



Профилирана природо-математическа гимназия "Никола Обрешков"
гр. Казанлък, ул. "Орешака" 2, тел.:0431/62334, e-mail: info-2400262@edu.mon.bg

ДИПЛОМЕН ПРОЕКТ

**Тема: : РАЗРАБОТКА НА УЕБ ПРИЛОЖЕНИЕ В
ПОМОЩ НА ТУРИСТИ „РЕЗЕРВИРАЙ
ПРЕЖИВЯВАНЕ“**

-

Ученик: Димитър Георгиев Димитров

Професия: код 481030 „Приложен програмист“

Специалност: код 4810301 „Приложно програмиране“

Консултант: Ани Кадиева

Казанлък, 2024 г.

Съдържание

Увод	2
Първа глава	3
1. Проучване.....	3
1.1 Навлизането на уеб приложенията.....	3
1.2 Вече съществуващи подобни уеб приложения	4
1.3 Различното в моето уеб приложение	6
Втора глава	7
2. Проектиране.....	7
2.1 Функционалности на приложението:	7
2.2 Use Case Диаграма	9
2.3 Потребителски интерфейс.....	13
2.4. Избрани технологии и софтуерните средства за разработка на приложението:	26
Трета глава.....	27
3.Програмна реализация	27
3.1 Структура на базата от данни	27
3.2. Структура на MVC приложението	28
Заклучение	52
Използвана литература.....	53
Електронни ресурси	53

Увод

Добре дошли в уеб приложението "Резервирай Преживяване" - вашата вратичка към разнообразието от туристически преживявания в България! Независимо дали сте ентузиаст на туризма, който желае да открие географските и културни богатства на нашата страна, или просто търсите идеално място за отдых и релакс, моето уеб приложение е тук, за да ви помогне да откриете вашето следващо приключение.

"Резервирай Преживяване" е уеб приложение, което предоставя обширна база данни с информация за различни туристически обекти и преживявания в България. То е създадено с цел да улесни потребителите в търсенето и резервирането на места за настаняване и забележителности. С "Резервирай Преживяване" ще имате възможността да създадете незабравими спомени и да откриете красотите на България по начин, който най-добре отговаря на вашите предпочитания и нужди.

Цели на проекта

- Реализиране на добре организирана трислойна архитектура, включваща слой за данни, за бизнес логика и презентационен слой.
- Изпълнение на CRUD операции за управление на информацията за туристическите обекти.
- Създаване на възможност за разглеждане на подробна информация за различни туристически обекти.
- Имплементация на най-съществената функционалност, която включва резервиране на преживявания и организиране на туристически пътувания.
- Реализиране на филтриране по различни критерии за хотелите и къщите за гости.
- Осигуряване на възможност за регистрация и достъп до системата чрез потребителско име и парола, като се разграничават различните роли и техните права.

Задачи на проекта

- Разделяне на уеб приложението на секции за разглеждане на хотелите и къщите за гости и различните туристически и исторически обекти.
- Предоставяне на администратора възможност да добавя, редактира и изтрива хотели и къщи за гости, да управлява информацията за туристическите обекти. Преглежда и управлява регистрираните потребители и техните дейности.
- Осигуряване на интуитивен потребителски интерфейс за лесно и удобно разглеждане на туристическите обекти и местата за настаняване.
- Реализация на филтри за хотели и къщи за гости по лесен и удобен начин за потребителя. Филтрите включват избор на цена за нощувка, локация, вид и оценка на мястото на настаняване.

Първа глава

1. Проучване

1.1 Навлизането на уеб приложенията

В последните години интернетът промени радикално начина, по който потребителите търсят, резервират и планират своите туристически преживявания. С разрастването на технологиите и наличието на интернет достъп, потребителите имат по-голяма възможност да получат информация и да избират между разнообразие от туристически дестинации и услуги.

Удобство и бързина:

Една от основните причини за нарастващото използване на уеб приложения в туризма е тяхното удобство и бързина. Вместо да търсят информация в различни източници или да посещават физически агенции, потребителите могат лесно да се информират и да резервират места за настаняване, екскурзии и други услуги директно от своите компютри или мобилни устройства.

Достъпност:

Уеб приложенията предоставят на потребителите постоянен и лесен достъп до информация за различни туристически обекти в България. Независимо от времето и мястото, потребителите могат да търсят и да сравняват различни възможности за туристически преживявания, което им осигурява по-голяма гъвкавост в планирането на своите пътувания.

Ограничаване на кръга от потребители:

Предприятията в сферата на туризма, които не предлагат онлайн достъп и услуги, рискуват да ограничат своята аудитория. С нарастването на конкуренцията в туристическия бизнес, предприятията, които не развиват онлайн присъствие, могат да загубят потенциални клиенти на конкурентите си, които предлагат по-удобни и достъпни услуги чрез уеб приложения.

В заключение, уеб приложенията играят ключова роля в подобряването на потребителското изживяване и улесняването на процеса на планиране и резервиране на туристически преживявания. Те не само предоставят на потребителите по-голямо разнообразие и удобство, но и допринасят за по-голямата конкурентоспособност на предприятията в туристическия сектор.

1.2 Вече съществуващи подобни уеб приложения

В туристическия сектор вече съществуват различни уеб приложения като Booking.com, Airbnb, Expedia, TripAdvisor и Agoda, които предлагат услуги за резервиране на преживявания, настаняване, екскурзии и други туристически активности.

Предимства на съществуващите уеб приложения:

1. **Лесно използване:** Мнозина от тези приложения предлагат интуитивен интерфейс, който прави процеса на търсене и резервиране на услуги много удобен за потребителите.
2. **Разнообразие от оферти:** Потребителите имат възможност да избират от голямо разнообразие от туристически обекти и преживявания, което им осигурява по-голяма свобода при планирането на своите пътувания.

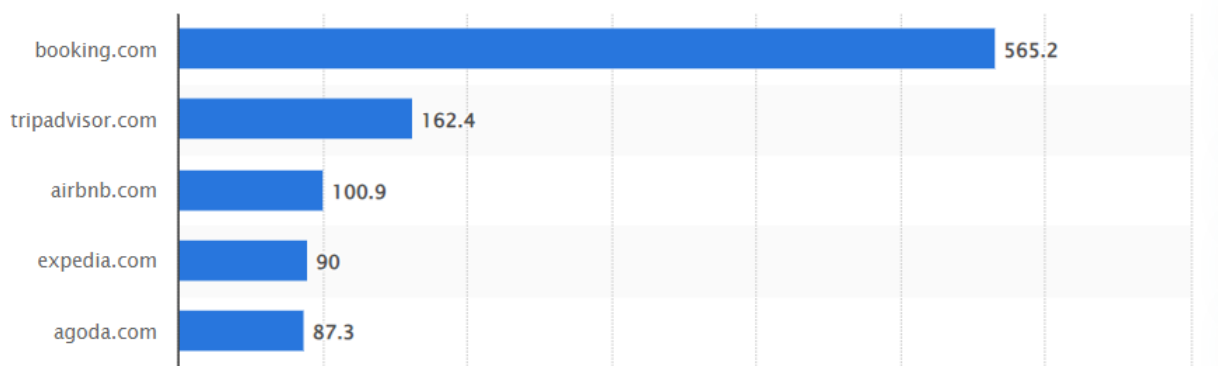
3. **Онлайн плащания:** Мнозина от тези приложения предлагат удобство при плащането, като поддържат различни методи за плащане, включително карти, PayPal и директни банкови трансфери.

Недостатъци на съществуващите уеб приложения:

1. **Ограничена информация:** Някои приложения могат да предоставят ограничена информация за дадените дестинации или обекти, което може да ограничи възможностите за избор на потребителите.
2. **Липса на персонализация:** Въпреки че някои приложения предлагат филтриране на резултатите, липсата на персонализирани препоръки може да затрудни потребителите в намирането на най-подходящите за тях оферти.
3. **Високи такси за обработка:** Някои приложения могат да вземат високи такси за обработка на резервации, което може да направи цената на услугите по-висока за потребителите.

Статистика на пазарния дял

Според последните направени проучвания за януари 2024г., booking.com е най-посещаваният уебсайт за пътувания и туризъм в света. През този месец уеб страницата на Booking записва приблизително 565 милиона посещения. Следват tripadvisor.com и airbnb.com в класацията, с приблизително 162 милиона и 101 милиона посещения, съответно.



Фиг. 1

1.3 Различното в моето уеб приложение

Във уеб приложението "Резервирай Преживяване" има няколко ключови различия от популярните приложения като Booking, TripAdvisor, Airbnb и Agoda:

1. Фокус върху България: Приложението е специализирано в предлагането на информация за туристически обекти и преживявания само в България. Това го прави идеално място за туристи, които търсят специфична информация за туристическите атракции, настаняването и дейностите в България.
2. Подробна информация за географски и културни забележителности: В приложението се съдържа подробна информация за различните географски и културни забележителности в България. Това включва описания, снимки, рейтинги, цени и други важни данни, които помагат на потребителите да вземат информирано решение за тяхното следващо пътешествие.
3. Локално знание и експертност: Приложението предлага локално знание и експертност за туристическите обекти и преживявания в България. Това включва препоръки за най-добрите места за посещение, съвети за пътуване и информация за специфични културни събития и традиции.
4. Персонализирани преживявания: Приложението предлага възможност за персонализирани преживявания, които отговарят на индивидуалните предпочитания и нужди на потребителите. Това включва избор на различни видове настаняване, активности и екскурзии, които могат да бъдат съчетани по уникален начин за създаване на персонализирано пътешествие.

Втора глава

2. Проектиране

2.1 Функционалности на приложението:

В уеб приложението „Резервирай Преживяване“ функционалността е разделена по роли:

- Администратор
- Потребител

Функционалности на администратора:

1. Създаване на почивно място:

- Име
- Вид (хотел или къща за гости)
- Оценка на мястото
- Снимка
- Описание
- Локация
- Удобства на мястото – изброяват се удобствата, които предлага почивното място, като безплатен Wi-Fi, безплатен паркинг, басейн, ресторант, фитнес, спа център и румсървис

2. Създаване на стая в почивното място:

- Вид на стаята (апартамент, двойна стая, двойна делукс стая или студио)
- Капацитет на стаята
- Описание
- Цена за нощувка
- Снимки

3. Изтриване на стая

4. Редактиране на стая

Функционалности на потребителя:

1. Разглеждане на почивните места
2. Филтриране на почивните места:
 - Според името на локацията
 - Според оценката на мястото
 - Според цената за нощувка
 - Според датата за настаняване и напускане
 - Според вида на мястото
 - Подреждане на почивните места според оценката на мястото
3. Премахване на филтрите
4. Извличане на информация за всеки един хотел
5. Добавяне на резервация:
 - Дата на настаняване
 - Дата на напускане
 - Брой на гостите
 - Цялата цена за престоя
 - Видът на стаята

Резервацията се визуализира на отделен екран. Цената за престоя се изчислява след въведените дни за настаняване и напускане. Резервацията се финализира след натискането на бутон „Резервирай“. След финализиране на резервация:

- Визуализира се нов екран, който уведомява потребителя, че му е направена резервация;
 - Резервацията се запазва в две отделни таблици в базата данни на приложението;
6. Разглеждане на географските и историческите обекти
 7. Извличане на информация за всеки един географски или исторически обект
 8. Преглед на направените досега резервации
 9. Редактиране на профила

2.2 Use Case Диаграма



Фиг. 2

Описание на Use Case диаграмата (Фиг. 2)

- Случай на употреба: Търсене на почивни места
 - Цел: Показва всичките почивни места, които са вкарани в базата данни на приложението
 - Успешни постусловия: Визуализира всички въведени почивни места в системата
 - Неуспешни постусловия: Съобщение, че няма налични хотели
 - Категория: CRUD операция
 - Актьори: Потребител
- Случай на употреба: Разглеждане на географски обекти
 - Цел: Показва всичките географски обекти, които са вкарани в базата данни на приложението
 - Успешни постусловия: Визуализира всички налични географски обекти
 - Неуспешни постусловия: Съобщение, че такъв обект не съществува
 - Категория: CRUD операция
 - Актьори: Потребител

