

Politechnika Wrocławска
Wydział Elektroniki
Katedra Automatyki, Mechatroniki i Systemów Sterowania

OPTYMALIZACJA DYSKRETNYCH PROCESÓW PRODUKCYJNYCH

Projekt

Autorzy:
SEBASTIAN KOWALEWSKI, 184145
BARTŁOMIEJ KOZDRAŚ, 184095
KAROL RUNGO, 184128

Prowadzący:
DR INŻ. JAROSŁAW PEMPERA

Termin:
WT TN 11¹⁵-13⁰⁰

Spis treści

1 Wstęp	2
1.1 Przedstawienie problemu	2
1.2 Problem komiwojażera	2
2 Zastosowane narzędzia, oprogramowanie i algorytmy	4
2.1 Microsoft Visual Studio 2013	4
2.2 Język C Sharp i platforma .NET	4
2.3 SQL Server Management Studio 2008	5
2.4 Technologia LINQ	5
2.5 Microsoft Blend	6
2.6 GMap.NET	6
2.7 Algorytm	6
2.7.1 Przedstawienie problemu na grafach	6
2.7.2 Algorytm rozwiązujejący problem komiwojażera	8
3 Opis aplikacji	10
3.1 Baza danych	10
3.2 Okna aplikacji	11
4 Prezentacja działania programu	14
5 Zakończenie	24
Bibliografia	25
Spis rysunków	26

Rozdział 1

Wstęp

1.1 Przedstawienie problemu

Przedstawiony w projekcie problem dotyczy serwisanta technicznego pracującego w firmie telekomunikacyjnej, do którego obowiązków należy serwis stacji bazowych (ang. *Base Transceiver Station*, BTS), które odgrywają podstawową rolę w systemach łączności. Głównym zagadnieniem jest wybór konkretnych miejscowości, w których znajdują się BTSy, a następnie odwiedzenie ich w takiej kolejności, aby zminimalizować czas lub koszt podróży, ewentualnie odległość pomiędzy miastami. W zagadnieniach optymalizacyjnych problem taki jest znany jako problem komiwojażera.

Projekt zakłada stworzenie aplikacji graficznej, do której dostęp będę mieli serwisanci oraz administratorzy. Administrator może edytować bazę danych poprzez dodanie lub usuwanie stacji bazowych oraz serwisantów. Serwisant korzysta z aplikacji wybierając miasta, które musi odwiedzić. Aplikacja na podstawie wskazanych miast wylicza najbardziej korzystną drogę oraz przedstawia jej przebieg na mapie.

1.2 Problem komiwojażera

Problem komiwojażera (ang. *Travelling salesman problem*) w dobry sposób opisuje następujące pytanie: znając listę miast i odległość pomiędzy każdą parą miast, jaka jest najkrótsza droga, aby odwiedzić dokładnie każde z nich jeden raz, a następnie wrócić do miasta początkowego? TSP jest problemem NP-trudnym stosowanym w optymalizacji kombinatorycznej, a także ważnym zagadnieniem w badaniach operacyjnych i algorytmice [3].

Nie ulega wątpliwości, że problem komiwojażera w bardzo dobry sposób ilustruje przedstawiony problem serwisanta, który musi w ramach swojej pracy odwiedzić wskazane stacje bazowe. W ten sposób w niniejszym projekcie zaprezentowano rzeczywisty problem, który może zostać rozwiązany za pomocą zagadnienia szeroko znanego w nauce.

Najczęściej spotykana wersją problemu komiwojażera jest minimalizacja przebytej drogi. Oczywiście długość drogi nie jest jedynym dopuszczalnym parametrem. Można założyć, że znana jest cena lub czas podróży. W takim przypadku rozwiązanie problemu polega na znalezieniu najbliższej bądź najszybszej drogi, który musi przebyć komiwojażer.

Problem komiwojażera często przedstawiany jest za pomocą grafów. Okazuje się, że teoria grafów dostarcza rozwiązań, które w dobry sposób reprezentują omawiane zagadnienie. Wierzchołki grafów mogą pełnić rolę miast, natomiast krawędzie

są połączeniami pomiędzy parą konkretnych miejscowości. Waga przypisana każdej z krawędzi może opisywać parametr (odległość, koszt lub czas) charakteryzujący połączenie pomiędzy miastami.

TSP jest problemem NP-trudnym. Oznacza to, że istniejące algorytmy rozwiązujące ten problem mają bardzo niekorzystną, wykładniczą złożoność obliczeniową. Dokładne algorytmy działają dla bardzo małej liczby miast, najlepiej w sytuacji gdy graf, nie jest grafem zupełnym. Istnieje jednak spora grupa algorytmów przybliżonych. W takich przypadkach nie otrzymuje się rozwiązania optymalnego, ale często uzyskany przybliżony rezultat jest zadowalający.

Rozdział 2

Zastosowane narzędzia, oprogramowanie i algorytmy

2.1 Microsoft Visual Studio 2013

Microsoft Visual Studio 2013 jest zintegrowanym środowiskiem programistycznym (ang. *Integrated Development Environment*, IDE), które umożliwia programistom tworzenie samodzielnych aplikacji. Produkt ten wspiera platformę .NET i język C Sharp, dlatego też został zastosowany w ramach realizowanego projektu. Ponadto możliwe jest tworzenie aplikacji dla systemów desktopowych — klasycznych, jak i wykorzystujących interfejs Modern UI¹. Co istotne zapewnia nowoczesne i zaawansowane elementy interfejsu WPF², co wydaje się być szczególnie istotne przy zastosowaniu Microsoft Blend.

Vicsual Studio zawiera edytor kodu, który wspiera IntelliSense³ oraz mechanizmy refaktoryzacji kodu. Zintegrowany ze środowiskiem debugger, pozwala programistom na badanie przebiegu wykonywania kodu i obserwację dowolnej używanej zmiennej. Poprzez udostępnienie designera do tworzenia aplikacji Windows Forms, skonstruowanie odpowiedniego interfejsu graficznego jest intuicyjne i relatywnie proste. Zastosowanie ma tutaj metoda "przeciagnij–upuść", co znacznie skraca czas tworzenia oprogramowania.

Istotną rolę odgrywa również fakt, że programista otrzymuje dużą swobodę podczas tworzenia oprogramowania z wykorzystaniem omawianego IDE. Visual Studio praktycznie na każdym poziomie aplikacji można rozszerzać za pomocą dodatków, a kod można wzbogacać o dołączone biblioteki.

2.2 Język C Sharp i platforma .NET

C Sharp (C#), to obiektowy język programowania zaprojektowany dla firmy Microsoft. Łączy w sobie najlepsze cechy takich języków jak C++ czy Java. Co istotne program napisany w tym języku kompilowany jest do specjalnego kodu

¹Modern UI (ang. *Modern User Interface*), to wcześniejsza wersja Microsoft design language, co jest wewnętrzną nazwą kodową dla języka projektowania firmy Microsoft. Interfejs Modern UI jest wykorzystywany, np. w Windowsie 8 czy Windows Phone.

²WPF (ang. *Windows Presentation Foundation*), to nazwa silnika graficznego i API bazującego na .NET.

³IntelliSense jest formą automatycznego uzupełniania zawartą w niektórych środowiskach programistycznych, co pozwala skrócić proces pisania kodu, redukując pomyłki stylistyczne, literówki, etc.

pośredniego wykonywanego w środowisku uruchomieniowym, jakim jest, np. platforma .NET. Nie jest możliwe wykonanie skompilowanego programu przez system operacyjny nieposiadający takiego środowiska.

Wśród głównych cech języka warto wymienić:

- obiektowość,
- Garbage Collector — mechanizm zajmujący się gospodarowaniem pamięcią, zwalniający programistę z obowiązku zwalniania zaalokowanej pamięci,
- type ogólne (generyczne) — mechanizm zbliżony do szablonów w C++,
- obsługa zdarzeń i delegatów.

Platforma programistyczna .NET Framework została opracowana i udostępniona w 2002 r. przez firmę Microsoft. Obejmuje środowisko uruchomieniowe oraz biblioteki klas, które służą dostarczeniu aplikacjom odpowiednich funkcjonalności. Głównym zadaniem platformy jest zarządzanie różnymi elementami systemu, m. in. kodem i/lub pamięcią. Platforma .NET nie jest ściśle związana z jednym językiem programowania. Jest wspierana, np. przez C++, C Sharp czy Visual Basic. Co ciekawe w 2014 r. Microsoft zapowiedział udostępnienie .NET na zasadach Open Source (licencja MIT).

Język C Sharp w połączeniu z platformą .NET umożliwia proste tworzenie rozbudowanych i wydajnych aplikacji. Bogate biblioteki klas umożliwiają tworzenie programów z graficznym interfejsem użytkownika (GUI), konsolowych, bazodanowych czy internetowych.

