Документация на проекта: Мениджър за хотелски резервации

---

1. Въведение

1.1 Цел на документа

Настоящата документация има за цел да опише структурата, функционалността, архитектурата и начина на използване на софтуерното приложение „Мениджър за хотелски резервации“ (Hotel Reservations Manager). Документът е насочен към разработчици, тестери, системни администратори и бъдещи поддръжници и потребители на системата, за да предостави пълна представа за проекта, неговите компоненти и логика.

1.2 Аудитория

Документацията е предназначена за:

- **Разработчици**, които ще поддържат или разширяват функционалността на приложението;
- **QA специалисти**, които ще извършват тестване на системата;
- **ИТ администратори**, които ще внедряват системата в реална среда;
- **Клиенти или заинтересовани страни**, които желаят да разберат какво предлага системата и как работи.

1.3 Обхват на проекта

„Мениджър за хотелски резервации“ е уеб базирано приложение, което цели да оптимизира процеса на управление на хотелски резервации, настаняване и съпътстваща информация за клиенти, стаи и потребители (служители). Системата предлага възможности за:

- Регистрация и управление на служители с различни нива на достъп;
- Добавяне, редактиране и търсене на клиенти;
- Управление на хотелски стаи;
- Създаване и администриране на резервации с автоматично изчисляване на дължимата сума;
- Филтриране и странициране на данни за по-добра визуализация и контрол.

Системата е проектирана с акцент върху сигурност, лесна поддръжка и разширяемост.

1.4 Технологии и инструменти

Приложението е разработено с помощта на следните технологии и инструменти:

- **ASP.NET Core 8 (MVC)** – основен фреймуърк за изграждане на уеб приложението;
- **Entity Framework Core (Code First)** – ORM за работа с база данни;
- **ASP.NET Core Identity** – за управление на потребители и роли;
- **C#** – основен език за програмиране;
- **SQL Server** – база данни;
- **Visual Studio 2022/2023** – основна среда за разработка;
- **Bootstrap** – за стилизиране и отзивчивост на интерфейса;
- **LINQ, AutoMapper** – за по-ефективна логика и трансформации на данни.

---

2. Архитектура на системата

2.1 Общ преглед на архитектурата

Проектът „Мениджър за хотелски резервации“ е реализиран с многослойна архитектура, която спазва принципите на разделение на отговорностите (*Separation of Concerns*). Това позволява по-добра поддръжка, модулност и мащабируемост на системата. Основните слоеве включват:

- **Слой за достъп до данни (Data Layer)**
- **Слой за бизнес логика и услуги (Service Layer)**
- **Презентационен слой (Presentation Layer – ASP.NET MVC)**

Всеки слой комуникира само с непосредствено свързания с него слой, като се избягва директен достъп от UI до базата данни.

2.2 Разделение на слоевете

2.2.1 Data Layer (Слой за достъп до данни)

Този слой отговаря за комуникацията с базата данни. Реализиран е с помощта на **Entity Framework Core** чрез подхода **Code First**. Включва:

- Контекст на базата данни (ApplicationDbContext)

- Entity класове (например: User, Room, Client, Reservation)
- Миграции

Този слой предоставя основата за създаване и поддръжка на структурата на базата данни.

2.2.2 Service Layer (Слой за услуги и бизнес логика)

Тук се реализира бизнес логиката на приложението. Всеки модул (напр. Резервации, Стаи, Клиенти) има собствена услуга (Service), която обработва заявките от презентационния слой. Примерни класове: ReservationService, RoomService, ClientService.

Слоят осигурява:

- Валидиране и обработка на данни
- Работа с entity-тата от Data Layer
- Абстракция между UI и база данни

2.2.3 Presentation Layer (Презентационен слой – MVC)

Изграден с ASP.NET Core MVC. Този слой съдържа:

- Контролери (Controllers)
- Изгледи (Views – Razor)
- ViewModels – използвани за пренос на данни между View и Controller

Презентационният слой е отговорен за визуализацията на данните, събиране на вход от потребителя и подаването му към Service Layer.

2.3 Използван подход – Code First с Entity Framework

Използван е подходът **Code First**, при който първо се създават C# класовете, описващи структурата на базата данни (entity-та), и след това чрез миграции се създава и актуализира базата. Това позволява пълен контрол върху модела на данните от страна на разработчиците и улеснява бъдещото разширяване на функционалността.