- Случай на употреба: Извличане на информация за почивно място
 - Цел: Показва информация за конкретното почивно място като снимки на стаите, описание на хотела и географски обекти в близост до хотела или къщата за гости
 - Успешни постусловия: Визуализира подробната информация за почивното място
 - Неуспешни постусловия: Съобщение, че такова място не съществува
 - Категория: CRUD операция
 - Актьори: Потребител
- Случай на употреба: Извличане на информация за географски обект
 - Цел: Показва снимка и информация за историята на географския обект
 - Успешни постусловия: Визуализира снимката и информацията за географския обект
 - Неуспешни постусловия: Съобщение, че такъв обект не съществува
 - Категория: CRUD операция
 - Актьори: Потребител
- Случай на употреба: Филтриране на почивни места
 - Цел: Дава възможност на потребителя за реализиране на филтриране по различни критерии на почивните места
 - Категория: Филтриране
 - Предусловия: Правилно въведени данни за датите за настаняване и напускане
 - Успешни постусловия: Успешно визуализиране на почивните места според направените филтри
 - Неуспешни постусловия: Съобщава за невалидни данни
 - Актьори: Потребител
- Случай на употреба: Премахване на филтрите
 - Цел: Вече поставените филтри от потребителя биват премахнати и се визуализират всички почивни места въведени в базата данни на приложението
 - Успешни постусловия: Премахва приложените филтри и визуализира всички въведени почивни места в системата
 - Неуспешни постусловия: Съобщение, че няма налични хотели

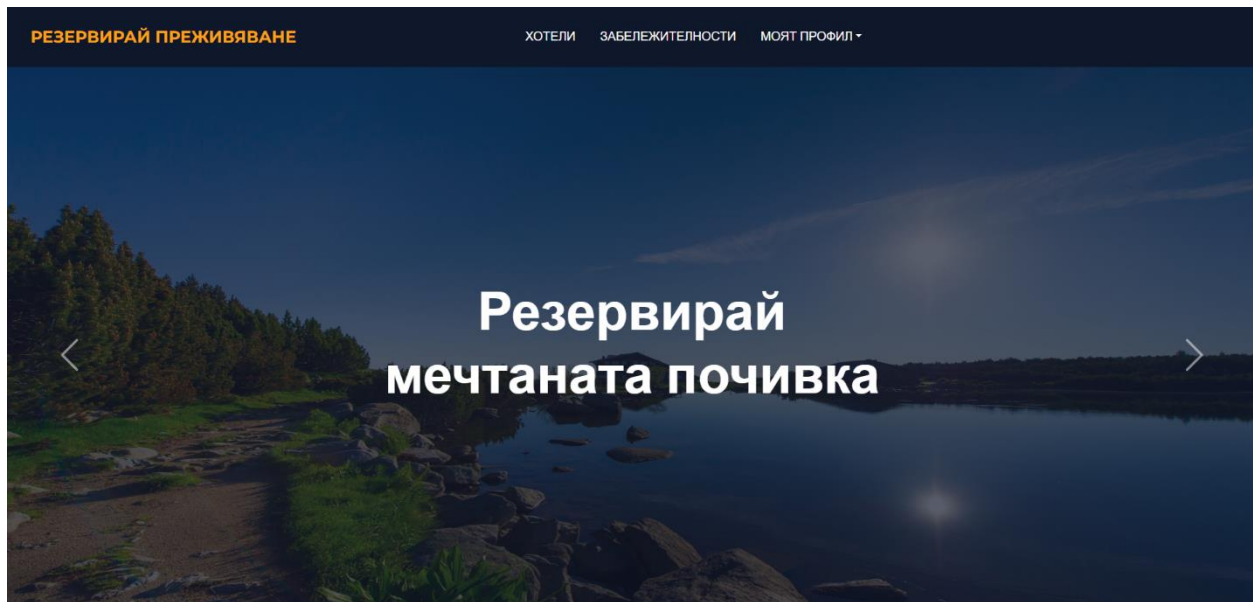
- Категория: Филтриране
- Актьори: Потребител
- Случай на употреба: Регистриране
 - Цел: Дава възможност до повече функционалност в приложението
 - Категория: Система за самоличност
 - Предусловия: Правилно въведени данни за създаването на регистрацията
 - Успешни постусловия: Успешно добавяне на новия потребител в системата
 - Неуспешни постусловия: Съобщава за невалидни данни
 - Актьори: Потребител
- Случай на употреба: Редактиране на профила
 - Цел: Дава възможност на потребителя да промени своята снимка или данни в системата
 - Категория: CRUD операции
 - Предусловия: Правилно въведени данни
 - Успешни постусловия: Успешно се променят данните на потребителя
 - Неуспешни постусловия: Съобщава за невалидни данни
 - Актьори: Потребител
- Случай на употреба: Резервация
 - Цел: Потребителят резервира стая на почивно място за определен от него период
 - Категория: CRUD операции
 - Предусловия: Вход в системата и правилно въведени данни за датата на настаняване и напускане
 - Успешни постусловия: Успешно се добавя резервацията в системата
 - Неуспешни постусловия: Съобщава за невалидни данни или съобщава, че стаята не е налична, понеже вече е направена резервация за тази стая
 - Актьори: Потребител
- Случай на употреба: Преглед на направените резервации
 - Цел: Дава се възможност на потребителя да следи своята дейност в приложението
 - Категория: CRUD операция
 - Предусловия: Вход в системата и направени резервации

- Успешни постусловия: Визуализират се направените резервации от конкретния потребител
- Неуспешни постусловия: Съобщава, че няма направени резервации от този потребител
- Актьори: Потребител
- Случай на употреба: Добавяне на почивно място
 - Цел: Добавяне на ново почивно място, когато има нужда от обновяване на наличните почивни места в системата
 - Категория: CRUD операция
 - Предусловия: Вход в системата с Администраторска роля
 - Успешни постусловия: Успешно добавяне на почивното място в системата
 - Неуспешни постусловия: Съобщава за невалидни данни
 - Актьори: Администратор
- Случай на употреба: Добавяне на стаи
 - Цел: Добавяне на нова стая, когато има нужда от обновяване на наличните стаи в системата
 - Категория: CRUD операция
 - Предусловия: Вход в системата с Администраторска роля
 - Успешни постусловия: Успешно добавяне на стаята в системата
 - Неуспешни постусловия: Съобщава за невалидни данни
 - Актьори: Администратор
- Случай на употреба: Редактиране на стаи
 - Цел: Обновяване на данните за конкретната стая
 - Категория: CRUD операция
 - Предусловия: Вход в системата с Администраторска роля
 - Успешни постусловия: Успешно обновяване на данните на стаята
 - Неуспешни постусловия: Съобщение за невалидни данни
 - Актьори: Администратор
- Случай на употреба: Изтриване на стая
 - Цел: Обновяване на данните в системата
 - Категория: CRUD операция
 - Предусловия: Вход в системата с Администраторска роля

- Успешни постусловия: Успешно премахване на стаята от базата данни на приложението
- Неуспешни постусловия: Съобщение за несъществуваща стая
- Актьори: Администратор

2.3 Потребителски интерфейс

2.3.1 Начална страница на приложението

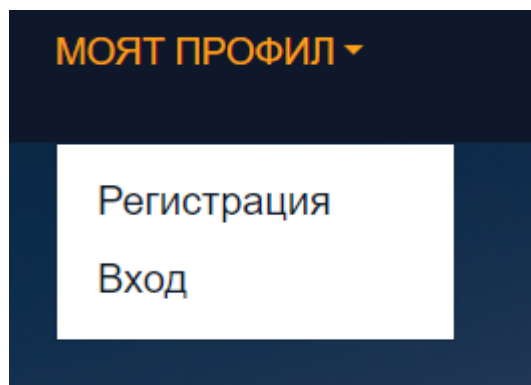


Фиг 3

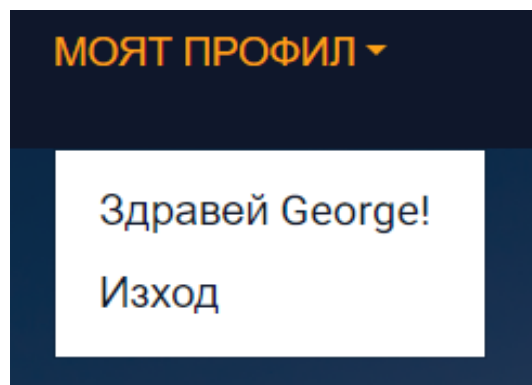
Тази страница (Фиг 3) съдържа навигационно меню, в което има бутони, първите два от които водят към страници, които визуализират всички почивни места и съответно всички географски обекти.

Третият бутон е падащо меню (Фиг. 4), което дава възможност за регистрация или вход в системата, ако такава е вече направена.

След вход в системата падащото меню (Фиг.5) се променя. Вече се дава възможност за преглед на профила на конкретния потребител и изход от системата

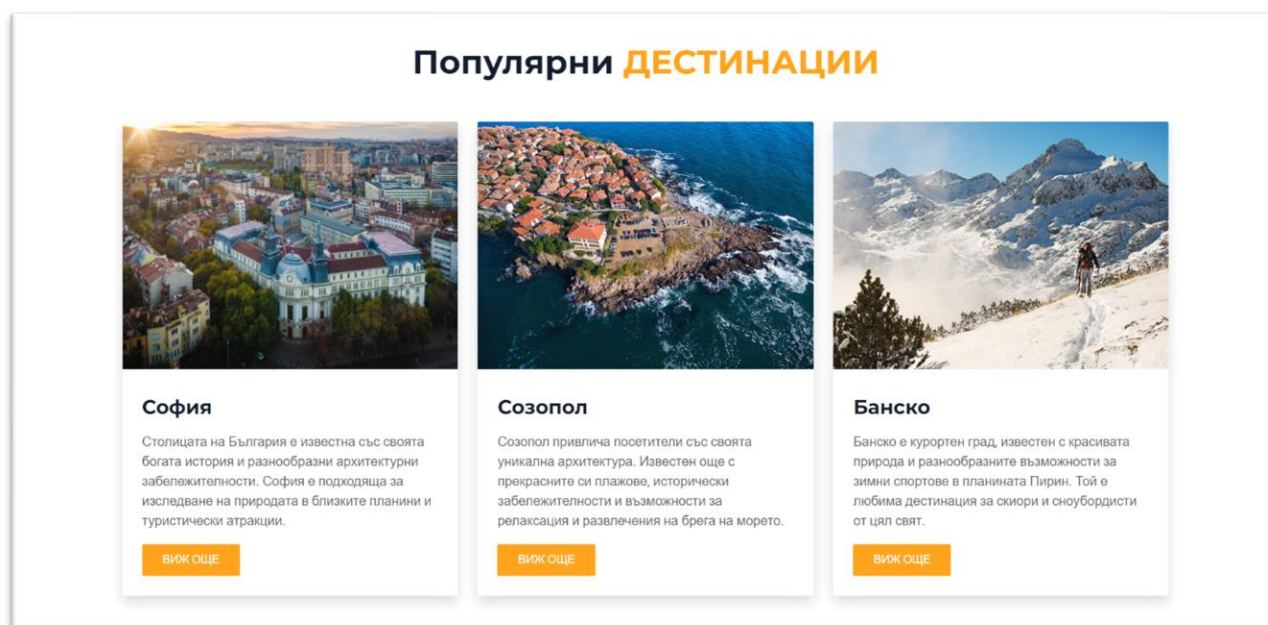


Фиг. 4



Фиг. 5

Началната страница продължава със секция за популярни дестинации, като се предоставя снимка и кратко описание на локацията. За всяка една дестинация има бутон, който отвежда в друга страница, която съдържа хотелите и географските обекти в близост до града



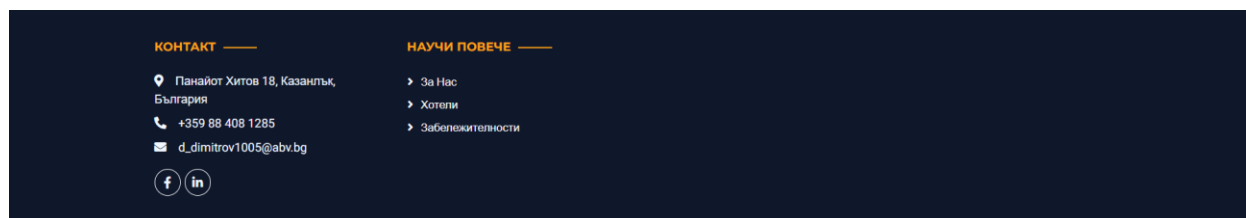
Фиг. 6

Началната страница продължава със секция за едни от най-красивите места в България. При натискането на някоя от снимките, се отваря нова страница с информация за обекта.



