych podczas wytwarzania oprogramowania oraz wszystkich pokrewnych produktów. Przedstawiono infrastrukturę projektu, potencjalnych użytkowników oraz przykładowe scenariusze użycia.

## Product backlog i opis infrastruktury projektu

### Cel Systemu

System ma za zadanie wspierać kierowców firmy logistycznej w generowaniu optymalnych tras przejazdu na podstawie zadanych kryteriów: czas, koszt, odległość. Dzięki naszemu rozwiązaniu, nasz klient będzie mógł zmniejszyć koszty zarządzania flotą pojazdów i zoptymalizować pracę swoich kierowców.

### Kontekst systemu

**Udziałowcy:**

* **Administrator** – jego rolą będzie zarządzanie systemem, oraz odpowiedną konfiguracją użytkowników. Główne zadania na tym stanowisku to zarządzanie użytkownikami – nadawanie im konkretnych uprawnień, dodawanie nowych kierowców. System może posiadać więcej niż jednego administratora.
* **Kierowca** – główny użytkownik naszego systemu. Może przeglądać trasy przypisane mu przez pracownika na konkretne dni. Ma również opcję wydrukowania trasy którą ma zrealizować konkretnego dnia.
* **Pracownik** – osoba która zarządza kierowcami, wyznacza im trasy na poszczególne dni. Może wygenerować kierowcy trasę dzienną na podstawie punktów w które musi dostarczyć przesyłki. Może dodać odpowiednią notatkę dla kierowcy do trasy.

### Wymagania funkcjonalne

Opracowywany system będzie używany w sposób ciągły przez różnych użytkowników. Odbiorca chce wydzielenia trzech grup użytkowników – kierowników, pracowników oraz kierowców.

Kierownicy to grupa użytkowników systemu zajmująca się zarządzaniem kontami użytkowników systemu. Jako jedyni mają oni uprawnienia tworzenia, modyfikacji czy też usuwania użytkowników. Ze względu na politykę bezpieczeństwa kierownicy powinni mieć również możliwość czasowego blokowania kont użytkowników. Kierownicy posiadają również uprawnienia przydzielania kurierów do pracowników, co pozwala na zorganizowanie pracy i ograniczenie uprawnień dostępu pracowników do informacji. Dzięki temu pracownicy mogą generować i modyfikować trasy dzienne tylko dla tych kierowców z którymi współpracują.

Pracownicy firmy mają mieć możliwość generowania tras dziennych dla kierowców. Generowanie ma się odbywać w prosty sposób, poprzez podanie punktów dostarczania przesyłek w postaci listy adresów. System ma generować trasy i zapisywać je w bazie danych. Generowane trasy mają być najkrótsze możliwe (optymalne), nie powinny prowadzić przez drogi płatne. System powinien pozwalać na wyświetlenie informacji o planowanych przejazdach dla danego kierowcy w formie kalendarza, w którym odpowiednio zaznaczone będą dni już rozplanowane jak również dni wolne. Pracownicy powinni mieć możliwość wyświetlenia listy kierowców z którymi współpracują. System ma również dostarczać możliwość modyfikacji jak też usuwania tras. Dla tras gotowych system powinien pozwalać na generowanie podglądu trasy w systemie map online google maps.

Kierowcy powinni mieć możliwość wyświetlania kalendarza, zawierającego zapisane w systemie kursy które ich dotyczą. Po wybraniu konkretnego dnia, kierowca powinien otrzymać informacje o planowanej trasie przejazdu, składającej się z adresów punktów doręczeń przesyłek, opisu trasy oraz notatek (które może dodawać zarówno pracownik jak i kierowca). Z tego samego miejsca kierowca powinien mieć możliwość wygenerowania podglądu trasy w systemie map online, tak jak pracownik. System, na żądanie kierowcy, powinien wyświetlić informacje o trasie dziennej w formie gotowej do wydruku.

System powinien dostarczać mechanizmu komunikacji wewnętrznej (wiadomości prywatne), pozwalającego na wymianę informacji pomiędzy użytkownikami. Aplikacja powinna pozwalać na dodawanie notatek do gotowych tras dziennych, zarówno przez kierowcę jak i pracownika. Powinna również istnieć możliwość modyfikowania i usuwania notatek przez ich autorów. System powinien rozróżniać kilka statusów tras (np. nowa, ukończona, w trakcie), i na tej podstawie odpowiednio wizualizować trasę w kalendarzu. System powinien operować na podstawie danych geolokacyjnych dostarczanych przez google maps. Aplikacja ma posiadać interfejs użytkownika w języku polskim.

### Wymagania poza funkcjonalne

* + System powinien mieć interfejs użytkownika wyświetlający się poprawnie w rozdzielczości 1366x768 oraz wyższych.
  + Generowanie pojedynczej trasy nie powinno trwać więcej niż 5 minut. System powinien informować użytkownika na bieżąco o postępie procesu generowania, jeżeli proces ten nie odbywa się natychmiastowo.
  + System powinien na bieżąco walidować dane wprowadzane przez użytkownika do formularzy. Nie powinien pozwolić na wysłanie formularza z niepoprawnymi danymi.
  + Komunikaty błędów generowane przez system powinny być jasne i zrozumiałe, sugerować metodę rozwiązania problemu.
  + Zaprojektowany interfejs powinien być utrzymywany w jasnej kolorystyce i nie rozpraszać użytkownika w pracy.

