Wydział Matematyki i Nauk Informacyjnych Politechniki Warszawskiej

System informacji dla komunikacji miejskiej i międzymiastowej

Bartłomiej Dach, Tymon Felski

Wersja 3.4

14 stycznia 2017

Lista zmian w dokumencie:

Data	Autor	Opis zmian	Wersja
16.10.2016	Bartłomiej Dach, Tymon Felski	Określenie wymagań projektu oraz harmonogramu prac	1.0
17.10.2016	Bartłomiej Dach, Tymon Felski	Specyfikacja architektury systemu	1.1
18.10.2016	Bartłomiej Dach, Tymon Felski	Dodanie administratora	1.2
19.10.2016	Tymon Felski	Usunięcie zduplikowanego przypadku użycia	1.3
9.11.2016	Bartłomiej Dach	Dodanie użytych bibliotek i ich licencji, instrukcji instalacji	1.4
10.11.2016	Tymon Felski	Dodanie wymagań systemowych, instrukcji uruchomienia i utrzymania	1.5
10.11.2016	Bartłomiej Dach	Dodanie diagramu sekwencji, instrukcji użycia	1.6
11.11.2016	Tymon Felski	Dodanie opisu modelu danych, scenariuszy i raportu z testów akceptacyjnych	1.7
19.11.2016	Bartłomiej Dach, Tymon Felski	Rozszerzenie specyfikacji o nowe przypadki użycia, wymagania niefunkcjonalne oraz aktualizacja harmonogramu prac i architektury rozwiązania	2.0
09.12.2016	Bartłomiej Dach, Tymon Felski	Aktualizacja tabeli bibliotek i ich licencji, wymagań systemowych, diagramu sekwencji oraz scenariuszy i raportu z testów akceptacyjnych	2.1
10.12.2016	Bartłomiej Dach, Tymon Felski	Aktualizacja instrukcji instalacji, uruchomienia, utrzymania oraz użycia	2.2
16.12.2016	Bartłomiej Dach	Rozszerzenie opisu biznesowego i wymagań funkcjonalnych (dodanie nowych przypadków użycia i user stories)	3.0
19.12.2016	Tymon Felski	Aktualizacja wymaganiań niefunkcjonalnych, harmonogramu prac i architektury rozwiązania	3.1
13.01.2017	Tymon Felski	Aktualizacja użytych bibliotek oraz diagramów sekwencji, dodanie dokumentacji Web Services	3.2
13.01.2017	Bartłomiej Dach	Aktualizacja modelu danych, dodanie instrukcji obsługi oraz testów akceptacyjnych dla aplikacji webowej	3.3
14.01.2017	Bartłomiej Dach, Tymon Felski	Aktualizacja instrukcji instalacji oraz uruchomienia uwzględniająca aplikację webową	3.4

Spis treści

1 Specyfikacja

1.1 Opis biznesowy

Niniejszy system służy do przechowywania danych o przewoźnikach i połączeniach komunikacji miejskiej oraz międzymiastowej. Dane wykorzystywane są do wyszukiwania konkretnych połączeń oraz sprzedaży biletów.

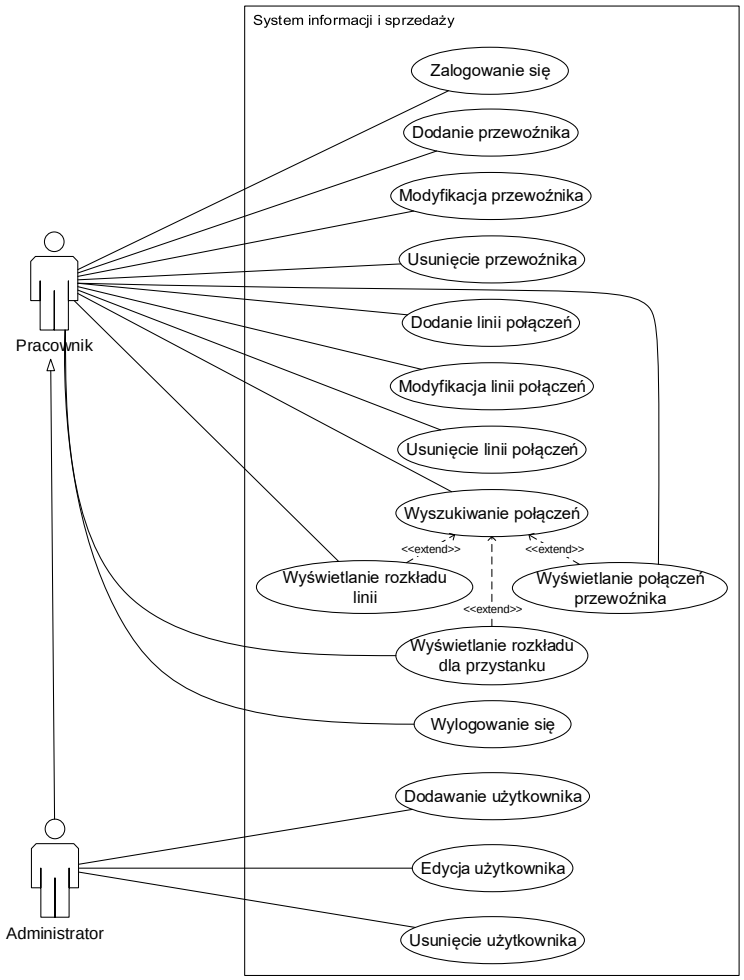
System składa się z następujących dwóch komponentów:

- komponent dla pracowników – przeznaczony dla pracowników firmy zajmującej się udostępnianiem danych, którzy za jego pośrednictwem są w stanie edytować lub dodawać nowe połączenia. Komponent ten ma charakter czysto wewnętrzny.
- komponent dla użytkowników – przeznaczony dla internautów, umożliwiający przeszukiwanie składowanych danych z poziomu przeglądarki internetowej. Za jego pośrednictwem można mieć dostęp tylko do danych publicznych.

1.2 Wymagania funkcjonalne

1.2.1 Przypadki użycia (interfejs administracyjny dla pracowników)

Poniższy diagram UML przedstawia zbiór przypadków użycia aplikacji dla aktora – pracownika firmy pośredniczącej w sprzedaży biletów wielu przewoźników.



Rysunek 1: Diagram przypadków użycia dla aplikacji

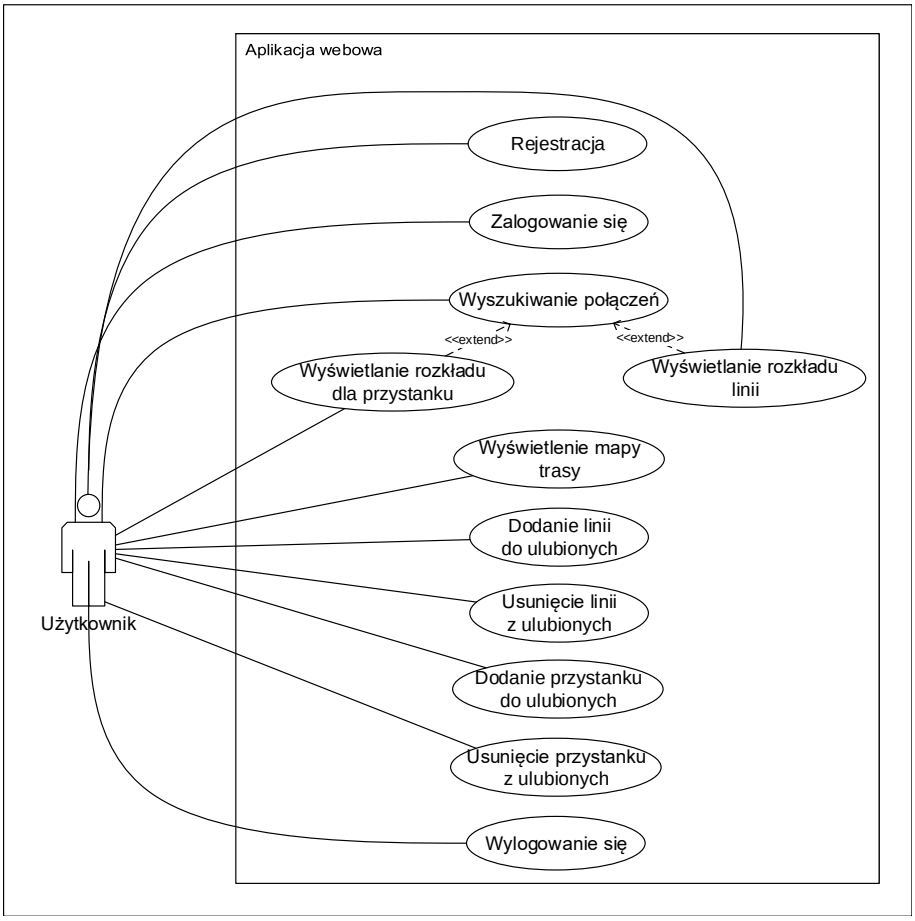
Poszczególne przypadki są opisane szerzej w poniższej tabeli:

Aktor	Nazwa	Opis	Odpowiedź systemu
Etap 1			
Administrator	Dodanie użytkownika	Dodanie nowego użytkownika do systemu	Potwierdzenie dodania użytkownika
	Modyfikacja użytkownika	Zmiana danych istniejącego użytkownika systemu	Potwierdzenie zmodyfikowania rekordu
	Usunięcie użytkownika	Usunięcie konta użytkownika i jego danych z systemu	Potwierdzenie usunięcia użytkownika
Pracownik	Zalogowanie się	Zalogowanie się użytkownika do systemu	Potwierdzenie zalogowania się lub komunikat o błędzie
	Wylogowanie się	Wylogowanie się pracownika z systemu	Potwierdzenie zakończenia pracy z systemem
	Dodanie przewoźnika	Dodanie informacji o nowym przewoźniku do bazy	Potwierdzenie dodania danych do bazy
	Modyfikacja przewoźnika	Zmiana danych przewoźnika przechowywanych w bazie	Potwierdzenie zmodyfikowania rekordu
	Usunięcie przewoźnika	Usunięcie danych przewoźnika przechowywanych w bazie	Potwierdzenie usunięcia rekordu
	Dodanie linii połączeń	Dodanie nowej linii połączeń danego przewoźnika	Potwierdzenie dodania linii do bazy
	Modyfikacja linii połączeń	Modyfikacja linii połączeń danego przewoźnika	Potwierdzenie modyfikacji rekordu
	Usunięcie linii połączeń	Usunięcie linii połączeń danego przewoźnika	Potwierdzenie usunięcia rekordu
	Wyświetlanie rozkładu linii	Wyświetlanie rozkładu jazdy wybranej linii	Widok zawierający informacje o przejazdach na wybranej linii
	Wyświetlanie połączeń przewoźnika	Wyświetlanie połączeń obsługiwanych przez danego przewoźnika	Widok zawierający informacje o liniach danej firmy
Etap 2			
Pracownik	Wyświetlanie rozkładu przystanku	Wyświetlanie rozkładu jazdy dla wybranego przystanku	Widok zawierający informacje o liniach dla wybranego przystanku
	Wyszukiwanie połączeń	Wyszukiwanie połączeń między wybranymi punktami	Widok z listą znalezionych połączeń

Tablica 3: Opisy przypadków użycia dla użytkownika

1.2.2 Przypadki użycia (interfejs webowy dla klientów)

Poniższy diagram UML przedstawia przypadki użycia dla internauty korzystającego z aplikacji webowej w celu wyszukiwania połączeń.



Rysunek 2: Diagram przypadków użycia dla użytkownika aplikacji webowej

Poszczególne przypadki są opisane szerzej w poniższej tabeli:

Aktor	Nazwa	Opis	Odpowiedź systemu
Etap 3			
Użytkownik	Rejestracja	Utworzenie konta użytkownika w systemie	Potwierdzenie utworzenia konta lub komunikat o błędzie
	Zalogowanie się	Zalogowanie się użytkownika do systemu	Potwierdzenie zalogowania się lub komunikat o błędzie
	Wylogowanie się	Wylogowanie się użytkownika z systemu	Potwierdzenie wylogowania użytkownika
	Wyszukiwanie połączeń	Wyszukiwanie połączeń między wybranymi punktami	Widok z listą znalezionych połączeń
	Wyświetlanie rozkładu linii	Wyświetlanie rozkładu jazdy wybranej linii	Widok zawierający informacje o przejazdach na wybranej linii
	Wyświetlanie rozkładu przystanku	Wyświetlanie rozkładu jazdy dla wybranego przystanku	Widok zawierający informacje o liniach dla wybranego przystanku
	Wyświetlanie mapy trasy	Wyświetlanie mapy dla wybranej trasy	Widok mapy przedstawiające kolejne przystanki na trasie
	Dodanie linii do ulubionych	Dodanie wybranej linii komunikacyjnej do listy ulubionych	Potwierdzenie dodania linii do ulubionych
	Usunięcie linii z ulubionych	Usunięcie wybranej linii z listy ulubionych	Potwierdzenie usunięcia linii z ulubionych
	Dodanie przystanku do ulubionych	Dodanie wybranego przystanku do listy ulubionych	Potwierdzenie dodania przystanku do ulubionych
	Usunięcie przystanku z ulubionych	Usunięcie wybranego przystanku z listy ulubionych	Potwierdzenie usunięcia przystanku z ulubionych

Tablica 5: Opisy przypadków użycia dla użytkownika

1.2.3 User stories

1. **Etap 1**

   1.1. Interfejs administracyjny dla administratora

   1.1.1. Jako zalogowany administrator dodaję/modyfikuję użytkownika systemu.
   Dowolny zalogowany administrator może dodać nowego użytkowanika lub zmodyfikować informacje o istniejącym użytkowniku, takie jak jego login, hasło oraz uprawnienia.

   1.1.2. Jako zalogowany administrator wyszukuję użytkownika.
   Dowolny zalogowany administrator może wyszukać istniejących użytkowników systemu.

   1.2. Interfejs administracyjny dla pracownika

   1.2.1. Jako zalogowany pracownik dodaję/modyfikuję przewoźnika.
   Dowolny zalogowany pracownik może dodać nowego przewoźnika lub zmodyfikować informacje o przewoźniku, takie, jak: nazwę i adres firmy, numer REGON oraz jej stronę internetową.

   1.2.2. Jako zalogowany pracownik dodaję/modyfikuję linię połączeń.
   Dowolny zalogowany pracownik może dodać nowe połączenie lub zmodyfikować informacje o istniejącym połączeniu takie jak: przystanki, czas odjazdu i przyjazdu na poszczególnych przystankach, ilość dostępnych miejsc w danym kursie, podstawowa cena biletu.

   1.2.3. Jako zalogowany pracownik wyszukuję połączenie.
   Dowolny zalogowany pracownik może wyszukać dostępne połączenia wybranej linii.

   1.2.4. Jako zalogowany pracownik wyświetlam rozkład jazdy danej linii.
   Dowolny zalogowany pracownik może wyszukać rozkład jazdy danej linii komunikacyjnej i go wyświetlić.

   1.2.5. Jako zalogowany pracownik wyświetlam połączenia dla danego przewoźnika.
   Dowolny zalogowany pracownik może wyświetlić połączenia od danego przewoźnika.

2. **Etap 2**

   2.1. Interfejs administracyjny dla pracownika

   2.1.1. Jako zalogowany pracownik wyświetlam rozkład jazdy dla danego przystanku.
   Dowolny zalogowany pracownik może wyświetlić rozkład jazdy dla danego przystanku.

   2.1.2. Jako zalogowany pracownik wyświetlam połączenia pomiędzy wybranymi punktami.
   Dowolny zalogowany pracownik może wyszukać dostępne bezpośrednie połączenia pomiędzy wybranymi punktami.

3. **Etap 3**

   3.1. Interfejs webowy

   3.1.1. Jako dowolny użytkownik wyświetlam rozkład jazdy danej linii.
   Dowolny użytkownik (zalogowany lub nie) może wyszukać rozkład jazdy danej linii komunikacyjnej i go wyświetlić.

   3.1.2. Jako dowolny użytkownik wyświetlam rozkład jazdy dla danego przystanku.
   Dowolny użytkownik (zalogowany lub nie) może wyświetlić rozkład jazdy dla danego przystanku.

   3.1.3. Jako dowolny użytkownik wyświetlam połączenia pomiędzy wybranymi punktami.
   Dowolny użytkownik (zalogowany lub nie) może wyszukać dostępne bezpośrednie połączenia pomiędzy wybranymi punktami.

   3.1.4. Jako niezalogowany użytkownik zakładam konto na witrynie.
   Dowolny niezalogowany użytkownik może utworzyć własne konto w aplikacji, specyfikując nazwę swojego konta oraz hasło.

   3.1.5. Jako zalogowany użytkownik dodaję linię połączeń do ulubionych/usuwam linię z ulubionych.
   Dowolny zalogowany użytkownik może oznaczać dowolną linię połączeń jako ulubioną oraz usuwać ją z listy ulubionych.

3.1.6. Jako zalogowany użytkownik dodaję przystanek do ulubionych/usuwam przystanek z ulubionych.
Dowolny zalogowany użytkownik może oznaczać dowolny przystanek jako ulubiony oraz usuwać go z listy ulubionych.

1.3 Wymagania niefunkcjonalne

Poniższa tabela zawiera rozpisane wymagania niefunkcjonalne narzucone dla systemu.