2.3 SQL Server Management Studio 2008

SQL Server Management Studio, to rozbudowana aplikacja, służąca do zarządzania bazami danych Microsoft. Narzędzie wyposażone jest w prosty, graficzny interfejs i posiada wiele użytecznych kreatorów ułatwiających pracę z bazą danych, m. in.: konfigurację oraz administrowanieinstancjami SQL Server.

Korzystanie z omawianego oprogramowania zapewnia szereg funkcjonalności do tworzenia, edytowania i zarządzania bazą danych. Użyta w projekcie wersja SQL Server Management Studio 2008 Express charakteryzuje się darmową licencją.

2.4 Technologia LINQ

LINQ (ang. *Language INtegrated Query*), to część technologii platformy dot NET. Jest technologią zapatań związanych z językiem programowania. Zapewnia wygodny dostęp do danych. Jako, że stanowi warstwę abstrakcji nad różnymi źródłami danych, może obsługiwać:

- bazę danych SQL,
- dokument XML,
- zwykły obiekt (pod warunkiem, że implementuje interfejs `IEnumerable<T>`).

Jako swój wynik zapytanie LINQ zwraca „kolejkę” obiektów typu ogólnego. Sama technologia oferuje wiele operatorów i funkcji do tworzenia bardzo skomplikowanych zapytań. Składnia języka jest relatywnie prosta i bardzo przypomina składnię znaną z SQL.

2.5 Microsoft Blend

Microsoft Blend⁴, to narzędzie opracowane przez firmę Microsoft służące do tworzenia zaawansowanych graficznych interfejsów aplikacji internetowych, a także desktopowych. Jest edytorem opartym na XAML (ang. *eXtensible Application Markup Language*)⁵. Z jego pomocą projektuje się interfejsy dla WPF oraz Microsoft Silverlight⁶. Do działania wymaga koniecznie platformy .NET.

W 2012 r., wraz z wydaniem Microsoft Windows 8, zmieniono nazwę na "Blend for Visual Studio". Od tego momentu program wydawany jest wraz z oprogramowaniem Microsoft Visual Studio. Należy jednak zaznaczyć, że nie jest podstawowym wyposażeniem wszystkich wersji wspomnianego IDE.

2.6 GMap.NET

GMap.NET, to wieloplatformowe, darmowe, opensource'owe narzędzie działające pod kontrolą .NET. Umożliwia wykonywanie wielu operacji na mapach:

- Google,
- Yahoo!,
- Bing,
- OviMap,
- WikiMapia,
- i innych [1].

Wśród zapewnionych funkcji możliwe jest rysowanie drogi oraz znalezienie lokacji za pomocą współrzędnych geograficznych lub podania jej nazwy. Narzędzie, to odgrywa ważną rolę w projekcie, gdyż pozwala na graficzne przedstawienie trasy, którą musi pokonać serwisant.

2.7 Algorytm

2.7.1 Przedstawienie problemu na grafach

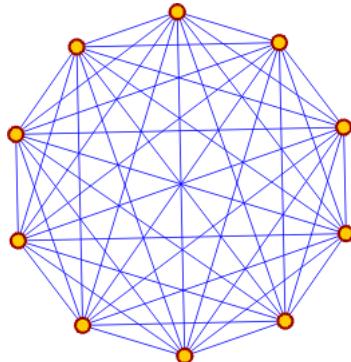
Problem komiwojażera został przybliżony w pierwszym rozdziale. W terminologii grafów należy dokonać założeń, że miasta są wierzchołkami grafu, natomiast trasy pomiędzy nimi — krawędziami. Waga krawędzi może być liczbą opisującą różne pojęcia, w zależności od tego, co jest celem optymalizacji. Może odpowiadać, np.: czasowi podróży, kosztom przejazdu, odległości pomiędzy miastami. Trasa komiwojażera jest cyklem, który przechodzi przez każdy wierzchołek grafu dokładnie jeden raz (z wyjątkiem pierwszego — tutaj cykl się zaczyna i kończy). Taki cykl określa się mianem cyklu hamiltonowskiego.

⁴Wcześniej znany jako Microsoft Expression Blend.

⁵XAML jest językiem opisu interfejsu użytkownika wykorzystywanym m. in. w technologii Windows Presentation Foundation.

⁶Microsoft Silverlight, to technologia internetowa, dzięki której możliwe jest wyświetlenie treści multimedialnych używając do tego przeglądarki internetowej.

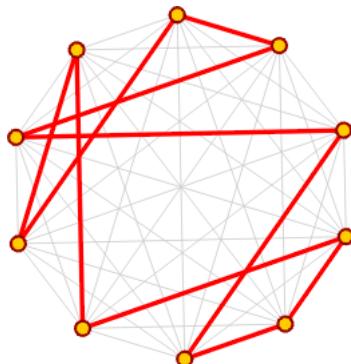
Znalezienie właściwego cyklu Hamiltona metodą przeglądu zupełnego jest zadaniem trudnym obliczeniowo. Na rys. 2.1 przedstawiono graf zupełny (każdy wierzchołek połączony z każdym) o dziesięciu wierzchołkach. Liczba różnych cykli hamiltonowskich jest równa $(n - 1)!$, gdzie n odpowiada liczbie wierzchołków, co prowadzi do złożoności obliczeniowej $O(n!)$.



Rysunek 2.1: Graf zupełny dla dziesięciu wierzchołków

Źródło: http://edu.i-lo.tarnow.pl/inf/alg/001_search/0140.php

Przykładowy cykl hamiltonowski zaprezentowano na rys. 2.2. Dla każdego znalezionej cyklu należałyby policzyć sumę wag i zapamiętać cykl, dla którego suma ta jest najmniejsza. Grafy, które odzwierciedlają rzeczywiste połączenia pomiędzy miastami nie są zupełnie, gdyż nieekonomiczne byłoby budowanie osobnych dróg pomiędzy każdą parą miast. Nie zmienia to jednak faktu, że problem nadal posiada klasę złożoności obliczeniowej $O(n!)$.



Rysunek 2.2: Przykładowy cykl Hamiltona w grafie o dziesięciu wierzchołkach

Źródło: http://edu.i-lo.tarnow.pl/inf/alg/001_search/0140.php

Istnieją algorytmy zwracające rozwiązanie przybliżone w czasie wielomianowym, lecz zazwyczaj są one bardzo zaawansowane. Przy założeniu jednak, że liczba krawędzi nie jest duża, co wydaje się być bardzo sensowne, gdyż serwisant raczej nie odwiedzi dużej liczby miast w krótkim okresie czasu, a graf nie jest grafem zupełnie, to rozwiązanie problemu komiwojażera można uzyskać prostym algorytmem. Algorytm ten polega na wyznaczeniu wszystkich cykłów Hamiltona, policzeniu sumy wag krawędzi i zwróceniu cyklu o najmniejszej sumie wag. Co najważniejsze otrzymany w ten sposób wynik jest optymalny, a nie przybliżony.

2.7.2 Algorytm rozwiązujący problem komiwojażera

Zastosowany algorytm rozwiązuje problem komiwojażera dla małych grafów. Uwzględniając jednak zagadnienie znalezienie najkrótszej drogi pomiędzy stacjami bazowymi przez serwisanta wydaje się być to sensowne, ponieważ pracownik nie jest w stanie odwiedzić zbyt dużej liczby BTSów w krótkim czasie.

Zaproponowany pomysł opiera się na rekurencyjnej procedurze [2]:

```
TSP(n, graph, v, vo, d, dH, S, SH, visited),
```

gdzie:

- **n** — liczba wierzchołków w grafie,
- **graph** — zadany w dowolnie wybrany sposób. Definicja grafu powinna udostępniać wagę krawędzi (w projekcie zrealizowany poprzez macierz sąsiedztwa),
- **v** — wierzchołek bieżący,
- **vo** — wierzchołek startowy,
- **d** — suma wag krawędzi cyklu hamiltonowskiego,
- **dH** — pomocnicza suma wag krawędzi,
- **S** — stos wierzchołków,
- **SH** — pomocniczy stos wierzchołków,
- **visited** — n-elementowa tablica logiczna odwiedzin.

Wyjściem procedury jest **S** — stos zawierający numery kolejnych wierzchołków cyklu Hamiltona o najmniejszej sumie wag krawędzi (stos pusty, gdy cykl Hamiltona nie istnieje) oraz **d** — suma wag krawędzi cyklu hamiltonowskiego. Wyróżnia się jeszcze wierzchołek pomocniczy **u**.

