**Обучаващ център: ПГЕЕ Апостол Арнаудов - Русе**

**Водещ преподавател: Агнеса Танчева**

**Група: Група 2 - Разград**

**Училище: НПТГ "Ш.Петьофи" гр.Разград**

**Документация на проекта : Мениджър на самолетни полети**

Съдържание

[1. Изисквания. 1](#_Toc68969653)

[2. Етапи от разработката на проекта: 2](#_Toc68969654)

[3. Архитектура на проекта. 2](#_Toc68969655)

[4. Структура на базата от данни. 3](#_Toc68969656)

[5. Ръководство за потребителя. 3](#_Toc68969657)

[*Как се влиза в потребителски акаунт?* 4](#_Toc68969658)

[*Как се прави резервация на билети?* 4](#_Toc68969659)

[*Как мога да променя телефонния ми номер?* 7](#_Toc68969660)

[*Как мога да променя моята парола?* 8](#_Toc68969661)

[*Как се страницира?* 10](#_Toc68969662)

[*Как се филтрира?* 10](#_Toc68969663)

[6. Потребителски акаунти и роли. 11](#_Toc68969664)

[7. Функционалности на уеб приложението: 11](#_Toc68969665)

[8. Използвани инструменти и технологии: 12](#_Toc68969666)

[9. Участници в проекта и техният принос към него: 13](#_Toc68969667)

[10. XML документация на подпроекта „Data“. 13](#_Toc68969668)

[11. XML документация на подпроекта „Flights.Test“. 28](#_Toc68969669)

[12. XML документация на подпроекта „Web“. 36](#_Toc68969670)

# Изисквания.

Мениджърът на самолетни полети (Flight Manager) е система, която служи за резервиране на самолетни билети и управление на полети. Тя дава възможност да се проследят вече отминали полети заедно с информация относно лицата, които са пътували. Изисква се да има два компонента: уеб приложение и база данни, с която да осъществява комуникация.

**Функционалности:**

Потребителят трябва да има: потребителско име, парола, имейл, собствено име, фамилно име, ЕГН, адрес, телефонен номер и роля (като администраторска роля има само първия създаден потребител, а всеки следващ получава ролята на служител).

Полети – един полет трябва да има: локация – от, лоцакия – за, дата и час на излитане, дата и час на кацане, вид на самолета, уникален номер на самолета, име на пилот, капацитет пътници (брой свободни обикновенни места на самолета) и капацитет бизнес класа (брой свободни бизнес места на самолета).

Резервации - всеки потребител (без вход) може да изпраща заявка за резервация за даден полет като трябва да посочи за всеки пътник (за когото иска да закупи билет): собствено име, бащино име, фамилно име, ЕГН, телефонен номер, националност и тип на билета (бизнес класа или обикновен).

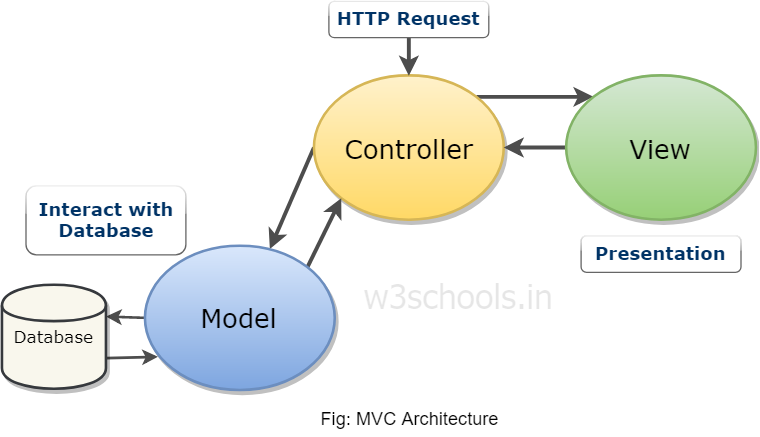
**Допълнителни изисквания:**

Всяка форма в уеб приложението да бъде валидирана. Помислете за случаите, в които това ще е нужно.

# Етапи от разработката на проекта:

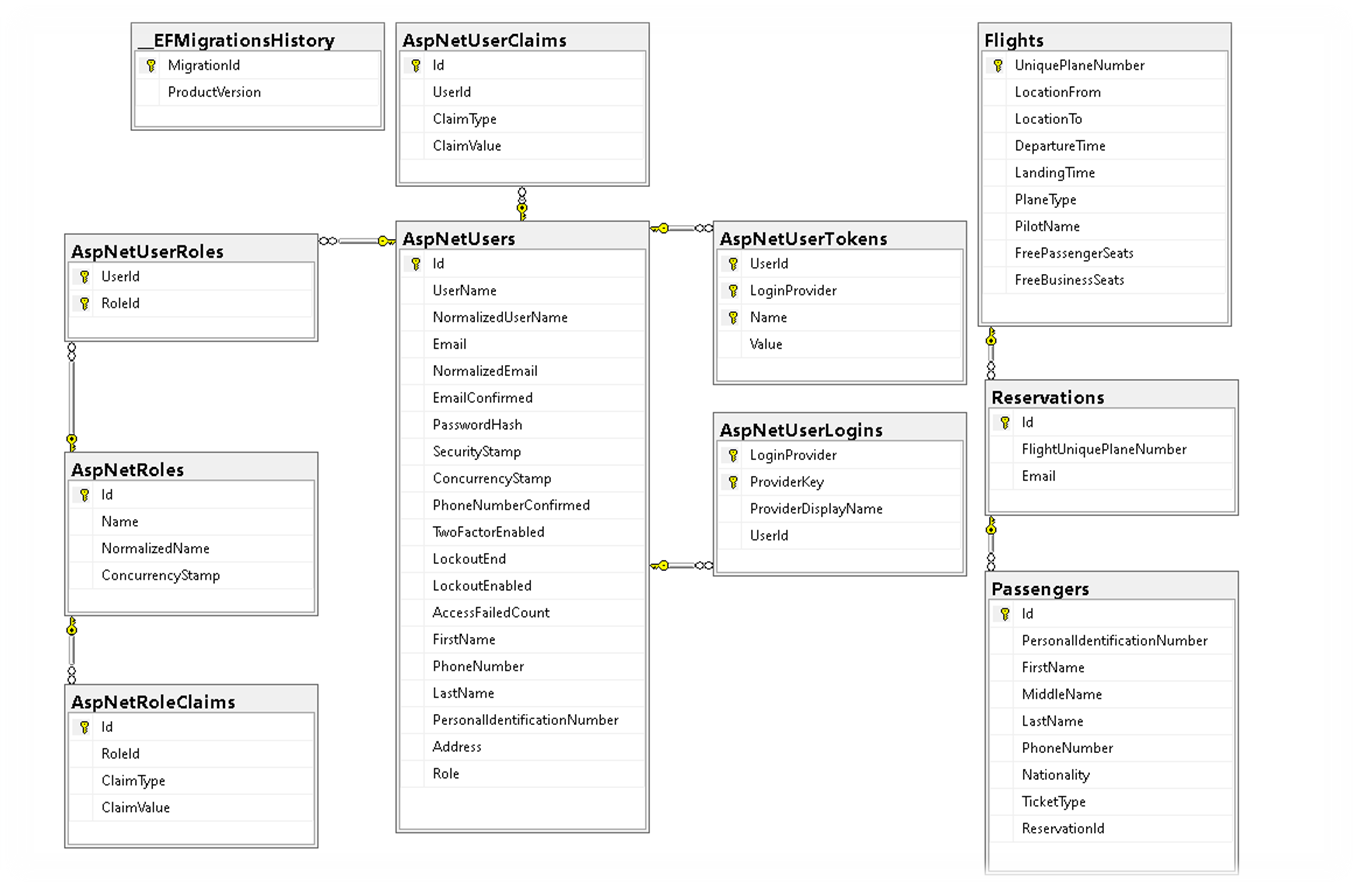
* Анализ на софтуерните изисквания;
* Планиране на етапите на разработка;
* Приготвяне на инструментите за работа;
* Разпределяне на задачите;
* Създаване на базата от данни и валидация на данните;
* Имплементация на функционалности за приложението;
* Имплементация на автоматизирани тестове;
* Документиране на проекта;
* Представяне и защита на проекта.

# Архитектура на проекта.



Проектът е разделен на 3 части. Първият подпроект съдържа предимно информация за базата от данни. Вторият подпроект служи за валидационни тестове на данните с Nunit. Третият подпроект е уеб приложението, което е ASP.NET CORE проект с MVC структура.

# Структура на базата от данни.



За имплементацията на автентикация и оторизация са използвани готовите шаблони на ASP.NET Core Identity. IdentityUser класът е разширен, за да отговаря на изискванията на проекта. Базата от данни е изградена по Code-First метода.

Останалите таблици са:

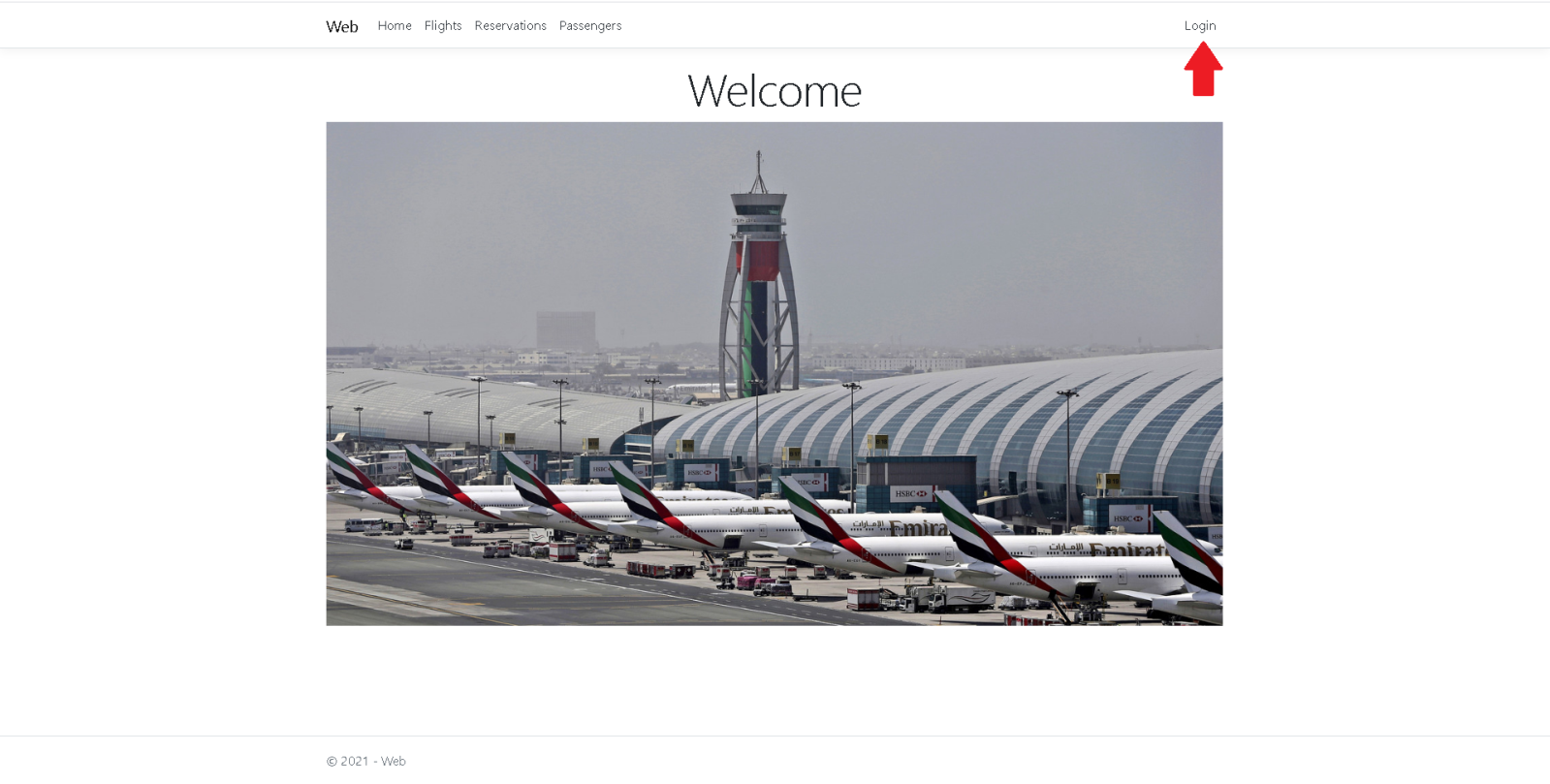
* Flights – съдържа информация за самолетните полети;
* Reservations – съдържа информация за резервираните самолетни билети;
* Passengers – съдържа информация за самолетните пътници.