Фиг. 7

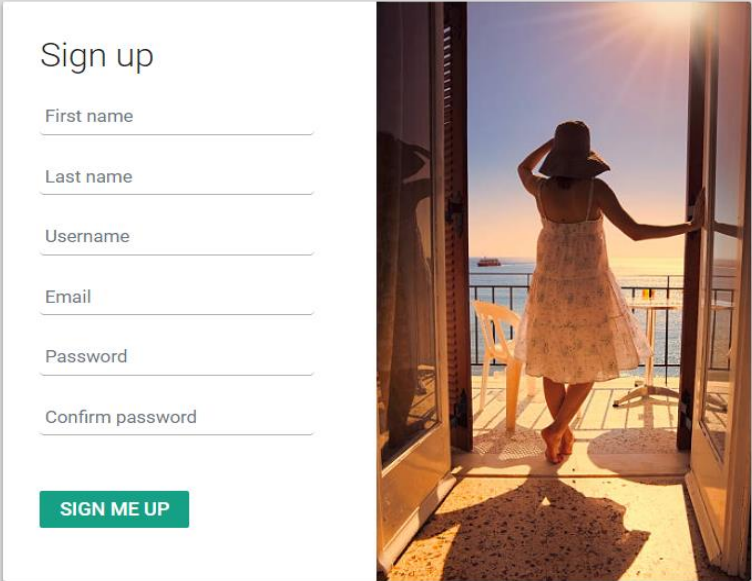
Началната страница завършва с меню за допълнителна информация. То съдържа информация за контакт със собственика на приложението. В графата научи повече има три бутона. Първият отвежда до нова страница, в която се съдържа подробна информация за самото уеб приложение. Другите два бутона отвеждат до всички почивни места и съответно до всички географски обекти.



Фиг. 8

2.3.2 Страница за регистриране

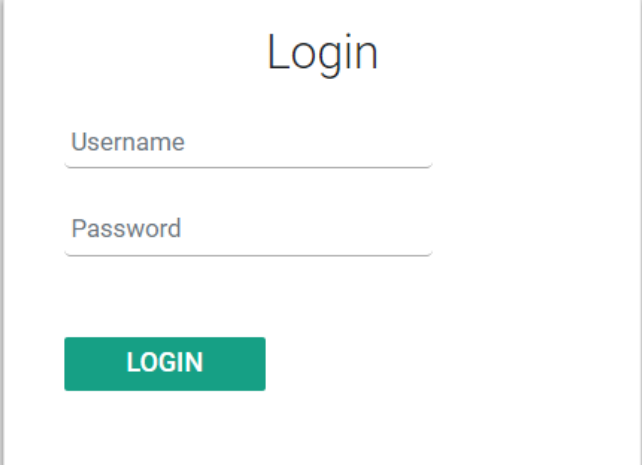
Тази страница (Фиг. 9) съдържа регистрационна форма. Когато потребителят се регистрира от него се изисква да въведе данни за своето собствено и фамилно име, потребителско име, имейл, парола и да потвърди паролата. Когато всички полета са попълнени, потребителя може успешно да се регистрира, като избере бутона “Sign me up”.

The image displays a registration form on the left and a scenic photograph on the right. The form, titled "Sign up", includes input fields for "First name", "Last name", "Username", "Email", "Password", and "Confirm password". A green button labeled "SIGN ME UP" is positioned at the bottom of the form. The photograph shows a woman in a white dress standing on a balcony, looking out at the ocean during a sunset or sunrise.

Фиг. 9

2.3.3 Страница за вход в системата

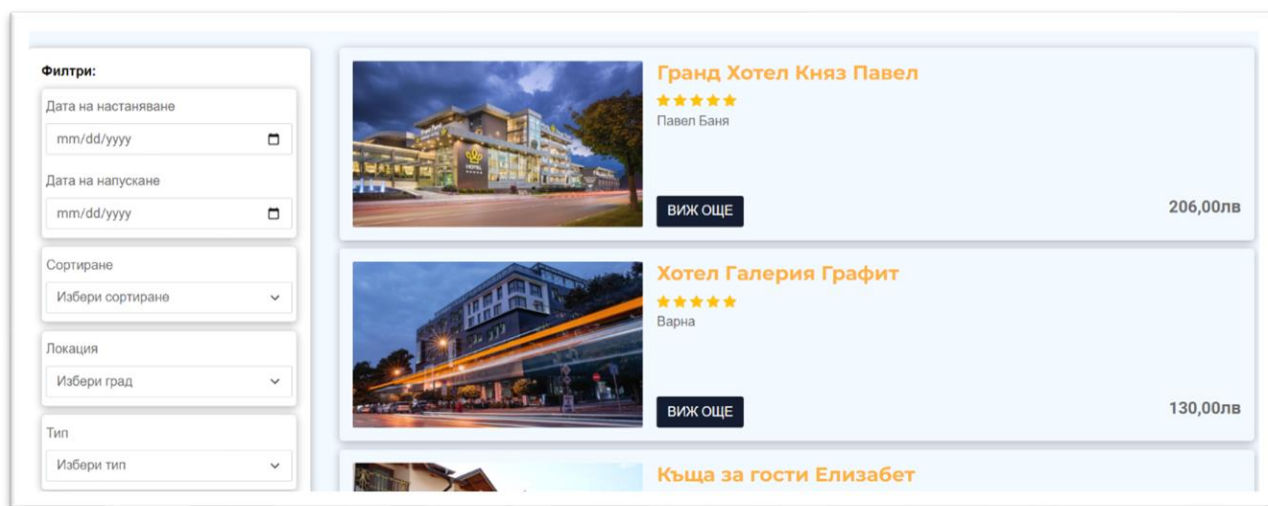
В тази страница (Фиг. 10) се съдържа форма за вход в системата. Когато потребителят е вече регистриран и има акаунт, той може да се идентифицира и да получи достъп до допълнителни функционалности. За да се регистрира, въвежда своето потребителско име и парола и избира бутона “Login”. След успешен вход потребителя бива пренасочен към началната страница. Ако не бъде разпознат, се показва съобщение за невалидни данни.

The image shows a login form centered on a light gray background. The form is a white rectangle with a subtle drop shadow. At the top of the form, the word "Login" is displayed in a large, dark gray, sans-serif font. Below this title, there are two input fields. The first field is labeled "Username" in a small, gray font, with the text positioned to the left of the input line. The second field is labeled "Password" in a similar small, gray font, also to the left of its input line. At the bottom of the form, there is a solid green rectangular button with the word "LOGIN" written in white, uppercase, sans-serif font.

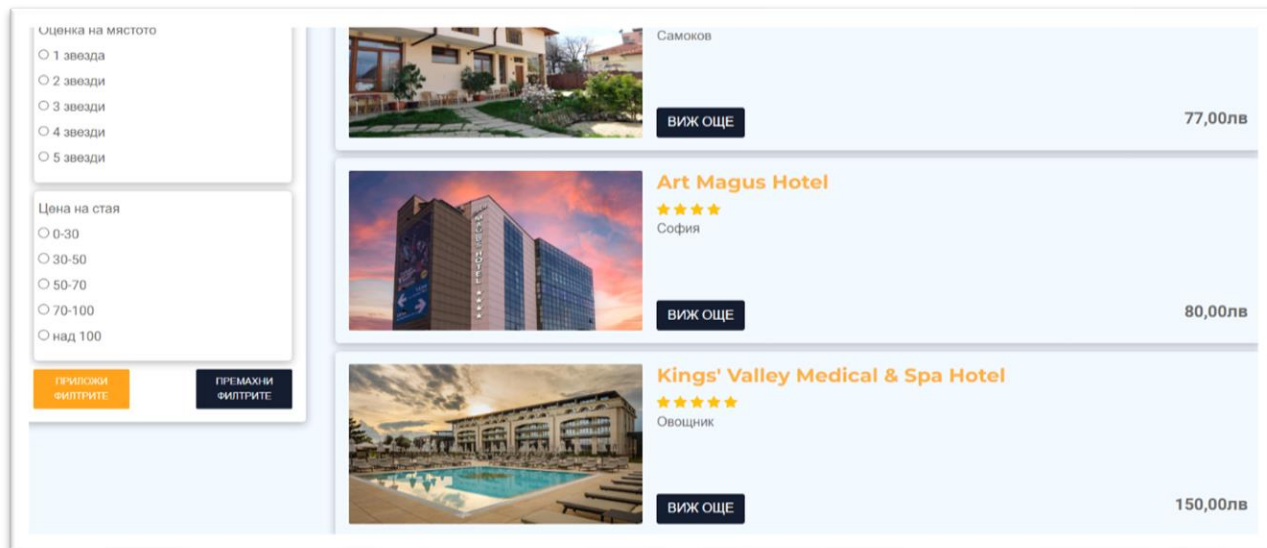
Фиг. 10

2.3.4 Страница за всички почивни места

В тази страница (Фиг. 11) се съдържа форма за всички налични почивни места. За да прегледа тази страница, потребителят не е задължен да е регистриран в системата. За всяко едно почивно място има бутон „ВИЖ ОЩЕ“, който при натискане отваря нова страница с подробна информация за мястото. Освен наличните почивни места, бутон за прилагане на въведените филтри по различни критерии и съответно бутон премахване на приложените филтри (Фиг. 12). След успешно прилагане на филтрите се визуализират почивните места, които отговарят на въведените филтри. Ако няма почивни места, които да съответстват на филтрите се изписва съобщение: „Няма намерени хотели“.



Фиг. 11




Фиг. 12

2.3.5 Страница за всяко едно почивно място

В тази страница (Фиг. 13 и 14) се съдържа форма с подробна информация за почивното място. Това включва описание и удобствата на хотела, снимки и характеристика на стаите, снимки на географските обекти в близост до мястото и бутон „РЕЗЕРВИРАЙ“, който изпраща потребителя до нова страница за направата на резервацията.