Lista kroków algorytmu głównego prezentuje się następująco:

- 1 Utwórz i wyzeruj **visited**, **S**, **SH**.
- 2 $d \leftarrow \infty$, $dH \leftarrow 0$ (początkowa suma największa z możliwych).
- 3 **TSP(vo)** (wyszukiwanie cyklu Hamiltona od wierzchołka startowego).
- 4 Jeśli **S.empty() = false**, to pisz **S** i **d**, inaczej pisz "Brak cyklu Hamiltona".
- 5 Zakończ

Natomiast lista kroków procedury rekurencyjnej jest znacznie dłuższa i została przedstawiona poniżej:

- 1 **SH.push(v)** (odwiedzony wierzchołek dopisany do ścieżki).
- 2 Jeśli **SH** nie zawiera **n** wierzchołków, to idź do 10 (brak ścieżki Hamiltona).
- 3 Jeśli nie istnieje krawędź z **v** do **vo**, to idź do 17 (brak cyklu Hamiltona).
- 4 $dH \leftarrow dH + \text{waga krawędzi z } v \text{ do } vo$ (uwzględnienie w sumie ostatniej wagi).

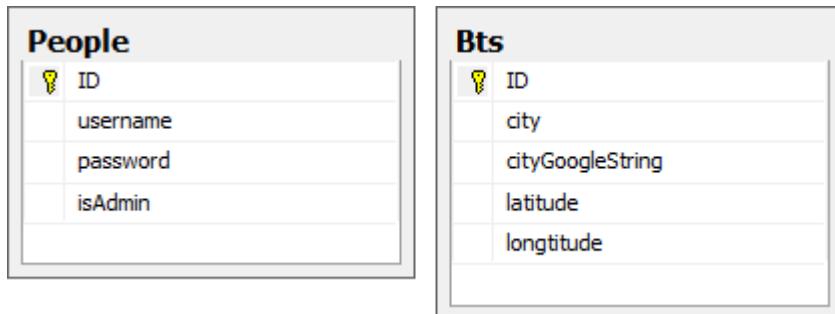
- 5 Jeśli $dH \geq d$, to idź do 8 (znaleziony cykl jest gorszy od bieżącego).
- 6 $d \leftarrow dH$ (zapamiętanie sumy wag cyklu).
- 7 Skopiuj stos SH do stosu S (zapamiętanie cyklu Hamiltona).
- 8 $dH \leftarrow dH -$ waga krawędzi z v do vo (usunięcie ostatniej wagi z sumy).
- 9 Idź do 17.
- 10 `visisted(v) ← true` (wierzchołek zaznaczony jako odwiedzony).
- 11 Dla każdego sąsiada u wierzchołka v wykonuj 12...15 (lista sąsiedztwa).
- 12 Jeśli `visisted(v) = true`, to przejdź do 11 (omijanie wierzchołków odwiedzonych).
- 13 $dH \leftarrow dH +$ waga krawędzi z v do u (nowa suma wag krawędzi cyklu).
- 14 `TSP(n, graph, u, vo, d, dH, S, SH, visisted)` (rekurencyjne poszukiwanie cyklu).
- 15 $dH \leftarrow dH -$ waga krawędzi z v do u (usunięcie wagi krawędzi z sumy).
- 16 `visisted(v) ← false` (zwolnienie bieżącego wierzchołka).
- 17 `SH.pop()` (usunięcie bieżącego wierzchołka ze ścieżki).
- 18 Zakończ.

Rozdział 3

Opis aplikacji

3.1 Baza danych

Baza danych została stworzona w środowisku SQL Server Management Studio 2008, które opisane została we wcześniejszym rozdziale. Poszczególne tabele, które są użyte w zaprojektowanej aplikacji, zostały przedstawiono na rys. 3.1.



Rysunek 3.1: Tabele w bazie danych

Źródło: opracowanie własne

Tabela *People* zawiera loginy i hasła wszystkich użytkowników korzystających z aplikacji. Podczas logowania do programu, następuje sprawdzenie czy w bazie danych istnieje użytkownik o podanym loginie i haśmie przy pomocy zapytania zrealizowanego w technologii LINQ. Tabela nie jest powiązana relacją z tabelą *Bts*. Dodanie nowego użytkownika do systemu możliwe jest tylko z poziomu administratora. Za sprawdzenie czy zalogowany użytkownik posiada kompetencje pozwalające na zarządzanie bazą odpowiada zmienna bitowa *isAdmin*. Działanie takie ma na celu zabezpieczenie przed umyślnym dodaniem nowego użytkownika bez wiedzy administratora.

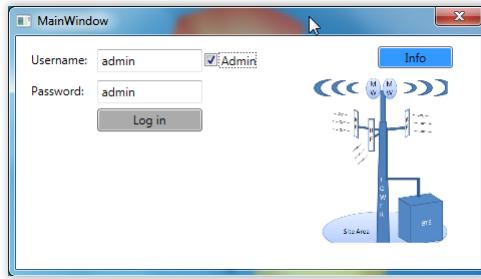
Bts jest tabelą, która zawiera lokalizacje konkretnych stacji bazowych. Nie jest powiązana żadną relacją z tabelą *People*. BTS jest identyfikowany poprzez przypisane mu miasto, długość oraz szerokość geograficzną. Zmienna *cityGoogleString* jest niezbędna do prawidłowego zastosowania narzędzia GMap.NET. Tylko użytkownik o statusie administratora może dodawać lub usuwać stacje bazowe z bazy danych.

Połączenie aplikacji okienkowej z bazą danych odbywa się za pomocą klasy *DataContext*. W celu uaktywnienia połączenia należy stworzyć obiekt tej klasy i w specjalnie wygenerowanym pliku wybrać tabele z bazy, na których aplikacja będzie operować.

3.2 Okna aplikacji

Aplikacja została wyposażona w wiele okien użytkownika. Każde z nich pełni osobną funkcjonalność. Wszystkie okna wyposażone zostały w różnego rodzaju kontrolki, które pozwalają na interakcję z operatorem. Rozmieszczenie elementów jest intuicyjne, a przyciski zostały pokolorowane odpowiednim kolorem powodującym właściwe skojarzenia (czerwony — usuń, zielony — akceptuj).

Na rys. 3.2 przedstawiono okno logowania aplikacji. W tym miejscu użytkownik powinien wpisać swój login oraz hasło, a jeżeli jest administratorem musi zaznaczyć także pole *Admin*. Następnie po wcisnięciu przycisku *Log in* zostanie przeniesiony, z odpowiednimi uprawnieniami, do menu głównego programu. Naciśnięcie przycisku *Info* powoduje wyświetlenie najważniejszych informacji na temat programu (autorzy, kurs oraz prowadzący).



Rysunek 3.2: Okno logowania

Źródło: opracowanie własne

Menu główne aplikacji została zaprezentowane na rys. 3.3. Warto zaznaczyć, że serwisant może kliknąć tylko i wyłącznie przycisk *Route planning*. Pozostałe możliwości są dostępne tylko dla użytkownika z uprawnieniami administratora. Klikając odpowiednie przyciski można przejść kolejno do okna: zarządzania stacjami bazowymi, planowania drogi oraz zarządzania pracownikami.

