

## Opis problemu

//TODO

Co trzeba było i czego nie zrobiliśmy. Cel projektu.

## Opis architektury

//TODO

Że niby MVC i dlaczego HTML i że niby jest to powszechne rozwiązanie od Metasploita po CUPS.

## Plan projektu

//TODO

Czyli dlaczego to ja najbardziej zjebałem i kto się czym zajmował.

# Opis algorytmu

//TODO

Algorytm blablabla

**Przestrzeń problemu** - graf połączeń między lotniskami i dwa jego wybrane wierzchołki oznaczone jako początkowy i końcowy. Każda krawędź grafu ma przypisane parametry: koszt, czas, komfort i bezpieczeństwo połączenia. Ponadto każdemu parametrowi odpowiada pewna waga.

**Przestrzeń rozwiązań** - ścieżki łączące zadane dwa wierzchołki.

**Osobniki** - ścieżki w grafie połączeń.

**Cechy** - kolejne wierzchołki należące do ścieżki. Pośrednio - długość ścieżki.

**Funkcja przystosowania** - ocena jakości ścieżki, będąca sumą uśrednionych wartości parametrów krawędzi należących do ścieżki.

**Krzyżowanie** - jednopunktowe. Dwie krzyżowane ścieżki rozcinane są w losowo (rozkład jednorodny) wybranych punktach. Pierwsza część pierwszej ścieżki łączona jest drugą częścią drugiej ścieżki, a druga część pierwszej ścieżki łączona jest z pierwszą częścią drugiej ścieżki.

**Mutacje** - dla każdego wierzchołka ścieżki istnieje pewna szansa na zajście mutacji. Mutacja polega na usunięciu, dodaniu lub zamienieniu wierzchołka w danym miejscu ścieżki. Wykonana operacja jest wybierana losowo.

każda krawędź ma współczynniki komfortu i bezpieczeństwa od 1 do 10  
każda krawędź jest przebywana z pewną szybkością od 1 do 10, mając dystans między wierzchołkami wyliczam czas podróży  
koszt jest średnią komfortu i szybkości przemnożoną przez dystans

oceniając ścieżkę  
obliczam dystans (w linii prostej) między punktami początkowym i końcowym  
obliczam sumaryczny czas podróży i dzielę przez niego dystans, dostaję a  
obliczam sumaryczny koszt podróży i dzielę przez dystans, dostaję b  
wyliczam średni komfort w czasie podróży, dostaję c  
wyliczam średnie bezpieczeństwo w czasie podróży, dostaję d

mnożę a, b, c, d przez odpowiednie współczynniki zadane przez użytkownika i sumuję, dostaję ocenę ścieżki

# Instrukcja uruchomienia aplikacji

//TODO

Na linuxie make, na Windowsie masz przejebane

Screenshoty od Andrzeja.

# Wygląd i obsługa aplikacji

## Zarys interfejsu

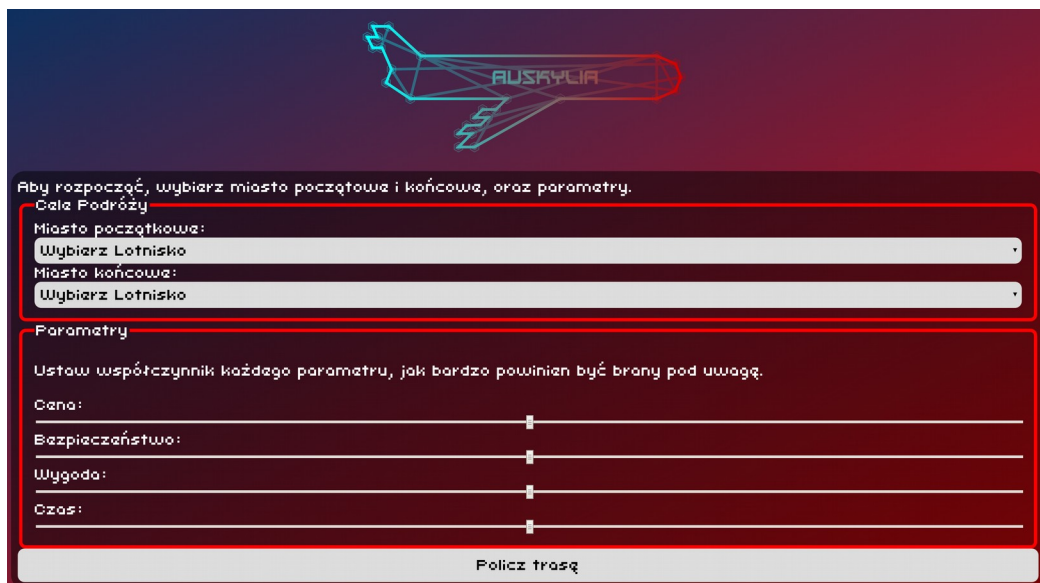
Główny interfejs jest stroną internetową. Za jego pomocą następuje komunikacja z resztą aplikacji, która działa jak mały serwer WWW.

