# Spis treści

# 1. Wprowadzenie

## 1.1. Przeznaczenie

Celem tej dokumentacji jest przedstawienie projektu na temat „System rezerwacji biletów lotniczych". Projekt został podzielony na 2 aplikację: serwera i klienta. Aplikacja serwera ma zapewniać dostęp do funkcjonalności dla klienta poprzez REST API.
Aplikacja zapewnia funkcjonalności takie jak:

- dodawanie/edycja/usuwanie lotu,
- przekazanie wszystkich lotów,
- rezerwację na dany lot,
- wyszukiwanie danej rezerwacji po numerze ID,
- usunięcie rezerwacji
- przesłanie pliku PDF ze szczegółami rezerwacji.

Aplikacja kliencka ma za zadanie w prosty sposób udostępnić powyższe metody dla użytkownika za pomocą aplikacji okienkowej.

## 1.2. Definicje, akronimy, skróty

- **Id** – identyfikator,
- **REST (Representational state transfer)** – styl architektury oprogramowania wywiedziony z doświadczeń przy pisaniu specyfikacji protokołu HTTP dla systemów rozproszonych. REST wykorzystuje m.in. jednorodny interfejs, bezstanową komunikację, zasoby, reprezentacje, hipermedia, HATEOAS [1].
- **GlassFish** – otwarty serwer aplikacji dla platformy Java EE, rozwijany przez Oracle. Rozpowszechniany jest na dwóch licencjach: CDDL oraz GPL. Rozwijana jest także wersja komercyjna − *Oracle GlassFish Enterprise Server* [2].
- **NetBeans -** projekt otwartego oprogramowania mający za zadanie dostarczanie efektywnych narzędzi programowania. Dwa najważniejsze produkty to NetBeans IDE oraz NetBeans Platform [3].
- **WADL -** bazujący na XML-u standard opisu aplikacji internetowych opartych na transmisji poprzez protokół HTTP, który jest łatwo odczytywalny przez maszyny. Zazwyczaj opisuje aplikacje napisane w zgodzie ze standardem REST.
  Celem WADL jest ułatwienie integracji aplikacji internetowych opartych na rozproszonych protokołach transmisji danych. Ułatwia on automatyczne tworzenie i konfigurowanie usług, dostarczając opisu aplikacji, który w innym przypadku powinien być ręcznie "odkryty" i opisany. WADL można traktować jako odpowiednik standardu WSDL (*Web Services Description Language*) w wersji 1.1. Wersja 2.0 umożliwia już opisywanie aplikacji REST-owych, stąd stanowi zamiennik standardu WSDL [4].
- **XML (Extensible Markup Language) -** uniwersalny język znaczników przeznaczony do reprezentowania różnych danych w strukturalizowany sposób. Jest niezależny od platformy, co umożliwia łatwą wymianę dokumentów pomiędzy heterogenicznymi (różnymi) systemami i znacząco przyczyniło się do popularności tego języka w dobie Internetu. XML jest standardem rekomendowanym oraz specyfikowanym przez organizację W3C [5].

- **WPF (Windows Presentation Foundation) –** struktura interfejsu użytkownika, która tworzy aplikacje klienckie dla komputerów stacjonarnych. Platforma programowa WPF obsługuje szeroki zestaw funkcji tworzenia aplikacji, w tym model aplikacji, zasoby, formanty, grafikę, układ, powiązanie danych, dokumenty i zabezpieczenia. Struktura jest częścią platformy .NET, więc jeśli wcześniej utworzono aplikacje z programem .NET przy użyciu ASP.NET lub Windows Forms, środowisko programowania powinno być znane. WPF używa języka znaczników aplikacji rozszerzalnej (XAML) [6].
- **HATEOAS (Hypermedia as the Engine of Application State)** - Hypermedia jako silnik stanu aplikacji jest składnikiem architektury aplikacji REST, który odróżnia ją od innych architektur aplikacji sieciowych. Dzięki HATEOAS klient wchodzi w interakcję z aplikacją sieciową, której serwery aplikacji dostarczają informacje dynamicznie poprzez hipermedia [7].

### 1.3. Referencje

**[1]** - https://pl.wikipedia.org/wiki/Representational_state_transfer
**[2]** - https://pl.wikipedia.org/wiki/GlassFish
**[3]** - https://pl.wikipedia.org/wiki/NetBeans
**[4]** - https://pl.wikipedia.org/wiki/Web_Application_Description_Language
**[5]** - https://pl.wikipedia.org/wiki/XML
**[6]** - https://docs.microsoft.com/pl-pl/visualstudio/designers/getting-started-with-wpf?view=vs-2019
**[7]** - https://en.wikipedia.org/wiki/HATEOAS

# 2. SPECYFIKACJA USŁUGI REST

## 2.1. Szczegóły dotyczące aplikacji wraz z opisem udostępnianych

Aplikacja serwera została stworzona w języku Java przy pomocy programu Apache NetBeans IDE 11.3. REST API posiada 2 klasy zasobów FlightResource i ReservationResource. FlightResource udostępnia dostęp do lotów, natomiast ReservationResource dostęp do rezerwacji.

**FlightResource** posiada następujące metody:

- **List<Flight> getFlights()** – metoda zwraca listę obiektów Flight (listę wszystkich lotów), dostęp poprzez żądanie GET pod ścieżką http://localhost:8080/rsiproj2/resources/flights.
  Klasa Flight zawiera następujące pola:
  - **int Id** – numer id lotu
  - **string To_City** – miasto do którego odbywa się lot
  - **string From_City** – miasto z którego odbywa się lot
  - **Date FlightDepartureDate** – data odlotu
  - **Date FlightArrivalDate** – data przylotu
  - **double Price** – cena lotu za 1 osobę
  - **int NumberOfSeats** – ilosć miejsc w samolocie na dany lot
  - **int NumberOfAvaiableSeats** – ile miejsc jest jeszcze wolnych

- **Flight getFlight(@PathParam("flightId") int id)** – motoda zwraca lot o podanym identyfikatorze podanym w ścieżce, dostęp poprzez żądanie GET, pod ścieżką http://localhost:8080/rsiproj2/resources/flights/{flightId}.

- **Flight AddFlight(FlightDTO flight)** – metoda udostępnia dodawanie danego lotu, do metody przekazywany jest obiekt dodawanego lotu i jest on dodawany do bazy (klasa AddEditFlightDTO ma podobne właściwości do klasy Flight), po dodaniu zwracany jest dodany obiekt klasy Flight. Dostęp poprzez żądanie POST pod ścieżką http://localhost:8080/rsiproj2/resources/flights. Obiekt przekazywany jest w ciele żądania, jako JSON. Metoda posiada autentykację za pomocą Basic Auth.