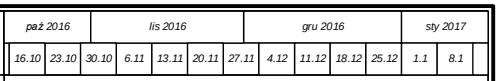
Obszar wymagań	Nr	Etap	Opis
Użyteczność (*Usability*)	1	1	Rozmiar czcionki użytej w aplikacji musi być nie mniejszy niż 12 punktów.
	2	1	Aplikacja powinna obsługiwać zmianę rozmiaru okna w sposób który umożliwia korzystanie ze wszystkich jej funkcjonalności (tzw. responsive design).
	3	2	Dane wprowadzane przez użytkownika powinny być sprawdzane pod kątem poprawności przed wysyłaniem zapytań do bazy.
Niezawodność (*Reliability*)	4	1	Aplikacja musi być odporna na dokonywanie jednoczesnych zmian tego samego rekordu bazy przez wielu pracowników jednocześnie.
	5	3	Interfejs webowy nie powinien umożliwiać modyfikowania danych w bazie (z wyjątkiem tabel dotyczących kont).
Wydajność (*Performance*)	6	1	Aplikacja powinna dodawać nowe obiekty do systemu w czasie nie dłuższym niż 1 sekundę, przy 50 żądaniach dodania obiektu na minutę.
	7	1	Zużycie pamięci RAM przez aplikację nie powinno przekroczyć 500 megabajtów.
	8	1	Wyszukiwanie połączenia między określonymi miastami powinno trwać mniej niż 2 sekundy, przy ok. 10 tys. rekordów.
Utrzymanie (*Supportability*)	9	1	Do aplikacji dołączona zostanie instrukcja wykonywania kopii zapasowej danych.

Tablica 7: Tabela wymagań niefunkcjonalnych

1.4 Harmonogram projektu

Prace przy projekcie będą realizowane według następującego harmonogramu:

ID	Nazwa zadania	Początek	Koniec	Czas trwania
1	**Etap 1**	**2016-10-15**	**2016-11-15**	**31d**
2	Analiza wymagań etapu	2016-10-15	2016-10-18	4d
3	Projekt architektury	2016-10-19	2016-10-22	4d
4	Wstępna implementacja	2016-10-23	2016-10-25	3d
5	**Właściwa implementacja**	**2016-10-26**	**2016-11-08**	**14d**
6	Utworzenie encji i serwisów	2016-10-26	2016-10-29	4d
7	Utworzenie głównego widoku	2016-10-30	2016-11-01	3d
8	Utworzenie widoków przewoźników i linii	2016-11-02	2016-11-05	4d
9	Utworzenie widoków wyszukiwania	2016-11-06	2016-11-08	3d
10	Końcowa dokumentacja, testy	2016-11-09	2016-11-12	4d
11	Poprawa błędów	2016-11-13	2016-11-14	2d
12	Zdanie etapu	2016-11-15	2016-11-15	0d
13	**Etap 2**	**2016-11-16**	**2016-12-13**	**27d**
14	Analiza wymagań etapu	2016-11-16	2016-11-19	4d
15	Adaptacja architektury projektu	2016-11-20	2016-11-22	3d
16	Wstępna implementacja	2016-11-23	2016-11-26	4d
17	**Właściwa implementacja**	**2016-11-27**	**2016-12-10**	**14d**
18	Rozproszenie aplikacji	2016-11-27	2016-11-29	3d
19	Dodanie walidacji danych wejściowych	2016-11-30	2016-12-03	4d
20	Widok rozkładu dla danego przystanku	2016-12-04	2016-12-06	3d
21	Wyszukiwanie połączeń między wybranymi punktami	2016-12-07	2016-12-10	4d
22	Końcowa dokumentacja, testy, poprawa błędów	2016-12-11	2016-12-12	2d
23	Zdanie etapu	2016-12-13	2016-12-13	0d
24	**Etap 3**	**14.12.2016**	**17.01.2017**	**34d**
25	Analiza wymagań etapu	2016-12-14	2016-12-16	3d
26	Zaprojektowanie architektury	2016-12-17	2016-12-20	4d
27	Wstępna implementacja	2016-12-21	2016-12-27	7d
28	**Właściwa implementacja**	**2016-12-28**	**13.01.2017**	**17d**
29	Projekt witryny	2016-12-28	2016-12-30	3d
30	Modyfikacja wyszukiwania połączeń	2016-12-31	2017-01-03	4d
31	Rejestracja, logowanie	2017-01-04	2017-01-06	3d
32	Mapy	2017-01-07	2017-01-10	4d
33	Ulubione przystanki i trasy	2017-01-10	2017-01-13	4d
34	Końcowa dokumentacja, testy, poprawa błędów	2017-01-14	2017-01-16	3d
35	Zdanie etapu	2017-01-17	17.01.2017	0d



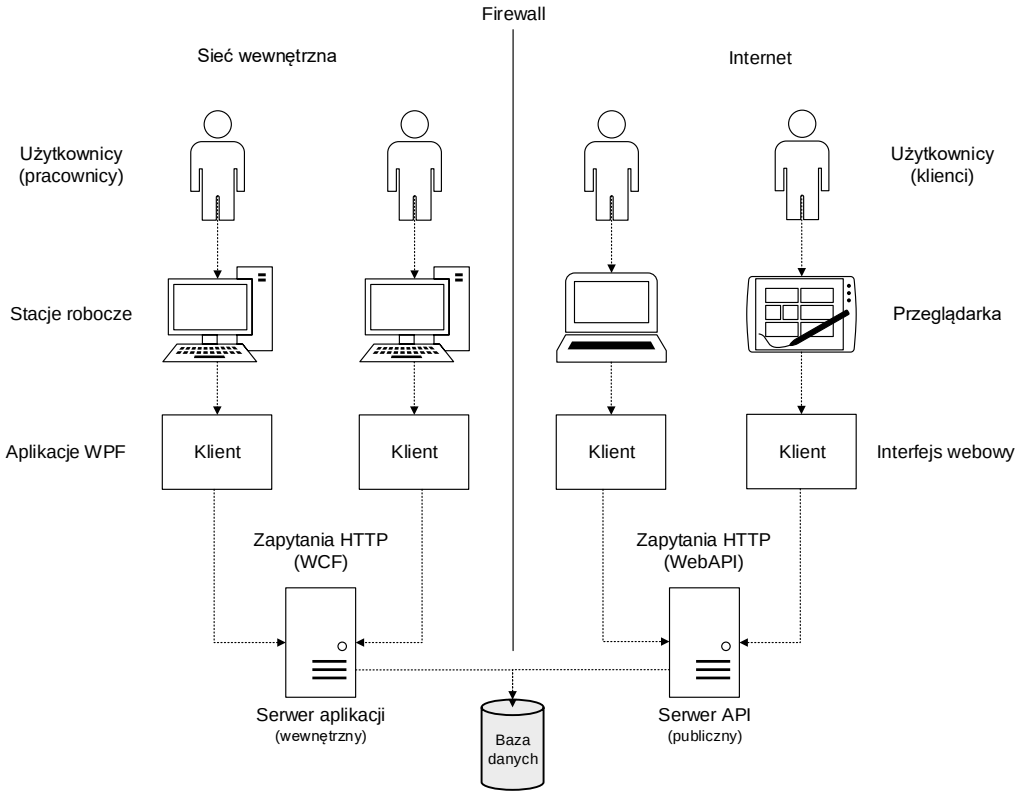
Rysunek 3: Diagram Gantta z planowanym harmonogramem projektu

Kamienie milowe:

1. **Etap pierwszy:**
   (a) 18 października: Zakończenie analizy wymagań funkcjonalnych i niefunkcjonalnych projektu.
   (b) 22 października: Zakończenie projektu architektury aplikacji, łącznie z wyróżnieniem komponentów oraz podsystemów.
   (c) 25 października: Wstępna implementacja projektu architektury, naniesienie ewentualnych poprawek do architektury wynikających z problemów implementacyjnych.
   (d) 29 października: Utworzenie encji biznesowych oraz serwisów wykorzystywanych przez użytkowników.
   (e) 1 listopada: Utworzenie głównego widoku aplikacji.
   (f) 5 listopada: Utworzenie widoków dodawania przewoźników oraz linii.
   (g) 8 listopada: Utworzenie widoków wyszukiwania połączeń oraz wyświetlania połączeń danej linii oraz przewoźnika.
   (h) 12 listopada: Zakończenie dokumentacji, testów aplikacji oraz identyfikacji błędów.
   (i) 15 listopada: Zakończenie poprawy znalezionych błędów, zdanie projektu łącznie z pełną dokumentacją.
2. **Drugi etap:**
   (a) 19 listopada: Zakończenie analizy nowych wymagań funkcjonalnych i niefunkcjonalnych projektu.
   (b) 22 listopada: Zakończenie aktualizacji architektury aplikacji, łącznie z wyróżnieniem komponentów oraz podsystemów.
   (c) 26 listopada: Wstępna próba rozproszenia aplikacji.
   (d) 29 listopada: Wydzielenie serwera aplikacyjnego oraz zintegrowanie go z aplikacją kliencką.
   (e) 3 grudnia: Zakończenie implementacji walidacji danych wejściowych.
   (f) 6 grudnia: Implementacja rozkładów dla poszczególnych przystanków oraz stworzenie odpowiedniego widoku w aplikacji klienckiej.
   (g) 10 grudnia: Dodanie funkcjonalności wyszukiwania połączeń między wybranymi przystankami.
   (h) 13 grudnia: Zakończenie poprawy znalezionych błędów, zdanie projektu łącznie z pełną dokumentacją.
3. **Trzeci etap:**
   (a) 16 grudnia: Zakończenie analizy nowych wymagań funkcjonalnych i niefunkcjonalnych projektu.
   (b) 20 grudnia: Zakończenie aktualizacji architektury aplikacji, łącznie z wyróżnieniem komponentów oraz podsystemów.
   (c) 23 grudnia: Wstępna próba wystawienia serwisów WebAPI.
   (d) 27 grudnia: Przygotowanie podstawowych widoków w interfejsie webowym.
   (e) 30 grudnia: Utworzenie projektu graficznego witryny.
   (f) 3 stycznia: Modyfikacja wyszukiwania trasy pomiędzy punktami.
   (g) 6 stycznia: Implementacja rejestracji oraz logowania na indywidualne konta użytkowników.
   (h) 10 stycznia: Dodanie wyświetlania trasy na mapie.
   (i) 13 stycznia: Implementacja funkcjonalności oznaczania tras oraz przystanków jako ulubione.
   (j) 17 stycznia: Zakończenie poprawy znalezionych błędów, zdanie projektu łącznie z pełną dokumentacją.

1.5 Architektura rozwiązania

Docelowym środowiskiem aplikacji są małe lub średnie firmy pośredniczące w sprzedaży biletów komunikacyjnych, tzn. przedsiębiorstwa zatrudniające do 250 pracowników, z czego dostęp do systemu miałby dość niski procent tej liczby (w założeniach ok. 20-30%). Dane, których przechowywanie jest niezbędne do spełnienia wymagań funkcjonalnych mają dość małą zmienność - stosunkowo rzadko ulegają zmianom lub przedawnieniom. Dodatkowo, ze względu na wewnętrzny charakter przechowywanych danych, system powinien być scentralizowany i znajdować się w jednym fizycznym położeniu.



Rysunek 4: Schemat architektury systemu

Biorąc pod uwagę opisany powyżej charakter zamówionego rozwiązania, jako część systemu przeznaczoną dla pracowników wybrany został system rozproszony składający się z aplikacji klienckich typu „gruby klient", instalowanych na stacjach roboczych pracowników oraz administratorów systemu, oraz serwera aplikacyjnego. Dla klientów indywidualnych przygotowano interfejs webowy dostępny z poziomu przeglądarki internetowej.

Planowana architektura rozwiązania ma charakter warstwowy. W aplikacji przeznaczonej dla pracowników wyróżnione zostały następujące warstwy:

- warstwa dostępu do danych - odpowiedzialna za kontakt z bazą oraz odczyt i zapis przechowywanych tam danych,
- warstwa biznesowa - odpowiedzialna za wykonywanie poszczególnych usług (np. dodania czy modyfikacji przewoźnika),
- warstwa prezentacji - odpowiedzialna za wyświetlanie interfejsu użytkownika.

Część przeznaczona dla internautów może być opisana następującymi warstawami:

- warstwa dostępu do danych - odpowiedzialna za kontakt z bazą,
- warstwa biznesowa - odpowiedzialna za wykonywanie poszczególnych usług (np. wyszukanie trasy przejazdu),

- warstwa prezentacji - odpowiedzialna za wyświetlanie interfejsu webowego.

Głównymi powodami zaproponowania architektury warstwowej były:

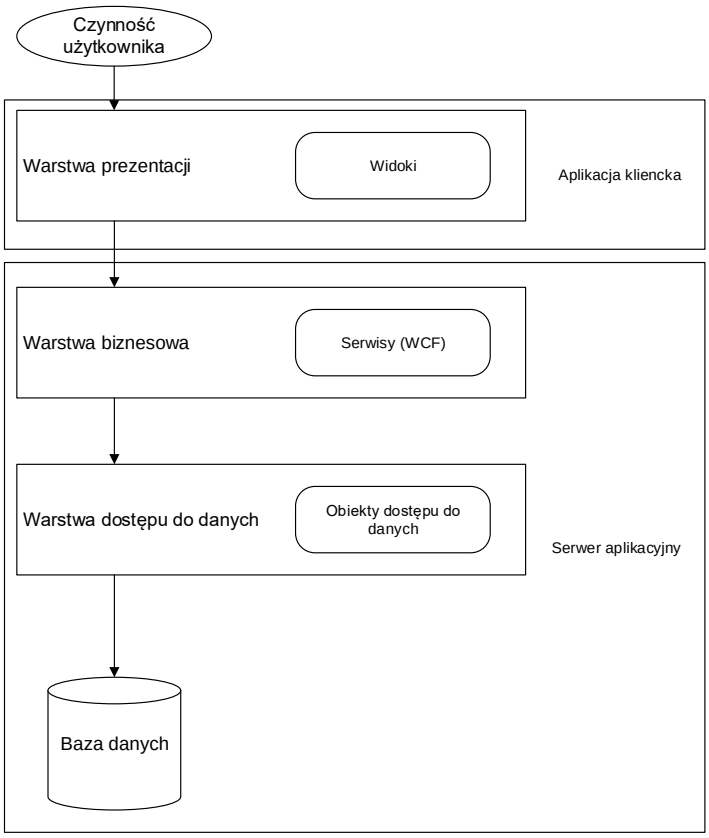
- możliwość wymiany silnika bazodanowego oraz warstwy prezentacji bez naruszania warstwy biznesowej,
- podział odpowiedzialności na poszczególne warstwy,
- spójny charakter wymagań - podział na podsystemy jest zbędny.

Ze względu na małą liczbę użytkowników niska skalowalność oraz wydajność rozwiązań warstwowych zostały uznane za ryzyko drugorzędne.

1.5.1 Serwer aplikacyjny

Serwer aplikacyjny to pojedyncza stacja robocza w sieci wewnętrznej, odpowiadająca za obsługę i odpowiedź na zapytania kierowane do niego przez aplikacje klienckie różnej postaci. Serwer udostępnia wszystkim klientom usługi, w których określone są typy danych wyjściowych i wejściowych. Komunikacja z klientami odbywa się za pomocą protokołu HTTP.

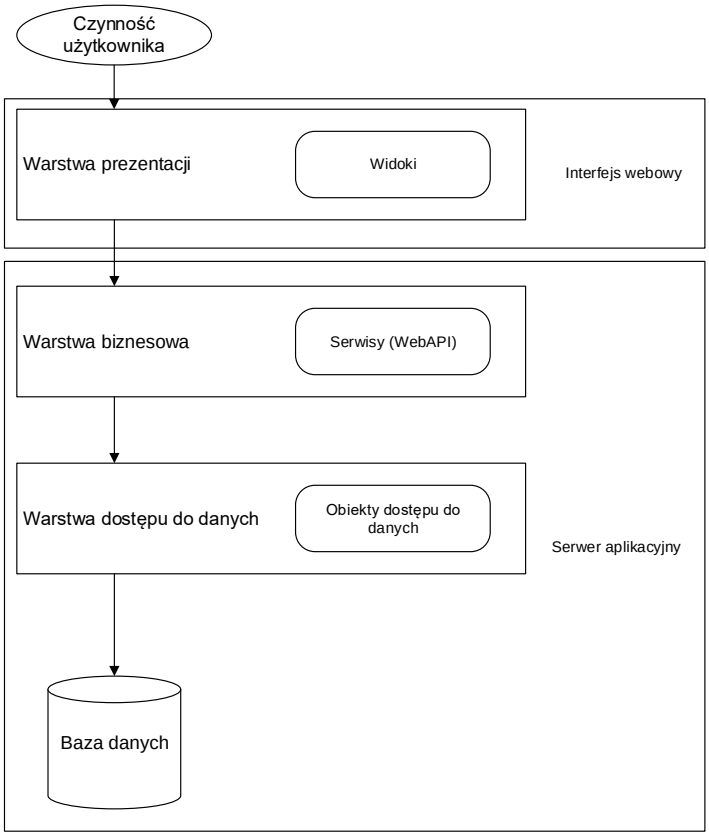
1.5.2 Aplikacja kliencka (dla pracowników)



Rysunek 5: Schemat wykonywania czynności pracownika

Aplikacje klienckie mają postać „grubego klienta”, instalowanego na stacjach roboczych konkretnych użytkowników. Wykonywanie wszystkich żądań użytkownika delegowane jest do serwera aplikacyjnego za pośrednictwem udostępnionych usług; odpowiedzią jest informacja o powodzeniu, lub błąd wykonania. W aplikacji klienckiej nie znajdują się elementy związane z logiką biznesową.

1.5.3 Interfejs webowy



Rysunek 6: Schemat wykonywania czynności klienta

Interfejs webowy umożliwia użytkownikom wysyłanie zapytań do serwera aplikacyjnego z poziomu przeglądarki internetowej. Podobnie jak w przypadku aplikacji klienckiej, nie ma tutaj logiki biznesowej.