### Scenariusze użycia

* Wygenerowanie trasy dziennej (Scenariusz I)

Pracownik Paweł chce wygenerować trasę dzienną dla kierowcy Daniela na poniedziałek 20.12.2014 r. W tym celu otwiera stronę główną systemu poprzez przeglądarkę internetową i uwierzytelnia się. Wybiera z menu systemu opcję <Kierowcy> co przenosi go do listy kierowców z którymi współpracuje. W następnym kroku wybiera on opcję <Kalendarz> znajdującej się obok imienia kierowcy. Przenosi to go do widoku kalendarza na miesiąc bieżący. Paweł, za pomocą kontrolek, wybiera interesujący go miesiąc. W polu kalendarza odpowiadającym dacie 20.12 klika przycisk <Dodaj>. Przenosi go to do strony zawierającej formularz definiujący trasę. Paweł wprowadza numery przesyłek, miejsca docelowe oraz

krótki opis trasy. Następnie wybiera on przycisk <Dodaj> i jest informowany przez system o  postępie

generowania trasy, a w końcu o jej poprawnym utworzeniu i zapisaniu. Po potwierdzeniu Paweł jest przenoszony do strony zawierającej opis wygenerowanej trasy.

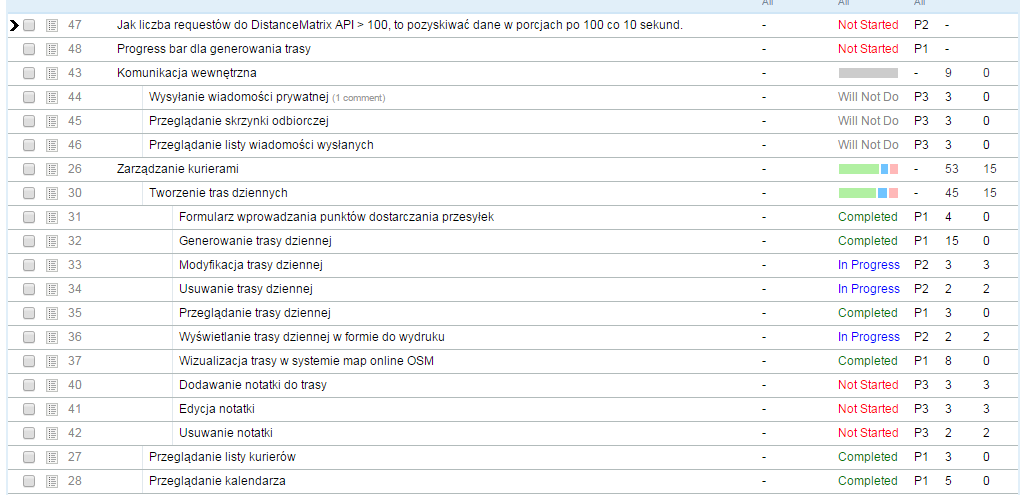
* Wydrukowanie trasy dziennej (Scenariusz II)

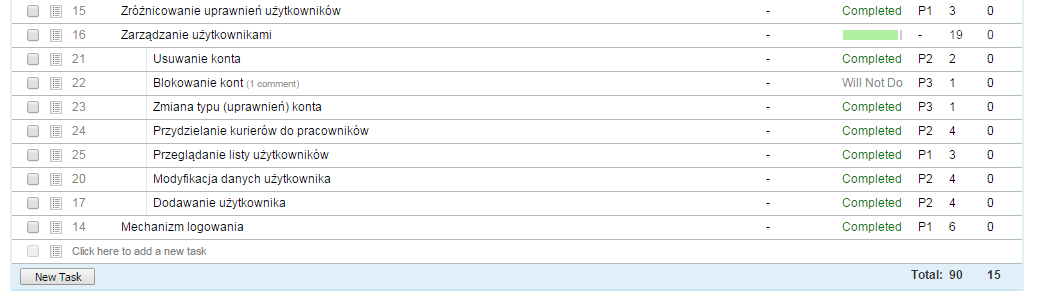
Andrzej, kierowca, chce wydrukować dzisiejszą trasę dzienną. W tym celu otwiera przeglądarkę www, odwiedza stronę główną systemu i uwierzytelnia się na niej. Z menu systemu wybiera opcję <Kalendarz> która przenosi go do widoku obecnego miesiąca. Następnie wybiera on pole odpowiadające dni bieżącemu. Andrzej jest przenoszony do widoku trasy dziennej, zawierającego takie informacje jak opis trasy, numery identyfikacyjne przesyłek oraz punkty ich dostarczenia. Widzi tam również notatki dodane wcześniej przez siebie. Andrzej zmienia status zlecenia na <W toku>, po czym klika przycisk <Drukuj>, co powoduje przeniesienie go do strony zawierającej identyczne informacje w wersji gotowej do wydruku. Jednocześnie na ekranie pojawia się dialog drukowania.

* Dodanie nowego kuriera (Scenariusz III)

Kierownik Jacek chce dodać nowo zatrudnionego kuriera do systemu. W tym celu loguje się do systemu poprzez przeglądarkę internetową. Wybiera następnie opcję <Kurierzy> z menu systemu, co przenosi go do listy kurierów. Jacek ignoruje wyświetlaną listę i wybiera przycisk <Dodaj>. System wyświetla formularz zawierający pola pozwalające na wprowadzenie danych osobowych kurierów. Jacek wprowadza je i wybiera opcję <Dodaj>. System informuje o pomyślnym dodaniu nowego użytkownika do systemu i zwraca uwagę na konieczność przydzielenia kierowcy jednemu z pracowników . Jacek podążając za sugestią systemu wybiera opcję <Pracownicy>. Wybiera następnie opcję <Kurierzy> widoczną przy imieniu pracownika Radek. Przenosi go to do listy kierowców współpracujących z Radkiem. Jacek wybiera opcję <Dodaj> co powoduje zmianę ekranu na listę zawierającą imiona i nazwiska kurierów nie przydzielonych jeszcze żadnemu pracownikowi. Jacek wybiera imię nowo zatrudnionego kuriera z listy i akceptuje wybór. Jest on ostatecznie przekierowany do listy kurierów przydzielonych do Radka, gdzie może zobaczyć, że nowy kierowca został poprawnie dodany do systemu.