Гранд Хотел Княз Павел

★★★★★




Описание


Гранд Хотел Княз Павел се намира в Павел Баня, на 31 км от еко пътека „Бяла река“. Предлага помещения за настаняване с безплатен частен паркинг, открит плувен басейн, фитнес център и градина. Този 5-звезден хотел разполага с тераса и климатизирани стаи с безплатен WiFi и самостоятелна баня. Гостите могат да ползват спа и

Заобикаляща природа

Връх Шипка



Тракийската гробница



Фиг. 13

Удобства

Безплатен WiFi
Безплатен Паркинг
Басейн
Ресторант
Фитнес
Спа
Румсървис

Направи резервация

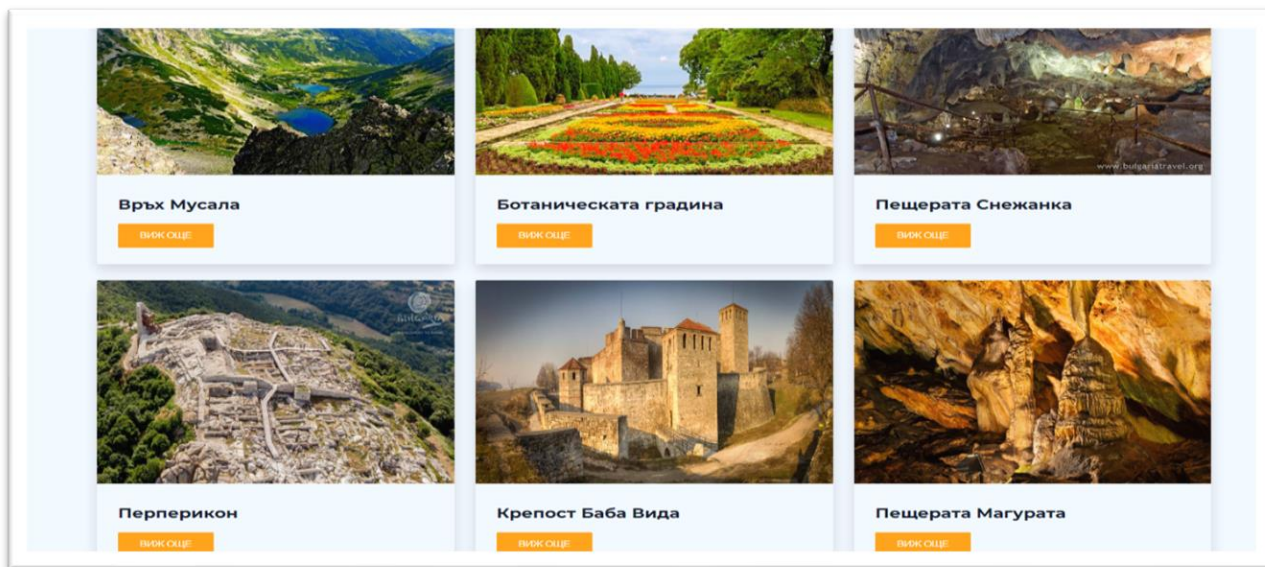
РЕЗЕРВИРАЙ

Тип стаи	Брой гости	Описание	Цена на нощувка
Апартамент	3	Нашият отлично обзаведен Гранд суит осигурява незабравим уют и комфорт. Тази стая разполага с 2 просторни спални, 2 просторни мраморни бани, състоящи се от душ и вана и голям балкон с маса за хранене и няколко люлки. Всички наши стаи разполагат със смарт телевизори с многоезични канали и интерактивни услуги за гости. Интелигентен панел, контролиращ както поддръжката на помещението, така и температурата. Кафе машина Неспресо и хигиенни удобства Омния са осигурени във всяка стая.	509,00 лв.
Двойна делукс стая	2	Нашите луксозно обзаведени двойни стаи предлагат незабравим уют и комфорт. Такава стая разполага с просторна мраморна баня и балкон. Всички наши стаи разполагат със спалня или две отделни легла, смарт телевизори с многоезични канали и интерактивни услуги за гости. Интелигентен панел, контролиращ както поддръжката на помещението, така и температурата. Хигиенни удобства Омния са осигурени във всяка стая.	220,00 лв.
Студио	2	Нашите отлично обзаведени стандартни двойни стаи осигуряват незабравим уют и комфорт. Такава стая разполага с просторна мраморна баня и балкон, две отделни легла, смарт телевизори с многоезични канали и интерактивни услуги за гости. Интелигентен панел, контролиращ както поддръжката на помещението, така и температурата. Хигиенни удобства Омния са осигурени във всяка стая.	220,00 лв.
Двойна стая	2	Нашите отлично обзаведени стандартни двойни стаи осигуряват незабравим уют и комфорт. Такава	206,00 лв.

Фиг. 14

2.3.6 Страница за всички географски обекти

В тази страница (Фиг. 15) се съдържа форма за всички географски обекти въведени в базата данни на приложението. За всеки един обект има бутон „ВИЖ ОЩЕ“, който при натискане отваря нова страница с информация за историята на всеки един обект.



Фиг. 15

2.3.7 Страница за всеки един географски обект


Тази страница (Фиг. 16) съдържа форма със снимка и подробна информация за конкретния географски обект.



Фиг. 16

2.3.8 Страница за резервация

Тази страница (Фиг. 17) съдържа форма със снимки и информация за стаята избрана от потребителя. Освен това има и секция за въвеждане на данни за резервацията. След въвеждането на данните се изписва цялата сума за престоя. Ако резервацията е валидна се показва съобщение за успешно направена резервация, а ако не е валидна – съобщение за невалидни данни. Задължително потребителят трябва да бъде регистриран в системата, за да може да направи резервация.



Апартамент

Нашият отлично обзаведен Гранд суит осигурява незабравим уют и комфорт. Тази стая разполага с 2 просторни спални, 2 просторни мраморни бани, състоящи се от душ и вана и голям балкон с маса за хранене и няколко люлки. Всички наши стаи разполагат със смарт телевизори с многоезични канали и интерактивни услуги за гости. Интелигентен панел, контролиращ както поддръжката на помещението, така и температурата. Кафе машина Неспресо и хигиенни удобства Омния са осигурени във всяка стая.

Направи резервация

Дата на настаняване
04/15/2024

Дата на напускане
04/16/2024

Брой гости
3

509.00 лв.

РЕЗЕРВИРАЙ

Фиг. 17

2.3.9 Страница за редактиране на профил

Тази страница (Фиг. 18) съдържа форма за редактиране на данните на потребителя. Вече регистрираният клиент има възможност да промени своето собствено и фамилно име, своето потребителско име и имейла си. След натискане на бутона „РЕДАКТИРАЙ“ се запазват новите данни в системата и се пренасочва към страницата за профил на потребителя с вече обновената информация.

Моят Профил

Име:	Георги
Фамилия:	Иванов
Потребителско име:	George
Имейл:	george@abv.bg

РЕДАКТИРАЙ

Фиг. 18

2.3.10 Страница за направените резервации за конкретния потребител

Тази страница (Фиг. 19) съдържа информация за резервациите, които е направил потребителя. Формата се състои от колони за периода на почивката, името на почивното място и града, в който се намира. Ако няма направени резервации, изписва съобщение „Няма направени резервации“.

Дата на настаняване	Дата на напускане	Хотел	Град
21.05.2024	22.05.2024	Art Magus Hotel	София
02.05.2024	03.05.2024	Къща за гости Елизабет	Самоков
26.04.2024	27.04.2024	Хотел Галерия Графит	Варна
17.04.2024	19.04.2024	Гранд Хотел Княз Павел	Павел Баня
05.04.2024	06.04.2024	Гранд Хотел Княз Павел	Павел Баня
12.04.2024	13.04.2024	Гранд Хотел Княз Павел	Павел Баня
19.04.2024	20.04.2024	Kings' Valley Medical & Spa Hotel	Овощник

Фиг. 19

2.3.11 Страница за управление на стаите на конкретно почивно място

Тази страница (Фиг. 20) е аналогична на Фиг. 14, като има допълнителни бутони за потребителя с Администраторска роля. Бутонът „ДОБАВИ СТАЯ“ изпраща до нова страница за попълване на данни на новата стая. Бутонът „РЕДАКТИРАЙ“ изпраща до нова страница за обновяване на данните на вече съществуваща стая. Бутонът „ПРЕМАХНИ“ изтрива стаята от базата данни на приложението.

Направи резервация				
РЕЗЕРВИРАЙ		ДОБАВИ СТАЯ		
Тип стаи	Брой гости	Описание	Цена на нощувка	
Апартамент	4	Нашият отлично обзаведен Гранд суит осигурява незабравим уют и комфорт. Тази стая разполага с 2 просторни спални, 2 просторни мраморни бани, състоящи се от душ и вана и голям балкон с маса за хранене и няколко люлки. Всички наши стаи разполагат със смарт телевизори с многоезични канали и интерактивни услуги за гости. Интелигентен панел, контролиращ както поддръжката на помещението, така и температурата. Кафе машина Неспресо и хигиенни удобства Омния са осигурени във всяка стая.	509,00 лв.	ПРЕМАХНИ РЕДАКТИРАЙ
Двойна делукс стая	2	Нашите луксозно обзаведени двойни стаи предлагат незабравим уют и комфорт. Такава стая разполага с просторна мраморна баня и балкон. Всички наши стаи разполагат със спалня или две отделни легла, смарт телевизори с многоезични канали и интерактивни услуги за гости. Интелигентен панел, контролиращ както поддръжката на помещението, така и температурата. Хигиенни удобства Омния са осигурени във всяка стая.	220,00 лв.	ПРЕМАХНИ РЕДАКТИРАЙ

Фиг. 20

2.3.12 Страница за добавяне на почивно място

Тази страница (Фиг. 20) представлява форма, до която има достъп само потребител с Администраторска роля. Тя съдържа текстови полета за име на почивното място, URL за снимката на мястото и описание, както и две падащи менюта за оценка и локация на мястото. Освен това има два радио бутона за вида на мястото (Хотел или Къща за гости) и анкетна кутия (checkbox) за удобствата на мястото (Wi-Fi, Паркинг, Басейн, Ресторант, Фитнес, Спа и Румсървис). При натискане на бутона „ДОБАВЯНЕ“, подадената информация се запазва в базата данни на приложението. Ако някое от полетата е невалидно или празно, съобщава за грешка.

Добавяне на хотел:

Име:

Вид:
☐ Хотел
☐ Къща за гости

Оценка на мястото:

Снимка на хотела:

Описание:

Удобства:
☐ Безплатен WiFi ☐ Безплатен паркинг ☐ Басейн ☐ Ресторант ☐ Фитнес ☐ Спа ☐ Румсървис

Локация:

ДОБАВЯНЕ

Фиг. 21

2.3.13 Страница за редактиране на стая

Тази страница (Фиг 21) представлява форма, до която има достъп само потребител с Администраторска роля. Тя съдържа текстови полета с информацията за конкретна стая. Администраторът има възможност да редактира тези данни, като полето за вид на стаята не може да бъде променено. След натискане на бутона „ЗАПАЗИ ПРОМЕНИТЕ“ , променените данни се запазват в базата данни на приложението. Ако има невалидна информация или празно поле, изписва грешка.

Редактиране на стая

Вид на стаята:

Апартамент

Капацитет на стаята:

4

Описание:

Нашият отлично обзаведен Гранд суит осигурява незабравим уют и комфорт. Тази стая разполага с 2 просторни спални, 2 просторни мраморни бани, състоящи се от душ и вана и голям балкон с маса за хранене и няколко люлки. Всички наши стаи разполагат със смарт телевизори с многоезични канали и интерактивни услуги за гости. Интелигентен панел, контролиращ както поддръжката на помещението, така и температурата. Кафе машина Неспресо и хигиенни удобства Омния са осигурени във всяка стая.

Цена за нощувка:

509,00

ЗАПАЗИ ПРОМЕНИТЕ

Фиг. 22

2.4. Избрани технологии и софтуерните средства за разработка на приложението:

1. Visual Studio - Visual Studio е IDE, в което може да се разработва софтуер на C#. Дава възможност да се създават, обвързват и програмират обекти.

2. Microsoft SQL Server е система за управление на релационни бази данни, разработена от Microsoft.

3. SSMS – SQL Server Management Studio - В SSMS се визуализира базата на приложението.

4. C# и ASP.NET Core - C# е стандартен силно типизиран език за програмиране. ASP.NET Core е framework за изготвяне на уеб приложения. Използвано е в приложението за цялостната разработка и осъществяване на MVC структурата.

5. Entity Framework - Entity Framework помага за създаване на базата по метода Code First.

6. JavaScript - JavaScript е интерпретируем скриптов език от високо ниво, който позволява на разработчиците да създават интерактивни уебстраници. JavaScript се използва за добавяне на специални ефекти към страниците и за показване на данни по интерактивен начин.

7. Bootstrap - Bootstrap е безплатен фреймуърк с отворен код и служи за създаване на интерфейсите на уеб приложения. Това е най-популярната HTML, CSS и JS библиотека за разработване на адаптивни, предназначени на първо място за мобилни устройства уеб интерфейси. Сайтовете изградени с Bootstrap притежават responsive дизайн, това означава че автоматично се мащабират за различните устройствата – независимо дали устройството е мобилен телефон, таблет, лаптоп, настолен компютър, екранен четец и т.н. Използван е в проекта за изработка на VIEWS

8. HTML ни помага да зададем основната структура на уеб страниците. Полезен е за редактиране на параграфи, заглавия и други основни елементи на уеб страница.

