Технически университет - София

Филиал Пловдив

Дипломна работа

Тема: Web-базирана информационна система за отдаване на помещения под наем

Студент: Деница Павлова Христева

Фак №: 367400

Специалност: КСТ

Образователна степен: ОКС бакалавър

Факултет: ФЕА

Ръководител: гл. ас. д-р Добринка Петрова

ТУ – София, Филиал Пловдив, 2020 г.

Съдържание

[Увод 3](#_Toc44938194)

[ГЛАВА 1. Обзор – състояние на проблема по литературни данни 4](#_Toc44938195)

[1.1 Анализ на особености и нужди за фирми и лица, занимаващи се с отдаване на помещения под наем 4](#_Toc44938196)

[1.2. Изисквания към Web-системата 4](#_Toc44938197)

[1.2.1 Нефункционални изисквания. 4](#_Toc44938198)

[1.2.2. Функционални изисквания. 5](#_Toc44938199)

[1.3. Структура на постъпващата информация. 6](#_Toc44938200)

[ГЛАВА 2. Описание на структурата на базата данни 7](#_Toc44938201)

[ГЛАВА 3. Избор на програмни езици, програмна среда, сървър за бази данни и архитектурен модел 11](#_Toc44938202)

[3.1 Програмни езици 11](#_Toc44938203)

[3.2 Програмна среда. 12](#_Toc44938204)

[3.3 Система за управление на база данни 14](#_Toc44938205)

[3.4. Архитектурен модел. 14](#_Toc44938206)

[ГЛАВА 4. Описание на компонентите на Web-системата и тяхното взаимодействие 18](#_Toc44938207)

[4.1. База данни. 18](#_Toc44938208)

[4.1.1 Класове - модели в приложението (Models) 18](#_Toc44938209)

[4.1.2. Достъп до данните от базата 19](#_Toc44938210)

[4.2. Модели за изглед (View Models) 21](#_Toc44938211)

[4.2.1. Библиотека AutoMapper 22](#_Toc44938212)

[4.2.2. Валидиране на данни чрез анотации (Data Annotation Validations) 23](#_Toc44938213)

[4.3. Изгледи (Views) 24](#_Toc44938214)

[ Начален изглед (Index View) (фиг.6) 24](#_Toc44938215)

[4.4. Контролери (Controllers) 31](#_Toc44938216)

[ГЛАВА 5. Приложимост. Изводи. 32](#_Toc44938217)

[Използвани литературни и други източници: 33](#_Toc44938218)

[ПРИЛОЖЕНИЕ 34](#_Toc44938219)

## Увод

Задача на настоящата дипломна работа е създаване на Web-базирана информационна система за отдаване на помещения под наем. В процеса на осъществяване на проекта са предприети действия, осигуряващи надеждни теоретични и софтуерни решения. Направени са проучвания за определяне както степента на приложимост, така и за целевата аудитория на бъдещата Web-система. Възоснова на тях е изградена представа за подходяща организация на базата данни и Web-приложението, идеи за потребителския интерфейс, визуализиране на графичните ресурси и статичната информация.

Избора на използваните технологии се базира на необходимостта от устойчива програмна среда за справяне със системните изисквания, характеристиките и планираните функционалности на Web-приложението. То ще е от тип клиент-сървър и като такова предвижда, наред с изграждането на модули с различни функции и логически издържано взаимодействие между тях, да се избере и подходящ дизайн, който не утежнява хода на действие на потребителя.

Цели се системата да отговаря на условията за бързодействие, увеличен капацитет, повишена надеждност, лесно откриваема информация и конкурентноспособност.

Приложението следва да поддържа функционалоност за регистриране на потребители, които ще имат опция както да добавят свое място за отдаване под наем, така и да наемат от вече наличните в базата данни на системата. Ще бъде изградена рейтингова система за поставяне коментари и оценки само от регистрирани потребители, които ще сформират обща оценка за дадено място. Приложението ще разполага и с контролна част. Тя ще разрешава на оторизиран потребител - администратор - да прави промени в съдържанието.

Съставена е база данни, която да съхранява информацията от споменатите дейности, а именно - данни за потребители, данни за обекти и техните характеристики, данни за резервации. За целта е използван Miscrosoft SQL Server. Програмната среда за разработване на Web-приложението е Microsoft Visual Studio 2019, .NET Core.

# ГЛАВА 1. Обзор – състояние на проблема по литературни данни

## 1.1 Анализ на особености и нужди за фирми и лица, занимаващи се с отдаване на помещения под наем

Tехнологиите са неизменна част от ежедневието на съвременния човек. Все повече обществени сфери се дигитализат – информацията и услугите стават все по-достъпни в Интернет пространството. Вследствие на това, дигиталната грамотност се превръща в жизнено важно условие за участие в съвременната икономика и общество. Веднъж създадена, информацията в Интернет, може да бъде посетена от всеки, който има интерес към нея. Сега съществуват електронни пазари, благодарение на които хората могат да закупуят определена стока или услуга от дома си, като това им дава възможност да спестят време и пари. В това число иновациите в технологиите довеждат до софтуерни решения за оптимизиране на процесите по наемане на обекти. Рационализират се бизнес операциите, автоматизират се определени задачи, премахва се ръчната работа за въвеждане и съхранение на данни, съответно се намалява или дори изключва риска от допускане на грешки, дублиране на информация и др.

Българските наемодатели все повече осъзнават потенциала на дигиталните технологии в подкрепа на техния бизнес и предприемат стъпки за използването им. Най-силно застъпените технологии към момента са мобилните и Web-приложенията.

## 1.2. Изисквания към Web-системата

### 1.2.1 Нефункционални изисквания.

**Безопасност на системата**

Системата трябва да отговаря на изисквания за сигурност на данните. Съобразено с това, че всички данни, които Web-приложението използва, се намират на сървърен компютър, безопасността във физическия смисъл е свързана с надеждността на самия сървър. Това включва предпазване от токов удар, сигурност и стабилност на базата данни.

**Системна производителност**

За да бъде пълноценна системата, всички страници, включително и тези, които се генерират при използване на търсачката в приложението, не трябва да отнемат повече от 5 секунди, за да се заредят. Това се налага, тъй като при по-бавното зареждане на страниците, е възможно да се появи загуба на интерес от страна на посетителя и той да напусне приложението неудовлетворен.

**Поддръжка на системата**

Системата трябва да е лесна за поддръжка – да е разработена така, че да се модифицира лесно с цел добавяне или изваждане на функции, подобряване на бързодействието и отстраняване на грешки.

### 1.2.2. Функционални изисквания.

За да бъде функционално, полезно и лесно за работа, Web-приложението трябва да притежава някои основни компоненти:

Клиентска част – това е частта, която се показва, когато потребител зареди URL-адреса на сайта в своя браузър. Началната страница трябва да бъде оформена по лесен за употреба от потребителя начин и той да добие ясна представа за целта и значението на изграждане на подобно Web-приложение.

Вход в системата - ще се допуска след извършване на регистрация на потребител, която се осъществява с email и парола, а email-a ще се подразбира и като username. Настройките на потребителския профил включват добавяне на мобилен телефон, смяна на email-а с друг, промяна на паролата и настройване на двуфакторно удостоверяване (two-factor authentication). Вход в контролната част на сайта ще се осъществява само от определена страница, достъпът до която е възможен само след правилно въведени email и парола на администраторския профил.

Добавянето на помещение за отдаване под наем трябва да се случва след попълване на форма за данни за мястото като: характеристики, местоположение, снимки, екстри, условия и цена. Всички съществуващи обекти в системата ще са достъпни за преглед от всеки посетител на приложението, но опцията за резервация ще изисква регистрация и вход.

При желание за наемане на даден обект, след избор на дати и потвърждение с бутoн за финализиране на дейността, ще се появява съобщение относно извършения процес и потребителят ще се пренасочва към списък с всички негови резервации.

Търсенето също е полезна функционалност в тази система. Ще се предостави възможност да се намерят обекти спрямо зададена локация, брой наематели и предпочитани дати за наемане. След избор на критериите и натискане на бутона за търсене, системата трябва да извежда всички резултати, които отговарят на аргументите. Друга опция ще предлага списък от всички помещения в дадена категория, като информацията за тях ще се поднася в съкратен вид, което допринася за бързо ориентиране на посетителя.

Всеки обект в системата ще притежава и Web-страница с подробна информация за характеристиките му, данни за собственика и календар, посочващ свободните дати за резервиране. В нея ще се предостави възможността за поставяне на коментари и оценки, които ще са достъпни за преглед.

## 1.3. Структура на постъпващата информация.

Релационната база данни ще съхранява информация за:

* регистрирани потребители в системата
* категории за обекти под наем
* характеристики за обект
* снимки за обект
* коментари и оценки за обект
* региони и градове в България
* резервации

# ГЛАВА 2. Теоретично решение на поставената задача

Таблиците и връзките между тях са визуализирани на диаграмата по- долу (фиг.1), след което следват техните описания. Ще бъде използван следния формат на описание:

* (PK) – първичен ключ (Primary Key)
* (FK, Таблица) – външен ключ, релативна таблица (Foreign Key)



фиг. 1 - Схема на базата данни

Таблица **AspNetUsers** – съдържа информация за потребителите:

* Id(PK) – идентификационен номер
* UserName – потребителско име
* Email – имейл
* EmailConfirmed – дали е потвърден email-а
* PasswordHash – хеш на паролата
* PhoneNumber – телефонен номер
* PhoneNumberConfirmed – дали е потвърден тел. номер
* TwoFactorEnabled – дали е настроено двуфакторно удостоверяване
* IsAdmin – роля на потребителя

Таблица **Places** – съдържа информация за обектите:

* Id (PK) – идентификационен номер
* UserId (FK, AspNetUsers) – id на собственик
* CityId (FK, Cities) – id на град
* CategoryId (FK, Categories) – id на категория
* Аddress – адрес
* Description – описание
* PriceByNight – цена за нощувка
* MaxGuest – максимален брой гости
* BathroomsNum – брой бани
* Pets – дали са позволени домашните любимци
* Smoking – дали е позволено е пушенето
* BedroomsNum – брой спални стаи
* BedsNum – брой легла
* Rating – оценка

Таблица **Images** - съдържа информация за снимките:

* Id (PK) – идентификационен номер
* PlaceId (FK, Places) – id на място, към което принадлежи
* Name – име на снимка
* Path – път към снимка
* Ext – разширение на файла

Таблица **Еxtras** – съдържа информация за екстрите:

* Id (PK) – идентификационен номер
* Name – име на екстрата

Таблица **PlaceExtras** – съдържа информация за екстрите на един обект:

* Id (PK) – идентификационен номер
* PlaceId (FK, Places) – id на обектът, към който принадлежи екстрата
* ExtraId (FK, Extras) – id на екстра

Таблица **Reservations** – съдържа информация за резервациите:

* Id(PK) – идентификационен номер
* PlaceId (FK, Places) – id на мястото, за който е резервацията
* UserId (FK, AspNetUsers) – id на наемателя
* StartDate – начална дата
* EndDate – крайна дата
* PricePerNight – цена за нощувка
* TotalPrice – обща сума
* NumNights - продължителност в дни
* Reviewed – получена ли е обратна връзка
* Active – дали е активна

Таблица **Reviews** – съдържа информация за обратната връзка:

* Id(PK) – идентификационен номер
* PlaceId (FK, Places) – id на мястото, за което се отнася
* UserId (FK, AspNetUsers) – id на потребителя, който е изпратил обратната връзка
* Rating – оценка
* Comment – коментар

Таблица **Categories** - съдържа информация за категории на места за отдаване под наем:

* Id(PK) – идентификационен номер
* Name – име на категория
* ImageUrl – път към снимка за категория
* ImageName – име на снимка за категория

Таблица **Cities** - съдържа информация за градове:

* Id(PK) – идентификационен номер
* Name – име на град
* RegionId (FK, Regions) - id на областта, в която се намира градът

Таблица **Regions** - съдържа информация за области:

* Id(PK) – идентификационен номер
* Name – име на област

Всички таблици съдържат полета:

* CreatedOn – дата на създаване
* ModifiedOn – дата на промяна
* IsDeleted – информация за изтриване
* DeletedOn - дата на създаване

# ГЛАВА 3. Описание на софтуерната част

## 3.1 Програмни езици

**C#** е обектно-ориентиран език за програмиране, разработен от Microsoft, като част от софтуерната платформа .NET.  Програмите на C# представляват един или няколко файла с разширение .cs., в които се съдържат дефиниции на класове и други типове. Тези файлове се компилират от компилатора на C# (csc) до изпълним код и в резултат се получават асемблита – файлове със същото име, но с различно разширение (.exe или .dll).

Едно от най-големите предимства на .NET Framework е вграденото автоматично управление на паметта. То освобождава програмистите от сложната задача сами да заделят памет за обектите и да търсят подходящия момент за нейното освобождаване. Това сериозно повишава производителността на програмистите и увеличава качеството на програмите, писани на C#.За управлението на паметта в .NET Framework се грижи специален компонент от CLR, наречен „система за почистване на паметта“ (garbage collector). Основната задача на системата Garbage collector-а е да следи кога заделената памет за променливи и обекти вече не се използва, да я освобождава и да я прави достъпна за последващи заделяния на нови обекти.

Предимства на C#:

* C# е създаден като прост, модерен с общо предназначение и обектно-ориентиран език за програмиране.
* Езикът е предназначен за използване в развиващите се софтуерни компоненти, той е подходящ и за разполагане в разпределена среда.
* На езика C# и върху .NET платформата може да бъде разработван разнообразен софтуер, като офис приложения, уеб приложения, уеб сайтове, настолни приложения, мултимедийни Интернет приложения, приложения за мобилни телефони, различни видове игри и много други.

**JavaScript** е програмен език, който позволява динамична промяна на поведението на браузъра в рамките на дадена HTML страницата. JavaScript се зарежда, интерпретира и изпълнява от Web браузъра, който му осигурява достъп до Обектния модел на браузъра. JavaScript функции могат да се свържат със събития на страницата (например: движение/натискане на мишката, клавиатурата или елемент от страницата, и други потребителски действия). Прието е JavaScript програмите да се наричат скриптове.

JavaScript е един от най-използваните скриптови езици за програмиране в Интернет. Toзи език осъществява моментното и динамично взаимодействие между потребител и браузър. Благодарение на JavaScript Web страниците, които използваме в момента, изпълняват много повече функции от това просто да зареждат данни. JavaScript позволява създаването на интерактивни и адаптивни интерфейси с много добър дизайн и динамични функционалности, изцяло с насоченост към крайния потребител.

**HTML** е основният маркиращ език за описание и дизайн на уеб страници. Той осигурява множество от специални елементи, които описват как трябва да изглежда една уеб страница.

**Cascading Style Sheets(CSS)** - определя изгледа и оформлението на текста и други материали. HTML елементите са основната градивна единица на уеб страниците. Чрез тях се оформят отделните части от текста на една уеб страница като заглавия, цитати, раздели, хипертекстови препратки и т.н.

**Bootstrap** е client-side среда с отворен код, която съдържа набор от инструменти за създаване на Web приложения и Webсайтове. В Bootstrap са включени HTML и CSS дизайн шаблони за типография, форми бутони, навигация и други компоненти за интерфейса на Web-приложението и/ли Web-страниците. Има няколко вградени файлове, които не е препоръчително да се променят. Тези файлове са с разширения .css и .js. Съдържа няколко JavaScript компонента под формата на jQuery плъгини. Те предоставят допълнителни потребителски интерфейс елементи като диалогови прозорци и пояснения. Също така разширяват функционалността на някои съществуващи интерфейс елементи. Характеристиките, които правят Bootstrap предпочитан за използване са: лесна приспособимост, responsive дизайн, перфектна grid система, обширен списък на компонените, пакетни Javascript плъгини, предоставени стилове за всички основни HTML елементи и добра документация.

## 3.2 Програмна среда.

**Microsoft Visual Studio** е среда за разработка на софтуерни приложения за Windows и за платформата .NET Core, наследник на .NET Framework. Използва се за разработка на конзолни и графични потребителски интерфейс приложения, както и Windows Forms или WPF приложения, Web сайтове, Web приложения и Web услуги на всички поддържани платформи от Microsoft Windows, Windows Mobile, Windows CE, .NET Framework, .NET Core, .NET Compact Framework и Microsoft Silverlight.

Visual Studio предоставя мощна интегрирана среда за писане на код, компилиране, изпълнение, дебъгване (както за високо така и за машинно ниво), тестване на приложения, дизайн на потребителски интерфейс (форми, диалози, Web страници, визуални контроли и други), моделиране на данни, моделиране на класове, изпълнение на тестове, пакетиране на приложения и стотици други функции. Могат да се добавят и плъгини, които повишават функционалността на почти всяко ниво – включително добавянето на поддръжка за source-control системи (като Subversion и Visual SourceSafe или GitHub).

Visual Studio поддържа различни езици за програмиране (C#, VB.NET, C/C++, F#, XML/XSLT, HTML/XHTML, JavaScript, CSS и други) и различни технологии за разработка на софтуер (Win32, COM, ASP.NET Core, ADO.NET Entity Framework, Windows Forms, WPF, Silverlight и още десетки други Windows и .NET технологии).

Версията Visual Studio 2019 предлага много подобрения на производителността в подкрепа на работата с приложения за Windows, разработката на мултиплатформени мобилни приложения, Web и cloud development и др.

**ASP.NET Core**

ASP.NET Core е многоплатформенa рамка (cross-platform framework) с отворен код. Предоставя възможност за създаване Web приложения и услуги, IoT приложения, програмни модули за мобилни приложения и всякакъв вид Web-базирани solutions. Програмният продукт може да бъде „качен“ в облачно пространство (например Azure) или да се ползва локално. ASP.NET Core притежава отлична документация. ASP.NET Core осигурява интеграция на модерни рамки от страна на клиента (Angular, Blazor и др.) и потоци за разработка (MVC, WebAPI, Razor Pages, SignalR). Приложенията на ASP.NET Core се поддържат и могат да бъдат стартирани и от .NET Framework.

**Entity Framework Core**

Entity Framework (EF) Core е многоплатформена версия на популярната технология за достъп до данни Entity Framework. EF Core служи като обектно-релационен mapper (O/RM), като позволява на .NET разработчиците да работят с база данни, посредством .NET обекти.

В EF Core достъпът до данни се осъществява с помощта на модел. Един модел се съставя от entity класове и контекстен обект (DbContext), който представлява сесия с базата данни и позволява да се подават заявки и да се запазват данни. Моделът може да се генерира от вече съществуваща база данни (Database First); да се напише програмен код за модел, съответстващ на такъв в базата (Model First); или да се използват EF миграции за създаване на база данни от модел (Code First).

## **3.3 Система за управление на база данни**

**Microsoft SQL Server** е система за управление на релационни бази данни, разработена от Microsoft. В тази си роля (на система за управление) SQL Server има основната функция да съхранява и извлича данни, по заявки на други софтуерни приложения, които могат да вървят на същия или друг компютър в дадена мрежа (в т.ч. и Интернет). MS SQL Server има над дузина различни издания, които са подходящи за различни аудитории и различни натоварвания: от приложения за отделни персонални компютри до масивни приложения, работещи постоянно в онлайн режим и обхващащи огромно количество компютри. Microsoft SQL Server има много висока степен на защита.

SQL Server Management Studio(SSMS) **–** е софтуерно приложение, което се използва за конфигуриране, управление и администриране на всички компоненти в SQL Server. То съчетава широка група от графични инструменти с голям брой текстови редактори, осигуряващи на разработчиците и администраторите всички нива на достъп до сървъра.

## 3.4. Архитектурен модел.

Избраният архитектурен шаблон за дизайн на приложението е **Модел-Изглед-Контролер** (Model-View-Controller или MVC).  Архитектурният модел на Model-View-Controller (MVC) разделя приложенито на три основни групи компоненти: модели, изгледи и контролери. Този шаблон на работа помага да се постигне отделяне на бизнес логиката от графичния интерфейс и данните в едно приложение. Потребителските заявки се препращат към контролер, който е отговорен за работата с модела за извършване на потребителски действия и/или извличане на резултати от заявки. Контролерът избира изгледа, който да се покаже на потребителя, и му предоставя всички данни от модела, които той изисква.

**Модел** – ядрото на приложението, предопределено от областта, за която се разработва; обикновено това са данните от реалния свят, които се моделират и над които се работи – въвеждане, промяна, показване и т.н. Трябва да се прави разлика между реалния обкръжаващ свят и въображаемият абстрактен моделен свят, който е продукт на разума, който се възприема като твърдения, формули, математическа символика, схеми и други помощни средства. Например в приложението това са класовете, описващи потребителите, техните обекти, резервации, които са осъществили и т.н.

**Изглед** (англ. View) – тази част от изходния код на приложението, отговорна за показването на данните от модела. Изглед модела (view model) позволява да се оформят няколко изгледа (views) от един или повече модели (models). Този модел е оптимизиран за потребление и изпълнение. В случая изгледът се състои ASP страници.

**Контролер** – тази част от сорс кода (клас или библиотека), която взима данните от модела или извиква допълнителни методи върху модела, предварително обработва данните, и чак след това ги дава на изгледа. Контролерите са класове, които се създават в MVC приложението. Намират се в папка Controllers. Всеки един клас, който е от този тип, трябва да има име завършващо с наставка „Controller“. Контролерите обработват постъпващите заявки, въведени от потребителя и изпълняват подходящата логика за изпълнение на приложението. Класът контролер е отговорен за следните етапи на обработка: намиране и извикване на най-подходящия метод за действие (action method) и валидиране, че може да бъде извикан, взимането на стойности, които да се използват като аргументи в метода за действие, отстраняване на всички грешки, които могат да възникнат по време на изпълнението метода за действие.

Това разграничаване на отговорности помага при писане и актуализиране на програмен код, отстраняване на грешки и тестване даден на компонент (модел, изглед или контролер). Тези действия ще са по-сложни ако имат зависимости в две или повече от тези три области. Например логиката за потребителския интерфейс има тенденция да се променя по-често от бизнес логиката. Ако презентационният код и бизнес логиката се комбинират в един общ компонент, то този, който съдържа бизнес логиката, ще трябва да се променя при всяка актуализация на потребителския интерфейс. Това често довежда до грешки и изисква повторно тестване на бизнес логиката след всяка минимална промяна на потребителския интерфейс.

**Предимства на MVC:**

* Моделът е независим от контролера и изгледа.
* Моделът може да бъде планиран и осъществен независимо от другите части на системата.
* За един и същи модел могат да бъдат осъществени различни изгледи (интерфейси) – например Web интерфейс и нативен интерфейс към Facebook.
* Контролерът и изгледът могат да бъдат променени, без да се налага промяна в модела.

**Недостатъци на MVC:**

* Софтуерната система достига по-високо ниво на сложност откъм четимост и боравене с нея.

**ASP.NET Core MVC – Основни понятия**

**Маршрутизация (Routing)**

ASP.NET Core MVC е изграден на основата за маршрутизация в ASP.NET Core - мощен компонент за URL mapping, който позволява създаването на приложения с разбираеми и леснодостъпни URL адреси. Това дава възможност на разработчиците да именуват URL адреси, които работят добре с механизмите за оптимизация на търсачките (SEO) в Интернет пространството.

**Обвързване на модели (Model Binding)**

Обвързването на моделите на ASP.NET Core MVC преобразува данните от клиентските заявки (като например попълване на форми) в обекти, с които контролерът може да се борави. В резултат на това логиката на контролера не трябва да върши работата по определяне на вида на данните от входящите заявки; той просто взима данните като параметри в своите методи.

Пример:

public async Task<IActionResult> Login(LoginViewModel model, string returnUrl = null) { ... }

**Валидиране на модели ( Model validation)**

ASP.NET Core MVC поддържа валидиране чрез добавяне на атрибути за анотация на данните към модела. Те служат за проверка на валидността на подаваните стойности още от страна на клиента, преди да бъдат прехвърлени към сървъра, а в друг случай - откъм сървърна страна - преди да се извика действието на даден контролер.

Пример:

using System.ComponentModel.DataAnnotations;

public class LoginViewModel

{

[Required]

[EmailAddress]

public string Email { get; set; }

[Required]

[DataType(DataType.Password)]

public string Password { get; set; }

[Display(Name = "Remember me?")]

public bool RememberMe { get; set; }

}

Метод на контролер:

public async Task<IActionResult> Login(LoginViewModel model, string returnUrl = null)

{

if (ModelState.IsValid)

{

// work with the model

}

// At this point, something failed, redisplay form

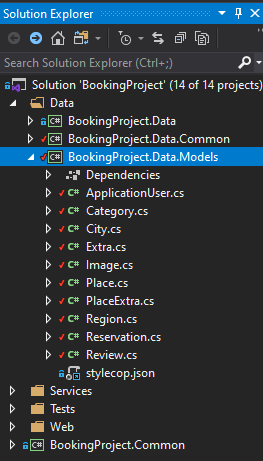
return View(model);

}

# ГЛАВА 4. Изчислителна част. Работа с програмната система. Функционално тестване

## 4.1. База данни.

Моделът на базата е изграден като библиотека от класове и всяка отделна таблица е описана като един клас. По този начин с таблиците се работи като с обекти в обектно-ориентираното програмиране (фиг.2).