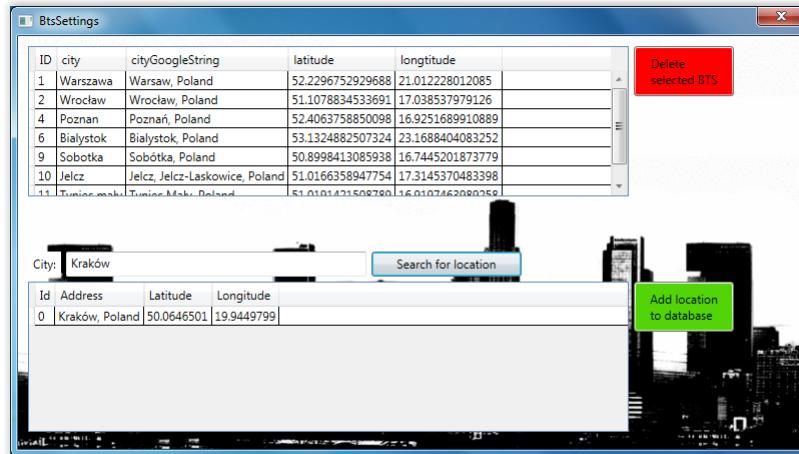


Rysunek 3.3: Menu główne programu

Źródło: opracowanie własne

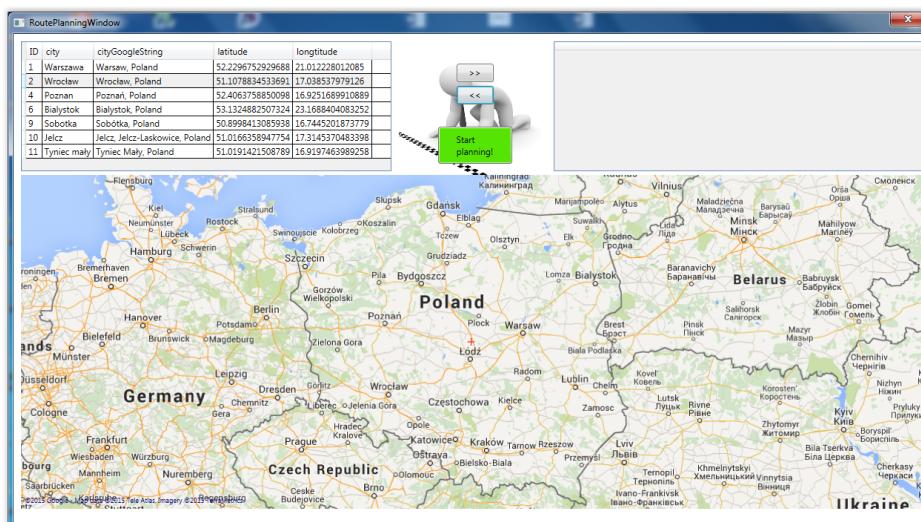
Okno zarządzania stacjami bazowymi (rys. 3.4) umożliwia dodane lub usuwanie miejscowości, w których znajdują się BTSy. W polu tekstowym obok etykiety *City* należy wpisać nazwę miasta, którym zainteresowany jest użytkownik. Aplikacja obsługuje polskie znaki diakrytyczne. Następnie po wcisnięciu przycisku *Search for location* w tabeli poniżej zostaną wyświetlane odpowiednie lokacje. Zaznaczenie lokacji odbywa się poprzez naciśnięcie jej lewym przyciskiem myszy. Gdy miejscowość zostanie zaznaczona, można kliknąć na zielony przycisk *Add location to database*, co spowoduje dodanie miasta do bazy danych. Analogicznie postępuje się w przypadku usuwania lokacji. W górnjej tabeli należy zaznaczyć miejscowości, a następnie

kliknąć na czerwonym przycisku *Delete selected BTS*. Należy zapamiętać, że można dodać oraz usuwać tylko jeden wiersz (aplikacja nie wspiera wielokrotnego jednorazowego dodawania oraz usuwania). Dostęp do omawianego okienka posiada tylko użytkownik o statusie administratora.



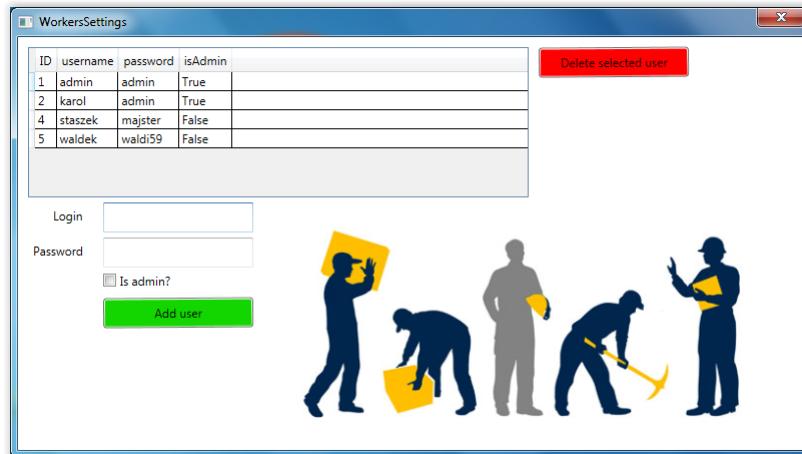
Rysunek 3.4: Okno zarządzania stacjami bazowymi
Źródło: opracowanie własne

Do okna planowania drogi (rys. 3.5) dostęp mają zarówno serwisanci jak i administratorzy. Obsługa jest bardzo intuicyjna. Zaznaczenie miasta w tabeli po lewej stronie, a następnie kliknięcie przycisku » powoduje jego przeniesienie do tabeli po prawej stronie. Miasta, które znajdują się w tabeli po prawej stronie biorą udział w ustalaniu optymalnej trasy, przy czym pierwsze miasto jest lokacją początkową. Miejscowość z prawej tabeli można oczywiście przenieść do lewej poprzez zaznaczenie jej, a następnie naciśnięcie przycisku «. Podobnie jak we wcześniejszych oknach, program umożliwia przeniesienie jednorazowo tylko jednego miasta. Kliknięcie przycisku *Start planning!* spowoduje obliczenie optymalnej trasy i narysowanie jej na mapie. Aby jednak zobaczyć wyrysowaną trasę należy wykonać operację przybliżenia/oddalenia na mapie.



Rysunek 3.5: Okno planowania drogi
Źródło: opracowanie własne

Ostatnie okno zostało przedstawiono na rys. 3.6. Umożliwia zarządzanie kadrą pracowniczą. Z przyczyn oczywistych jest dostępne tylko dla użytkownika o statusie administratora. Aby dodać serwisanta, bądź administratora, należy wpisać w odpowiednie pola login oraz hasło, a także zaznaczyć prawa dostępu (pole *Is admin?*), a następnie kliknąć przycisk *Add user*. Usuwanie użytkownika odbywa się poprzez jego zaznaczenie w tabeli, a następnie naciśnięcie czerwonego przycisku *Delete selected user*. Podobnie jak wcześniej można usunąć tylko jednego użytkownika przy jednym kliknięciu przycisku.



Rysunek 3.6: Okno zarządzania pracownikami
 Źródło: opracowanie własne

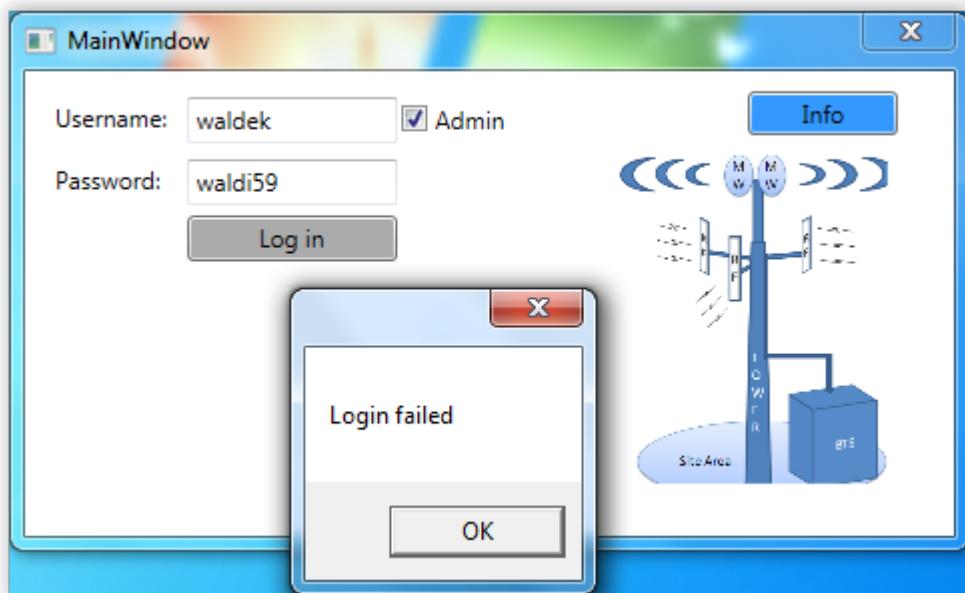
Rozdział 4

Prezentacja działania programu

W niniejszym rozdziale zaprezentowano przykładowe działanie programu. Starano się przedstawić jego funkcjonalność oraz zastosowanie. Opis został wzbo-gacony za pomocą zrzutów ekranu przedstawiających testowaną aplikację.