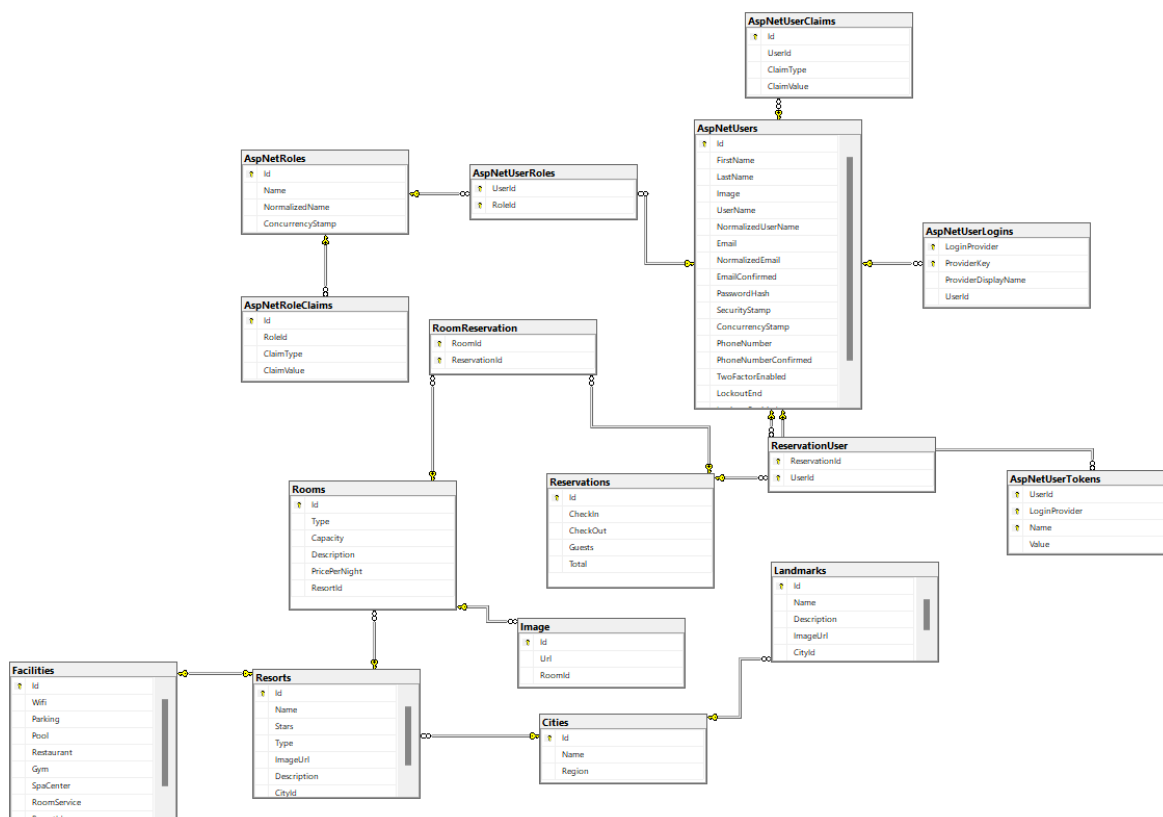
9. CSS- език за описание на стилове, които указват как елементите в една Интернет страница ще бъдат форматирани. С негова помощ стилизираме едновременно всички страници от сайта. Чрез CSS се отделя описанието на съдържанието и структурата на сайтовете от стилизирането им. Всички браузъри поддържат CSS.

10. Cloudinary - API за съхранение на файлове. Там се съхраняват снимките от проекта.

Трета глава

3. Програмна реализация

3.1 Структура на базата от данни



Фиг. 23

Основната таблица е `AspNetUsers`, към която има връзка от таблицата `AspNetUserLogins` чрез полето `UserId`. `UserId` се явява foreign key в таблица `AspNetUserLogins` и така се реализира връзка 1:много. Таблиците се създават от Identity системата, като в основната таблица има допълнително добавени полета (`FirstName`, `LastName`, `Image`).

Таблицата `AspNetUserRoles` се свързва с таблицата `AspNetUsers` чрез полето `UserId`. Таблица `AspNetUserRoles` се свързва и към таблицата `AspNetRoles` чрез полето `RoleId` и така между таблиците `AspNetUsers` и `AspNetRoles` се реализира връзка много: много.

Таблицата `ReservationUser` се свързва чрез полето `UserId` с таблица `AspNetUsers`. Таблица `ReservationUser` се свързва и към таблица `Reservations` чрез полето `ReservationId` и така между таблиците `AspNetUsers` и `Reservations` е реализирана връзка много: много.

Таблицата RoomReservation се свързва чрез полето RoomId с таблица Rooms. Таблица RoomReservation се свързва и към таблица Reservations чрез полето ReservationId и така между таблиците Rooms и Reservations е реализирана връзка много: много.

Таблицата Rooms се свързва с таблицата Resorts чрез полето ResortId. ResortId се явява foreign key в таблицата Rooms и така се реализира връзка 1: много.

Таблицата Image се свързва с таблицата Rooms чрез полето RoomId. RoomId се явява foreign key в таблицата Image и така се реализира връзка 1: много.

Таблицата Resorts се свързва с таблицата Cities чрез полето CityId. CityId се явява foreign key в таблицата Resorts и така се осъществява връзка 1: много. По аналогичен начин се осъществява връзката 1: много между таблиците Landmarks и Cities.

Таблицата Facilities се свързва с таблицата Resorts чрез полето ResortId. ResortId се явява foreign key в таблицата Facilities и така се реализира връзка 1: 1.

3.2. Структура на MVC приложението

MVC е съкратено от Model-view-controller. Представява архитектурен шаблон при програмния дизайн. Той отговаря за разделянето на бизнес логиката на три взаимосвързани части: Model, View и Controller.

- Model –моделите пазят всички данни, с които искаме да работим;
- View – интерфейсът на приложението, който визуализира наличните, обработени данни;
- Controller – съдържат бизнес логиката, която е описана в методи на сървиси, които се извикват в контролера и така обработените данни се подават към изгледите. Контролерът действа като посредник – той комбинира модела с изглед и предоставя резултат на крайния потребител

3.2.1 Описване на CRUD операции

Create операции

Създаване на почивно място

В разработеното приложение има два екшън метода **Create** на почивно място, които служат за изпълнение на първата от четирите CRUD операции – Create. Първият метод се използва за извикване на Get.

```
[HttpGet]
[Authorize(Roles = "Administrator")]
public async Task<IActionResult> Add()
{
    ViewBag.CityId = new SelectList(await service.GetAllCitiesAsync(), "Id", "Name");
    return View();
}
```

Този метод връща изглед:

```
@model Резервирай_Преживяване.Models.ResortViewModels.AddResortViewModel

<form method="post" action="Add" class="mb-5">
  <div class="container">
    <h1>Добавяне на хотел: </h1>
    <div class="mb-3">
      <label class="form-label">Име: </label>
      <input type="text" class="form-control" asp-for="Name" />
      <span asp-validation-for="@Model.Name" class="text-danger"></span>
    </div>

    <div class="mb-3">
      <label class="form-label">Вид: </label><br />
      <input type="radio" name="Type" value="Хотел" />
      <label class="form-label">Хотел</label><br />
      <input type="radio" name="Type" value="Къща за гости" />
      <label class="form-label">Къща за гости</label>
      <span asp-validation-for="@Model.Type" class="text-danger"></span>
    </div>

    <div class="mb-3">
      <label class="form-label">Оценка на мястото: </label>
      <select class="form-select" asp-for="Stars">
        <option value="1">1</option>
        <option value="2">2</option>
        <option value="3">3</option>
        <option value="4">4</option>
        <option value="5">5</option>
      </select>
      <span asp-validation-for="@Model.Stars" class="text-danger"></span>
    </div>

    <div class="mb-3">
      <label class="form-label">Снимка на хотела: </label>
      <input type="text" class="form-control" asp-for="ImageUrl" />
      <span asp-validation-for="@Model.ImageUrl" class="text-danger"></span>
    </div>

    <div class="mb-3">
      <label class="form-label">Описание: </label><br />
      <textarea cols="159" rows="5" class="form-control" asp-
for="Description"></textarea>
```

```

        <span asp-validation-for="@Model.Description" class="text-danger"></span>
    </div>

    <div class="mb-3">
        <label class="form-label">Удобства: </label><br />
        <input type="checkbox" value="true" name="Wifi"/>
        <label class="form-label" style="margin-right: 10px;">Безплатен WiFi</label>
        <input type="checkbox" value="true" name="Parking" />
        <label class="form-label" style="margin-right: 10px;">Безплатен
паркинг</label>
        <input type="checkbox" value="true" name="Pool" />
        <label class="form-label" style="margin-right: 10px;">Басейн</label>
        <input type="checkbox" value="true" name="Restaurant" />
        <label class="form-label" style="margin-right: 10px;">Ресторант</label>
        <input type="checkbox" value="true" name="Gym" />
        <label class="form-label" style="margin-right: 10px;">Фитнес</label>
        <input type="checkbox" value="true" name="SpaCenter" />
        <label class="form-label" style="margin-right: 10px;">Спа</label>
        <input type="checkbox" value="true" name="RoomService" />
        <label class="form-label">Румсървис</label>
    </div>

    <div class="mb-3">
        <label class="form-label">Локация: </label>
        <select class="form-select" asp-for="CityId" asp-items="@ViewBag.CityId">
            <option value="" disabled selected>Избери град</option>
        </select>
        <span asp-validation-for="@Model.CityId" class="text-danger"></span>
    </div>

    <div class="mb-3">
        <button type="submit" class="btn btn-primary">Добавяне</button>
    </div>
</div>

```

Вторият Create метод се използва за извикване на Post заявка. При извикването на този метод заявката взима попълнените във формата данни, праща ги на сървъра и чрез тях създава нов запис в базата данни.

```

[HttpPost]
[Authorize(Roles = "Administrator")]
public async Task<IActionResult> Add(AddResortViewModel model)
{
    if (!ModelState.IsValid)
    {
        return View(model);
    }
    try
    {
        await service.AddAsync(model);
        return RedirectToAction("Index");
    }
    catch (Exception ex)
    {
        ModelState.AddModelError("", ex.Message);
        return View(model);
    }
}

```

Вторият Create метод извиква в себе си сървис, в който са запазени всички операции, които използваме за работата с таблицата Resorts. В този сървис е описан методът AddAsync(), който извършва добавянето на ново почивно място в базата данни.

```
public async Task AddAsync(AddResortViewModel model)
{
    var resort = new Resort()
    {
        Id = Guid.NewGuid(),
        Name = model.Name,
        Type = model.Type,
        Description = model.Description,
        ImageUrl = model.ImageUrl,
        CityId = model.CityId,
        Stars = model.Stars
    };

    var facility = new Facility()
    {
        Id = Guid.NewGuid(),
        Wifi = model.Wifi,
        Parking = model.Parking,
        Pool = model.Pool,
        Restaurant = model.Restaurant,
        Gym = model.Gym,
        SpaCenter = model.SpaCenter,
        RoomService = model.RoomService,
        ResortId = resort.Id,
    };

    await context.Resorts.AddAsync(resort);
    await context.Facilities.AddAsync(facility);
    await context.SaveChangesAsync();
}
```

По аналогичен начин е реализиран Create екшън за добавяне на стая.