## Wymagania przeglądarki

Interfejs wymaga od przeglądarki obsługi najnowszych standardów HTML5 i CSS3. Najnowsze wersje Chrome/Chromium, Firefoxa i Opery bez problemu poradzą sobie z zadaniem. Jeśli przeglądarka nie posiada niektórych funkcjonalności, wpłynie to jedynie na wygląd, a nie na działanie programu. Ze względu na użytą wersję jQuery, aplikacja nie zadziała na Internet Explorer 8 i starszych.

## Uruchomienie

Domyślnie aplikacja działa na porcie 5005, aby uzyskać interfejs, należy wpisać <http://localhost:5005>. Po uruchomieniu i wczytaniu, strona ściąga listę wszystkich lotnisk na świecie. Trwa to zazwyczaj kilka sekund. W tym czasie wyświetlane jest okno ładowania, a interfejs jest zablokowany. Po wczytaniu lotnisk, możemy podać dane do obliczeń. Interfejs wygląda wtedy następująco.



The screenshot shows a web application interface for a flight calculator. At the top, there is a header with a dark blue gradient background and a stylized airplane icon with the word "ALSKYLIA" on its side. Below the header, the main content area has a dark red background. It contains a form with the following elements:

- A text instruction: "Aby rozpocząć, wybierz miasto początkowe i końcowe, oraz parametry."
- A section titled "Cela Podróży" (Travel Purpose) with two dropdown menus:
  - "Miasto początkowe:" (Starting city) with a dropdown menu showing "Wybierz Lotnisko" (Select Airport).
  - "Miasto końcowe:" (Destination city) with a dropdown menu showing "Wybierz Lotnisko" (Select Airport).
- A section titled "Parametry" (Parameters) with a text instruction: "Ustaw współczynniki każdego parametru, jak bardzo powinien być brany pod uwagę." (Set coefficients for each parameter, how much it should be taken into account). Below this are four sliders for:
  - "Cena:" (Price)
  - "Bezpieczeństwo:" (Safety)
  - "Wygoda:" (Convenience)
  - "Czas:" (Time)
- A button at the bottom labeled "Policz trasę" (Calculate route).

## Wprowadzanie danych

Za pomocą dwóch list możemy wybrać lotniska początkowe i końcowe. Standardowo przy tego typu kontrolkach, wpisanie litery na klawiaturze odsyła nas do pierwszego miasta na tę literę.



Po wyborze miast, możemy użyć suwaków poniżej do wprowadzenia parametrów znajdowania drogi. Są to:

- ➔ **Cena za bilet.** Powinna być jak najmniejsza. Przesuwając suwak w lewo zmniejszamy wpływ tego parametru na obliczenia – nie obchodzi nas, czy zapłacimy dużo, czy mało. Przesuwając w prawo zwiększamy wpływ parametru, chodzi nam o to, żeby cena przelotów była jak najmniejsza kosztem innych własności.
- ➔ **Bezpieczeństwo podróży.** Bywają połączenia nad niebezpiecznymi rejonami i starymi maszynami. Suwak z lewej strony zmniejszy nam wpływ tego parametru, suwak z prawej wybierze nam najnowsze i najbezpieczniejsze połączenia.
- ➔ **Wygoda.** Niektórzy nie obejdą się bez pierwszej klasy i obiadu posypanego płatkami złota. Przesuwając trzeci suwak w prawo wybierzemy tylko najwygodniejsze połączenia rejsowe. Suwak z lewej zagwarantuje nam miejsce w luku bagażowym.
- ➔ **Czas podróży.** Są historie ludzi spędzających na lotnisku całe dni. Jeśli chcemy do celu dolecieć jak najszybciej, warto ustawić ten ostatni suwak z prawej strony. Ustawienie go z lewej zmniejszy nam wpływ czasu na kalkulację i może zwiększyć ogólny czas podróży.