- **bool updateFlight(@PathParam("flightId") int id, FlightDTO flightDTO)** – metoda udostępnia edycję danego lotu, do metody przekazywane jest id oraz obiekt edytowanego lotu i edytowany jest obiekt o danym id w bazie, metoda zwraca wartość boolean, która mówi, czy operacja została zakończona powodzeniem (true) czy nie udało się poprawnie edytować danego lotu (false). Dostęp poprzez żądanie PUT pod ścieżką http://localhost:8080/rsiproj2/resources/flights/{flightId}. Obiekt przekazywany jest w ciele żądania, jako JSON, a id w ścieżce. Metoda posiada autentykację za pomocą Basic Auth.

- **bool deleteFlight(@PathParam("flightId") int id)**– metoda udostępnia usuwanie danego lotu, do metody przekazywane jest id danego lotu brane ze ścieżki, metoda zwraca wartość boolean, która mówi, czy lot o podanym id został usunięty (true), czy nie (false). Dostęp poprzez żądanie DELETE pod ścieżką http://localhost:8080/rsiproj2/resources/flights/{flightId}. Metoda posiada autentykację za pomocą Basic Auth.

**ReservationResource** posiada następujące metody:

- **ReservationDTO getReservationById(@PathParam("reservationId") int id, @Context UriInfo uriInfo)** – metoda przekazuje szczegóły rezerwacji dla podanego id rezerwacji. Dostęp poprzez żądanie GET pod ścieżką http://localhost:8080/rsiproj2/resources/reservation/{reservationId}.

  Klasa ReservationDTO posiada następujące pola:
  - **int Id** – numer id rezerwacji
  - **int FlightId** – numer id lotu
  - **Flight Flight** – obiekt danego lotu
  - **List<Person> People** – lista osób, które są w danej rezerwacji.
  - **List<Link**> - lista linków dla zwracanego obiektu (HATEOAS).

  Klasa Person podsiada następujące właściwości:
  - **int Id** – id osoby dla daej rezerwacji
  - **string FirstName** – imie
  - **string Surname -** nazwisko
  - **DateTime BirthDate** – data urodzenia
  - **int ReservationId** – numer id rezerwacji.

- **int addReservation(AddReservationDTO addReservationDTO)** – metoda udostępnia dodawanie rezerwacji dla danego lotu, przekazywany jest do metody obiekt dodawanej rezerwacji, czyli osób które rezerwują dany lot (AddReservationDTO jest podobne do klasy ReservationDTO), metoda zwraca numer id dodanej rezerwacji, lub w przypadku, gdy rezerwacja nie została dodana, zwracany jest liczba -1 (id rezerwacji jest większy od 0). Dostęp poprzez żądanie POST pod ścieżką http://localhost:8080/rsiproj2/resources/reservation. Obiekt przekazywany jest w ciele żądania, jako JSON

- **bool deleteReservationId(int reservationId)** – metoda udostępnia usuwanie rezerwacji, do metody przekazywane jest id rezerwacji, metoda zwraca wartość boolean, która mówi, czy lot o podanym id został usunięty (true), czy nie (false). Dostęp poprzez żądanie DELETE pod ścieżką http://localhost:8080/rsiproj2/resources/reservation/{reservationId}.

- **byte[] getReservationConfirmationPdf(int reservationId)** – metoda zwraca plik PDF danej rezerwacji, do metody przekazywane jest id rezerwacji. Dostęp poprzez żądanie GET pod ścieżką http://localhost:8080/rsiproj2/resources/reservation/{reservationId}/pdf.