2 Dokumentacja końcowa (powykonawcza)

2.1 Wymagania systemowe

Aby zapewnić poprawne działanie aplikacji klienckiej, wymagane są następujące komponenty:

1. System operacyjny Windows 7 lub nowszy.
2. .NET Framework 4.5.2 lub nowszy.

Aby zapewnić poprawne działanie serwera aplikacyjnego, wymagane są następujące komponenty:

1. System operacyjny Windows Server 2008 lub nowszy.
2. .NET Framework 4.5.2 lub nowszy.
3. MS SQL Server 2014 lub nowszy.

2.2 Biblioteki wraz z określeniem licencji

W budowie aplikacji zostały użyte następujące biblioteki oraz komponenty firm trzecich:

Nr	Komponent i wersja	Opis	Licencja	
1	Castle.Core, 3.3.3	Wykorzystywana do tworzenia obiektów *proxy*. Zależność biblioteki Moq.	Apache License 2.0	[1]
2	Entity Framework, 6.1.3	Framework do mapowania obiektowo-relacyjnego (ORM).	Apache License 2.0	[2]
3	FluentAssertions, 4.17.0	Wykorzystywany w testach jednostkowych w celu ułatwienia pisania asercji.	Apache License 2.0	[3]
4	Moq, 4.5.28	Używany w testach jednostkowych do tworzenia obiektów zastępczych (tzw. *mock object*).	BSD 3-Clause	[4]
5	NUnit, 3.5.0	Framework do wykonywania testów jednostkowych.	MIT	[5]
6	ReactiveUI, 6.5.2	Biblioteka wspomagająca w realizacji wzorca MVVM w aplikacji klienckiej, zintegrowana z Reactive Extensions.	MS-PL	[6]
7	Reactive Extensions, 2.2.5	Biblioteka wspomagająca w programowaniu aplikacji opartych na asynchronicznym przetwarzaniu danych oraz zdarzeniach. Zależność ReactiveUI.	Apache License 2.0	[7]
8	Splat, 1.4.0	Kontener IoC wspomagający w realizacji wzorca wstrzykiwania zależności.	MIT	[8]
9	MahApps.Metro, 1.3.0	Biblioteka zawierająca kontrolki interfejsu użytkownika.	MS-PL	[9]
10	AngularJS, 1.5.9	Framework wspomagający tworzenie i rozwijanie Single Page Application.	MIT	[10]
11	Angular Spinner, 1.5.9	Dyrektywa pokazująca animowany spinner ładowania.	MIT	[11]
12	Spin.js, 2.3.2.1	Biblioteka definiująca animowany spinner ładowania.	MIT	[12]

13	Angular toastr, 2.1.1	Biblioteka pokazująca powiadomienia.	MIT	[13]
14	Angular event aggregator, 1.0.0	Biblioteka zawierająca implementację agregatora eventów.	MIT	[14]
15	Angular UI Router, 0.3.1	Biblioteka wspomagająca nawigację na stronie.	MIT	[15]
16	Angular UI Bootstrap, 2.3.2	Biblioteka dyrektyw dla Bootstrapa.	MIT	[16]
17	Font Awesome, 4.7.0	Biblioteka czcionek i ikonek.	Open Font License 1.1	[17]
18	Bootstrap, 3.3.7	Framework CSS.	MIT	[18]
19	OAuth 2.0, 3.0.1	Protokół zabezpieczający API.	Apache License 2.0	[19]
20	Unity, 4.0.1	Kontener dependency injection.	MS-PL	[20]
21	AngularJS Google Maps, 1.17.0	Dyrektywa wspierająca mapy Google.	MIT	[21]
22	jQuery, 3.1.1	Biblioteka ułatwiająca korzystanie z JavaScriptu.	MIT	[22]

Tablica 10: Lista użytych bibliotek i komponentów

2.3 Instrukcja instalacji

Aby zainstalować aplikację na serwerze aplikacyjnym, należy wykonać następujące kroki:

1. Zainstalowanie **.NET Framework w wersji 4.0 lub późniejszej**, usług **Internet Information Service w wersji 8 lub późniejszej**, **Microsoft SQL Server w wersji 2014 lub późniejszej** oraz **ASP.NET** na serwerze aplikacyjnym.
   Aplikacja do funkcjonowania wymaga instalacji serwera bazy danych Microsoft SQL Server w wersji 2014 lub późniejszej. Instrukcję instalacji SQL Server można znaleźć w pozycji bibliografii [23]. Instrukcję instalacji usługi IIS można znaleźć w pozycji [24]. Instrukcję konfiguracji usługi ASP.NET można znaleźć w pozycji [25].

2. **Instalacja serwisów WCF, kontrolerów API oraz Web na serwerze aplikacyjnym**
   Po skopiowaniu plików dostarczonych jako część aplikacji do wybranej lokalizacji na serwerze, należy otworzyć aplikację **IIS Manager**. Tworzymy nową stronę w zakładce **Sites** po lewej, po czym klikamy **View Applications** po prawej stronie. Następnie z prawej strony wybieramy **Add Application**. W okienku wpisujemy **Alias**, który będzie służyć jako prefiks endpointów oraz podajemy ścieżkę do plików serwisowych aplikacji. Proces dodawania strony internetowej powtarzamy dla każdej strony, którą chcemy później uruchomić (WCF, API, Web).

   - **Uwaga:** Serwisy WCF powinny zostać zainstalowane na osobnym serwerze, zaś API oraz witryna Web – na innym. Serwer obsługujący serwisy WCF powinien być odseparowany od publicznego Internetu. Szczegółowe informacje znajdują się w sekcji 1.5.

3. **Konfiguracja connection stringów**
   Po dodaniu aplikacji należy kliknąć na nią po lewej stronie w okienku **Connections**, następnie na **Connection strings** i wyedytować connection string, aby wskazywał on na zainstalowaną instancję bazy.

4. **Konfiguracja API URL i Google Maps Token**
   W pliku **app.config.js** znajduje się adres dostępu do API pod zmienną **baseApiUrl** oraz token konieczny do korzystania z Google Maps pod zmienną **googleMapsToken**. Należy je odpowiednio uzupełnić przed uruchomieniem strony.

5. **Uruchomienie serwera**
   Przed pierwszym uruchomieniem aplikacji należy upewnić się, że serwer działa, uruchamiając

SQL Server Configuration Manager i sprawdzając, czy status usługi **MSSQLSERVER** to **Running**.

6. **Pierwsze uruchomienie grubego klienta**
   Po uruchomieniu aplikacji należy wpisać dowolne dane logowania i wcisnąć przycisk **Login**. W tym momencie przycisk powinien się zablokować, a po kilkunastu sekundach powinien pojawić się komunikat o błędnych danych logowania. Oznacza to, że schemat bazy danych został pomyślnie utworzony; aby to potwierdzić, należy uruchomić **SQL Server Management Studio** i zweryfikować, że schemat bazy danych został utworzony.

7. **Wykonanie skryptu z przykładowymi danymi**
   Po wykonaniu poprzedniego kroku, należy za pośrednictwem **SQL Server Management Studio** wykonać dostarczony skrypt T-SQL, aby dodać do bazy danych przykładowe dane. Wówczas można zalogować się do aplikacji używając danych wyspecyfikowanych w poniższej sekcji, a następnie dokonywać dalszego dostosowywania systemu do własnych potrzeb.

2.4 Instrukcja uruchomienia

Na serwerze aplikacyjnym należy upenić się, że usługa IIS dla naszej strony oraz instancja serwera MS SQL jest uruchomiona.

1. Otwórz **SQL Server Configuration Manager** i spradź, czy status usługi (**MSSQLSERVER**) to **Running**. Jeżeli nie, uruchom usługę za pomocą przycisku **Start Service** na pasku pod menu.
2. Otwórz konsolę **IIS**.
3. Wybierz swoją stronę po lewej stronie okna.
4. Po prawej stronie okna wybierz opcję **Start**, jeżeli usługa nie jest uruchomiona.
5. Powtórz punkty 3. i 4. dla wszystkich stron, które chcesz uruchomić (WCF, API, Web).

Na stacji roboczej możemy wówczas uruchomić aplikację, aby połączyć się z serwerem aplikacyjnym.

1. Klikamy dwukrotnie plik wykonywalny **PublicTransport.exe**, aby uruchomić aplikację.

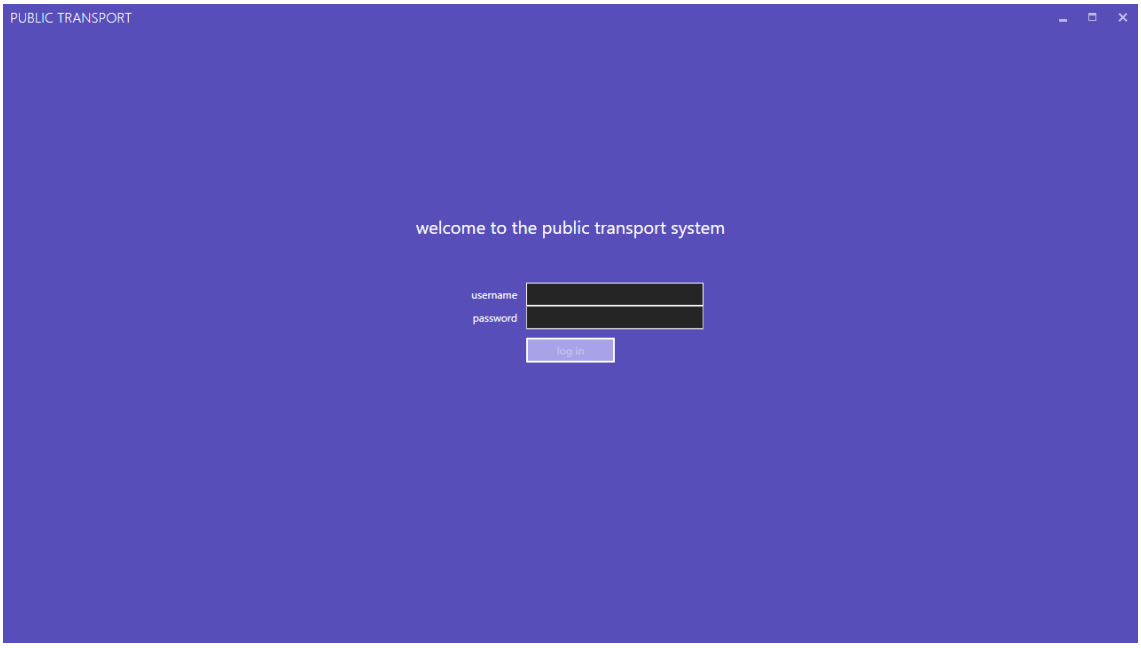
Po uruchomieniu stron API oraz Web, można korzystać z interfejsu webowego, wchodząc na odpowiedni adres w przeglądarce internetowej.

2.5 Instrukcja użycia

2.5.1 Aplikacja dla pracowników

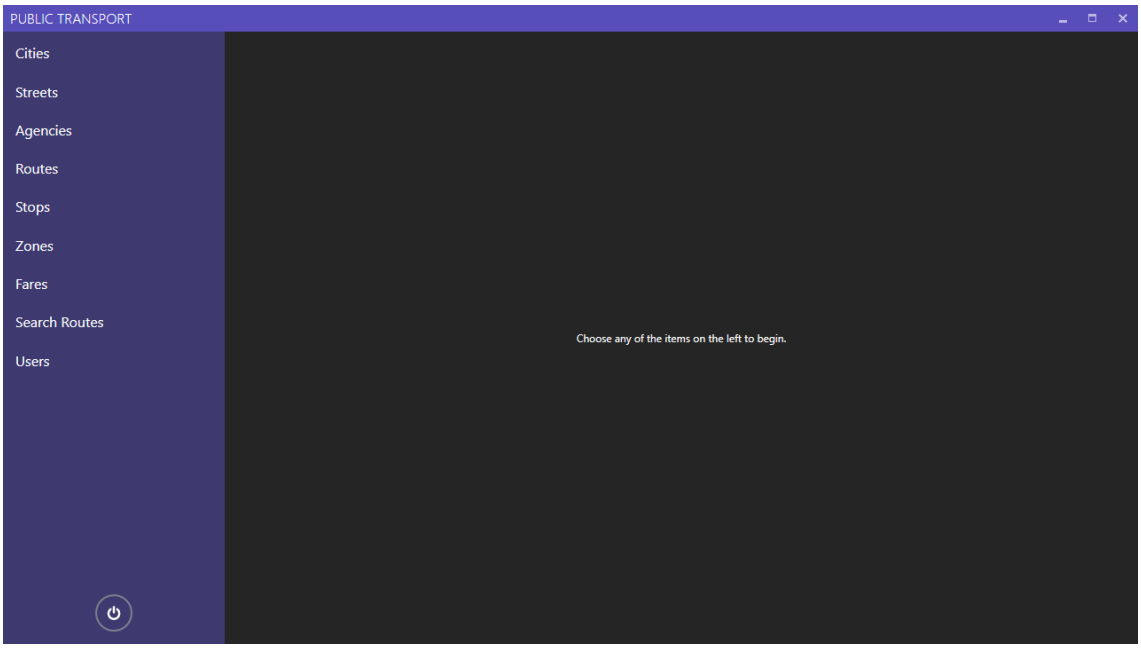
Logowanie do systemu Po uruchomieniu aplikacji przez użytkownika pojawia się okno logowania. Predefiniowane są następujące konta użytkowników:

- użytkownik `root`, hasło `root`: konto z uprawnieniami administratora,
- użytkownik `employee`, hasło `password`: konto z uprawnieniami użytkownika,
- użytkownik `guest`, hasło `guest`: konto bez nadanych uprawnień.



Rysunek 7: Okno logowania do systemu

Główne okno aplikacji Po podaniu prawidłowej kombinacji nazwy użytkownika i hasła, wyświetlony zostaje główne okno aplikacji. Po lewej stronie znajduje się menu nawigacyjne, które umożliwia dostęp do poszczególnych części systemu, zaś pod menu znajduje się zaś przycisk odpowiadający za wylogowanie użytkownika z systemu.



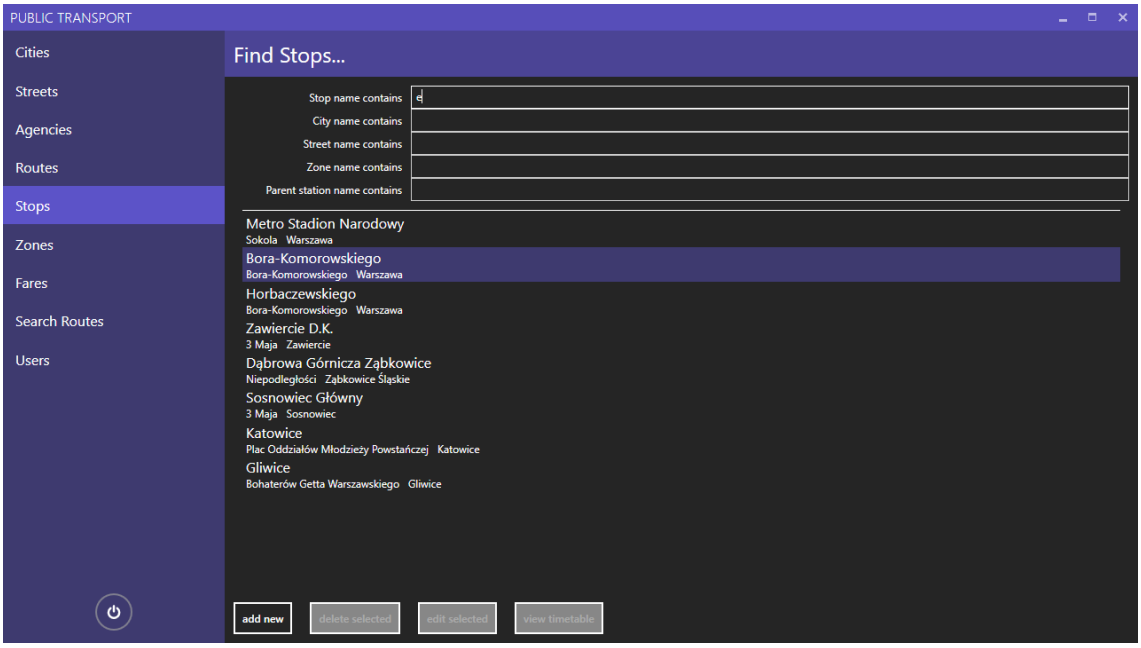
Rysunek 8: Główny widok aplikacji

Po kliknięciu dowolnej opcji menu, po prawej stronie aplikacji wyświetla się formularz wyszukiwania odpowiadający wybranej opcji. Dostępne opcje to:

- **Cities**, **Streets** – łączy funkcjonalności związane z miastami i ulicami,
- **Agencies** – agreguje informacje dotyczące przewoźników,
- **Routes** – wyświetla dane o trasach i przejazdach,

- **Stops** – pozwala wyszukiwać, edytować i dodawać przystanki,
- **Zones** – umożliwia wyznaczanie stref taryfowych, na podstawie których obliczana będzie cena biletu,
- **Fares** – zbiera dane dotyczące taryf przejazdowych i cen biletów,
- **Users** – zawiera informacje o użytkownikach; widok ten dostępny jest tylko dla administratorów.

Wyszukiwanie W przypadku wszystkich wyżej wymienionych opcji, po wybraniu po prawej stronie pokazuje się widok pozwalający na przeszukiwanie danych zawartych w bazie dotyczących wybranej zakładki.