2.4 Диаграма на архитектурата

[UI Layer - ASP.NET MVC]

↓

[Service Layer - Business Logic]

↓

[Data Layer - EF Core + DbContext]

↓

[SQL Server Database]

---

3. Управление на потребителите

3.1 ASP.NET Core Identity – обосновка за избора

За управление на потребителите в приложението е използвана вградената библиотека **ASP.NET Core Identity**. Тя предлага цялостна инфраструктура за:

- Регистрация и вход на потребители
- Управление на роли и права за достъп
- Хеширане и съхранение на пароли
- Работа с „claims“, „tokens“ и политики за достъп

Изборът ѝ е мотивиран от това, че тя:

- предоставя готова архитектура за аутентикация и авторизация;
- е интегрирана с Entity Framework;
- е поддържана от Microsoft и отговаря на добрите практики за сигурност.

3.2 Модел на потребителя

Потребителите в приложението наследяват класа **IdentityUser**, като към него са добавени допълнителни свойства, отговарящи на нуждите на системата:

Свойство	Тип	Описание
Username	string	Уникално потребителско име
PasswordHash	string	Хеширана парола
FirstName	string	Собствено име
MiddleName	string	Бащино име

Свойство	Тип	Описание
LastName	string	Фамилия
EGN	string	Единен граждански номер
PhoneNumber	string	Телефонен номер (валидира се)
Email	string	Имейл адрес
HireDate	DateTime	Дата на назначаване
IsActive	bool	Активен/неактивен акаунт
ReleaseDate	DateTime?	Дата на освобождаване от длъжност (може да е null)

3.3 Роли и права на достъп

Системата използва ролево базирана авторизация (Role-based Access Control - RBAC). Дефинирани са две основни роли:

Роля	Описание
Администратор	Има достъп до всички функционалности – управление на потребители, стаи, резервации и клиенти.
Служител	Може да извършва резервации и да работи с клиенти, но няма достъп до управление на потребители и стаи.

Използват се атрибути като [Authorize(Roles = "Admin")] за защита на съответните контролери и действия.

3.4 Аутентикация и авторизация

ASP.NET Core Identity се грижи за:

- **Аутентикация** – потвърждаване на самоличността чрез потребителско име и парола;
- **Авторизация** – проверка дали даден потребител има право да достъпи конкретна функционалност, в зависимост от ролята му.

Системата използва **cookie-based authentication** с опция за запомняне на сесията и автоматично излизане при изтичане на времето.

3.5 Управление на състоянието на акаунта

- При създаване на акаунт, потребителят се счита за **активен** (IsActive = true).
- При освобождаване от длъжност, стойността се променя на IsActive = false и се попълва ReleaseDate.
- Неактивните потребители **не могат да влизат** в системата и **нямат достъп** до никаква функционалност.
- Данните за неактивни потребители и направените от тях резервации **се запазват** в базата и са достъпни само за преглед от администратор.

---

4. Описание на основните модули

Системата се състои от четири основни модула, които реализират основната ѝ бизнес логика:

- **Потребители (ApplicationUsers)**
- **Клиенти (Clients)**
- **Стаи (Rooms)**
- **Резервации (Reservations)**

4.1 Модул: Потребители

Цел:

Управление на служителите в системата чрез създаване, редакция, активиране/деактивиране и задаване на роли.

Функционалности:

- CRUD операции върху потребителите (само за администратор);
- Странициране и филтриране по потребителско име, имена и имейл;
- Ограничаване на достъпа за неактивни акаунти;
- Назначаване на роли (Admin/Служител);
- Показване на списък с всички потребители с възможност за филтриране и сортиране.

Взаимодействие:

- Служителите са асоциирани с направените резервации;
- Достъп до модулите се контролира чрез Identity роли.

4.2 Модул: Клиенти

Цел:

Съхраняване и използване на клиентска информация за резервации и история на престои.

Функционалности:

- CRUD за клиентите;
- Преди създаване на резервация се проверява дали клиентът вече съществува;
- Разделение на клиенти по възрастова категория (възрастни и деца);
- Странициране и филтриране по имена;
- Преглед на история на резервации на даден клиент.