### 2.2. Adres usługi

**http://localhost:8080/rsiproj2**

### 2.3. Specyfikacja wadl


```
▼<application xmlns="http://wadl.dev.java.net/2009/02">
   <doc xmlns:jersey="http://jersey.java.net/" jersey:generatedBy="Jersey: 2.28 2019-01-25 15:18:13"/>
   <doc xmlns:jersey="http://jersey.java.net/" jersey:hint="This is full WADL including extended resources.
   To get simplified WADL with users resources only do not use the query parameter detail. Link:
   http://localhost:8080/rsiproj2/resources/application.wadl"/>
  ▼<grammars>
   ▼<include href="application.wadl/xsd0.xsd">
      <doc title="Generated" xml:lang="en"/>
    </include>
   </grammars>
  ▼<resources base="http://localhost:8080/rsiproj2/resources/">
   ▼<resource path="javaee8">
    ▼<method id="ping" name="GET">
     ▼<response>
        <representation mediaType="*/*"/>
      </response>
     </method>
    ▼<method id="apply" name="OPTIONS">
     ▼<request>
        <representation mediaType="*/*"/>
      </request>
     ▼<response>
        <representation mediaType="application/vnd.sun.wadl+xml"/>
      </response>
      <jersey:extended xmlns:jersey="http://jersey.java.net/">true</jersey:extended>
     </method>
    ▼<method id="apply" name="OPTIONS">
     ▼<request>
        <representation mediaType="*/*"/>
      </request>
     ▼<response>
        <representation mediaType="text/plain"/>
      </response>
      <jersey:extended xmlns:jersey="http://jersey.java.net/">true</jersey:extended>
     </method>
    ▼<method id="apply" name="OPTIONS">
     ▼<request>
        <representation mediaType="*/*"/>
      </request>
     ▼<response>
        <representation mediaType="*/*"/>
      </response>
      <jersey:extended xmlns:jersey="http://jersey.java.net/">true</jersey:extended>
     </method>
    </resource>
   ▼<resource path="reservation">
    ▼<method id="addReservation" name="POST">
     ▼<request>
        <representation mediaType="application/json"/>
      </request>
     ▼<response>
        <representation mediaType="*/*"/>
      </response>
     </method>
```

```
<method id="apply" name="OPTIONS">
  <request>
    <representation mediaType="*/*"/>
  </request>
  <response>
    <representation mediaType="application/vnd.sun.wadl+xml"/>
  </response>
  <jersey:extended xmlns:jersey="http://jersey.java.net/">true</jersey:extended>
</method>
<method id="apply" name="OPTIONS">
  <request>
    <representation mediaType="*/*"/>
  </request>
  <response>
    <representation mediaType="text/plain"/>
  </response>
  <jersey:extended xmlns:jersey="http://jersey.java.net/">true</jersey:extended>
</method>
<method id="apply" name="OPTIONS">
  <request>
    <representation mediaType="*/*"/>
  </request>
  <response>
    <representation mediaType="*/*"/>
  </response>
  <jersey:extended xmlns:jersey="http://jersey.java.net/">true</jersey:extended>
</method>
<resource path="/{reservationId}">
  <param xmlns:xs="http://www.w3.org/2001/XMLSchema" name="reservationId" style="template"
  type="xs:int"/>
  <method id="getReservationById" name="GET">
    <response>
      <representation mediaType="application/json"/>
    </response>
  </method>
  <method id="deleteReservation" name="DELETE">
    <response>
      <representation mediaType="*/*"/>
    </response>
  </method>
  <method id="apply" name="OPTIONS">
    <request>
      <representation mediaType="*/*"/>
    </request>
    <response>
      <representation mediaType="application/vnd.sun.wadl+xml"/>
    </response>
    <jersey:extended xmlns:jersey="http://jersey.java.net/">true</jersey:extended>
  </method>
  <method id="apply" name="OPTIONS">
    <request>
      <representation mediaType="*/*"/>
    </request>
    <response>
      <representation mediaType="text/plain"/>
    </response>
    <jersey:extended xmlns:jersey="http://jersey.java.net/">true</jersey:extended>
  </method>
```

```
    <method id="apply" name="OPTIONS">
      <request>
        <representation mediaType="*/*"/>
      </request>
      <response>
        <representation mediaType="*/*"/>
      </response>
      <jersey:extended xmlns:jersey="http://jersey.java.net/">true</jersey:extended>
    </method>
  </resource>
  <resource path="/{reservationId}/pdf">
    <param xmlns:xs="http://www.w3.org/2001/XMLSchema" name="reservationId" style="template"
    type="xs:int"/>
    <method id="getReservationConfirmationPdf" name="GET">
      <response>
        <representation mediaType="*/*"/>
      </response>
    </method>
    <method id="apply" name="OPTIONS">
      <request>
        <representation mediaType="*/*"/>
      </request>
      <response>
        <representation mediaType="application/vnd.sun.wadl+xml"/>
      </response>
      <jersey:extended xmlns:jersey="http://jersey.java.net/">true</jersey:extended>
    </method>
    <method id="apply" name="OPTIONS">
      <request>
        <representation mediaType="*/*"/>
      </request>
      <response>
        <representation mediaType="text/plain"/>
      </response>
      <jersey:extended xmlns:jersey="http://jersey.java.net/">true</jersey:extended>
    </method>
    <method id="apply" name="OPTIONS">
      <request>
        <representation mediaType="*/*"/>
      </request>
      <response>
        <representation mediaType="*/*"/>
      </response>
      <jersey:extended xmlns:jersey="http://jersey.java.net/">true</jersey:extended>
    </method>
  </resource>
</resource>
<resource path="flights">
  <method id="getFlights" name="GET">
    <response>
      <representation mediaType="application/json"/>
    </response>
  </method>
  <method id="addFlight" name="POST">
    <request>
      <representation mediaType="application/json"/>
    </request>
    <response>
      <representation mediaType="application/json"/>
    </response>
  </method>
```

```
<method id="apply" name="OPTIONS">
  <request>
    <representation mediaType="*/*"/>
  </request>
  <response>
    <representation mediaType="application/vnd.sun.wadl+xml"/>
  </response>
  <jersey:extended xmlns:jersey="http://jersey.java.net/">true</jersey:extended>
</method>
<method id="apply" name="OPTIONS">
  <request>
    <representation mediaType="*/*"/>
  </request>
  <response>
    <representation mediaType="text/plain"/>
  </response>
  <jersey:extended xmlns:jersey="http://jersey.java.net/">true</jersey:extended>
</method>
<method id="apply" name="OPTIONS">
  <request>
    <representation mediaType="*/*"/>
  </request>
  <response>
    <representation mediaType="*/*"/>
  </response>
  <jersey:extended xmlns:jersey="http://jersey.java.net/">true</jersey:extended>
</method>
<resource path="/{flightId}">
  <param xmlns:xs="http://www.w3.org/2001/XMLSchema" name="flightId" style="template" type="xs:int"/>
  <method id="updateFlight" name="PUT">
    <request>
      <representation mediaType="application/json"/>
    </request>
    <response>
      <representation mediaType="*/*"/>
    </response>
  </method>
  <method id="deleteFlight" name="DELETE">
    <response>
      <representation mediaType="*/*"/>
    </response>
  </method>
  <method id="getFlight" name="GET">
    <response>
      <representation mediaType="application/json"/>
    </response>
  </method>
  <method id="apply" name="OPTIONS">
    <request>
      <representation mediaType="*/*"/>
    </request>
    <response>
      <representation mediaType="application/vnd.sun.wadl+xml"/>
    </response>
    <jersey:extended xmlns:jersey="http://jersey.java.net/">true</jersey:extended>
  </method>
```

```
      <method id="apply" name="OPTIONS">
        <request>
          <representation mediaType="*/*"/>
        </request>
        <response>
          <representation mediaType="text/plain"/>
        </response>
        <jersey:extended xmlns:jersey="http://jersey.java.net/">true</jersey:extended>
      </method>
      <method id="apply" name="OPTIONS">
        <request>
          <representation mediaType="*/*"/>
        </request>
        <response>
          <representation mediaType="*/*"/>
        </response>
        <jersey:extended xmlns:jersey="http://jersey.java.net/">true</jersey:extended>
      </method>
    </resource>
  </resource>
  <resource path="application.wadl">
    <method id="getWadl" name="GET">
      <response>
        <representation mediaType="application/vnd.sun.wadl+xml"/>
        <representation mediaType="application/xml"/>
      </response>
      <jersey:extended xmlns:jersey="http://jersey.java.net/">true</jersey:extended>
    </method>
    <method id="apply" name="OPTIONS">
      <request>
        <representation mediaType="*/*"/>
      </request>
      <response>
        <representation mediaType="text/plain"/>
      </response>
      <jersey:extended xmlns:jersey="http://jersey.java.net/">true</jersey:extended>
    </method>
    <method id="apply" name="OPTIONS">
      <request>
        <representation mediaType="*/*"/>
      </request>
      <response>
        <representation mediaType="*/*"/>
      </response>
      <jersey:extended xmlns:jersey="http://jersey.java.net/">true</jersey:extended>
    </method>
    <resource path="{path}">
      <param xmlns:xs="http://www.w3.org/2001/XMLSchema" name="path" style="template" type="xs:string"/>
      <method id="getExternalGrammar" name="GET">
        <response>
          <representation mediaType="application/xml"/>
        </response>
        <jersey:extended xmlns:jersey="http://jersey.java.net/">true</jersey:extended>
      </method>
      <method id="apply" name="OPTIONS">
        <request>
          <representation mediaType="*/*"/>
        </request>
        <response>
          <representation mediaType="text/plain"/>
        </response>
        <jersey:extended xmlns:jersey="http://jersey.java.net/">true</jersey:extended>
      </method>
      <method id="apply" name="OPTIONS">
        <request>
          <representation mediaType="*/*"/>
        </request>
        <response>
          <representation mediaType="*/*"/>
        </response>
        <jersey:extended xmlns:jersey="http://jersey.java.net/">true</jersey:extended>
      </method>
      <jersey:extended xmlns:jersey="http://jersey.java.net/">true</jersey:extended>
    </resource>
    <jersey:extended xmlns:jersey="http://jersey.java.net/">true</jersey:extended>
  </resource>
</resources>
</application>
```

### 2.4. Szczegółowy opis wybranych udostępnianych operacji

- CreateReservation – tworzy rezerwację na dany lot, dla podanej liczby osób.