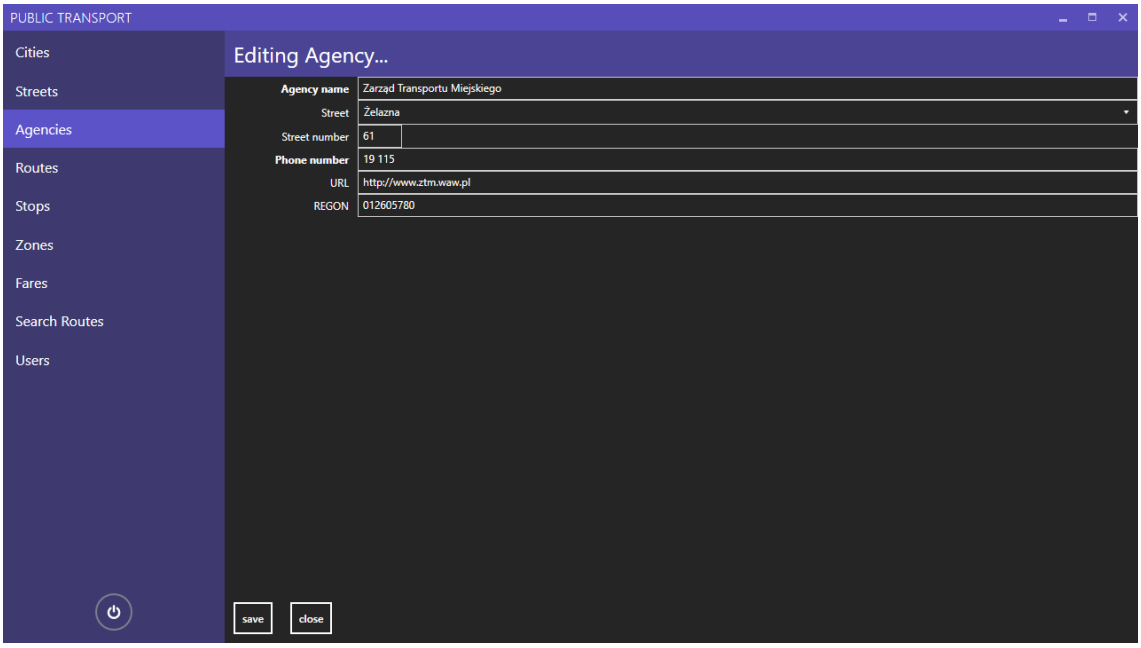


Rysunek 9: Przykładowy widok wyszukiwania

Po wybraniu opcji lista znalezionych rekordów w bazie będzie pusta. Aby rozpocząć wyszukiwanie, należy zacząć wprowadzanie kryteriów wyszukiwania w polach znajdujących się nad listą. Zawartość listy zaktualizuje się automatycznie w ciągu pół sekundy od zakończenia wprowadzania danych.

Poniżej listy umieszczony jest pasek narzędziowy, umożliwiający dodanie nowego rekordu (**Add New**) oraz edycję (**Edit Selected**) bądź usunięcie (**Delete Selected**) obecnie zaznaczonego rekordu. Pierwszy z tych przycisków jest aktywny zawsze, zaś pozostałe uaktywniają się, gdy zaznaczony jest jeden z elementów listy z wynikami wyszukiwania.

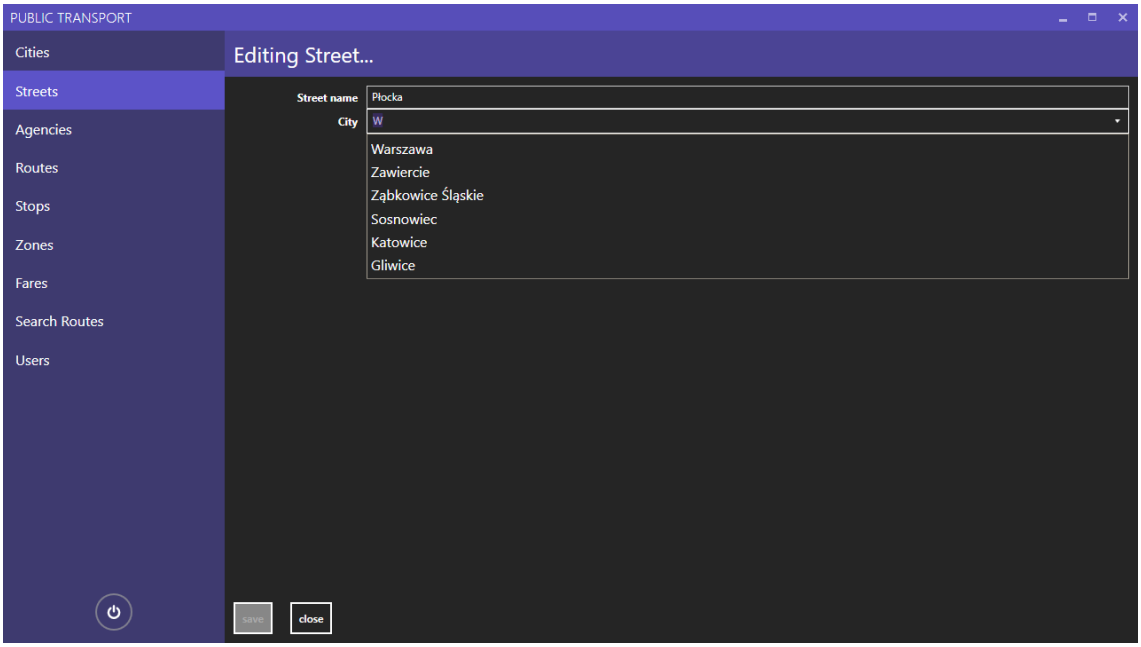
Edycja Po wybraniu opcji dodania lub edycji rekordu, po prawej stronie wyświetli się formularz umożliwiający na wprowadzenie danych nowego rekordu.



Rysunek 10: Przykładowy widok edycji dla przewoźników

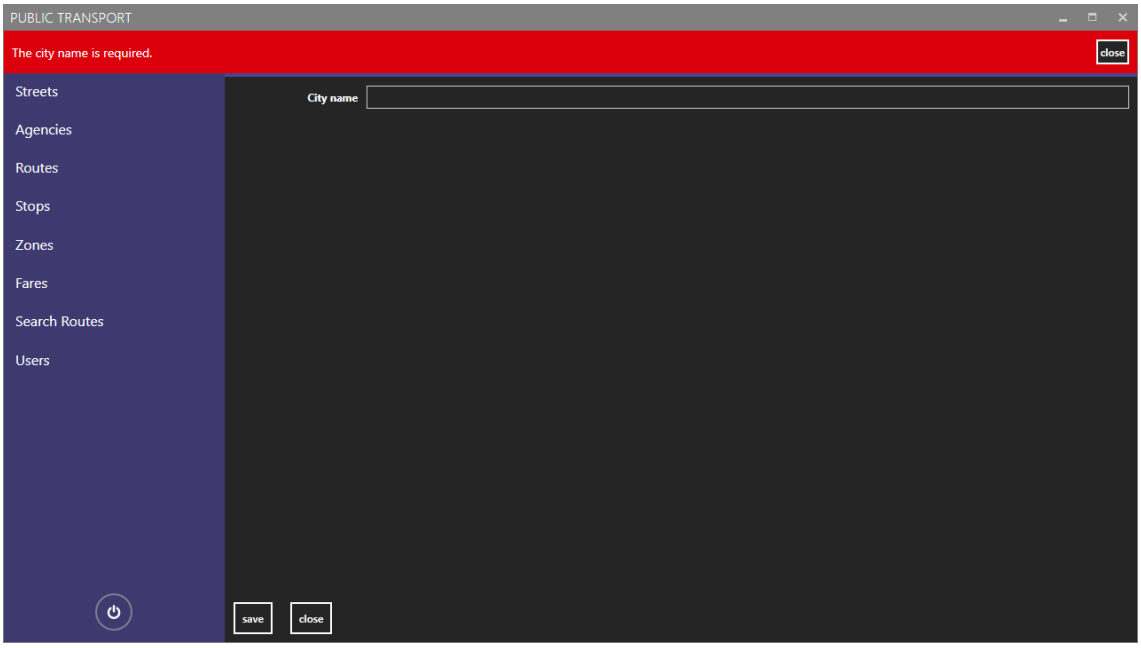
Etykiety pól obowiązkowych oznaczone są pogrubioną czcionką. Można wyróżnić dwa rodzaje pól:

- zwykłe – nie są związane z żadnymi innymi obiektami systemu,
- menu rozwijane – zawartość tego pola jest związane z inną częścią systemu. Przykładem takiego pola jest pole **Street** na powyższym rysunku. Aby wybrać wartość w tym polu, należy wprowadzić początek nazwy żądanego obiektu – po chwili pojawi się menu rozwijane z sugestiami, z którego można wybrać żądany obiekt.



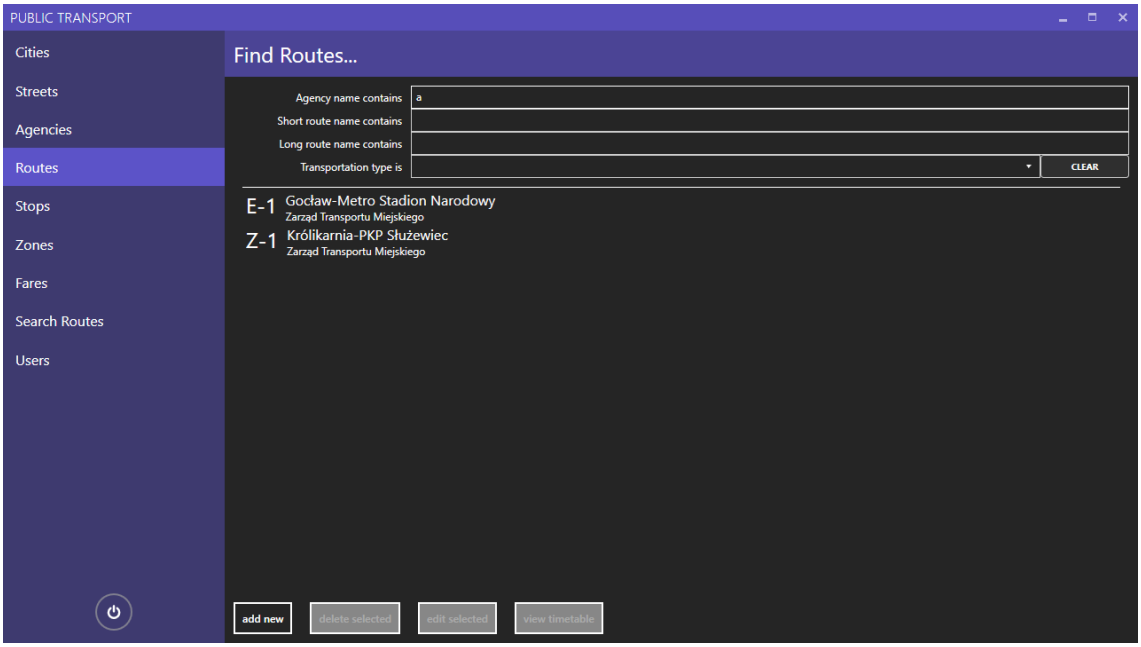
Rysunek 11: Przykład podpowiedzi

Na dole ekranu znajdują się przyciski zapisu (**Save**) umożliwiający zatwierdzenie zmian oraz zamknięcia (**Close**), który pozwala na cofnięcie się do widoku wyszukiwania i odrzucenie ostatnich zmian. W przypadku próby zapisu obiektu, który nie ma wypełnionych wszystkich pól, pojawia się poniżej przedstawiony pasek z wiadomością o błędzie:



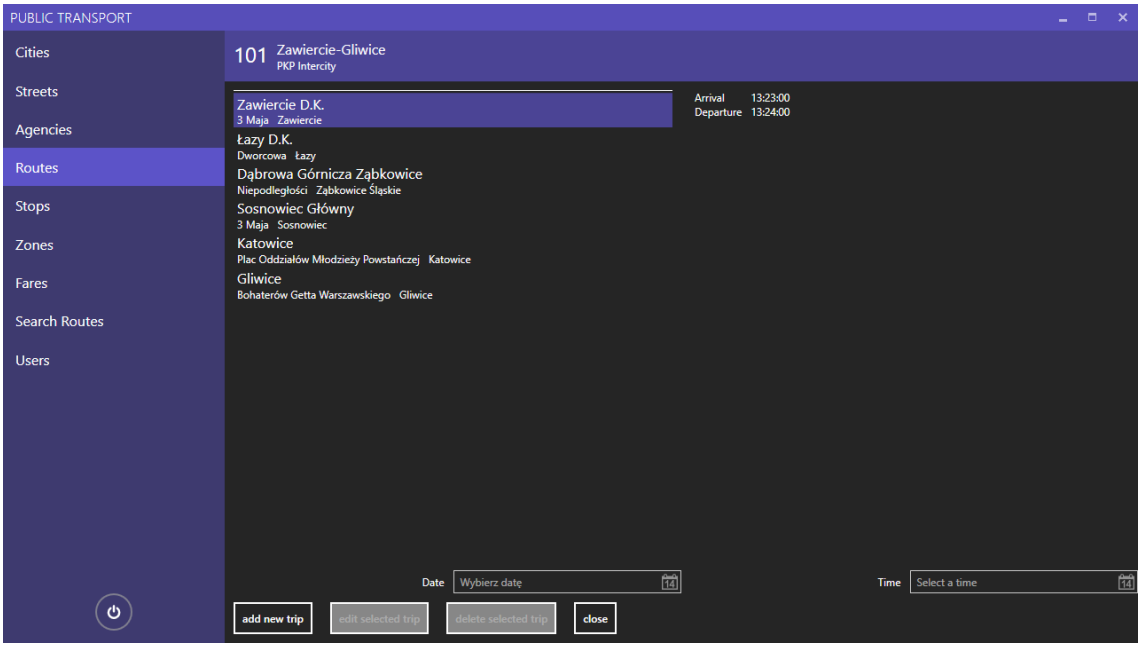
Rysunek 12: Błąd zapisu

Wyświetlanie rozkładu jazdy wybranej linii Szczególnym widokiem aplikacji jest widok rozkładu jazdy wybranej linii transportowej. Aby wyświetlić rozkład, należy przejść do zakładki **Routes**, wyszukać i wybrać jedną z tras, i wreszcie wybrać przycisk **View Timetable**.



Rysunek 13: Widok tras z przyciskiem wyświetlania rozkładu

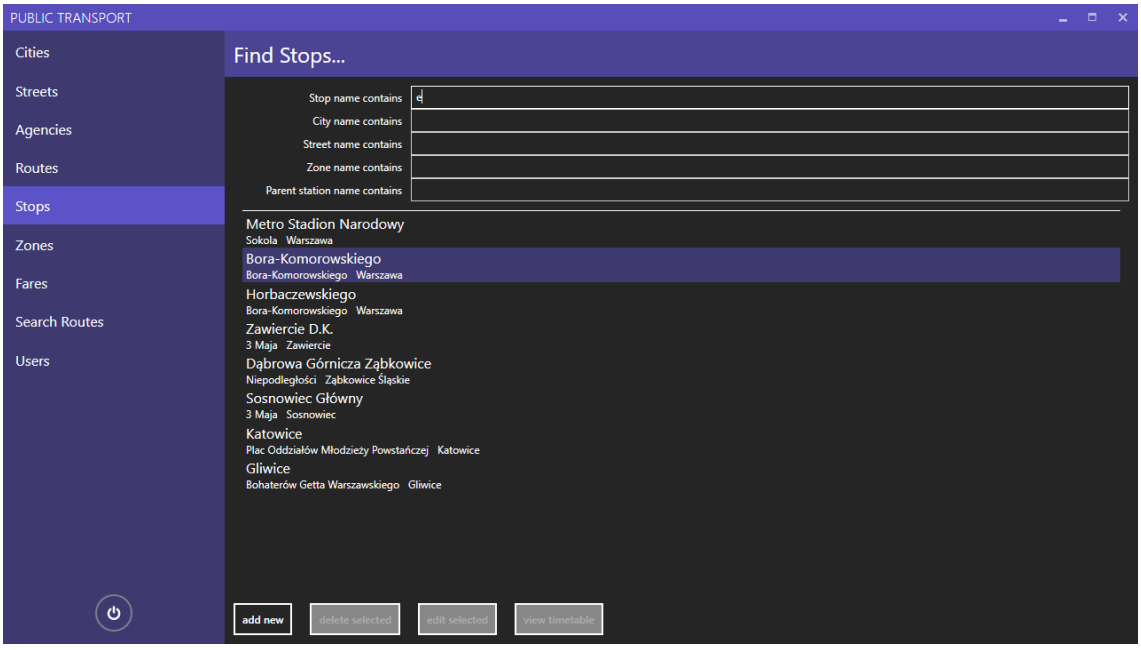
Widok rozkładu podzielony jest na dwie części. Po lewej stronie znajdują się poszczególne przystanki, między którymi kursuje wybrana trasa. Po wybraniu konkretnego przystanku, po prawej stronie pojawiają się czasy przyjazdu i odjazdu wszystkich kursów tej linii dotyczące wybranej opcji.