фиг.2 Класове в приложението

### 4.1.1 Класове - модели в приложението (Models)

**ApplicationUser.cs -** клас, описващ таблицата AspNetUsers (потребители)

**Category.cs -** клас, описващ таблицата Categories (категории за места под наем)  
**City.cs -** клас, описващ таблицата Cities (градове)   
**Extra.cs -**клас, описващ таблицата Extras (екстри)

**Image.cs -** kлас, описващ таблицата Images (снимки)

**Place.cs -** клас, описващ таблицата Places (места за отдаване под наем)

**PlaceExtra.cs** - клас, описващ таблицата PlaceExtras (екстри към място)  
**Region.cs -** клас, описващ таблицата Regions (области)

**Reservation.cs** **-** клас, описващ таблицата Reservations (резервации)

**Review.cs** - **-** клас, описващ таблицата Reviews (ревюта)

### 4.1.2. Достъп до данните от базата

Някои от командите на EF Core Tools (например командите за миграции) изискват да се създаде инстанция на DbContext по време на проектиране (at design time), с цел събиране на подробности за типовете обекти на приложението и как те ще се проектират в схемата на базата данни. В повечето случаи е желателно създаденият DbContext да бъде конфигуриран по подобен начин, както би бил конфигуриран по време на изпълнение(at run time). Има различни начини за създадаване на DbContext. За разработеното приложение е избрано DbContext да бъде осигурен с помощта на Services от доставчика на услуги (Application Service Provider). ASP.NET Core използва depencency injection като основна функция за управление на зависимости. За да знае средата как да борави с тези зависимости или „services“, те първо трябва да бъдат конфигурирани.

public class Program

{

public static void Main(string[] args)

=> CreateHostBuilder(args).Build().Run();

// EF Core uses this method at design time to access the DbContext

public static IHostBuilder CreateHostBuilder(string[] args)

=> Host.CreateDefaultBuilder(args)

.ConfigureWebHostDefaults(

webBuilder => webBuilder.UseStartup<Startup>());

}

public class Startup

{ // This method gets called by the runtime. Use this method to add services to the container.

public void ConfigureServices(IServiceCollection services)

{

services.AddDbContext<ApplicationDbContext>(

options => options.UseSqlServer(this.configuration.GetConnectionString("DefaultConnection")));

}

}

public class ApplicationDbContext : IdentityDbContext<ApplicationUser, ApplicationRole, string>

{

public ApplicationDbContext(DbContextOptions<ApplicationDbContext> options)

: base(options)

{

}

public DbSet<Place> Places { get; set; }

public DbSet<Review> Reviews { get; set; }

public DbSet<Image> Images { get; set; }

public DbSet<Category> Categories { get; set; }

public DbSet<Reservation> Reservations { get; set; }

public DbSet<City> Cities { get; set; }

public DbSet<Region> Regions { get; set; }

public DbSet<Extra> Extras { get; set; }

public DbSet<PlaceExtra> PlaceExtras { get; set; }

}

Пример – CategoriesService.cs

namespace BookingProject.Services.Data

{

using System.Collections.Generic;

using System.Linq;

using BookingProject.Data.Common.Repositories;

using BookingProject.Data.Models;

using BookingProject.Services.Mapping;

public class CategoriesService : ICategoriesService

{

private readonly IDeletableEntityRepository<Category> categoriesRepository;

public CategoriesService(

IDeletableEntityRepository<Category> categoriesRepository)

{

this.categoriesRepository = categoriesRepository;

}

public IEnumerable<T> GetAll<T>(int? count = null)

{

IQueryable<Category> query =

this.categoriesRepository.All().OrderBy(x => x.Name);

if (count.HasValue)

{

query = query.Take(count.Value);

}

return query.To<T>().ToList();

}

public T GetByName<T>(string name)

{

var category = this.categoriesRepository.All()

.Where(x => x.Name.Replace(" ", "-") == name.Replace(" ", "-"))

.To<T>().FirstOrDefault();

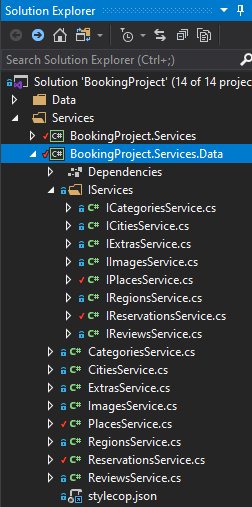
return category;

}

}

}

Папка Booking.Project.Data(фиг.3) съдържа всички Services, в които е описана логиката за извличане на данни от базата. Програмният код за тях е приложен в.

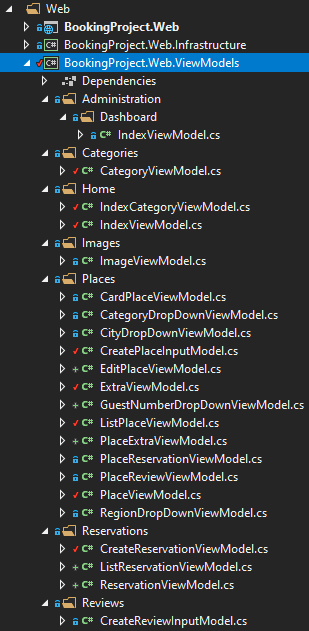
**

фиг.3 Структура на Service класовете

## 4.2. Модели за изглед (View Models)

Моделът на изглед е абстракция на изгледа(view), съставен от публични полета(public properties) и команди. Моделът на изглед е описан като състояние на данните в модела. За да функционира ефективно, се изисква свързваща технология или код. Тези модели определят правила, с които потребителя от клиентска страна трябва да се съобразява, за да се запази консистентност на данните в сървъра.

Създадените модели за изглед в Web-приложението са следните: (фиг.4)



фиг.4 Модели за изглед (View Models)

### 4.2.1. Библиотека AutoMapper

За създаване на зависимост (dependency injection) в приложението между класовете-модели (models) и моделите за изглед (view models) е използвана библиотеката AutoMapper. Тя служи за трансформиране на един тип обект в друг.

Пример:

**Основен модел**

public class User

{

public int Id { get; set; }

public string FirstName { get; set; }

public string LastName { get; set; }

public string Email { get; set; }

public string Address { get; set; }

}

**Модел за изглед**

public class UserViewModel

{

public string FirstName { get; set; }

public string LastName { get; set; }

public string Email { get; set; }

}

**Трансформиране (mapping):**

public UserProfile()

{

CreateMap<User, UserViewModel>();

}

public UserProfile()

{

CreateMap<User, UserViewModel>()

.ForMember(dest =>

dest.FName,

opt => opt.MapFrom(src => src.FirstName))

.ForMember(dest =>

dest.LName,

opt => opt.MapFrom(src => src.LastName))

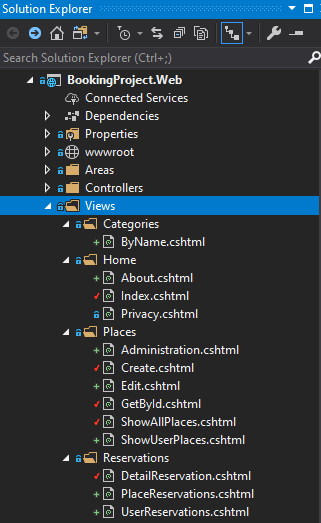
}

### 4.2.2. Валидиране на данни чрез анотации (Data Annotation Validations)

При създаване на mapping логика понякога може да има разминаване в именуването на полетата на основния модел и моделите за изглед. Затова се приема за добра практика валидационните атрибути за данни(data annotation validations ) да се описват и в моделите за изглед, като съответно извеждат на клиентска страна и подходящо съобщение за грешка.

## 4.3. Изгледи (Views)

Изгледите (Views) (фиг.5) са отговорни за представянето на съдържание в потребителския интерфейс. Те използват механизма за изглед на Razor (Razor view engine), за да вградят .NET код в HTML маркиране във файлове с разширение ''.cshtml''. Всяка логика в тях е минимална и трябва да се отнася само до представяне на съдържанието.



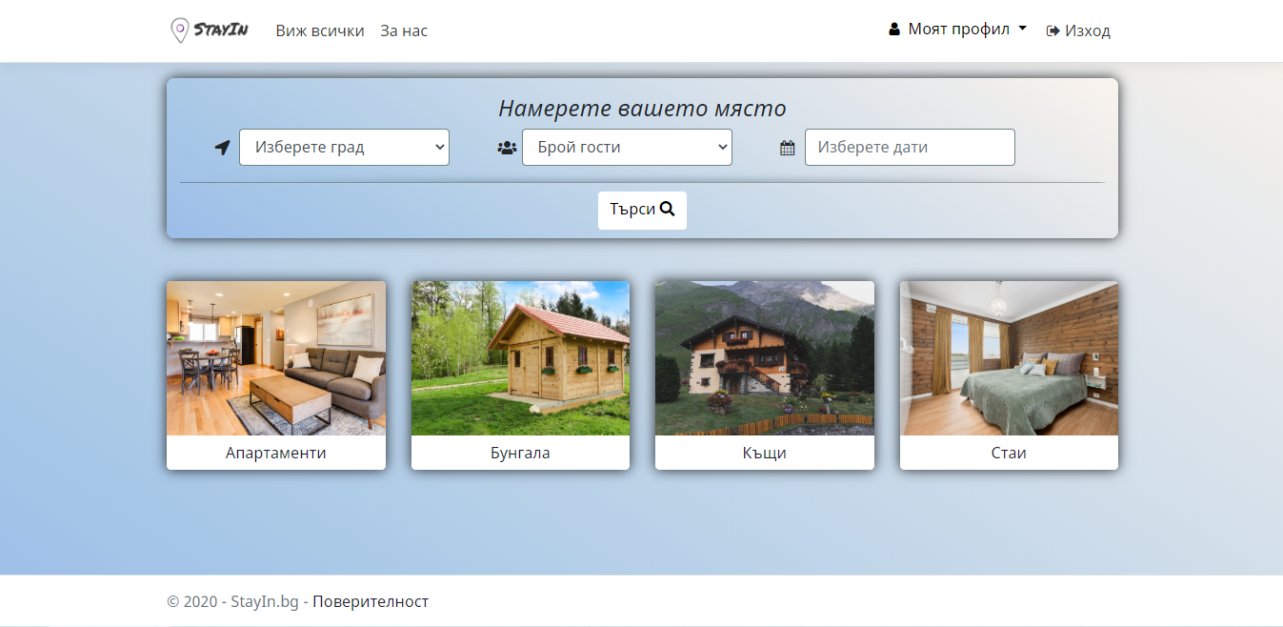
фиг.5 Изгледи (Views)

**Резултатите след компилиране на ''.cshtml'' – файловете, показани в браузър, са следните:**

### Начален изглед (Index View) (фиг.6)

Съдържа компонентите, които потребителят вижда при начално зареждане на Web-приложението. Един от тях е предоставя на потребителя филтри за търсене на подходящ за наем обект. Филтрите съдържат 3 полета: град, брой наематели и дати на наемане. Първите две представляват полета за въвеждане. При избор на последното поле се отваря календар, в който потребителя може да избере начална и крайна дата на резервацията. За подредба на резултатите е приложен и филтър за сортиране спрямо цената на обекта (по нарастващ или намаляващ ред).

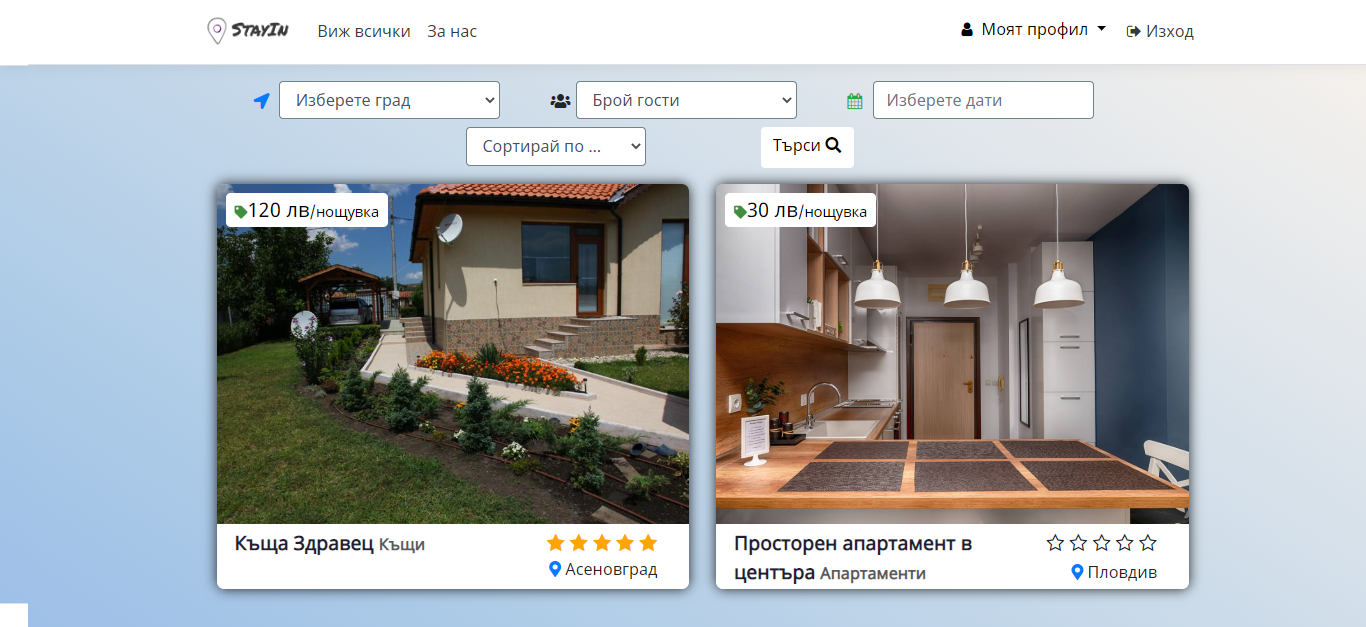
Друг вариант за търсене е да се търсят всички места от дадена категория без да се прилагат филтри.



фиг.6 Index View

* **Изглед за всички обекти в системата (ShowAllPlaces View)** (фиг.7)

Този изглед визуализира всички възможни обекти за наемане. Формата за търсене от началната страница също отвежда до него - ако има избрани филтри, резултатите се свеждат до тези, които отговарят на изискванията. За всеки обект е изведена основна информация за име, категория, град, оценка получена от ревюта. При посочване с мишката върху обект се появява и допълнителна информация за него.

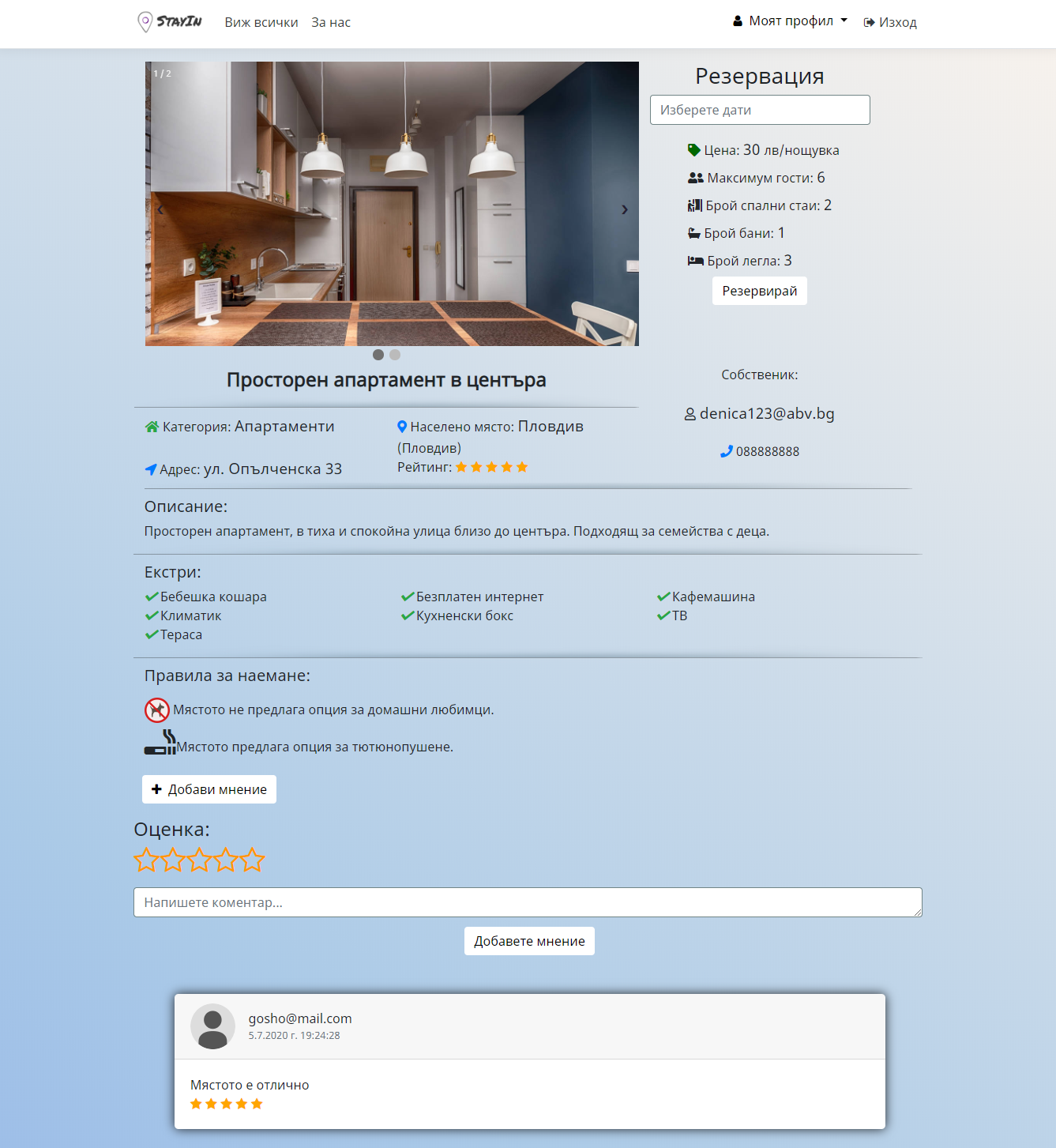


фиг.7 ShowAllPlaces View

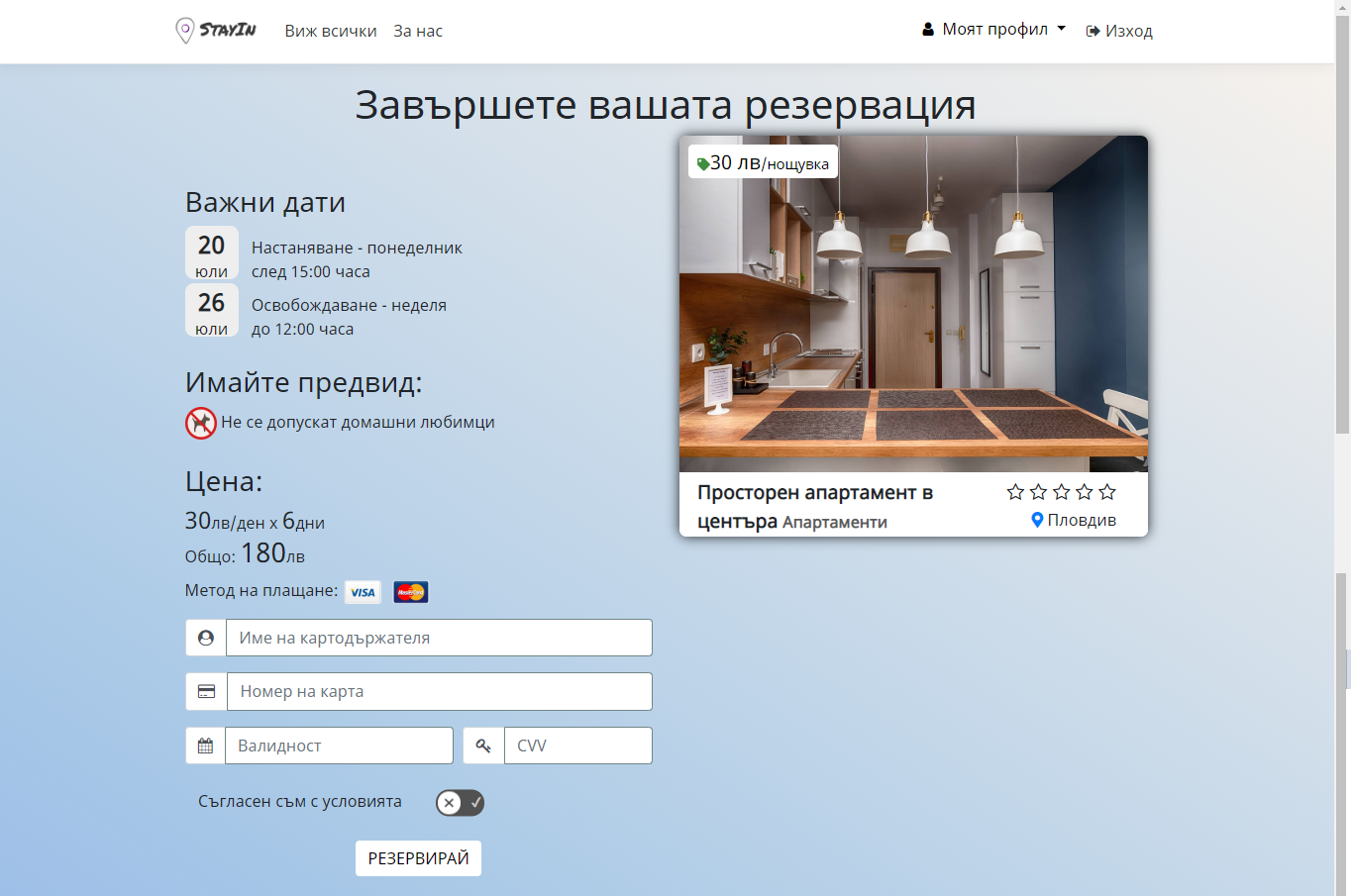
* **Изглед за обект по идентификационен номер (GetById View)** (фиг.8)

Изгледът визуализира детайлите за определен обект. Интерфейсът разполага с удобна галерия. След това в страницата има подробна информация за обекта и неговите екстри, а най-отдолу стои секцията за оценки и коментари.

На тази страница се намира и секцията за резервация. При избиране на дати за резервация, системата ще изиска от потребителя да се идентифицира чрез вход в системата или да бъде направена нова регистрация. Ако потребителят е вече в системата, то той ще се отведе към изгледа за потвърждение на резервацията. (фиг.9) Там са показват най-важните неща за нея, а именно – датите, правилата определени от собственика и обобщена информация за обекта, който предстои да се наеме. След въвеждане на данни за кредитна карта и съгласяване с условията, бутонът за резервация става активен. След успешно запазване, потребителят получава известие в попъп прозорец и бива пренасочен към страница, съдържаща всички негови предстоящи и предходни резервации.