### Product Backlog

****

****

### Infrastruktura projektu

#### Organizacja zespołu projektu

mgr inż. Tomasz Gawron – właściciel produktu, mistrz Scrum

Paweł Troka – developer

Łukasz Adrych – developer

#### Komunikacja w zespole

Środki komunikacji:

* E-mail
* Skype
* Facebook

Spotkania:

* Konsultacje z opiekunem projektu – Politechnika Gdańska EA 624, wtorki 17:15
* Codzienny Scrum – codziennie 10 minut, o różnych porach jeżeli jesteśmy za zajęciach w tym samym czasie, miejscem spotkania jest Politechnika Gdańska. W przypadku dni wolnych, lub dni bez wspólnych zajęć spotkanie odbywa się na skype, lubi pisemnie na facebook’u.

#### Dokumentacja

Dokumentacja zostanie wytworzona za pomocą programu Microsoft Word 2010. Product backlog jak i sprint backlog znajdują się na stronie pod adresem: (<http://rga.acunote.com/projects>). Oprócz dokumentacji powstanie również plik infrastruktury projektu.

Dokumenty powstające przy każdym sprincie:

* Sprint x Backlog (gotowy na początku sprintu)
* Sprint x wydanie (gotowe na końcu sprintu)

#### Współdzielenie dokumentów i kodu

* system kontroli wersji Git
* projekt przechowywany na codeplex pod adresem (<https://rga.codeplex.com>)
* współdzielenie dokumentacji oraz innych plików związanych z projektem na DropBox’ie

#### Narzędzia

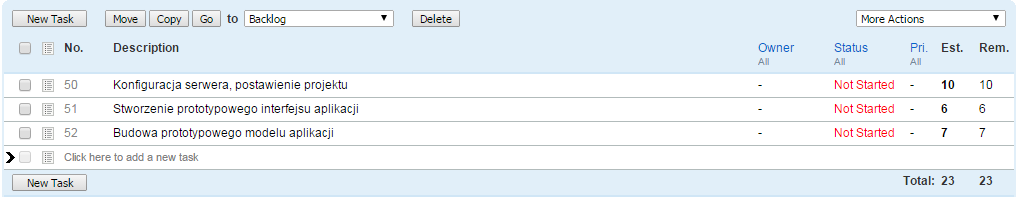
* Środowisko programistyczne – Visual Studio 2013
* Testowanie systemu
  + Testy jednostkowe
  + Testy funkcjonalne
* Komunikatory
  + Facebook, Skype
* Prowadzenie backlog’a
  + Acunote
* Wersjonowanie kodu i dokumentów
  + Wykorzystanie Git’a
* Współdzielenie materiałów dydaktycznych
  + Za pomocą DropBox’a na prywatnych kontach
* Wykorzystane biblioteki (najbardziej istotne dla projektu zostały pogrubione)
  + Bing APIs (Portable)
  + angularjs.TypeScript.DefinitelyTyped (wersja 2.2.3)
  + Antlr (wersja 3.5.0.2)
  + Bing.RestClient (wersja 0.8.0)
  + **bootstrap** (wersja 3.3.1)
  + **EntityFramework** (wersja 6.1.1)
  + **gmaps-api-net** (wersja 0.12.115)
  + google.maps.d.ts (wersja 1.0.10)
  + **GoogleMapsApi** (wersja 0.40.0.0)
  + **jQuery** (wersja 2.1.1)
  + jquery.blockUI.TypeScript.DefinitelyTyped (wersja 0.0.3)
  + jquery.TypeScript.DefinitelyTyped (wersja 1.4.0)
  + **jQuery.UI.Combined** (wersja 1.11.2)
  + jquery.ui.datetimepicker.TypeScript.DefinitelyTyped (wersja 0.0.6)
  + jquery.ui.layout.TypeScript.DefinitelyTyped (wersja 0.0.7)
  + **jQuery.Validation** (wersja 1.13.1)
  + jqueryui.TypeScript.DefinitelyTyped (wersja 0.3.5)
  + Microsoft.AspNet.Identity.Core (wersja 2.1.0)
  + Microsoft.AspNet.Identity.EntityFramework (wersja 2.1.0)
  + Microsoft.AspNet.Identity.Owin (wersja 2.1.0)
  + **Microsoft.AspNet.Mvc** (wersja 5.2.2)
  + **Microsoft.AspNet.Razor** (wersja 3.2.2)
  + Microsoft.AspNet.Web.Optimization (wersja 1.1.3)
  + Microsoft.AspNet.WebPages (wersja 3.2.2)
  + Microsoft.Bcl (wersja 1.1.9)
  + Microsoft.Bcl.Async (wersja 1.0.168)
  + Microsoft.Bcl.Build (wersja 1.0.14)
  + **Microsoft.jQuery.Unobtrusive.Ajax** (wersja 3.2.2)
  + Microsoft.jQuery.Unobtrusive.Validation (wersja 3.2.2)
  + Microsoft.Net.Http (wersja 2.2.22)
  + Microsoft.Owin (wersja 3.0.0)
  + Microsoft.Owin.Host.SystemWeb (wersja 3.0.0)
  + Microsoft.Owin.Security (wersja 3.0.0)
  + Microsoft.Owin.Security.Cookies (wersja 3.0.0)
  + Microsoft.Owin.Security.Facebook (wersja 3.0.0)
  + Microsoft.Owin.Security.Google (wersja 3.0.0)
  + Microsoft.Owin.Security.MicrosoftAccount (wersja 3.0.0)
  + Microsoft.Owin.Security.OAuth (wersja 3.0.0)
  + Microsoft.Owin.Security.Twitter (wersja 3.0.0)
  + Microsoft.Web.Infrastructure (wersja 1.0.0.0)
  + Modernizr (wersja 2.8.3)
  + **Newtonsoft.Json** (wersja 6.0.6)
  + OsmSharp (wersja 4.2.0.723)
  + Owin (wersja 1.0)
  + PortableRest (wersja 3.0.0)
  + protobuf-net (wersja 2.0.0.668)
  + Respond (wersja 1.4.2)
  + WebGrease (wersja 1.6.0)
  + Zlib.Portable (wersja 1.9.2)