# Ръководство за потребителя.

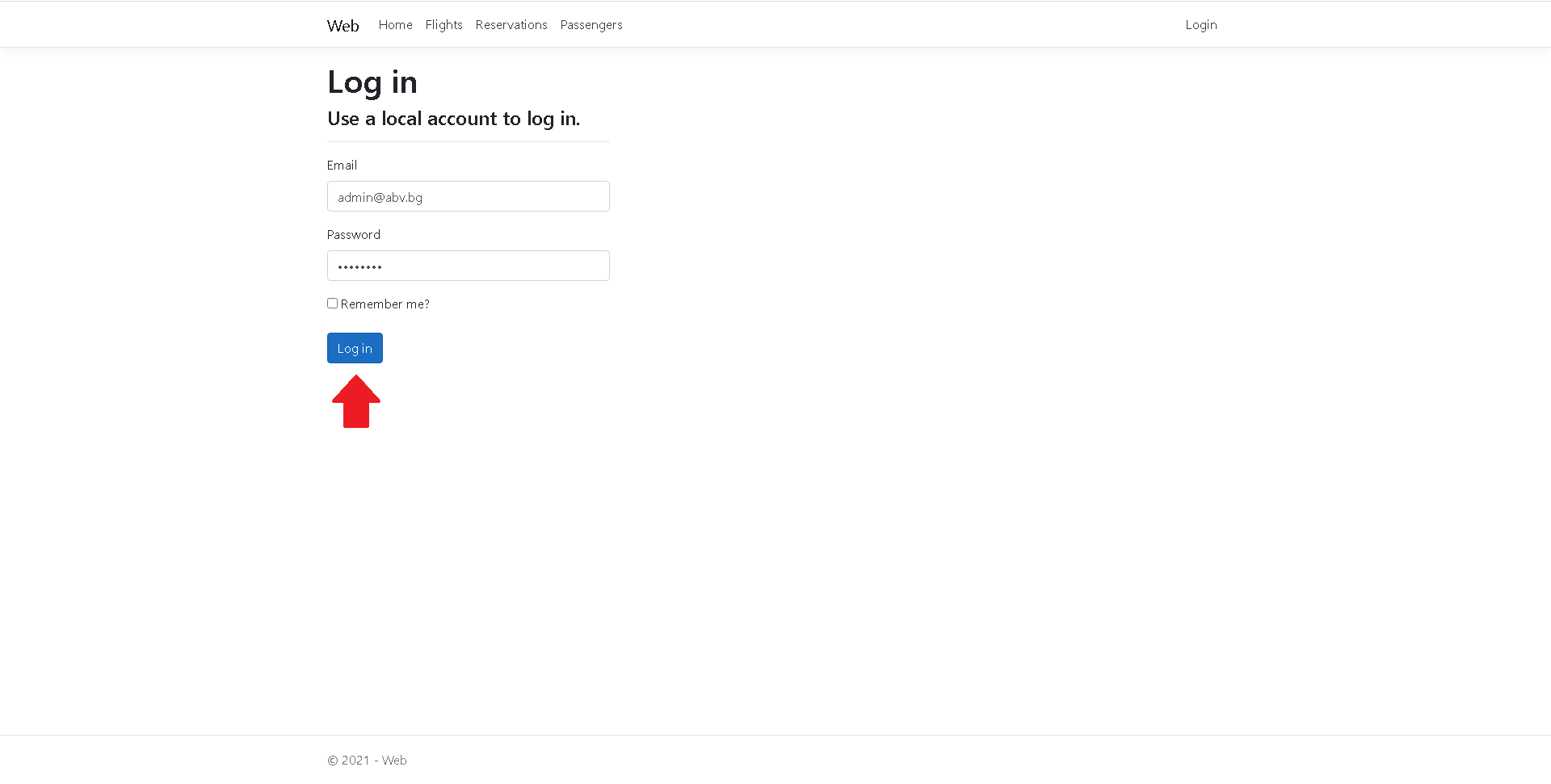
**Само администраторът може да прави профили.**

### *Как се влиза в потребителски акаунт?*

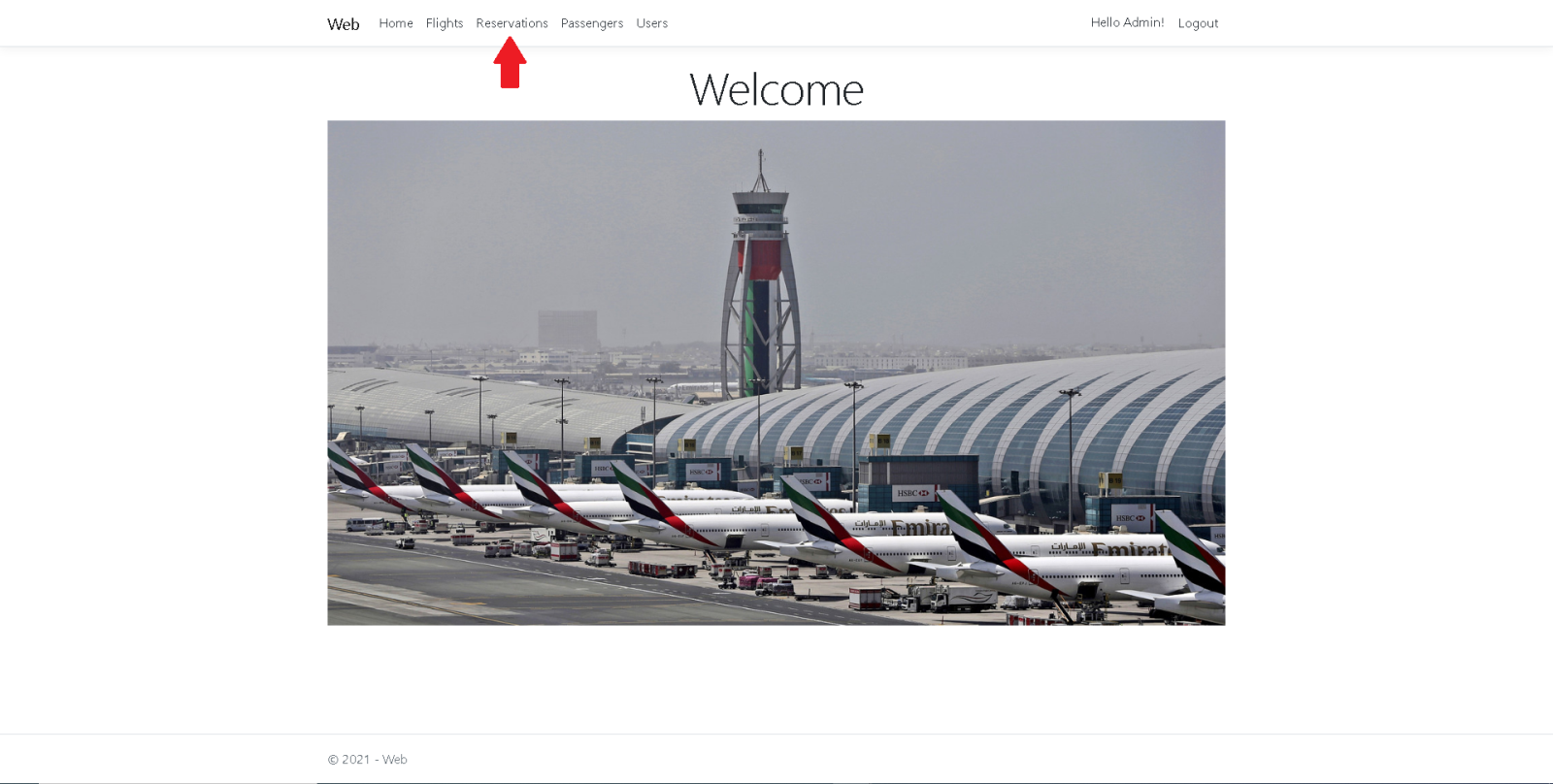
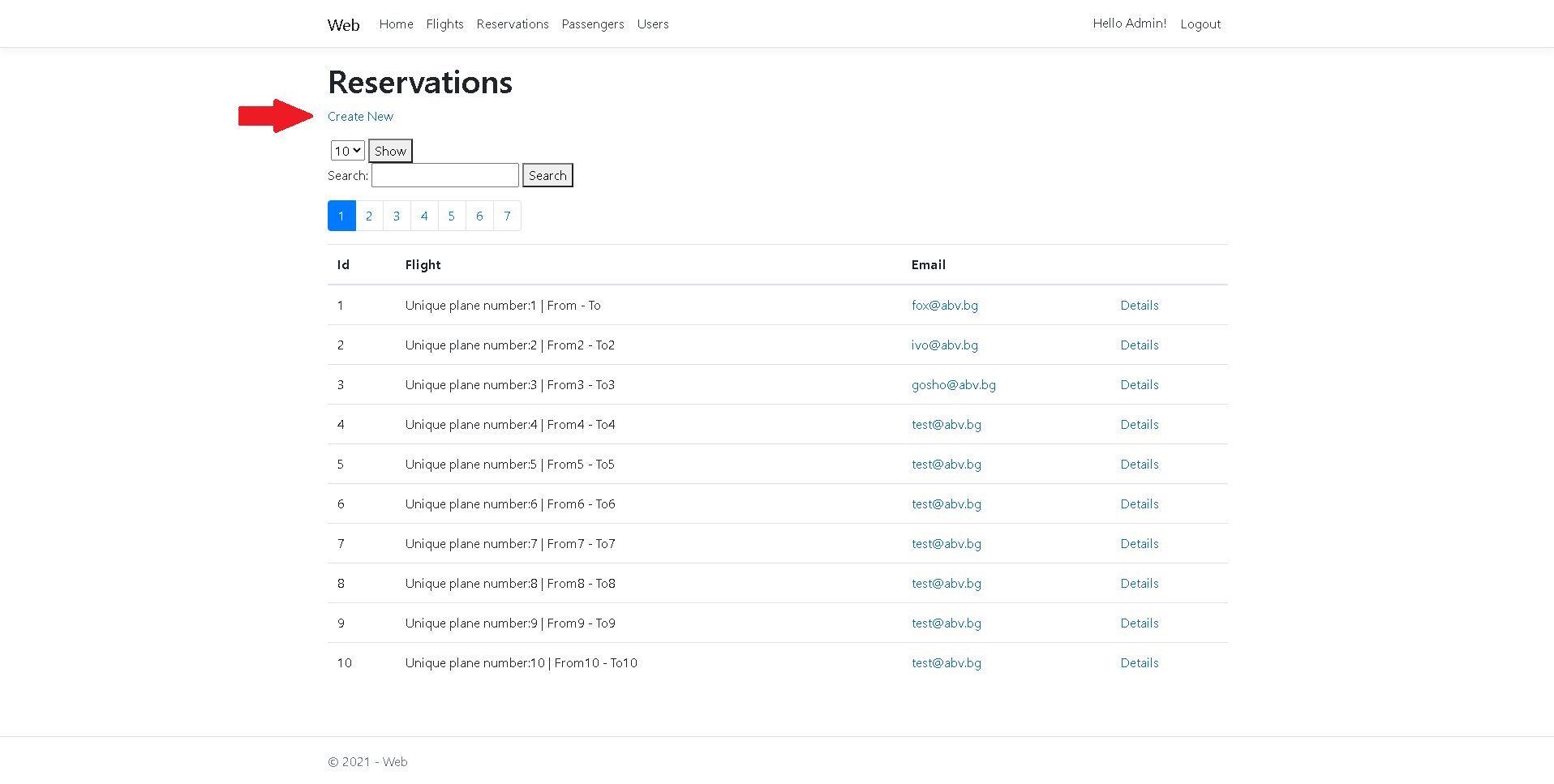
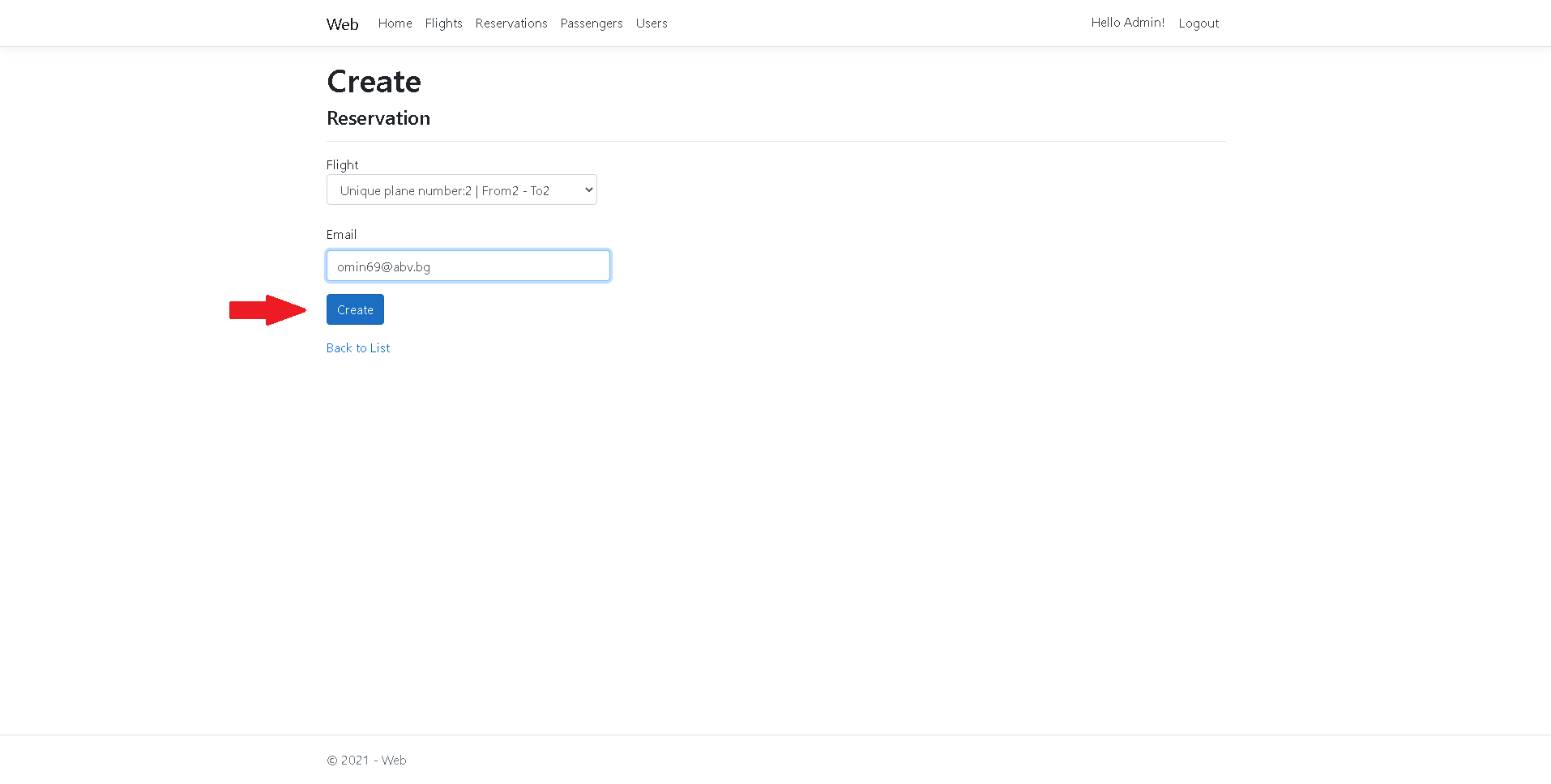
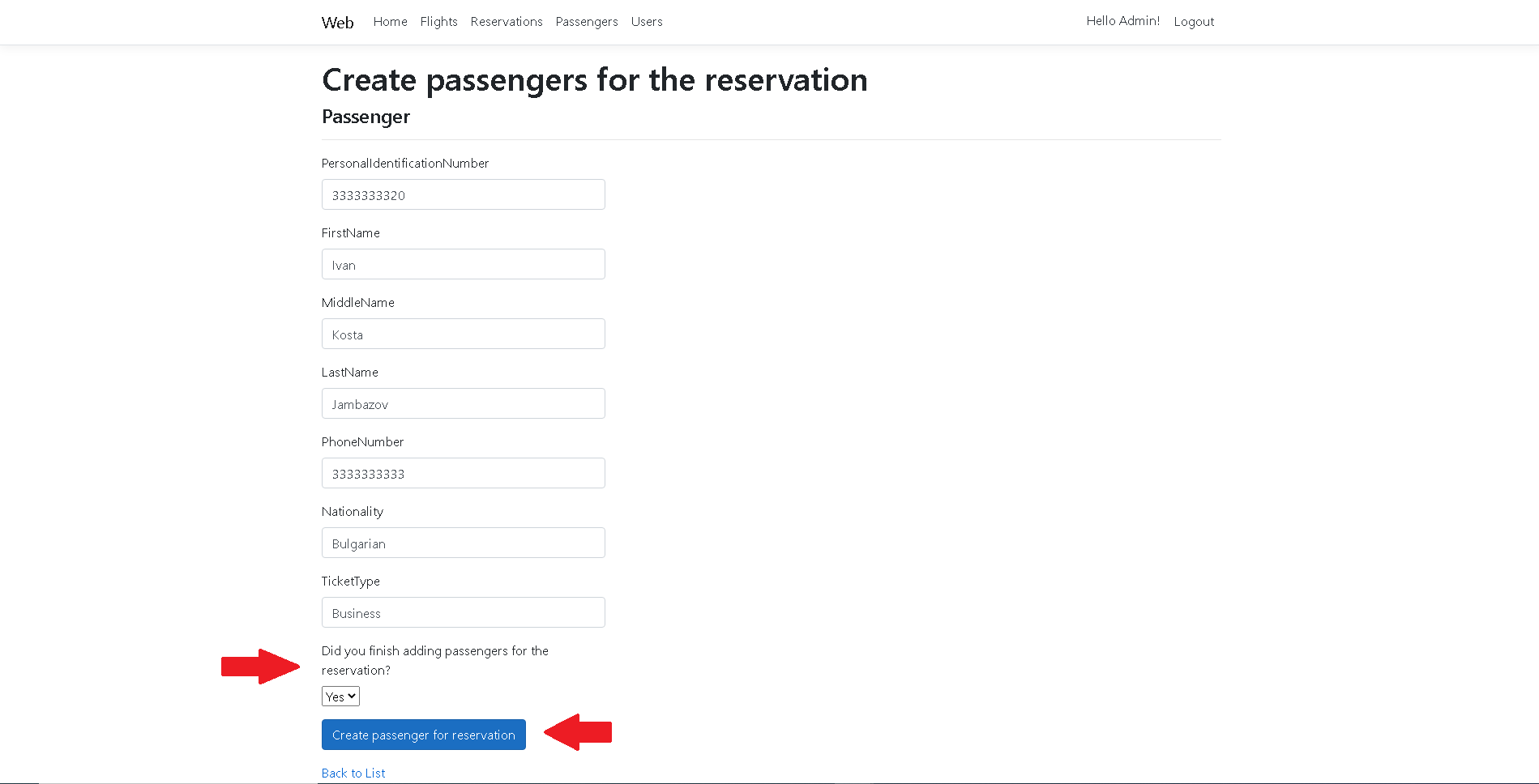
1. Кликвате „Login“, за да влезете във вашия профил. Ако нямате такъв, трябва администраторът да ви създаде профил.