фиг.8 GetById View



фиг.9 DetailReservation View

**Изгледите за потребителите на системата включват: изглед за преглед на резервации, за добавяне на собствено място под наем, за списък с всички негови места и за настройки на профила.**

* **Изглед за резервации, направени от даден потребител (UserReservations View)** (фиг.10)

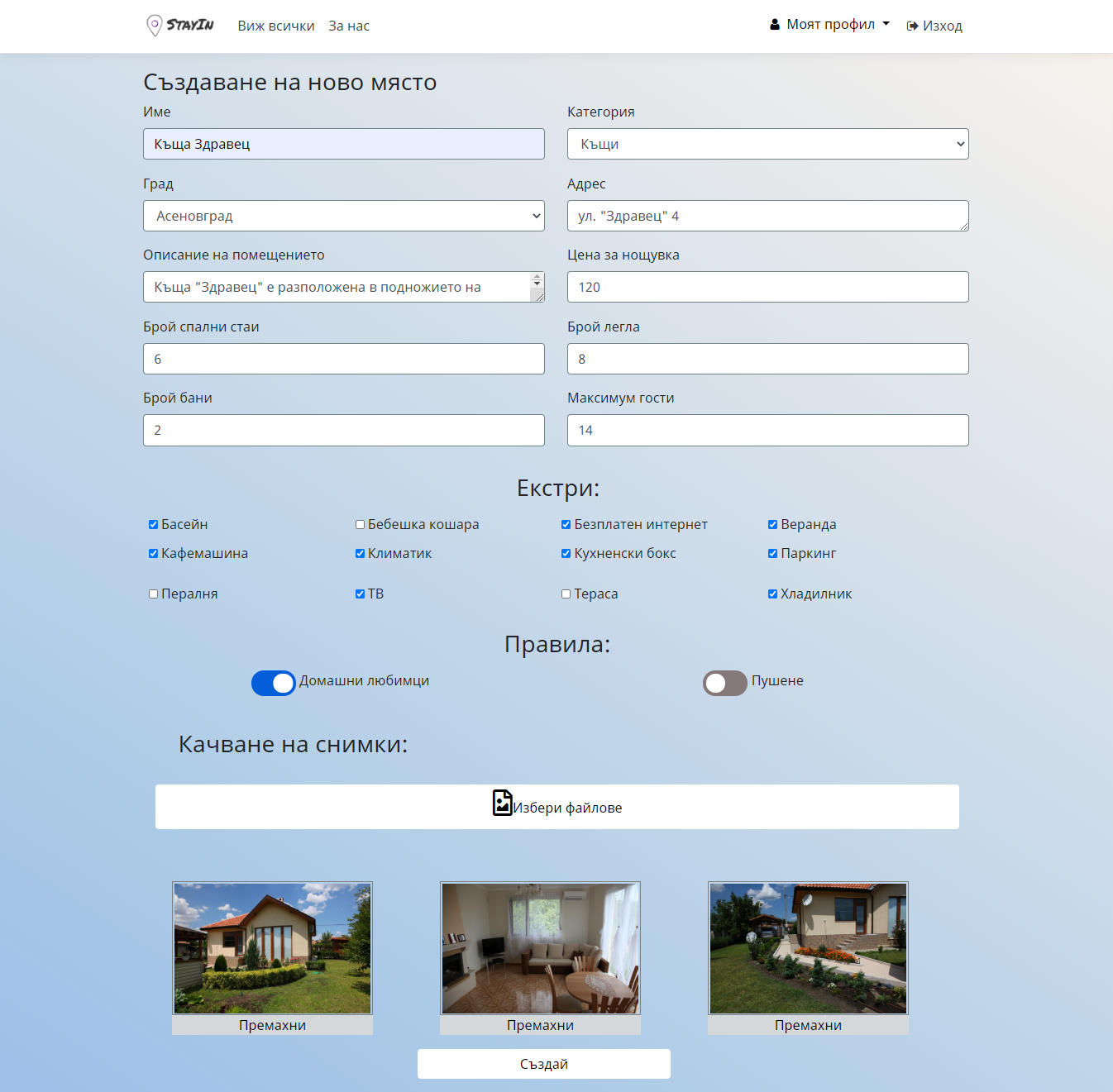
Резервациите се извеждат в списък от предстоящи и отминали. Във всеки запис за резервация се съдържа основна информация относно обекта, собственика, датите, съответно за колко дни е нает, както и сумата за престоя.



фиг.10 UserReservations View

* **Изглед за добавяне на ново място в системата (Create View)** (фиг.11)

Страницата за добавяне на новo място представлява форма, в която се въвежда пълната информация за новата обява. В началото се попълва основната информация като име, адрес, цена на нощувка и описание. Категория и град се зареждат се избират от падащо меню. Въвеждането на екстри и условията на обекта се осъществява с бутони от тип Checkbox. Собственикът трябва да прикачи поне една снимка, за да довърши операцията. Той има възможността да избере файлове чрез бутон, който отваря файловия мениджър. След тяхното прикачване снимките излизат подредени една до друга за финален преглед от потребителя преди да бъдат изпратени към сървъра. В този момент могат да бъдат премахнати или да се добавят още снимки.

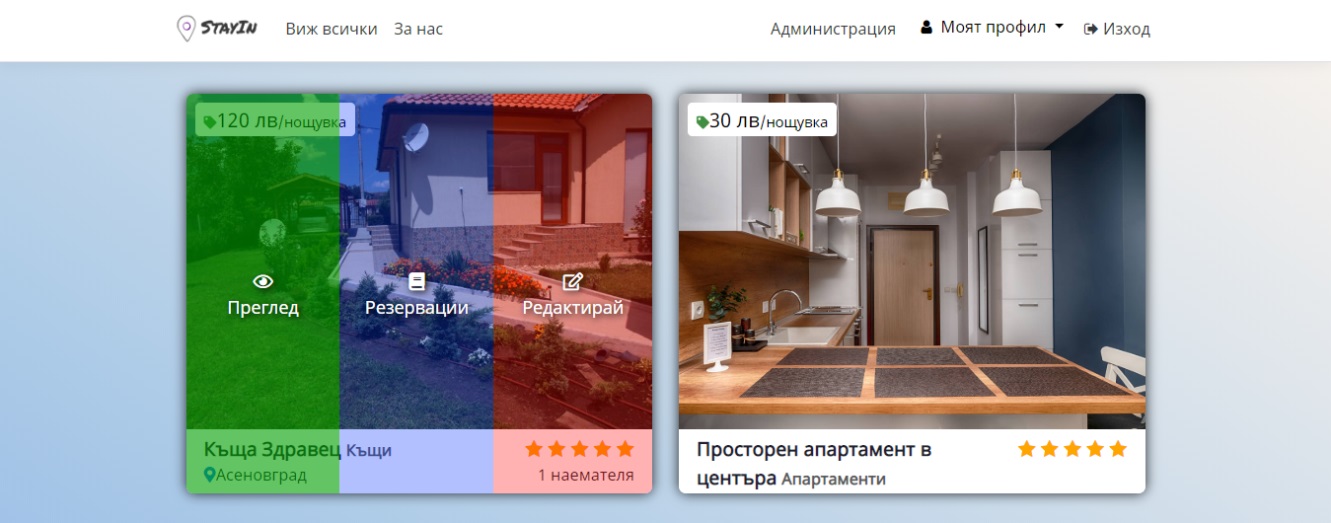


фиг.11 Create View

* **Изглед за управление на обектите (ShowUserPlaces View)** (фиг.12)

След създаване на обявата, потребителят се пренасочва към панела за управление. Извежда се списък от всички притежавани обекти на съответния юзър като при задържане на мишката върху някой обект се показват 3 възможности – Преглед, Резервации, Редактирай.

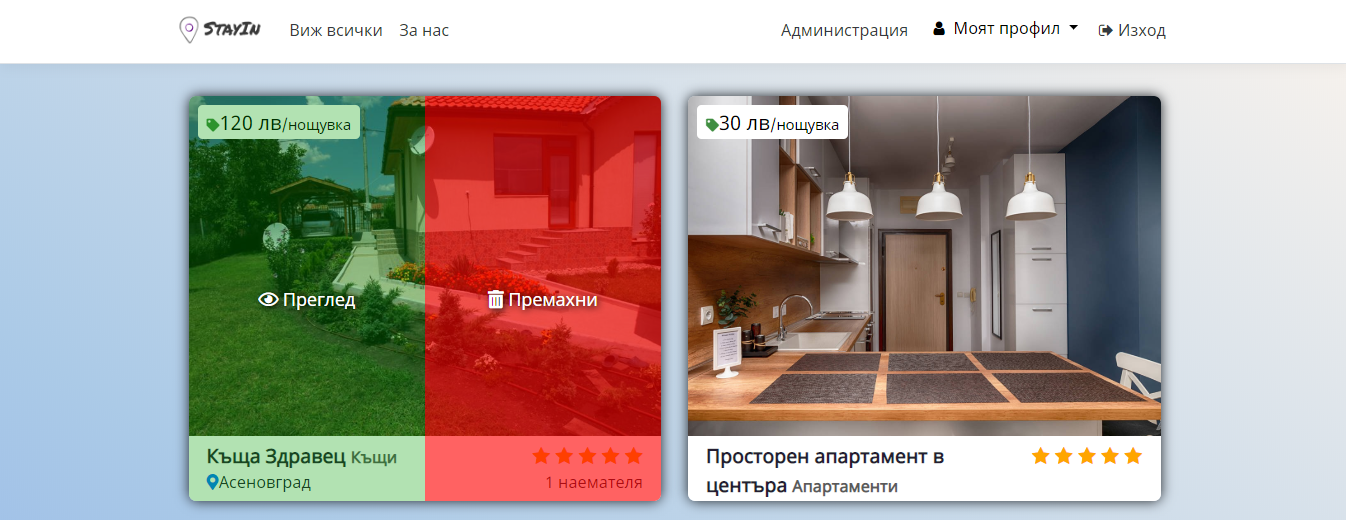
При избор на преглед потребителят се пренасочва към детайлната страница на обекта. В таб резервации се отваря страница, съдържаща информация за предстоящи или отминали резервации за конкретното място. Всеки запис съдържа данните за наемателя, датите, за които е запазена и общата сума за престоя.



фиг.12 ShowUserPlaces View

* **Изглед за контролен панел на администратор (Administration View)** (фиг.13)

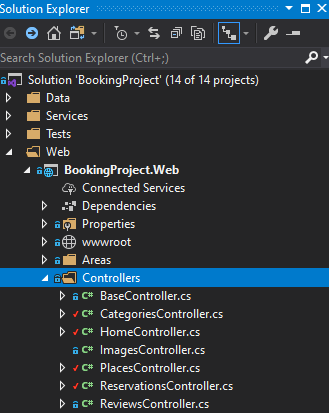
Администраторът на сайта разполага с допълнително поле „Администрация“ в главното меню. То отваря списък от всички съществуващи обекти в системата. При задържане на мишката върху някой от тях администраторът има опции за преглед и изтриване на мястото. Така той го премахва от Web- приложението, то спира да е видимо за юзърите, но данните за него остават в базата.



фиг.13 Administration View

## 4.4. Контролери (Controllers)

Контролерите(Controllers) в приложението (фиг.14) са компонентите, които управляват взаимодействието с потребителя, работят с модела и в крайна сметка избират изглед, който да изобразяват. В MVC модела, контролерът е входната точка за въвеждане и е отговорен за избора на кои модели да работи и кой изглед да изобрази (оттук и името му - контролира как приложението отговаря на дадена заявка).



фиг.14 Контролери (Controllers)

# ГЛАВА 5. Приложимост на дипломната работа. Изводи.

Разработена е Web-система за отдаване на помещения под наем, обхващаща всички основни нужди на един потребител. Основната цел на тази дипломна работа е създаде продукт, осигуряващ на ползвателя безпроблемно резервиране на място или добавяне на собствен обект в системата. Изграден е приятен и удобен за употреба интерфейс с адаптивен дизайн, който улеснява работата с приложението както на персонален компютър, така и на мобилно устройство.

От тази гледна точка настоящата дипломна работа е актуална, като описва концепцията и инструментите за проектиране, създаване и администриране на Web-приложение, като се използват съвременни технологии и е избрана тематика с добра приложимост.

Стига се до заключението, че развитието на познанията в компютърните науки позволява да се разработи лесноизползваем и достъпен софтуер, който улеснява значително операциите с данни и който би могъл да се внедри в най-различни сфери.

# Използвани литературни и други източници:

1. Lock. ASP.NET Core in Action, Manning Publications, 2018, 1st Edition, ISBN-10: 1617294616
2. <https://docs.microsoft.com/en-us/dotnet/core/>
3. <https://docs.microsoft.com/bg-bg/sql/ssms>
4. <http://www.w3schools.com>
5. <https://docs.microsoft.com/en-us/ef/>
6. <https://bg.wikipedia.org/wiki/C_Sharp>
7. <https://bg.wikipedia.org/wiki/Microsoft_Visual_Studio>

# ПРИЛОЖЕНИЕ

Seeders

namespace BookingProject.Data.Seeding

{

using System;

using System.Collections.Generic;

using System.Threading.Tasks;

using Microsoft.Extensions.DependencyInjection;

using Microsoft.Extensions.Logging;

public class ApplicationDbContextSeeder : ISeeder

{

public async Task SeedAsync(ApplicationDbContext dbContext, IServiceProvider serviceProvider)

{

if (dbContext == null)

{

throw new ArgumentNullException(nameof(dbContext));

}

if (serviceProvider == null)

{

throw new ArgumentNullException(nameof(serviceProvider));

}

var logger = serviceProvider.GetService<ILoggerFactory>().CreateLogger(typeof(ApplicationDbContextSeeder));

var seeders = new List<ISeeder>

{

new RolesSeeder(),

new SettingsSeeder(),

new CategoriesSeeder(),

new CitiesSeeder(),

new ExtrasSeeder(),

};

foreach (var seeder in seeders)

{

await seeder.SeedAsync(dbContext, serviceProvider);

await dbContext.SaveChangesAsync();

logger.LogInformation($"Seeder {seeder.GetType().Name} done.");

}

}

}

}

CategoriesSeeder.cs

namespace BookingProject.Data.Seeding

{

using System;

using System.Collections.Generic;

using System.Linq;

using System.Threading.Tasks;

using BookingProject.Data.Models;

public class CategoriesSeeder : ISeeder

{

public async Task SeedAsync(ApplicationDbContext dbContext, IServiceProvider serviceProvider)

{

if (dbContext.Categories.Any())

{

return;

}

var categories = new List<string> { "Къщи", "Апартаменти", "Стаи", "Бунгала" };

foreach (var category in categories)

{

await dbContext.Categories.AddAsync(new Category

{

Name = category,

});

}

}

}

}

CitiesSeeder.cs

namespace BookingProject.Data.Seeding

{

using System;

using System.Collections.Generic;

using System.IO;

using System.Linq;

using System.Threading.Tasks;

using BookingProject.Data.Models;

using Microsoft.EntityFrameworkCore;

using Newtonsoft.Json;

public class CitiesSeeder : ISeeder

{

public async Task SeedAsync(ApplicationDbContext dbContext, IServiceProvider serviceProvider)

{

if (dbContext.Cities.Any() || dbContext.Regions.Any())

{

return;

}

// read file into a string and deserialize JSON to a type

var cities = JsonConvert.DeserializeObject<List<CityForSeed>>(File.ReadAllText(@"C:\Users\Georgi Arihtev\source\repos\Denitsa\BookingProject-D\Data\BookingProject.Data\Seeding\cities.json"));

Console.WriteLine(cities);

foreach (var city in cities)

{

Region reg;

if (!await dbContext.Regions.AnyAsync(r => r.Name == city.Region))

{

reg = new Region

{

Name = city.Region,

};

await dbContext.Regions.AddAsync(reg);

await dbContext.SaveChangesAsync();

}

else

{

reg = await dbContext.Regions.Where(r => r.Name == city.Region).FirstOrDefaultAsync();

}

await dbContext.Cities.AddAsync(new City

{

Name = city.Name,

RegionId = reg.Id,

});

}

}

}

}

ExtrasSeeder.cs

namespace BookingProject.Data.Seeding

{

using System;

using System.Collections.Generic;

using System.Linq;

using System.Threading.Tasks;

using BookingProject.Data.Models;

public class ExtrasSeeder : ISeeder

{

public async Task SeedAsync(ApplicationDbContext dbContext, IServiceProvider serviceProvider)

{

if (dbContext.Extras.Any())

{

return;

}

var extras = new List<string> { "Паркинг", "Климатик", "ТВ", "Безплатен интернет", "Тераса", "Веранда", "Басейн", "Кухненски бокс", "Кафемашина", "Хладилник", "Пералня", "Бебешка кошара" };

foreach (var extra in extras)

{

await dbContext.Extras.AddAsync(new Extra

{

Name = extra,

});

}

}

}

}

ApplicationDbContext.cs

namespace BookingProject.Data

{

using System;

using System.Linq;

using System.Reflection;

using System.Threading;

using System.Threading.Tasks;

using BookingProject.Data.Common.Models;

using BookingProject.Data.Models;

using Microsoft.AspNetCore.Identity.EntityFrameworkCore;

using Microsoft.EntityFrameworkCore;

public class ApplicationDbContext : IdentityDbContext<ApplicationUser, ApplicationRole, string>

{

private static readonly MethodInfo SetIsDeletedQueryFilterMethod =

typeof(ApplicationDbContext).GetMethod(

nameof(SetIsDeletedQueryFilter),

BindingFlags.NonPublic | BindingFlags.Static);

public ApplicationDbContext(DbContextOptions<ApplicationDbContext> options)

: base(options)

{

}

public DbSet<Setting> Settings { get; set; }

public DbSet<Place> Places { get; set; }

public DbSet<Review> Reviews { get; set; }

public DbSet<Image> Images { get; set; }

public DbSet<Category> Categories { get; set; }

public DbSet<Reservation> Reservations { get; set; }

public DbSet<City> Cities { get; set; }

public DbSet<Region> Regions { get; set; }

public DbSet<Extra> Extras { get; set; }

public DbSet<PlaceExtra> PlaceExtras { get; set; }

public override int SaveChanges() => this.SaveChanges(true);

public override int SaveChanges(bool acceptAllChangesOnSuccess)

{

this.ApplyAuditInfoRules();

return base.SaveChanges(acceptAllChangesOnSuccess);

}

public override Task<int> SaveChangesAsync(CancellationToken cancellationToken = default) =>

this.SaveChangesAsync(true, cancellationToken);

public override Task<int> SaveChangesAsync(

bool acceptAllChangesOnSuccess,

CancellationToken cancellationToken = default)

{

this.ApplyAuditInfoRules();

return base.SaveChangesAsync(acceptAllChangesOnSuccess, cancellationToken);

}

protected override void OnModelCreating(ModelBuilder builder)

{

// Needed for Identity models configuration

base.OnModelCreating(builder);

this.ConfigureUserIdentityRelations(builder);

EntityIndexesConfiguration.Configure(builder);

var entityTypes = builder.Model.GetEntityTypes().ToList();

// Set global query filter for not deleted entities only

var deletableEntityTypes = entityTypes

.Where(et => et.ClrType != null && typeof(IDeletableEntity).IsAssignableFrom(et.ClrType));

foreach (var deletableEntityType in deletableEntityTypes)

{

var method = SetIsDeletedQueryFilterMethod.MakeGenericMethod(deletableEntityType.ClrType);

method.Invoke(null, new object[] { builder });

}

// Disable cascade delete

var foreignKeys = entityTypes

.SelectMany(e => e.GetForeignKeys().Where(f => f.DeleteBehavior == DeleteBehavior.Cascade));

foreach (var foreignKey in foreignKeys)

{

foreignKey.DeleteBehavior = DeleteBehavior.Restrict;

}

}

private static void SetIsDeletedQueryFilter<T>(ModelBuilder builder)

where T : class, IDeletableEntity

{

builder.Entity<T>().HasQueryFilter(e => !e.IsDeleted);

}

// Applies configurations

private void ConfigureUserIdentityRelations(ModelBuilder builder)

=> builder.ApplyConfigurationsFromAssembly(this.GetType().Assembly);

private void ApplyAuditInfoRules()

{

var changedEntries = this.ChangeTracker

.Entries()

.Where(e =>

e.Entity is IAuditInfo &&

(e.State == EntityState.Added || e.State == EntityState.Modified));

foreach (var entry in changedEntries)

{

var entity = (IAuditInfo)entry.Entity;

if (entry.State == EntityState.Added && entity.CreatedOn == default)

{

entity.CreatedOn = DateTime.UtcNow;

}

else

{

entity.ModifiedOn = DateTime.UtcNow;

}

}

}

}

}

Models

ApplicationUser.cs

// ReSharper disable VirtualMemberCallInConstructor

namespace BookingProject.Data.Models

{

using System;

using System.Collections.Generic;

using BookingProject.Data.Common.Models;

using Microsoft.AspNetCore.Identity;

public class ApplicationUser : IdentityUser, IAuditInfo, IDeletableEntity

{

public ApplicationUser()

{

this.Id = Guid.NewGuid().ToString();

this.Roles = new HashSet<IdentityUserRole<string>>();

this.Claims = new HashSet<IdentityUserClaim<string>>();

this.Logins = new HashSet<IdentityUserLogin<string>>();

this.Reviews = new HashSet<Review>();

this.Places = new HashSet<Place>();

}

// Audit info

public DateTime CreatedOn { get; set; }

public DateTime? ModifiedOn { get; set; }

// Deletable entity

public bool IsDeleted { get; set; }

public DateTime? DeletedOn { get; set; }

public bool IsAdmin { get; set; } = false;

public virtual ICollection<IdentityUserRole<string>> Roles { get; set; }

public virtual ICollection<IdentityUserClaim<string>> Claims { get; set; }

public virtual ICollection<IdentityUserLogin<string>> Logins { get; set; }

public virtual ICollection<Place> Places { get; set; }

public virtual ICollection<Review> Reviews { get; set; }

}

}

Category.cs

namespace BookingProject.Data.Models

{

using System.Collections.Generic;

using System.ComponentModel.DataAnnotations;

using BookingProject.Data.Common.Models;

public class Category : BaseDeletableModel<int>

{

public Category()

{

this.Places = new HashSet<Place>();

}

public new int Id { get; set; }

[StringLength(50)]

[Required]

public string Name { get; set; }

public string ImageName { get; set; }

public string ImageUrl { get; set; }

public virtual ICollection<Place> Places { get; set; }

}

}

City.cs

namespace BookingProject.Data.Models

{

using BookingProject.Data.Common.Models;

using System.ComponentModel.DataAnnotations;

public class City : BaseDeletableModel<int>

{

public new int Id { get; set; }

[StringLength(50)]

[Required]

public string Name { get; set; }

public int RegionId { get; set; }

public virtual Region Region { get; set; }

}

}

Extra.cs

namespace BookingProject.Data.Models

{

using BookingProject.Data.Common.Models;

using System.Collections.Generic;

using System.ComponentModel.DataAnnotations;

public class Extra : BaseDeletableModel<int>

{

public new int Id { get; set; }

[StringLength(50)]

[Required]

public string Name { get; set; }

public List<PlaceExtra> PlaceExtras { get; set; }

}

}

Image.cs

namespace BookingProject.Data.Models

{

using System;

using System.Collections.Generic;

using System.ComponentModel.DataAnnotations;

using System.Diagnostics.CodeAnalysis;

using System.Text;

using BookingProject.Data.Common.Models;

public class Image : BaseDeletableModel<int>

{

public new int Id { get; set; }

[StringLength(50)]

[Required]

public string Name { get; set; }

[StringLength(50)]

[Required]

public string Ext { get; set; }

[StringLength(50)]

[Required]

public string Path { get; set; }

[Required]

public int PlaceId { get; set; }

public virtual Place Place { get; set; }

}

}

Place.cs

namespace BookingProject.Data.Models

{

using System;

using System.Collections.Generic;

using System.ComponentModel.DataAnnotations;

using System.Diagnostics.CodeAnalysis;

using System.Linq;

using BookingProject.Data.Common.Models;

public class Place : BaseDeletableModel<int>

{

public Place()

{

this.Reviews = new HashSet<Review>();

this.Images = new HashSet<Image>();

this.PlaceExtras = new HashSet<PlaceExtra>();

this.Reservations = new HashSet<Reservation>();

}

public new int Id { get; set; }

[Required(AllowEmptyStrings = true)]

public string Name { get; set; }

[Required]

public string UserId { get; set; }

public virtual ApplicationUser User { get; set; }

[Required]

public int CategoryId { get; set; }

public virtual Category Category { get; set; }

[Required]

public int CityId { get; set; }

public virtual City City { get; set; }

[StringLength(500)]

[Required]

public string Address { get; set; }

public string Description { get; set; }

[Required]

public int PriceByNight { get; set; }

[Required]

public int BedroomsNum { get; set; }

[Required]

public int BathroomsNum { get; set; }

[Required]

public int BedsNum { get; set; }

[Required]

public int MaxGuest { get; set; }

public bool Pets { get; set; }

public bool Smoking { get; set; }

public virtual ICollection<PlaceExtra> PlaceExtras { get; set; }

[AllowNull]

public virtual ICollection<Image> Images { get; set; }

[AllowNull]

public virtual ICollection<Review> Reviews { get; set; }

[AllowNull]

public virtual ICollection<Reservation> Reservations { get; set; }

public virtual int Rating { get; set; }

}

}

PlaceExtra.cs

using BookingProject.Data.Common.Models;

using System;

using System.Collections.Generic;

using System.ComponentModel.DataAnnotations;

using System.Text;

namespace BookingProject.Data.Models

{

public class PlaceExtra : BaseDeletableModel<int>

{

[Required]

public int PlaceId { get; set; }

public virtual Place Place { get; set; }

[Required]

public int ExtraId { get; set; }

public virtual Extra Extra { get; set; }

}

}

Region.cs

namespace BookingProject.Data.Models

{

using BookingProject.Data.Common.Models;

using System.ComponentModel.DataAnnotations;

public class Region : BaseDeletableModel<int>

{

public new int Id { get; set; }

[Required]

public string Name { get; set; }

}

}

Reservation.cs

namespace BookingProject.Data.Models

{

using System;

using System.Collections.Generic;

using System.ComponentModel.DataAnnotations;

using System.Text;

using BookingProject.Data.Common.Models;

public class Reservation : BaseDeletableModel<int>

{

public new int Id { get; set; }

[Required]

public string UserId { get; set; }

public virtual ApplicationUser User { get; set; }

[Required]

public int PlaceId { get; set; }

public virtual Place Place { get; set; }

[Required]

public DateTime StartDate { get; set; }

[Required]

public DateTime EndDate { get; set; }

[Required]

public double PricePerNight { get; set; }

[Required]

public double TotalPrice { get; set; }

[Required]

public int NumNights { get; set; }

public bool Active { get; set; }

public bool Reviewed { get; set; }

}

}

Review.cs

namespace BookingProject.Data.Models

{

using System;

using System.Collections.Generic;

using System.ComponentModel.DataAnnotations;

using System.Text;

using BookingProject.Data.Common.Models;

public class Review : BaseDeletableModel<int>

{

public new int Id { get; set; }

[Required]

public string UserId { get; set; }

public virtual ApplicationUser User { get; set; }

[Required]

public int PlaceId { get; set; }

public virtual Place Place { get; set; }

[Required]

public int Rating { get; set; }

[Required]