Създаване на резервация

В разработеното приложение има два екшън метода **Create** на резервация, които служат за изпълнение на първата от четирите CRUD операции – Create. Първият метод се използва за извикване на Get.

```
[HttpGet]
public async Task<IActionResult> Add(Guid id)
{
    var model = new AddReservationViewModel();
    model.Room = await roomService.RoomToReservateAsync(id);
    return View(model);
}
```

Този метод връща изглед:

```
@model Резервирай_Преживяване.Models.ReservationViewModels.AddReservationViewModel;

<form class="mt-4 mb-5" method="post" action="Add">
    <div class="row">
        <div class="col-10" style="margin-left: auto; margin-right: auto">
            <div id="header-carousel" class="carousel slide" data-bs-ride="carousel">
                <div class="carousel-inner container">
                    <div class="carousel-item active">
                        
                    </div>
                    @if (Model.Room.Images.Count > 1)
                    {
                        for (int i = 1; i < Model.Room.Images.Count; i++)
                        {
                            <div class="carousel-item">
                                
                            </div>
                        }
                    }
                </div>
                <button class="carousel-control-prev" type="button" data-bs-
target="#header-carousel" data-bs-slide="prev">
                    <span class="carousel-control-prev-icon" aria-hidden="true"></span>
                    <span class="visually-hidden">Previous</span>
                </button>
                <button class="carousel-control-next" type="button" data-bs-
target="#header-carousel" data-bs-slide="next">
                    <span class="carousel-control-next-icon" aria-hidden="true"></span>
                    <span class="visually-hidden">Next</span>
                </button>
            </div>
            <div class="col-4" style="margin-left: 165px; margin-top: 10px; padding-left:
50px; padding-right: 50px; padding-top: 10px; background-color: white">
                <h2>@Model.Room.Type</h2>
                <p class="mt-3" style="font-size: 20px">@Model.Room.Description</p>
            </div>
            <div class="col-4 mb-3" style="margin-left: 57px; margin-top: 10px; background-
color: white">
                <h3 style="padding-left: 90px;">Направи резервация</h3>
            </div>
        </div>
    </div>
</form>
```

```

        <div class="mb-4">
            <label class="form-label">Дата на настаняване</label>
            <input type="date" class="form-control" asp-for="@Model.CheckIn" value=""
id="checkInDate" onchange="calculate(@Model.Room!.PricePerNight)" />
            <span asp-validation-for="@Model.CheckIn" class="text-danger"></span>
        </div>

        <div class="mb-4">
            <label class="form-label">Дата на напускане</label>
            <input type="date" class="form-control" asp-for="@Model.CheckOut"
value="" onchange="calculate(@Model.Room!.PricePerNight)" id="checkOutDate" />
            <span asp-validation-for="@Model.CheckOut" class="text-danger"></span>
        </div>

        <div class="mb-5">
            <label class="form-label">Брой гости</label>
            <input class="form-control" asp-for="@Model.Guests" />
            <span asp-validation-for="@Model.Guests" class="text-danger"></span>
        </div>

        <div class="mb-3">
            <label asp-for="Total" class="form-label" style="font-size: 25px"
id="labelTotal">0.00 лв</label>
            <input type="hidden" asp-for="Total" value="" id="inputTotal" />
            <input type="hidden" asp-for="RoomId" value="@Model.Room.Id" />
        </div>

        <div class="mb-3" style="display: flex; flex-direction: row; justify-content:
center">
            <button type="submit" class="btn btn-primary">Резервирай</button>
        </div>
    </div>
</div>
</form>

```

Вторият Create метод се използва за извикване на Post заявка. При извикването на този метод заявката взима попълнените във формата данни, праща ги на сървъра и чрез тях създава нов запис в базата данни.

```

[HttpPost]
public async Task<IActionResult> Add(AddReservationViewModel model)
{
    if (!ModelState.IsValid)
    {
        return View(model);
    }

    try
    {
        var user = await userManager.GetUserAsync(this.User);
        await reservationService.AddReservationAsync(model, user);
        return RedirectToAction("Confirm");
    }
    catch (Exception ex)
    {
        ModelState.AddModelError("", ex.Message);
        return View(model);
    }
}

```

Вторият Create метод извиква в себе си сървис, в който са запазени всички операции, които използваме за работата с таблицата Resorts. В този сървис е описан методът AddAsync(), който извършва добавянето на ново почивно място в базата данни.

```
public async Task AddReservationAsync(AddReservationViewModel model, User user)
{
    if (user == null)
    {
        throw new ArgumentNullException("Невалиден потребител");
    }
    if (model == null)
    {
        throw new ArgumentNullException("Попълни данните");
    }
    if (model.CheckIn < DateTime.Now || model.CheckIn > model.CheckOut || model.CheckOut <
DateTime.Now)
    {
        throw new ArgumentException("Невалидна дата");
    }

    var room = await context.Rooms.Include(x => x.RoomReservations).ThenInclude(x =>
x.Reservation).ThenInclude(x => x!.ReservationUsers).FirstOrDefaultAsync(x => x.Id ==
model.RoomId);
    if (room == null)
    {
        throw new ArgumentNullException("Няма такава стая");
    }
    if (int.Parse(model.Guests!) > int.Parse(room.Capacity!))
    {
        throw new ArgumentException("Капацитетът на стаята е по-малък от въведените гости");
    }

    var totalAsString = model.Total!.Replace('.', ',');
    var total = Convert.ToDecimal(totalAsString);

    if (room.RoomReservations.Count == 0)
    {
        var newReservation = new Reservation()
        {
            Id = Guid.NewGuid(),
            CheckIn = model.CheckIn,
            CheckOut = model.CheckOut,
            Guests = int.Parse(model.Guests!),
            Total = total,
        };

        var roomReservation = new RoomReservation()
        {
            RoomId = room.Id,
            Room = room,
            ReservationId = newReservation.Id,
            Reservation = newReservation,
        };
        room.RoomReservations!.Add(roomReservation);

        var reservationUser = new ReservationUser()
        {
            ReservationId = newReservation.Id,
            Reservation = newReservation,
            UserId = user.Id,
            User = user
        };
    }
}
```

```

        user.ReservationUsers.Add(reservationUser);

        context.Reservations.Add(newReservation);
    }
    else
    {
        foreach (var reservation in room.RoomReservations)
        {
            if ((reservation.Reservation!.CheckIn >= model.CheckIn &&
reservation.Reservation!.CheckOut >= model.CheckIn)
                || (reservation.Reservation!.CheckIn >= model.CheckOut &&
reservation.Reservation!.CheckOut >= model.CheckOut))
            {
                throw new ArgumentException("Стаята не е налична");
            }
            var newReservation = new Reservation()
            {
                Id = Guid.NewGuid(),
                CheckIn = model.CheckIn,
                CheckOut = model.CheckOut,
                Guests = int.Parse(model.Guests!),
                Total = total,
            };

            var roomReservation = new RoomReservation()
            {
                RoomId = room.Id,
                Room = room,
                ReservationId = newReservation.Id,
                Reservation = newReservation,
            };
            room.RoomReservations!.Add(roomReservation);

            var reservationUser = new ReservationUser()
            {
                ReservationId = newReservation.Id,
                Reservation = newReservation,
                UserId = user.Id,
                User = user
            };
            user.ReservationUsers.Add(reservationUser);

            context.Reservations.Add(newReservation);
            break;
        }
    }

    await context.SaveChangesAsync();
}

```

Read операции

Всички почивни места

В разработеното приложение има един екшън метод **Read** на всички почивни места, който служи за изпълнение на втората от четирите CRUD операции – Read. Методът се използва за извикване на Get.

```
[HttpGet]
[AllowAnonymous]
public async Task<IActionResult> Index()
{
    try
    {
        var model = new IndexResortsViewModel();
        model.Resorts = await service.GetAllResortsAsync();
        ViewBag.Cities = new SelectList(await service.GetAllCitiesAsync(), "Id", "Name");
        return View(model);
    }
    catch (Exception ex)
    {
        ModelState.AddModelError("", ex.Message);
        return RedirectToAction("Index", "Home");
    }
}
```

Този метод връща изглед:

```
@model Резервирай_Преживяване.Models.ResortViewModels.IndexResortsViewModel

<form class="mb-3 mt-3" method="get">
    @if (this.User.IsInRole("Administrator"))
    {
        <button asp-controller="Resorts" asp-action="Add" style="margin-left: 20px;"
class="btn btn-primary">Добави хотел</button>
    }
    <div class="row">
        <div class="col-3" style="margin-bottom: 50px;">
            <div class="card" style="margin-left: 20px; margin-top: 20px; box-shadow: 0
2px 10px rgba(0, 0, 0, 0.3); border-radius: 5px;">
                <div class="card-body">
                    <p class="card-title" style="font: 20px; font-weight: bold; color:
black;">Филтри: </p>

                    <div class="card" style="box-shadow: 0 2px 10px rgba(0, 0, 0, 0.3);
border-radius: 5px; margin-bottom: 10px;">
                        <div style="margin-left: 5px; margin-top: 10px; margin-bottom:
10px; margin-right: 5px;">
                            <p class="card-text">Дата на настаняване</p>
                            <input type="date" class="form-control" asp-
for="FilterCheckIn" value="null" />
                        </div>
                        <div style="margin-left: 5px; margin-top: 10px; margin-bottom:
10px; margin-right: 5px;">
                            <p class="card-text">Дата на напускане</p>
                            <input type="date" class="form-control" asp-
for="FilterCheckOut" value="null" />
                        </div>
                    </div>
                <div class="card" style="box-shadow: 0 2px 10px rgba(0, 0, 0, 0.3);
border-radius: 5px; margin-bottom: 10px;">
```

```

<div style="margin-left: 5px; margin-top: 10px; margin-bottom:
10px; margin-right: 5px;">
    <p class="card-text">Сортиране</p>
    <select class="form-select" asp-for="OrderByOption">
        <option value="" disabled selected>Избери
сортиране</option>
        <option value="1">Оценка на мястото</option>
    </select>
</div>
</div>
<div class="card" style="box-shadow: 0 2px 10px rgba(0, 0, 0, 0.3);
border-radius: 5px; margin-bottom: 10px;">
    <div style="margin-left: 5px; margin-top: 10px; margin-bottom:
10px; margin-right: 5px;">
        <p class="card-text">Локация</p>
        <select class="form-select" asp-for="FilterCityName" asp-
items="@ViewBag.Cities">
            <option value="" disabled selected>Избери град</option>
        </select>
    </div>
</div>
<div class="card" style="box-shadow: 0 2px 10px rgba(0, 0, 0, 0.3);
border-radius: 5px; margin-bottom: 10px;">
    <div style="margin-left: 5px; margin-top: 10px; margin-bottom:
10px; margin-right: 5px;">
        <p class="card-text">Тип</p>
        <select class="form-select" asp-for="FilterType">
            <option value="" disabled selected>Избери тип</option>
            <option value="Хотел">Хотели</option>
            <option value="Къща за гости">Къщи за гости</option>
        </select>
    </div>
</div>
<div class="card" style="box-shadow: 0 2px 10px rgba(0, 0, 0, 0.3);
border-radius: 5px; margin-bottom: 10px;">
    <div style="margin-left: 5px; margin-top: 10px; margin-bottom:
10px; margin-right: 5px;">
        <p class="card-text">Оценка на мястото</p>
        <input type="radio" asp-for="FilterStarResort" value="1" />
        <label class="form-label">1 звезда</label><br />
        <input type="radio" asp-for="FilterStarResort" value="2" />
        <label class="form-label">2 звезди</label><br />
        <input type="radio" asp-for="FilterStarResort" value="3" />
        <label class="form-label">3 звезди</label><br />
        <input type="radio" asp-for="FilterStarResort" value="4" />
        <label class="form-label">4 звезди</label><br />
        <input type="radio" asp-for="FilterStarResort" value="5" />
        <label class="form-label">5 звезди</label>
    </div>
</div>
<div class="card" style="box-shadow: 0 2px 10px rgba(0, 0, 0, 0.3);
border-radius: 5px; margin-bottom: 10px;">
    <div style="margin-left: 5px; margin-top: 10px; margin-bottom:
10px; margin-right: 5px;">
        <p class="card-text">Цена на стая</p>
        <input type="radio" asp-for="PricePerNight" value="30" />
        <label class="form-label">0-30</label><br />
        <input type="radio" asp-for="PricePerNight" value="30-50" />
        <label class="form-label">30-50</label><br />
        <input type="radio" asp-for="PricePerNight" value="50-70" />
        <label class="form-label">50-70</label><br />
        <input type="radio" asp-for="PricePerNight" value="70-100" />
        <label class="form-label">70-100</label><br />
        <input type="radio" asp-for="PricePerNight" value="100" />
        <label class="form-label">над 100</label>
    </div>