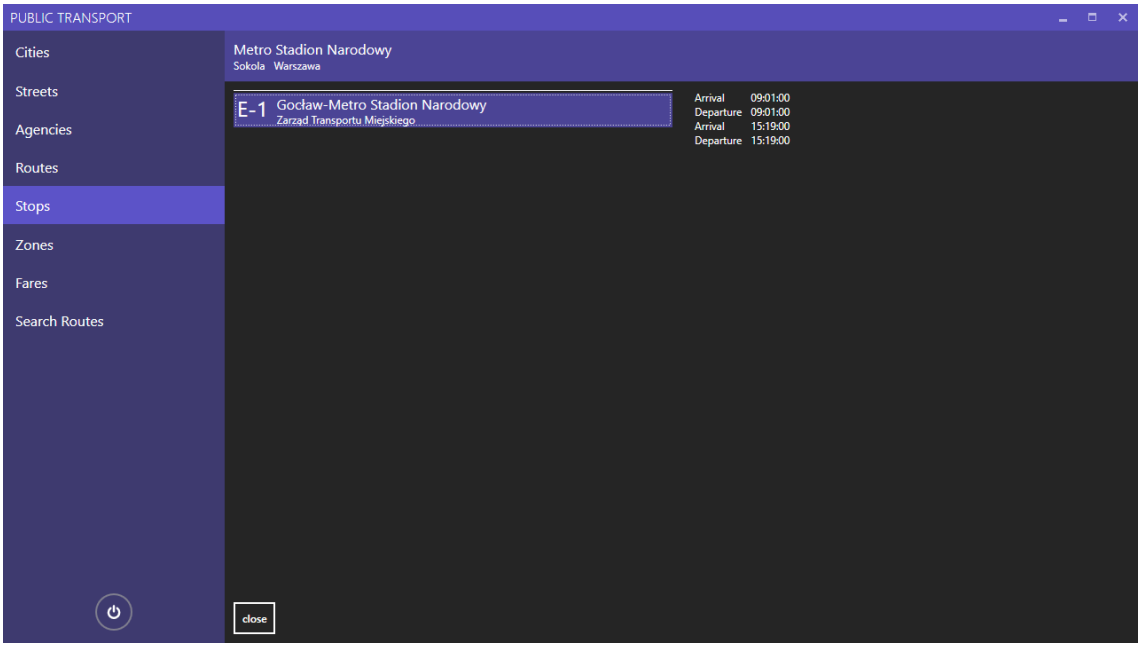
Rysunek 14: Widok rozkładu jazdy

Pod listami przystanków i godzin widoczne są jeszcze dwa pola, które umożliwiają wyszukiwanie połączeń w konkretnych dniach oraz o godzinach późniejszych niż godzina wybrana w polu **Time**. Dodatkowo, dostępne są opcje dodawania (**Add Trip**), edycji (**Edit Trip**) oraz usuwania (**Delete Trip**) wybranego przejazdu.

Wyświetlanie rozkładu jazdy dla wybranego przystanku W podobny sposób można wyświetlić rozkład jazdy wszystkich linii dla jednego z przystanków, wybierając zakładkę **Stops**, wyszukując i wybierając jeden z przystanków, i wciskając przycisk **View Timetable**.

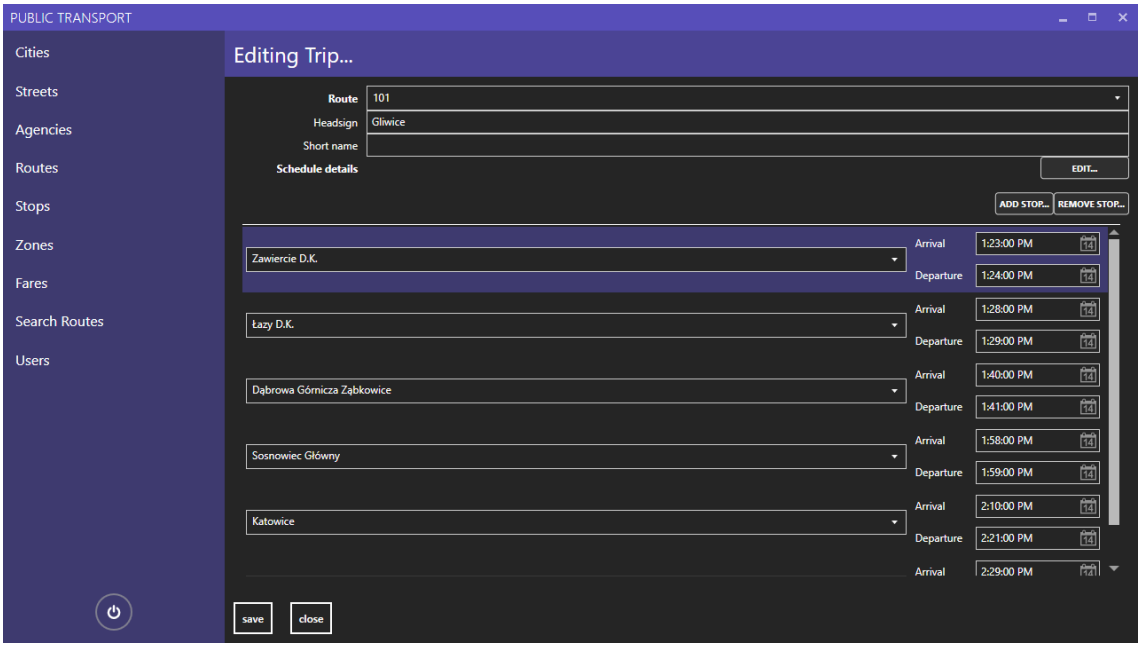


Rysunek 15: Widok przystanków z przyciskiem wyświetlania rozkładu



Rysunek 16: Rozkład jazdy dla wybranego przystanku

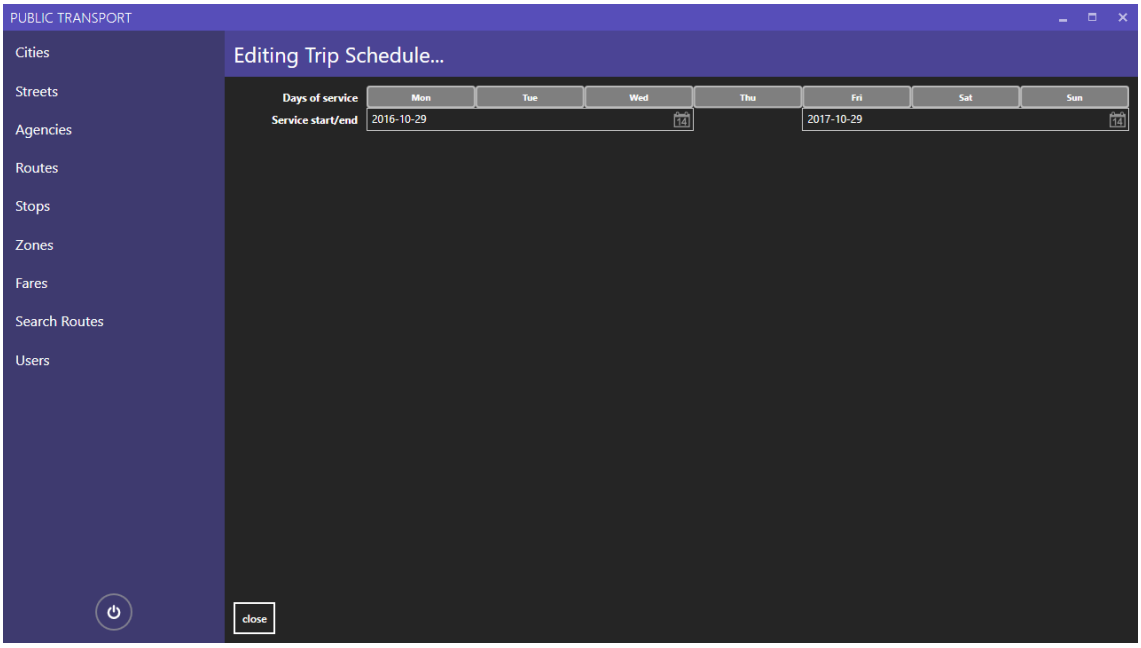
Edycja przejazdu Poniżej widoczny jest ekran edycji pojedynczego kursu.



Rysunek 17: Widok edycji przejazdu

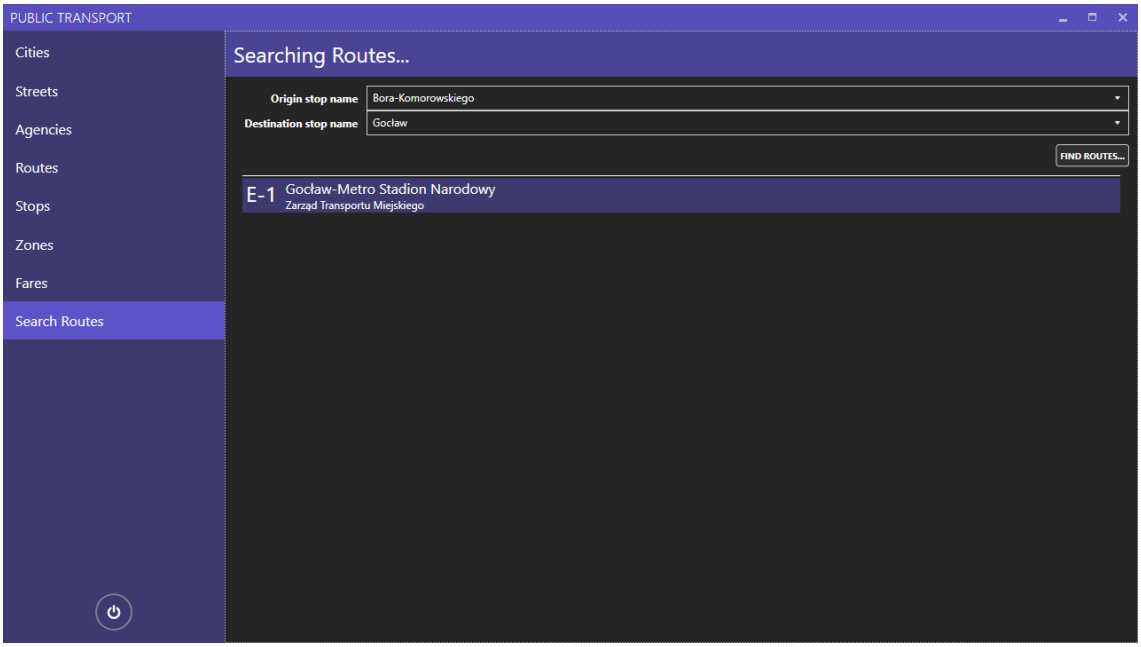
Przewijana lista w dolnej części ekranu zawiera informacje o kolejnych przystankach edytowanego przejazdu oraz czasach przyjazdu i odjazdu. Można dodawać oraz usuwać przystanki za pomocą opcji **Add stop...** i **Remove stop...**

Przycisk **Edit...** obok etykiety **Schedule details** pozwala na edycję informacji o harmonogramie danego kursu, takich, jak: pierwszy i ostatni dzień funkcjonowania kursu oraz dni tygodnia, w które odbywa się dany przejazd.



Rysunek 18: Widok edycji harmonogramu

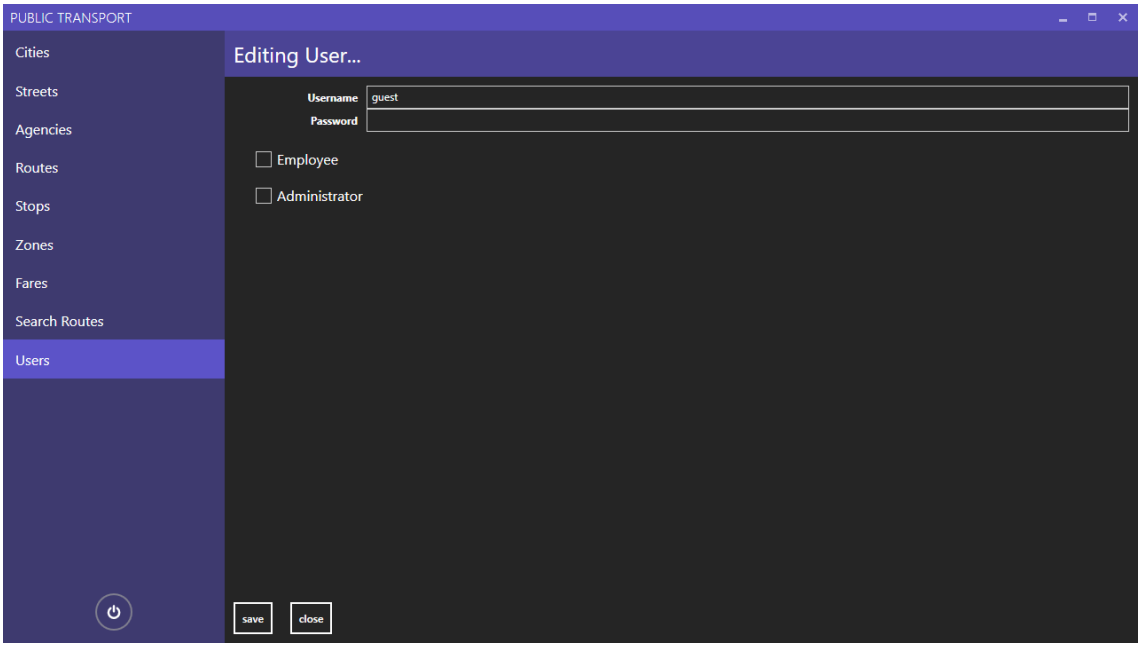
Wyszukiwanie tras między wybranymi przystankami Aplikacja umożliwia wyszukiwanie tras przejeżdżających przez wybrane przystanki. Funkcjonalność ta dostępna jest z poziomu opcji **Search Routes** po lewej stronie okna.



Rysunek 19: Widok wyszukiwania tras

Aby dokonać wyszukiwania, należy wpisać nazwy przystanków: startowego oraz docelowego w pola **Origin stop name** oraz **Destination stop name** odpowiednio, wybrać jeden z sugerowanych elementów, a następnie wcisnąć przycisk **Find Routes**.

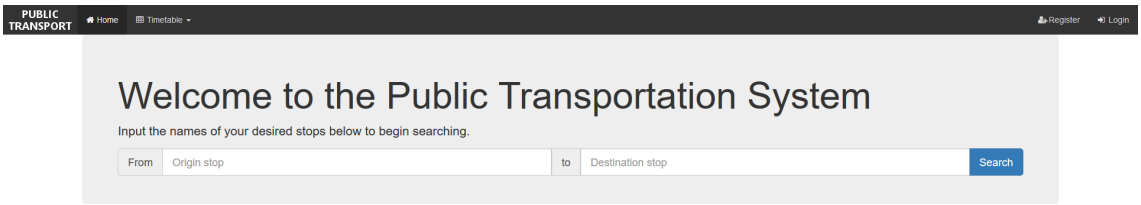
Edycja użytkowników Konta administratorów mają dostęp również do widoku edycji użytkowników.



Rysunek 20: Widok edycji użytkowników

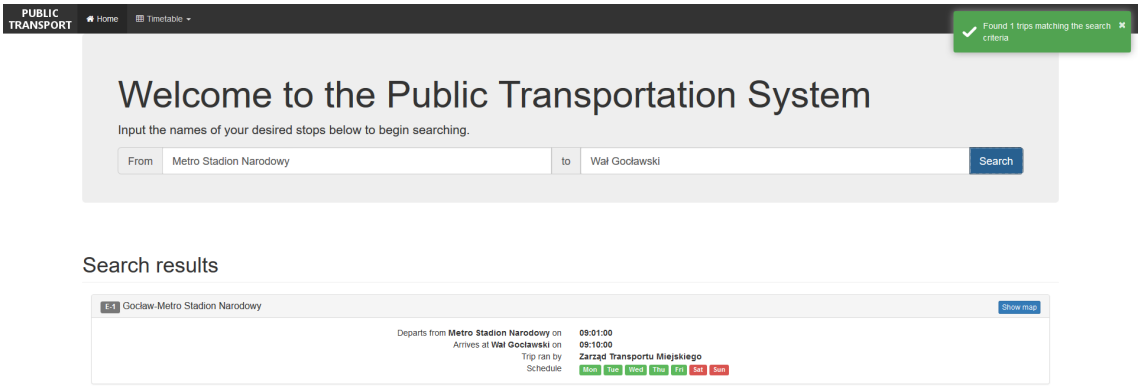
W widoku tym można przydzielać oraz odbierać danym użytkownikom role oraz resetować ich hasło, podając hasło tymczasowe (nie ma możliwości wyświetlenia hasła danego użytkownika). Aby zmiany zostały wprowadzone, dany użytkownik musi wylogować się z aplikacji i zalogować się ponownie.

2.5.2 Aplikacja webowa



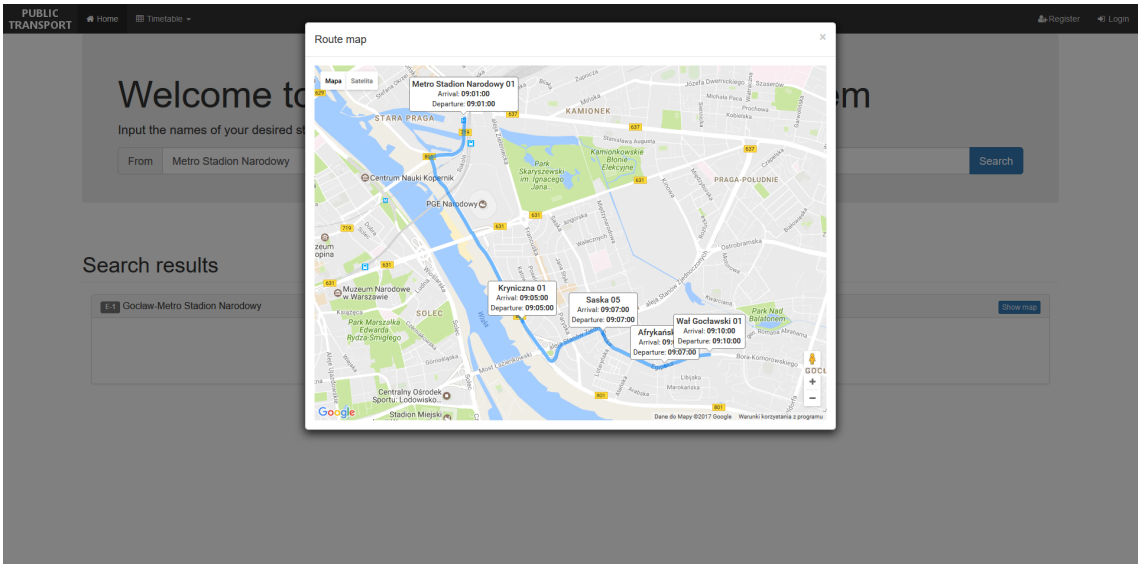
Rysunek 21: Strona główna aplikacji

Strona główna – wyszukiwanie połączeń Strona główna aplikacji webowej składa się z paska nawigacyjnego, umożliwiającego przejście do pozostałych podstron witryny, oraz widoku głównego, odpowiedzialnego za wyszukiwanie połączeń. Aby wyszukać połączenie między wybranymi przystankami, należy rozpocząć wprowadzanie ich nazw do odpowiednich pól tekstowych **From** i **To**; po około pół sekundy w menu rozwijanym pojawią się sugestie dla użytkownika. Właściwe wyszukiwanie dokonywane jest po wciśnięciu przycisku **Search**.