## Sprint pierwszy

* Data 13.10.2014 – 26.10.2014
* Wersja produktu – *brak*
* Skład zespołu
  + Paweł Troka
  + Łukasz Adrych
  + Daniel Mielwski
* Cel Sprintu

Celem pierwszego sprintu jest konfiguracja środowiska pracy, utworzenie projektu, odpowiednia konfiguracja narzędzi do współpracy z innymi członkami. Planowane jest powstanie prototypowego interfejsu aplikacji oraz modelu bazy danych wykorzystywanej w systemie. Pierwszy sprint jest również przeznaczony za dogłębne zapoznanie się z technologią ASP.NET MVC która będzie wykorzystywana przy budowie systemu.

* Sprint Backlog



### Funkcje i ograniczenia produktu, znane błędy

Obecna wersja systemu nie posiada żadnych funkcjonalności, projekt pozwala na zapoznanie się z technologią ASP.NET, silnikiem Razor do generowania widoków oraz EntityFramework – ORM wykorzystywanym do połączenia z bazą.

### Zmiany w stosunku do poprzedniego wydania

*Nie dotyczy*

### Wymagania na środowisko i instrukcja uruchomienia

W celu uruchomienia programu w obecnej postaci konieczne jest posiadanie środowiska programistycznego Xamarin Studio i kompilacja projektu. Wymagane jest również zainstalowanie mod\_mono w celu posiadania odpowiednich narzędzi do uruchomienia programu. Aktualny system pracuje na lokalnej bazie MySQL w celu poprawnego połączenia z bazą wymagane jest posiadanie klienta bazy MySQL na własnym komputerze. Potrzebne są również pakiety EF oraz jQuery UI które można pobrać przy pomocy NuGet Package Manager, który jest zintegrowany z IDE Xamarin Studio.

Docelowa wersja systemu będzie aplikacją webową, ogólne wymagania opisane w poprzednim punkcie będą więc dotyczyły serwera/chmury na którym docelowo zostanie umieszczona aplikacja webowa. Wymagania odnośnie użytkownika będą wymagały posiadania aktualnej przeglądarki internetowej dowolnego dostawcy.

### Instrukcja użytkowania

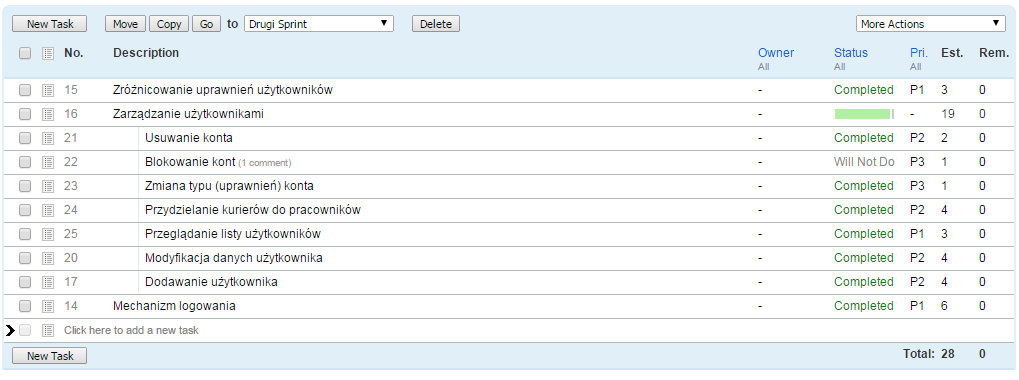
W celu skorzystania z obecnej wersji systemu, wystarczy skompilować projekt. Obecny system posiada jedynie prototypowe widoki bez żadnej funkcjonalności.

## Drugi Sprint

* Data 27.10.2014 – 09.11.2014
* Wersja produktu – *v0.2*
* Skład zespołu
  + Paweł Troka
  + Łukasz Adrych
  + Daniel Mielwski
* Cel Sprintu

Głównym celem sprintu jest stworzenie modułów odpowiedzialnych za zarządzanie użytkownikami. Główne funkcjonalności które mają być zrealizowane w tym sprincie to logowanie, rejestracja, aktywacja konta, system uprawnień, zarządzanie użytkownikami. Planowane jest również dogłębne przetestowanie tych funkcjonalności w celu uniknięcia problemów na późniejszym etapie projektu. Inny zadaniem wykonanym w ramach tego etapu będzie stworzenie komunikacji wewnętrznej w systemie, dzięki której kierowcy i pracownicy będą mogli się bezproblemowo komunikować.