```

```

        </div>
    </div>
    <div class="row" style="justify-content: space-between">
        <div class="col-5" style="margin-right: 50px">
            <button class="btn btn-primary" style="font-size: 12px" asp-
controller="Resorts" asp-action="Filter">Приложи филтрите</button>
        </div>
        <div class="col-5">
            <button class="btn btn-dark" style="font-size: 12px" asp-
controller="Resorts" asp-action="RemoveFilters">Премахни филтрите</button>
        </div>
    </div>
</div>
</div>
<div class="col-9">
    <div style="margin-bottom: 50px; padding: 10px; padding-right: 20px;">
        @if (Model.Resorts.Any())
        {
            @foreach (var resort in Model.Resorts)
            {
                <div class="hotel" style="margin-top: 10px">
                    <div class="image">
                        
                    </div>
                    <div class="details">
                        <div>
                            <header>
                                <h4 style="margin-left: 95px;"
class="name">@resort.Name</h4>
                            </header>
                            <div style="margin-left: 95px">
                                <for (int i = 0; i < int.Parse(resort.Stars!);
i++)
                                {
                                    <i class="bi bi-star-fill" style="color:
#FFBF00"></i>
                                }
                            </div>
                            <p style="margin-left: 95px"
class="persuasions">@resort?.CityName</p>
                        </div>
                        <div class="info">
                            <h3 class="name"></h3>
                            <div class="facilities">
                                </div>
                        </div>
                        <div class="room">
                            <button asp-controller="Resorts" asp-action="Info"
asp-route-id="@resort?.ResortId" style="margin-left: 95px; border-radius: 3px;" class="btn
btn-dark">Виж още</button>
                            <div>
                                @if (resort!.Rooms.Any())
                                {
                                    <p style="font-weight: bold; font-size:
20px">@resort?.Rooms.Min(x => x.PricePerNight)лв</p>
                                }
                            </div>
                        </div>
                    </div>
                </div>
            }
        }
        else
        {
            <p class="text-center" style="font-size: 30px">Няма намерени
хотели</p>
        }
    </div>

```

Read методът извиква в себе си сървиса, който използвах и в Create метода. В този сървис е описан методът GetAllResortsAsync(), който връща списък от всички почивни места, които са записани в базата данни на приложението.

```
public async Task<List<ResortViewModel>> GetAllResortsAsync()
{
    return await context.Resorts.Include(x => x.City)
        .ThenInclude(x => x!.Landmarks)
        .Include(x => x.Rooms)
        .Include(x => x.Facility)
        .Select(x => new ResortViewModel
        {
            ResortId = x.Id,
            Name = x.Name,
            Stars = x.Stars,
            Type = x.Type,
            ImageUrl = x.ImageUrl,
            Description = x.Description,
            CityName = x.City!.Name,
            CityId = x.CityId,
            Rooms = x.Rooms,
            Landmarks = x.City.Landmarks,
            Facility = x.Facility,
        }).ToListAsync();
}
```

По аналогичен начин е реализиран Read метода за всички географски обекти

Извличане на информация за почивно място

В разработеното приложение има един екшън метод **Read** на всички почивни места, който служи за изпълнение на втората от четирите CRUD операции – Read. Методът се използва за извикване на Get.

```
[HttpGet]
[AllowAnonymous]
public async Task<IActionResult> Info(Guid id)
{
    var model = await service.InfoAsync(id);
    if (model == null)
    {
        return RedirectToAction("Index");
    }
    return await Task.Run(() => View("Info", model));
}
```

Този метод връща изглед:

```
@model Резервирай_Преживяване.Models.ResortViewModels.ResortViewModel;

<form class="mt-5 mb-5">
    <div class="row">
        <div class="col-8" style="padding-left: 100px">
            <h3 style="padding-left: 22px">@Model.Name</h3>
            <div style="padding-left: 22px">
                @for (int i = 0; i < int.Parse(Model.Stars!); i++)
                {
                    <i class="bi bi-star-fill" style="color: #FFBF00"></i>
                }
            </div>
            <div id="header-carousel" class="carousel slide" data-bs-ride="carousel">
                <div class="carousel-inner container">
                    <div class="carousel-item active">
                        
                    </div>
                    @foreach (var room in Model.Rooms)
                    {
                        @foreach (var img in room.Images)
                        {
                            <div class="carousel-item">
                                
                            </div>
                        }
                    }
                </div>
                <button class="carousel-control-prev" type="button" data-bs-
target="#header-carousel" data-bs-slide="prev">
                    <span class="carousel-control-prev-icon" aria-hidden="true"></span>
                    <span class="visually-hidden">Previous</span>
                </button>
                <button class="carousel-control-next" type="button" data-bs-
target="#header-carousel" data-bs-slide="next">
                    <span class="carousel-control-next-icon" aria-hidden="true"></span>
                    <span class="visually-hidden">Next</span>
                </button>
            </div>
        </div>
    </div>
</form>
```

```

<div class="mt-3" style="padding-left: 22px; margin-right: 20px">
  <h3>Описание</h3>
  <p style="font-size: 16px;">@Model.Description</p>
</div>
<div style="padding-left: 22px; margin-right: 20px">
</div>
</div>
<div class="col-4" style="padding-top: 42px; padding-right: 100px">
  <div class="card" style="margin-left: 20px; margin-top: 20px; box-shadow: 0
2px 10px rgba(0, 0, 0, 0.3); border-radius: 5px;">
    <div class="card-body">
      <h4 class="card-title" style="color: black;">Заобикаляща природа</h4>
      @foreach (var landmark in Model.Landmarks)
      {
        <div class="mt-2">
          <a asp-controller="Landmarks" asp-action="EachLandmark" asp-
route-id="@landmark.Id" style="text-decoration: none">
            <p class="card-text">@landmark.Name</p>
            
          </a>
        </div>
      }
    </div>
  </div>
</div>
<div class="mt-4" style="padding-left: 115px">
  <h3>Удобства</h3>
  <div class="row">
    @if (Model.Facility?.Wifi ?? false)
    {
      <div class="col-2">
        <i class="bi bi-wifi" style="color: #FEA116; font-size:
20px"></i>
        <label class="form-label" style="font-size: 20px">Безплатен
WiFi</label>
      </div>
    }

    @if (Model.Facility?.Parking ?? false)
    {
      <div class="col-2" style="margin-left: -30px">
        <i class="fa-solid fa-square-parking" style="color: #FEA116;
font-size: 20px"></i>
        <label class="form-label" style="font-size: 20px">Безплатен
Паркинг</label>
      </div>
    }

    @if (Model.Facility?.Pool ?? false)
    {
      <div class="col-1" style="margin-right: 20px">
        <i class="fa-solid fa-person-swimming" style="color: #FEA116;
font-size: 20px"></i>
        <label class="form-label" style="font-size: 20px">Басейн</label>
      </div>
    }

    @if (Model.Facility?.Restaurant ?? false)
    {
      <div class="col-2">
        <i class="fa-solid fa-utensils" style="color: #FEA116; font-size:
20px"></i>
        <label class="form-label" style="font-size:
20px">Ресторант</label>
      </div>
    }
  }

```

```

@if (Model.Facility?.Gym ?? false)
{
    <div class="col-2" style="margin-left: -90px">
        <i class="fa-solid fa-dumbbell" style="color: #FEA116; font-size:
20px"></i>
        <label class="form-label" style="font-size: 20px">Финтес</label>
    </div>
}

@if (Model.Facility?.SpaCenter ?? false)
{
    <div class="col-2" style="margin-left: -100px">
        <i class="fa-solid fa-spa" style="color: #FEA116; font-size:
20px"></i>
        <label class="form-label" style="font-size: 20px">Спа</label>
    </div>
}

@if (Model.Facility?.RoomService ?? false)
{
    <div class="col-2" style="margin-left: -140px">
        <i class="fa-solid fa-burger" style="color: #FEA116; font-size:
20px"></i>
        <label class="form-label" style="font-size:
20px">Румсървис</label>
    </div>
}
</div>
<div class="mt-4" style="padding-left: 115px; padding-right: 170px">
    <h3>Направи резервация</h3>
    <button asp-controller="Reservations" asp-action="Index" asp-route-
id="@Model.ResortId" class="btn btn-primary">Резервирай</button>
    @if (User.IsInRole("Administrator"))
    {
        <button class="btn btn-primary" style="margin-left: 20px" asp-
controller="Rooms" asp-action="Add" asp-route-id="@Model.ResortId">Добави стая</button>
    }
    @if (Model.Rooms.Any())
    {
        <table class="table">
            <thead>
                <tr>
                    <th style="width: 200px">Тип спаи</th>
                    <th style="width: 110px">Брой гости</th>
                    <th style="width: 700px">Описание</th>
                    <th style="width: 120px">Цена на нощувка</th>
                </tr>
            </thead>
            <tbody>
                @foreach (var room in Model.Rooms)
                {
                    <tr>
                        <td>@room.Type</td>
                        <td>
                            @for (int i = 0; i < int.Parse(room.Capacity!); i++)
                            {
                                <i class="bi bi-person-fill"></i>
                            }
                        </td>
                        <td>@room.Description</td>
                        <td>@room.PricePerNight лв.</td>
                        @if (User.IsInRole("Administrator"))

```

```

        {
            <td>
                <button asp-controller="Rooms" asp-action="Delete"
asp-route-id="@room.Id" class="btn btn-danger" style="font-size: 10px;">Премахни</button>
            </td>
            <td>
                <button asp-controller="Rooms" asp-action="Edit"
asp-route-id="@room.Id" class="btn btn-primary" style="font-size:
10px;">Редактирай</button>
            </td>
        }
    </tr>
}
</tbody>
</table>

}
else
{
    <p class="text-center" style="font-size: 30px;">Няма налични стаи</p>
}
</div>
</div>
</form>

```

Read методът извиква в себе си сървиса, който използвах и в Create метода. В този сървис е описан методът InfoAsync(), който връща модел на изгледа (ViewModel) конкретното почивно място, което се открива по подадено Id.

```

public async Task<ResortViewModel> InfoAsync(Guid id)
{
    var resort = await context.Resorts
        .Include(x => x.City)
        .ThenInclude(x => x!.Landmarks)
        .Include(x => x.Rooms)
        .ThenInclude(x => x.Images)
        .Include(x => x.Facility)
        .FirstOrDefaultAsync(x => x.Id == id);
    if (resort == null)
    {
        throw new ArgumentNullException("Няма такъв хотел");
    }
    var model = new ResortViewModel()
    {
        ResortId = resort.Id,
        Name = resort.Name,
        Stars = resort.Stars,
        Type = resort.Type,
        ImageUrl = resort.ImageUrl,
        Description = resort.Description,
        CityName = resort.City?.Name,
        CityRegion = resort.City?.Region,
        CityId = resort.CityId,
        Rooms = resort.Rooms,
        Facility = resort.Facility,
    };

    var landmarks = context.Landmarks.Include(x => x.City).Where(x => x.City!.Name ==
model.CityRegion).ToHashSet();
    model.Landmarks = landmarks;
    return model;
}

```

По аналогичен начин е реализиран Read метода за извличане на информация за географски обект.