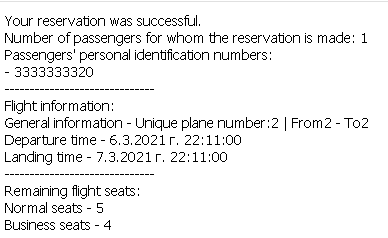


1. След като отворите „Login“ страницата, въведе вашите имейл и парола. После натиснете бутона „Log in“.

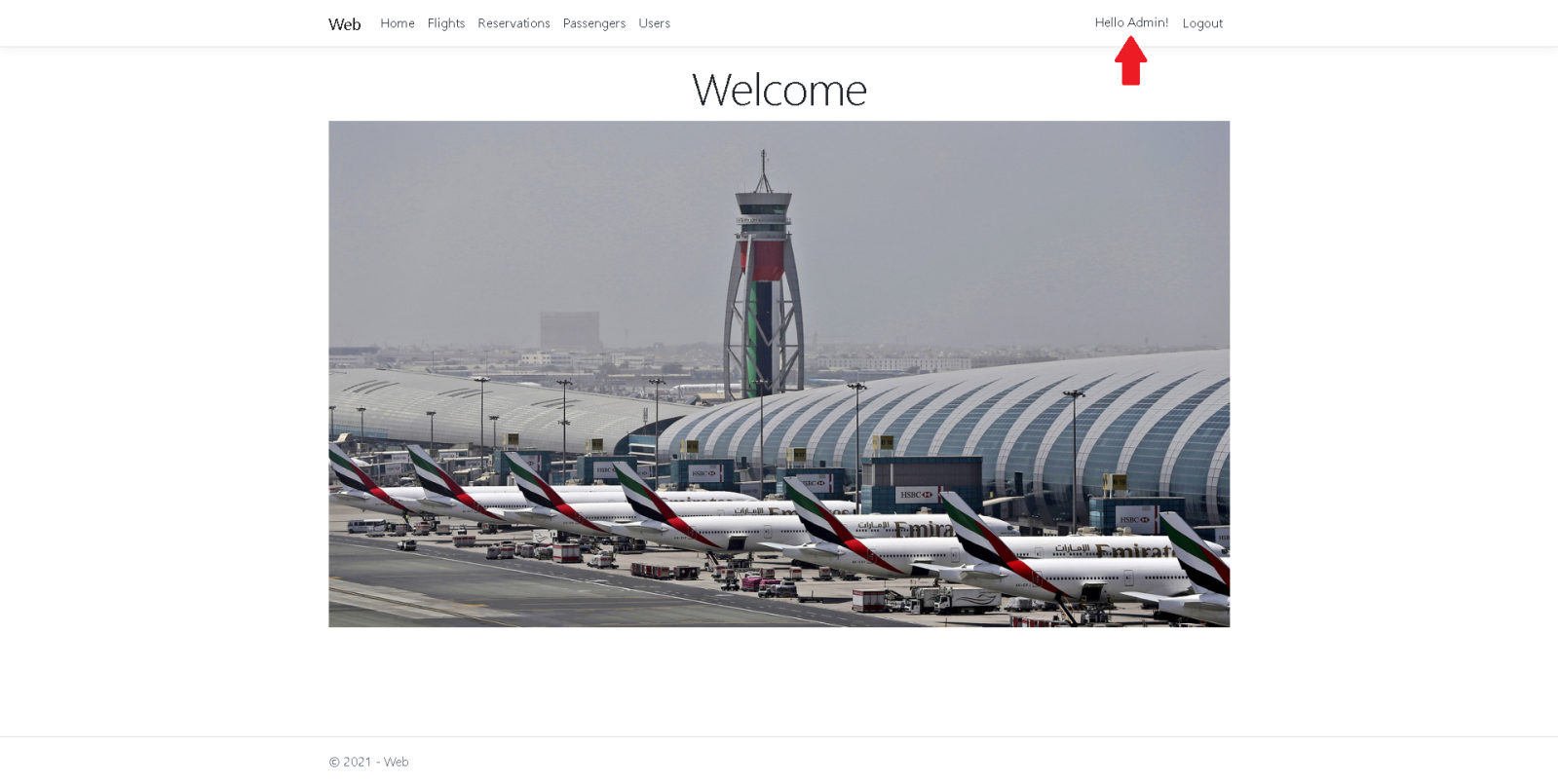


### *Как се прави резервация на билети?*

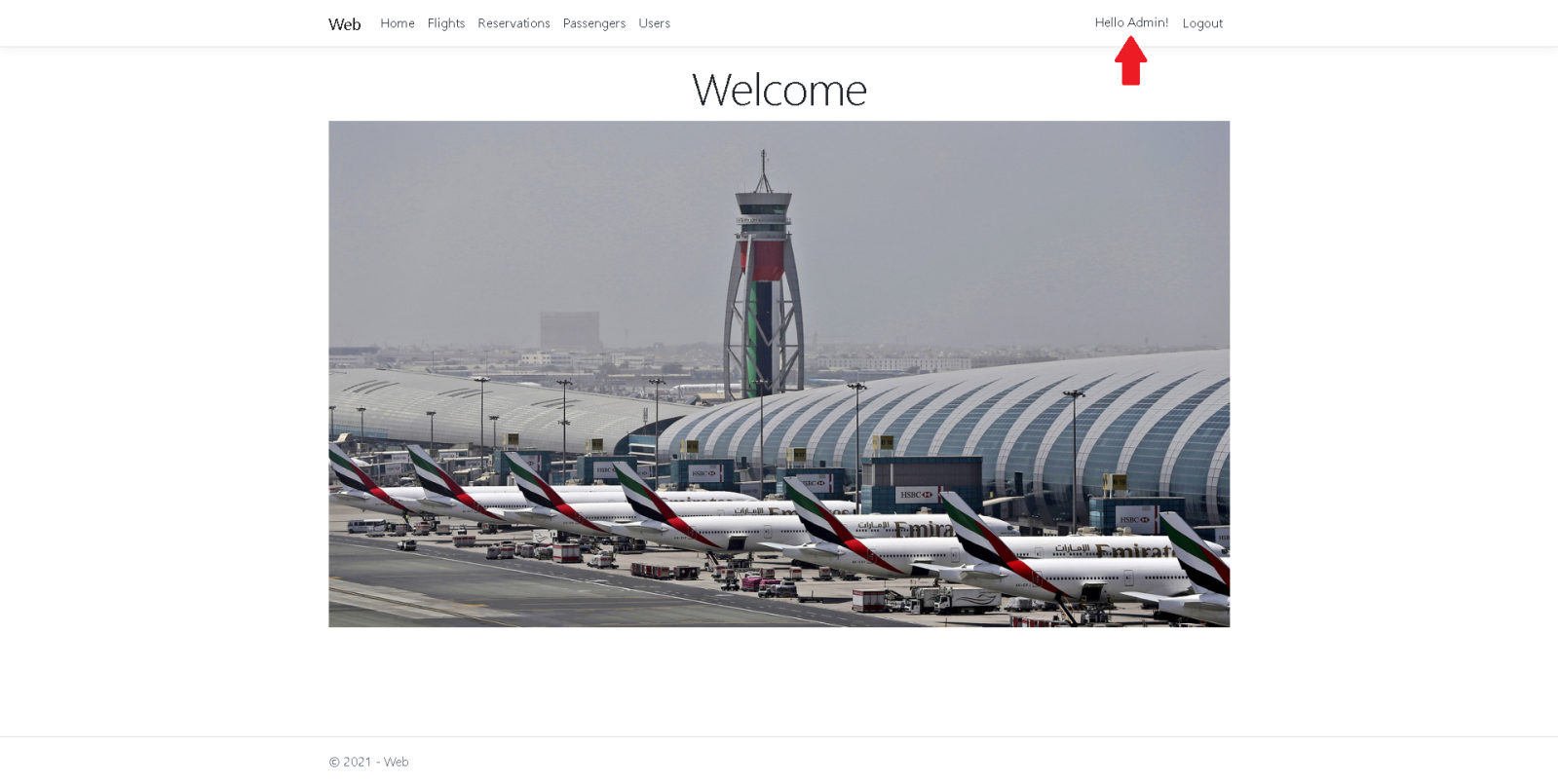
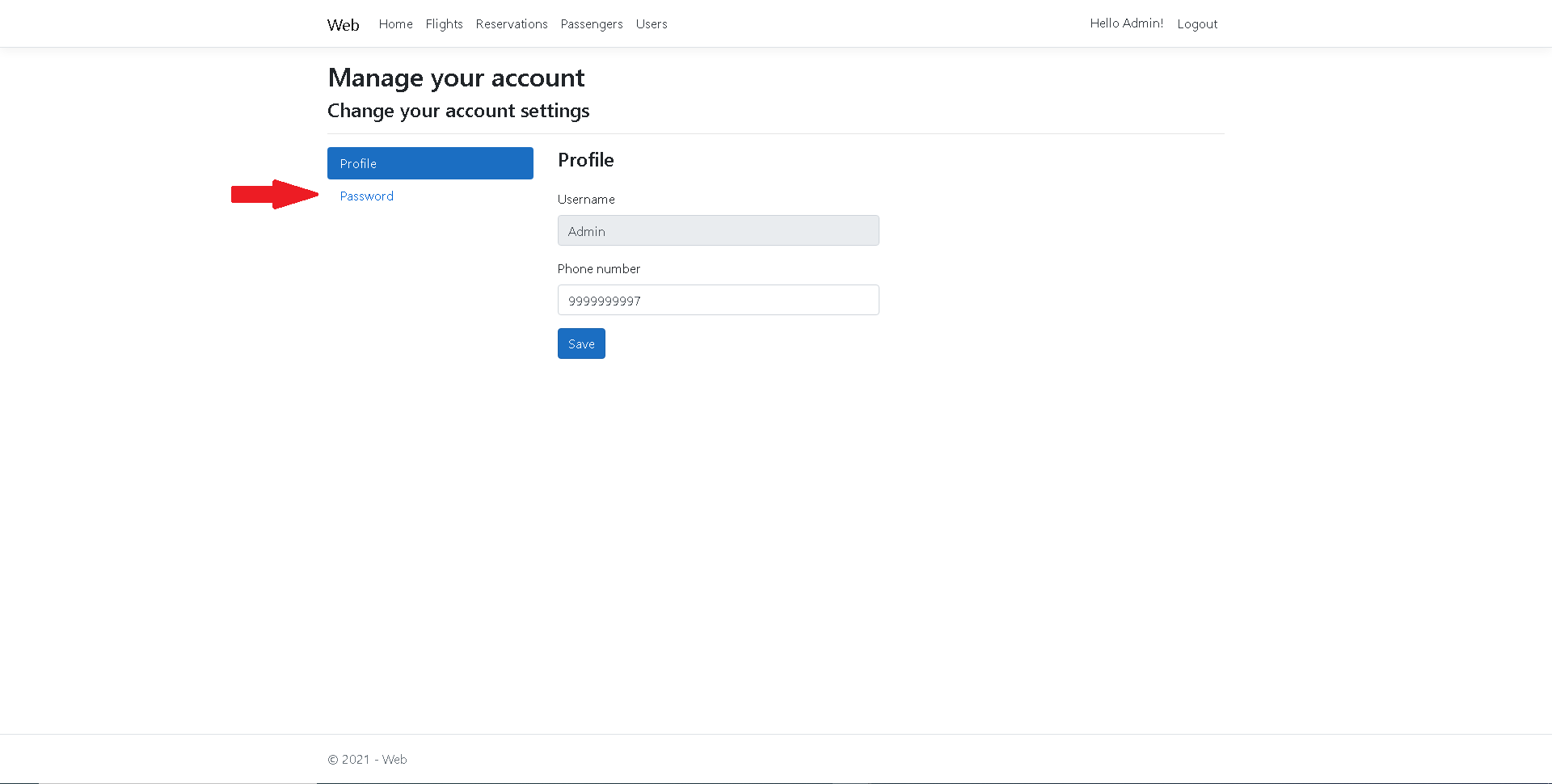
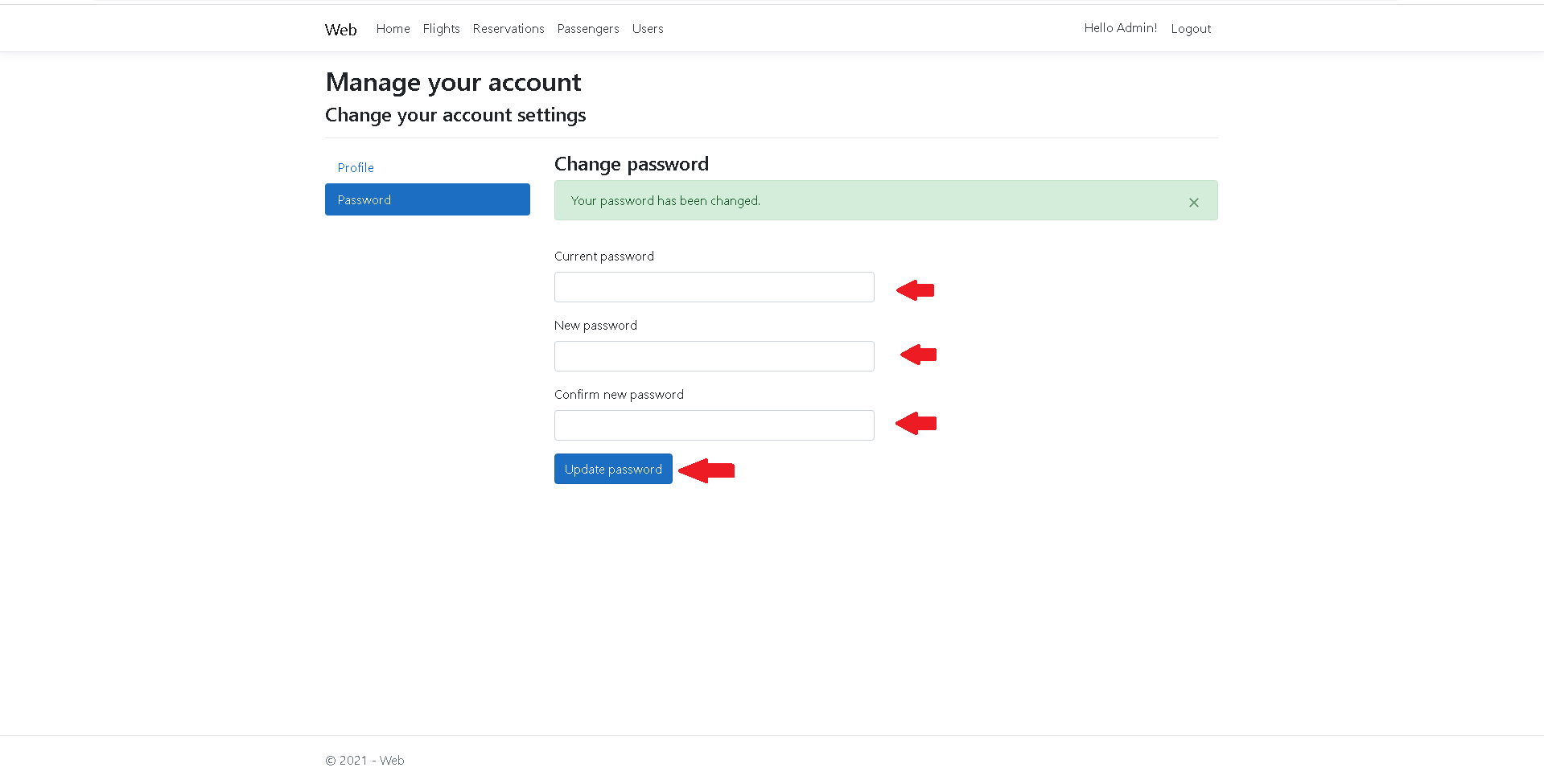
1. Кликвате върху „Reservations“ в началната страница.
2. Изберете опцията „Create New“.
3. Изберете полет от падащия списък. Въведете вашия имейл адрес. Кликнете върху бутона „Create“.