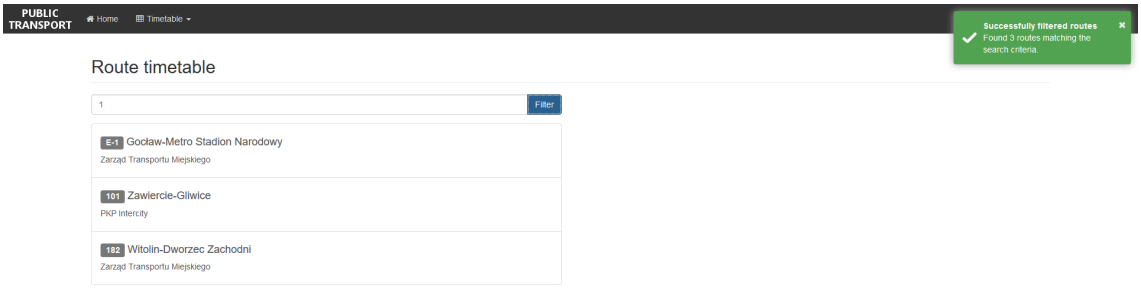
Rysunek 22: Wyświetlone wyniki wyszukiwania

Po wciśnięciu przycisku **Search** ukaże się powiadomienie o sukcesie (lub braku połączeń spełniających podane kryteria) oraz wyniki wyszukiwania, zawierające informacje takie, jak: nazwa i identyfikator połączenia, godzina odjazdu z przystanku źródłowego, godzina przyjazdu na przystanek końcowy, nazwę agencji transportowej obsługującą połączenie oraz dni tygodnia, w które odbywa się kurs. Dodatkowo, po wciśnięciu przycisku **Show map** w prawym górnym rogu można wyświetlić w oknie modalnym trasę w postaci mapy.



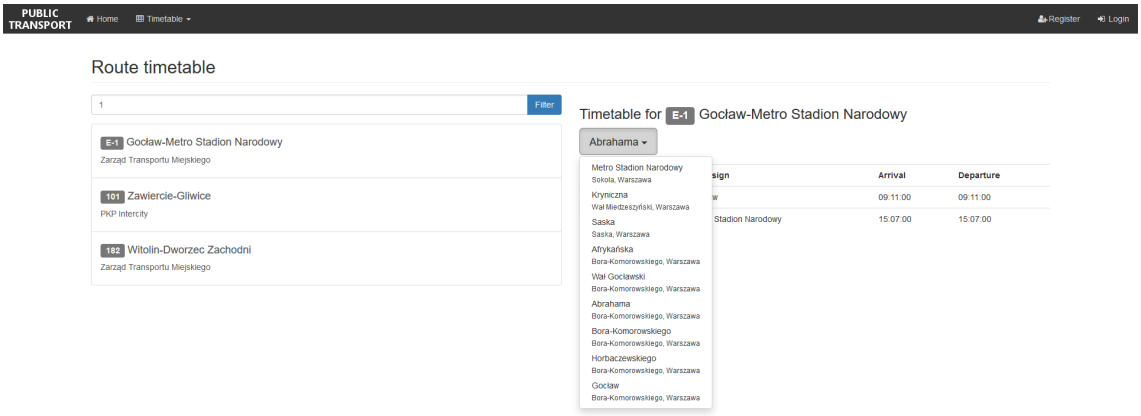
Rysunek 23: Widok mapy

Rozkłady jazdy Wybierając w menu nawigacyjnym u góry opcję **Timetable**, otrzymujemy dostęp do menu rozwijanego umożliwiającego przejście do widoków odpowiedzialnych za wyświetlanie rozkładów jazdy dla poszczególnych przystanków (**Search by stop**) oraz tras (**Search by route**). Po wybraniu jednej z tych opcji nastąpi przejście do widoku składającego się z dwóch części. Po lewej znajduje się pole tekstowe, w które należy wpisać nazwę wyszukiwanej trasy; po kliknięciu przycisku **Filter** poniżej pola pojawi się lista tras, których nazwa zawiera wprowadzony ciąg znaków.



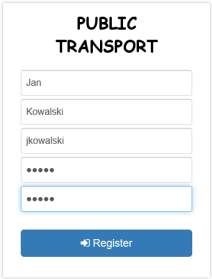
Rysunek 24: Wyszukiwanie rozkładów dla tras

Kliknięcie na dowolny z wyników wyszukiwania spowoduje pojawienie się po prawej stronie ekranu menu rozwijanego, z którego można wybrać przystanek trasy, dla którego chcemy sprawdzić rozkład jazdy. Po wybraniu jednej z opcji menu rozkład pojawi się poniżej w formie tabeli.



Rysunek 25: Menu rozwijane wyboru przystanku i widok rozkładu

Rejestracja Aby utworzyć konto w serwisie, należy wybrać opcję **Register** na pasku nawigacyjnym.



Rysunek 26: Okno rejestracji

Użytkownik zostanie poproszony o wprowadzenie swojego imienia, nazwiska, żądanej nazwy konta oraz wprowadzenie i potwierdzenie hasła do konta. Po pomyślnym wypełnieniu formularza i wciśnięciu przycisku **Register** użytkownik otrzyma powiadomienie, że może zalogować się już do swojego konta.

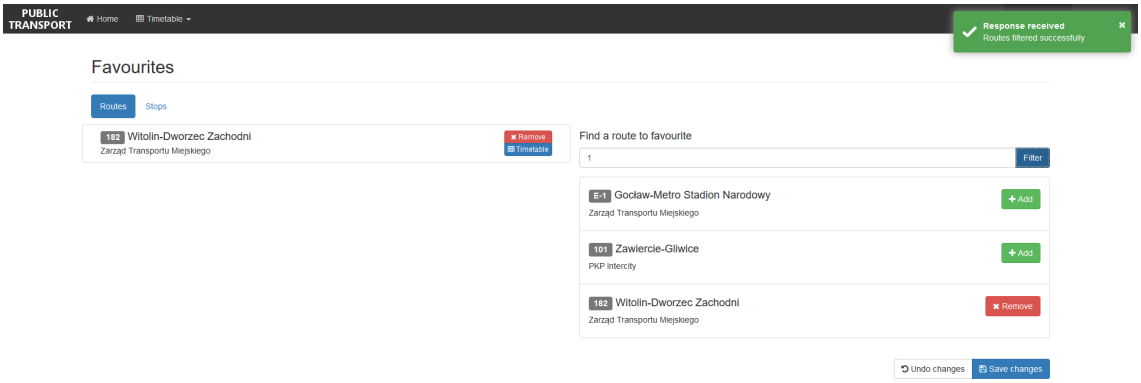
Logowanie Opcja zalogowania się do serwisu jest dostępna pod przyciskiem **Login** na pasku nawigacyjnym.

Rysunek 27: Okno logowania

Po wprowadzeniu nazwy użytkownika oraz hasła użytkownik zostanie zalogowany do systemu.

Po zalogowaniu w prawym górnym rogu ekranu pojawi się menu rozwijane dla użytkownika, umożliwiające wylogowanie z systemu (**Logout**) oraz zmianę hasła za pomocą przycisku **Settings**.

Ulubione Zarejestrowani użytkownicy mają możliwość zapisywania wybranych tras oraz przystanków jako ulubione, aby uniknąć powtarzających się wyszukiwań. Aby zmodyfikować ulubione, po zalogowaniu się należy wybrać opcję **Favourites** na pasku nawigacyjnym.



Rysunek 28: Widok dodawania ulubionych

Pod nagłówkiem znajdują się dwa przyciski, które umożliwiają przełączanie się między zakładkami dla przystanków oraz tras. Po lewej stronie widoczne są obecne ulubione, zaś po prawej można wyszukać inne trasy i dodać (przycisk **Add**) lub usunąć (**Remove**) je z ulubionych. Aby zatwierdzić dokonane zmiany, należy wcisnąć przycisk **Save changes**; zmiany można też bezpiecznie cofnąć, klikając **Undo changes** lub opuszczając stronę bez kliknięcia przycisku **Save changes**.

Dodatkowo, dla tras lub przystanków wybranych jako ulubione, można bezpośrednio przejść do ich rozkładu jazdy, wybierając przycisk **Timetable**.

2.6 Instrukcja utrzymania

Uruchamianie, restartowanie lub zatrzymywanie serwera aplikacyjnego.

1. Otwórz konsolę **IIS**.
2. Wybierz swoją stronę (**Web Site**) po lewej stronie okna.
3. Po prawej stronie okna wybierz jedną z opcji **Start, Stop** lub **Restart**.

Uruchamianie, restartowanie lub zatrzymywanie serwera bazodanowego.

1. Uruchom **SQL Server Configuration Manager**.
2. Jeśli pojawi się okno dialogowe **Kontrola konta użytkownika**, kliknij przycisk **Tak**.
3. Wybierz instancję serwera SQL (MSSQLSERVER), kliknij prawym przyciskiem myszy, a następnie wybierz jedną z opcji **Start, Stop, Pause, Resume** lub **Restart**.

Tworzenie kopii zapasowej bazy danych.

1. Połącz się z instancją serwera bazodanowego i w **Eksploratorze obiektów** przejdź do węzła **Databases**.
2. Kliknij prawym przyciskiem bazę danych **PublicTransport**, przejdź do zakładki **Tasks**, a następnie **BackUp**.
3. Zostanie wyświetlone okno dialogowe **Back Up Database**. W polu listy **Database** zweryfikuj nazwę bazy danych.
4. W polu listy **Backup type** wybierz opcję **Full**.
5. W obszarze **Backup component** kliknij opcję **Database**.
6. Wybierz lokalizację docelową kopii zapasowej i wciśnij przycisk **OK**.

2.7 Raport odstępstw od specyfikacji wymagań

Brak.