Update операции

Редактиране на стая

В разработеното приложение има два екшън метода **Update** на стая, които служат за изпълнение на третата от четирите CRUD операции – Read. Първият метод се използва за извикване на Get.

```
[HttpGet]
[Authorize(Roles = "Administrator")]
public async Task<IActionResult> Edit(Guid id)
{
    var room = await service.RoomToEditAsync(id);
    return View(room);
}
```

Този метод връща изглед:

```
@model Резервирай_Преживяване.Models.RoomViewModels.RoomViewModel;

<form method="post" action="Edit" class="mt-5 mb-5">
    <div class="container">
        <h1>Редактиране на стая</h1>
        <div class="mb-3">
            <label class="form-label">Вид на стаята:</label><br />
            <input class="form-control" asp-for="Type" readonly/>
        </div>
        <div class="mb-3">
            <label class="form-label">Капацитет на стаята:</label>
            <input type="text" class="form-control" asp-for="Capacity" />
            <span asp-validation-for="@Model.Capacity" class="text-danger"></span>
        </div>
        <div class="mb-3">
            <label class="form-label">Описание: </label><br />
            <textarea cols="159" rows="5" class="form-control" asp-
for="Description"></textarea>
            <span asp-validation-for="@Model.Description" class="text-danger"></span>
        </div>
        <div class="mb-3">
            <label class="form-label">Цена за нощувка:</label>
            <input type="text" class="form-control" asp-for="PricePerNight" />
            <span asp-validation-for="@Model.PricePerNight" class="text-danger"></span>
        </div>
        <input type="hidden" asp-for="Id" value="@Model.Id" />

        <div class="row mb-3">
            <div class="col-2">
                <div>
                    <button type="submit" class="btn btn-primary">Запази
промените</button>
                </div>
            </div>
        </div>
    </div>
</form>
```

Вторият Update метод се използва за извикване на Post заявка. При извикването на този метод заявката взима попълнените във формата данни, праща ги на сървъра и чрез тях обновява записа в базата данни.

```
[HttpGet]
[Authorize(Roles = "Administrator")]
public async Task<IActionResult> Edit(Guid id)
{
    var room = await service.RoomToEditAsync(id);
    return View(room);
}

[HttpPost]
[Authorize(Roles = "Administrator")]
public async Task<IActionResult> Edit(RoomViewModel model)
{
    if (!ModelState.IsValid)
    {
        return View(model);
    }
    try
    {
        var resortId = await service.GetResortIdByGivenRoomId(model.Id);
        await service.EditAsync(model);

        return RedirectToAction("Info", "Resorts", new { id = resortId });
    }
    catch (Exception ex)
    {
        ModelState.AddModelError("", ex.Message);
        return View("Index", model);
    }
}
```

Вторият Update метод извиква в себе си сървис, в който са запазени всички операции, които използваме за работата с таблицата Rooms. В този сървис е описан методът EditAsync(), който извършва обновяването на стаята в базата данни.

```
public async Task EditAsync(RoomViewModel model)
{
    var room = await context.Rooms.FindAsync(model.Id);

    if (room == null)
    {
        throw new ArgumentNullException("Няма такава стая");
    }

    room.Capacity = model.Capacity;
    room.Description = model.Description;
    room.PricePerNight = model.PricePerNight;

    await context.SaveChangesAsync();
}
```

По аналогичен начин се осъществява редактирането на профила на потребителя.

Delete операции

Изтриване на стая

В разработеното приложение има един екшън метод **Delete** на стая, който служи за изпълнение на четвъртата от четирите CRUD операции – Delete. Методът се използва за извикване на Delete заявка.

```
[Authorize(Roles = "Administrator")]
public async Task<IActionResult> Delete(Guid id)
{
    try
    {
        var resortId = await service.GetResortIdByGivenRoomId(id);
        await service.DeleteRoomAsync(id);

        return RedirectToAction("Info", "Resorts", new { id = resortId });
    }
    catch (Exception ex)
    {
        ModelState.AddModelError("", ex.Message);
        return RedirectToAction("Index", "Home");
    }
}
```

Delete методът извиква в себе си сървис, който използвах и в Update метода. В този сървис е описан методът DeleteRoomAsync(), който изтрива стаята от базата данни, като стаята се открива по подадено Id.

```
public async Task DeleteRoomAsync(Guid id)
{
    var room = await context.Rooms.FirstOrDefaultAsync(x => x.Id == id);
    if (room == null)
    {
        throw new ArgumentNullException("Няма такава стая");
    }
    context.Rooms.Remove(room);
    await context.SaveChangesAsync();
}
```

3.2.2 Филтриране

В моето уеб приложение за резервации "Резервирай Преживяване" предлагам множество възможности за филтриране и сортиране на местата за настаняване, които да удовлетворят нуждите на нашите потребители.

```
public async Task<List<ResortViewModel>> FilterResortsAsync(IndexResortsViewModel model)
{
    var location = model.FilterCityName;
    var stars = model.FilterStarResort;
    var prices = model?.PricePerNight?.Split('-').ToList();
    var orderByOption = model?.OrderByOption;
    var type = model?.FilterType;
    var checkIn = model?.FilterCheckIn;
    var checkOut = model?.FilterCheckOut;
    var resorts = await context.Resorts
        .Include(x => x.City)
        .ThenInclude(x => x.Landmarks)
        .Include(x => x.Rooms)
        .ThenInclude(x => x.RoomReservations)
        .ThenInclude(x => x.Reservation)
        .Include(x => x.Facility)
        .Select(x => new ResortViewModel
        {
            ResortId = x.Id,
            Name = x.Name,
            Stars = x.Stars,
            Type = x.Type,
            ImageUrl = x.ImageUrl,
            Description = x.Description,
            CityName = x.City!.Name,
            CityId = x.CityId,
            Rooms = x.Rooms,
            Landmarks = x.City.Landmarks,
            Facility = x.Facility,
        }).ToListAsync();
}
```


1. **Локация:** Потребителите могат да филтрират местата за настаняване по конкретна локация.

```
if (location != null)
{
    resorts = resorts.Where(x => x.CityId.ToString() == location).ToList();
}
```

2. **Оценка на мястото:** Потребителите могат да използват филтъра за оценка, за да намерят почивни места с желания от тях рейтинг.

```
if (stars != null)
{
    resorts = resorts.Where(x => x.Stars == stars).ToList();
}
```

3. **Цена на нощувката:** Потребителите могат да избират почивни места в съответствие с техния бюджет, като филтрират по цена на нощувка.

```
if (prices != null)
{
    if (prices?.Count == 1)
    {
        if (prices[0] == "30")
        {
            resorts = resorts.Where(x => x.Rooms.Min(y => y.PricePerNight) <=
30).ToList();
        }
        else
        {
            resorts = resorts.Where(x => x.Rooms.Min(y => y.PricePerNight) >
100).ToList();
        }
    }
    else
    {
        foreach (var resort in resorts)
        {
            if (prices?[0] == "30" && prices?[1] == "50")
            {
                resorts = resorts.Where(x => x.Rooms.Min(y => y.PricePerNight) > 30 &&
x.Rooms.Min(y => y.PricePerNight) <= 50).ToList();
            }
            else if (prices?[0] == "50" && prices?[1] == "70")
            {
                resorts = resorts.Where(x => x.Rooms.Min(y => y.PricePerNight) > 50 &&
x.Rooms.Min(y => y.PricePerNight) <= 70).ToList();
            }
            else
            {
                resorts = resorts.Where(x => x.Rooms.Min(y => y.PricePerNight) > 70 &&
x.Rooms.Min(y => y.PricePerNight) <= 100).ToList();
            }
        }
    }
}
```

4. **Вид на почивното място:** Потребителите могат да филтрират според вид на почивното място, включително хотели и къщи за гости. Това позволява на потребителите да намерят точно този тип настаняване, който отговаря на техните предпочитания и нужди.

```
if (type != null)
{
    resorts = resorts.Where(x => x.Type == type).ToList();
}
```

5. **Дата на настаняване и напускане:** Потребителите могат да зададат желаните дати на престой, което помага да се ограничи търсенето само до места, които са свободни през определен период.

```
if (checkIn != null && checkOut != null)
{
    if (checkIn < DateTime.Now || checkIn > checkOut || checkOut < DateTime.Now)
    {
        throw new ArgumentException("Невалидна дата");
    }
    resorts = resorts.Where(x => x.Rooms.Any(x => x.RoomReservations.Any(x =>
(x.Reservation!.CheckIn < checkIn && x.Reservation!.CheckOut < checkIn)
|| (x.Reservation!.CheckIn < checkOut && x.Reservation!.CheckOut <
checkOut)))) || x.Rooms.Any(x => x.RoomReservations.Count == 0)).ToList();
}
```

6. **Сортиране по оценка на мястото:** Потребителите имат възможност да сортират резултатите от търсенето по оценка на мястото, за да видят първо най-добре оценените места.

```
resorts = orderByOption switch
{
    OrderBy.Stars => resorts.OrderByDescending(x => x.Stars).ToList(),
    _ => resorts
};
```

Тези опции за филтриране и сортиране на места за настаняване в "Резервирай Преживяване" позволяват на потребителите да намерят точно това, което търсят, и да направят своето престой по-приятен и комфортен.

Премахване на филтрите

В моето приложение използвам сървис за премахването на филтрите, който съдържа методът `RemoveFiltersAsync()`, който връща списък от модел на изгледа (ViewModel) за всички налични почивни места.

```
public async Task<IndexResortsViewModel> RemoveFiltersAsync()
{
    var model = new IndexResortsViewModel();
    model.Resorts = await context.Resorts
        .Include(x => x.City)
        .ThenInclude(x => x!.Landmarks)
        .Include(x => x.Rooms)
        .ThenInclude(x => x.RoomReservations)
        .ThenInclude(x => x.Reservation)
        .Select(x => new ResortViewModel
        {
            ResortId = x.Id,
            Name = x.Name,
            Stars = x.Stars,
            Type = x.Type,
            ImageUrl = x.ImageUrl,
            Description = x.Description,
            CityName = x.City!.Name,
            CityId = x.CityId,
            Rooms = x.Rooms,
            Landmarks = x.City.Landmarks,
            Facility = x.Facility,
        }).ToListAsync();
    return model;
}
```

3.2.3 Ауторизация и оторизация на потребителите на

- **Регистрация**

При регистрация в моето уеб приложение „Резервирай Преживяване“, потребителите ще трябва да въведат следните данни:

1. FirstName (Собствено име)
2. LastName (Фамилно име)
3. Email (Електронна поща)
4. Username (Потребителско име) – потребителското име ще се използва за вход в системата, затова то трябва да бъде уникално
5. Password (Парола) – паролата трябва да съдържа поне 6 знака, като задължително трябва да има една малка и една голяма буква, една цифра и един специален символ
6. ConfirmPassword (Потвърди паролата) – потребителят отново трябва да въведе своята парола отново за потвърждение

След като потребителите въведат тези данни, те ще могат успешно да създадат свой акаунт в приложението и да получат достъп до допълнителната функционалност.

- **Вход/Изход**

При вход в приложението, потребителят трябва да въведе данни за следните полета:

1. Username (Потребителско име) – потребителското име е уникално за всеки потребител и затова се ползва при вход в системата
2. Password (Парола) – потребителят трябва да въведе своята парола, която е създал по време на регистрацията

След вход в системата потребителят получава достъп до страниците за резервация и може да подготвя своите пътувания за предстоящите си преживявания

Ако потребителя реши да излезе от системата, той ще бъде пренасочен към началната страница. Той ще може да разглежда приложението, но ще загуби част от функционалността, която би му помогнала за реализация на резервациите.

- **Роли**

В приложението има две роли. Те са администратор и потребител, като всяка от тях има различни права. Кодът, чрез който се създават ролите е:

```
private HashSet<IdentityRole> CreateRole()
{
    return new HashSet<IdentityRole>
    {
        new()
        {
            Id = "528726ea-e421-4a80-b303-f035355599de",
            Name = "Administrator",
            NormalizedName= "ADMINISTRATOR",
        },
        new()
        {
            Id = "5dd65fa9-eb2c-4372-8084-8c501347e74f",
            Name = "User",
            NormalizedName= "USER",
        }
    };
}
```

Заклучение

Разработеното WEB приложение, покрива напълно дейностите за сайт за резервации, поддържа CRUD операции за почивните места и осигурява на потребителите възможност да разглеждат филтрирани резултати за почивни места и да резервират наличните места. Допълнително е реализирано и ограничаване на достъпа до функционалности за нерегистрирани потребители, възможност за разглеждане на географски обекти в България, както и преглед на всички направени резервации от регистрирания потребител.

Идеите ми за бъдеща разработка са добавяне на функционалности за интеграция със социални мрежи и разширение на базата данни с още туристически обекти и дестинации.

Използвана литература

1. Наков С., Колев В. и колектив, Въведение в със С#, София, 2015, ISBN 978-954- 400-527-6

Електронни ресурси

1. <https://www.w3schools.com/>
2. <https://stackoverflow.com/>
3. <https://www.geeksforgeeks.org/>
4. <https://getbootstrap.com/>