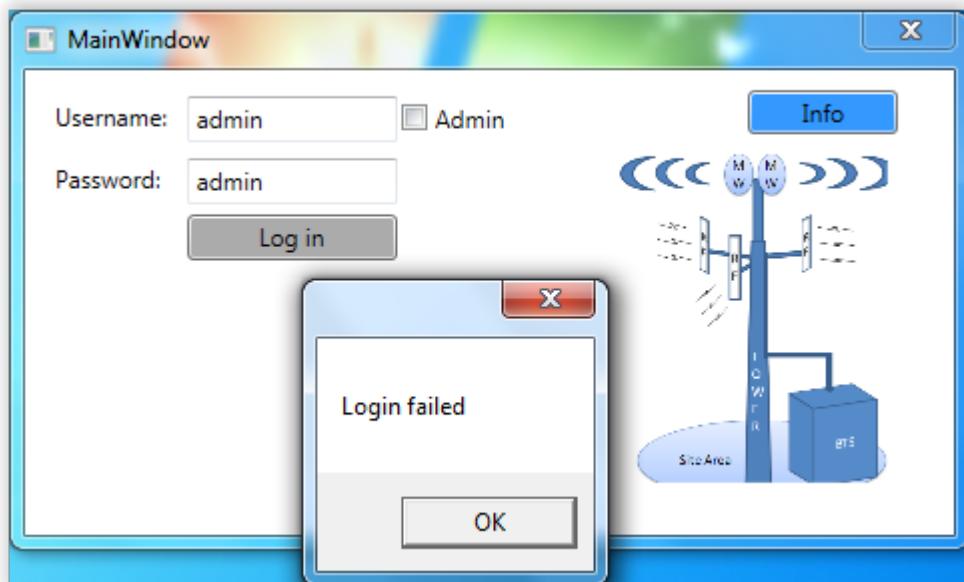
Ecranem, który wita użytkownika po uruchomieniu aplikacji, jest okno logowania. Program dopuszcza obecność dwóch grup użytkowników: serwisantów, którzy mogą tylko planować drogę, oraz administratorów, którzy poza znajdowaniem optymalnej trasy posiadają także możliwość zarządzania użytkownikami oraz lokalizacją stacji bazowych.

Na rys. 4.1 przedstawiono nieudaną próbę logowania przez serwisanta, który zaznaczył jako aktywną opcję *Admin*. Niestety pan Waldek nie posiada praw administratora, więc przy próbie wejścia do systemu otrzymał komunikat o błędzie.



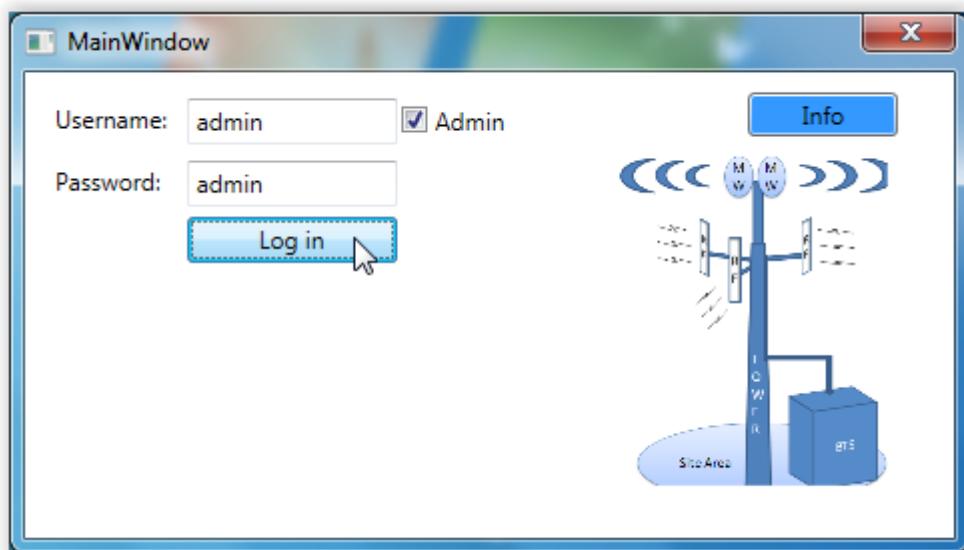
Rysunek 4.1: Błędne logowanie serwisanta
 Źródło: opracowanie własne

Podobna sytuacja może spotkać administratora. Została przedstawiona na rys. 4.2. Program nie pozwoli na zalogowanie użytkownika o statusie administratora bez zaznaczonej funkcji *Admin*. Za pomocą komunikatu o błędzie przypomina o tym, że odpowiednie pole musi zostać zaznaczone.



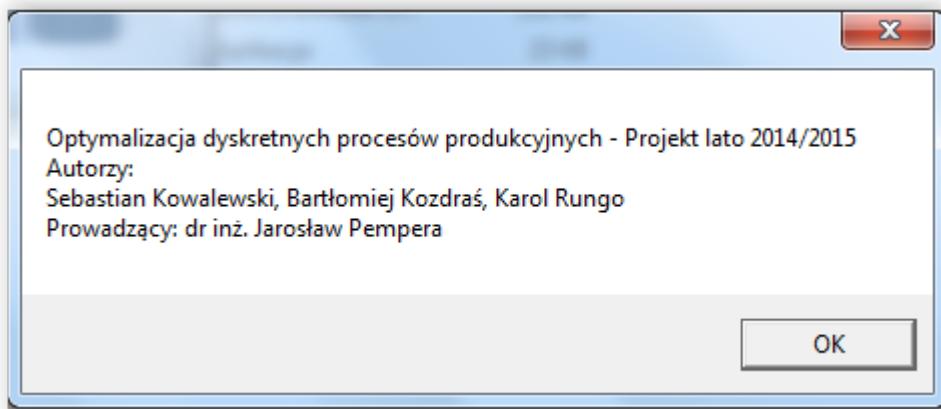
Rysunek 4.2: Błędne logowanie administratora
Źródło: opracowanie własne

Na rys. 4.3 przedstawiono prawidłowo przeprowadzoną próbę logowania. Administrator wpisał właściwy login oraz hasło, a ponadto zaznaczył pole *Admin*. W ten sposób po kliknięciu przycisku *Log in* zostanie on zalogowany do systemu.



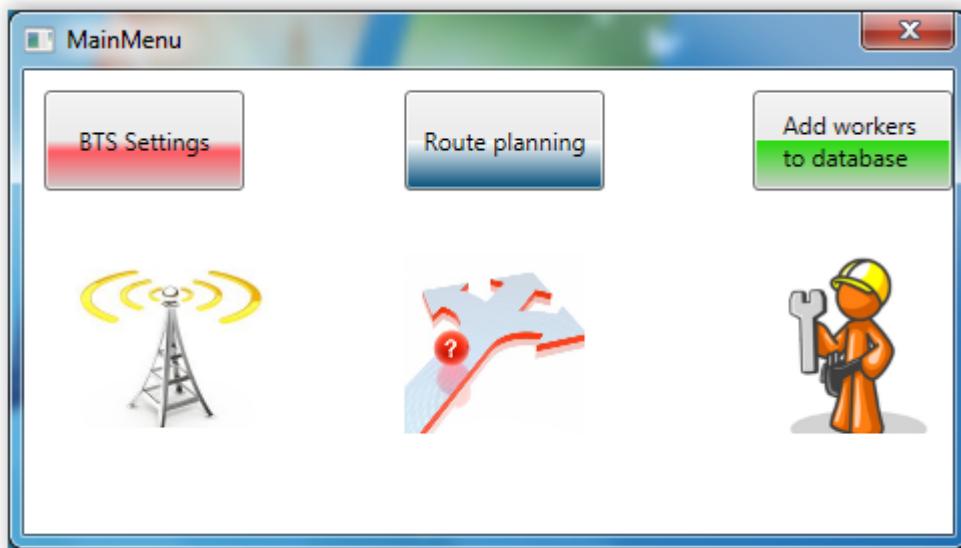
Rysunek 4.3: Logowanie administratora zakończone sukcesem
Źródło: opracowanie własne

Okienko logowania posiada również przycisk *Info*. Po jego kliknięciu pojawi się okienko informujące o autorach aplikacji oraz podstawowych informacjach z nią związanych (rys. 4.4).



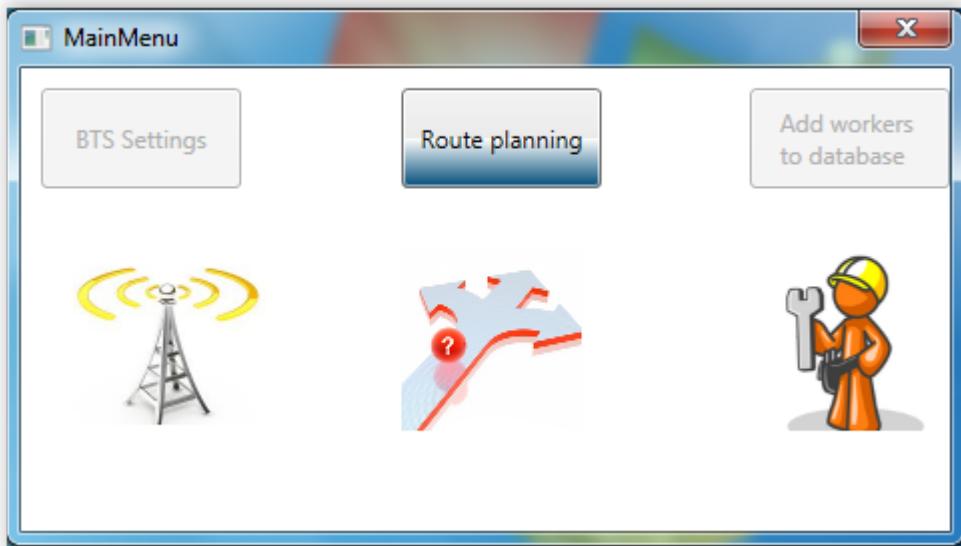
Rysunek 4.4: Okienko informacyjne programu
Źródło: opracowanie własne

Po prawidłowym przejściu procesu logowania oczom administratora ukaże się okienko głównego menu (rys. 4.5). Użytkownik o statusie administratora, klikając odpowiedni przycisk, może wybrać jedną z trzech akcji: zarządzania stacjami bazowymi, planowanie trasy, zarządzanie użytkownikami. Intuicyjny rysunek pod każdym z nich przywodzi na myśl odpowiednie skojarzenia z funkcjonalnością, która kryje się pod każdym przyciskiem. Powoduje to, że aplikacja jest prosta, intuicyjna i przyjemna w użyciu.