2.8 Dokumentacja usług Web Services

```
swagger: "2.0"
info:
  title: Public Transport
  version: 3.0.0
  contact:
    name: Bartłomiej Dach, Tymon Felski
    url: http://github.com/bdach/public-transport/
    email: dachb@student.mini.pw.edu.pl, felskit@student.mini.pw.edu.pl
host: localhost:49878
basePath: /api
definitions:
  Agency:
    type: object
    properties:
      Name:
        type: string
        description: The full name of the transit agency.
      Phone:
        type: string
        description: Contains a single voice telephone number for the specified agency.
        pattern: "[0-9+#]+"
      Url:
        type: string
        description: Contains the URL of the transit agency. The value must be a fully
                     qualified URL that includes ‘http://‘ or ‘https://‘ and any special
                     characters in the URL must be correctly escaped.
      Regon:
        type: string
        description: Contains the REGON number of the transit agency.
      StreetName:
        type: string
        description: Contains the name of the street the agency is located on.
      StreetNumber:
        type: string
        description: Contains the street number of the agency.
      CityName:
        type: string
        description: Contains the name of the city the agency is located in.
  ErrorMessage:
    type: object
    properties:
      Message:
        type: string
        description: Message detailing the error cause.
  FavouriteInfo:
    type: object
    properties:
      UserName:
        type: string
        required: true
        description: The username of the user sending the request to save changes to
                     the favourites.
```

```
    Changes:
      type: object
      request: true
      description: This is a key-value pair. The key is an integer indicating the ID
                   of a stop or route, and the value indicates whether the stop/route
                   should be added to the favourite list (true) or removed from
                   the list (false). This object can have multiple pairs.
LoginData:
  type: object
  properties:
    UserName:
      type: string
      required: true
      description: The username of the user logging in.
    Password:
      type: string
      required: true
      description: The password of the user logging in.
MapMarker:
  type: object
  properties:
    Latitude:
      type: number
      minimum: 0
      maximum: 90
      exclusiveMaximum: false
      description: The latitude of the point represented by the marker.
    Longtitude:
      type: number
      minimum: -180
      maximum: 180
      exclusiveMaximum: false
      description: The longtitude of the point represented by the marker.
    DescriptionText:
      type: string
      description: String to be used as a description for the marker.
    ArrivalTime:
      type: string
      description: The time of arrival at the marker's location.
    DepartureTime:
      type: string
      description: The time of departure at the marker's location.
PasswordChangeRequest:
  type: object
  properties:
    UserName:
      type: string
      description: The username of the user requesting a password change.
      required: true
    OldPassword:
      type: string
      description: The user's old password.
      required: true
    NewPassword:
      type: string
      description: The user's new password
      required: true
RegistrationInfo:
```

```
  type: object
  properties:
    FullName:
      type: string
      description: The user's full name to be stored by the app.
      required: true
    UserName:
      type: string
      description: Username to be used by the user to log in to the app.
      required: true
    Password:
      type: string
      description: The user's desired password, to be used when logging in.
      required: true
Route:
  type: object
  properties:
    Id:
      type: integer
      format: int64
      description: The ID number of the route.
    ShortName:
      type: string
      description: Contains the short name of a route. This often will be a short,
                   abstract identifier like "32", "100X" or "Green" that riders
                   use to identify a route, but which doesn't give any indication
                   of what places the route serves.
    LongName:
      type: string
      description: Contains the full name of a route. This name is generally more
                   descriptive than the short name and will often include
                   the route's destination or stop.
    RouteType:
      type: integer
      description: Describes the type of transportation used on a route.
      enum:
        - Tram
        - Subway
        - Rail
        - Bus
        - Ferry
    Agency:
      $ref: '#/definitions/Agency'
      description: Contains information about the agency operating the particular route.
RouteFilter:
  type: object
  properties:
    AgencyNameFilter:
      type: string
      description: Agency name filter.
    LongNameFilter:
      type: string
      description: Route long name filter.
    ShortNameFilter:
      type: string
      description: Route short name filter.
    RouteTypeFilter:
      type: integer # these enums are borked rn - getting serialized into numbers
```

```
         and not strings
      enum:
        - Tram
        - Subway
        - Rail
        - Bus
        - Ferry
      description: Route type filter.
RouteTimetableEntry:
  type: object
  properties:
    Key:
      $ref: '#/definitions/Route'
    Value:
      type: array
      items:
        $ref: '#/definitions/TimetableEntry'
ServiceDetails:
  type: object
  properties:
    Monday:
      type: boolean
      description: Contains a binary value that indicates whether the service is valid
                   for all Mondays in the date range.
    Tuesday:
      type: boolean
      description: Contains a binary value that indicates whether the service is valid
                   for all Tuesdays in the date range.
    Wednesday:
      type: boolean
      description: Contains a binary value that indicates whether the service is valid
                   for all Wednesdays in the date range.
    Thursday:
      type: boolean
      description: Contains a binary value that indicates whether the service is valid
                   for all Thursdays in the date range.
    Friday:
      type: boolean
      description: Contains a binary value that indicates whether the service is valid
                   for all Fridays in the date range.
    Saturday:
      type: boolean
      description: Contains a binary value that indicates whether the service is valid
                   for all Saturdays in the date range.
    Sunday:
      type: boolean
      description: Contains a binary value that indicates whether the service is valid
                   for all Sundays in the date range.
    StartDate:
      type: string
      format: date-time
      description: Contains the start date for the service.
    EndDate:
      type: string
      format: date-time
      description: Contains the end date for the service.
Stop:
  type: object
```

```
  properties:
    Id:
      type: integer
      format: int64
      description: Contains the ID of the stop.
    Name:
      type: string
      description: Contains the name of a stop or station.
    StreetName:
      type: string
      description: Contains the name of the street the stop is located on.
    CityName:
      type: string
      description: Contains the name of the city the stop is located in.
    ParentStation:
      $ref: '#/definitions/Stop'
      description: For stops that are physically located inside stations, this field
                   identifies the station associated with the stop. Used only if
                   the value of `IsStation` is false.
    IsStation:
      type: boolean
      description: Identifies whether this stop object represents a stop or station.
StopFilter:
  type: object
  properties:
    StopNameFilter:
      type: string
      description: Contains the stop name string filter parameter.
    StreetNameFilter:
      type: string
      description: Contains the street name string filter parameter.
    CityNameFilter:
      type: string
      description: Contains the city name string filter parameter.
    ZoneNameFilter:
      type: string
      description: Contains the zone name string filter parameter.
    ParentStationNameFilter:
      type: string
      description: Contains the parent station name string filter parameter.
    OnlyStations:
      type: boolean
      description: Limits the search query only to stops which are stations.
StopTimetableEntry:
  type: object
  properties:
    Key:
      $ref: '#/definitions/Route'
    Value:
      type: array
      items:
        $ref: '#/definitions/TimetableEntry'
TimetableEntry:
  type: object
  properties:
    ArrivalTime:
      type: string
      format: date-time
```

```
      description: Specifies the arrival time at a specific stop for a specific
                   trip on a route.
    DepartureTime:
      type: string
      format: date-time
      description: Specifies the departure time at a specific stop for a specific
                   trip on a route.
    ShortName:
      type: string
      description: Contains the text that appears in   schedules and sign boards
                   to identify the trip to passengers, for example, to identify
                   train numbers for commuter rail trips. If riders do not commonly
                   rely on trip names, this field will be left blank.
    Headsign:
      type: string
      description: Contains the text that appears on a sign that identifies
                   the trip's destination to passengers. This field is used to
                   distinguish between different patterns of service in
                   the same route.
    Direction:
      type: boolean
      description: Contains a binary value that indicates the direction of travel
                   for a trip. Use this field to distinguish between bi-directional
                   trips on the same route. This field is not used for routing;
                   it provides a way to separate trips by direction when publishing
                   time tables.
TripInfo:
  type: object
  properties:
    Id:
      type: integer
      format: int64
      description: The ID of the trip.
    OriginStop:
      $ref: '#/definitions/TripStop'
    DestinationStop:
      $ref: '#/definitions/TripStop'
    Route:
      $ref: '#/definitions/Route'
    ServiceDetails:
      $ref: '#/definitions/ServiceDetails'
TripSearchFilter:
  type: object
  properties:
    OriginStopIdFilter:
      type: integer
      format: int64
      required: true
      description: The ID of the origin stop.
    DestinationStopIdFilter:
      type: integer
      format: int64
      required: true
      description: The ID of the destination stop.
TripSegmentFilter:
  type: object
  properties:
    TripId:
```

```
      type: integer
      format: int64
      required: true
      min: 1
      description: The ID of the trip for which to display the segment.
    OriginSequenceNumber:
      type: integer
      format: int64
      required: true
      description: The lower bound of the sequence numbers in the trip.
    DestinationSequenceNumber:
      type: integer
      format: int64
      required: true
      description: The upper bound of the sequence numbers in the trip.
TripStop:
  type: object
  properties:
    Stop:
      $ref: '#/definitions/Stop'
    ArrivalTime:
      type: string
      format: date-time
      description: Specifies the arrival time at a specific stop for a specific trip
                   on a route.
    DepartureTime:
      type: string
      format: date-time
      description: Specifies the departure time at a specific stop for a specific trip
                   on a route.
    SequenceNumber:
      type: integer
      format: int64
      description: Identifies the order of the stops for a particular trip. The values
                   must be non-negative integers, and they must increase along the trip.
UserInfo:
  type: object
  properties:
    FullName:
      type: string
      required: true
      description: The full name of the user.
    UserName:
      type: string
      required: true
      description: The username of the user.
    Token:
      type: string
      required: true
      description: Assigned OAuth token used for request authentication.
    Roles:
      type: array
      required: true
      items:
        type: integer
        enum:
          - Employee
          - Administrator
```

```
          description: Contains the roles of the user.
paths:
  # leaving login for now
  /route/filter:
    post:
      description: Searches for routes in the database, using the supplied filtering
                   criteria; at least one of the fields must be non-empty
      consumes:
        - application/json
      produces:
        - application/json
      parameters:
        - name: routeFilter
          in: body
          required: true
          description: Object containing the desired filtering criteria.
          schema:
            $ref: '#/definitions/RouteFilter'
      responses:
        '200':
          description: Array containing routes which meet the supplied criteria.
          schema:
            type: array
            items:
              $ref: "#/definitions/Route"
        '400':
          description: The supplied filter is invalid.
          schema:
            $ref: "#/definitions/ErrorMessage"
  /session:
    get:
      description: Retrieves the user's previous session if it hasn't expired.
      produces:
        - application/json
      parameters:
        - name: Authorization
          in: header
          required: true
          description: OAuth token.
          type: string
      responses:
        '200':
          description: The user's previous session data.
          schema:
            $ref: '#/definitions/UserInfo'
        '401':
          description: The supplied authentication token has expired.
          schema:
            $ref: '#/definitions/ErrorMessage'
  /stop/filter:
    post:
      description: Searches for stops in the database, using the supplied filtering
                   criteria; at least one of the fields must be non-empty
      consumes:
        - application/json
      produces:
        - application/json
      parameters:
```

```
      - name: stopFilter
        in: body
        required: true
        description: Object containing the desired filtering criteria.
        schema:
          $ref: '#/definitions/StopFilter'
    responses:
      '200':
        description: Array containing stops which meet the supplied criteria.
        schema:
          type: array
          items:
            $ref: '#/definitions/Stop'
/timetable/stop/{stopId}:
  get:
    description: Fetches a timetable for the stop with the supplied ID number
    produces:
      - application/json
    parameters:
      - name: stopId
        in: path
        required: true
        description: The ID of the stop for which to fetch the timetable.
        type: integer
    responses:
      '200':
        description: Array containing the timetable data.
        schema:
          type: array
          items:
            $ref: '#/definitions/StopTimetableEntry'
/timetable/route/{routeId}:
  get:
    description: Fetches a timetable for the route with the supplied ID number
    produces:
      - application/json
    parameters:
      - name: routeId
        in: path
        required: true
        description: The ID of the route for which to fetch the timetable.
        type: integer
    responses:
      '200':
        description: Array containing the timetable data.
        schema:
          type: array
          items:
            $ref: '#/definitions/RouteTimetableEntry'
/token:
  post:
    description: Fetches user information along with an authentication token
                 for the desired user.
    consumes:
      - application/json
    produces:
      - application/json
    parameters:
```

```
      - name: loginData
        in: body
        required: true
        description: The object containing the login information.
        schema:
          $ref: '#/definitions/LoginData'
    responses:
      '200':
        description: Returns the user information for the desired user.
        schema:
          $ref: '#/definitions/UserInfo'
      '401':
        description: The supplied credentials were invalid.
        schema:
          $ref: '#/definitions/ErrorMessage'
/trip/mapData:
  post:
    description: Returns a list of map markers representing a part of the desired trip
    consumes:
      - application/json
    produces:
      - application/json
    parameters:
      - name: tripSegmentFilter
        in: body
        required: true
        description: The object containing the query parameters.
        schema:
          $ref: '#/definitions/TripSegmentFilter'
    responses:
      '200':
        description: A list of map markers representing the requested part of the trip.
        schema:
          type: array
          items:
            $ref: '#/definitions/MapMarker'
      '400':
        description: The request sent by the user was invalid.
        schema:
          $ref: '#/definitions/ErrorMessage'
/trip/search:
  post:
    description: Returns a list of trips that run between the two specified stops,
                 along with their arrival/destination times and service
                 availability information
    consumes:
      - application/json
    produces:
      - application/json
    parameters:
      - name: tripSearchFilter
        in: body
        required: true
        description: The object containing the query parameters.
        schema:
          $ref: '#/definitions/TripSearchFilter'
    responses:
      '200':
```

```
        description: A list of trips running through the desired stops.
        schema:
          type: array
          items:
            $ref: '#/definitions/TripInfo'
      '400':
        description: The request sent by the user was invalid.
        schema:
          $ref: "#/definitions/ErrorMessage"
/user/changepassword:
  post:
    description: Allows a user to change their account password
    consumes:
      - application/json
    produces:
      - application/json
    parameters:
      - name: passwordChangeRequest
        in: body
        required: true
        description: Object containing request data.
        schema:
          $ref: '#/definitions/PasswordChangeRequest'
    responses:
      '200':
        description: The user's password was successfully changed.
      '400':
        description: The request sent by the user was invalid.
        schema:
          $ref: "#/definitions/ErrorMessage"
/user/register:
  post:
    description: Registers a user for the application
    consumes:
      - application/json
    produces:
      - application/json
    parameters:
      - name: registrationInfo
        in: body
        required: true
        description: Object storing the registration request data
        schema:
          $ref: '#/definitions/RegistrationInfo'
    responses:
      '200':
        description: The user was registered successfully.
      '400':
        description: The request sent by the user was invalid.
        schema:
          $ref: "#/definitions/ErrorMessage"
/user/favoriteroutes:
  get:
    description: Returns a list of an user's favourited routes
    produces:
      - application/json
    parameters:
      - name: Authorization
```

```
        in: header
        required: true
        description: OAuth token for the desired user
        schema:
          type: string
    responses:
      '200':
        description: A list of the user's favourited routes.
        schema:
          type: array
          items:
            $ref: '#/definitions/Route'
      '401':
        description: Authorization has been denied for this request.
        schema:
          $ref: '#/definitions/ErrorMessage'
  post:
    description: Saves changes of the user's favourite routes
    consumes:
      - application/json
    produces:
      - application/json
    parameters:
      - name: Authorization
        in: header
        required: true
        description: OAuth token for the desired user
        schema:
          type: string
      - name: favouriteInfo
        in: body
        required: true
        description: Information about the changes to be saved
        schema:
          $ref: '#/definitions/FavouriteInfo'
    responses:
      '200':
        description: A list of the user's favourited routes after the update.
        schema:
          type: array
          items:
            $ref: '#/definitions/Stop'
      '401':
        description: Authorization has been denied for this request.
        schema:
          $ref: '#/definitions/ErrorMessage'
/user/favoritestops:
  get:
    description: Returns a list of an user's favourited stops
    produces:
      - application/json
    parameters:
      - name: Authorization
        in: header
        required: true
        description: OAuth token for the desired user
        schema:
          type: string
```

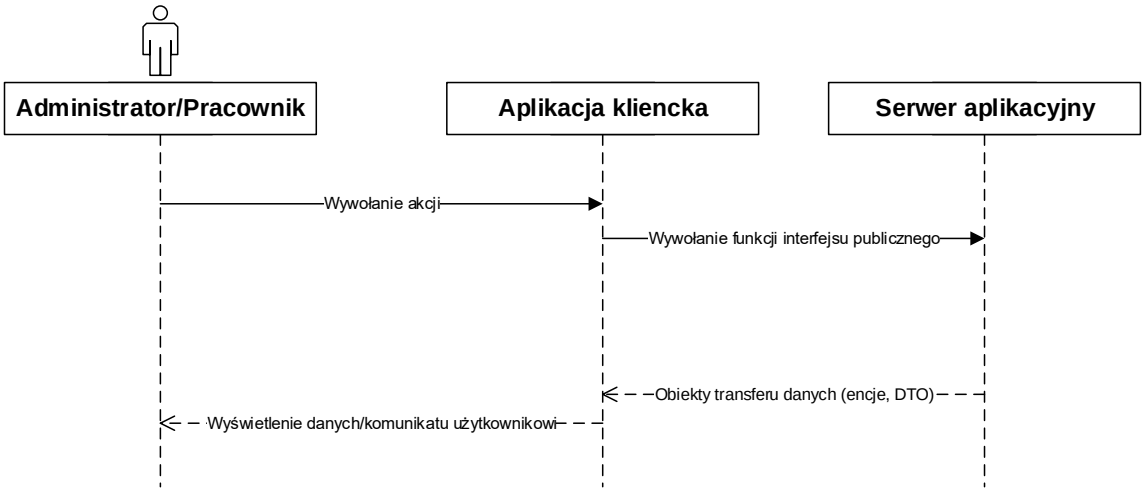
```
  responses:
    '200':
      description: A list of the user's favourited stops.
      schema:
        type: array
        items:
          $ref: '#/definitions/Stop'
    '401':
      description: Authorization has been denied for this request.
      schema:
        $ref: '#/definitions/ErrorMessage'
post:
  description: Saves changes of the user's favourite stops
  consumes:
    - application/json
  produces:
    - application/json
  parameters:
    - name: Authorization
      in: header
      required: true
      description: OAuth token for the desired user
      schema:
        type: string
    - name: favouriteInfo
      in: body
      required: true
      description: Information about the changes to be saved
      schema:
        $ref: '#/definitions/FavouriteInfo'
  responses:
    '200':
      description: A list of the user's favourited stops after the update.
      schema:
        type: array
        items:
          $ref: '#/definitions/Stop'
    '401':
      description: Authorization has been denied for this request.
      schema:
        $ref: '#/definitions/ErrorMessage'
```

3 Dokumentacja końcowa (powykonawcza) – punkty wymagane przez prowadzącego zajęcia

3.1 Diagramy sekwencji

Przebieg komunikacji klienta z relacyjną bazą danych poprzez serwer aplikacyjny jest przedstawiony na poniższych diagramach sekwencji.

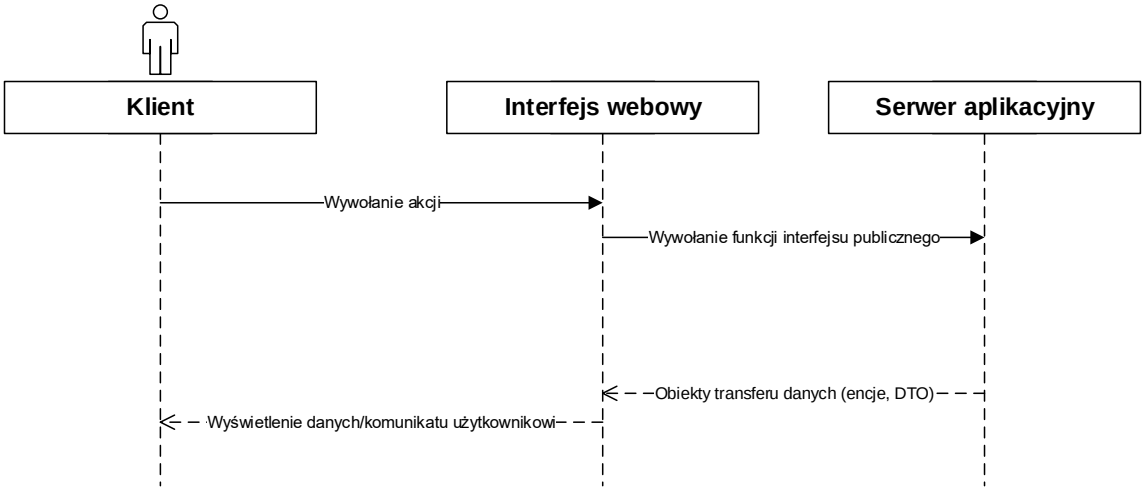
3.1.1 Aplikacja kliencka



Rysunek 29: Diagram sekwencji – komunikacja aplikacji klienckiej z serwerem aplikacyjnym

Funkcjonalność	System	Opis
Wywołanie akcji	Aplikacja kliencka	**Działanie biznesowe:** Administrator lub pracownik wykonują akcję za pośrednictwem interfejsu graficznego aplikacji klienckiej. **Wejście:** Akcja wykonana w interfejsie graficznym. **Wyjście:** Wyświetlenie rezultatu akcji – wyszukiwanych danych lub komunikatu o niepowodzeniu akcji.
Wywołanie funkcji interfejsu	Obiekty serwisowe	**Działanie biznesowe:** Aplikacja kliencka przesyła dane dotyczące akcji podjętej przez użytkownika do warstwy serwisowej za pomocą jej interfejsu publicznego. **Wejście:** Dane dotyczące podjętej akcji. **Wyjście:** Obiekty transferu danych zawierające informacje o rezultacie danej akcji.

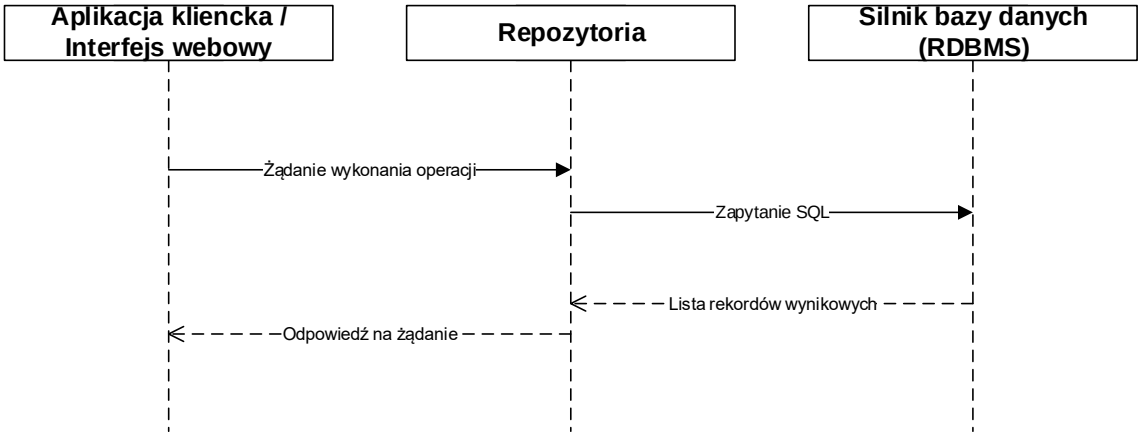
3.1.2 Interfejs webowy



Rysunek 30: Diagram sekwencji – komunikacja interfejsu webowego z serwerem aplikacyjnym

Funkcjonalność	System	Opis
Wywołanie akcji	Interfejs webowy	**Działanie biznesowe:** Klient wykonuje akcję za pośrednictwem interfejsu webowego. **Wejście:** Akcja wykonana w interfejsie webowym. **Wyjście:** Wyświetlenie rezultatu akcji – wyszukiwanych danych lub komunikatu o niepowodzeniu akcji.
Wywołanie funkcji interfejsu	Obiekty serwisowe	**Działanie biznesowe:** Interfejs webowy przesyła dane dotyczące akcji podjętej przez użytkownika do warstwy serwisowej za pomocą jej interfejsu publicznego. **Wejście:** Dane dotyczące podjętej akcji. **Wyjście:** Obiekty transferu danych zawierające informacje o rezultacie danej akcji.