[StringLength(500)]

public string Comment { get; set; }

}

}

Services

CategoriesService.cs

namespace BookingProject.Services.Data

{

using System.Collections.Generic;

using System.Linq;

using BookingProject.Data.Common.Repositories;

using BookingProject.Data.Models;

using BookingProject.Services.Mapping;

public class CategoriesService : ICategoriesService

{

private readonly IDeletableEntityRepository<Category> categoriesRepository;

public CategoriesService(

IDeletableEntityRepository<Category> categoriesRepository)

{

this.categoriesRepository = categoriesRepository;

}

public IEnumerable<T> GetAll<T>(int? count = null)

{

IQueryable<Category> query =

this.categoriesRepository.All().OrderBy(x => x.Name);

if (count.HasValue)

{

query = query.Take(count.Value);

}

return query.To<T>().ToList();

}

public T GetByName<T>(string name)

{

var category = this.categoriesRepository.All()

.Where(x => x.Name.Replace(" ", "-") == name.Replace(" ", "-"))

.To<T>().FirstOrDefault();

return category;

}

}

}

CitiesService.cs

namespace BookingProject.Services.Data

{

using System.Collections.Generic;

using System.Linq;

using BookingProject.Data.Common.Repositories;

using BookingProject.Data.Models;

using BookingProject.Services.Mapping;

public class CitiesService : ICitiesService

{

private readonly IDeletableEntityRepository<City> citiesRepository;

public CitiesService(IDeletableEntityRepository<City> citiesRepository)

{

this.citiesRepository = citiesRepository;

}

public IEnumerable<T> GetAll<T>(int? count = null)

{

IQueryable<City> query =

this.citiesRepository.All().OrderBy(x => x.Name);

if (count.HasValue)

{

query = query.Take(count.Value);

}

return query.To<T>().ToList();

}

public T GetByName<T>(string name)

{

var city = this.citiesRepository.All().Where(x => x.Name == name)

.To<T>().FirstOrDefault();

return city;

}

}

}

ExtrasService.cs

namespace BookingProject.Services.Data

{

using System;

using System.Collections.Generic;

using System.Linq;

using BookingProject.Data.Common.Repositories;

using BookingProject.Data.Models;

using BookingProject.Services.Mapping;

public class ExtrasService : IExtrasService

{

private readonly IDeletableEntityRepository<Extra> extrasRepository;

public ExtrasService(IDeletableEntityRepository<Extra> extrasRepository)

{

this.extrasRepository = extrasRepository;

}

public IEnumerable<T> GetAll<T>(int? count = null)

{

IQueryable<Extra> query =

this.extrasRepository.All().OrderBy(x => x.Name);

if (count.HasValue)

{

query = query.Take(count.Value);

}

return query.To<T>().ToList();

}

public T GetById<T>(int id)

{

var extra = this.extrasRepository.All().Where(x => x.Id == id).To<T>().FirstOrDefault();

return extra;

}

}

}

**ImagesService.cs**

namespace BookingProject.Services.Data

{

using System;

using System.Collections.Generic;

using System.IO;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using BookingProject.Data.Common.Repositories;

using BookingProject.Data.Models;

using BookingProject.Services.Data.IServices;

using Microsoft.AspNetCore.Hosting;

using Microsoft.AspNetCore.Http;

public class ImagesService : IImagesService

{

private readonly IDeletableEntityRepository<Image> imagesRepository;

private readonly IHostingEnvironment ihost;

public ImagesService(IDeletableEntityRepository<Image> imagesRepository, IHostingEnvironment ihost)

{

this.imagesRepository = imagesRepository;

this.ihost = ihost;

}

public object ViewData { get; private set; }

public async Task<List<Image>> UploadImages(List<IFormFile> fileobj)

{

List<Image> images = new List<Image>();

foreach (IFormFile img in fileobj)

{

string imageExt = Path.GetExtension(img.FileName);

if (imageExt == ".jpg" || imageExt == ".png")

{

var imgPath = Path.Combine(this.ihost.WebRootPath, "images", img.FileName);

var stream = new FileStream(imgPath, FileMode.Create);

await img.CopyToAsync(stream);

Image image = new Image { Name = img.FileName, Ext = imageExt, Path = imgPath };

images.Add(image);

// await this.imagesRepository.AddAsync(image);

// await this.imagesRepository.SaveChangesAsync();

}

}

return images;

}

}

}

PlacesService.cs

namespace BookingProject.Services.Data

{

using System;

using System.Collections.Generic;

using System.Linq;

using System.Threading.Tasks;

using BookingProject.Data.Common.Repositories;

using BookingProject.Data.Models;

using BookingProject.Services.Data.IServices;

using BookingProject.Services.Mapping;

using BookingProject.Web.ViewModels.Places;

using Microsoft.AspNetCore.Http;

using Microsoft.EntityFrameworkCore;

public class PlacesService : IPlacesService

{

private readonly IDeletableEntityRepository<Place> placesRepository;

private readonly IDeletableEntityRepository<Reservation> reservationsRepository;

private readonly IDeletableEntityRepository<Image> imagesRepository;

private readonly IDeletableEntityRepository<PlaceExtra> placeExtrasRepository;

private readonly IDeletableEntityRepository<Review> reviewsRepository;

private readonly IExtrasService extrasService;

private readonly IImagesService imagesService;

public PlacesService(

IDeletableEntityRepository<Place> placesRepository,

IDeletableEntityRepository<Reservation> reservationsRepository,

IDeletableEntityRepository<Image> imagesRepository,

IDeletableEntityRepository<PlaceExtra> placeExtrasRepository,

IDeletableEntityRepository<Review> reviewsRepository,

IExtrasService extrasService,

IImagesService imagesService)

{

this.placesRepository = placesRepository;

this.reservationsRepository = reservationsRepository;

this.imagesRepository = imagesRepository;

this.placeExtrasRepository = placeExtrasRepository;

this.reviewsRepository = reviewsRepository;

this.extrasService = extrasService;

this.imagesService = imagesService;

}

public async Task<int> CreateAsync(

string name,

int categoryId,

int cityId,

string address,

string description,

string userId,

int priceByNight,

int bedroomsNum,

int bathroomsNum,

int bedsNum,

int maxGuest,

bool pets,

bool smoking,

List<ExtraViewModel> extras,

List<IFormFile> images)

{

var place = new Place

{

Name = name,

CategoryId = categoryId,

CityId = cityId,

Address = address,

Description = description,

UserId = userId,

PriceByNight = priceByNight,

BedroomsNum = bedroomsNum,

BathroomsNum = bathroomsNum,

BedsNum = bedsNum,

MaxGuest = maxGuest,

Pets = pets,

Smoking = smoking,

};

List<PlaceExtra> placesExtras = new List<PlaceExtra>();

foreach (var extra in extras)

{

if (extra.IsSelected)

{

PlaceExtra newPlaceExtra = new PlaceExtra { ExtraId = extra.Id, Place = place };

placesExtras.Add(newPlaceExtra);

}

}

place.PlaceExtras = placesExtras;

List<Image> placeImages = await this.imagesService.UploadImages(images);

place.Images = placeImages;

await this.placesRepository.AddAsync(place);

await this.placesRepository.SaveChangesAsync();

return place.Id;

}

public IEnumerable<T> GetAll<T>(int? count = null)

{

IQueryable<Place> query =

this.placesRepository.All().OrderBy(x => x.Name);

if (count.HasValue)

{

query = query.Take(count.Value);

}

return query.To<T>().ToList();

}

public IEnumerable<T> GetAllByUser<T>(string userId)

{

IQueryable<Place> query =

this.placesRepository.All().Where(p => p.UserId == userId).OrderBy(x => x.Name);

return query.To<T>().ToList();

}

public T GetById<T>(int id)

{

var place = this.placesRepository.All().Where(x => x.Id == id).Include(place => place.PlaceExtras).ThenInclude(pExtra => pExtra.Extra).Select(p => p).Include(p => p.City).ThenInclude(c => c.Region)

.To<T>().FirstOrDefault();

return place;

}

public T GetByName<T>(string name)

{

var place = this.placesRepository.All().Where(x => x.Name == name)

.To<T>().FirstOrDefault();

return place;

}

public IEnumerable<T> GetByCategoryId<T>(int categoryId, int? take = null, int skip = 0)

{

var query = this.placesRepository.All()

.OrderBy(x => x.Name)

.Where(x => x.CategoryId == categoryId);

if (take.HasValue)

{

query = query.Take(take.Value);

}

return query.To<T>().ToList();

}

public int GetCountByCategoryId(int categoryId)

{

return this.placesRepository.All().Count(x => x.CategoryId == categoryId);

}

public double GetRating(int placeId)

{

Place place = this.GetById<Place>(placeId);

return (int)Math.Round((double)place.Reviews.Select(x => x.Rating).Average());

}

public async Task DeleteById(int id)

{

var place = this.placesRepository.All().Include(p => p.Reservations).Include(p => p.Images).Include(p => p.PlaceExtras).Include(p => p.Reviews).FirstOrDefault(place => place.Id == id);

place.Reservations.ToList<Reservation>().ForEach(x => this.reservationsRepository.Delete(x));

place.PlaceExtras.ToList<PlaceExtra>().ForEach(x => this.placeExtrasRepository.Delete(x));

place.Images.ToList<Image>().ForEach(x => this.imagesRepository.Delete(x));

place.Reviews.ToList<Review>().ForEach(x => this.reviewsRepository.Delete(x));

this.placesRepository.Delete(place);

await this.placesRepository.SaveChangesAsync();

await this.reservationsRepository.SaveChangesAsync();

await this.imagesRepository.SaveChangesAsync();

await this.reviewsRepository.SaveChangesAsync();

}

}

}

RegionsService.cs

namespace BookingProject.Services.Data

{

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using BookingProject.Data.Common.Repositories;

using BookingProject.Data.Models;

using BookingProject.Services.Mapping;

public class RegionsService : IRegionsService

{

private readonly IDeletableEntityRepository<Region> regionsRepository;

public RegionsService(IDeletableEntityRepository<Region> regionsRepository)

{

this.regionsRepository = regionsRepository;

}

public IEnumerable<T> GetAll<T>(int? count = null)

{

IQueryable<Region> query =

this.regionsRepository.All().OrderBy(x => x.Name);

if (count.HasValue)

{

query = query.Take(count.Value);

}

return query.To<T>().ToList();

}

public T GetByName<T>(string name)

{

var region = this.regionsRepository.All().Where(x => x.Name == name)

.To<T>().FirstOrDefault();

return region;

}

public T GetById<T>(int id)

{

throw new System.NotImplementedException();

}

}

}

ReservationsService.cs

namespace BookingProject.Services.Data

{

using System;

using System.Collections.Generic;

using System.Globalization;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using BookingProject.Data.Common.Repositories;

using BookingProject.Data.Models;

using BookingProject.Services.Data.IServices;

using BookingProject.Services.Mapping;

public class ReservationsService : IReservationsService

{

private readonly IDeletableEntityRepository<Reservation> reservationsRepository;

private readonly IPlacesService placesService;

public ReservationsService(IDeletableEntityRepository<Reservation> reservationsRepository, IPlacesService placesService)

{

this.reservationsRepository = reservationsRepository;

this.placesService = placesService;

}

public IEnumerable<T> GetAll<T>()

{

IQueryable<Reservation> query =

this.reservationsRepository.All().OrderBy(r => r.Id);

return query.To<T>().ToList();

}

public IEnumerable<T> GetAllByUser<T>(string userId)

{

IQueryable<Reservation> query =

this.reservationsRepository.All().Where(r => r.UserId == userId).OrderBy(r => r.Id);

return query.To<T>().ToList();

}

public IEnumerable<T> GetAllByPlace<T>(int placeId)

{

IQueryable<Reservation> query =

this.reservationsRepository.All().Where(r => r.PlaceId == placeId).OrderBy(r => r.Id);

return query.To<T>().ToList();

}

public async Task<int> CreateReservationAsync(

int placeId,

string userId,

string dates,

double pricePerNight,

double totalPrice,

int numNights)

{

string[] splitDates = dates.Split(" - ");

var reservation = new Reservation

{

PlaceId = placeId,

UserId = userId,

StartDate = DateTime.Parse(splitDates[0]),

EndDate = DateTime.Parse(splitDates[1]),

PricePerNight = pricePerNight,

TotalPrice = totalPrice,

NumNights = numNights,

};

await this.reservationsRepository.AddAsync(reservation);

await this.reservationsRepository.SaveChangesAsync();

return reservation.Id;

}

}

}

ReviewsService.cs

namespace BookingProject.Services.Data

{

using System;

using System.Collections.Generic;

using System.Linq;

using System.Threading.Tasks;

using BookingProject.Data.Common.Repositories;

using BookingProject.Data.Models;

using BookingProject.Services.Mapping;

public class ReviewsService : IReviewsService

{

private readonly IDeletableEntityRepository<Review> reviewsRepository;

public ReviewsService(IDeletableEntityRepository<Review> reviewsRepository)

{

this.reviewsRepository = reviewsRepository;

}

public async Task Create(int placeId, string userId, string comment, int rating)

{

var review = new Review

{

PlaceId = placeId,

UserId = userId,

Comment = comment,

Rating = rating,

};

await this.reviewsRepository.AddAsync(review);

await this.reviewsRepository.SaveChangesAsync();

}

public bool IsInPlaceId(int reviewId, int placeId)

{

var reviewPlaceId = this.reviewsRepository.All().Where(x => x.Id == reviewId)

.Select(x => x.PlaceId).FirstOrDefault();

return reviewPlaceId == placeId;

}

}

}

**namespace BookingProject.Services.Mapping**

{

using System;

using System.Collections.Generic;

using System.Linq;

using System.Reflection;

using AutoMapper;

using AutoMapper.Configuration;

public static class AutoMapperConfig

{

private static bool initialized;

public static IMapper MapperInstance { get; set; }

public static void RegisterMappings(params Assembly[] assemblies)

{

if (initialized)

{

return;

}

initialized = true;

var types = assemblies.SelectMany(a => a.GetExportedTypes()).ToList();

var config = new MapperConfigurationExpression();

config.CreateProfile(

"ReflectionProfile",

configuration =>

{

// IMapFrom<>

foreach (var map in GetFromMaps(types))

{

configuration.CreateMap(map.Source, map.Destination);

}

// IMapTo<>

foreach (var map in GetToMaps(types))

{

configuration.CreateMap(map.Source, map.Destination);

}

// IHaveCustomMappings

foreach (var map in GetCustomMappings(types))

{

map.CreateMappings(configuration);

}

});

MapperInstance = new Mapper(new MapperConfiguration(config));

}

private static IEnumerable<TypesMap> GetFromMaps(IEnumerable<Type> types)

{

var fromMaps = from t in types

from i in t.GetTypeInfo().GetInterfaces()

where i.GetTypeInfo().IsGenericType &&

i.GetGenericTypeDefinition() == typeof(IMapFrom<>) &&

!t.GetTypeInfo().IsAbstract &&

!t.GetTypeInfo().IsInterface

select new TypesMap

{

Source = i.GetTypeInfo().GetGenericArguments()[0],

Destination = t,

};

return fromMaps;

}

private static IEnumerable<TypesMap> GetToMaps(IEnumerable<Type> types)

{

var toMaps = from t in types

from i in t.GetTypeInfo().GetInterfaces()

where i.GetTypeInfo().IsGenericType &&

i.GetTypeInfo().GetGenericTypeDefinition() == typeof(IMapTo<>) &&

!t.GetTypeInfo().IsAbstract &&

!t.GetTypeInfo().IsInterface

select new TypesMap

{

Source = t,

Destination = i.GetTypeInfo().GetGenericArguments()[0],

};

return toMaps;

}

private static IEnumerable<IHaveCustomMappings> GetCustomMappings(IEnumerable<Type> types)

{

var customMaps = from t in types

from i in t.GetTypeInfo().GetInterfaces()

where typeof(IHaveCustomMappings).GetTypeInfo().IsAssignableFrom(t) &&

!t.GetTypeInfo().IsAbstract &&

!t.GetTypeInfo().IsInterface

select (IHaveCustomMappings)Activator.CreateInstance(t);

return customMaps;

}

private class TypesMap

{

public Type Source { get; set; }

public Type Destination { get; set; }

}

}

}

Controllers

HomeController.cs

namespace BookingProject.Web.Controllers

{

using System.Collections;

using System.Collections.Generic;

using System.Diagnostics;

using System.Linq;

using BookingProject.Data.Common.Repositories;

using BookingProject.Data.Models;

using BookingProject.Services.Data;

using BookingProject.Services.Mapping;

using BookingProject.Web.ViewModels;

using BookingProject.Web.ViewModels.Home;

using BookingProject.Web.ViewModels.Places;

using BookingProject.Web.ViewModels.Reservations;

using Microsoft.AspNetCore.Mvc;

public class HomeController : BaseController

{

private readonly ICategoriesService categoriesService;

private readonly ICitiesService citiesService;

public HomeController(ICategoriesService categoriesService, ICitiesService citiesService)

{

this.categoriesService = categoriesService;

this.citiesService = citiesService;

}

public IActionResult Index()

{

var cities = this.citiesService.GetAll<CityDropDownViewModel>();

var numbers = Enumerable.Range(1, 30).ToList();

IEnumerable<GuestNumberDropDownViewModel> guestNumbers = numbers.Select(x => new GuestNumberDropDownViewModel { Id = x, Name = x.ToString() });

var viewModel = new IndexViewModel

{

Categories =

this.categoriesService.GetAll<IndexCategoryViewModel>(),

Cities = cities,

GuestNumbers = guestNumbers,

};

return this.View(viewModel);

}

public IActionResult Privacy()

{

return this.View();

}

[ResponseCache(Duration = 0, Location = ResponseCacheLocation.None, NoStore = true)]

public IActionResult Error()

{

return this.View(

new ErrorViewModel { RequestId = Activity.Current?.Id ?? this.HttpContext.TraceIdentifier });

}

public IActionResult About()

{

return this.View();

}

}

}

CategoriesController.cs

namespace BookingProject.Web.Controllers

{

using System;

using System.Collections.Generic;

using System.Linq;

using System.Threading.Tasks;

using BookingProject.Services.Data;

using BookingProject.Web.ViewModels.Places;

using BookingProject.Services.Data;

using BookingProject.Web.ViewModels.Categories;

using Microsoft.AspNetCore.Http;

using Microsoft.AspNetCore.Mvc;

using Microsoft.Extensions.Logging;

public class CategoriesController : Controller

{

private const int PlacesPerPage = 6;

private readonly ICategoriesService categoriesService;

private readonly IPlacesService placesService;

private readonly IHttpContextAccessor http;

public CategoriesController(

ICategoriesService categoriesService,

IPlacesService placesService,

IHttpContextAccessor http)

{

this.categoriesService = categoriesService;

this.placesService = placesService;

this.http = http;

}

public IActionResult ByName(string name, int page = 1)

{

var viewModel =

this.categoriesService.GetByName<CategoryViewModel>(name);

if (viewModel == null)

{

return this.NotFound();

}

viewModel.CardPlaces = this.placesService.GetByCategoryId<CardPlaceViewModel>(viewModel.Id, PlacesPerPage, (page - 1) \* PlacesPerPage);

var count = this.placesService.GetCountByCategoryId(viewModel.Id);

viewModel.PagesCount = (int)Math.Ceiling((double)count / PlacesPerPage);

if (viewModel.PagesCount == 0)

{

viewModel.PagesCount = 1;

}

viewModel.CurrentPage = page;

return this.View(viewModel);

}

}

}

PlacesController.cs

namespace BookingProject.Web.Controllers

{

using System;

using System.Collections.Generic;

using System.Globalization;

using System.Linq;

using System.Threading.Tasks;

using BookingProject.Data.Models;

using BookingProject.Services.Data;

using BookingProject.Web.ViewModels.Home;

using BookingProject.Web.ViewModels.Places;

using BookingProject.Web.ViewModels.Reservations;

using Microsoft.AspNetCore.Authorization;

using Microsoft.AspNetCore.Identity;

using Microsoft.AspNetCore.Mvc;

using Newtonsoft.Json;

public class PlacesController : Controller

{

private const int PlacesPerPage = 8;

private readonly IPlacesService placesService;

private readonly ICategoriesService categoriesService;

private readonly ICitiesService citiesService;

private readonly IExtrasService extrasService;

private readonly UserManager<ApplicationUser> userManager;

public PlacesController(

IPlacesService placesService,

ICategoriesService categoriesService,

ICitiesService citiesService,

IExtrasService extrasService,

UserManager<ApplicationUser> userManager)

{

this.placesService = placesService;

this.categoriesService = categoriesService;

this.citiesService = citiesService;

this.extrasService = extrasService;

this.userManager = userManager;

}

public async Task<IActionResult> ShowUserPlaces()

{

var user = await this.userManager.GetUserAsync(this.User);

IEnumerable<PlaceViewModel> places = this.placesService.GetAllByUser<PlaceViewModel>(user.Id).ToList();

var viewModel = new ListPlaceViewModel

{

Places = places,

};

if (viewModel == null)

{

return this.NotFound();

}

foreach (var place in viewModel.Places)

{

if (place.Reviews.Count > 0)

{

place.Rating = (int)place.Reviews.Select(x => x.Rating).Average();

}

else

{

place.Rating = 0;

}

}

return this.View(viewModel);

}

public async Task<IActionResult> Administration()

{

var user = await this.userManager.GetUserAsync(this.User);

if (!user.IsAdmin)

{

return this.NotFound();

}

IEnumerable<PlaceViewModel> places = this.placesService.GetAll<PlaceViewModel>().ToList();

var viewModel = new ListPlaceViewModel

{

Places = places,

};

if (viewModel == null)

{

return this.NotFound();

}

foreach (var place in viewModel.Places)

{

if (place.Reviews.Count > 0)

{

place.Rating = (int)place.Reviews.Select(x => x.Rating).Average();

}

else

{

place.Rating = 0;

}

}

return this.View(viewModel);

}

public IActionResult ShowAllPlaces(int? cityId, int? guestNumber, string? dates, string? sortBy, string? categoryName)

{

var cities = this.citiesService.GetAll<CityDropDownViewModel>();

var numbers = Enumerable.Range(1, 30).ToList();

IEnumerable<GuestNumberDropDownViewModel> guestNumbers = numbers.Select(x => new GuestNumberDropDownViewModel { Id = x, Name = x.ToString() });

IEnumerable<PlaceViewModel> places = this.placesService.GetAll<PlaceViewModel>().ToList();

if (dates != null)

{

string[] splitDates = dates.Split(" - ");

DateTime startDate = DateTime.Parse(splitDates[0]);

DateTime endDate = DateTime.Parse(splitDates[1]);

places = places.Where(p => this.CheckDates(startDate, endDate, p.Reservations.OrderBy(r => r.StartDate).ToList()));

}

if (categoryName != null)

{

places = places.Where(x => x.CategoryName == categoryName);

}

if (cityId != null && cityId != 0)

{

places = places.Where(x => x.CityId == cityId);

}

if (guestNumber != null && guestNumber != 0)

{

places = places.Where(x => x.MaxGuest >= guestNumber);

}

if (sortBy != null && sortBy != "id")

{

if (sortBy == "desc")

{

places = places.OrderBy(p => p.PriceByNight);

} else

{

places = places.OrderByDescending(p => p.PriceByNight);

}

}

var viewModel = new ListPlaceViewModel

{

Places = places,

Categories =

this.categoriesService.GetAll<IndexCategoryViewModel>(),

Cities = cities,

GuestNumbers = guestNumbers,

};

if (viewModel == null)

{

return this.NotFound();

}

foreach (var place in viewModel.Places)

{

if (place.Reviews.Count > 0)

{

place.Rating = (int)place.Reviews.Select(x => x.Rating).Average();

}

else

{

place.Rating = 0;

}

}

//viewModel.CardsPlaces = this.placesService.GetByCategoryId<CardPlaceViewModel>(viewModel.Id, PlacesPerPage, (page - 1) \* PlacesPerPage);

//var count = this.placesService.GetCountByCategoryId(viewModel.Id);

//viewModel.PagesCount = (int)Math.Ceiling((double)count / PlacesPerPage);

//if (viewModel.PagesCount == 0)

//{

// viewModel.PagesCount = 1;

//}

//viewModel.CurrentPage = page;

return this.View(viewModel);

}

public IActionResult GetById(int id)

{

var placeViewModel = this.placesService.GetById<PlaceViewModel>(id);

if (placeViewModel == null)

{

return this.NotFound();

}

if (placeViewModel.Reviews.Count > 0)

{

placeViewModel.Rating = (int)placeViewModel.Reviews.Select(x => x.Rating).Average();

List<string> reservedDays = new List<string>();

DateTime now = DateTime.Now;

foreach (var res in placeViewModel.Reservations)

{

for (DateTime date = res.StartDate; date <= res.EndDate; date = date.AddDays(1))

{

reservedDays.Add(date.ToString("dd/MM/yyyy", CultureInfo.InvariantCulture));

}

}

placeViewModel.ReservedDays = JsonConvert.SerializeObject(reservedDays);