Dane wejściowe (JSON):

- Obiekt AddReservationDTO, podany w ciele żądania, zawierający pola:
  - int id – pole niewymagane do podania,
  - int flightId – id lotu, na który ma się dokonać rezerwacja,
  - List<PersonDTO> people – lista osób z danej rezerwacji, PersonDTO zawiera pola:
    - int id – identyfikator (przekazana wartość nie zostanie ustawiona),
    - String FirstName – imie,
    - String Surname – nazwisko,
    - String BirthDate – data urodzenia,
    - reservationId – id rezerwacji (przekazana wartość nie zostanie ustawiona).

Dane wyjściowe:

- IdRezerwacji – w przypadku gdy wartość jest większa od o 0, zwracane jest id dodanej rezerwacji, a gdy wartość jest równa -1 oznacza niepowodzenie.

- GetReservationConfirmationPdf – zwraca tablicę bajtów pliku PDF

Dane wejściowe (JSON):

- reservationId – id rezerwacji, dla którego chcemy potwierdzenie PDF, podany w ścieżce.

Dane wyjściowe:

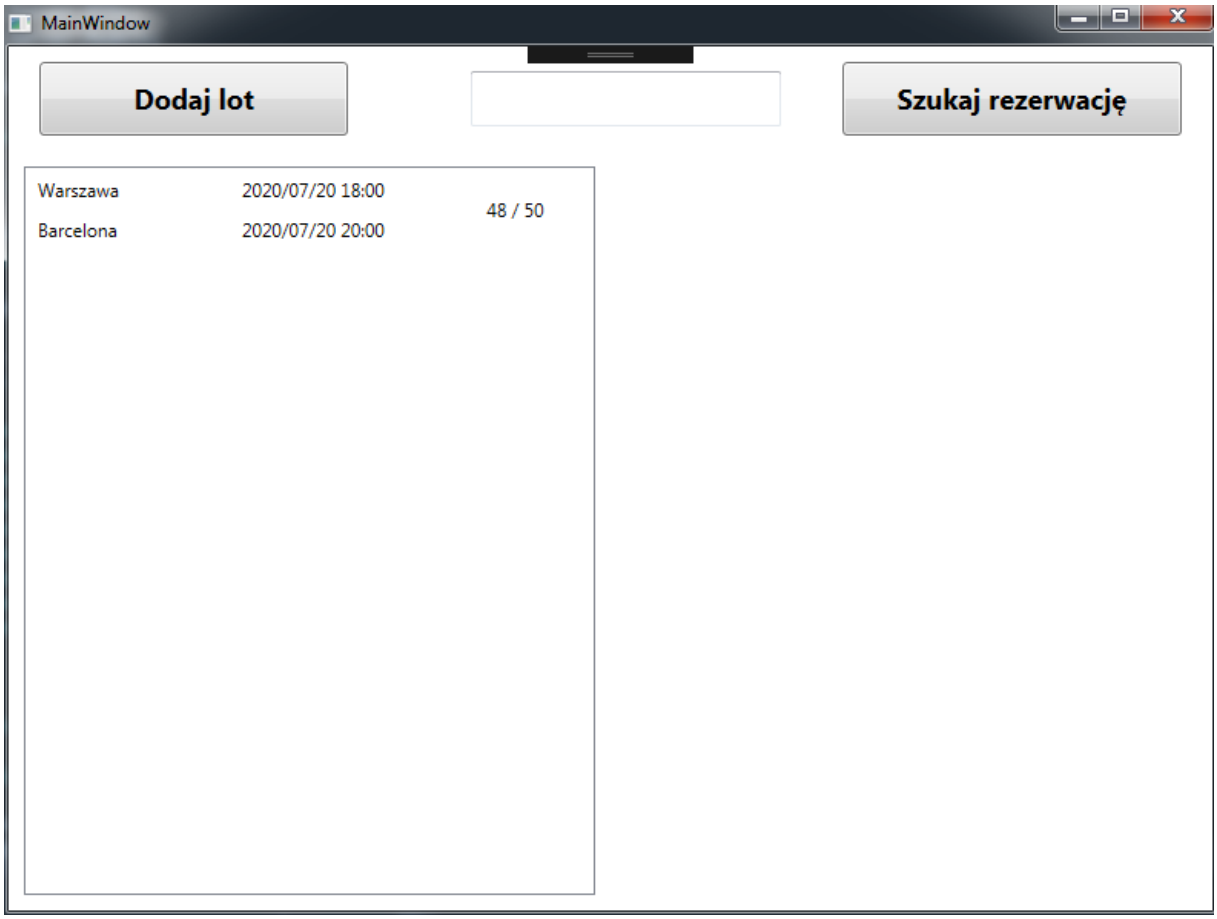
- tablica bajtów pliku PDF.