Основни свойства:

- Име, фамилия, телефон, имейл, възрастова категория (булева стойност: възрастен/дете)

Взаимодействие:

- Свързани с множество резервации;
- Използват се при създаване на резервации.

4.3 Модул: Стаи

Цел:

Управление на хотелските стаи – техните типове, капацитет и наличност.

Функционалности:

- CRUD операции (само за администратор);
- Филтриране по капацитет, тип и заетост;
- Преглед на налични стаи;

- Автоматично маркиране на стаи като заети/свободни след резервация или освобождаване.

Основни свойства:

- Номер, капацитет, тип, свободна/заета, цена на нощувка за възрастни/деца

Взаимодействие:

- Свързани с резервации;
- Избират се при създаване на нова резервация.

4.4 Модул: Резервации

Цел:

Централен модул, който управлява настаняването на клиенти, избора на стаи и изчисляване на дължимата сума.

Функционалности:

- Пълен CRUD;
- Избор на съществуващи клиенти и стаи;
- Възможност за добавяне на закуска и all inclusive;
- Автоматично изчисление на дължимата сума въз основа на:
    - Брой нощувки
    - Вид на стаята
    - Брой възрастни и деца
    - Допълнителни услуги (закуска, all inclusive)
- Промяна на статуса на стая след освобождаване;
- Подробен изглед с всички данни за резервацията и клиентите в нея.

Основни свойства:

- Дата на настаняване и освобождаване
- Клиенти
- Стая
- Служител

- Допълнителни услуги (закуска, all inclusive)
- Изчислена стойност на резервацията

Взаимодействие:

- Зависи от модулите „Клиенти“, „Стаи“ и „Потребители“;
- Променя статуса на стаите при резервация и освобождаване;
- Съхранява се историята на направените резервации.

---

5. Управление на данни и валидация

5.1 Общ преглед

Системата „Мениджър за хотелски резервации“ разчита както на **клиентска**, така и на **сървърна** валидация, за да осигури:

- коректност и пълнота на въведените данни;
- избягване на логически грешки (напр. несъществуващи стаи или неправилни дати);
- предотвратяване на атаки чрез въвеждане на невалидни стойности;
- запазване на консистентността в базата данни.

Използвани технологии:

- **Data Annotations** за валидация на моделите
- **Custom Validation Attributes** при нужда от по-сложна логика
- **JavaScript/jQuery** за визуална обратна връзка на клиента
- **ModelState и Fluent Validation** за проверка на сървърно ниво

5.2 Валидация на основни модели

Потребител

- Username – задължително, уникално
- Password – минимална дължина 6 символа, задължителна
- PhoneNumber – дължина точно 10 цифри, само числа
- EGN – точно 10 цифри, уникално, само числа
- Email – задължително, с валиден формат

- HireDate – не може да бъде бъдеща дата
- ReleaseDate – ако е зададена, трябва да е **по-късна от HireDate**

Клиент

- FirstName и LastName – задължителни, без специални символи
- PhoneNumber – както при потребител
- IsAdult – булева стойност, изчислява се по възраст при нужда
- Email – по избор, но ако се попълни, трябва да е валиден

Стая

- RoomNumber – уникален в хотела
- Capacity – >= 1
- Type – валидиран спрямо избрани стойности (enum)
- PricePerAdult / PricePerChild – положителни стойности
- IsFree – булева стойност, обновява се автоматично след резервация

Резервация

- CheckInDate – не по-рано от днешна дата
- CheckOutDate – **задължително по-късна от CheckInDate**
- RoomId – трябва да сочи към свободна стая
- Clients – поне един клиент трябва да бъде избран
- TotalAmount – изчислява се автоматично чрез бизнес логика (не е потребителски вход)
- Ако е избрана опция за all inclusive или закуска – цената се променя според фиксирани надбавки

5.3 Валидация във формите (UI ниво)

Всички форми използват:

- **ASP.NET Tag Helpers**, които автоматично връзват валидационните атрибути към HTML;
- **jQuery Unobtrusive Validation**, за да се покаже грешка без презареждане на страницата;

- Изрични съобщения за грешки при празни полета, грешни формати и логически несъответствия.