}

else

{

placeViewModel.Rating = 0;

placeViewModel.ReservedDays = JsonConvert.SerializeObject(new List<string>());

}

return this.View(placeViewModel);

}

public IActionResult Create()

{

var categories = this.categoriesService.GetAll<CategoryDropDownViewModel>();

var cities = this.citiesService.GetAll<CityDropDownViewModel>();

var extras = this.extrasService.GetAll<ExtraViewModel>();

var viewModel = new CreatePlaceInputModel

{

Categories = categories,

Cities = cities,

Extras = extras.ToList(),

};

return this.View(viewModel);

}

[HttpPost]

[Authorize]

public async Task<IActionResult> Create(CreatePlaceInputModel input)

{

var user = await this.userManager.GetUserAsync(this.User);

if (!this.ModelState.IsValid)

{

return this.View(input);

}

var placeId = await this.placesService.CreateAsync(

input.Name,

input.CategoryId,

input.CityId,

input.Address,

input.Description,

user.Id,

input.PriceByNight,

input.BedroomsNum,

input.BathroomsNum,

input.BedsNum,

input.MaxGuest,

input.Pets,

input.Smoking,

input.Extras,

input.Images);

this.TempData["InfoMessage"] = "Place created!";

return this.RedirectToAction(nameof(this.GetById), new { id = placeId });

}

[HttpGet]

public async Task<IActionResult> Edit(int id)

{

var user = await this.userManager.GetUserAsync(this.User);

var place = this.placesService.GetById<EditPlaceViewModel>(id);

if (place == null || place.User.Id != user.Id)

{

return this.NotFound();

}

var categories = this.categoriesService.GetAll<CategoryDropDownViewModel>();

var cities = this.citiesService.GetAll<CityDropDownViewModel>();

var extras = this.extrasService.GetAll<ExtraViewModel>();

var viewModel = new EditPlaceViewModel

{

Name = place.Name,

CategoryId = place.CategoryId,

CityId = place.CityId,

Address = place.Address,

Description = place.Description,

PriceByNight = place.PriceByNight,

BedroomsNum = place.BedroomsNum,

BathroomsNum = place.BathroomsNum,

MaxGuest = place.MaxGuest,

BedsNum = place.BedsNum,

User = place.User,

Pets = place.Pets,

Smoking = place.Smoking,

Categories = categories,

Cities = cities,

Extras = extras.ToList(),

};

return this.View(viewModel);

}

public async Task<IActionResult> Delete(int id)

{

var user = await this.userManager.GetUserAsync(this.User);

var place = this.placesService.GetById<EditPlaceViewModel>(id);

if (!user.IsAdmin)

{

return this.Forbid();

}

if (place == null)

{

return this.NotFound();

}

await this.placesService.DeleteById(id);

return this.RedirectToAction(nameof(this.Administration));

}

private bool CheckDates(DateTime start, DateTime end, List<Reservation> reservations)

{

if (reservations.Count == 0)

{

return true;

}

if (end < reservations.First().StartDate || start > reservations.Last().EndDate)

{

return true;

}

for (int i = 0; i < reservations.Count; i++)

{

DateTime resEnd = reservations[i].EndDate;

try

{

var nextRes = reservations[i + 1];

if (resEnd < start && end < nextRes.StartDate)

{

return true;

}

}

catch (Exception)

{

return false;

}

}

return false;

}

}

}

ReservationsController.cs

namespace BookingProject.Web.Controllers

{

using System;

using System.Linq;

using System.Threading.Tasks;

using BookingProject.Data.Models;

using BookingProject.Services.Data;

using BookingProject.Services.Data.IServices;

using BookingProject.Web.ViewModels.Places;

using BookingProject.Web.ViewModels.Reservations;

using Microsoft.AspNetCore.Authorization;

using Microsoft.AspNetCore.Identity;

using Microsoft.AspNetCore.Mvc;

public class ReservationsController : BaseController

{

private readonly IPlacesService placesService;

private readonly IReservationsService reservationsService;

private readonly UserManager<ApplicationUser> userManager;

public ReservationsController(IReservationsService reservationsService, IPlacesService placesService, UserManager<ApplicationUser> userManager)

{

this.placesService = placesService;

this.reservationsService = reservationsService;

this.userManager = userManager;

}

[HttpGet]

[Authorize]

public async Task<IActionResult> DetailReservation(string dates, int placeId)

{

var user = await this.userManager.GetUserAsync(this.User);

if (!this.ModelState.IsValid)

{

return this.View(dates);

}

var place = this.placesService.GetById<PlaceViewModel>(placeId);

string[] splitDates = dates.Split(" - ");

DateTime startDate = DateTime.Parse(splitDates[0]);

DateTime endDate = DateTime.Parse(splitDates[1]);

int numNights = (int)(endDate - startDate).TotalDays;

double totalPrice = Math.Round(double.Parse(place.PriceByNight) \* numNights);

var viewModel = new CreateReservationViewModel

{

PlaceId = placeId,

Dates = dates,

Place = place,

StartDate = startDate,

EndDate = endDate,

TotalPrice = totalPrice,

NumNights = numNights,

};

return this.View(viewModel);

}

[HttpPost]

public async Task<IActionResult> CreateReservation(CreateReservationViewModel input)

{

var user = await this.userManager.GetUserAsync(this.User);

if (!this.ModelState.IsValid)

{

return this.View(input);

}

var reservationId = await this.reservationsService.CreateReservationAsync(

input.PlaceId,

user.Id,

input.Dates,

input.PriceByNight,

input.TotalPrice,

input.NumNights);

this.TempData["InfoMessage"] = "Reservation created!";

return this.RedirectToAction(nameof(this.UserReservations));

}

public async Task<IActionResult> UserReservations()

{

var user = await this.userManager.GetUserAsync(this.User);

var viewModel = new ListReservationViewModel

{

Reservations =

this.reservationsService.GetAllByUser<ReservationViewModel>(user.Id).ToList(),

};

if (viewModel == null)

{

return this.NotFound();

}

return this.View(viewModel);

}

public async Task<IActionResult> PlaceReservations(int id)

{

var user = await this.userManager.GetUserAsync(this.User);

var place = this.placesService.GetById<PlaceViewModel>(id);

if (place == null || place.User.Id != user.Id)

{

return this.NotFound();

}

var viewModel = new ListReservationViewModel

{

Place = place,

Reservations =

this.reservationsService.GetAllByPlace<ReservationViewModel>(id).ToList(),

};

if (viewModel == null)

{

return this.NotFound();

}

return this.View(viewModel);

}

}

}

ReviewsController.cs

namespace BookingProject.Web.Controllers

{

using System;

using System.Collections.Generic;

using System.Linq;

using System.Threading.Tasks;

using BookingProject.Data.Models;

using BookingProject.Services.Data;

using BookingProject.Web.ViewModels.Reviews;

using Microsoft.AspNetCore.Authorization;

using Microsoft.AspNetCore.Identity;

using Microsoft.AspNetCore.Mvc;

public class ReviewsController : BaseController

{

private readonly IReviewsService reviewsService;

private readonly UserManager<ApplicationUser> userManager;

public ReviewsController(

IReviewsService reviewsService,

UserManager<ApplicationUser> userManager)

{

this.reviewsService = reviewsService;

this.userManager = userManager;

}

[HttpPost]

[Authorize]

public async Task<IActionResult> Create(CreateReviewInputModel input)

{

var userId = this.userManager.GetUserId(this.User);

await this.reviewsService.Create(input.PlaceId, userId, input.Comment, input.Rating);

return this.RedirectToAction("GetById", "Places", new { id = input.PlaceId });

}

}

}

ViewModels

CategoryViewModel.cs

namespace BookingProject.Web.ViewModels.Categories

{

using BookingProject.Data.Models;

using BookingProject.Services.Mapping;

using BookingProject.Web.ViewModels.Places;

using System.Collections.Generic;

public class CategoryViewModel : IMapFrom<Category>

{

public int Id { get; set; }

public string Name { get; set; }

public string ImageName { get; set; }

public string ImageUrl { get; set; }

public int CurrentPage { get; set; }

public int PagesCount { get; set; }

// public IEnumerable<PlaceViewModel> Places { get; set; }

public IEnumerable<CardPlaceViewModel> CardPlaces { get; set; }

}

}

* Home

IndexCategoryViewModel.cs

namespace BookingProject.Web.ViewModels.Home

{

using AutoMapper;

using BookingProject.Data.Models;

using BookingProject.Services.Mapping;

public class IndexCategoryViewModel : IMapFrom<Category>

{

public int Id { get; set; }

public string Name { get; set; }

public string ImageName { get; set; }

public string ImageUrl { get; set; }

public string Url => $"/b/{this.Name.Replace(' ', '-')}";

}

}

IndexViewModel.cs

namespace BookingProject.Web.ViewModels.Home

{

using System.Collections.Generic;

using System.ComponentModel.DataAnnotations;

using BookingProject.Web.ViewModels.Places;

using BookingProject.Web.ViewModels.Reservations;

public class IndexViewModel

{

public IEnumerable<IndexCategoryViewModel> Categories { get; set; }

public IEnumerable<CityDropDownViewModel> Cities { get; set; }

public IEnumerable<GuestNumberDropDownViewModel> GuestNumbers { get; set; }

public int CityId { get; set; }

public int GuestNumber { get; set; }

public string Dates { get; set; }

}

}

* Images

ImageViewModel.cs

namespace BookingProject.Web.ViewModels.Image

{

using System.Collections.Generic;

using System.Text;

using BookingProject.Data.Models;

using BookingProject.Services.Mapping;

public class ImageViewModel : IMapFrom<Image>

{

public int Id { get; set; }

public string Name { get; set; }

public string Path { get; set; }

public string Ext { get; set; }

}

}

* Places

PlaceViewModel.cs

namespace BookingProject.Web.ViewModels.Places

{

using System;

using System.Collections.Generic;

using System.ComponentModel.DataAnnotations;

using System.Net;

using System.Text.RegularExpressions;

using BookingProject.Data.Models;

using BookingProject.Services.Mapping;

public class PlaceViewModel : IMapFrom<Place>, IMapTo<Place>

{

public int Id { get; set; }

public string Name { get; set; }

public string CategoryName { get; set; }

public ApplicationUser User { get; set; }

public virtual City City { get; set; }

public int CityId { get; set; }

public string CityName { get; set; }

public string Address { get; set; }

public string Description { get; set; }

public string PriceByNight { get; set; }

public int BedroomsNum { get; set; }

public int BathroomsNum { get; set; }

public int BedsNum { get; set; }

public int MaxGuest { get; set; }

public bool Pets { get; set; }

public bool Smoking { get; set; }

public string ReviewsCount { get; set; }

public List<PlaceExtra> PlaceExtras { get; set; }

public List<Image> Images { get; set; }

public List<Reservation> Reservations { get; set; }

// Reservation data

public int PlaceId { get; set; }

[Required]

public string Dates { get; set; }

// Review data

public virtual int Rating { get; set; }

public List<PlaceReviewViewModel> Reviews { get; set; }

public string ReservedDays { get; set; }

}

}

CategoryDropDownViewModel.cs

using BookingProject.Data.Models;

using BookingProject.Services.Mapping;

namespace BookingProject.Web.ViewModels.Places

{

public class CategoryDropDownViewModel : IMapFrom<Category>

{

public int Id { get; set; }

public string Name { get; set; }

}

}

CityDropDownViewModel.cs

namespace BookingProject.Web.ViewModels.Places

{

using BookingProject.Data.Models;

using BookingProject.Services.Mapping;

public class CityDropDownViewModel : IMapFrom<City>

{

public int Id { get; set; }

public string Name { get; set; }

}

}

CreatePlaceInputModel.cs

namespace BookingProject.Web.ViewModels.Places

{

using System;

using System.Collections.Generic;

using System.ComponentModel.DataAnnotations;

using System.Diagnostics.CodeAnalysis;

using System.Text;

using BookingProject.Data.Models;

using BookingProject.Services.Mapping;

using BookingProject.Web.ViewModels.Image;

using Microsoft.AspNetCore.Http;

public class CreatePlaceInputModel : IMapTo<Place>

{

[Required(ErrorMessage = "Полето 'Име' е задължително.")]

public string Name { get; set; }

[Required(ErrorMessage = "Полето 'Категория' е задължително.")]

[Range(1, int.MaxValue)]

[Display(Name = "Category")]

public int CategoryId { get; set; }

[Required(ErrorMessage = "Полето 'Град' е задължително.")]

[Range(1, int.MaxValue)]

[Display(Name = "City")]

public int CityId { get; set; }

[Required(ErrorMessage = "Полето 'Адрес' е задължително.")]

[StringLength(50)]

public string Address { get; set; }

[Required(ErrorMessage = "Полето 'Описание' е задължително.")]

[StringLength(500)]

public string Description { get; set; }

[Required(ErrorMessage = "Полето 'Цена за нощувка' е задължително.")]

[Range(1, int.MaxValue, ErrorMessage = "Въведете стойност, по-голяма от 0.")]

public int PriceByNight { get; set; }

[Required(ErrorMessage = "Полето 'Брой спални стаи' е задължително.")]

public int BedroomsNum { get; set; }

[Required(ErrorMessage = "Полето 'Брой бани' е задължително.")]

public int BathroomsNum { get; set; }

[Required(ErrorMessage = "Полето 'Брой легла' е задължително.")]

[Range(1, 30, ErrorMessage = "Въведете стойност между 1 и 30.")]

[Display(Name = "BedsNum")]

public int BedsNum { get; set; }

[Required(ErrorMessage = "Полето 'Максимум гости' е задължително.")]

[Range(1, 30, ErrorMessage = "Въведете стойност между 1 и 30.")]

public int MaxGuest { get; set; }

public bool Pets { get; set; }

public bool Smoking { get; set; }

public List<Extra> PlaceExtras { get; set; }

public IEnumerable<CategoryDropDownViewModel> Categories { get; set; }

public IEnumerable<CityDropDownViewModel> Cities { get; set; }

// public IEnumerable<RegionDropDownViewModel> Regions { get; set; }

public List<ExtraViewModel> Extras { get; set; }

public List<Image> ImagesList { get; set; }

[Required(ErrorMessage = "Моля, добавете минимум една снимка.")]

public List<IFormFile> Images { get; set; }

}

}

ExtraViewModel.cs

namespace BookingProject.Web.ViewModels.Places

{

using BookingProject.Data.Models;

using BookingProject.Services.Mapping;

public class ExtraViewModel : IMapFrom<Extra>, IMapTo<Extra>

{

public int Id { get; set; }

public string Name { get; set; }

public bool IsSelected { get; set; } = false;

}

}

ListPlaceViewModel.cs

namespace BookingProject.Web.ViewModels.Places

{

using System;

using System.Collections.Generic;

using System.Text;

using BookingProject.Data.Models;

using BookingProject.Services.Mapping;

using BookingProject.Web.ViewModels.Home;

using BookingProject.Web.ViewModels.Reservations;

public class ListPlaceViewModel : IMapFrom<Place>

{

public int Id { get; set; }

public int Rating { get; set; }

public string ReviewsCount { get; set; }

public int CurrentPage { get; set; }

public int PagesCount { get; set; }

public List<PlaceReviewViewModel> Reviews { get; set; }

public IEnumerable<PlaceViewModel> Places { get; set; }

public IEnumerable<IndexCategoryViewModel> Categories { get; set; }

public IEnumerable<CityDropDownViewModel> Cities { get; set; }

public IEnumerable<GuestNumberDropDownViewModel> GuestNumbers { get; set; }

public int CityId { get; set; }

public int GuestNumber { get; set; }

public string Dates { get; set; }

public string SortBy { get; set; }

}

}

PlaceExtraViewModel.cs

namespace BookingProject.Web.ViewModels.Places

{

using AutoMapper;

using BookingProject.Data.Models;

using BookingProject.Services.Mapping;

public class PlaceExtraViewModel : IMapFrom<PlaceExtra>

{

public int Id { get; set; }

public string Name { get; set; }

public bool IsSelected { get; set; }

}

}

PlaceReservationViewModel.cs

namespace BookingProject.Web.ViewModels.Places

{

using BookingProject.Web.ViewModels.Reservations;

public class PlaceReservationViewModel

{

public PlaceViewModel PlaceViewModel { get; set; }

public CreateReservationViewModel CreateReservationViewModel { get; set; }

}

}

PlaceReviewViewModel.cs

namespace BookingProject.Web.ViewModels.Places

{

using System;

using BookingProject.Data.Models;

using BookingProject.Services.Mapping;

using Ganss.XSS;

public class PlaceReviewViewModel : IMapFrom<Review>

{

public int Id { get; set; }

public DateTime CreatedOn { get; set; }

public string Comment { get; set; }

public string UserUserName { get; set; }

public int Rating { get; set; }

}

}

* Resevations

CreateReservationViewModel.cs

using BookingProject.Data.Models;

using BookingProject.Services.Mapping;

using BookingProject.Web.ViewModels.Places;

using System;

using System.Collections.Generic;

using System.ComponentModel.DataAnnotations;

using System.Text;

namespace BookingProject.Web.ViewModels.Reservations

{

public class CreateReservationViewModel : IMapFrom<Reservation>, IMapTo<Reservation>

{

public int PlaceId { get; set; }

[Required]

public string Dates { get; set; }

public DateTime StartDate { get; set; }

public DateTime EndDate { get; set; }

public double PriceByNight { get; set; }

public double TotalPrice { get; set; }

public int NumNights { get; set; }

public PlaceViewModel Place { get; set; }

}

}

ListReservationViewModel.cs

namespace BookingProject.Web.ViewModels.Reservations

{

using System.Collections.Generic;

using BookingProject.Data.Models;

using BookingProject.Services.Mapping;

using BookingProject.Web.ViewModels.Places;

public class ListReservationViewModel : IMapFrom<Reservation>, IMapTo<Reservation>

{

public PlaceViewModel Place { get; set; }

public IEnumerable<ReservationViewModel> Reservations { get; set; }

}

}

ReservationViewModel.cs

namespace BookingProject.Web.ViewModels.Reservations

{

using System;

using System.ComponentModel.DataAnnotations;

using BookingProject.Data.Models;

using BookingProject.Services.Mapping;

using BookingProject.Web.ViewModels.Places;

public class ReservationViewModel : IMapFrom<Reservation>, IMapTo<Reservation>

{

public int Id { get; set; }

[Required]

public string UserId { get; set; }

public virtual ApplicationUser User { get; set; }

public int PlaceId { get; set; }

public virtual PlaceViewModel Place { get; set; }

public DateTime StartDate { get; set; }

public DateTime EndDate { get; set; }

public double PricePerNight { get; set; }

public double TotalPrice { get; set; }

public int NumNights { get; set; }

public bool Active { get; set; }

public bool Reviewed { get; set; }

}

}

* Reviews

CreateReviewViewModel.cs

namespace BookingProject.Web.ViewModels.Reviews

{

using System;

using System.Collections.Generic;

using System.Text;

public class CreateReviewInputModel

{

public int PlaceId { get; set; }

public string UserId { get; set; }

public string Comment { get; set; }

public int Rating { get; set; } = 5;

}

}

Views

* Categories

ByName.cshtml

@model BookingProject.Web.ViewModels.Categories.CategoryViewModel

@{ this.ViewData["Title"] = Model.Name; }

<h3 class="display-3">@Model.Name</h3>

<div class="container mt-3">

<div class="row">

@foreach (var place in Model.CardPlaces)

{

<div class="place-wrapper col-6 mb-4">

<div class="media-body">

<div class="place-card p-0">

<div class="place-price">

<i class="fas fa-tag custom-green"></i><span class="h5">@place.PriceByNight лв</span>/нощувка

</div>

<div class="place-fav">

<i class="far fa-heart"></i>

</div>

<a asp-controller="Places" asp-action="GetById" asp-route-id="@place.Id"><img class="place-img" src="~/images/backgroundA.jpg" alt="error"></a>

<a asp-controller="Places" asp-action="GetById" asp-route-id="@place.Id">

<div class="place-info">

<div class="row">

<div class="col-7 text-left pr-0">

@place.Name <span class="place-year">@place.CategoryName</span>

</div>

<div class="col-5 text-right">

@for (int i = 0; i < place.Rating; i++)

{

<span class="fa fa-star"></span>

}

@for (int i = place.Rating; i < 5; i++)

{

<span class="fa fa-star-o"></span>

}

<span>@place.Rating</span>(@place.ReviewsCount)

</div>

<div class="col-6 text-left place-city">

<i class="fas fa-map-marker-alt text-primary" style="font-size: 1rem"></i>@place.CityName

</div>

</div>

<div class="row mt-2 text-left">

<div class="col-6 pr-0">

<i class="fas fa-user-friends"></i><span class="place-second-info">@place.MaxGuest </span>

</div>

<div class="col-6 pr-0">

<i class="fas fa-bed"></i><span class="place-second-info"> @place.BedsNum</span>

</div>

</div>

<div class="row mt-1 text-left">

<div class="col-6 pr-0">

<i class="fas fa-bath"></i><span class="place-second-info"> @place.BathroomsNum</span>

</div>

<div class="col-6 pr-0">

<i class="fas fa-question"></i><span class="place-second-info"> ???</span>

</div>

</div>

</div>

</a>

</div>

</div>

</div>

}

</div>

</div>

<nav>

<ul class="pagination justify-content-center">

@if (this.Model.CurrentPage == 1)

{

<li class="page-item disabled">

<a class="page-link" href="#">Previous</a>

</li>

}

else

{

<li class="page-item">

<a class="page-link"

asp-route="bookingPlace"

asp-route-name="@this.Model.Name"

asp-route-page="@(this.Model.CurrentPage - 1)">Previous</a>

</li>}

@for (int i = 1; i <= this.Model.PagesCount; i++)

{

var active = i == this.Model.CurrentPage ? "active" : string.Empty;

<li class="page-item @active">

<a class="page-link"

asp-route="bookingPlace"

asp-route-name="@this.Model.Name"

asp-route-page="@i">@i</a>

</li>}

@if (this.Model.CurrentPage == this.Model.PagesCount)

{

<li class="page-item disabled">

<a class="page-link" href="#">Next</a>

</li> }

else

{

<li class="page-item">

<a class="page-link"

asp-route="bookingPlace"

asp-route-name="@this.Model.Name"

asp-route-page="@(this.Model.CurrentPage + 1)">Next</a>

</li>}

</ul>

</nav>

* Home

Index.cshtml

@using BookingProject.Common

@model BookingProject.Web.ViewModels.Home.IndexViewModel

@{

this.ViewData["Title"] = "stayin.bg - Първо провери при нас!";

var cities = Model.Cities.Select(x => new SelectListItem(x.Name, x.Id.ToString()));

var guestsNum = Model.GuestNumbers.Select(x => new SelectListItem(x.Name, x.Id.ToString()));

}

<div class="search-form">

<div class="form-container text-center">

<h4 class="text-center pt-3 font-italic">Намерете вашето място</h4>

<div class="row container m-0">

<form class="col-12 row m-0 p-0" asp-controller="Places" asp-action="ShowAllPlaces" method="get" enctype="multipart/form-data">

<div class="col-12 col-sm-4 row">

<div class="col-2 icon-container">

<i class="fas fa-location-arrow"></i>

</div>

<div class="col-10">

<select asp-for="CityId" asp-items="cities" class="form-control">

<option selected value="0"> Изберете град </option>

</select>

</div>

</div>

<div class="col-12 col-sm-4 row">

<div class="col-2 icon-container">

<i class="fas fa-users"></i>

</div>

<div class="col-10">

<select asp-for="GuestNumber" asp-items="guestsNum" class="form-control">

<option selected value="0"> Брой гости </option>

</select>

</div>

</div>

<div class="col-12 col-sm-4 row">

<div class="col-2 icon-container">

<i class="fa fa-calendar cal-start"></i>

</div>

<div class="col-10">

<input class="form-control" asp-for="Dates" type="text" autocomplete="off" placeholder="Изберете дати" />

</div>

</div>

<img class="w-100 linebreak" src="~/images/linebreak2.png" alt="">

<div class="col-12">

<button class="btn btn-secondary" type="submit">

Търси <i class="fas fa-search mb-2"></i>

</button>

</div>

</form>

</div>

</div>

</div>

<div class="row">

@foreach (var category in Model.Categories)

{

<div class="col-md-3 media">

<div class="card" style="width: 18rem;">

<a asp-controller="Places" asp-action="ShowAllPlaces" asp-route-categoryName="@category.Name" class="p-0">

<img class="card-img-top" src="~/images/categories/@category.ImageName" alt="@category.Name">

</a>

<a asp-controller="Places" asp-action="ShowAllPlaces" asp-route-categoryName="@category.Name" class="btn btn-dark">@category.Name</a>

</div>

</div>

}

</div>

@section scripts{

<script type="text/javascript">

$(function () {

$('input[name="Dates"]').daterangepicker({

autoUpdateInput: false,

locale: {

format: 'DD/MM/YYYY',

cancelLabel: 'Изчисти',

applyLabel: 'Запази',

fromLabel: 'От',

toLabel: 'До',

weekLabel: 'С',

daysOfWeek: [

'Нд',

'Пн',

'Вт',

'Ср',

'Чт',

'Пт',

'Сб'