3.1.3 Serwer aplikacyjny

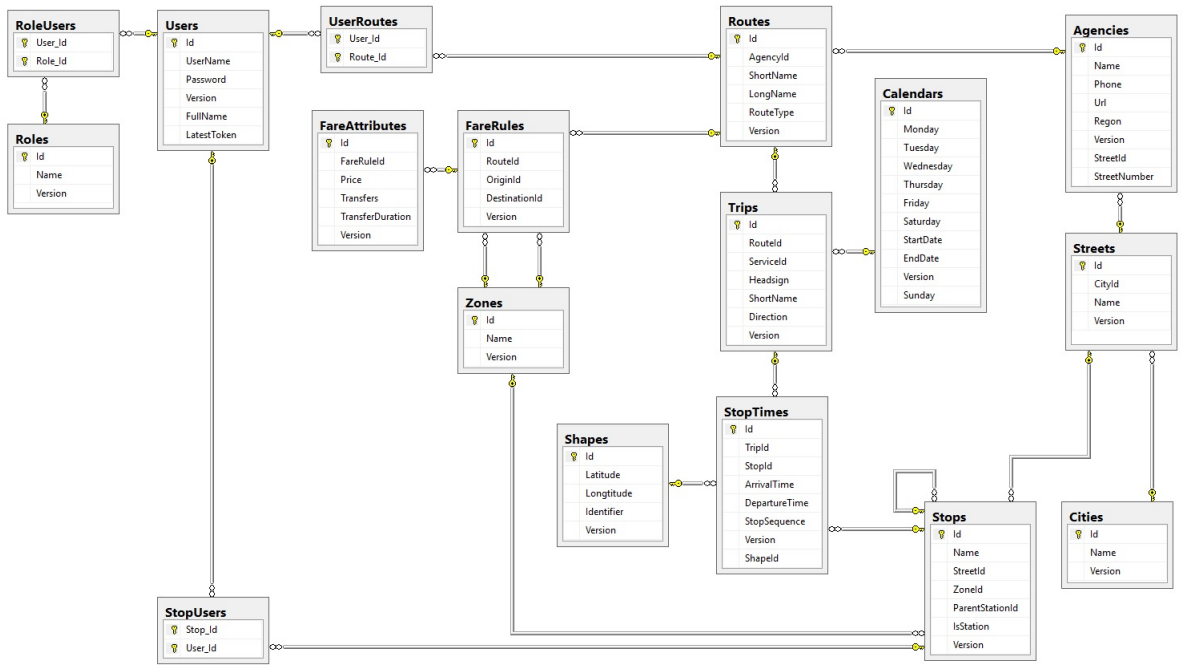


Rysunek 31: Diagram sekwencji – przepływ danych w obrębie serwera aplikacyjnego

Funkcjonalność	System	Opis
Otrzymanie żądania	Serwer aplikacyjny	**Działanie biznesowe:** Serwer aplikacyjny otrzymuje żądanie od jednej z aplikacji klienckich lub interfejsu webowego za pośrednictwem protokołu HTTP. **Wejście:** Żądanie XML SOAP do wykonania. **Wyjście:** Odpowiedź na żądanie aplikacji klienckiej lub interfejsu webowego.
Wywołanie funkcji repozytorium	Obiekty serwisowe	**Działanie biznesowe:** Serwer aplikacyjny przetwarza otrzymane wcześniej żądanie i buduje z niego zapytanie SQL. **Wejście:** Wejściowy obiekt DTO. **Wyjście:** Wynikowy obiekt DTO lub informacja o błędzie (wyjątek).
Zapytanie SQL	Silnik bazy danych	**Działanie biznesowe:** Dane odebrane od klienta są przetwarzane na zapytania SQL do bazy danych. **Wejście:** Zapytanie SQL służące pobraniu lub modyfikacji danych. **Wyjście:** Rekordy wynikowe pobrane z bazy danych.

3.2 Model danych

Model danych użyty w systemie został przedstawiony w formie diagramu relacji na poniższej grafice:



Rysunek 32: Diagram relacji

Poniżej opisano znaczenie i rodzaj relacji zachodzących pomiędzy encjami w systemie.

1. **User - Role** jest relacją wiele do wielu zrealizowaną przy pomocy tabeli pomocniczej UserRoles. Są to tabele niezależne od reszty systemu, ponieważ służą jedynie zdefiniowaniu elementów aplikacji dostępnych dla danego użytkownika.

2. **City - Street** to relacja jeden do wielu. Ulice zdefiniowane w systemie zawierają informację o mieście, w którym są.

3. Informacje o ulicy (a co za tym idzie również o mieście) zawarte są w poszczególnych agencjach (przewoźnikach) oraz przystankach, stąd relacje jeden do wielu **Street - Agency** oraz **Street - Stop**.

4. Każdy przewoźnik zapewnia wiele połączeń różnymi środkami komunikacji, dlatego relacja **Agency - Route** jest relacją jeden do wielu.

5. Poszczególne połączenia są jedynie definicją trasy. Sam przejazd (których może być wiele) pomiędzy punktami trasy zawarty jest w tabeli **Trips**. Przejazd musi zostać ponadto umieszczony w czasie, stąd dodatkowa tabela **Calendars**, która mówi w jakich dniach połączenie będzie funkcjonować. Tak określone relacje **Route - Trip** oraz **Calendar - Trip** są jeden do wielu.

6. Należy również określić konkretne czasy postojów na trasie przejazdu. Tym zajmuje się tabela **StopTimes**, w której zdefiniowane poszczególne postoje są skojarzone z konkretnym przejazdem i przystankiem. To powoduje, że relacje **Trip - StopTime** oraz **Stop - StopTime** są jeden do wielu.

7. Należy równiez zdefiniować strefy przejazdu, czym zajmuje się tabela **Zones**. Każdy przystanek ma przypisaną konkretną strefę w której się znajduje, więc relacja **Zone - Stop** jest jeden do wielu.

8. Każda trasa może mieć inny cennik, a cena biletu może się zmieniać w zależności od początkowej i końcowej strefy. Stąd relacja **Route - FareAttribute** jest jeden do wielu. Ponadto wiele tras może korzystać z jednego cennika, dlatego relacja **FareRule - FareAttribute** jest również jeden do wielu.

9. Z tabelą **StopTimes** powiązana jest tabela **Shapes**, która zawiera dane określające położenie geograficzne dla poszczególnych postojów. Dane te wykorzystywane są do generowania map.
10. Tabele **UserRoutes** i **StopUsers** odpowiadają za przechowywanie ulubionych tras i przystanków użytkowników.

3.3 Scenariusze testów akceptacyjnych i raport z ich przeprowadzenia

3.3.1 Aplikacja wewnętrzna

Rola	Testowana funkcjonalność	Opis czynności	Oczekiwany rezultat	Wynik testu
Administrator	Dodanie nowego użytkownika	Wybranie zakładki **Users** na bocznym pasku aplikacji, przejście do widoku dodawania użytkownika po wciśnięciu przycisku **Add New** oraz zapisanie zmian przyciskiem **Save**.	Nowy użytkownik zostanie dodany do bazy danych.	Pozytywny
Administrator	Modyfikacja istniejącego użytkownika	Wybranie zakładki **Users** na bocznym pasku aplikacji, wyszukanie istniejącego użytkownika, przejście do widoku edycji użytkownika po zaznaczeniu jednego z listy i wciśnięciu przycisku **Edit Selected** oraz zapisanie zmian przyciskiem **Save**.	Dane wybranego użytkownika zostaną zmodyfikowane.	Pozytywny
Administrator	Usunięcie istniejącego użytkownika	Wybranie zakładki **Users** na bocznym pasku aplikacji, wyszukanie istniejącego użytkownika i wciśnięcie przycisku **Delete Selected** po wybraniu jednego z listy.	Dane wybranego użytkownika zostaną usunięte.	Pozytywny
Administrator	Usunięcie istniejącego użytkownika	Wybranie zakładki **Users** na bocznym pasku aplikacji, wyszukanie istniejącego użytkownika i wciśnięcie przycisku **Delete Selected** po wybraniu jednego z listy.	Dane wybranego użytkownika zostaną usunięte.	Pozytywny

Użytkownik	Zalogowanie się	Wpisanie loginu i hasła w odpowiednie pola na ekranie logowania i wciśnięcie przycisku **Login**	Zalogowanie się do systemu w przypadku poprawnych danych, odmowa dostępu w przypadku niepoprawych danych	Pozytywny
Zalogowany użytkownik	Wylogowanie się	Wciśniecię przycisku **Logout** na bocznym pasku aplikacji.	Poprawne wylogowanie się z zystemu i przejście do ekranu logowania.	Pozytywny
Pracownik	Dodanie nowego przewoźnika	Wybranie zakładki **Agencies** w bocznym pasku aplikacji, przejście do widoku dodania przewoźnika po wciśnięciu przycisku **Add New** oraz zapisanie zmian przyciskiem **Save**.	Nowy przewoźnik zostanie dodany do bazy danych.	Pozytywny
Pracownik	Modyfiacja istniejącego przewoźnika	Wybranie zakładki **Agencies** na bocznym pasku aplikacji, wyszukanie istniejącego przewoźnika, przejście do widoku edycji przewoźnika po zaznaczeniu jednego z listy i wciśnięciu przycisku **Edit Selected** oraz zapisanie zmian przyciskiem **Save**.	Dane wybranego przewoźnika zostaną zmodyfikowane.	Pozytywny
Pracownik	Usunięcie istniejącego przewoźnika	Wybranie zakładki **Agencies** na bocznym pasku aplikacji, wyszukanie istniejącego przewoźnika i wciśnięcie przycisku **Delete Selected** po wybraniu jednego z listy.	Dane wybranego przewoźnika zostaną usunięte.	Pozytywny

Pracownik	Dodanie nowej linii połączeń	Wybranie zakładki **Routes** w bocznym pasku aplikacji, przejście do widoku dodania połączenia po wciśnięciu przycisku **Add New** oraz zapisanie zmian przyciskiem **Save**.	Nowe połączenie zostanie dodane do bazy danych.	Pozytywny
Pracownik	Modyfikacja istniejącej linii połączeń	Wybranie zakładki **Routes** na bocznym pasku aplikacji, wyszukanie istniejącego połączenia, przejście do widoku edycji połączenia po zaznaczeniu jednego z listy i wciśnięciu przyciskiem **Edit Selected** oraz zapisanie zmian przyciskiem **Save**.	Dane dotyczące wybranego połączenia zostaną zmodyfikowane.	Pozytywny
Pracownik	Usunięcie istniejącej linii połączeń	Wybranie zakładki **Routes** na bocznym pasku aplikacji, wyszukanie istniejącego połączenia i wciśnięcie przycisku **Delete Selected** po wybraniu jednego z listy.	Dane dotyczące wybranego połączenia zostaną usunięte.	Pozytywny
Pracownik	Wyświetlanie połączeń przewoźnika	Wybranie zakładki **Routes** na bocznym pasku aplikacji i wybranie konkretnego przewoźnika w opcjach filtrowania.	Wyświetlona lista połączeń zapewnionych przez wybranego przewoźnika.	Pozytywny

Pracownik	Wyświetlanie rozkładu jazdy dla linii połączeń	Wybranie zakładki **Routes** na bocznym pasku aplikacji, wyszukanie istniejącego połączenia i wciśnięcie przycisku **View Timetable** po zaznaczeniu jednego z listy. Następnie przełączając się w liście przystanków po lewej możemy przeglądać godziny przyjazdu i odjazdu na ten przystanek dla danej linii.	Informacje o godzinach przyjazdu i odjazdu danej linii na konkretne przystanki.	Pozytywny
Pracownik	Wyświetlanie rozkładu jazdy dla przystanku	Wybranie zakładki **Stops** na bocznym pasku aplikacji, wyszukanie istniejącego przystanku i wciśnięcie przycisku **View Timetable** po zaznaczeniu jednego z listy. Następnie przełączając się w liście połączeń po lewej możemy przeglądać godziny przyjazdu i odjazdu na ten przystanek dla danej linii.	Informacje o godzinach przyjazdu i odjazdu poszczególnych linii połączeń na danym przystanku.	Pozytywny
Pracownik	Wyszukiwanie połączeń pomiędzy dwoma przystankami	Wybranie zakładki **Search Routes** na bocznym pasku aplikacji, wybranie dwóch przystanków i wciśnięcie przycisku **Find Routes**.	Linie połączeń, którymi można przejechać pomiędzy podanymi przystankami.	Pozytywny

3.3.2 Aplikacja webowa

Testowana funkcjonalność	**Opis czynności**	**Oczekiwany rezultat**	**Wynik**
Niezalogowany użytkownik			
Wyszukanie trasy	Otwarcie witryny w przeglądarce, wpisanie nazwy i wybranie dwóch przystanków i wciśnięcie przycisku **Search**.	Wyświetlenie na dole strony listy tras kursujących między wybranymi przystankami.	Pozytywny

Wyświetlenie mapy dla połączenia	Otwarcie witryny w przeglądarce, wpisanie nazwy, wybranie dwóch przystanków i wciśnięcie przycisku **Search** i wciśnięcie przycisku **Show map** dla jednego z wyników wyszukiwania.	Okno dialogowe pokazujące na mapie trasę wybranego połączenia.	Pozytywny
Wyświetlanie rozkładu dla trasy	Wybranie opcji **Timetable** $\rightarrow$ **Search by route**, wyszukanie i wybranie trasy oraz przystanku.	Pojawienie się listy rozwijanej z przystankami trasy, pojawienie się rozkładu po wybraniu przystanku.	Pozytywny
Wyświetlanie rozkładu dla trasy	Wybranie opcji **Timetable** $\rightarrow$ **Search by stop**, wyszukanie i wybranie przystanku oraz trasy.	Pojawienie się listy rozwijanej z trasami przejeżdżającymi przez dany przystanek, pojawienie się rozkładu po wybraniu trasy.	Pozytywny
Rejestracja konta	Wybranie opcji **Register**, wypełnienie formularza, wciśnięcie przycisku **Register**.	Komunikat o pomyślnym utworzeniu konta.	Pozytywny
Logowanie	Wybranie opcji **Login**, wprowadzenie nazwy użytkownika oraz hasła, wciśnięcie przycisku **Login**.	Komunikat o pomyślnym zalogowaniu do aplikacji.	Pozytywny
Zalogowany użytkownik			
Dodanie, usunięcie ulubionego przystanku	Wybranie opcji **Favourites**, usunięcie jednego oraz dodanie innego przystanku w zakładce **Stops**, kliknięcie przycisku **Save changes**.	Komunikat o zapisaniu zmian w ulubionych.	Pozytywny
Dodanie, usunięcie ulubionej trasy	Wybranie opcji **Favourites**, usunięcie jednej oraz dodanie innej trasy w zakładce **Routes**, kliknięcie przycisku **Save changes**.	Komunikat o zapisaniu zmian w ulubionych.	Pozytywny
Zmiana hasła	Wybranie opcji **Settings** z menu rozwijanego w prawym górnym rogu etykietowanego nazwą użytkownika, wypełnienie formularza zmiany hasła, wciśnięcie przycisku **Save changes**, wylogowanie się i zalogowanie z nowym hasłem.	Pomyślne zalogowanie się do systemu.	Pozytywny
Wylogowanie	Wybranie opcji **Logout** z menu rozwijanego w prawym górnym rogu etykietowanego nazwą użytkownika.	Powiadomienie o wylogowaniu się z aplikacja, zniknięcie opcji **Favourites** i menu rozwijanego z paska nawigacyjnego.	Pozytywny

4 Lista użytych skrótów

BSD Berkeley Software Distribution

IoC Inversion of Control

MIT Massachussetts Institute of Technology

MS-PL Microsoft Software Public License

MVVM **ang.** Model-View-ViewModel – wzorzec używany w projektach realizowanych w technologii WPF pozwalający na odseparowanie logiki aplikacji od warstwy prezentacyjnej.

WPF Windows Presentation Framework

5 Bibliografia

[1] Castle Project, *Castle Core*, `https://github.com/castleproject/Core`

[2] ASP.NET, *Entity Framework 6*, `https://github.com/aspnet/EntityFramework6`

[3] Dennis Doomen, *FluentAssertions*, `https://github.com/dennisdoomen/fluentassertions`

[4] Moq, *Moq 4*, `https://github.com/moq/moq4`

[5] NUnit, *NUnit*, `https://github.com/nunit/nunit`

[6] ReactiveUI, *ReactiveUI*, `https://github.com/reactiveui/ReactiveUI`

[7] Reactive Extensions, *Rx.NET*, `https://github.com/Reactive-Extensions/Rx.NET`

[8] Paul Betts, *Splat*, `https://github.com/paulcbetts/splat`

[9] MahApps, *Metro*, `https://mahapps.com`

[10] AngularJS, Google, `https://angularjs.org/`.

[11] angular-spinner, Uri Shaked, `https://github.com/urish/angular-spinner`.

[12] spin.js, Felix Gnass, `http://spin.js.org/`.

[13] angular-toastr, Jesús Rodríguez, `https://github.com/Foxandxss/angular-toastr`.

[14] angular-event-aggregator, Vladimir Gurovich, `https://github.com/vladgurovich/angular-event-aggregator`.

[15] Angular UI Router, Angular UI Team, `https://github.com/angular-ui/ui-router`.

[16] Angular UI Bootstrap, Angular UI Team, `https://github.com/angular-ui/bootstrap`.

[17] Font Awesome, Angular UI Team, `http://fontawesome.io/`.

[18] Bootstrap, Dave Gandy, `http://getbootstrap.com`.

[19] OAuth 2.0, OWIN, `https://www.asp.net/aspnet/overview/owin-and-katana/owin-oauth-20-authorization-server`.

[20] Unity, Microsoft, `https://msdn.microsoft.com/en-us/library/dn507457(v=pandp.30).aspx`.

[21] AngularJS Google Maps, Allen Kim, `https://github.com/allenhwkim/angularjs-google-maps`.

[22] jQuery, The jQuery Foundation, `http://jquery.com/`.

[23] Installing SQL Server 2014 Step By Step Tutorial, Microsoft TechNet, http://social.technet.microsoft.com/wiki/contents/articles/23878.installing-sql-server-2014-step-by-step-tutorial.aspx.

[24] Installing IIS 8 on Windows Server 2012, Microsoft, https://www.iis.net/learn/get-started/whats-new-in-iis-8/installing-iis-8-on-windows-server-2012.

[25] IIS 8.0 ASP.NET Configuration Management, Microsoft, https://www.iis.net/learn/get-started/whats-new-in-iis-8/iis-80-aspnet-configuration-management.