5.4 Бизнес правила за валидация

- Не може да се направи резервация без поне една свободна стая.
- Ако стаята няма капацитет за избраните клиенти, се отказва резервацията.
- Потребител без роля или неактивен потребител не може да влиза в системата.
- Не може да се изтрие потребител или клиент, ако има направени резервации от/за него – допуска се само маркиране като „неактивен“.
- При приключване на резервация, стаята автоматично става свободна.

5.5 Обработка на грешки

- При всяка операция, която не преминава валидация, потребителят получава ясно съобщение.
- Всички грешки се логват чрез middleware за по-лесна поддръжка и отстраняване.
- При възникване на вътрешна грешка – системата показва потребителско съобщение, без да разкрива вътрешна логика.

---

6. Архитектура на системата и структура на проекта

6.1 Архитектурен модел

Проектът „Мениджър за хотелски резервации“ следва архитектурен стил **Layered Architecture (Многослойна архитектура)**, като логиката е ясно разделена в следните слоеве:

- **Презентационен слой (Presentation Layer / Web Layer)**
  Отговаря за взаимодействието с потребителя – представя UI чрез Razor Pages или MVC изгледи, приема вход от формите и предава данните към бизнес слоя.
- **Слой за услуги (Service Layer / Business Logic Layer)**
  Реализира основната бизнес логика – обработка на заявки, валидации,

изчисления, комуникация с репозиториите и трансформиране на данни за UI.

- **Слой за достъп до данни (Data Access Layer)**
  Съдържа моделите на базата данни и комуникира с нея чрез Entity Framework Core, използвайки **Code First подхода**.
- **База данни (Database Layer)**
  Построена чрез миграции в EF Core, съдържа таблици за всички основни обекти – потребители, клиенти, стаи, резервации и техните връзки.

6.2 Използвани технологии и рамки

- **ASP.NET Core 8 (MVC)** – за изграждане на уеб интерфейс и цялостната логика на приложението.
- **Entity Framework Core (Code First)** – ORM, чрез който се дефинира и създава базата чрез C# класове.
- **ASP.NET Core Identity** – за автентикация и управление на потребители и роли.
- **SQL Server** – като релационна база данни.
- **AutoMapper** – за преобразуване на модели (DTO <-> ViewModel <-> Entity).

6.3 Структура на проекта

Проектът е разделен в отделни папки по отговорности:

```
HotelReservationsManager/
│
├── Data/
│   ├── Entities/          # Класове за базата (User, Client, Room, Reservation)
│   ├── Context/           # AppDbContext – главната конфигурация на EF Core
│   └── Migrations/        # Миграции за Code First подхода
│
├── Services/
│   ├── Interfaces/        # Интерфейси на услугите
```

```
│   └── Implementations/      # Конкретни имплементации (RoomService, ReservationService и т.н.)
│
├── Web/
│   ├── Controllers/          # MVC контролери
│   ├── Views/                # Razor изгледи за всеки контролер
│   ├── ViewModels/           # Модели за визуализация на данни във формите
│   └── wwwroot/              # Статични ресурси (CSS, JS, изображения)
│
├── Models/                   # Общи модели за споделяне между слоевете
├── MappingProfiles/          # AutoMapper конфигурации
├── Program.cs                # Конфигурация на приложението
└── appsettings.json          # Настройки на базата, Identity и други
```

6.4 Принципи и добри практики

- Проектът следва **SOLID** принципите и **Dependency Injection**.
- Във всеки слой се спазва **единствена отговорност** – няма смесване на логика от различни слоеве.
- Всички методи и класове са придружени с **коментари**, обясняващи тяхната роля.
- Използвани са **стилови насоки на Microsoft за C#**, включително именуване, подредба и модулност.
- Във всички View-и се използват **strongly typed ViewModels**, за по-добра сигурност и поддръжка.

---

7. Управление на потребителите и роли

7.1 Въведение

Управлението на потребители и техните роли е реализирано чрез **ASP.NET Core Identity**, което предоставя готова и гъвкава инфраструктура за:

- Регистрация и вход на потребители

- Управление на пароли
- Ролева авторизация
- Заключване/активиране на акаунти
- Проследяване на активността на потребителите

Системата е конфигурирана да използва **потребителски клас**, който наследява IdentityUser, с добавени допълнителни свойства, специфични за нуждите на хотела.