4. Попълнете данните за пътника. Посочете дали това е последният пътник за тази резервация. Кликнете върху бутона „Create passenger for reservation“. Ако сте посочили, че не сте приключили с добавянето на пътници, ще продължите да добавяте пътници.
5. Проверете посочения от вас имейл адрес. Ще получите съобщение дали резервацията е била успешна.



### *Как мога да променя телефонния ми номер?*

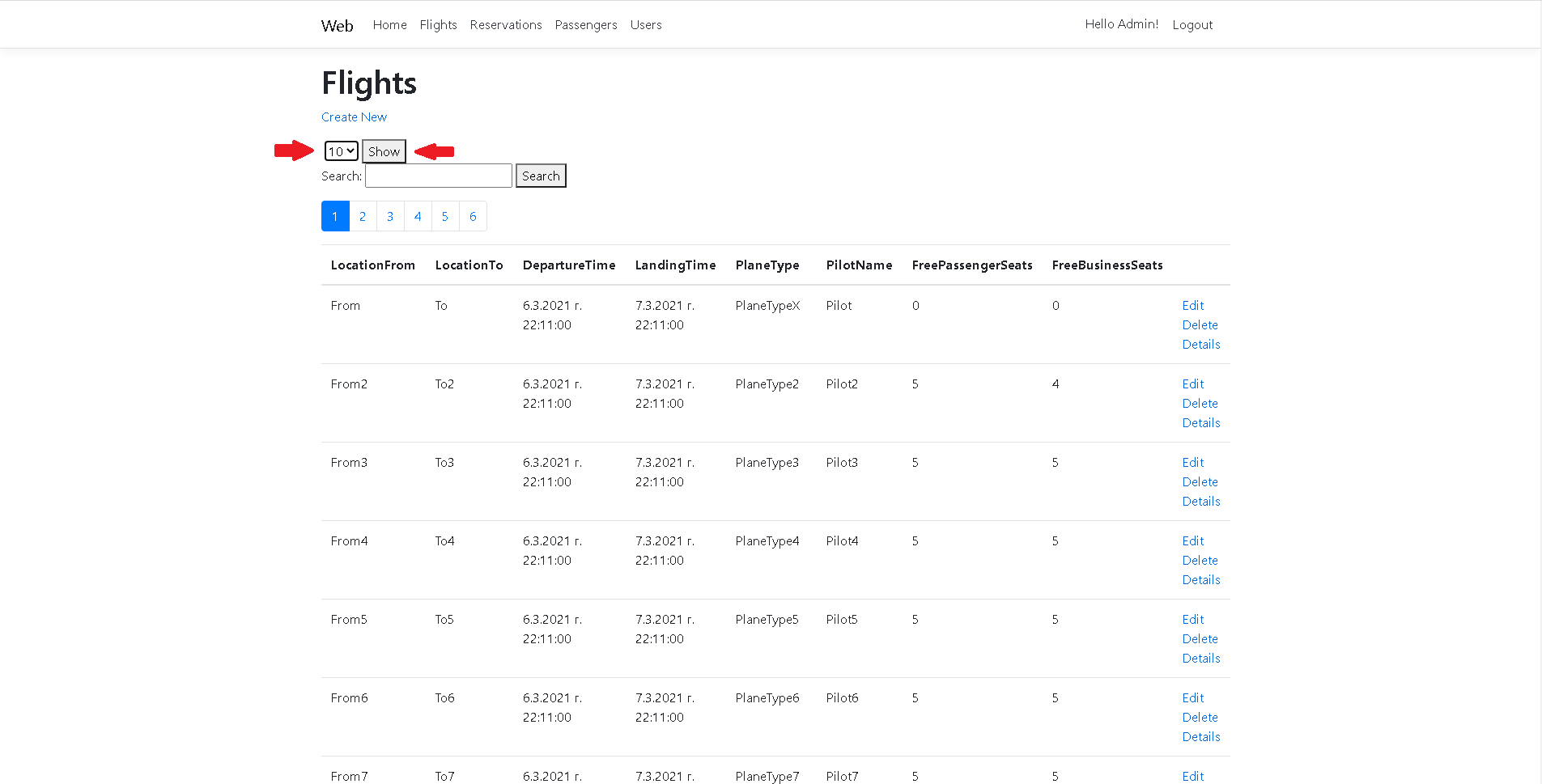
1. Кликнете върху вашия профил горе в дясно, когато се влезли във вашия акаунт.
2. Променете телефонния си номер и натиснете бутона „Save“.