],

monthNames: [

'Януари',

'Февруари',

'Март',

'Април',

'Май',

'Юни',

'Юли',

'Август',

'Септември',

'Октомври',

'Ноември',

'Декември'

],

"firstDay": 1

},

minDate: "@String.Format("{0:M/d/yyyy}", DateTime.Now)"

});

$('input[name="Dates"]').on('apply.daterangepicker', function(ev, picker) {

$(this).val(picker.startDate.format('DD/MM/YYYY') + ' - ' + picker.endDate.format('DD/MM/YYYY'));

});

$('input[name="Dates"]').on('cancel.daterangepicker', function (ev, picker) {

$(this).val('');

});

});

</script>

}

* Places

Administration.cshtml

@using BookingProject.Common

@model BookingProject.Web.ViewModels.Home.IndexViewModel

@{

this.ViewData["Title"] = "stayin.bg - Първо провери при нас!";

var cities = Model.Cities.Select(x => new SelectListItem(x.Name, x.Id.ToString()));

var guestsNum = Model.GuestNumbers.Select(x => new SelectListItem(x.Name, x.Id.ToString()));

}

<div class="search-form">

<div class="form-container text-center">

<h4 class="text-center pt-3 font-italic">Намерете вашето място</h4>

<div class="row container m-0">

<form class="col-12 row m-0 p-0" asp-controller="Places" asp-action="ShowAllPlaces" method="get" enctype="multipart/form-data">

<div class="col-12 col-sm-4 row">

<div class="col-2 icon-container">

<i class="fas fa-location-arrow"></i>

</div>

<div class="col-10">

<select asp-for="CityId" asp-items="cities" class="form-control">

<option selected value="0"> Изберете град </option>

</select>

</div>

</div>

<div class="col-12 col-sm-4 row">

<div class="col-2 icon-container">

<i class="fas fa-users"></i>

</div>

<div class="col-10">

<select asp-for="GuestNumber" asp-items="guestsNum" class="form-control">

<option selected value="0"> Брой гости </option>

</select>

</div>

</div>

<div class="col-12 col-sm-4 row">

<div class="col-2 icon-container">

<i class="fa fa-calendar cal-start"></i>

</div>

<div class="col-10">

<input class="form-control" asp-for="Dates" type="text" autocomplete="off" placeholder="Изберете дати" />

</div>

</div>

<img class="w-100 linebreak" src="~/images/linebreak2.png" alt="">

<div class="col-12">

<button class="btn btn-secondary" type="submit">

Търси <i class="fas fa-search mb-2"></i>

</button>

</div>

</form>

</div>

</div>

</div>

<div class="row">

@foreach (var category in Model.Categories)

{

<div class="col-md-3 media">

<div class="card" style="width: 18rem;">

<a asp-controller="Places" asp-action="ShowAllPlaces" asp-route-categoryName="@category.Name" class="p-0">

<img class="card-img-top" src="~/images/categories/@category.ImageName" alt="@category.Name">

</a>

<a asp-controller="Places" asp-action="ShowAllPlaces" asp-route-categoryName="@category.Name" class="btn btn-dark">@category.Name</a>

</div>

</div>

}

</div>

@section scripts{

<script type="text/javascript">

$(function () {

$('input[name="Dates"]').daterangepicker({

autoUpdateInput: false,

locale: {

format: 'DD/MM/YYYY',

cancelLabel: 'Изчисти',

applyLabel: 'Запази',

fromLabel: 'От',

toLabel: 'До',

weekLabel: 'С',

daysOfWeek: [

'Нд',

'Пн',

'Вт',

'Ср',

'Чт',

'Пт',

'Сб'

],

monthNames: [

'Януари',

'Февруари',

'Март',

'Април',

'Май',

'Юни',

'Юли',

'Август',

'Септември',

'Октомври',

'Ноември',

'Декември'

],

"firstDay": 1

},

minDate: "@String.Format("{0:M/d/yyyy}", DateTime.Now)"

});

$('input[name="Dates"]').on('apply.daterangepicker', function(ev, picker) {

$(this).val(picker.startDate.format('DD/MM/YYYY') + ' - ' + picker.endDate.format('DD/MM/YYYY'));

});

$('input[name="Dates"]').on('cancel.daterangepicker', function (ev, picker) {

$(this).val('');

});

});

</script>

}

Create.cshtml

@model BookingProject.Web.ViewModels.Places.CreatePlaceInputModel

@{ this.ViewData["Title"] = "Създаване на ново място";

var categories = Model.Categories.Select(x => new SelectListItem(x.Name, x.Id.ToString()));

var cities = Model.Cities.Select(x => new SelectListItem(x.Name, x.Id.ToString())); }

<h3>@this.ViewData["Title"]</h3>

<form method="post" asp-action="Create" enctype="multipart/form-data" class="row" id="uploadForm">

<div class="form-group col-6">

<label asp-for="Name">Име</label>

<input asp-for="Name" type="text" class="form-control" placeholder="Въведете име"/>

<span asp-validation-for="Name" class="text-danger"></span>

</div>

<div class="form-group col-6">

<label asp-for="CategoryId">Категория</label>

<select asp-for="CategoryId" asp-items="categories" class="form-control">

<option disabled selected value=""> -- изберете опция -- </option>

</select>

<span asp-validation-for="CategoryId" class="text-danger"></span>

</div>

<div class="form-group col-6">

<label asp-for="CityId">Град</label>

<select asp-for="CityId" asp-items="cities" class="form-control" placeholder="Изберете град">

<option disabled selected value=""> -- изберете опция -- </option>

</select>

<span asp-validation-for="CityId" class="text-danger"></span>

</div>

<div class="form-group col-6" style="height: 2em">

<label asp-for="Address">Адрес</label>

<textarea asp-for="Address" type="text" class="form-control" placeholder="Въведете адрес"></textarea>

<span asp-validation-for="Address" class="text-danger"></span>

</div>

<div class="form-group col-6" style="height: 2em">

<label asp-for="Description">Описание на помещението</label>

<textarea asp-for="Description" type="text" class="form-control" placeholder="Въведете oписание"></textarea>

<span asp-validation-for="Description" class="text-danger"></span>

</div>

<div class="form-group col-6">

<label asp-for="PriceByNight">Цена за нощувка</label>

<input asp-for="PriceByNight" type="number" class="form-control" id="PricePerNight" placeholder="Въведете цена за нощувка">

<span asp-validation-for="PriceByNight" class="text-danger"></span>

</div>

<div class="form-group col-6">

<label asp-for="BedroomsNum">Брой спални стаи</label>

<input asp-for="BedroomsNum" type="number" class="form-control" id="BedroomsNum" placeholder="Въведете брой спални стаи">

<span asp-validation-for="BedroomsNum" class="text-danger"></span>

</div>

<div class="form-group col-6">

<label asp-for="BedsNum">Брой легла</label>

<input asp-for="BedsNum" type="number" class="form-control" id="BedsNum" placeholder="Въведете общ брой легла">

<span asp-validation-for="BedsNum" class="text-danger"></span>

</div>

<div class="form-group col-6">

<label asp-for="BathroomsNum">Брой бани</label>

<input asp-for="BathroomsNum" type="number" class="form-control" id="BathroomsNum" placeholder="Въведете брой бани">

<span asp-validation-for="BathroomsNum" class="text-danger"></span>

</div>

<div class="form-group col-6">

<label asp-for="MaxGuest">Максимум гости</label>

<input asp-for="MaxGuest" type="number" class="form-control" id="MaxGuest" placeholder="Въведете максимален брой гости/вкл. деца/">

<span asp-validation-for="MaxGuest" class="text-danger"></span>

</div>

<div class="col-12 row">

<h3 class="col-12 text-center m-3">Екстри:</h3>

@for (int i = 0; i < Model.Extras.Count; i++)

{

<div class="form-group form-check col-3">

<input type="checkbox" asp-for="@Model.Extras[i].IsSelected" />

<label asp-for="@Model.Extras[i].IsSelected">@Model.Extras[i].Name</label>

<input type="hidden" asp-for="@Model.Extras[i].Id" />

<input type="hidden" asp-for="@Model.Extras[i].Name" />

</div>}

</div>

<div class="col-12 row">

<h3 class="col-12 text-center m-3">Правила:</h3>

<div class="form-group form-check col-6 text-center">

<label class="switch">

<input asp-for="Pets" type="checkbox">

<span class="slider round"></span>

</label>

<span>Домашни любимци</span>

</div>

<div class="form-group form-check col-6 text-center">

<label class="switch">

<input asp-for="Smoking" type="checkbox">

<span class="slider round"></span>

</label>

<span>Пушене</span>

</div>

</div>

<div class="col-12">

<h3 class="col-md-4 text-center m-3">Качване на снимки:</h3>

<div class="col-12 row" id='filecontainer'>

<span class="btn btn-default btn-file col-12 text-center m-3">

<i class="far fa-file-image fa-2x"></i><label for="imageInput"> Избери файлове</label>

</span>

<input asp-for="Images" type="file" class="custom-file-input" id="imageInput" multiple accept="image/png,image/jpg,image/jpeg"/>

<span asp-validation-for="Images" class="text-danger"></span>

</div>

</div>

<div class="col-12 row">

<div class="col-4"></div>

<div class="col-4">

<input type="submit" value="Създай" class="btn btn-secondary col-12">

</div>

<div class="col-4"></div>

</div>

</form>

@section scripts{

<script type="text/javascript">

//var inputLocalFont = document.getElementById("imageInput");

//inputLocalFont.addEventListener("change", previewImages, false);

//function previewImages() {

// var fileList = this.files;

// var anyWindow = window.URL || window.webkitURL;

// for (var i = 0; i < fileList.length; i++) {

// var objectUrl = anyWindow.createObjectURL(fileList[i]);

// $('.preview-area').append('<img src="' + objectUrl + '" width="270px" height="180px" padding:"10px" />');

// window.URL.revokeObjectURL(fileList[i]);

// }

//}

$(document).ready(function () {

if (window.File && window.FileList && window.FileReader) {

$("#imageInput").on("change", function (e) {

var files = e.target.files,

filesLength = files.length;

for (var i = 0; i < filesLength; i++) {

var f = files[i]

var fileReader = new FileReader();

fileReader.onload = (function (e) {

var file = e.target;

$("<div class=\"col-4 text-center imgBox\">" +

"<span class=\"pip\">" +

"<img class=\"imageThumb\" src=\"" + e.target.result + "\" title=\"" + file.name + "\"/>" +

"<br/><span class=\"remove\">Премахни</span>" +

"</span>" +

"</div>").insertAfter("#imageInput");

$(".remove").click(function () {

$(this).parent(".pip").parent(".imgBox").remove();

});

});

fileReader.readAsDataURL(f);

}

});

} else {

alert("Your browser doesn't support to File API")

}

});

</script>

}

Edit.cshtml

@model BookingProject.Web.ViewModels.Places.EditPlaceViewModel

@{ this.ViewData["Title"] = "Създаване на ново място";

var categories = Model.Categories.Select(x => new SelectListItem(x.Name, x.Id.ToString()));

var cities = Model.Cities.Select(x => new SelectListItem(x.Name, x.Id.ToString())); }

<h3>@this.ViewData["Title"]</h3>

<form method="post" asp-action="Create" enctype="multipart/form-data" class="row" id="uploadForm">

<div class="form-group col-6">

<label asp-for="Name">Име</label>

<input asp-for="@Model.Name" class="form-control" placeholder="Въведете име" />

<span asp-validation-for="Name" class="text-danger"></span>

</div>

<div class="form-group col-6">

<label asp-for="CategoryId">Категория</label>

<select asp-for="CategoryId" asp-items="categories" class="form-control" disabled>

<option disabled selected value=""> -- изберете опция -- </option>

</select>

<span asp-validation-for="CategoryId" class="text-danger"></span>

</div>

<div class="form-group col-6">

<label asp-for="CityId">Град</label>

<select asp-for="CityId" asp-items="cities" class="form-control" placeholder="Изберете град" disabled>

<option disabled selected value=""> -- изберете опция -- </option>

</select>

<span asp-validation-for="CityId" class="text-danger"></span>

</div>

<div class="form-group col-6" style="height: 2em">

<label asp-for="Address">Адрес</label>

<textarea asp-for="Address" type="text" class="form-control" placeholder="Въведете адрес" value="Test"></textarea>

<span asp-validation-for="Address" class="text-danger"></span>

</div>

<div class="form-group col-6" style="height: 2em">

<label asp-for="Description">Описание на помещението</label>

<textarea asp-for="Description" type="text" class="form-control" placeholder="Въведете oписание" value="Test"></textarea>

<span asp-validation-for="Description" class="text-danger"></span>

</div>

<div class="form-group col-6">

<label asp-for="PriceByNight">Цена за нощувка</label>

<input asp-for="PriceByNight" type="number" class="form-control" id="PricePerNight" value=5 placeholder="Въведете цена за нощувка">

<span asp-validation-for="PriceByNight" class="text-danger"></span>

</div>

<div class="form-group col-6">

<label asp-for="BedroomsNum">Брой спални стаи</label>

<input asp-for="BedroomsNum" type="number" class="form-control" id="BedroomsNum" value=5 placeholder="Въведете брой спални стаи">

<span asp-validation-for="BedroomsNum" class="text-danger"></span>

</div>

<div class="form-group col-6">

<label asp-for="BedsNum">Брой легла</label>

<input asp-for="BedsNum" type="number" class="form-control" id="BedsNum" value=5 placeholder="Въведете общ брой легла">

<span asp-validation-for="BedsNum" class="text-danger"></span>

</div>

<div class="form-group col-6">

<label asp-for="BathroomsNum">Брой бани</label>

<input asp-for="BathroomsNum" type="number" class="form-control" id="BathroomsNum" value=5 placeholder="Въведете брой бани">

<span asp-validation-for="BathroomsNum" class="text-danger"></span>

</div>

<div class="form-group col-6">

<label asp-for="MaxGuest">Максимум гости</label>

<input asp-for="MaxGuest" type="number" class="form-control" id="MaxGuest" value=5 placeholder="Въведете максимален брой гости/вкл. деца/">

<span asp-validation-for="MaxGuest" class="text-danger"></span>

</div>

<div class="col-12 row">

<h3 class="col-12 text-center m-3">Екстри:</h3>

@for (int i = 0; i < Model.Extras.Count; i++)

{

<div class="form-group form-check col-3">

<input type="checkbox" asp-for="@Model.Extras[i].IsSelected" />

<label asp-for="@Model.Extras[i].IsSelected">@Model.Extras[i].Name</label>

<input type="hidden" asp-for="@Model.Extras[i].Id" />

<input type="hidden" asp-for="@Model.Extras[i].Name" />

</div>}

</div>

<div class="col-12 row">

<h3 class="col-12 text-center m-3">Правила:</h3>

<div class="form-group form-check col-6 text-center">

<label class="switch">

<input asp-for="Pets" type="checkbox">

<span class="slider round"></span>

</label>

<span>Домашни любимци</span>

</div>

<div class="form-group form-check col-6 text-center">

<label class="switch">

<input asp-for="Smoking" type="checkbox">

<span class="slider round"></span>

</label>

<span>Пушене</span>

</div>

</div>

<div class="col-12">

<input type="submit" value="Запази" class="btn btn-secondary col-4" style="margin-left: 27rem">

</div>

</form>

GetById.cshtml

@using BookingProject.Common

@model BookingProject.Web.ViewModels.Places.PlaceViewModel

@{

this.ViewData["Title"] = "stayin.bg - Първо провери при нас!";

}

<div class="container">

<div class="row">

<div class="col-8 primary-info row">

<div class="slideshow-container col-12 pr-0">

@for (int i = 0; i < Model.Images.Count; i++)

{

<div class="mySlides fade">

<div class="numbertext">@(i+1) / @Model.Images.Count</div>

<img src="~/images/@Model.Images[i].Name" style="width:100%">

</div>

}

<a class="prev" onclick="plusSlides(-1)">&#10094;</a>

<a class="next" onclick="plusSlides(1)">&#10095;</a>

</div>

<br>

<div class="col-12 text-center">

@for (int i = 1; i <= Model.Images.Count; i++)

{

<span class="dot" onclick="currentSlide(@i)"></span>

}

</div>

</div>

<div class="col-4 row">

<div class="col-12">

<form asp-controller="Reservations" asp-action="DetailReservation" method="get" enctype="multipart/form-data">

<div class="row text-left primary-info">

<div class="col-12 h3 text-center">Резервация</div>

<div class="col-12 text-center">

<input class="form-control mb-3" asp-for="Dates" type="text" autocomplete="off" placeholder="Изберете дати" />

</div>

<input type="hidden" asp-for="PlaceId" value="@Model.Id" />

<div class="col-12 mb-2 ml-5">

<i class="fas fa-tag custom-green"></i>

<span class="place-info-title">Цена: </span>

<span class="place-info"> @Model.PriceByNight </span>

<span class="place-info-title">лв/нощувка</span>

</div>

<div class="col-12 mb-2 ml-5">

<i class="fas fa-user-friends"></i>

<span class="place-info-title">Максимум гости: </span>

<span class="place-info">@Model.MaxGuest</span>

</div>

<div class="col-12 mb-2 ml-5">

<i class="fas fa-person-booth"></i>

<span class="place-info-title">Брой спални стаи: </span>

<span class="place-info">@Model.BedroomsNum</span>

</div>

<div class="col-12 mb-2 ml-5">

<i class="fas fa-bath"></i>

<span class="place-info-title">Брой бани: </span>

<span class="place-info">@Model.BathroomsNum</span>

</div>

<div class="col-12 mb-2 ml-5">

<i class="fas fa-bed"></i>

<span class="place-info-title">Брой легла: </span>

<span class="place-info">@Model.BedsNum</span>

</div>

<div class="col-12 text-center">

<input class="btn btn-secondary" type="submit" value="Резервирай" />

</div>

</div>

</form>

</div>

</div>

</div>

<div class="row primary-info">

<div class="col-8 row">

<div class="col-12 text-center">

<span class="place-name">@Model.Name</span>

</div>

<img class="w-100 linebreak" src="~/images/linebreak2.png" alt="">

<div class="col-6">

<i class="fas fa-home text-success" style="font-size: 1rem"></i>

<span class="place-info-title">Категория:</span>

<span class="place-info">@Model.CategoryName</span>

</div>

<div class="col-6">

<i class="fas fa-map-marker-alt text-primary" style="font-size: 1rem"></i>

<span class="place-info-title">Населено място:</span>

<span class="place-info">@Model.CityName <span class="place-info-title">(@Model.City.Region.Name)</span></span>

</div>

<div class="col-6">

<i class="fas fa-location-arrow text-primary" style="font-size: 0.9rem"></i>

<span class="place-info-title">Адрес:</span>

<span class="place-info">@Model.Address</span>

</div>

<div class="col-6">

<span class="place-info-title">Рейтинг:</span>

@for (int i = 0; i < Model.Rating; i++)

{

<span class="fa fa-star" style="color:orange"></span>

}

@for (int i = Model.Rating; i < 5; i++)

{

<span class="fa fa-star-o"></span>

}

@\*<span class="place-info">@Model.Rating</span>(@Model.ReviewsCount)\*@

</div>

</div>

<div class="container col-4 row text-center">

<div class="col-12">

<span class="place-info-title">Собственик:</span>

</div>

<div class="col-12">

<i class="far fa-user"></i><span class="place-info"> @Model.User.Email</span>

</div>

<div class="col-12">

<i class="fas fa-phone text-primary"></i><span class="place-info-title"> @Model.User.PhoneNumber</span>

</div>

</div>

</div>

<img class="w-100 linebreak" src="~/images/linebreak2.png" alt="">

<div class="row">

@if (Model.Description.Length > 0)

{

<div class="col-12 h5 mb-2">

Описание:

</div>

<div class="col-12">

@Model.Description

</div>

<img class="w-100 linebreak" src="~/images/linebreak2.png" alt="">

}

@if (Model.PlaceExtras.Count > 0)

{

<div class="col-12 h5 mb-2">

Екстри:

</div>

<div class="col-12 row">

@foreach (var pExtra in Model.PlaceExtras)

{

<div class="col-4">

<i class="fas fa-check font-italic text-success"></i><span class="place-info"> @pExtra.Extra.Name</span>

</div>

}

</div>

<img class="w-100 linebreak" src="~/images/linebreak2.png" alt="">

}

<div class="col-12 h5 mb-3">

Правила за наемане:

</div>

@if (@Model.Pets == true)

{

<div class="col-12">

<label asp-for="Pets"><i class="fas fa-dog fa-2x"></i>Мястото предлага опция за домашни любимци.</label>

</div>

}

else

{

<div class="col-12">

<label asp-for="Pets"><img src="~/images/no-pets.png" width="36" height="36"> Мястото не предлага опция за домашни любимци.</label>

</div>

}

@if (Model.Smoking == true)

{

<div class="col-12">

<label asp-for="Smoking"><i class="fas fa-smoking fa-2x"></i>Мястото предлага опция за тютюнопушене.</label>

</div>

}

else

{

<div class="col-12">

<label asp-for="Smoking"><img src="~/images/no-smoking.svg" width="36" height="36"> Мястото не предлага опция за тютюнопушене.</label>

</div>

}

</div>

</div>

<div class="row">

<div class="col-12 col-lg-8 row pr-0">

@if (this.User.Identity.IsAuthenticated)

{

<div class="px-4 pt-3">

<button type="button" class="btn btn-primary float-right" onclick="showAddReviewForm(0)"><i class="fa fa-plus"></i>&nbsp; Добави мнение</button>

</div>

<div class="clearfix"></div>}

</div>

</div>

</div>

<div class="container">

<form asp-controller="Reviews" asp-action="Create" method="post" id="AddReviewForm" style="display: none">

<input type="hidden" name="PlaceId" value="@this.Model.Id" />

<div class="rating-box">

<h4>Оценка:</h4>

<div class="ratings">

<span class="fa fa-star-o"></span>

<span class="fa fa-star-o"></span>

<span class="fa fa-star-o"></span>

<span class="fa fa-star-o"></span>

<span class="fa fa-star-o"></span>

</div>

<input type="hidden" name="Rating" id="rating-value" />

</div>

<div class="mt-3 mb-3">

<textarea name="Comment" id="Comment" class="form-control" placeholder="Напишете коментар..."></textarea>

</div>

<div class="text-center mb-5">

<input type="submit" class="btn btn-primary" value="Добавете мнение" />

</div>

</form>

</div>

@foreach (var review in this.Model.Reviews)

{

<div class="container-fluid mt-100">

<div class="row">

<div class="col-md-12">

<div class="card mb-4">

<div class="card-header">

<div class="media flex-wrap w-100 align-items-center">

<img src="https://virtualgallery1-293846.c.cdn77.org/img\_HTML/mr\_nobody\_new.svg" class="d-block ui-w-40 rounded-circle" alt="" height="64">

<div class="media-body ml-3">

@review.UserUserName

<div class="text-muted small">

@review.CreatedOn

</div>

</div>

</div>

</div>

<div class="card-body">

<article>

@review.Comment

</article>

@for (int i = 0; i < review.Rating; i++)

{

<span class="fa fa-star"></span>

}

@for (int i = review.Rating; i < 5; i++)

{

<span class="fa fa-star-o"></span>

}

</div>

</div>

</div>

</div>

</div>

}

@section scripts{

<script type="text/javascript">

var slideIndex = 1;

showSlides(slideIndex);

function plusSlides(n) {

showSlides(slideIndex += n);

}

function currentSlide(n) {

showSlides(slideIndex = n);

}

function showSlides(n) {

var i;

var slides = document.getElementsByClassName("mySlides");

var dots = document.getElementsByClassName("dot");

if (n > slides.length) { slideIndex = 1 }

if (n < 1) { slideIndex = slides.length }

for (i = 0; i < slides.length; i++) {

slides[i].style.display = "none";

}

for (i = 0; i < dots.length; i++) {

dots[i].className = dots[i].className.replace(" active", "");

}

slides[slideIndex - 1].style.display = "block";

dots[slideIndex - 1].className += " active";

}

function showAddReviewForm() {

$("#AddReviewForm").show();

$([document.documentElement, document.body]).animate({

scrollTop: $("#AddReviewForm").offset().top

}, 1000);

}

$(function () {

var arr = @Html.Raw(Model.ReservedDays)

$('input[name="Dates"]').daterangepicker({

autoUpdateInput: false,

locale: {

format: 'DD/MM/YYYY',

cancelLabel: 'Изчисти',

applyLabel: 'Запази',

fromLabel: 'От',

toLabel: 'До',

weekLabel: 'С',

daysOfWeek: [

'Нд',

'Пн',

'Вт',

'Ср',

'Чт',

'Пт',

'Сб'

],

monthNames: [

'Януари',

'Февруари',

'Март',

'Април',

'Май',

'Юни',

'Юли',

'Август',

'Септември',

'Октомври',

'Ноември',

'Декември'

],

"firstDay": 1

},

minDate: "@String.Format("{0:M/d/yyyy}", DateTime.Now)",

isInvalidDate: function (date) {

var formatted = date.format('DD/MM/YYYY');

return arr.indexOf(formatted) > -1;

}

});