7.2 Модел на потребителя

Разширеният потребителски модел (ApplicationUser) съдържа следните свойства:

```
// Персонализиран клас за потребители, наследяващ IdentityUser
  public class ApplicationUser : IdentityUser
  {
    [Required]
    [StringLength(50)]
    public string FirstName { get; set; } // Собствено име

    [Required]
    [StringLength(50)]
    public string MiddleName { get; set; } // Бащино име

    [Required]
    [StringLength(50)]
    public string LastName { get; set; } // Фамилно име

    [Required]
    [StringLength(10, MinimumLength = 10, ErrorMessage = "ЕГН трябва да е точно 10 символа.")]
```

```
    public string EGN { get; set; } // ЕГН


    [Required]
    [StringLength(10, ErrorMessage = "Телефонният номер не може да надвишава 10 символа.")]
    public string PhoneNumber { get; set; } // Телефонен номер (predefinira se ot IdentityUser)


    [Required]
    public DateTime HireDate { get; set; } // Дата на назначаване


    public bool IsActive { get; set; } = true; // Aktiven ili neaktiven akaunt


    public DateTime? ReleaseDate { get; set; } // Дата на освобождаване (opcionална)
  }
```

7.3 Роли в системата

Системата поддържа две основни роли:

Роля	**Описание**
Администратор	Пълен контрол – управление на потребители, стаи, резервации и клиенти
Служител	Ограничени права – може да създава и управлява резервации и клиенти

Ролите се задават при създаването на потребителя от администратора чрез административен интерфейс. Не е позволено обикновен служител да създава нови акаунти.

7.4 Управление на акаунти

Създаване на потребители

- Само администратор може да създава нови потребители.

- При създаване се задават личните данни, потребителско име, роля и парола.

Редактиране и изтриване

- Администраторът може да редактира всички потребителски данни (без парола – тя се сменя поотделно).
- Потребител не може да бъде изтрит, ако има създадени резервации – в такъв случай се маркира като „неактивен“.

Деактивиране на потребител

- Полето IsActive контролира достъпа. Ако стойността е false, потребителят не може да влезе в системата.
- Ако е зададена дата на освобождаване (ReleaseDate), IsActive се превключва автоматично.

7.5 Вход и защита на системата

- Потребителите влизат чрез стандартна форма за логин (Login).
- Използва се **Cookie Authentication** от ASP.NET Core Identity.
- Неправилните опити за вход се броят – след определен брой неуспешни опити акаунтът може да бъде временно заключен.

7.6 Авторизация и защита на изгледи

Системата използва **Role-based Authorization**, като достъпът до различни контролери/действия се ограничава чрез атрибути като:

```
[Authorize(Roles = "Administrator")]
public IActionResult ManageUsers() { ... }

[Authorize(Roles = "Employee,Administrator")]
public IActionResult CreateReservation() { ... }
```

Във View-овете се използват проверки като:

```
@if (User.IsInRole("Administrator"))
```

```
{
  <a href="/Users/Create">Създай нов потребител</a>
}
```

7.7 Управление на пароли и сигурност

- При създаване на акаунт се изисква парола с минимална дължина и поне една цифра.
- Потребителите могат да сменят паролата си от профилната страница.
- Администраторът може да инициира смяна на парола (reset), но няма достъп до текущата парола.
- Всички пароли се съхраняват **хеширани** чрез ASP.NET Identity.