### *Как мога да променя моята парола?*

1. Кликнете върху вашия профил горе в дясно, когато се влезли във вашия акаунт.
2. Изберете опцията „Password“.
3. Попълнете полетата и запазете новата парола с бутона „Update password“.

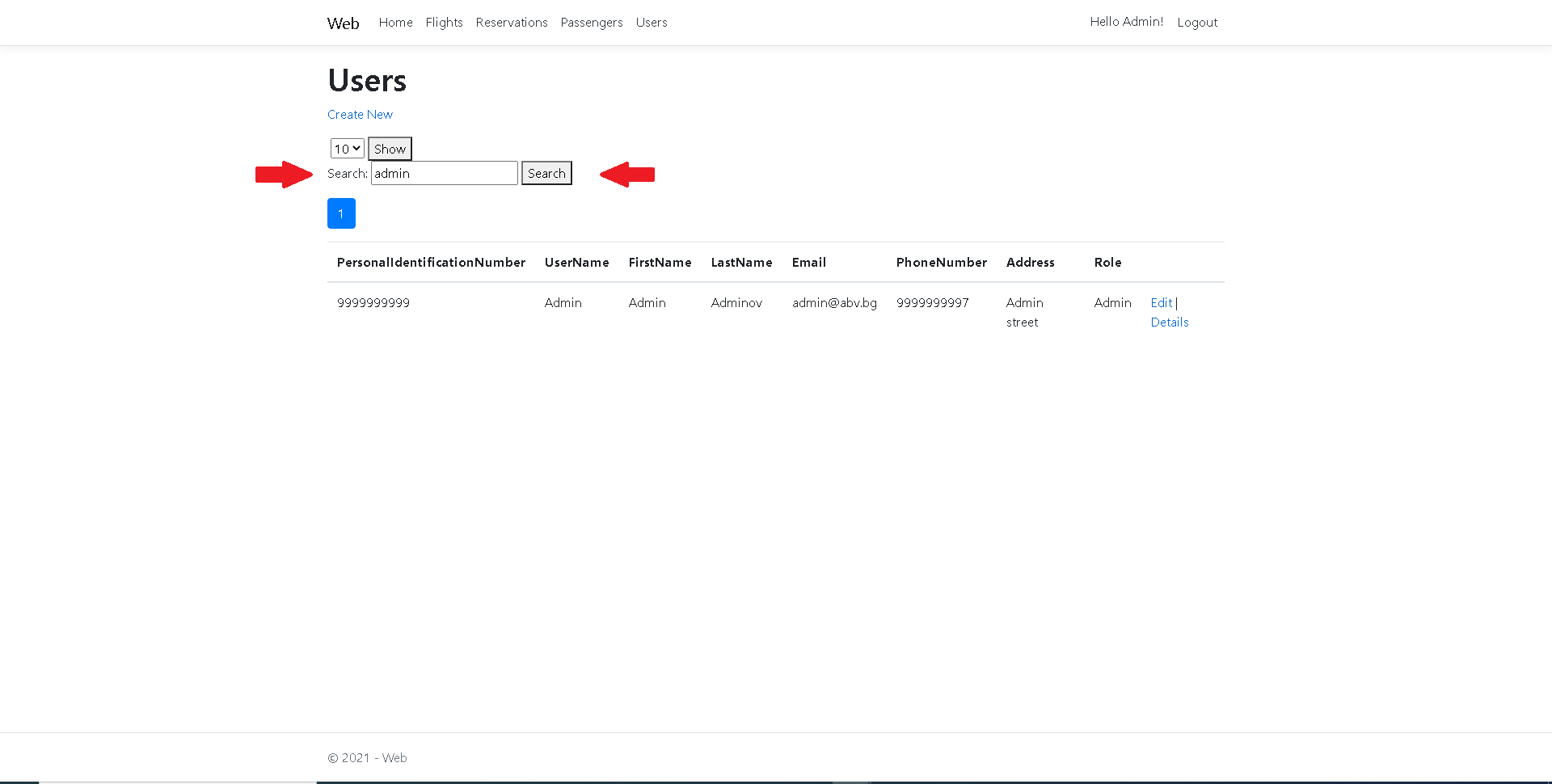
### *Как се страницира?*

1. Отидете на някой изглед от основните 4. Променете броя на показаните записи чрез падащия списък с числа. Натиснете бутона „Show“.

**

### *Как се филтрира?*

1. Отидете на някой изглед от основните 4. Попълнете полето „Search“. Кликнете върху бутона до него, който също се казва „Search“.



# Потребителски акаунти и роли.

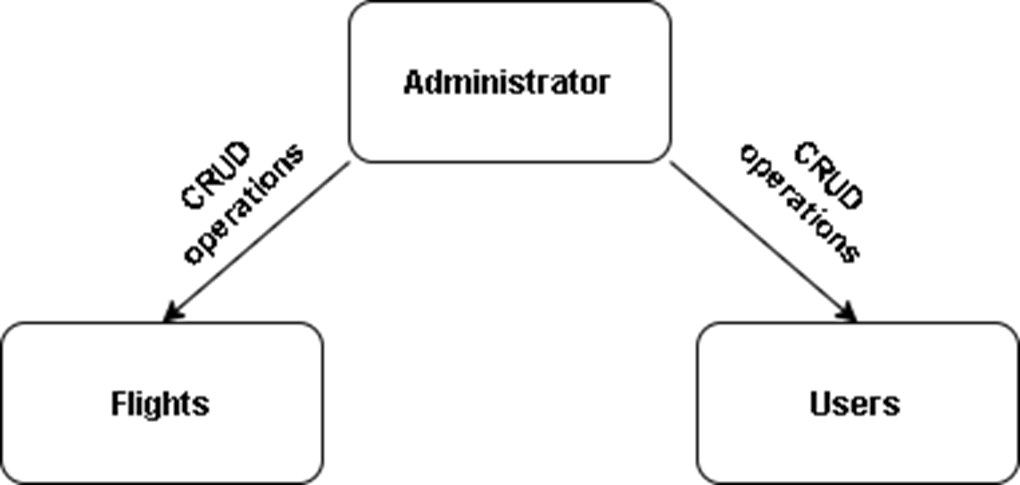
Потребителски акаунти имат само служителите на авиокомпанията, използваща софтуера.

Съществуват две възможни роли за потребителските акаунти:

* User – обикновен потребителски акаунт и няма специални привилегии
* Admin – администраторски акаунт, който има достъп до всички функционалности на уеб приложението
* Съществува само един администраторски акаунт, който се създава автоматично при първоначалното стартиране на приложението. Само администраторът може да управлява полетите и другите потребителски акаунти.

\*Имейл адрес на администратора по подразбиране: admin@abv.bg

\*Парола по подразбиране за администраторския акаунт: admin



# Функционалности на уеб приложението:

* Изгледи - позволяват на потребителя да види информация за полети, резервации, пътници и потребителски акаунти;
* Филтриране – данните, показани в изгледите, могат да се филтрират.

\***Филтрирането работи отделно от страницирането(всички резултати се извеждат в една страница)**

\*Филтрирането за отделните страници работи по следния начин:

-Полети ”Flights”: се филтрира по:

<локация от> (“LocationFrom”) и

<локация до> (“LocationTo”)

-Пътници “Passengers”: се филтрира по:

< собствено име>(“FirstName”),

<бащино име>(“MiddleName”),

<фамилия>(“LastName”) и

<ЕГН>(“PersonalIdentificationNumber”)

-Резервации “Reservations”: се филтрира по:

<електронна поща>(“Email”)

-Потребители “Users” се филтрира по:

<собствено име>(“FirstName”),

<фамилия >(“LastName”),

<електронна поща>(“Email”)

* Странициране – потребителят може да ограничи броя записи, показани на страницата

\*Страницирането позволява показването само на:“10“, “15“, “20“, “25“, “30“, “50“ записа на страница.

Броят страници зависи от броят записи в базата данни както и избраното количесвто за показване

**\*Страницирането не работи когато се търси запис**

* + Пътниците и резервациите не могат да се редактират и изтриват.
  + Резервация на билети:
  1. Всеки потребител може да резервира билети като посочи своя имейл адрес и избере полет. Входът в системата не е задължителен
  2. Може да се резервират билети за повече от 1 пътник наведнъж
  3. За успешна резервация е необходимо полетът да има свободни места
  4. Потребителя получава обратна връзка на електронната си поща с информация за своята резервация
  + Детайлни изгледи – потребителят може да разгледа подробна информация за конкретен запис на отделна страница
  + Смяна на паролата и телефонния номер – достъпно само за тези с потребителски акаунт

# Използвани инструменти и технологии:

* Visual Studio 2019
* C#
* Microsoft SQL Server
* Entity Framework Core
* Microsoft SQL Server Management Studio
* Git
* Github
* Trello
* ASP.NET Core Identity
* Razor Pages
* NUnit
* Mailkit

# Участници в проекта и техният принос към него:

* Онур Солак – изграждане на базата от данни, валидация на данните, имплементация на автентикация, оторизация, CRUD операции и функционалността за резервиране на билети;
* Джан Неджмидин – оформление на изгледите, изготвяне на презентация за проекта, имплементация на опции за филтриране и странициране;
* Станил Петров – имплементация на автоматизирани NUnit тестове и изготвяне на документация.

# XML документация на подпроекта „Data“.

<?xml version="1.0"?>

<doc>

<assembly>

<name>Data</name>

</assembly>

<members>

<member name="T:Data.Attributes.EmailUserUniqueAttribute">

<summary>

A custom attribute which ensures that the user's email address is unique to the database.

</summary>

</member>

<member name="M:Data.Attributes.EmailUserUniqueAttribute.IsValid(System.Object,System.ComponentModel.DataAnnotations.ValidationContext)">

<summary>

Performs a validation check on the value of the user's email address to ensure that it is unique.

</summary>

<param name="value">The value of the user's email address.</param>

<param name="validationContext">Describes the context in which a validation check is performed.</param>

<returns>A container for the results of a validation request.</returns>

</member>

<member name="M:Data.Attributes.EmailUserUniqueAttribute.GetErrorMessage(System.String)">

<summary>

Prepares an error message for duplicate email address.

</summary>

<param name="email">The duplicate email address.</param>

<returns>An error message for duplicate email address as a string.</returns>

</member>

<member name="T:Data.Attributes.UniquePassengerPIN">

<summary>

A custom attribute which ensures that the passenger's personal identification number is unique to the database.

It also ensures that the passenger's personal identification number is unique to the list which holds pending passengers.<see cref="T:Data.Static.ReservationAssistant"/>

</summary>

</member>

<member name="M:Data.Attributes.UniquePassengerPIN.IsValid(System.Object,System.ComponentModel.DataAnnotations.ValidationContext)">

<summary>

Performs a validation check on the value of the passenger's personal identification number to ensure that it is unique.

</summary>

<param name="value">The value of the passenger's personal identification number.</param>

<param name="validationContext">Describes the context in which a validation check is performed.</param>

<returns>A container for the results of a validation request.</returns>

</member>

<member name="M:Data.Attributes.UniquePassengerPIN.CheckPassengerPIN(System.Object)">

<summary>

Checks if the passenger's personal identification number is unique to the database.