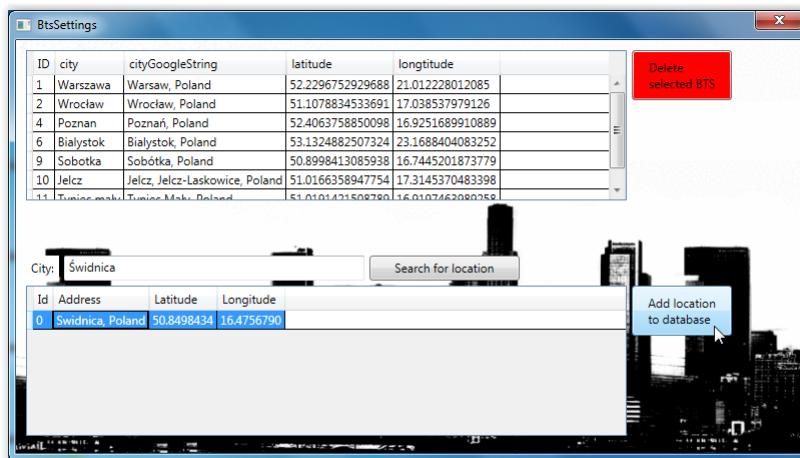
Rysunek 4.5: Funkcje dostępne w głównym menu dla administratora
Źródło: opracowanie własne

Ekran głównego menu będzie wyglądał trochę inaczej, jeżeli zalogowanym użytkownikiem jest serwisant (rys. 4.6). Jego uprawnienia pozwalają tylko i wyłącznie na planowanie drogi. Wydaje się być to sensownym rozwiązańem, gdyż nie ma większego sensu, aby umożliwić pracownikowi dodawanie BTSów, których konserwacją zajmuje się firma, a tym bardziej zapewnienie możliwości zarządzania innymi użytkownikami.



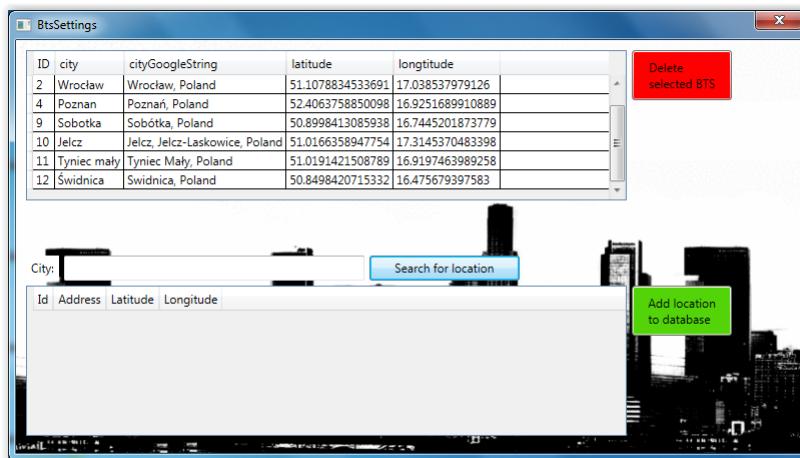
Rysunek 4.6: Funkcje dostępne w głównym menu dla serwisanta
Źródło: opracowanie własne

Okienko zarządzania stacjami bazowymi jest aktywne tylko dla użytkownika z prawami administratora. Na rys. 4.7 przedstawiono sposób znalezienia oraz dodania miejscowości do bazy danych. W przykładzie w odpowiednim polu wpisano nazwę miasta — Świdnica, a następnie kliknięto przycisk *Search for location*. Aplikacja znalazła odpowiednie powiązania, które zostały wyświetcone w tabeli poniżej. Zaznaczenie miejscowości, a następnie kliknięcie na przycisk *Add location to database* skutkuje zapisaniem miejscowości w bazie danych.



Rysunek 4.7: Znalezienie i dodanie miejscowości
Źródło: opracowanie własne

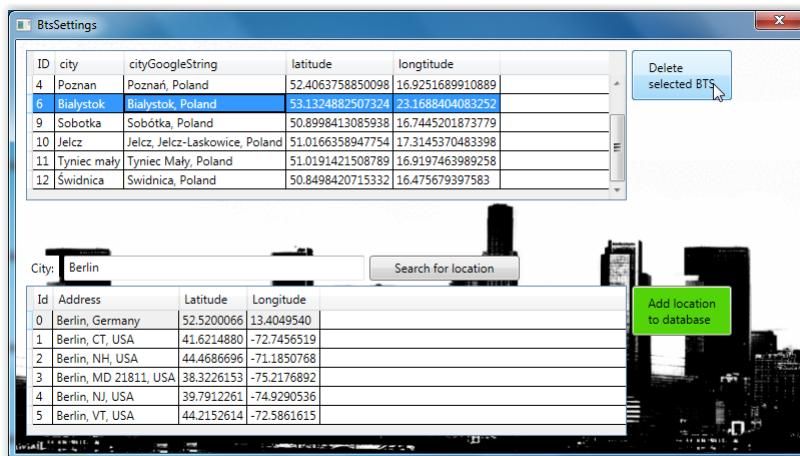
Na rys. 4.8 można przekonać się o tym, że Świdnica faktycznie została dodana do bazy danych. W górnej tabeli pod numerem ID równym dwanaście znajduje się wspomniana miejscowości.



Rysunek 4.8: Dodana miejscowości

Źródło: opracowanie własne

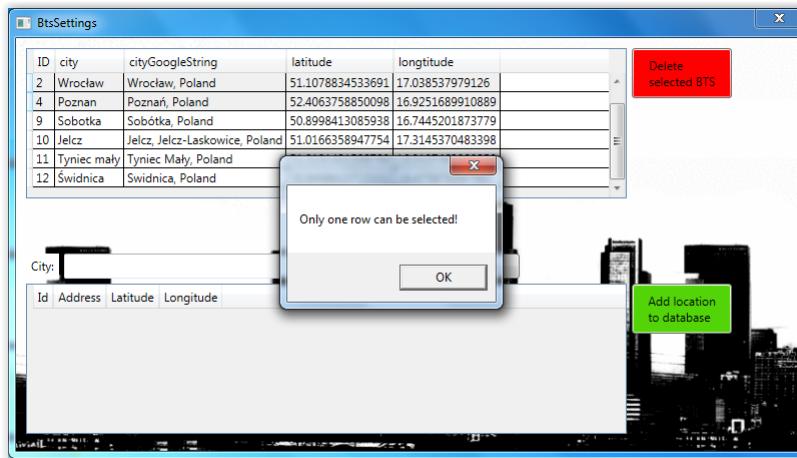
Administrator nie tylko jest zobowiązany do dodawania stacji bazowych do bazy danych, ale także do utrzymania należytego porządku, dlatego też czasami trzeba jakąś miejscowościę usunąć. Na rys. 4.9 przedstawiono sytuację, w której firma rezygnuje ze świadczenia usług w Białymstoku. Zaznaczenie wspomnianej miejscowości, a następnie kliknięcie przycisku *Delete selected BTS*, powoduje usunięcie lokacji z bazy danych. Co ciekawe na tym samym rysunku zaprezentowano sytuację, w której wyszukiwarka zwraca więcej niż jeden wynik (dla hasła Berlin zwrócono stolicę Niemiec oraz cztery miejscowości w Stanach Zjednoczonych).