* Sprint Backlog



### Funkcje i ograniczenia produktu, znane błędy

Funkcje:

* Rejestracja użytkownika w systemie
* Zalogowanie użytkownika do systemu
* Utworzenie ról dla użytkowników, dostęp do widoków na podstawie posiadanych praw
* Wyświetlanie listy użytkowników z poziomu panelu administratora
* Usuwanie użytkownika z poziomu panelu administratora
* Zmiana uprawnień użytkownika z poziomu panelu administratora

Błędy i ograniczenia:

* Bugi podczas logowania
* Błąd podczas zmiany hasła w systemie

### Zmiany w stosunku do poprzedniego wydania

Do systemu zostały dodane nowe funkcjonalności związane z użytkownikiem, doinstalowano kilka nowych pakietów wymaganych do wizualizacji wyników i interakcji z użytkownikiem.

### Wymagania na środowisko i instrukcja uruchomienia

W celu uruchomienia programu w obecnej postaci konieczne jest posiadanie środowiska programistycznego Xamarin Studio i kompilacja projektu. Wymagane jest również zainstalowanie mod\_mono w celu posiadania odpowiednich narzędzi do uruchomienia programu. Aktualny system pracuje na lokalnej bazie MySQL w celu poprawnego połączenia z bazą wymagane jest posiadanie klienta bazy MySQL na własnym komputerze. Potrzebne są również pakiety EF oraz jQueryUI które można pobrać przy pomocy NuGet Package Manager, który jest zintegrowany z IDE Xamarin Studio.

Docelowa wersja systemu będzie aplikacją webową, ogólne wymagania opisane w poprzednim punkcie będą więc dotyczyły serwera/chmury na którym docelowo zostanie umieszczona strona. Wymagania odnośnie użytkownika będą wymagały posiadania aktualnej przeglądarki internetowej dowolnego dostawcy.

### Instrukcja użytkowania

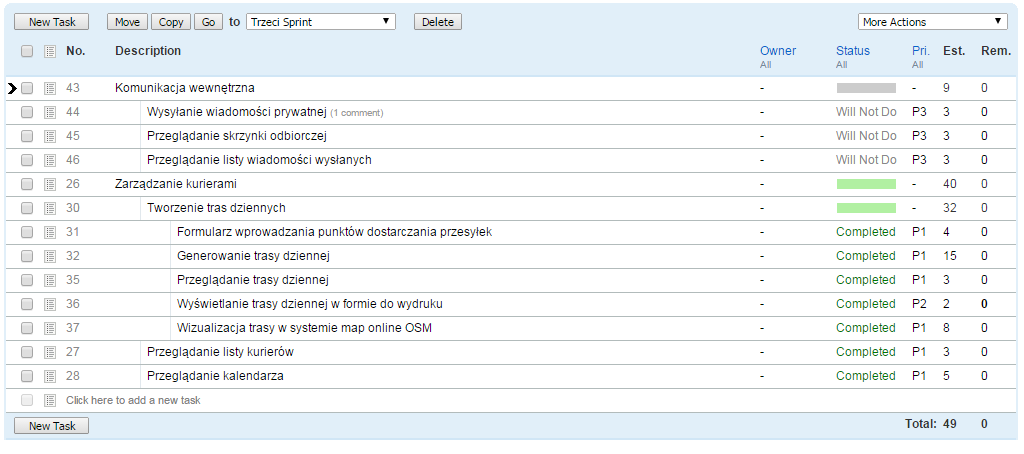
W celu przetestowania nowo wprowadzonych funkcji wymagana jest początkowa rejestracja w systemie. Konto utworzone w ten sposób nie wymaga akceptacji administratora. Po akceptacji możliwe jest zalogowanie do systemu. Logowanie jak i rejestracja wykonywana jest przez formularze do których znajdują się odnośniki w prawej górnej części serwisu. W celu przeglądania listy użytkowników należy zalogować się na konto administratora. Z poziomu panelu administratora mamy dostęp do funkcji związanych z zarządzaniem użytkownikami. W celu usunięcia użytkownika z naszego systemu należy kliknąć <Usuń> przy odpowiedniej osobie – dana osoba jest nieodwracalnie usunięta z systemu. Mamy również możliwość edytowanie użytkowników zarejestrowanych w systemie. Aby edytować dane o pracowniku, należy kliknąć przycisk <Edytuj> i zostaniemy przeniesieni do formularza do edycji wybranego użytkownika. Formularz pozwala na zmianę danych konta, takich jak telefon, adres oraz zmianę roli przypisanej aktualnie do wybranego użytkownika systemu.

## Trzeci Sprint

* Data 10.11.2014 – 23.11.2014
* Wersja produktu – *v0.3, v.0.4*
* Skład zespołu
  + Paweł Troka
  + Łukasz Adrych
* Cel Sprintu

Główne zadania do wykonania w tym etapie, to utworzenie systemu generowania tras oraz ich wizualizacji. Zadania do pod wykonania w tym etapie to utworzenie formularza do wprowadzania punktów przesyłek jak i wprowadzania punktu startowego. Powstanie prototypowego kalendarza który będzie przechowywał trasy wygenerowane dla poszczególnych kierowców. Usunięcie usterek powstałych w poprzednim sprincie oraz wykonanie komunikacji wewnętrznej która nie została zrealizowana w drugim etapie. Głównym celem etapu jest zaprezentowanie w „pełni” funkcjonalnego prototypu naszemu odbiorcy, którym będzie mógł przetestować generowanie tras według zaimplementowanych algorytmów.