Jeśli jesteśmy zadowoleni z danych, możemy wysłać je do aplikacji i rozpocząć obliczanie ścieżki. Naciśnięcie ostatniego przycisku zablokuje interfejs na czas wysłania i dostaniemy możliwość monitorowania postępu.

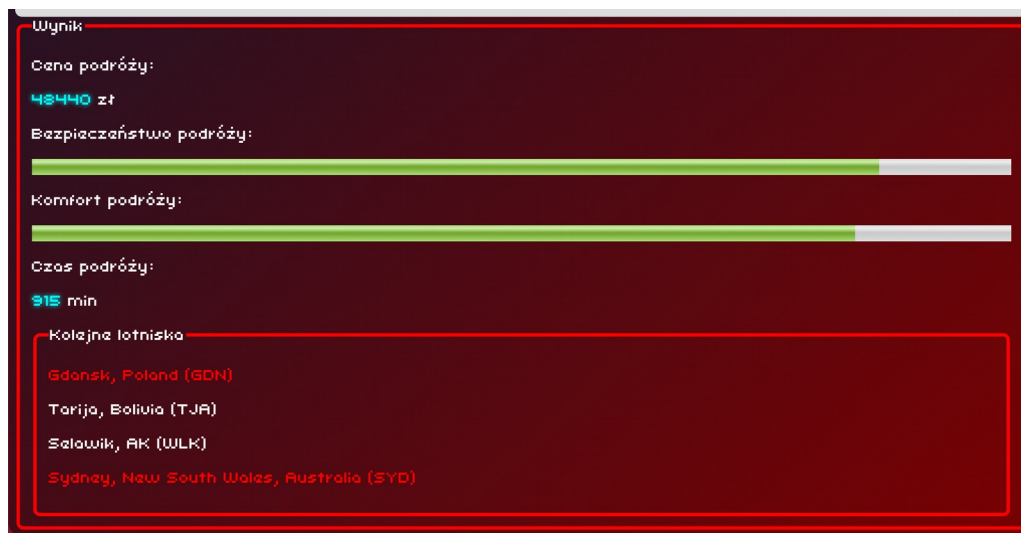
## Oczekiwanie na wynik

W czasie oczekiwania na wynik, pokaże się pasek postępu. Zazwyczaj aplikacja działa na tyle szybko, że ów pasek od razu wypełni się do 100%. Poniżej paska znajduje się opcja przerwania obliczeń. Po przerwaniu dostaniemy błąd serwera o celowym niedokończeniu obliczania wyniku. Po uzyskaniu 100% pokaże się wynik.



## Wyświetlenie wyniku

Program po obliczeniu wyświetli wynik pod przyciskiem i schowa okno postępu. Aby przeliczyć nową trasę, wystarczy ponownie ustawić wartości i załączyć obliczenia. Ono wyniku składa się z 5 pól. Na niego składają się obliczona cena przelotu, bezpieczeństwo, komfort, czas i lista lotnisk.

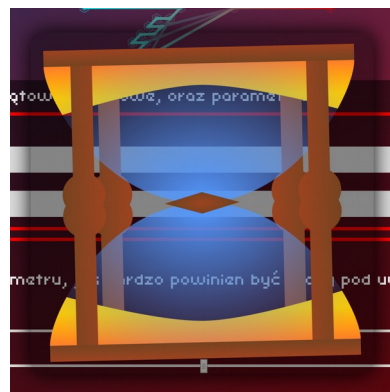
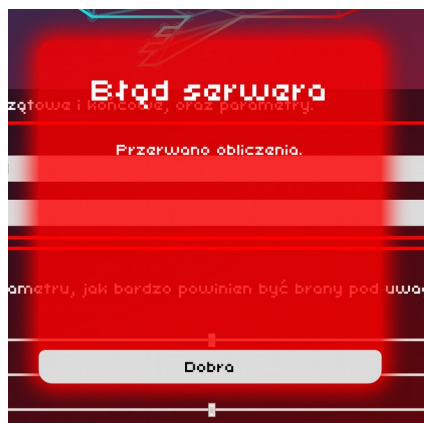