</summary>

<param name="value">The personal identification number of the passenger.</param>

<returns>A bool value which indicates whether the passenger's personal identification number is unique to the database.</returns>

</member>

<member name="T:Data.Attributes.UniqueUserPIN">

<summary>

A custom attribute which ensures that the user's personal identification number is unique to the database.

</summary>

</member>

<member name="M:Data.Attributes.UniqueUserPIN.IsValid(System.Object,System.ComponentModel.DataAnnotations.ValidationContext)">

<summary>

Performs a validation check on the value of the user's personal identification number to ensure that it is unique.

</summary>

<param name="value">The value of the user's personal identification number.</param>

<param name="validationContext">Describes the context in which a validation check is performed.</param>

<returns>A container for the results of a validation request.</returns>

</member>

<member name="M:Data.Attributes.UniqueUserPIN.CheckUserPIN(System.Object)">

<summary>

Checks if the user's personal identification number is unique to the database.

</summary>

<param name="value">The personal identification number of the user.</param>

<returns>A bool value which indicates whether the user's personal identification number is unique to the database.</returns>

</member>

<member name="T:Data.Attributes.UnlikeAttribute">

<summary>

A custom attribute which ensures that the value of a certain property does not match the value of another property.

</summary>

</member>

<member name="P:Data.Attributes.UnlikeAttribute.OtherProperty">

<summary>

The name of the other property which will be used for validation.

</summary>

</member>

<member name="M:Data.Attributes.UnlikeAttribute.#ctor(System.String)">

<summary>

The constructor of the UnlikeAttribute class which prepares the data for validation.

</summary>

<param name="otherProperty">The name of the other property</param>

</member>

<member name="M:Data.Attributes.UnlikeAttribute.IsValid(System.Object,System.ComponentModel.DataAnnotations.ValidationContext)">

<summary>

Performs a validation check on the value of the current property to ensure that it is different from the other property's value.

</summary>

<param name="value">The value of the current property which should not match the other property's value.</param>

<param name="validationContext">Describes the context in which a validation check is performed.</param>

<returns>A container for the results of a validation request.</returns>

</member>

<member name="T:Data.Attributes.ValidFlightTimesAttribute">

<summary>

A custom attribute which ensures that the flight's departure time is before its landing time.

</summary>

</member>

<member name="M:Data.Attributes.ValidFlightTimesAttribute.IsValid(System.Object,System.ComponentModel.DataAnnotations.ValidationContext)">

<summary>

Performs a validation check which attempts to confirm that the flight's departure time is before its landing time.

</summary>

<param name="value">The flight's departure time.</param>

<param name="validationContext">Describes the context in which a validation check is performed.</param>

<returns>A container for the results of a validation request.</returns>

</member>

<member name="T:Data.Context.FlightManagerDbContext">

<summary>

The database context for the "FlightManager" project. It inherits the IdentityDbContext class with an extended version of IdentityUser class.

The database contains all tables needed for implementing authentication and authorization. As an addition, it contains the "Flights", "Reservations" and "Passengers" tables

which exist for providing the rest of the functionalities of the project.

</summary>

</member>

<member name="P:Data.Context.FlightManagerDbContext.Flights">

<summary>

A <see cref="T:Microsoft.EntityFrameworkCore.DbSet`1"/> for the <see cref="T:Data.Models.Flight"/> class.

</summary>

</member>

<member name="P:Data.Context.FlightManagerDbContext.Reservations">

<summary>

A <see cref="T:Microsoft.EntityFrameworkCore.DbSet`1"/> for the <see cref="T:Data.Models.Reservation"/> class.

</summary>

</member>

<member name="P:Data.Context.FlightManagerDbContext.Passengers">

<summary>

A <see cref="T:Microsoft.EntityFrameworkCore.DbSet`1"/> for the <see cref="T:Data.Models.Passenger"/> class.

</summary>

</member>

<member name="M:Data.Context.FlightManagerDbContext.#ctor(Microsoft.EntityFrameworkCore.DbContextOptions{Data.Context.FlightManagerDbContext})">

<summary>

A constructor for the FlightManagerDbContext with default options.

</summary>

<param name="options"><see cref="T:Microsoft.EntityFrameworkCore.DbContextOptions`1"/></param>

</member>

<member name="M:Data.Context.FlightManagerDbContext.OnModelCreating(Microsoft.EntityFrameworkCore.ModelBuilder)">

<summary>

A Fluent API which lets you modify how your database entities are mapped. In this case, there are added restrictions which make

the personal identification numbers of both the users and the passengers unique columns.

</summary>

<param name="builder"><see cref="T:Microsoft.EntityFrameworkCore.ModelBuilder"/></param>

</member>

<member name="M:Data.Context.FlightManagerDbContext.OnConfiguring(Microsoft.EntityFrameworkCore.DbContextOptionsBuilder)">

<summary>

A method which lets you configure the database to be used for this context. The connection string for the database is set here.

Lazy-loading proxies are enabled.

</summary>

<param name="optionsBuilder"><see cref="T:Microsoft.EntityFrameworkCore.DbContextOptionsBuilder"/></param>

</member>

<member name="T:Data.Models.Flight">

<summary>

The class contains information about plane flights. Reservations can be made for these flights.

</summary>

</member>

<member name="P:Data.Models.Flight.UniquePlaneNumber">

<summary>

An unique plane number for the flight. It serves the purpose of a primary key for the database.

</summary>

</member>

<member name="P:Data.Models.Flight.LocationFrom">

<summary>

The location from where the plane departs. The property is required and its length cannot exceed 50 characters.

</summary>

</member>

<member name="P:Data.Models.Flight.LocationTo">

<summary>

The destination where the plane will arrive. The property is required and its length cannot exceed 50 characters.

Also it must be different from the departure location.

</summary>

</member>

<member name="P:Data.Models.Flight.DepartureTime">

<summary>

The time when the plane will depart. This property is required. It must be before the landing time.

</summary>

</member>

<member name="P:Data.Models.Flight.LandingTime">

<summary>

The time when the plane will arrive at its destination. This property is required.

</summary>

</member>

<member name="P:Data.Models.Flight.PlaneType">

<summary>

The type of the plane. The property is required and its length cannot exceed 50 characters.

</summary>

</member>

<member name="P:Data.Models.Flight.PilotName">

<summary>

The name of the person who pilots the plane. The property is required and its length cannot exceed 50 characters.

Also it can contain only letters.

</summary>

</member>

<member name="P:Data.Models.Flight.FreePassengerSeats">

<summary>

The amount of free passenger seats for the economy class. This property is required.

</summary>

</member>

<member name="P:Data.Models.Flight.FreeBusinessSeats">

<summary>

The amount of free passenger seats for the business class. This property is required.

</summary>

</member>

<member name="P:Data.Models.Flight.FlightGeneralInfo">

<summary>

A property which is not mapped to the database. It returns general information about the flight which includes

its unique plane number, departure location and destination.

</summary>

</member>

<member name="P:Data.Models.Flight.Reservations">

<summary>

A navigation property used for configuring relation to the <see cref="T:Data.Models.Reservation"/> entities.

</summary>

</member>

<member name="M:Data.Models.Flight.#ctor">

<summary>

A constructor which sets an empty HashSet as a value to the <see cref="P:Data.Models.Flight.Reservations"/> navigation property.

</summary>

</member>

<member name="T:Data.Models.FlightIndexViewModel">

<summary>

Class for paging Flights Index view

</summary>

</member>

<member name="T:Data.Models.Passenger">

<summary>

The class represents the passengers who will have reservation for the flights.

</summary>

</member>

<member name="P:Data.Models.Passenger.Id">

<summary>

The primary key for the database.

</summary>

</member>

<member name="P:Data.Models.Passenger.PersonalIdentificationNumber">

<summary>

Each passenger has unique personal identification number (PIN) which consists of only 10 digits.

The personal identification number is the equivalent of the bulgarian "ЕГН". This property is required.

</summary>

</member>

<member name="P:Data.Models.Passenger.FirstName">

<summary>

The first name of the passenger. This property is required. It cannot exceed 50 characters and contains only letters.

</summary>

</member>

<member name="P:Data.Models.Passenger.MiddleName">

<summary>

The middle name of the passenger. This property is required. It cannot exceed 50 characters and contains only letters.

</summary>

</member>

<member name="P:Data.Models.Passenger.LastName">

<summary>

The last name of the passenger. This property is required. It cannot exceed 50 characters and contains only letters.

</summary>

</member>

<member name="P:Data.Models.Passenger.PhoneNumber">

<summary>

The phone number of the passenger. It consists of 10 digits and it is required.

</summary>

</member>

<member name="P:Data.Models.Passenger.Nationality">

<summary>

The nationality of the passenger. This property is required. It cannot exceed 50 characters and contains only letters.

</summary>

</member>

<member name="P:Data.Models.Passenger.TicketType">

<summary>

The ticket type which the passenger has. It can be either normal (economy) class or business class.

</summary>

</member>

<member name="P:Data.Models.Passenger.ReservationId">

<summary>

A navigation property used for configuring the relation to the <see cref="T:Data.Models.Reservation"/> entities.

</summary>

</member>

<member name="P:Data.Models.Passenger.Reservation">

<summary>

A navigation property used for configuring the relation to the <see cref="T:Data.Models.Reservation"/> entities.

</summary>

</member>

<member name="T:Data.Models.PassengerIndexViewModel">

<summary>

Class for paging Passengers Index view

</summary>

</member>

<member name="T:Data.Models.Reservation">

<summary>

The class which represents the flight reservations. One reservation can be made for more than one passenger. The only limit

to how many passengers a reservation can have is the amount of free seats for the flight.

</summary>

</member>

<member name="P:Data.Models.Reservation.Id">

<summary>

The primary key for the database.

</summary>

</member>

<member name="P:Data.Models.Reservation.FlightUniquePlaneNumber">

<summary>

A navigation property used for configuring the relation to the <see cref="T:Data.Models.Flight"/> entities.

</summary>

</member>

<member name="P:Data.Models.Reservation.Flight">

<summary>

A navigation property used for configuring the relation to the <see cref="T:Data.Models.Flight"/> entities.

</summary>

</member>

<member name="P:Data.Models.Reservation.Email">

<summary>

The email address on which the reservation is made. It will receive an email whenever a reservation is made. This email will

notify the user if the reservation is successful and give him extra information about the flight for which the reservation is

made. This property is required.

</summary>

</member>

<member name="P:Data.Models.Reservation.Passengers">

<summary>

A navigation property used for configuring the relation to the <see cref="T:Data.Models.Passenger"/> entities.

</summary>

</member>

<member name="M:Data.Models.Reservation.#ctor">

<summary>

A constructor which sets an empty HashSet as a value to the <see cref="P:Data.Models.Reservation.Passengers"/> navigation property.

</summary>

</member>

<member name="T:Data.Models.ReservationIndexViewModel">

<summary>

Class for paging Reservations Index view

</summary>

</member>

<member name="T:Data.Models.User">

<summary>

The class represents the users who are registered on the FlightManager website. This class is an extended version

of the <see cref="T:Microsoft.AspNetCore.Identity.IdentityUser"/> class. It is used for authentication and authorization.

</summary>

</member>

<member name="P:Data.Models.User.FirstName">

<summary>

The first name of the user. This property is required. It cannot exceed 50 characters and contains only letters.

</summary>

</member>

<member name="P:Data.Models.User.PhoneNumber">

<summary>

The phone number of the user. This property is required. It consists of only 10 digits.

</summary>

</member>

<member name="P:Data.Models.User.LastName">

<summary>

The last name of the user. This property is required. It cannot exceed 50 characters and contains only letters.

</summary>

</member>

<member name="P:Data.Models.User.PersonalIdentificationNumber">

<summary>

Each user has unique personal identification number (PIN) which consists of only 10 digits.

The personal identification number is the equivalent of the bulgarian "ЕГН". This property is required.

</summary>

</member>

<member name="P:Data.Models.User.Address">

<summary>

The address where the user lives. This property is required. It cannot exceed 50 characters and contains only letters.

</summary>

</member>

<member name="P:Data.Models.User.Role">

<summary>

The role of the user. In this application there are only 2 roles. The "User" role indicates that the user is an employee of the FlightManager company and it

does not offer him special privileges on the website. The "Admin" role grants the user full access to the website. There can be only 1 administrator. They have

access to all CRUD operations over the other users in the database. The administrator can also add flights to the database and modify them.

</summary>

</member>

<member name="T:Data.Models.UserEditViewModel">

<summary>

Almost identical to the <see cref="T:Data.Models.User"/> class. The difference is that this model is used for

the validation of user data which is being modified. The <see cref="P:Data.Models.UserEditViewModel.PersonalIdentificationNumber"/> property does not contain

the attribute which ensures that its value is unique to the database.