Rysunek 4.9: Usunięcie lokacji z BTSem

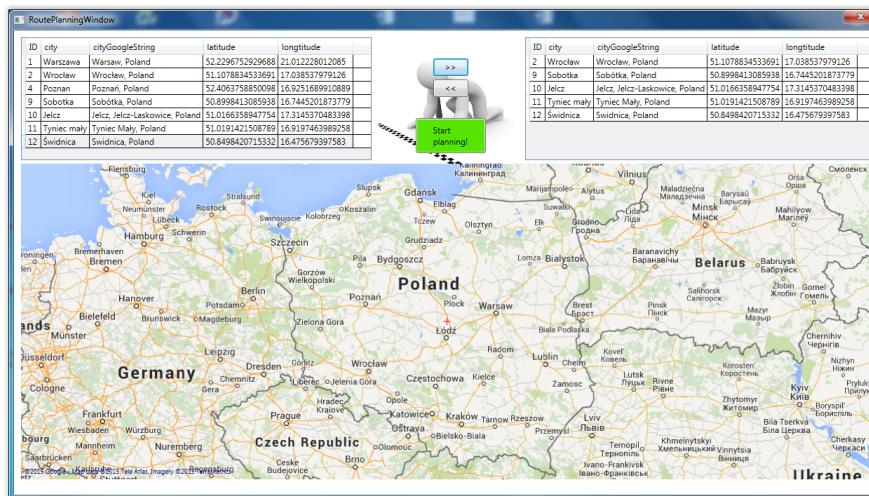
Źródło: opracowanie własne

Aplikacja nie wspiera usuwania, a także dodawania, wielu lokacji za jednym razem. Zaznaczenie kilku miejscowości, a następnie kliknięcie przycisku *Delete selected BTS* zakończy się wyświetleniem stosownego komunikatu o błędzie.



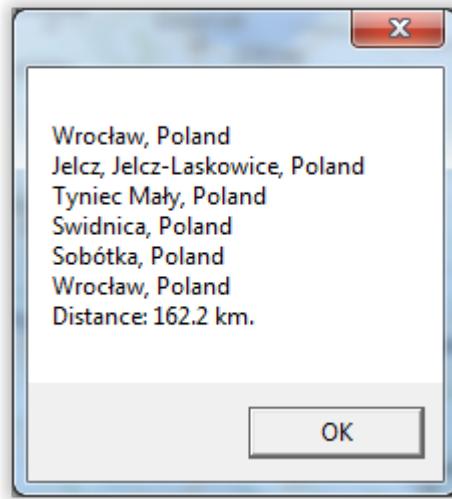
Rysunek 4.10: Błąd wynikający z zaznaczenia więcej niż jednej pozycji w tabeli
Źródło: opracowanie własne

Na rys. 4.11 przedstawiono okno planowania trasy. Z okna tego może korzystać zarówno administrator jak i serwisant. W tabeli po lewej stronie znajdują się miasta dostępne w bazie danych, po prawej — miasta, które należy odwiedzić. Przekazywanie lokacji pomiędzy tabeli odbywa się za pomocą przycisków » praz «, co zostało dokładnie opisane w rozdziale 3. W przedstawionym przykładzie użytkownik planuje odwiedzić kilka stacji bazowych na Dolnym Śląsku. Wybrano pięć miast: Wrocław, Sobótkę, Jelcz–Laskowice, Tyniec Mały oraz Świdnicę. Kliknięcie przycisku *Start planning!* rozpoczęte procedurę wyznaczania najlepszej trasy.



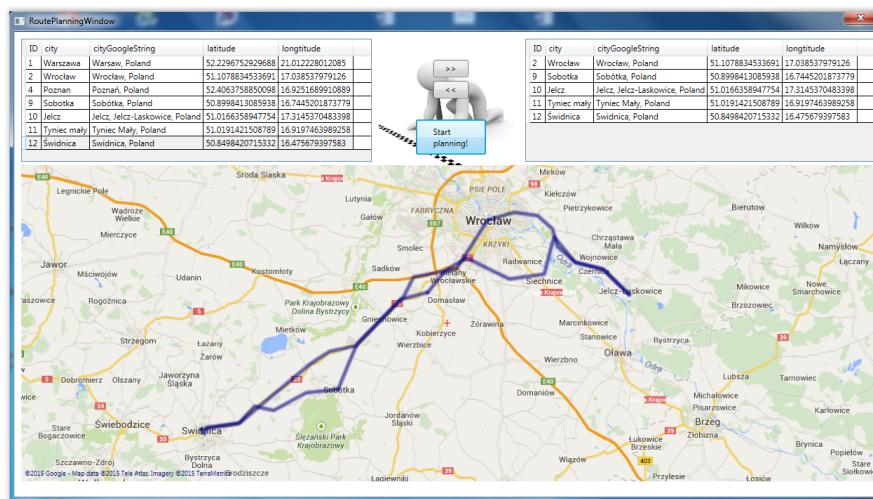
Rysunek 4.11: Wybór miast do odwiedzenia przez serwisanta
Źródło: opracowanie własne

Po zakończeniu działania procedury rozwiązujące problem komiwojażera program zwraca optymalną trasę (rys. 4.12). Jak widać najlepsza kolejność, w sensie najkrótszej drogi, dla wskazanych miast jest następująca: Wrocław → Jelcz–Laskowice → Tyniec Mały → Świdnica → Sobótka → Wrocław. Serwisant podczas odwiedzania wskazanych lokacji przejedzie trasę wynoszącą 162,2 km. Kliknięcie przycisku *OK* zamyka aktualne okno.



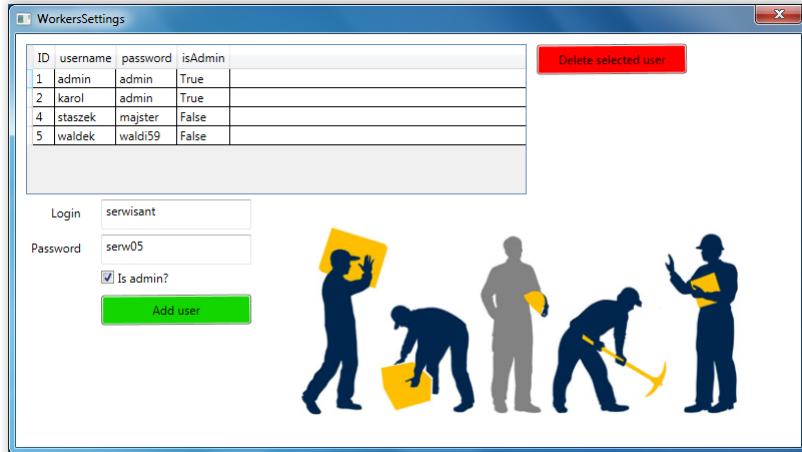
Rysunek 4.12: Przedstawienie optymalnej trasy
Źródło: opracowanie własne

Na rys. 4.13 przedstawiono wyrysowaną optymalną, wyznaczoną przez program trasę dla wskazanych miast. W tym przypadku należy mieć na uwadze ograniczenia aplikacji wspomniane w rozdziale 3.



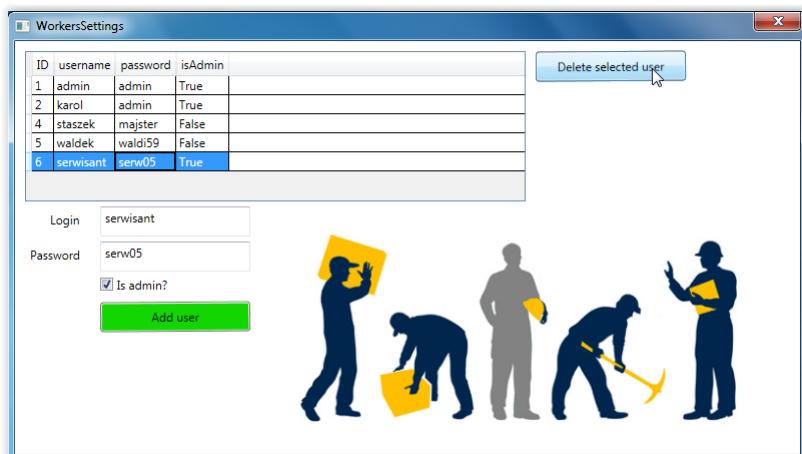
Rysunek 4.13: Optymalna trasa
Źródło: opracowanie własne

Rysunek 4.14 przedstawia okno zarządzania pracownikami, które dostępne jest tylko dla użytkownika z uprawnieniami administratora. W podanym przykładzie administrator wprowadził nowego serwisanta (poprzez podanie jego loginu i hasła oraz kliknięcie przycisku *Add user*). Niestety pomylił się i przyznał mu prawa administratora.