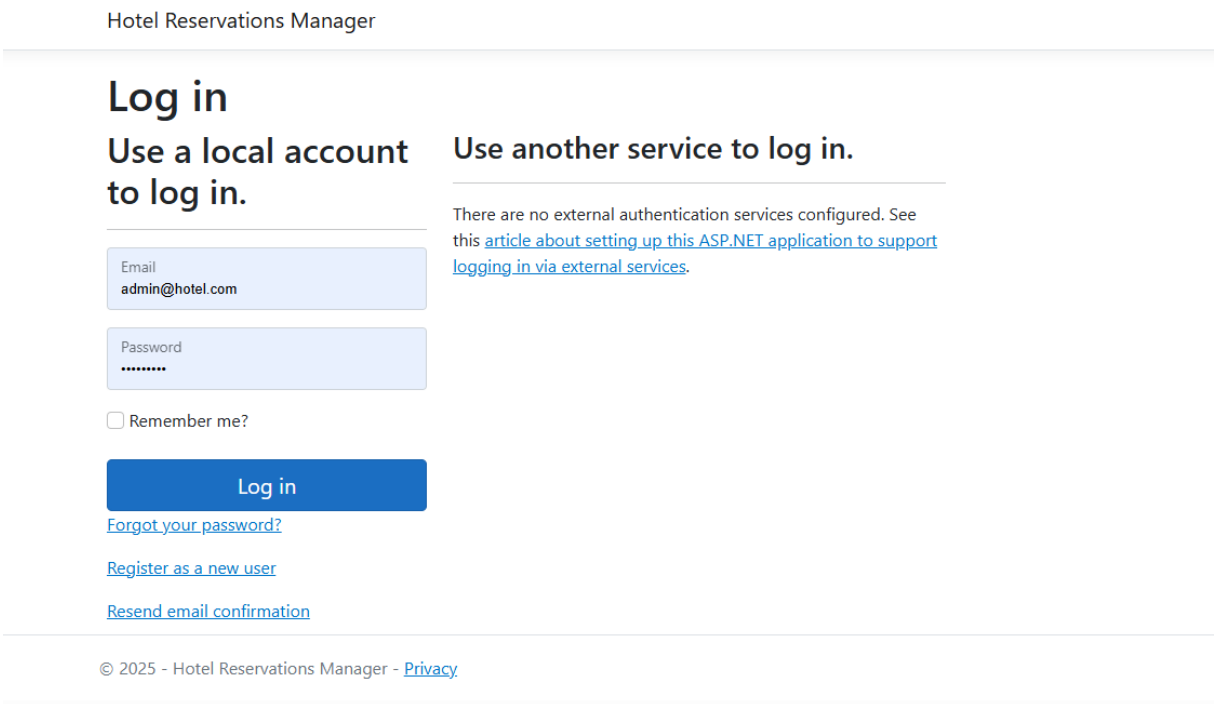
---

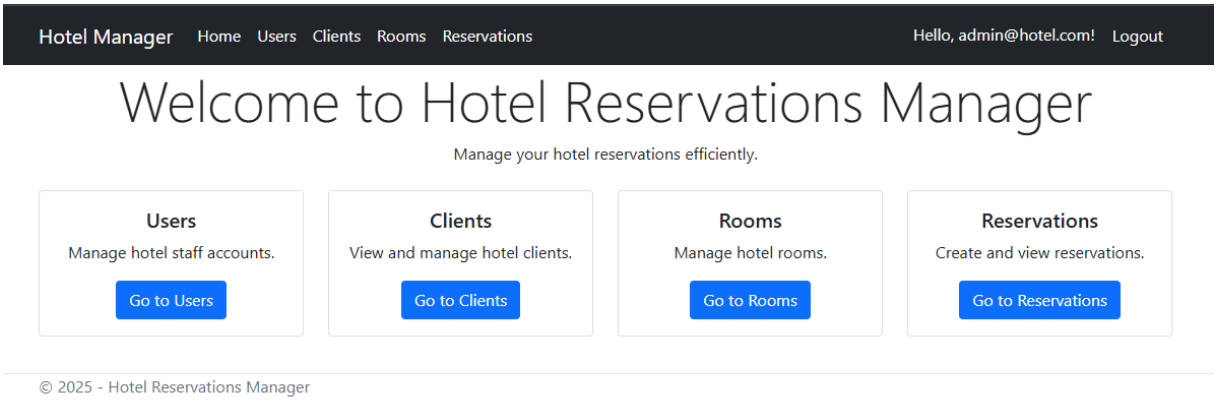
8. Функционалност на приложението

8.1 Общ преглед

Приложението предоставя уеб интерфейс за работа със системата за резервации. Достъпът до всяка функционалност се контролира според ролята на потребителя – **администратор** или **служител**. Всички действия са реализирани чрез **CRUD операции**, валидирани формуляри и отзивчив интерфейс със странициране и филтриране.