# 3. APLIKACJA KLIENTA

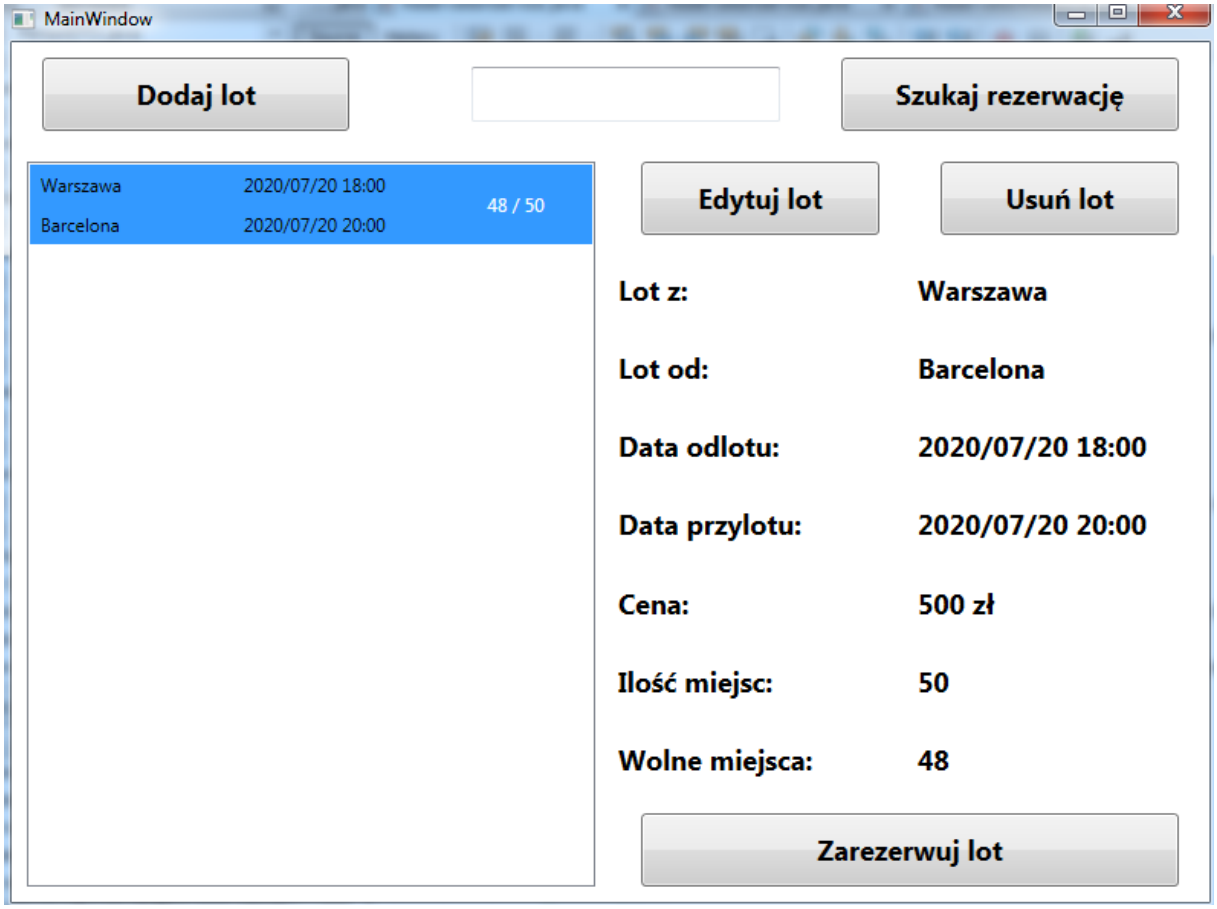
## 3.1. Szczegóły dotyczące aplikacji

Aplikacja klienta została napisana w języku C# za pomocą technologii WPF. Jest to aplikacja okienkowa, która dostarcza interfejs graficzny użytkownikowi. Poniżej został przedstawiony cały interfejs graficzny aplikacji:

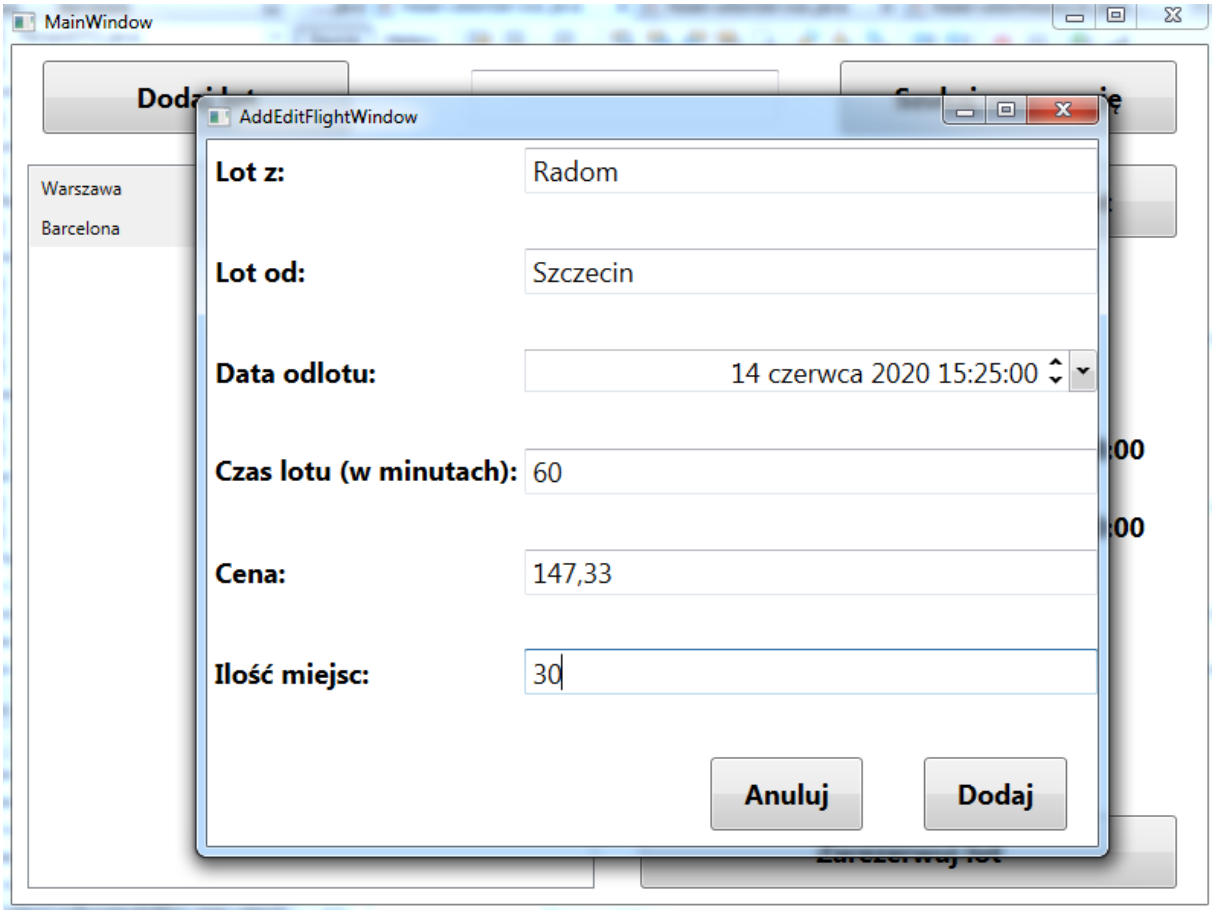
- **Strona główna** – po uruchomieniu aplikacji zostajemy przeniesieni do widoku głównego, w którym znajduje się lista lotów, na liście podane są informacje (od góry do dołu i od lewej do prawej): miasto startowe lotu, miasto docelowe lotu, data odlotu, data przylotu, liczba wolnych miejsc i pojemność samolotu. Powyżej listy znajduje się przycisk, który wyświetli nam okno z dodawaniem lotu, po prawej znajduje się przycisk, który wyszuka rezerwację o podanym id w wejściu tekstowym znajdującym się obok.

