Създаване на ASP .NET Core MVC Уеб приложение за управление на хотелски резервации

Автори: Атанас Георгиев и Даниел Димов

Email и Password за Admin:

Email: admin@hotelreservations.com

Password: Admin123!

Цели

Целта на проекта **HotelReservationsManager** е създаване на уеб базирана система за **управление на хотелски резервации**, която включва:

* Управление на клиенти (гости)
* Управление на хотелски стаи
* Създаване и обработка на резервации
* Администрация и управление на потребителски достъп (роли: администратор, служител)

Основни етапи в реализирането на проекта

Проектът е реализиран през следните основни етапи, разделени между Атанас Георгиев и Даниел Димов:

**1. Създаване на Модели и База Данни**

**Атанас Георгиев:** Създаване на моделите Room, Reservation, Client, User и **ReservationClient**(която е помощна таблица за да се осъществява връзка **много към много** между таблиците Reservation и Client) и настройка на Entity Framework Core и създаване на релации между моделите – един клиент може да има множество резервации; всяка резервация се свързва със стая.

**2. Разработка на Контролери**

**Даниел Димов:** Създаване на контролери за управлението на стаи и резервации (RoomController, ReservationController) и контролери за управление на потребители и клиенти; внедряване на бизнес логика за резервации.

**3.** **Изгледи и Потребителски Интерфейс**

**Даниел Димов:** Отговорен за създаването и конфигурирането на razor изгледи за създаване и редакция на резервации, регистрация и вход.

**Атанас Георгиев:** Подпомогнал с динамична визуализация на заети/свободни стаи, списъци с резервации, използване на layout с меню за навигация.

**4. Автентикация и Авторизация**

Даниел Димов: Използване на ASP.NET Core Identity, включваща вход и управление на роли(Admin и User).

Атанас Георгиев: Подпомогнал при oграничения на достъп до действия според ролите

**5. Тестване и Оптимизация**

Съвместно тестване, фиксиране на грешки, работа с кеширане и логика за подобряване работата на приложението

Ниво на сложност на проекта

Проектът представлява средно до високо сложно приложение, създадено с ASP.NET Core MVC. Основните проблеми, които сме срещнали по време на реализацията му, включват:

Конфигурирането и използването на Entity Framework Core за работа с базата данни. Изграждането на релациите между различните модели, настройката на миграции и синхронизирането на схемата на базата данни с бизнес логиката изискваха задълбочено разбиране на ORM принципите и внимателно структуриране на кода. Особено предизвикателство беше управлението на сложни заявки и оптимизирането на производителността при работа с по-големи обеми от данни.

Предизвикателство беше създаването на динамични изгледи с Razor и правилното свързване на front-end логиката с контролерите. Наложи се ръчно управление на състоянията на формите и визуалната обратна връзка към потребителя при различни действия.

Тези две задачи бяха най-сложните части от проекта поради необходимостта от точна координация между различни технологии и архитектури.

Логическо и функционално описание на решението

Архитектурна Структура

Приложението използва трислойна архитектура:

1. Presentation Layer (Представителен Слой - View): Razor изгледи за работа със стаи, клиенти и резервации.
2. Business Logic Layer (Бизнес Логика - Controller & Services): Обработка на заявките, валидация, управление на резервиране и отмяна.
3. Data Access Layer (Слой за Достъп до Данни - Model & Database Context): Използване на EF Core и SQL Server, управлявайки таблиците и операциите със данните.

Функционално Описание

Приложението предоставя следните основни функционалности:

3.4.1 Управление на Ресурсите (CRUD Операции)

Създаване: Добавяне на нови данни в базата

Четене: Извличане и показване на всички стаи

Актуализиране: Редакция на съществуващи данни.

Изтриване: Премахване на данни.

Пример за CRUD операции върху стаи:

| **Операция** | **URL Маршрут** | **Описание** |
| --- | --- | --- |
| Създаване | /Rooms/Create | Форма за добавяне на нова стая |
| Четене | /Rooms/Index | Списък с всички книги |
| Актуализиране | /Rooms/Edit | Форма за редактиране на стая |
| Изтриване | /Rooms/Delete | Потвърждение и изтриване |

3.4.2 Автентикация и Авторизация

Вход чрез вече съществуващ профил, направен от Admin; управление на роли като администратор или стандартен потребител; ограничен достъп според ролята.