</summary>

</member>

<member name="P:Data.Models.UserEditViewModel.FirstName">

<summary>

<inheritdoc cref="P:Data.Models.User.FirstName"/>

</summary>

</member>

<member name="P:Data.Models.UserEditViewModel.PhoneNumber">

<summary>

<inheritdoc cref="P:Data.Models.User.PhoneNumber"/>

</summary>

</member>

<member name="P:Data.Models.UserEditViewModel.LastName">

<summary>

<inheritdoc cref="P:Data.Models.User.LastName"/>

</summary>

</member>

<member name="P:Data.Models.UserEditViewModel.PersonalIdentificationNumber">

<summary>

<inheritdoc cref="P:Data.Models.User.PersonalIdentificationNumber"/>

</summary>

</member>

<member name="P:Data.Models.UserEditViewModel.Address">

<summary>

<inheritdoc cref="P:Data.Models.User.Address"/>

</summary>

</member>

<member name="P:Data.Models.UserEditViewModel.Role">

<summary>

<inheritdoc cref="P:Data.Models.User.Role"/>

</summary>

</member>

<member name="T:Data.Models.UserIndexViewModel">

<summary>

Class for paging Users Index view

</summary>

</member>

<member name="T:Data.Shared.PagerViewModel">

<summary>

Class for properties used for paging the Index views

</summary>

</member>

<member name="T:Data.Static.ReservationAssistant">

<summary>

A static class which provides methods for the creation of reservations. It also stores data of pending passengers and their reservation which are not

ready to be added to the database.

</summary>

</member>

<member name="P:Data.Static.ReservationAssistant.PendingReservation">

<summary>

The pending reservation which cannot be completed because the amount of free seats for the flight are not checked and the user has not finished adding passengers

to the reservation. This static property stores data temporarily.

</summary>

</member>

<member name="P:Data.Static.ReservationAssistant.PendingPassengers">

<summary>

The pending passengers which cannot be added to the database because the amount of free seats for the flight are not checked and the user has not finished adding passengers

to the reservation. This static property stores data temporarily.

</summary>

</member>

<member name="M:Data.Static.ReservationAssistant.ClearPendingPassengers">

<summary>

Removes the pending passengers from the static <see cref="T:Data.Static.ReservationAssistant"/> class because they will be replaced by new ones for future reservations.

It is used when the pending reservation has changed or been finished.

</summary>

</member>

<member name="M:Data.Static.ReservationAssistant.PrepareEmail(System.Text.StringBuilder)">

<summary>

Prepares the whole email by configuring its sender, receiver, subject and body. A dedicated gmail account exists as the sender.

The receiver is the email which the user specifies while creating the reservation. The email is about the status of the reservation

and notifies the user if it is successful or not.

</summary>

<param name="emailBody">The completed email body which is formatted text. The email body is a StringBuilder object.<see cref="T:System.Text.StringBuilder"/></param>

<returns>A <see cref="T:MimeKit.MimeMessage"/> object representing an email which is ready to be sent.</returns>

</member>

<member name="M:Data.Static.ReservationAssistant.GetFlightByUniquePlaneNumber(System.Int32)">

<summary>

Extracts a flight from the database using its unique plane number.

</summary>

<param name="uniquePlaneNumber">The unique plane number of the flight.</param>

<returns>A <see cref="T:Data.Models.Flight"/> object.</returns>

</member>

<member name="M:Data.Static.ReservationAssistant.GetSuccessfulReservationInformation(System.Text.StringBuilder,Data.Models.Flight)">

<summary>

Prepares a part of the email body when the reservation is successful. It includes information

about the passengers and the flight related to the reservation.

</summary>

<param name="emailBody">A <see cref="T:System.Text.StringBuilder"/> object which contains the email body.</param>

<param name="flight">The fight for which the reservation is being made.<see cref="T:Data.Models.Flight"/></param>

<returns>The modified email body which is formatted text. The email body is still a <see cref="T:System.Text.StringBuilder"/> object.</returns>

</member>

<member name="M:Data.Static.ReservationAssistant.GetFailedReservationInformation(System.Text.StringBuilder)">

<summary>

Prepares a part of the email body when the reservation is unsuccessful. It includes information

about the passengers related to the reservation.

</summary>

<param name="emailBody">A <see cref="T:System.Text.StringBuilder"/> object which contains the email body.</param>

<returns>The modified email body which is formatted text. The email body is still a <see cref="T:System.Text.StringBuilder"/> object.</returns>

</member>

<member name="M:Data.Static.ReservationAssistant.PrepareTheWholeEmailBody(System.Boolean)">

<summary>

Prepares the whole email body depending on the status of the reservation. It can either create an email body for successful or failed

reservation.

</summary>

<param name="isSuccessfulReservation">The status of the reservation. A bool which indicates whether or not the reservation is successful.</param>

<returns>The completed email body which is formatted text. The email body is a <see cref="T:System.Text.StringBuilder"/> object.</returns>

</member>

<member name="M:Data.Static.ReservationAssistant.SendEmail(MimeKit.MimeMessage)">

<summary>

Connects to a dedicated gmail account which sends an email to the user.

</summary>

<param name="email">A <see cref="T:MimeKit.MimeMessage"/> object representing an email which is ready to be sent.</param>

</member>

</members>

</doc>

# XML документация на подпроекта „Flights.Test“.

<?xml version="1.0"?>

<doc>

<assembly>

<name>Flights.Test</name>

</assembly>

<members>

<member name="T:Flights.Test.FlightsTests">

<summary>

This class runs tests on class <see cref="T:Data.Models.Flight"/>

</summary>

</member>

<member name="M:Flights.Test.FlightsTests.Setup">

<summary>

Prepares valid data for the <see cref="T:Data.Models.Flight"/> object in this class before every test.

</summary>

</member>

<member name="M:Flights.Test.FlightsTests.ValidateFlight">

<summary>

This is the method which checks if the flight is valid.

</summary>

</member>

<member name="M:Flights.Test.FlightsTests.CheckInvalidPilotName">

<summary>

Checks an invalid pilot name which does not contain only letters.

</summary>

</member>

<member name="M:Flights.Test.FlightsTests.CheckInvalidPilotNameSize">

<summary>

Checks an invalid pilot name which exceeds the 50 character length limit.

</summary>

</member>

<member name="M:Flights.Test.FlightsTests.CheckCompletelyValidFlight">

<summary>

Check a completely valid flight which should not throw <see cref="T:System.ComponentModel.DataAnnotations.ValidationException"/>.

</summary>

</member>

<member name="M:Flights.Test.FlightsTests.CheckInvalidFreePassengerSeats">

<summary>

Checks an invalid free passenger seats value which is negative.

</summary>

</member>

<member name="M:Flights.Test.FlightsTests.CheckInvalidFreeBusinessSeats">

<summary>

Checks an invalid free business seats value which is negative.

</summary>

</member>

<member name="M:Flights.Test.FlightsTests.CheckInvalidLocacionFrom">

<summary>

Check an invalid flight which has the same departure location as destination.

</summary>

</member>

<member name="T:Flights.Test.PassengerTests">

<summary>

This class runs tests on class <see cref="T:Data.Models.Passenger"/>

</summary>

</member>

<member name="M:Flights.Test.PassengerTests.Setup">

<summary>

Prepares valid data for the <see cref="T:Data.Models.Passenger"/> object in this class before every test.

</summary>

</member>

<member name="M:Flights.Test.PassengerTests.ValidatePassenger">

<summary>

This is the method which checks if the passenger is valid.

</summary>

</member>

<member name="M:Flights.Test.PassengerTests.CheckCompletelyValidPassenger">

<summary>

Check completely valid passenger which should not throw <see cref="T:System.ComponentModel.DataAnnotations.ValidationException"/>.

</summary>

</member>

<member name="M:Flights.Test.PassengerTests.CheckInvalidFirstName">

<summary>

Checks invalid first name which does not contain only letters.

</summary>

</member>

<member name="M:Flights.Test.PassengerTests.CheckInvalidFirstNameSize">

<summary>

Check invalid first name which exceeds the 50 character length limit.

</summary>

</member>

<member name="M:Flights.Test.PassengerTests.CheckInvalidPersonalIdentificationNumberWhichContainsLetters">

<summary>

Checks an invalid personal identification number which contains letters.

</summary>

</member>

<member name="M:Flights.Test.PassengerTests.CheckPersonalIdentificationNumberWhichContainsInvalidNumberOfDigits">

<summary>

Checks a personal identification number which contains invalid number of digits.

</summary>

</member>

<member name="M:Flights.Test.PassengerTests.CheckInvalidPhoneNumberWhichContainsLetters">

<summary>

Checks an invalid phone number which contains letters.

</summary>

</member>

<member name="M:Flights.Test.PassengerTests.CheckInvalidPhoneNumberLength">

<summary>

Checks an invalid phone number which contains invalid number of digits.

</summary>

</member>

<member name="T:Flights.Test.ReservationTests">

<summary>

This class runs tests on class <see cref="T:Data.Models.Reservation"/>

</summary>

</member>

<member name="M:Flights.Test.ReservationTests.Setup">

<summary>

Prepares valid data for the <see cref="T:Data.Models.Reservation"/> object in this class before every test.

</summary>

</member>

<member name="M:Flights.Test.ReservationTests.ValidateReservation">

<summary>

This is the method which checks if the reservation is valid.

</summary>

</member>

<member name="M:Flights.Test.ReservationTests.CheckCompletelyValidReservation">

<summary>

Checks a completely valid reservation which should not throw <see cref="T:System.ComponentModel.DataAnnotations.ValidationException"/>.

</summary>

</member>

<member name="M:Flights.Test.ReservationTests.CheckInvalidEmail">

<summary>

Checks an invalid email address.

</summary>

</member>

<member name="T:Flights.Test.UserEditViewModelTests">

<summary>

This class runs tests on class <see cref="T:Data.Models.UserEditViewModel"/>

</summary>

</member>

<member name="M:Flights.Test.UserEditViewModelTests.Setup">

<summary>

Prepares valid data for the <see cref="T:Data.Models.UserEditViewModel"/> object in this class before every test.

</summary>

</member>

<member name="M:Flights.Test.UserEditViewModelTests.ValidateUserEdit">

<summary>

This is the method which checks if the user is valid.

</summary>

</member>

<member name="M:Flights.Test.UserEditViewModelTests.CheckCompletelyValidUserEdit">

<summary>

Checks a completely valid user which should not throw <see cref="T:System.ComponentModel.DataAnnotations.ValidationException"/>.

</summary>

</member>

<member name="M:Flights.Test.UserEditViewModelTests.CheckInvalidPersonalIdentificationNumberWhichContainsLetters">

<summary>

Checks an invalid personal identification number which contains letters.

</summary>

</member>

<member name="M:Flights.Test.UserEditViewModelTests.CheckInvalidFirstName">

<summary>

Checks an invalid first name which does not contain only letters.

</summary>

</member>

<member name="M:Flights.Test.UserEditViewModelTests.CheckInvalidFirstNameSize">

<summary>

Checks an invalid first name which exceeds the 50 character length limit.

</summary>

</member>

<member name="M:Flights.Test.UserEditViewModelTests.CheckInvalidLastName">

<summary>

Checks an invalid last name which does not contain only letters.

</summary>

</member>

<member name="M:Flights.Test.UserEditViewModelTests.CheckInvalidLastNameSize">

<summary>

Checks an invalid last name which exceeds the 50 character length limit.

</summary>

</member>

<member name="M:Flights.Test.UserEditViewModelTests.CheckPersonalIdentificationNumberWhichContainsInvalidNumberOfDigits">

<summary>

Checks a personal identification number which contains invalid number of digits

</summary>

</member>

<member name="M:Flights.Test.UserEditViewModelTests.CheckInvalidPhoneNumberWhichContainsLetters">

<summary>

Checks an invalid phone number which contains letters.

</summary>

</member>

<member name="M:Flights.Test.UserEditViewModelTests.CheckInvalidPhoneNumberWithInvalidLength">

<summary>

Checks an invalid phone number which contains invalid number of digits.

</summary>

</member>

<member name="T:Flights.Test.UserTests">

<summary>

This class runs tests on class <see cref="T:Data.Models.User"/>

</summary>

</member>

<member name="M:Flights.Test.UserTests.Setup">

<summary>

Prepares valid data for the <see cref="T:Data.Models.User"/> object in this class before every test.

</summary>

</member>

<member name="M:Flights.Test.UserTests.ValidateUser">

<summary>

This is the method which checks if the user is valid.

</summary>

</member>

<member name="M:Flights.Test.UserTests.CheckCompletelyValidUser">

<summary>

Check a completely valid user which should not throw <see cref="T:System.ComponentModel.DataAnnotations.ValidationException"/>.