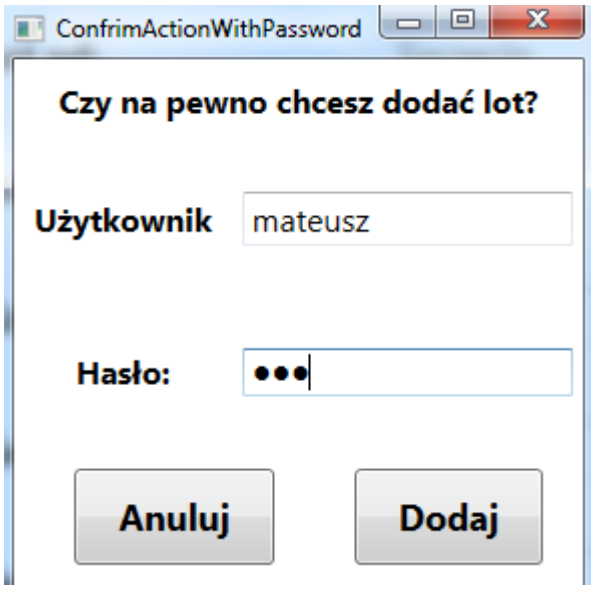


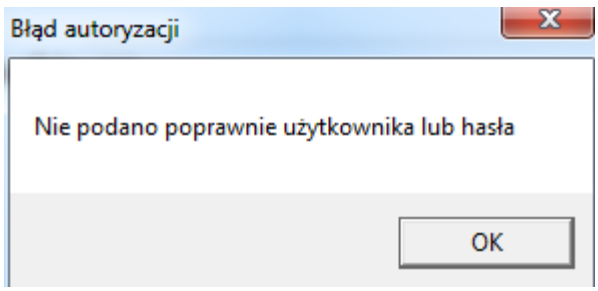
- **Strona główna naciśnięcie na dany lot z listy** – po naciśnięciu na dany lot zostaną wyświetlone szczegóły lotu po prawej strony od listy, dodatkowo pojawią się kolejne przyciski: edytuj lot, usuń lot i zarezerwuj lot. Naciśnięcie na przycisk edytuj lot, spowoduje wyświetlenie okna z edycją lotu. Naciśnięcie na przycisk usuń lot, spowoduje wyświetlenie okna z potwierdzeniem, czy chcemy usunąć dany lot. Naciśnięcie na przycisk zarezerwuj lot, spowoduje wyświetlenie okna z rezerwacją lotu.



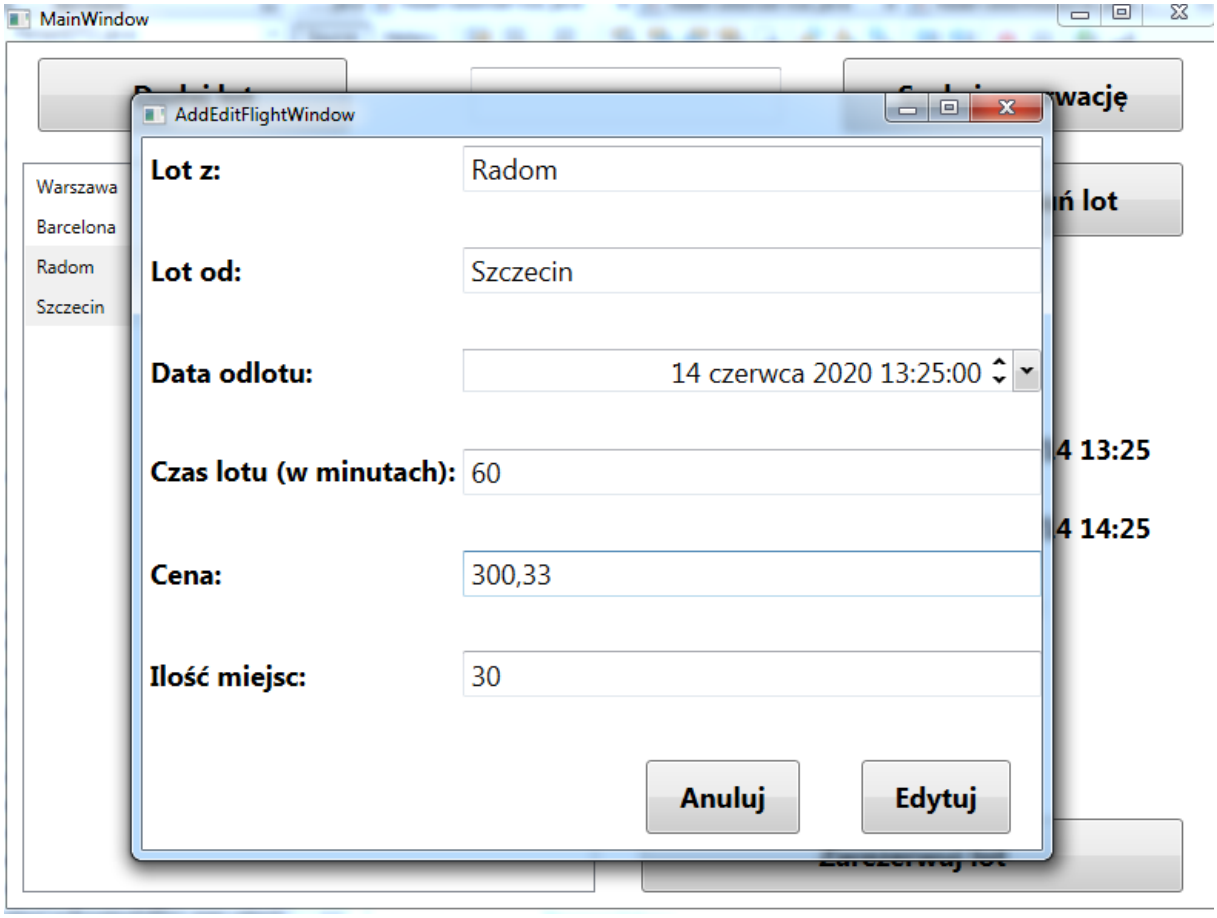
- **Dodaj lot** – po naciśnięciu na przycisk dodaj lot wyświetli nam się okno z formularzem dodawania lotu. Po wypełnieniu formularza, możemy dodać lot do bazy naciskając przycisk dodaj, bądź anulować. W przypadku wpisania niepoprawnych wartości, zostaniemy poinformowani. Po naciśnięciu na przycisk dodaj, będziemy musieli podać poprawnie użytkownika i hasło. W przypadku podania nieprawidłowego użytkownika bądź hasła, zostanie wyświetlony komunikat.