$('input[name="Dates"]').on('apply.daterangepicker', function(ev, picker) {

$(this).val(picker.startDate.format('DD/MM/YYYY') + ' - ' + picker.endDate.format('DD/MM/YYYY'));

});

$('input[name="Dates"]').on('cancel.daterangepicker', function (ev, picker) {

$(this).val('');

});

});

</script>

}

<style>

.card {

max-width: 1000px;

margin: 0 auto;

}

</style>

ShowAllPlaces.cshtml

@using BookingProject.Common

@model BookingProject.Web.ViewModels.Places.ListPlaceViewModel

@{

this.ViewData["Title"] = $"Welcome to {GlobalConstants.SystemName}!";

var cities = Model.Cities.Select(x => new SelectListItem(x.Name, x.Id.ToString()));

var guestsNum = Model.GuestNumbers.Select(x => new SelectListItem(x.Name, x.Id.ToString()));

var sortOprtions = new List<SelectListItem>();

sortOprtions.Add(new SelectListItem { Text = "Цена възходяща", Value = "asc" });

sortOprtions.Add(new SelectListItem { Text = "Цена низходяща", Value = "desc" });

}

<div class="container mt-3">

@\*FILTERS START\*@

<div class="row">

<div class="row container m-0">

<form class="col-12 row m-0 p-0" asp-controller="Places" asp-action="ShowAllPlaces" method="get" enctype="multipart/form-data">

<div class="col-4 row">

<div class="col-2 icon-container">

<i class="fas fa-location-arrow text-primary"></i>

</div>

<div class="col-10">

<select asp-for="CityId" asp-items="cities" class="form-control">

<option selected value="0"> Изберете град </option>

</select>

</div>

</div>

<div class="col-4 row">

<div class="col-2 icon-container">

<i class="fas fa-users"></i>

</div>

<div class="col-10">

<select asp-for="GuestNumber" asp-items="guestsNum" class="form-control">

<option selected value="0"> Брой гости </option>

</select>

</div>

</div>

<div class="col-4 row">

<div class="col-2 icon-container">

<i class="fa fa-calendar cal-start text-success"></i>

</div>

<div class="col-10">

<input class="form-control" asp-for="Dates" type="text" autocomplete="off" placeholder="Изберете дати" />

</div>

</div>

<div class="col-12 mt-2 text-center row">

<div class="col-3"></div>

<div class="col-3 text-center">

<select asp-for="SortBy" asp-items="sortOprtions" class="form-control sort-select">

<option selected value="id"> Сортирай по ... </option>

</select>

</div>

<div class="col-3">

<button class="btn btn-secondary" type="submit">

Търси <i class="fas fa-search mb-2"></i>

</button>

</div>

<div class="col-3"></div>

</div>

</form>

</div>

@\*FILTERS END\*@

@if (Model.Places.Count() == 0)

{

<div class="text-center">

<h3 class="display-4">Не намерихме резултати отговарящи на вашето търсене</h3>

</div>

}

@foreach (var place in Model.Places)

{

<div class="place-wrapper col-6 mt-3 mb-4">

<div class="media-body">

<div class="place-card p-0">

<div class="place-price">

<i class="fas fa-tag custom-green"></i><span class="h5">@place.PriceByNight лв</span>/нощувка

</div>

<div class="place-fav">

<i class="far fa-heart"></i>

</div>

<a asp-controller="Places" asp-action="GetById" asp-route-id="@place.Id"><img class="place-img" src="~/images/@place.Images[0].Name" alt="error"></a>

<a asp-controller="Places" asp-action="GetById" asp-route-id="@place.Id">

<div class="place-info">

<div class="row">

<div class="col-7 text-left pr-0">

@place.Name <span class="place-year">@place.CategoryName</span>

</div>

<div class="col-5 text-right row p-0">

<div class="col-12">

@for (int i = 0; i < place.Rating; i++)

{

<span class="fa fa-star"></span>

}

@for (int i = place.Rating; i < 5; i++)

{

<span class="fa fa-star-o"></span>

}

</div>

<div class="col-12 pl-0">

<div class="place-city">

<i class="fas fa-map-marker-alt text-primary" style="font-size: 1rem"></i> @place.CityName

</div>

</div>

</div>

</div>

<div class="row mt-2 text-left">

<div class="col-6 pr-0">

<i class="fas fa-user-friends"></i><span class="place-second-info-title"> Места: </span><span class="place-second-info"> @place.MaxGuest </span>

</div>

<div class="col-6 pr-0">

<i class="fas fa-bed"></i><span class="place-second-info-title"> Брой легла: </span><span class="place-second-info"> @place.BedsNum</span>

</div>

</div>

<div class="row mt-1 text-left">

<div class="col-6 pr-0">

<i class="fas fa-bath"></i><span class="place-second-info-title"> Брой бани: </span><span class="place-second-info"> @place.BathroomsNum</span>

</div>

<div class="col-6 pr-0">

<i class="fas fa-address-book"></i><span class="place-second-info-title"> Предишни наематели: </span><span class="place-second-info"> @place.Reservations.Count </span>

</div>

</div>

</div>

</a>

</div>

</div>

</div>

}

</div>

</div>

@\*<nav>

<ul class="pagination justify-content-center">

@if (this.Model.CurrentPage == 1)

{

<li class="page-item disabled">

<a class="page-link" href="#">Previous</a>

</li>

}

else

{

<li class="page-item">

<a class="page-link"

asp-route="cardsList"

asp-route-name="@this.Model.Id"

asp-route-page="@(this.Model.CurrentPage - 1)">Previous</a>

</li>

}

@for (int i = 1; i <= this.Model.PagesCount; i++)

{

var active = i == this.Model.CurrentPage ? "active" : string.Empty;

<li class="page-item @active">

<a class="page-link"

asp-route="cardsList"

asp-route-name="@this.Model.Id"

asp-route-page="@i">@i</a>

</li>

}

@if (this.Model.CurrentPage == this.Model.PagesCount)

{

<li class="page-item disabled">

<a class="page-link" href="#">Next</a>

</li>

}

else

{

<li class="page-item">

<a class="page-link"

asp-route="cardsList"

asp-route-name="@this.Model.Id"

asp-route-page="@(this.Model.CurrentPage + 1)">Next</a>

</li>

}

</ul>

</nav>\*@

@section scripts{

<script type="text/javascript">

$(function () {

$('input[name="Dates"]').daterangepicker({

autoUpdateInput: false,

locale: {

format: 'DD/MM/YYYY',

cancelLabel: 'Изчисти',

applyLabel: 'Запази',

fromLabel: 'От',

toLabel: 'До',

weekLabel: 'С',

daysOfWeek: [

'Нд',

'Пн',

'Вт',

'Ср',

'Чт',

'Пт',

'Сб'

],

monthNames: [

'Януари',

'Февруари',

'Март',

'Април',

'Май',

'Юни',

'Юли',

'Август',

'Септември',

'Октомври',

'Ноември',

'Декември'

],

"firstDay": 1

},

minDate: "@String.Format("{0:M/d/yyyy}", DateTime.Now)"

});

$('input[name="Dates"]').on('apply.daterangepicker', function(ev, picker) {

$(this).val(picker.startDate.format('DD/MM/YYYY') + ' - ' + picker.endDate.format('DD/MM/YYYY'));

});

$('input[name="Dates"]').on('cancel.daterangepicker', function (ev, picker) {

$(this).val('');

});

});

</script>

}

<style>

.sort-select {

max-width: 200px;

}

</style>

ShowUserPlaces.cshtml

@using BookingProject.Common

@model BookingProject.Web.ViewModels.Places.ListPlaceViewModel

@{

this.ViewData["Title"] = $"Welcome to {GlobalConstants.SystemName}!";

}

<div class="container mt-3">

<div class="row">

@if (Model.Places.Count() == 0)

{

<div class="col-12 text-center">

<h3 class="display-4 text-center">Все още нямате места</h3>

<a asp-action="Create" asp-controller="Places">

<h3 class="text-secondary font-italic">Създайте вашето първо място тук</h3>

</a>

</div>

}

@foreach (var place in Model.Places)

{

<div class="place-wrapper col-6 mt-3 mb-4">

<div class="media-body">

<div class="choose">

<a role="button" asp-controller="Places" asp-action="GetById" asp-route-id="@place.Id">

<div class="p-0 place-view">

<div class="view-wrapper">

<div class="title">

<i class="far fa-eye"></i> Преглед

</div>

</div>

</div>

</a>

<a role="button" asp-controller="Reservations" asp-action="PlaceReservations" asp-route-id="@place.Id">

<div class="p-0 place-reservations">

<div class="reservations-wrapper">

<div class="title">

<i class="fas fa-book"></i> Резервации

</div>

</div>

</div>

</a>

<a role="button" asp-controller="Places" asp-action="Edit" asp-route-id="@place.Id">

<div class="p-0 place-edit">

<div class="edit-wrapper">

<div class="title">

<i class="far fa-edit"></i> Редактирай

</div>

</div>

</div>

</a>

<div class="place-card p-0">

<div class="place-price">

<i class="fas fa-tag custom-green"></i><span class="h5">@place.PriceByNight лв</span>/нощувка

</div>

<div class="place-fav">

<i class="far fa-heart"></i>

</div>

<a asp-controller="Places" asp-action="GetById" asp-route-id="@place.Id"><img class="place-img" src="~/images/@place.Images[0].Name" alt="error"></a>

<a asp-controller="Places" asp-action="GetById" asp-route-id="@place.Id">

<div class="place-info">

<div class="row">

<div class="col-7 text-left pr-0">

@place.Name <span class="place-year">@place.CategoryName</span>

</div>

<div class="col-5 text-right">

@for (int i = 0; i < place.Rating; i++)

{

<span class="fa fa-star"></span>

}

@for (int i = place.Rating; i < 5; i++)

{

<span class="fa fa-star-o"></span>

}

</div>

<div class="col-6 text-left place-city">

<i class="fas fa-map-marker-alt text-primary" style="font-size: 1rem"></i>@place.CityName

</div>

<div class="col-6 text-right place-city">

@place.ReviewsCount наемателя

</div>

</div>

<div class="row mt-2 text-left">

<div class="col-6 pr-0">

<i class="fas fa-user-friends"></i><span class="place-second-info">@place.MaxGuest </span>

</div>

<div class="col-6 pr-0">

<i class="fas fa-bed"></i><span class="place-second-info"> @place.BedsNum</span>

</div>

</div>

<div class="row mt-1 text-left">

<div class="col-6 pr-0">

<i class="fas fa-bath"></i><span class="place-second-info"> @place.BathroomsNum</span>

</div>

<div class="col-6 pr-0">

<i class="fas fa-question"></i><span class="place-second-info"> ???</span>

</div>

</div>

</div>

</a>

</div>

</div>

</div>

</div>

}

</div>

</div>

@section scripts{

<script type="text/javascript">

$(function () {

$('input[name="Dates"]').daterangepicker({

autoUpdateInput: false,

locale: {

format: 'DD/MM/YYYY',

cancelLabel: 'Изчисти',

applyLabel: 'Запази',

fromLabel: 'От',

toLabel: 'До',

weekLabel: 'С',

daysOfWeek: [

'Нд',

'Пн',

'Вт',

'Ср',

'Чт',

'Пт',

'Сб'

],

monthNames: [

'Януари',

'Февруари',

'Март',

'Април',

'Май',

'Юни',

'Юли',

'Август',

'Септември',

'Октомври',

'Ноември',

'Декември'

],

"firstDay": 1

},

minDate: "@String.Format("{0:M/d/yyyy}", DateTime.Now)"

});

$('input[name="Dates"]').on('apply.daterangepicker', function(ev, picker) {

$(this).val(picker.startDate.format('DD/MM/YYYY') + ' - ' + picker.endDate.format('DD/MM/YYYY'));

});

$('input[name="Dates"]').on('cancel.daterangepicker', function (ev, picker) {

$(this).val('');

});

});

</script>

}

<style>

.choose {

position: relative;

border-radius: 8px;

height: 100%;

transition: all 0.5s ease;

cursor: pointer;

}

.choose .title {

position: absolute;

top: 50%;

left: 50%;

transform: translate(-50%, -50%);

transition: all 0.3s ease;

opacity: 0;

color: white;

text-shadow: 2px 2px 8px black;

text-align: center;

font-size: 1.1rem;

transition: all 0.5s ease;

}

.choose .place-view {

display: block;

width: 33%;

height: 100%;

position: absolute;

z-index: 100;

transition: all 0.5s ease;

background-color: rgba(0, 160, 0, 0.3);

opacity: 0;

border-top-left-radius: 8px;

border-bottom-left-radius: 8px;

}

.choose .place-edit {

width: 34%;

height: 100%;

position: absolute;

z-index: 100;

right: 0;

transition: all 0.5s ease;

background-color: rgba(255, 0, 0, 0.3);

opacity: 0;

}

.choose .place-reservations {

width: 33%;

height: 100%;

position: absolute;

z-index: 100;

left: 33%;

transition: all 0.5s ease;

background-color: rgba(0, 47, 255, 0.3);

opacity: 0;

}

.choose:hover .place-view {

opacity: 1;

border-top-left-radius: 8px;

border-bottom-left-radius: 8px;

}

.choose:hover .place-view:hover .view-wrapper {

background-color: rgba(0, 160, 0, 0.45);

}

.choose:hover .place-edit {

opacity: 1;

border-top-right-radius: 8px;

border-bottom-right-radius: 8px;

}

.choose:hover .place-edit:hover .edit-wrapper {

background-color: rgba(255, 0, 0, 0.45);

}

.choose:hover .place-reservations {

opacity: 1;

}

.choose:hover .place-reservations:hover .reservations-wrapper {

background-color: rgba(0, 47, 255, 0.45);

}

.choose:hover .title {

opacity: 1;

transition: all 0.5s ease;

}

.place-view :hover > .place-view {

background-color: red !important;

}

.view-wrapper {

height: 100%;

width: 100%;

transition: all 0.5s ease;

border-top-left-radius: 8px;

border-bottom-left-radius: 8px;

}

.edit-wrapper {

height: 100%;

width: 100%;

transition: all 0.5s ease;

border-top-right-radius: 8px;

border-bottom-right-radius: 8px;

}

.reservations-wrapper {

height: 100%;

width: 100%;

transition: all 0.5s ease;

}

</style>

* Reservations

DetailReservation.cshtml

@model BookingProject.Web.ViewModels.Reservations.CreateReservationViewModel

@{ this.ViewData["Title"] = "Завършете вашата резервация";

var place = Model.Place;

var culture = new System.Globalization.CultureInfo("bg-BG");

}

<h1 class="text-center">@this.ViewData["Title"]</h1>

<div class="container">

<div class="row">

<div class="col-6">

<h3 class="mt-5">

Важни дати

</h3>

<div class="row">

<div class="col-12">

<div class="date-box text-center p-1">

<span class="font-weight-bold h4">

@Model.StartDate.Day

</span>

<br class="p-0 m-0" />

@Model.StartDate.ToString("MMM", culture)

</div>

<div class="date-info">

Настаняване - @Model.StartDate.ToString("dddd", culture)

<br class="p-0 m-0" />

след 15:00 часа

</div>

</div>

<div class="col-12">

<div class="date-box text-center p-1">

<span class="font-weight-bold h4">

@Model.EndDate.Day

</span>

<br class="p-0 m-0" />

@Model.EndDate.ToString("MMM", culture)

</div>

<div class="date-info">

Освобождаване - @Model.EndDate.ToString("dddd", culture)

<br class="p-0 m-0" />

до 12:00 часа

</div>

</div>

</div>

<h3 class="mt-4 mb-2">

Имайте предвид:

</h3>

@if (!place.Pets)

{

<div class="icon-item">

<img src="~/images/no-pets.png" width="36" height="36"> Не се допускат домашни любимци

</div>

}

@if (!place.Smoking)

{

<div class="icon-item">

<img src="~/images/no-smoking.svg" width="36" height="36"> Пушенето забранено

</div>

}

<h3 class="mt-4 mb-2">

Цена:

</h3>

<div class="price-backup">

<div>

<span class="h4">@place.PriceByNight</span>лв/ден х <span class="h4">@Model.NumNights</span>дни

</div>

<div>

<span>Общо: <span class="h3">@Math.Round(Double.Parse(place.PriceByNight) \* Model.NumNights, 2)</span>лв</span>

</div>

</div>

<div class="icon-item">

Метод на плащане:

<img src="~/images/visa-mastercard-icon.png"

class="visa-mc-icon"

alt="" />

</div>

<form asp-controller="Reservations" asp-action="CreateReservation" method="post" enctype="multipart/form-data" autocomplete="off">

<input type="hidden" asp-for="NumNights" />

<input type="hidden" asp-for="StartDate" />

<input type="hidden" asp-for="EndDate" />

<input type="hidden" asp-for="TotalPrice" />

<input type="hidden" asp-for="Dates" />

<input type="hidden" asp-for="PlaceId" />

<div class="form-group">

<div class="input-group">

<div class="input-group-prepend">

<div class="input-group-text bg-white">

<i class="fa fa-user-circle width18"></i>

</div>

</div>

<input name="card\_owner\_name"

class="form-control"

pattern="^[A-Z][A-Z\s]\*$"

placeholder="Име на картодържателя"

required />

</div>

</div>

<div class="form-group">

<div class="input-group">

<div class="input-group-prepend">

<div class="input-group-text bg-white">

<i class="fa fa-credit-card"></i>

</div>

</div>

<input name="card\_number"

class="form-control"

placeholder="Номер на карта"

pattern="^\d{16}"

required />

</div>

</div>

<div class="form-row">

<div class="form-group col-md-7">

<div class="input-group">

<div class="input-group-prepend">

<div class="input-group-text bg-white">

<i class="fa fa-calendar m-0"></i>

</div>

</div>

<input name="expiration\_date"

pattern="^\d{2}\/\d{2}$"

class="form-control"

placeholder="Валидност"

required />

</div>

</div>

<div class="form-group col-md-5">

<div class="input-group">

<div class="input-group-prepend">

<div class="input-group-text bg-white">

<i class="fa fa-key"></i>

</div>

</div>

<input type="password"

name="cvv"

class="form-control"

placeholder="CVV"

pattern="^\d{3}"

required />

</div>

</div>

</div>

<div class="icon-item row">

<div class="col-12 col-md-8 mb-3 row">

<div class="centered-text col-10">

Съгласен съм с условията

</div>

<div class="can-toggle demo-rebrand-2 col-2 ml-0">

<input id="e"

type="checkbox"

name="confirmation"

required />

<label for="e">

<div class="can-toggle\_\_switch"

data-checked="&#10004;"

data-unchecked="&#10006;"></div>

</label>

</div>

</div>

<div class="col-12 w-100 text-center">

<button type="submit"

class="btn btn-primary mt-0">

РЕЗЕРВИРАЙ

</button>

</div>

</div>

</form>

</div>

<div class="col-6">

<div class="place-wrapper mb-4">

<div class="media-body">

<div class="place-card p-0">

<div class="place-price">

<i class="fas fa-tag custom-green"></i><span class="h5">@place.PriceByNight лв</span>/нощувка

</div>

<div class="place-fav">

<i class="far fa-heart"></i>

</div>

<img class="place-img" src="~/images/@place.Images.First().Name" alt="error">

<div class="place-info">

<div class="row">

<div class="col-7 text-left pr-0">

@place.Name <span class="place-year">@place.CategoryName</span>

</div>

<div class="col-5 text-right row p-0">

<div class="col-12">

@for (int i = 0; i < place.Rating; i++)

{

<span class="fa fa-star"></span>

}

@for (int i = place.Rating; i < 5; i++)

{

<span class="fa fa-star-o"></span>

}

</div>

<div class="col-12 pl-0">

<div class="place-city">

<i class="fas fa-map-marker-alt text-primary" style="font-size: 1rem"></i> @place.CityName

</div>

</div>

</div>

</div>

<div class="row mt-2 text-left">

<div class="col-6 pr-0">

<i class="fas fa-user-friends"></i><span class="place-second-info-title"> Места: </span><span class="place-second-info"> @place.MaxGuest </span>

</div>

<div class="col-6 pr-0">

<i class="fas fa-bed"></i><span class="place-second-info-title"> Брой легла: </span><span class="place-second-info"> @place.BedsNum</span>

</div>

</div>

<div class="row mt-1 text-left">

<div class="col-6 pr-0">

<i class="fas fa-bath"></i><span class="place-second-info-title"> Брой бани: </span><span class="place-second-info"> @place.BathroomsNum</span>

</div>

<div class="col-6 pr-0">

<i class="fas fa-address-book"></i><span class="place-second-info-title"> Предишни наематели: </span><span class="place-second-info"> @place.Reservations.Count </span>

</div>

</div>

</div>

</div>

</div>

</div>

</div>

</div>

</div>

<style>

.date-box {

display: inline-block;

height: 60px;

width: 60px;

border-radius: 10px;

background-color: rgb(238, 238, 238);

font-size: 1rem;

}

.date-info {

display: inline-block;

margin-left: 10px;

font-size: 1rem;

}

.icon-item {

margin: 10px 0;

}

.icon-item .custom-icons

{

height: 40px;

width: 40px;

margin-right: 10px;

}

.icon-item .visa-mc-icon {

height: 40px;

}

.can-toggle {

position: relative;

display: inline-block;

}

.can-toggle \*, .can-toggle \*:before, .can-toggle \*:after {

box-sizing: border-box;

}

.can-toggle input[type="checkbox"] {

opacity: 0;

position: absolute;

top: 0;

left: 0;

}

.can-toggle input[type="checkbox"][disabled] ~ label {

pointer-events: none;

}

.can-toggle input[type="checkbox"][disabled] ~ label .can-toggle\_\_switch {

opacity: 0.4;

}

.can-toggle input[type="checkbox"]:checked ~ label .can-toggle\_\_switch:before {

content: attr(data-unchecked);

left: 0;

}

.can-toggle input[type="checkbox"]:checked ~ label .can-toggle\_\_switch:after {

content: attr(data-checked);

}

.can-toggle label {

user-select: none;

position: relative;

display: flex;

align-items: center;

}

.can-toggle label .can-toggle\_\_label-text {

flex: 1;

padding-left: 32px;

}

.can-toggle label .can-toggle\_\_switch {

position: relative;

}

.can-toggle label .can-toggle\_\_switch:before {

content: attr(data-checked);

position: absolute;

top: 0;

text-transform: uppercase;

text-align: center;

}

.can-toggle label .can-toggle\_\_switch:after {

content: attr(data-unchecked);

position: absolute;

z-index: 5;

text-transform: uppercase;

text-align: center;

background: white;

transform: translate3d(0, 0, 0);

}

.can-toggle.demo-rebrand-2 {

top: 50%;

transform: translateY(-50%);

}

.can-toggle.demo-rebrand-2 input[type="checkbox"][disabled] ~ label {

color: rgba(68, 68, 68, 0.5);

}

.can-toggle.demo-rebrand-2 input[type="checkbox"]:focus ~ label .can-toggle\_\_switch, .can-toggle.demo-rebrand-2 input[type="checkbox"]:hover ~ label .can-toggle\_\_switch {

background-color: #444;

}

.can-toggle.demo-rebrand-2 input[type="checkbox"]:focus ~ label .can-toggle\_\_switch:after, .can-toggle.demo-rebrand-2 input[type="checkbox"]:hover ~ label .can-toggle\_\_switch:after {

color: #2b2b2b;

}

.can-toggle.demo-rebrand-2 input[type="checkbox"]:hover ~ label {

color: #373737;

}

.can-toggle.demo-rebrand-2 input[type="checkbox"]:checked ~ label:hover {

color: #2591b1;

}

.can-toggle.demo-rebrand-2 input[type="checkbox"]:checked ~ label .can-toggle\_\_switch {

background-color: #2dacd3;

}

.can-toggle.demo-rebrand-2 input[type="checkbox"]:checked ~ label .can-toggle\_\_switch:after {

color: #248aa9;

}

.can-toggle.demo-rebrand-2 input[type="checkbox"]:checked:focus ~ label .can-toggle\_\_switch, .can-toggle.demo-rebrand-2 input[type="checkbox"]:checked:hover ~ label .can-toggle\_\_switch {

background-color: #289bbe;

}

.can-toggle.demo-rebrand-2 input[type="checkbox"]:checked:focus ~ label .can-toggle\_\_switch:after, .can-toggle.demo-rebrand-2 input[type="checkbox"]:checked:hover ~ label .can-toggle\_\_switch:after {

color: #1f7994;

}

.can-toggle.demo-rebrand-2 label .can-toggle\_\_label-text {

flex: 1;

}

.can-toggle.demo-rebrand-2 label .can-toggle\_\_switch {

transition: background-color 0.3s cubic-bezier(0.86, 0, 0.07, 1);

background: #515151;

}

.can-toggle.demo-rebrand-2 label .can-toggle\_\_switch:before {

color: rgba(255, 255, 255, 0.7);

}

.can-toggle.demo-rebrand-2 label .can-toggle\_\_switch:after {

-webkit-transition: -webkit-transform 0.3s cubic-bezier(0.86, 0, 0.07, 1);

transition: transform 0.3s cubic-bezier(0.86, 0, 0.07, 1);

color: #444;