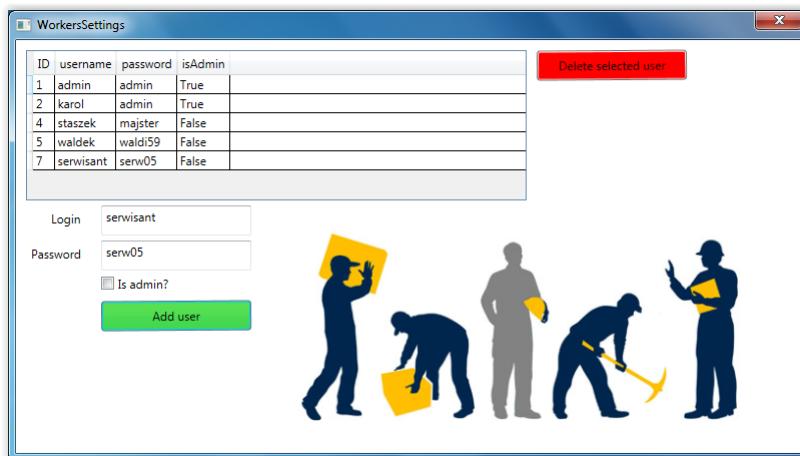
Rysunek 4.14: Dodanie serwisanta z błędnymi uprawnieniami
Źródło: opracowanie własne

W celu naprawienia zaistniałej pomyłki administrator zaznacza odpowiedni rekord w bazie danych, a następnie klika przycisk *Delete selected user* (rys. 4.15).



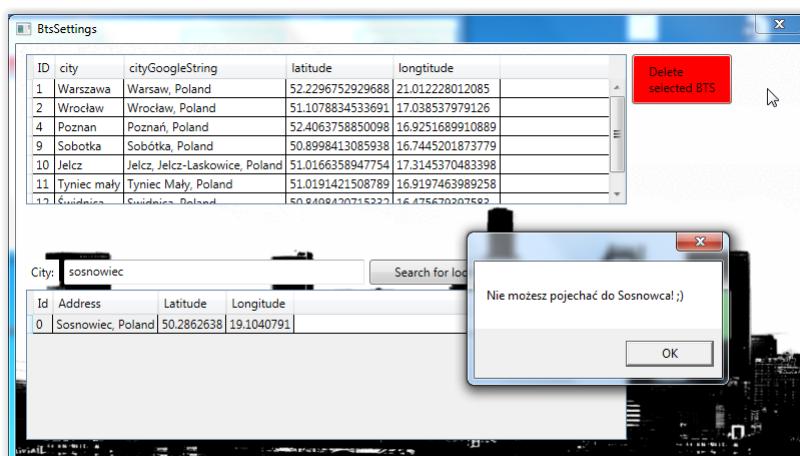
Rysunek 4.15: Usunięcie błędnie dodanego serwisanta
Źródło: opracowanie własne

Na rys. 4.16 widać, że serwisant z prawami administratora został usunięty, a dodano nowego pracownika, który nie będzie mógł już zarządzać stacjami bazowymi oraz innymi użytkownikami aplikacji.



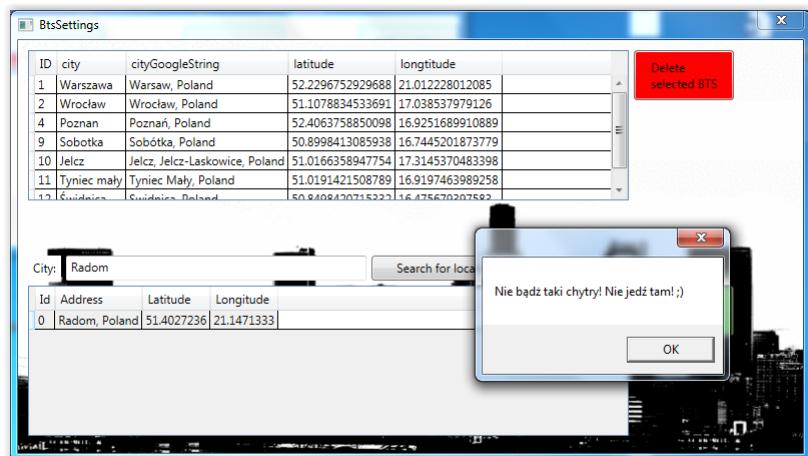
Rysunek 4.16: Dodanie serwisanta z poprawnymi uprawnieniami
Źródło: opracowanie własne

W trosce o bezpieczeństwo pracowników wprowadzono w aplikacji odpowiednie ulepszenia, które zapewniają jak najlepsze warunki pracy serwisantów. Uwzględniając niezbyt pochlebna reputację Sosnowca dojazd tam jest niemożliwy (rys. 4.17).



Rysunek 4.17: Błędne wskazanie miejscowości — Sosnowiec
Źródło: opracowanie własne

Podobnie jak w przypadku Sosnowca nie jest także możliwy wyjazd do Radomia (rys. 4.18).



Rysunek 4.18: Błędne wskazanie miejscowości — Radom
Źródło: opracowanie własne

Niniejsza prezentacja działania programu powinna utwierdzić w przekonaniu, że zaprojektowana aplikacja jest skutecznym narzędziem pozwalającym planować trasę dla serwisanta stacji bazowych.

Rozdział 5

Zakończenie

W ramach niniejszego projektu wykonano kompleksową aplikację, która ma zastosowanie praktyczne. Dzięki zaprojektowanemu programowi możliwe jest wyznanie optymalnej trasy dla serwisanta firmy telekomunikacyjnej, którego obowiąkiem jest konserwacja stacji bazowych w różnych miejscowościach. Przedstawione zagadnienie jest rzeczywistą realizacją znanego w literaturze specjalistycznej problemu komiwojażera.

Aplikacja została napisana w języku programowania C Sharp. Wykorzystano również wiele narzędzi dostarczonych przez firmę Microsoft, które pozwalają na skonstruowanie zaawansowanego interfejsu graficznego. Wśród zastosowanych narzędzi wymienić należy: Microsoft Visual Studio, platforma .NET, GMap.NET, Microsoft Blend, Windows Presentation Foundation.

Poza interfejsem graficznym zaimplementowano także algorytm komiwojażera, który wyznacza optymalną trasę dla małych grafów. Założenie taki jest sensowne, gdyż serwisant nie jest w stanie odwiedzić w ciągu dnia zbyt dużej liczby miejscowości.

Całość aplikacji wzbogaca baza danych zrealizowana w aplikacji SQL Server Management Studio 2008. Dzięki zastosowaniu języka C Sharp połączenie z bazą danych jest proste i intuicyjne poprzez zastosowanie technologii LINQ.

W ramach niniejszej dokumentacji projektu zaprezentowano również rozbudowany przykład praktyczny, który przedstawia funkcjonalności aplikacji oraz sposób jej użycia. Przybliżono również zastosowane narzędzia, oprogramowanie oraz algorytm.

Bibliografia

- [1] *GMap.NET — Great Maps for Windows Forms & Presentation.* Dostępny na: <https://greatmaps.codeplex.com/> (dostęp: 29.03.2015 r.).
- [2] *Problem komiwojazera.* Dostępny na: http://edu.i-lo.tarnow.pl/inf/alg/001_search/0140.php (dostęp: 29.03.2015 r.).
- [3] *Travelling salesman problem.* Dostępny na: http://en.wikipedia.org/wiki/Travelling_salesman_problem (dostęp: 29.03.2015 r.).

Spis rysunków

2.1	Graf zupełny dla dziesięciu wierzchołków	7
2.2	Przykładowy cykl Hamiltona w grafie o dziesięciu wierzchołkach	7
3.1	Tabele w bazie danych	10
3.2	Okno logowania	11
3.3	Menu główne programu	11
3.4	Okno zarządzania stacjami bazowymi	12
3.5	Okno planowania drogi	12
3.6	Okno zarządzania pracownikami	13
4.1	Błędne logowanie serwisanta	14
4.2	Błędne logowanie administratora	15
4.3	Logowanie administratora zakończone sukcesem	15
4.4	Okienko informacyjne programu	16
4.5	Funkcje dostępne w głównym menu dla administratora	16
4.6	Funkcje dostępne w głównym menu dla serwisanta	17
4.7	Znalezienie i dodanie miejscowości	17
4.8	Dodana miejscowość	18
4.9	Usunięcie lokacji z BTSem	18
4.10	Błąd wynikający z zaznaczenia więcej niż jednej pozycji w tabeli	19
4.11	Wybór miast do odwiedzenia przez serwisanta	19
4.12	Przedstawienie optymalnej trasy	20
4.13	Optymalna trasa	20
4.14	Dodanie serwisanta z błędymi uprawnieniami	21
4.15	Usunięcie błędnie dodanego serwisanta	21
4.16	Dodanie serwisanta z poprawnymi uprawnieniami	22
4.17	Błędne wskazanie miejscowości — Sosnowiec	22
4.18	Błędne wskazanie miejscowości — Radom	23