* Sprint Backlog



### Funkcje i ograniczenia produktu, znane błędy

Funkcje:

* Tworzenie nowego użytkownika z poziomu panelu administracyjnego.
* Dodanie opcji przypisywania kierowców do konkretnego pracownika.
* Dodanie nowych uprawnień dla administratora/kierownika systemu.
* Dodano prototyp kalendarza dla każdego kierowcy.
* Definiowanie trasy dla konkretnego dnia tygodnia.
* Utworzono formularz wprowadzania punktów dostarczenia przesyłek, wraz z wizualizacją punktu. ma mapie Google.
* Dodawanie nowego punktu do wprowadzania przesyłek.
* Dodano możliwość wyboru dostawcy macierzy odległości punktów na mapie.
* Możliwość wygenerowania trasy dziennej dla kierowcy.

Błędy i ograniczenia:

* Brak możliwości zapisania trasy.
* Ograniczenie do 8 punktów przy generowaniu trasy narzucone przez Google Maps.
* Błędy związane z kalendarzem, możliwość generowania tras dla wcześniejszych dni.
* Optymalizacja jedynie po czasie przejazdu, brak uwzględniania innych kryteriów.
* Brak funkcjonalności dodania notatki do trasy, pomimo istnienia funkcji w interfejsie.

### Zmiany w stosunku do poprzedniego wydania

W stosunku do poprzedniego etapu zmienił się skład zespołu projektowego. Z współpracy zrezygnował kolega Daniel tłumacząc to faktem, że niektóre biblioteki nie należą do wolnego oprogramowania z których nie ma zamiaru korzystać. Innym aspektem jak sam określił dla którego postanowił zrezygnować był fakt, słabej komunikacji i słabego rozwoju zadania w początkowym etapie wytwarzania. Te czynniki spowodowały zmianę IDE Xamarin Studio na Visual Studio 2013, zmianę dostawcy bazy danych na MS Server oraz pełne wykorzystanie narzędzi Microsoftu.

Ze względu na okrojenie składu zespołu harmonogram prac jak ich zakres uległ zmianie. Zadanie związanie z komunikacją wewnętrzną w systemie(wysyłanie, przeglądanie, usuwanie, zarządzanie wiadomościami) zostało wyrzucone z Product Backlog ponieważ nie są one głównym celem działania systemu. Postanowiliśmy skupić się na głównej funkcjonalności systemu, to znaczy optymalnym generowaniu tras dziennych dla pracowników firmy logistycznej.

W systemie dodano nowe funkcjonalności, główna z nich to prototyp kalendarza dzięki któremu możemy tworzyć trasę dla konkretnego dnia. W ramach tego etapu wypuszczono dwie wersje produktu, które charakteryzowały się:

* *v0.3*

Możliwość generowania tras dziennych – jeden dostawca macierzy odległości (Google Maps). Dwóch dostawców optymalizacji trasy (RGA – nasz własny i GoogleMaps), jeden algorytm optymalizacji HeldKarp.

* *v0.4*

Dodano nowego dostawcę macierzy odległości – MapQuest oraz optymalizację udostępnianą przez MapQuest API. Obecna wersja systemu pozwala na wybór dwóch parametrów przy optymalizacji trasy, jednak zalecane jest korzystanie z algorytmu optymalizacji **RGA**.

### Wymagania na środowisko i instrukcja uruchomienia

W celu uruchomienia programu w obecnej postaci konieczne jest posiadanie środowiska programistycznego Visual Studio 2013 i kompilacja projektu. Wymagane jest również zainstalowanie .NET Framework w wersji 4.5 lub wyższej aby posiadać odpowiednich narzędzia do uruchomienia programu. Aktualny system pracuje na bazie danych MS SQL Server która jest wbudowana w Visual Studio 2013 Potrzebne są również pakiety EF oraz jQueryUI które można pobrać przy pomocy NuGet Package Manager, który jest domyślnie zainstalowany w używanym IDE.

Docelowa wersja systemu będzie aplikacją webową, ogólne wymagania opisane w poprzednim punkcie będą więc dotyczyły serwera/chmury na którym docelowo zostanie umieszczona aplikacja. Wymagania odnośnie użytkownika będą wymagały posiadania aktualnej przeglądarki internetowej dowolnego dostawcy.

### Instrukcja użytkowania

Obecne wersje systemu zostały wyposażone w dużo nowych funkcjonalności, poniżej krótka instrukcja użytkowania tych funkcji. W celu dodania nowego kierowcy do systemu należy posiadać uprawnienia administratora. Po popranym zalogowaniu do systemu zostaniemy przeniesieni na stronę główną serwisu. Należy wybrać opcję na górnej belce „Zarządzaj użytkownikami” i kliknąć. Zostaniemy przekierowani do strony z wszystkimi użytkownikami serwisu. Na samej górze otwartej podstrony wybieramy przycisk „Dodaj Nowego” który przenosi nas do formularza dodawania nowego użytkownika do systemu. Jeżeli wszystkie pola zostanę wypełnione poprawnie nowy użytkownik zostanie dodany do systemu.