</summary>

</member>

<member name="M:Flights.Test.UserTests.CheckInvalidPersonalIdentificationNumberWhichContainsLetters">

<summary>

Checks an invalid personal identification number which contains letters.

</summary>

</member>

<member name="M:Flights.Test.UserTests.CheckPersonalIdentificationNumberWhichContainsInvalidNumberOfDigits">

<summary>

Check a personal identification number which contains invalid number of digits.

</summary>

</member>

<member name="M:Flights.Test.UserTests.CheckInvalidPhoneNumberWhichContainsLetters">

<summary>

Checks an invalid phone number which contains letters.

</summary>

</member>

<member name="M:Flights.Test.UserTests.CheckPersonalPhoneNumberWithInvalidLength">

<summary>

Checks an invalid phone number which contains invalid number of digits.

</summary>

</member>

<member name="M:Flights.Test.UserTests.CheckInvalidFirstName">

<summary>

Checks an invalid first name which does not contain only letters.

</summary>

</member>

<member name="M:Flights.Test.UserTests.CheckInvalidFirstNameSize">

<summary>

Checks an invalid first name which exceeds the 50 character length limit.

</summary>

</member>

<member name="M:Flights.Test.UserTests.CheckInvalidLastName">

<summary>

Checks an invalid last name which does not contain only letters.

</summary>

</member>

<member name="M:Flights.Test.UserTests.CheckInvalidLastNameSize">

<summary>

Checks an invalid last name which exceeds the 50 character length limit.

</summary>

</member>

</members>

</doc>

# XML документация на подпроекта „Web“.

<?xml version="1.0"?>

<doc>

<assembly>

<name>Web</name>

</assembly>

<members>

<member name="T:Web.Controllers.FlightsController">

<summary>

The controller which is responsible for flight CRUD operations.

</summary>

</member>

<member name="M:Web.Controllers.FlightsController.#ctor(Data.Context.FlightManagerDbContext)">

<summary>

Standard constructor which initializes a context used by the controller.

</summary>

<param name="context">The flight manager context.</param>

</member>

<member name="M:Web.Controllers.FlightsController.Index(System.String,Data.Models.FlightIndexViewModel,System.Int32)">

<summary>

The Flights controller for the Index page. Supports pagination and filtration

</summary>

<param name="searchString">string for filtering Flights with given from location and to location</param>

<param name="model">FlightIndexViewModel object used to display filtered or paged records from Flights</param>

<param name="pages">int for the amount of records to be shown on a single page</param>

<returns>Returns Flights Index View with the filtered Flights(if entered search string) otherwise pages all records</returns>

</member>

<member name="M:Web.Controllers.FlightsController.Details(System.Nullable{System.Int32})">

<summary>

The Flights controller for the Details page. Shows details about the Flight and Passengers in selected Flight

</summary>

<param name="id">int for Flight id to find Passengers</param>

<returns>Returns Flight Detail page showing information for both the Flight and its Passengers</returns>

</member>

<member name="M:Web.Controllers.FlightsController.Create">

<summary>

Returns the form for flight creation which the user will fill. Accessible only by the administrator.

</summary>

<returns>Flight creation view.</returns>

</member>

<member name="M:Web.Controllers.FlightsController.Create(Data.Models.Flight)">

<summary>

Creates a <see cref="T:Data.Models.Flight"/> object with the information provided by the user and adds it to the database. Accessible only by the administrator.

</summary>

<param name="flight">The flight which will be added to the database.</param>

<returns>The index page for flights.</returns>

</member>

<member name="M:Web.Controllers.FlightsController.Edit(System.Nullable{System.Int32})">

<summary>

Finds a flight from the database by using its primary key and prepares it for edit. Accessible only by the administrator.

</summary>

<param name="id">The id of the flight which will be edited.</param>

<returns>Flight edit view</returns>

</member>

<member name="M:Web.Controllers.FlightsController.Edit(System.Int32,Data.Models.Flight)">

<summary>

Edits a flight and saves the changes to the database. Accessible only by the administrator.

</summary>

<param name="id">The primary key of the flight</param>

<param name="flight">The modified flight which will be saved.</param>

<returns>The index page for flights.</returns>

</member>

<member name="M:Web.Controllers.FlightsController.Delete(System.Nullable{System.Int32})">

<summary>

Prepares a flight for deletion. Accessible only by the administrator.

</summary>

<param name="id">Primary key of the flight which will be deleted</param>

<returns>Flight delete view</returns>

</member>

<member name="M:Web.Controllers.FlightsController.DeleteConfirmed(System.Int32)">

<summary>

Deletes the flight after receiving confirmation from the user. Accessible only by the administrator.

</summary>

<param name="id">Primary key of the deleted flight.</param>

<returns>The index page for flights.</returns>

</member>

<member name="M:Web.Controllers.FlightsController.FlightExists(System.Int32)">

<summary>

Checks if a flight exists in the database.

</summary>

<param name="id">Primary key of the flight.</param>

<returns>A bool indicator which shows if the flight exists.</returns>

</member>

<member name="T:Web.Controllers.HomeController">

<summary>

Responsible for controlling the start page and successful start of the project

</summary>

</member>

<member name="M:Web.Controllers.HomeController.Index">

<summary>

Ensures that roles and administrator account exist on project start. The administrator has a default password which can be changed in the future.

</summary>

<returns>The home page</returns>

</member>

<member name="T:Web.Controllers.PassengersController">

<summary>

The controller provides CRUD operations for the passengers in the database.

</summary>

</member>

<member name="M:Web.Controllers.PassengersController.#ctor(Data.Context.FlightManagerDbContext)">

<summary>

Standard constructor which initializes a context used by the controller.

</summary>

<param name="context">The flight manager context.</param>

</member>

<member name="M:Web.Controllers.PassengersController.Index(System.String,Data.Models.PassengerIndexViewModel,System.Int32)">

<summary>

The Passengers controller for the Index page. Supports pagination and filtration

</summary>

<param name="searchString">string for filtering flights with given FirstName,MiddleName,LastName,PersonalIdentificationNumber</param>

<param name="model">PassengerIndexViewModel object used to display filtered or paged records from Passengers</param>

<param name="pages">int for the amount of records to be shown on a single page</param>

<returns>Returns Passengers Index View with the filtered Passengers(if entered search string) otherwise pages all records</returns>

</member>

<member name="M:Web.Controllers.PassengersController.Details(System.Nullable{System.Int32})">

<summary>

Shows a detailed view of a passenger.

</summary>

<param name="id">Primary key of passenger.</param>

<returns>Passenger detail view.</returns>

</member>

<member name="M:Web.Controllers.PassengersController.Create(System.String)">

<summary>

Prepares information about the reservation which is used in the creating form for passengers.

</summary>

<param name="reservationId">Primary key of the reservation for which the passenger is made.</param>

<returns>Passenger creation view.</returns>

</member>

<member name="M:Web.Controllers.PassengersController.Create(Data.Models.Passenger,System.String)">

<summary>

Creates a passenger which goes to a pending list. This process repeats until the user stops adding passengers to the reservation.

After the users confirmation a final method for finishing the reservation is called.

</summary>

<param name="passenger">A passenger created by the user</param>

<param name="finishAdding">A bool value which indicates if the user has finished adding passengers to the reservation.</param>

<returns>The <see cref="M:Web.Controllers.PassengersController.FinishReservation"/> method.</returns>

</member>

<member name="M:Web.Controllers.PassengersController.FinishReservation">

<summary>

Finishes the reservation of passengers. It adds the pending passengers and reservation to the database if it is possible

and updates the free flight seats number. An email is sent to the email address provided by the user for the reservation.

Depending on the circumstances this email will notify the user if the reservation has been successful or not by providing

information about the flight and passengers related to the reservation.

</summary>

<returns></returns>

</member>

<member name="T:Web.Controllers.ReservationsController">

<summary>

A controller which provides CRUD operations for the reservations in the database.

</summary>

</member>

<member name="M:Web.Controllers.ReservationsController.#ctor(Data.Context.FlightManagerDbContext)">

<summary>

Standard constructor which initializes a context used by the controller.

</summary>

<param name="context">The flight manager context.</param>

</member>

<member name="M:Web.Controllers.ReservationsController.Index(System.String,Data.Models.ReservationIndexViewModel,System.Int32)">

<summary>

The Reservations controller for the Index page. Supports pagination and filtration

</summary>

<param name="searchString">string for filtering Reservations with given Email</param>

<param name="model">ReservationIndexViewModel object used to display filtered or paged records from Reservations</param>

<param name="pages">int for the amount of records to be shown on a single page</param>

<returns>Returns Reservations Index View with the filtered Reservations(if entered search string) otherwise pages all records</returns>

</member>

<member name="M:Web.Controllers.ReservationsController.Details(System.Nullable{System.Int32})">

<summary>

The Reservation controller for the Details page. Shows details about the Reservation and the Passengers in it

</summary>

<param name="id">int id for the reserved Flight</param>

<returns>Returns Reservation Detail page showing information for both the Reservation and the Passengers</returns>

</member>

<member name="M:Web.Controllers.ReservationsController.Create">

<summary>

Prepares flight information for reservation creation and sends it to the reservation creation view.

</summary>

<returns>Reservation creation view.</returns>

</member>

<member name="M:Web.Controllers.ReservationsController.Create(Data.Models.Reservation)">

<summary>

Creates a pending reservation and redirects towards passenger creation method because the reservation needs passengers.

</summary>

<param name="reservation">The reservation created by the user</param>

<returns><see cref="M:Web.Controllers.PassengersController.Create(System.String)"/></returns>

</member>

<member name="T:Web.Controllers.UsersController">

<summary>

The controller which is responsible for user CRUD operations. Accessible only by the administrator.

</summary>

</member>

<member name="M:Web.Controllers.UsersController.#ctor(Data.Context.FlightManagerDbContext)">

<summary>

Standard constructor which initializes a context used by the controller.

</summary>

<param name="context">The flight manager context.</param>

</member>

<member name="M:Web.Controllers.UsersController.Index(System.String,Data.Models.UserIndexViewModel,System.Int32)">

<summary>

The Users controller for the Index page. Supports pagination and filtration

</summary>

<param name="searchString">string for filtering users with given UserName,FirstName,LastName,Email</param>

<param name="model">UserIndexViewModel object used to display filtered or paged records from Users</param>

<param name="pages">int for the amount of records to be shown on a single page</param>

<returns>Returns Users Index View with the filtered Users(if entered search string) otherwise pages all records</returns>

</member>

<member name="M:Web.Controllers.UsersController.Details(System.String)">

<summary>

Returns a detailed view for a user.

</summary>

<param name="id">Primary key of the user</param>

<returns>User detail view.</returns>

</member>

<member name="M:Web.Controllers.UsersController.Create">

<summary>

Redirects to the user registration form.

</summary>

<returns>User registration form.</returns>

</member>

<member name="M:Web.Controllers.UsersController.Edit(System.String)">

<summary>

Finds an user from the database using its primary key and prepares it for edit.

</summary>

<param name="id">Primary key of the user</param>

<returns>User edit view.</returns>

</member>

<member name="M:Web.Controllers.UsersController.Edit(System.String,Data.Models.UserEditViewModel)">

<summary>

Takes an edited <see cref="T:Data.Models.User"/> object and saves its changes to the database.

</summary>

<param name="id">Primary key of the user</param>

<param name="model">The edited user.</param>

<returns>The index page for users.</returns>

</member>

<member name="M:Web.Controllers.UsersController.Delete(System.String)">

<summary>

Extracts an user from the database and prepares it for delete.

</summary>

<param name="id">Primary key of user.</param>

<returns>User delete view.</returns>

</member>

<member name="M:Web.Controllers.UsersController.DeleteConfirmed(System.String)">

<summary>

Deletes an user from the database after receiving a confirmation. The administrator account cannot be deleted.

</summary>

<param name="id">Primary key of user</param>

<returns>The index page for users.</returns>

</member>

<member name="M:Web.Controllers.UsersController.UserExists(System.String)">

<summary>

Checks if an user exists in the database by searching for it with primary key.

</summary>

<param name="id">Primary key of user</param>

<returns>A bool value which indicates if the user exists.</returns>

</member>

<member name="T:Web.Pagers.FlightPager">

<summary>

Static class to store searched string and paging amount for Flights

</summary>

</member>

<member name="T:Web.Pagers.PassengerPager">

<summary>

Static class to store searched string and paging amount for Passengers

</summary>

</member>

<member name="T:Web.Pagers.ReservationPager">

<summary>

Static class to store searched string and paging amount for Reservations

</summary>

</member>

<member name="T:Web.Pagers.UserPager">

<summary>

Static class to store searched string and paging amount for Users

</summary>

</member>

<member name="T:Web.Startup">

<summary>

This class is used for configuring startup options for the flight manager web application.

</summary>

</member>

</members>

</doc>