}

.can-toggle.demo-rebrand-2 input[type="checkbox"]:focus ~ label .can-toggle\_\_switch:after, .can-toggle.demo-rebrand-2 input[type="checkbox"]:hover ~ label .can-toggle\_\_switch:after {

box-shadow: 0 4px 4px rgba(0, 0, 0, 0.4);

}

.can-toggle.demo-rebrand-2 input[type="checkbox"]:checked ~ label .can-toggle\_\_switch:after {

transform: translate3d(25.5px, 0, 0);

}

.can-toggle.demo-rebrand-2 input[type="checkbox"]:checked:focus ~ label .can-toggle\_\_switch:after, .can-toggle.demo-rebrand-2 input[type="checkbox"]:checked:hover ~ label .can-toggle\_\_switch:after {

box-shadow: 0 4px 4px rgba(0, 0, 0, 0.4);

}

.can-toggle.demo-rebrand-2 label {

font-size: 14px;

}

.can-toggle.demo-rebrand-2 label .can-toggle\_\_switch {

height: 30px;

flex: 0 0 55px;

border-radius: 30px;

}

.can-toggle.demo-rebrand-2 label .can-toggle\_\_switch:before {

left: 27.5px;

font-size: 14px;

line-height: 30px;

width: 27.5px;

padding: 0 12px;

}

.can-toggle.demo-rebrand-2 label .can-toggle\_\_switch:after {

top: 2px;

left: 2px;

border-radius: 15px;

width: 25.5px;

line-height: 26px;

font-size: 14px;

}

.can-toggle.demo-rebrand-2 label .can-toggle\_\_switch:hover:after {

box-shadow: 0 4px 4px rgba(0, 0, 0, 0.4);

}

</style>

PlaceReservations.cshtml

@model BookingProject.Web.ViewModels.Reservations.ListReservationViewModel

@{

ViewData["Title"] = "UserReservations";

var culture = new System.Globalization.CultureInfo("bg-BG");

List<BookingProject.Web.ViewModels.Reservations.ReservationViewModel> upcoming = new List<BookingProject.Web.ViewModels.Reservations.ReservationViewModel>();

List<BookingProject.Web.ViewModels.Reservations.ReservationViewModel> previous = new List<BookingProject.Web.ViewModels.Reservations.ReservationViewModel>();

foreach (var res in Model.Reservations)

{

if (res.StartDate > DateTime.Now)

{

upcoming.Add(res);

}

else

{

previous.Add(res);

}

}

}

<div class="container">

<h1 class="text-center">Резервации за @Model.Place.Name</h1>

@if (upcoming.Count == 0 && previous.Count == 0)

{

<h2 class="text-center">Нямате предстоящи резервации за това място</h2>

}

@if (upcoming.Count > 0)

{

<h3>Предстоящи:</h3>

@foreach (var res in Model.Reservations)

{

<div class="col-12 text-center mt-4">

<div class="reservation row">

<div class="col-12 col-lg-6 row">

<img class="place-img col-12 col-md-6 p-0"

src="~/images/@res.Place.Images[0].Name"

alt="" />

<div class="col-12 col-md-6">

<h4>

@res.Place.Name

</h4>

<hr class="m-1" />

@res.Place.User.UserName

<br />

<i class="fas fa-mobile-alt"></i> @res.Place.User.PhoneNumber

</div>

</div>

<div class="col-12 col-lg-6 row text-left">

<div class="col-12 col-md-6 col-lg-8">

<div class="col-12 p-1">

<div class="date-box text-center p-1">

<span class="font-weight-bold h4">

@res.StartDate.Day

</span>

<br class="p-0 m-0" />

@res.StartDate.ToString("MMM", culture)

</div>

<div class="date-info">

Настаняване - @res.StartDate.ToString("dddd", culture)

<br class="p-0 m-0" />

след 15:00 часа

</div>

</div>

<div class="col-12 p-1">

<div class="date-box text-center p-1">

<span class="font-weight-bold h4">

@res.EndDate.Day

</span>

<br class="p-0 m-0" />

@res.EndDate.ToString("MMM", culture)

</div>

<div class="date-info">

Освобождаване - @res.EndDate.ToString("dddd", culture)

<br class="p-0 m-0" />

до 12:00 часа

</div>

</div>

</div>

<div class="col-12 col-md-6 col-lg-4 text-center price">

<p class="m-1">Платено:</p>

<h5>@res.TotalPrice лв</h5>

<hr class="m-1" />

<p class="m-1">Продължителност:</p>

<h5>@res.NumNights дни</h5>

<hr class="m-1" />

</div>

</div>

</div>

</div>

}

}

@if (previous.Count > 0)

{

<h3>Отминали:</h3>

@foreach (var res in Model.Reservations)

{

<div class="col-12 text-center mt-4">

<div class="reservation row">

<div class="col-12 col-lg-6 row">

<img class="place-img col-12 col-md-6 p-0"

src="~/images/@res.Place.Images[0].Name"

alt="" />

<div class="col-12 col-md-6">

<h4>

@res.Place.Name

</h4>

<h5>

@res.Place.CategoryName

</h5>

<hr class="m-1" />

@res.Place.User.UserName

<br />

<i class="fas fa-mobile-alt"></i> @res.Place.User.PhoneNumber

</div>

</div>

<div class="col-12 col-lg-6 row text-left">

<div class="col-12 col-md-6 col-lg-8">

<div class="col-12 p-1">

<div class="date-box text-center p-1">

<span class="font-weight-bold h4">

@res.StartDate.Day

</span>

<br class="p-0 m-0" />

@res.StartDate.ToString("MMM", culture)

</div>

<div class="date-info">

Настаняване - @res.StartDate.ToString("dddd", culture)

<br class="p-0 m-0" />

след 15:00 часа

</div>

</div>

<div class="col-12 p-1">

<div class="date-box text-center p-1">

<span class="font-weight-bold h4">

@res.EndDate.Day

</span>

<br class="p-0 m-0" />

@res.EndDate.ToString("MMM", culture)

</div>

<div class="date-info">

Освобождаване - @res.EndDate.ToString("dddd", culture)

<br class="p-0 m-0" />

до 12:00 часа

</div>

</div>

</div>

<div class="col-12 col-md-6 col-lg-4 text-center price">

<p class="m-1">Платено:</p>

<h5>@res.TotalPrice лв</h5>

<hr class="m-1" />

<p class="m-1">Продължителност:</p>

<h5>@res.NumNights дни</h5>

<hr class="m-1" />

</div>

</div>

</div>

</div>

}

}

</div>

<style>

.reservation {

border-radius: 8px;

background: linear-gradient(to right, #57a6ff, #bbd3ff, white);

}

.date-box {

display: inline-block;

height: 60px;

width: 60px;

border-radius: 10px;

background-color: #eee;

font-size: 1rem;

}

.date-info {

display: inline-block;

margin-left: 10px;

font-size: 0.9rem;

}

.place-img {

width: 100%;

height: 9em;

object-fit: cover;

border-radius: 8px;

cursor: pointer;

}

.owner-img {

height: 30px;

width: 30px;

border-radius: 50%;

}

.price {

font-size: 1rem;

padding: 0;

}

.price .fa-times {

font-size: 1.6rem;

}

.price .fa-star {

font-size: 1.6rem;

color: #f80;

}

.price .pointer {

cursor: pointer;

}

</style>

UserReservations.cshtml

@model BookingProject.Web.ViewModels.Reservations.ListReservationViewModel

@{

ViewData["Title"] = "UserReservations";

var culture = new System.Globalization.CultureInfo("bg-BG");

List<BookingProject.Web.ViewModels.Reservations.ReservationViewModel> upcoming = new List<BookingProject.Web.ViewModels.Reservations.ReservationViewModel>();

List<BookingProject.Web.ViewModels.Reservations.ReservationViewModel> previous = new List<BookingProject.Web.ViewModels.Reservations.ReservationViewModel>();

foreach (var res in Model.Reservations)

{

if (res.StartDate > DateTime.Now)

{

upcoming.Add(res);

}

else

{

previous.Add(res);

}

}

}

<div class="container">

<h1 class="text-center">Резервации за @Model.Place.Name</h1>

@if (upcoming.Count == 0 && previous.Count == 0)

{

<h2 class="text-center">Нямате предстоящи резервации за това място</h2>

}

@if (upcoming.Count > 0)

{

<h3>Предстоящи:</h3>

@foreach (var res in Model.Reservations)

{

<div class="col-12 text-center mt-4">

<div class="reservation row">

<div class="col-12 col-lg-6 row">

<img class="place-img col-12 col-md-6 p-0"

src="~/images/@res.Place.Images[0].Name"

alt="" />

<div class="col-12 col-md-6">

<h4>

@res.Place.Name

</h4>

<hr class="m-1" />

@res.Place.User.UserName

<br />

<i class="fas fa-mobile-alt"></i> @res.Place.User.PhoneNumber

</div>

</div>

<div class="col-12 col-lg-6 row text-left">

<div class="col-12 col-md-6 col-lg-8">

<div class="col-12 p-1">

<div class="date-box text-center p-1">

<span class="font-weight-bold h4">

@res.StartDate.Day

</span>

<br class="p-0 m-0" />

@res.StartDate.ToString("MMM", culture)

</div>

<div class="date-info">

Настаняване - @res.StartDate.ToString("dddd", culture)

<br class="p-0 m-0" />

след 15:00 часа

</div>

</div>

<div class="col-12 p-1">

<div class="date-box text-center p-1">

<span class="font-weight-bold h4">

@res.EndDate.Day

</span>

<br class="p-0 m-0" />

@res.EndDate.ToString("MMM", culture)

</div>

<div class="date-info">

Освобождаване - @res.EndDate.ToString("dddd", culture)

<br class="p-0 m-0" />

до 12:00 часа

</div>

</div>

</div>

<div class="col-12 col-md-6 col-lg-4 text-center price">

<p class="m-1">Платено:</p>

<h5>@res.TotalPrice лв</h5>

<hr class="m-1" />

<p class="m-1">Продължителност:</p>

<h5>@res.NumNights дни</h5>

<hr class="m-1" />

</div>

</div>

</div>

</div>

}

}

@if (previous.Count > 0)

{

<h3>Отминали:</h3>

@foreach (var res in Model.Reservations)

{

<div class="col-12 text-center mt-4">

<div class="reservation row">

<div class="col-12 col-lg-6 row">

<img class="place-img col-12 col-md-6 p-0"

src="~/images/@res.Place.Images[0].Name"

alt="" />

<div class="col-12 col-md-6">

<h4>

@res.Place.Name

</h4>

<h5>

@res.Place.CategoryName

</h5>

<hr class="m-1" />

@res.Place.User.UserName

<br />

<i class="fas fa-mobile-alt"></i> @res.Place.User.PhoneNumber

</div>

</div>

<div class="col-12 col-lg-6 row text-left">

<div class="col-12 col-md-6 col-lg-8">

<div class="col-12 p-1">

<div class="date-box text-center p-1">

<span class="font-weight-bold h4">

@res.StartDate.Day

</span>

<br class="p-0 m-0" />

@res.StartDate.ToString("MMM", culture)

</div>

<div class="date-info">

Настаняване - @res.StartDate.ToString("dddd", culture)

<br class="p-0 m-0" />

след 15:00 часа

</div>

</div>

<div class="col-12 p-1">

<div class="date-box text-center p-1">

<span class="font-weight-bold h4">

@res.EndDate.Day

</span>

<br class="p-0 m-0" />

@res.EndDate.ToString("MMM", culture)

</div>

<div class="date-info">

Освобождаване - @res.EndDate.ToString("dddd", culture)

<br class="p-0 m-0" />

до 12:00 часа

</div>

</div>

</div>

<div class="col-12 col-md-6 col-lg-4 text-center price">

<p class="m-1">Платено:</p>

<h5>@res.TotalPrice лв</h5>

<hr class="m-1" />

<p class="m-1">Продължителност:</p>

<h5>@res.NumNights дни</h5>

<hr class="m-1" />

</div>

</div>

</div>

</div>

}

}

</div>

<style>

.reservation {

border-radius: 8px;

background: linear-gradient(to right, #57a6ff, #bbd3ff, white);

}

.date-box {

display: inline-block;

height: 60px;

width: 60px;

border-radius: 10px;

background-color: #eee;

font-size: 1rem;

}

.date-info {

display: inline-block;

margin-left: 10px;

font-size: 0.9rem;

}

.place-img {

width: 100%;

height: 9em;

object-fit: cover;

border-radius: 8px;

cursor: pointer;

}

.owner-img {

height: 30px;

width: 30px;

border-radius: 50%;

}

.price {

font-size: 1rem;

padding: 0;

}

.price .fa-times {

font-size: 1.6rem;

}

.price .fa-star {

font-size: 1.6rem;

color: #f80;

}

.price .pointer {

cursor: pointer;

}

</style>

* Shared

\_Layout.cshtml

@using BookingProject.Common

<!DOCTYPE html>

<html>

<head>

<meta charset="utf-8" />

<meta name="viewport" content="width=device-width, initial-scale=1.0" />

<title>@this.ViewData["Title"] - @GlobalConstants.SystemName</title>

<environment names="Development">

<link href="~/lib/bootstrap/dist/css/bootstrap.css" rel="stylesheet" asp-append-version="true" />

<link href="~/css/site.css" rel="stylesheet" asp-append-version="true" />

<link href="https://cdn.jsdelivr.net/npm/daterangepicker/daterangepicker.css" rel="stylesheet" asp-append-version="true" />

<!-- Font Awesome -->

<link rel="stylesheet" href="https://use.fontawesome.com/releases/v5.8.2/css/all.css">

<link href="https://fonts.googleapis.com/css2?family=Noto+Sans:ital,wght@0,400;1,400;1,700&display=swap" rel="stylesheet">

<link rel="stylesheet" type="text/css" href="https://stackpath.bootstrapcdn.com/font-awesome/4.7.0/css/font-awesome.min.css">

</environment>

<environment names="Staging,Production">

<link href="~/lib/bootstrap/dist/css/bootstrap.min.css" rel="stylesheet" asp-append-version="true" />

<link href="~/css/site.min.css" rel="stylesheet" asp-append-version="true" />

<!-- Font Awesome -->

<link rel="stylesheet" href="https://use.fontawesome.com/releases/v5.8.2/css/all.css">

<link rel="stylesheet" type="text/css" href="https://stackpath.bootstrapcdn.com/font-awesome/4.7.0/css/font-awesome.min.css">

</environment>

</head>

<body>

<header>

<nav class="navbar navbar-expand-sm navbar-light bg-white border-bottom box-shadow mb-3">

<div class="container">

<a class="navbar-brand" asp-area="" asp-controller="Home" asp-action="Index"><img border="0" alt="" src="~/images/logo.png" width="100" height="100"></a>

<button class="navbar-toggler" type="button" data-toggle="collapse" data-target=".navbar-collapse" aria-controls="navbarSupportedContent"

aria-expanded="false" aria-label="Toggle navigation">

<span class="navbar-toggler-icon"></span>

</button>

<div class="navbar-collapse collapse d-sm-inline-flex flex-sm-row-reverse">

<partial name="\_LoginPartial" />

<ul class="navbar-nav flex-grow-1">

<li class="nav-item">

<a class="nav-link text-dark" asp-area="" asp-controller="Places" asp-action="ShowAllPlaces">Виж всички</a>

</li>

<li class="nav-item">

<a class="nav-link text-dark" asp-controller="Home" asp-action="About">За нас</a>

</li>

</ul>

</div>

</div>

</nav>

</header>

<div class="container">

<partial name="\_CookieConsentPartial" />

<main role="main" class="pb-3">

@this.RenderBody()

</main>

</div>

<footer class="border-top footer text-muted">

<div class="container">

&copy; @DateTime.Now.Year - @GlobalConstants.SystemName - <a asp-area="" asp-controller="Home" asp-action="Privacy">Поверителност</a>

</div>

</footer>

<environment names="Development">

<script src="~/lib/jquery/dist/jquery.js" asp-append-version="true"></script>

<script src="~/lib/jquery-validation/dist/jquery.validate.js" asp-append-version="true"></script>

<script src="~/lib/jquery-validation-unobtrusive/dist/jquery.validate.unobtrusive.js" asp-append-version="true"></script>

<script src="~/lib/bootstrap/dist/js/bootstrap.js" asp-append-version="true"></script>

<script src="https://cdn.jsdelivr.net/momentjs/latest/moment.min.js" asp-append-version="true"></script>

<script src="https://cdn.jsdelivr.net/npm/daterangepicker/daterangepicker.min.js" asp-append-version="true"></script>

<script src="~/js/site.js" asp-append-version="true"></script>

</environment>

<environment names="Staging,Production">

<script src="~/lib/jquery/dist/jquery.min.js" asp-append-version="true"></script>

<script src="~/lib/jquery-validation/dist/jquery.validate.min.js" asp-append-version="true"></script>

<script src="~/lib/jquery-validation-unobtrusive/dist/jquery.validate.unobtrusive.js" asp-append-version="true"></script>

<script src="~/lib/bootstrap/dist/js/bootstrap.min.js" asp-append-version="true"></script>

<script src="https://cdn.jsdelivr.net/momentjs/latest/moment.min.js" asp-append-version="true"></script>

<script src="https://cdn.jsdelivr.net/npm/daterangepicker/daterangepicker.min.js" asp-append-version="true"></script>

<script src="~/js/site.min.js" asp-append-version="true"></script>

</environment>

@this.RenderSection("Scripts", required: false)

</body>

</html>

\_LoginPartial.cshtml

@using BookingProject.Common

@using BookingProject.Data.Models

@using Microsoft.AspNetCore.Identity

@inject SignInManager<ApplicationUser> SignInManager

@inject UserManager<ApplicationUser> UserManager

@{

var user = await this.UserManager.GetUserAsync(this.User);

}

<ul class="navbar-nav">

@if (this.SignInManager.IsSignedIn(this.User))

{

if (user.IsAdmin)

{

<li class="nav-item">

<a class="nav-link text-dark" asp-controller="Places" asp-action="Administration">Администрация</a>

</li>

}

<li>

<div class="btn-group">

<button type="button" class="btn btn-secondary dropdown-toggle" data-toggle="dropdown" aria-haspopup="true" aria-expanded="false">

<i class="fa fa-user fa-fw"></i> Моят профил

</button>

<div class="dropdown-menu">

<a class="btn btn-secondary w-100" role="button" aria-pressed="true" asp-area="" asp-controller="Reservations" asp-action="UserReservations">

<i class="fas fa-book" aria-hidden="true"></i> Резервации

</a>

<a class="btn btn-secondary w-100" role="button" aria-pressed="true" asp-area="" asp-controller="Places" asp-action="Create">

<i class="fa fa-plus-circle" aria-hidden="true"></i> Добави място

</a>

<a class="btn btn-secondary w-100" role="button" aria-pressed="true" asp-area="" asp-controller="Places" asp-action="ShowUserPlaces">

<i class="fas fa-home" aria-hidden="true"></i> Моите места

</a>

<a class="btn btn-secondary w-100" role="button" aria-pressed="true" asp-area="Identity" asp-page="/Account/Manage/Index" title="Manage">

<i class="fas fa-user-cog" aria-hidden="true"></i> Настройки

</a>

</div>

</div>

</li>

<li class="nav-item">

<form class="form-inline" asp-area="Identity" asp-page="/Account/Logout" asp-route-returnUrl="@(this.Url.Action("Index", "Home", new { area = string.Empty }))">

<button type="submit" class="nav-link btn btn-link text-dark"><i class="fa fa-sign-out"></i> Изход</button>

</form>

</li>

}

else

{

<li class="nav-item">

<a class="nav-link text-dark" asp-area="Identity" asp-page="/Account/Register">Регистрация</a>

</li>

<li class="nav-item">

<a class="nav-link text-dark" asp-area="Identity" asp-page="/Account/Login">Вход</a>

</li>

}

</ul>

Program.cs

namespace BookingProject.Web

{

using Microsoft.AspNetCore.Hosting;

using Microsoft.Extensions.Hosting;

public static class Program

{

public static void Main(string[] args)

{

CreateHostBuilder(args).Build().Run();

}

public static IHostBuilder CreateHostBuilder(string[] args) =>

Host.CreateDefaultBuilder(args)

.ConfigureWebHostDefaults(webBuilder =>

{

webBuilder.UseStartup<Startup>();

});

}

}

Startup.cs

namespace BookingProject.Web

{

using System.Reflection;

using BookingProject.Data;

using BookingProject.Data.Common;

using BookingProject.Data.Common.Repositories;

using BookingProject.Data.Models;

using BookingProject.Data.Repositories;

using BookingProject.Data.Seeding;

using BookingProject.Services.Data;

using BookingProject.Services.Data.IServices;

using BookingProject.Services.Mapping;

using BookingProject.Services.Messaging;

using BookingProject.Web.ViewModels;

using Microsoft.AspNetCore.Builder;

using Microsoft.AspNetCore.Hosting;

using Microsoft.AspNetCore.Http;

using Microsoft.EntityFrameworkCore;

using Microsoft.Extensions.Configuration;

using Microsoft.Extensions.DependencyInjection;

using Microsoft.Extensions.Hosting;

public class Startup

{

private readonly IConfiguration configuration;

public Startup(IConfiguration configuration)

{

this.configuration = configuration;

}

// This method gets called by the runtime. Use this method to add services to the container.

public void ConfigureServices(IServiceCollection services)

{

services.AddDbContext<ApplicationDbContext>(

options => options.UseSqlServer(this.configuration.GetConnectionString("DefaultConnection")));

services.AddDefaultIdentity<ApplicationUser>(IdentityOptionsProvider.GetIdentityOptions)

.AddRoles<ApplicationRole>().AddEntityFrameworkStores<ApplicationDbContext>();

services.Configure<CookiePolicyOptions>(

options =>

{

options.CheckConsentNeeded = context => true;

options.MinimumSameSitePolicy = SameSiteMode.None;

});

services.AddControllersWithViews();

services.AddRazorPages();

services.AddSingleton(this.configuration);

// Data repositories

services.AddScoped(typeof(IDeletableEntityRepository<>), typeof(EfDeletableEntityRepository<>));

services.AddScoped(typeof(IRepository<>), typeof(EfRepository<>));

services.AddScoped<IDbQueryRunner, DbQueryRunner>();

// Application services

services.AddTransient<IEmailSender, NullMessageSender>();

services.AddTransient<IPlacesService, PlacesService>();

services.AddTransient<ICitiesService, CitiesService>();

services.AddTransient<ICategoriesService, CategoriesService>();

services.AddTransient<IExtrasService, ExtrasService>();

services.AddTransient<IImagesService, ImagesService>();

services.AddTransient<IReservationsService, ReservationsService>();

services.AddTransient<IReviewsService, ReviewsService>();

}

// This method gets called by the runtime. Use this method to configure the HTTP request pipeline.

public void Configure(IApplicationBuilder app, IWebHostEnvironment env)

{

AutoMapperConfig.RegisterMappings(typeof(ErrorViewModel).GetTypeInfo().Assembly);

// Seed data on application startup

using (var serviceScope = app.ApplicationServices.CreateScope())

{

var dbContext = serviceScope.ServiceProvider.GetRequiredService<ApplicationDbContext>();

if (env.IsDevelopment())

{

dbContext.Database.Migrate();

}

new ApplicationDbContextSeeder().SeedAsync(dbContext, serviceScope.ServiceProvider).GetAwaiter().GetResult();

}

if (env.IsDevelopment())

{

app.UseDeveloperExceptionPage();

app.UseDatabaseErrorPage();

}

else

{

app.UseExceptionHandler("/Home/Error");

app.UseHsts();

}

app.UseHttpsRedirection();

app.UseStaticFiles();

app.UseCookiePolicy();

app.UseRouting();

app.UseAuthentication();

app.UseAuthorization();

app.UseEndpoints(

endpoints =>

{

endpoints.MapControllerRoute("bookingPlace", "b/{name:minlength(3)}", new { controller = "Categories", action = "ByName" });

endpoints.MapControllerRoute("areaRoute", "{area:exists}/{controller=Home}/{action=Index}/{id?}");

endpoints.MapControllerRoute("default", "{controller=Home}/{action=Index}/{id?}");

endpoints.MapRazorPages();

});

}

}

}

Site.js

if (document.querySelector("#rating-value")) {

const stars = document.querySelector(".ratings").children;

const ratingValue = document.querySelector("#rating-value");

let index;

for (let i = 0; i < stars.length; i++) {

stars[i].addEventListener("mouseover", function () {

// console.log(i)

for (let j = 0; j < stars.length; j++) {

stars[j].classList.remove("fa-star");

stars[j].classList.add("fa-star-o");

}

for (let j = 0; j <= i; j++) {

stars[j].classList.remove("fa-star-o");

stars[j].classList.add("fa-star");

}

})

stars[i].addEventListener("click", function () {

ratingValue.value = i + 1;

index = i;

})

stars[i].addEventListener("mouseout", function () {

for (let j = 0; j < stars.length; j++) {

stars[j].classList.remove("fa-star");

stars[j].classList.add("fa-star-o");

}

for (let j = 0; j <= index; j++) {

stars[j].classList.remove("fa-star-o");

stars[j].classList.add("fa-star");

}

})

}

}