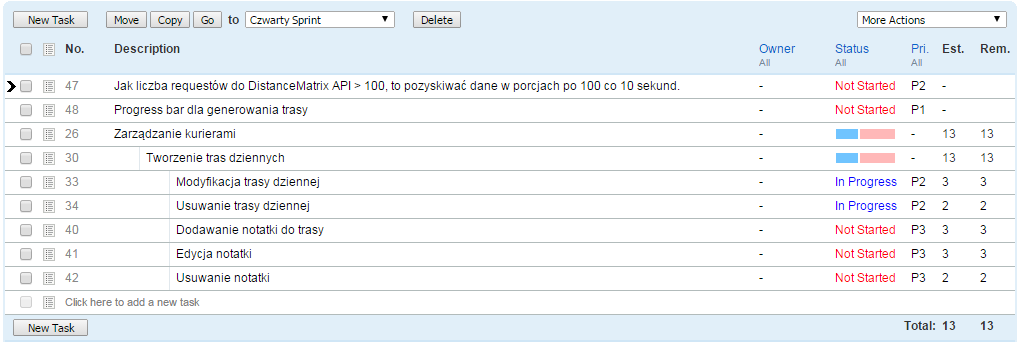
Drugą bardzo ważną funkcjonalnością daną do systemu jest tworzenie jak i przeglądanie tras dziennych w obecnej wersji systemu. W celu wygenerowania trasy należy przejść przez szereg pobocznych funkcjonalności, począwszy od zalogowania się na konto i posiadania uprawnienia „Pracownik”. Po poprawnym uwierzytelnianiu znajdujemy się na stronie początkowej serwisu, należy wybrać opcję „Twoi Kierowcy” która znajduje się na górnej belce systemu. Zostaniemy przekierowani do listy kierowców przypisanych do zalogowanego obecnie pracownika. Do każdego kierowcy przypisany jest kalendarz do którego możemy przejść klikając w „Kalendarz” znajdujący się przy wybranym użytkowniku na liście. Obecna strona wyświetla kalendarz dla wybranego kierowcy, klikający w wybrany dzień możemy przejść do opcji generowania trasy – klikamy przycisk „Generuj trasę dzienną” znajdujący się pod kalendarzem – zostajemy przekierowanie do podstrony gdzie będziemy definiować dane dla nowej trasy. Pierwsze dwa pola wyborów w opcjach generowania trasy pozwalają na zmianę wcześniej zadeklarowanych danych to znaczy daty trasy oraz kierowcy dla którego będzie generowana trasa. Pierwszym ważnym polem formularza jest definicja adresu bazy – z tego punktu rozpoczynamy generowanie trasy i do tego punktu musimy wrócić. Następnie definiujemy przesyłki – są to miejsca w które trzeba dotrzeć opisane za pomocą adresu jak i numeru przesyłki. Po zdefiniowaniu docelowych punktów należy wybrać kryteria działania algorytmu. Obecnie optymalizacja przebiega tylko po czasie. Musimy jeszcze określić trzy zmienne algorytmu: dostawca macierzy odległości, dostawca optymalizacji i algorytm optymalizacji, z racji na wczesną wersję systemu zalecane jest korzystanie z ustawień domyślnych. Klikamy w przycisk generuj trasę i jeżeli dane zostały zdefiniowane prawidłowo zostajemy przeniesieni do strony z wygenerowaną trasą. Na stronie mamy przedstawione szczegóły trasy, takie jak "Wizualizacja trasy w systemie map online", czas trwania przejazdu, czy dystans do pokonania.

## Czwarty Sprint

* Data 24.11.2014 – 07.12.2014
* Wersja produktu – *v0.5*
* Skład zespołu
  + Paweł Troka
  + Łukasz Adrych
* Cel Sprintu

Podstawowe zadanie na ostatni Sprint to dodanie różnych drobnych funkcjonalności, które nie zostały zrealizowane we wcześniejszych etapach. Przykładem może być dodanie optymalizacji po innych parametrach niż czas. W tym sprincie przebudowany zostanie również interfejs aplikacji, aby system wyglądał schludnie i osoby korzystające były zadowolone z rozmieszczenia poszczególnych widoków oraz ch prezentacji. Na ostatnio sprint zaplanowane jest również przeprowadzenie testów funkcjonalnych systemu. W ramach tego etapu powstanie również dokumentacja projektu, zawierająca między innymi instrukcję użytkowania systemu, jak i dogłębny opis przeprowadzonych prac w celu wytworzenie produktu. W przypadku szybkiego wykonania wyżej przedstawionych prac, planuje również zrealizować funkcjonalności nie wykonane w ramach Sprintu 3.

* Sprint Backlog



### Funkcje i ograniczenia produktu, znane błędy

Funkcje:

* Dodano wizualizację w systemie map online
* Dodano nowy algorytm generowania trasy – BruteForce (wolny, ale pozwala weryfikować poprawność)
* Wprowadzono nowe API – MapQuest jako dostawca macierzy odległości oraz optymalizacji trasy.
* Poprawiono obsługę generowania tras
* Zniesiono limity na ilość punktów przesyłek narzucane przez google maps API, teraz można generować trasy składające się maksymalnie z 25 punktów (wcześniej było 8)
* Dodano możliwość podglądu wydruku wygenerowanej trasy
* Dodano nowy widok zbiorczych tras przedstawiany kierowcom
* Dodano modyfikację trasy dziennej
* Dodano usuwanie trasy dziennej
* Stworzono prosty progres bar, który jest wyświetlany w momencie tworzenia trasy

Błędy i ograniczenia:

* Bugi podczas generowania trasy – *naprawiono*
* Poprawiono obsługę kalendarza
* Poprawka widoków dla kierowców
* Zmieniono sposób wyświetlania tras
* Poprawiono widoki dla kierowców, pracownik widzi teraz w kalendarzu kiedy jego konkretny kierowca ma już wygenerowane trasy, oraz nie może wygenerować drugiej trasy dla tego samego dnia.
* Dodano obsługę wyjątków podczas generowania trasy.