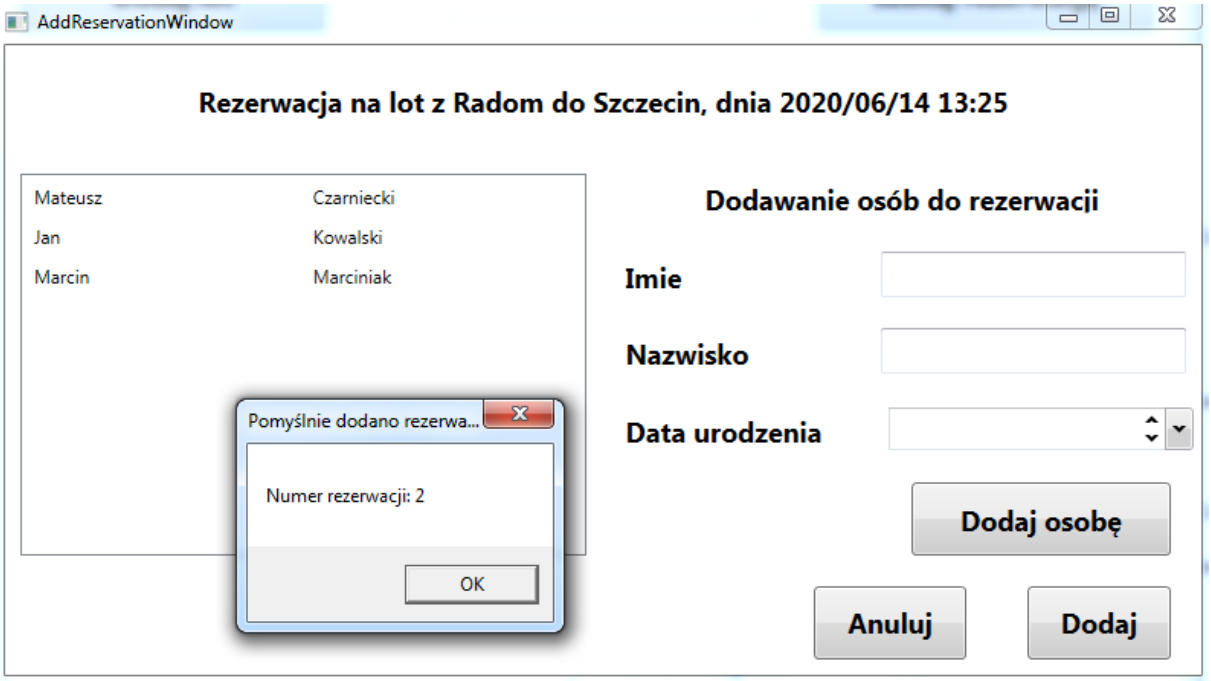



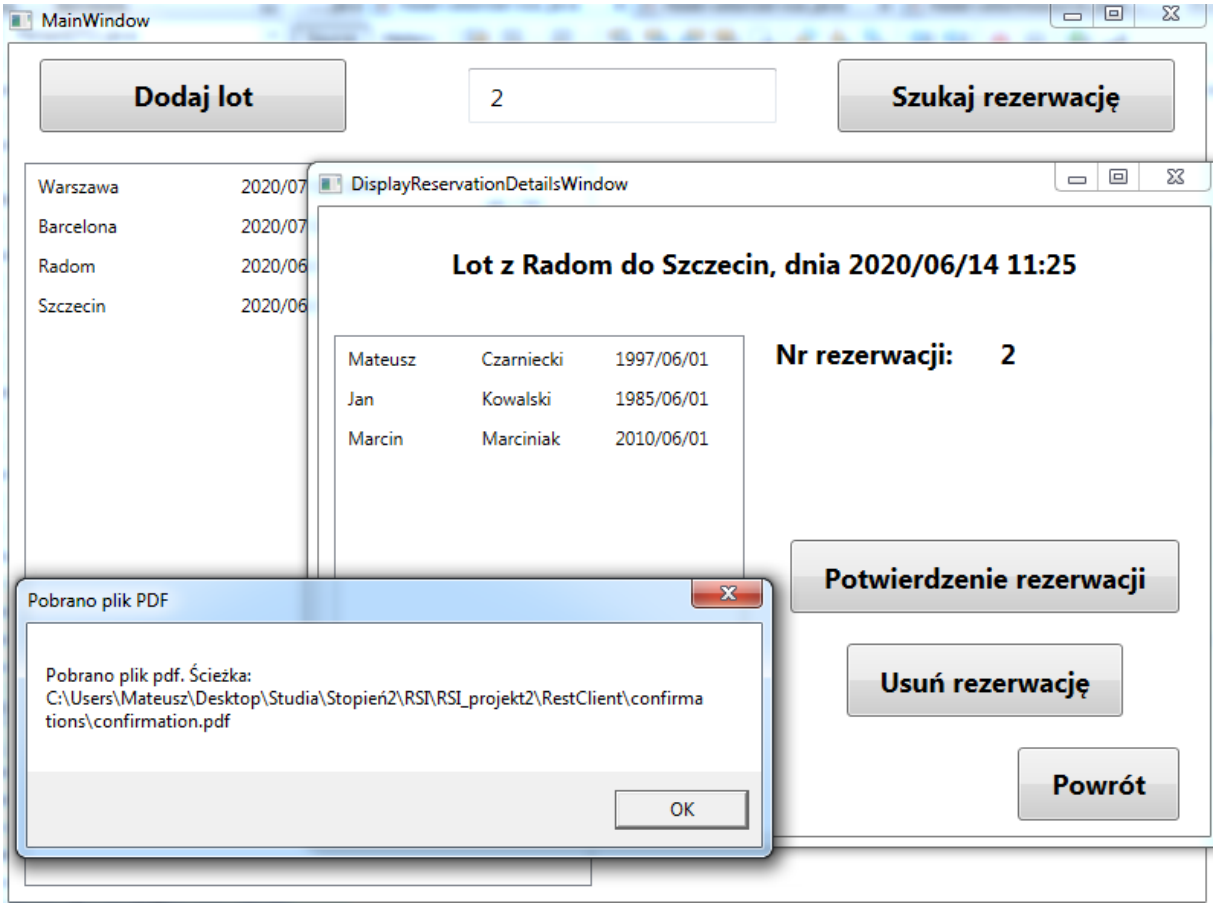
- **Edytuj lot** – po naciśnięciu na przycisk edytuj lot, wyświetlony zostanie okno z wypełnionym formularzem. Możemy zmienić szczegóły danego lotu. Po naciśnięciu na przycisk edytuj, zostanie zedytowany dany lot. Naciśnięcie na przycisk anuluj, spowoduje anulowanie edycji. Po naciśnięciu na przycisk edytuj, będziemy musieli podać poprawnie użytkownika i hasło. W przypadku podania nieprawidłowego użytkownika bądź hasła, zostanie wyświetlony komunikat.