Пример код:

[Authorize(Roles = "Admin")]

public IActionResult AdminPanel()

{

return View();

}

3.4.3 Работа с База Данни

Използване Entity Framework Core; включват релации между таблици.

Примерен модел:

Csharp

Public class Client{

[Key]

public int ClientId { get; set; }

[Required]

[StringLength(50)]

public string FirstName { get; set; }

[Required]

[StringLength(50)]

public string LastName { get; set; }

public class ApplicationDbContext : DbContext {

public DbSet<Client> Clients { get; set; }

}

3.4.4 Навигация и Изгледи

Основната навигация е дефинирана в \_Layout.cshtml. Страниците използват Razor синтаксис.

Примерен изглед (Views/Clients/Index.cshtml):

@model IEnumerable<HotelReservastionsManager.Models.Client>

@{ ViewData["Title"] = "Index"; }

3.4.5 API Поддръжка

Приложението може да предоставя RESTful API за работа с клиенти като SPA или мобилни приложения.

Поток На Заявките В Системата

1. Потребителят изпраща заявка към ReservationController.Create().

2. Контролерът извлича данните от базата през ApplicationDbContext.

3. HTML страница се генерира от View(Views/Reservations/Index.cshtml).

Реализация − обосновка за използвани технологични средства

При разработката на нашето ASP.NET Core MVC уеб приложение сме избрали следните технологии и инструменти поради техните предимства и възможности:

1. ASP .NET CORE MVC

* Платформено независим, модулен
* Вградена сигурност, Routing, Middleware

1. Entity Framework Core(EF Core)

 ORM: улеснява работата с бази данни

 Конфигурация в ApplicationDbContext

1. SQL Server

 Използван за локално съхранение на данни

 Конфигурация в appsettings.json:

"ConnectionStrings": {

"DefaultConnection": "Server=(localdb)\\MSSQLLocalDB;Database=HotelReservationsDb;Trusted\_Connection=True;"

}

1. ASP.NET Core Identity

* Управление на регистрации, вход, роли и сигурност
* Вградени механизми за сигурност

1. Razor View Engine

* Динамично съдържание чрез смесване на HTML и C#
* Частични изгледи за резервации, клиенти и други

1. Bootstrap или Tailwind CSS

* За стилизиране и респонсив дизайн

Източници На Информация:

За разработката са ползвани информационните ресурси от официалния сайт на Microsoft Learn, както и различни онлайн обучителни материали по темата.

Описание на Базата Данни

Базата данни на приложението е проектирана да поддържа управлението на каталог за книги, като включва релации между различните таблици. Използва се Microsoft SQL Server като основна база данни, а Entity Framework Core (EF Core) служи за Object-Relational Mapping (ORM).

Таблиците и Атрибутите

Основните таблици в базата включват:

1. Таблица Rooms (Стаи)

Тази таблица съдържа информация за стаите.

| **Колона** | **Тип На Данните** | **Описание** |
| --- | --- | --- |
| RoomNumber | int | Уникален идентификатор на стая |
| Capacity | int | Капацитет на стаята |
| RoomType | enum | Тип на стаята |
| IsAvailable | bool | Свободна или заета |
| AdultPrice | decimal | Цена за възрастен на нощувка |
| ChildPrice | decimal | Цена за дете на нощувка |
| Reservations | ICollection<Reservation> | Колекция на резервации, в които участва стаята |

Допълнителни таблици:

Client.cs, Reservation.cs, ErrorViewModel.cs, User.cs, ReservationClient.cs,

Връзки между таблиците:

Много Към Много: Reservations – Clients, чрез ReservationClients

Едно към Много: Rooms - Reservations

Заключение

Проектът демонстрира ефективното изграждане на ASP.NET Core MVC приложение с добре структурирана архитектура и ясно разделение между слоевете на логиката, данните и визуализацията. Включва базова автентикация, ролево управление и съвременен потребителски интерфейс, съобразен с UX практики.

Приложение в Практиката

Може да бъде използвано като основа за вътрешни системи за управление на съдържание или корпоративни портали.

Възможности за развитие И усъвършенстване

Интеграция с API: Добавяне на външни REST API за синхронизация с други системи.

Нотификации в реално време: Използване на SignalR за показване на известия към потребителите без презареждане.

Респонсив дизайн: По-добро адаптиране към мобилни устройства за по-широка употреба.