- ➔ Cena podróży to sumaryczna wartość przelotów na wszystkich krawędziach. Tyle musimy przygotować funduszy na całą podróż.
- ➔ Pasek bezpieczeństwa to średnia z bezpieczeństw każdego z przelotów. Jeśli jeden przelot jest bardzo bezpieczny, a drugi mało, to wynik skaże wartość pomiędzy.
- ➔ Komfort również jest średnią z każdej krawędzi. Komfortowe samoloty podwyższają wartość, a ekonomiczne obniżają.
- ➔ Czas podróży jest sumą wszystkich czasów na ścieżce, bez brania pod uwagę oczekiwania na lotniskach.

Następnie jest lista kolejnych lotnisk przez które przechodzi nasz lot. Pierwsze i ostatnie są zaznaczone kolorem.

## Okno i stany programu

Program w czasie wysyłania i odbierania danych wyświetla okno i animacje klepsydry. Jeśli serwer zwróci błąd, lub nie uda się wysłać/odebrać danych, wyświetli się powiadomienie o błędzie.





Wszystkie grafiki są w formacie wektorowym i zostały stworzone na potrzeby tego projektu. Animacje pojawiania się i znikania okien dostarczyło jQuery, a animacja obracania klepsydry jest natywna dla CSS3. Gradient tła również jest osiągnięty przez CSS, a nie przez grafikę.

## Wnioski i podsumowanie

Najważniejszy wniosek, jaki wyciągnęliśmy z projektu, jest taki, że gdybyśmy mieli robić go jeszcze raz, to zrobilibyśmy go inaczej. Przede wszystkim należy zmienić sposób modelowania ścieżek. Zamiast ciągu kolejnych wierzchołków ścieżki, lepsze byłoby zastosowanie zapewne kodowania bitowego do zapisania kolejności oraz obecności wierzchołków w ścieżce. Jednym słowem algorytm całkowicie generyczny. Rozwiązało by to wiele problemów, takich jak konieczność sprawdzania poprawności ścieżki w celu uniknięcia cykli lub zmienna liczba cech osobników.

Pierwotna koncepcja zakładała zastosowanie krzyżowania równomiernego (dla odpowiednio znormalizowanych ścieżek o różnych długościach - z losowo wstawionymi genami "pustymi").

Pomysł ten, poza problematyczną implementacją, jak łatwo przewidzieć nie sprawdził się zbyt dobrze. Jako, że dopasowanie do optimum każdej cechy zależało nie tylko od wierzchołka na danej pozycji, ale też i tych na sąsiednich pozycjach, krzyżowanie dwóch dobrych ścieżek dawało rezultaty zupełnie losowe.

Znaczącą poprawę wprowadziła zmiana krzyżowania na jednopunktowe (częściowo rozwiązując przy okazji problem ścieżek różnej długości). Dzięki temu dwie dobre ścieżki generują potomka, u którego złą może być tylko jedna krawędź. Pozwoliło to też prosto generować potomków o różnych długościach.

Innym problemem okazały się powtarzające się ścieżki oraz bardzo krótkie ścieżki. Bardzo szybko prowadziły one do wypełnienia populacji jednakowymi osobnikami, zabijając dalsze działanie algorytmu. Zastąpienie potomków bardzo krótkich ścieżek oraz powtarzających się osobników nowo losowanymi ścieżkami rozwiązało problem.

Warty uwagi jest także problem generowania grafu. Parametry połączeń generowane zupełnie losowo prowadziły do powstania grafu, gdzie - uśredniając - wszystkie punkty były od siebie mniej więcej równoodległe, co prowadziło do matematycznej niemożliwości. W efekcie uzyskiwane populacje zawsze składały się z bardzo krótkich ścieżek.

Sytuację poprawiło losowanie pozycji punktów na pomocniczej płaszczyźnie - dzięki temu jeśli A jest blisko B, a daleko C, B również będzie daleko od C, co oznacza, że ścieżka złożona z punktów leżących blisko odcinka łączącego punkt początkowy z końcowym nie będzie wiele dłuższa od tegoż odcinka.

Inną kwestią godną wspomnienia jest funkcja przystosowania. Bardzo wyraźnie można było zaobserwować wpływ odpowiedniego dobrania funkcji na jakość algorytmu.