### Zmiany w stosunku do poprzedniego wydania

Wypuszczono nowy prototyp systemu, wersję nazwano v0.5 jej funkcjonalność w stosunku do poprzednich wersji została wzbogacona o obsługę nowych provider’ów do generowania map. Można teraz korzystać z dostawcy macierzy odległości, jak i dostawy optymalizacji. Dodano nowy algorytm optymalizacji BruteForce jest on znacznie wolniejszy od algorytmu HeldKarp’a, ale pozwala weryfikować poprawność algorytmu. Zmianie w stosunku do poprzedniej wersji uległ również interfejs. W poprzedniej wersji systemu większość danych tabelarycznych była prezentowana bez użycia szablonów do prezentowania treści. Wprowadzono również możliwość usuwania i edycji powstałej trasy, co wcześniej nie było możliwe.

### Wymagania na środowisko i instrukcja uruchomienia

W celu uruchomienia programu w obecnej postaci konieczne jest posiadanie środowiska programistycznego Visual Studio 2013 i kompilacja projektu. Wymagane jest również zainstalowanie .NET Framework w wersji 4.5 lub wyższej aby posiadać odpowiednich narzędzia do uruchomienia programu. Aktualny system pracuje na bazie danych MS SQL Server, która jest wbudowana w Visual Studio 2013 Potrzebne są również pakiety EF oraz jQueryUI, które można pobrać przy pomocy NuGet Package Manager, który jest domyślnie zainstalowany w używanym IDE.

Docelowa wersja systemu będzie aplikacja webowa, ogólne wymagania opisane w poprzednim punkcie będą więc dotyczyły serwera/chmury na którym docelowo zostanie umieszczona aplikacja. Wymagania odnośnie użytkownika będą wymagały posiadania aktualnej przeglądarki internetowej dowolnego dostawcy.

### Instrukcja użytkowania

W celu edycji algorytmu dla danej trasy musimy wykonać następujące czynności. Logujemy się do systemu, musimy posiadać uprawnienia pracownika i mieć przypisanego do siebie przynajmniej jednego kierowcę. Wchodzimy w zakładkę "Twoi kierowcy", która znajduje się na górnej belce systemu. Następnie przechodzimy do kalendarza wybranego kierowcy. W kalendarzu mamy wyróżnione pola, jedno z nich podświetlone na biało to aktualny dzień, możemy również zaznaczyć konkretny dzień i on zostanie również podświetlony na biało. Numery dni kalendarza podświetlone na czarno oznaczają dni dla których została wygenerowana trasa dzienna. Wybieramy „wolny” dzień i klikamy „Generuj trasę dzienną”. Gdy jesteśmy w oknie generowania trasy dziennej przechodzimy na sam dół formularza, mamy tam do wyboru pole – „Algoryrm optymalizacji” dla którego możemy wybrać opcję: HeldKarp lub BruteForce. Dodano również nowe kryterium optymalizacji – w obecnej wersji systemu trasę można optymalizować zarówno po długości jak i po czasie przejazdu sugerowanym przez wybranego dostawcę.

# Słownik pojęć i skrótów

* EF – Entity Framework – ORM wykorzystywany w systemie
* IDE - Integrated Development Environment – Zintegrowane środowisko programistyczne
* RGA – nazwa aplikacji (skrót od Route Generation Assistant)

# Bibliografia

* K. Schwaber, J. Sutherland, The Scrum Guide, Przewodnik po Scrumie: Reguły Gry, Scrum.org, 2013
* Dokumentacja API systemów map online
* <https://developers.google.com/maps/documentation/distancematrix/>
* <https://developers.google.com/maps/documentation/directions/>
* <https://developers.google.com/maps/documentation/javascript/directions>
* <http://www.mapquestapi.com/directions/>
* <http://www.codeproject.com/Articles/762581/Held-Karp-algorithm-implementation-in-Csharp>
* <http://en.wikipedia.org/wiki/Travelling_salesman_problem>
* <http://en.wikipedia.org/wiki/Brute-force_search>
* <http://stackoverflow.com/questions/18348956/how-does-google-maps´-optimizewaypoints-solve-travelling-salesmen>
* <https://plus.google.com/+ResearchatGoogle/posts/fcDYLZjHnRj>
* <https://code.google.com/p/or-tools/source/browse/trunk/examples/cpp/tsp.cc>
* <https://code.google.com/p/or-tools/source/browse/trunk/src/constraint_solver/routing.h>
* <http://stackoverflow.com/questions/19116578/drawing-roadmap-with-more-than-8-waypoints-using-google-directions-api-issue>
* <http://jsfiddle.net/67tbw/28/>
* <http://blog.stevensanderson.com/2008/12/22/editing-a-variable-length-list-of-items-in-aspnet-mvc/>
* <http://blog.stevensanderson.com/2010/01/28/validating-a-variable-length-list-aspnet-mvc-2-style/>
* <http://ivanz.com/2011/06/16/editing-variable-length-reorderable-collections-in-asp-net-mvc-part-1/>
* <http://ivanz.com/2011/06/20/editing-variable-length-reorderable-collections-in-asp-net-mvc-part-2/>
* <http://ivanz.com/2011/06/29/editing-variable-length-reorderable-collections-in-asp-net-mvc-part-3/>