- **Zarezerwuj lot** – po naciśnięciu na przycisk zarezerwuj lot, wyświetlone zostanie okno z formularzem rezerwacji miejsc na dany lot. Po prawej stronie jest formularz w którym możemy dodawać osoby do rezerwacji. Osoby będą wyświetlane na liście po lewej stronie. Dodajemy osobę poprzez przycisk dodaj osobę, po czym dodawana będzie osoba oraz czyszczony zostanie formularz. Po naciśnięciu na daną osobę, pojawi się przycisk usuń osobę, który pozwoli nam na usunięcie osoby z listy. Naciśnięcie na przycisk dodaj spowoduje dodanie rezerwacji do bazy. W przypadku, gdy ilość dostępnych miejsc na dany lot zostanie przekroczona dostaniemy komunikat. W przeciwnym wypadku, gdy stworzona zostanie rezerwacja, wyświetli nam się okienko z informacją o id dodanej rezerwacji.



- **Wyszukaj rezerwację** – po podaniu id rezerwacji i naciśnięciu na przycisk szukaj rezerwację, pojawi się okno z informacjami o danej rezerwacji. Dodatkowo pojawią się 2 przyciski. Naciśnięcie przycisku potwierdzenie rezerwacji spowoduje pobranie pliku PDF oraz wyświetlenie ścieżki do danego pliku PDF. Naciśnięcie przycisku usuń rezerwację spowoduje usunięcie danej rezerwacji, po ówczesnym potwierdzeniu usunięcia.




**Potwierdzenie rezerwacji**

Nr rezerwacji: 2

Lot z: Radom

Lot do: Szczecin

Data odlotu: Sun Jun 14 13:25:00 CEST 2020

Data przylotu: Sun Jun 14 14:25:00 CEST 2020

Osoby:

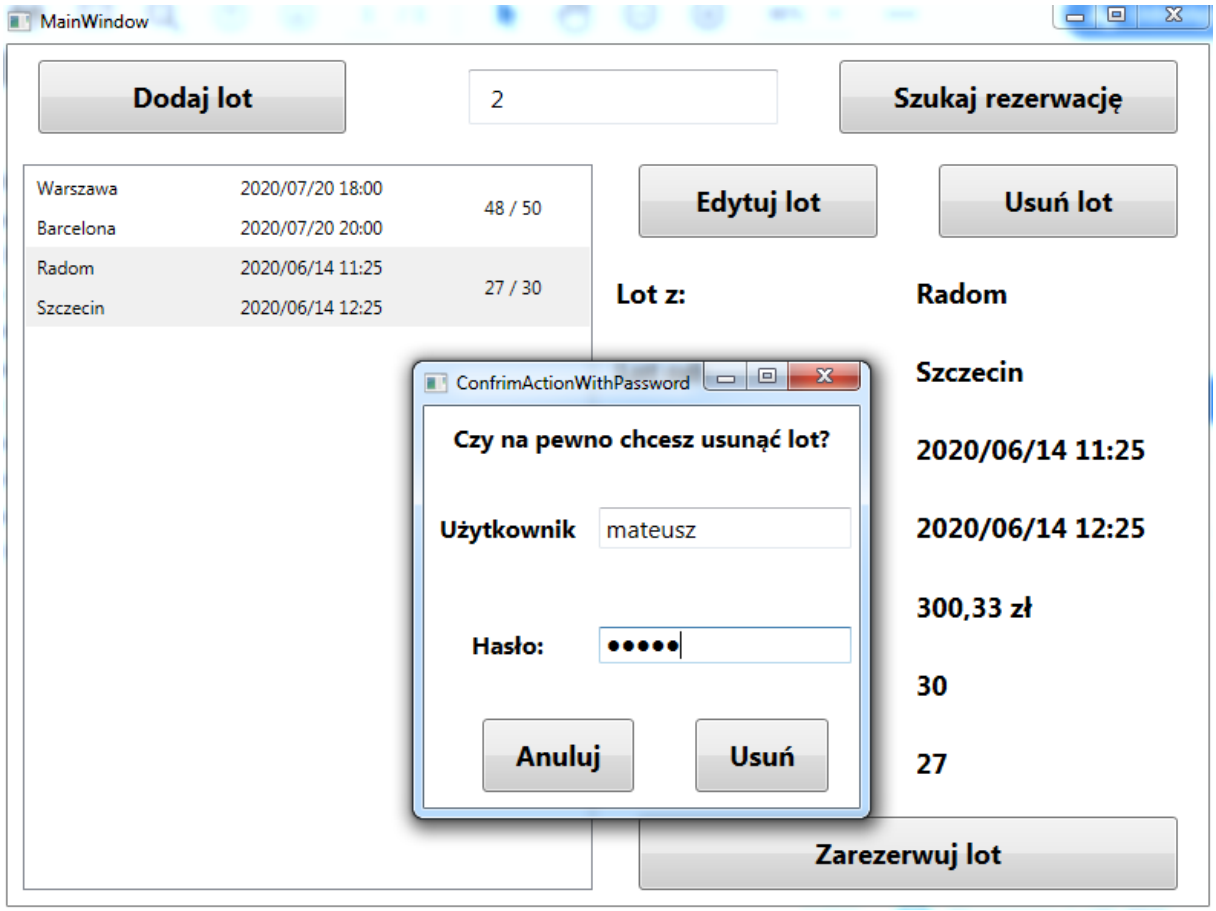
Mateusz Czarniecki, urodzony Sun Jun 01 15:43:28 CEST 1997

Jan Kowalski, urodzony Sat Jun 01 15:43:41 CEST 1985

Marcin Marciniak, urodzony Tue Jun 01 15:43:47 CEST 2010

Cena: 900.99

- **Usuń lot** – po naciśnięciu na przycisk usuń lot wyskoczy nam okienko z potwierdzeniem, czy chcemy usunąć dany lot. Po naciśnięciu na przycisk usuń, będziemy musieli podać hasło. Przed wywołaniem funkcji zostanie wykonany handler sprawdzający poprawność hasła. W przypadku podania nieprawidłowego hasła, zostanie wyświetlony komunikat.




-