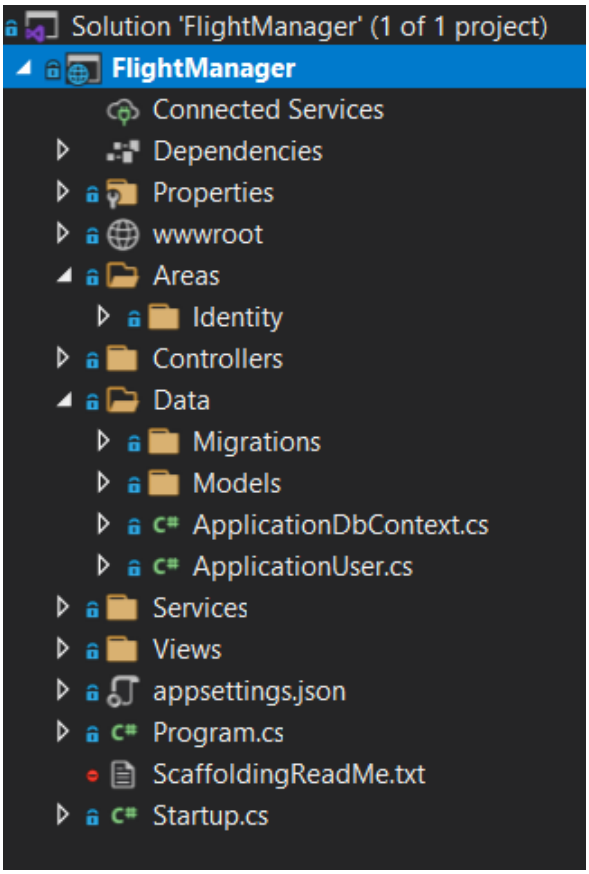
Мениджър за самолетни полети

FlightManager – ASP.NET Core Web App (MVC)

FlightManager е уеб базирано **Model-View-Controller** приложение, реализирано, чрез **ASP.NET Core** платформата, с използвани **EntityFramework** и **CRUD операции**, и **Identity Authorization,** което представлява мениджър на самолетни полети, резервирането им и също мениджър на потребителите.

1. Структура на проекта

Проектът се състои от слой за данни, бизнес слой и презентационен такъв, които са представени, чрез съдържанието на папките съответно Data, Services, Controllers и Views.



2. Реализация на проекта

При създаването на проекта със зададената ауторизация, автоматично се генерира контекст класът, който използва базата от данни за потребителите и папката с миграции, заедно с празните папки за Изгледите и

Контролерите. За реализирането на проекта по дадените изисквания е нужно да модифицираме класът с данните, които ще притежава един потребител, като това се случва със създаването на нов клас, който в случая е **ApplicationUser**, който наследява клас **IdentityUser**, автоматично генериран при създаването на проекта. В **ApplicationUser** дописваме липсващите полетата за нашия потребител: **FirstName**, **Surname**, **SSN** (ЕГН) и **Address**. Създаваме папката **Models**, в която се добавяме класът **Flight** и **FlightBooking**, по чиито модел ще се създадат таблиците за полетите и резервациите, като между тях има one-to-many връзка, като повече от една резервация може да бъде за същият полет. След което пренаписваме контекст класът за добавянето на новите таблици и употребата на модифицираният вече потребител. След което, използвайки Package Manager конзолата добавяме нова миграция и обновяваме базата данни. В същата тази папка с модели, по-късно се добавят и моделите за изгледите от папката Views.

Създаваме папката Services, която представлява бизнес логиката на приложението и в която се намират класовете **FlightService**, **ReservationService** и **EmailSender**, заедно с папката Contracts, която се намират съответните интерфейси за тези класове. Чрез тях по-лесно ще можем да управляваме действията, които извършваме с полетите и резервациите. В тях са реализирани CRUD (create-read-update-delete) операциите. А чрез класът EmailSender и неговият интерфейс ще управляваме изпращането на имейли, които ще бивat автоматично генерирани и изпратени за всеки бъдещ пътник за потвърждение при направена резервация, и също така, в случай на изтриването на даден полет, имейли за отмяната на този полет ще бъдат изпратени за всички пътници, които преди това са го резервирали.

След което променяме **Startup.cs** класа, добавяйки новите services, също така променяме service-а за потребителите, като вместо IdentityUser използваме вече новосъзданият ApplicationUser. На още няколко места в програмата се налага да променим IdentityUser на ApplicationUser.

Чрез Identity Scaffolding генерираме липсващите класове и изгледи за Register и тези от папката Manage, които ще модифицираме, за проекта ни, добавяйки новите полета и валидации за ApplicationUser, правейки Register достъпно единствено за потребител с администраторска роля, тъй като първоначалното стартиране на програмата започва с вградени Admin и User роли и един потребител с администраторска роля. Кодът за тях е реализиран в **Index** методът на **HomeController**. Потребителите с администраторска роля имат пълен достъп до всичките функции и операции (създаване, редактиране, изтриване и разглеждане на детайли), свързани с

управлението на полети, резервации и потребители, докато тези с роля User могат единствено да управляват резервациите и да виждат детайлите за полетите.

Създаваме класове-контролери в папката **Controllers** за полетите, резервациите и потребителите, реализирайки в тях действията за началната (**Index**) страница, **Create**, **Edit**, **Delete** и **Details** и съответните за тях изгледи в папката **Views**, както и допълнителните изгледи **FullPlaneView**, **CannotDelete**, **NotFound**, **SuccessfulReservation** и **Confirm**, които извикваме в специални случаи като когато няма повече свободни места за дадена класа или самолетът като цяло, когато е невъзможно даден елемент да се изтрие, когато даден елемент не е намерен, при направата на резервация (представлява изглед, в който се изписва за потребителя да провери пощата си за имейл потвърждение) и при успешно потвърдена резервация.

Редактира се също **_Layout.cshtml** изгледът с промяна нивото на достъп до елементи за съответните роли.

За реализацията на изпращачът на имейли, използваме **gmail** като **smtp** клиент, с предварително създаден gmail адрес, данните за който записваме в **appsettings.json** файла.

Програмата работи без да се изисква потребителски вход от страна на клиентите, които могат да си резервират билет за съответен полет, следвайки линковете на Home страницата към полетите.

На страниците с полетите и потребителите има реализирани функции за търсене и филтриране.

Данни за вход с администраторски акаунт:
Username: *admin*
Password: *Admin.1*

Детайли за имейл адресът:
EmailAddress: *cloudexpress21@gmail.com*
Password: *cloudExpress.21*
Username*: cloudexpress21*

Линк на проекта в GitHub: *https://github.com/ami566/FlightManager*

Изготвили: *Амира Емин, Ади Хаджиев, Селви Хутев, Амина Бисерова и Дениза Дурева*