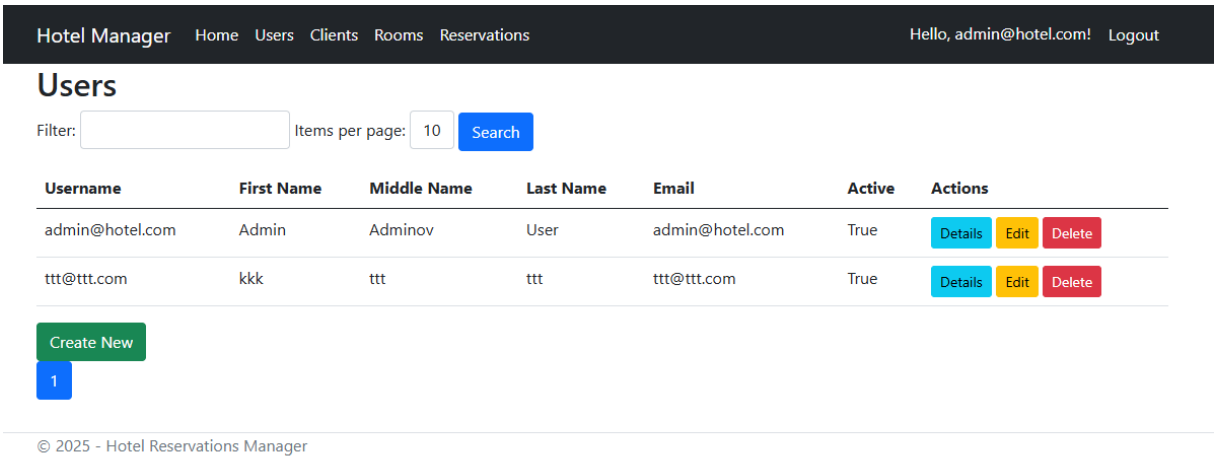
Системата автоматично генерира един потребител администратор с потребителско име admin@hotels.com и парола 123@Admin.



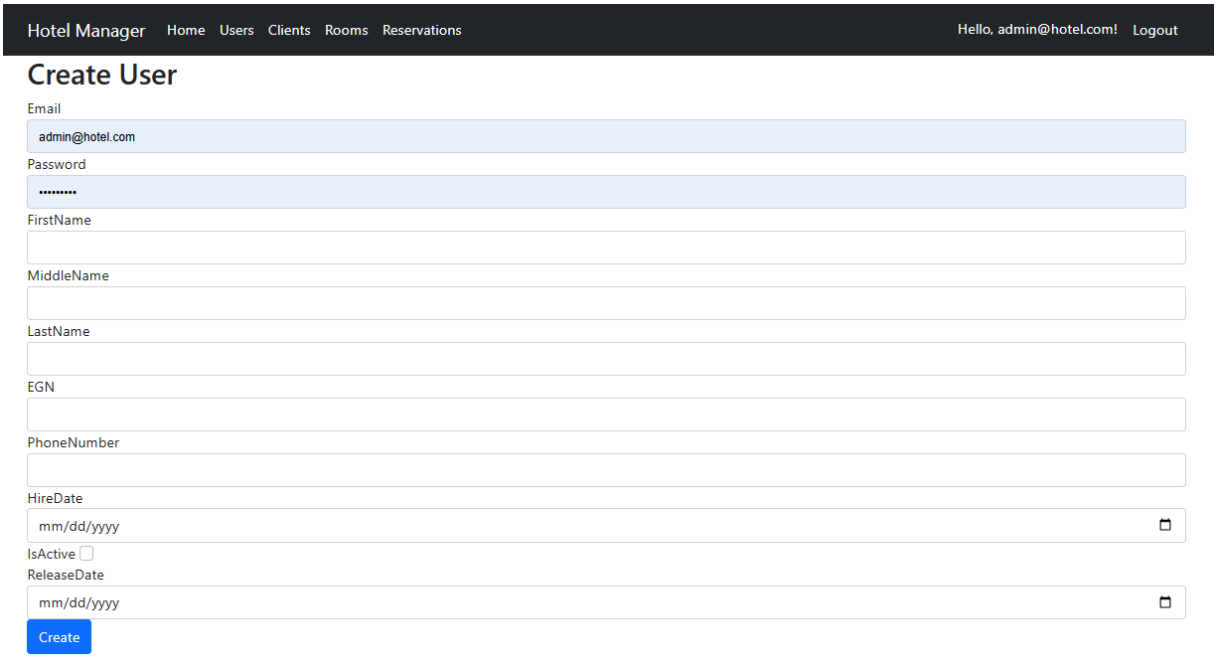


8.2 Работа с потребители (само администратор)

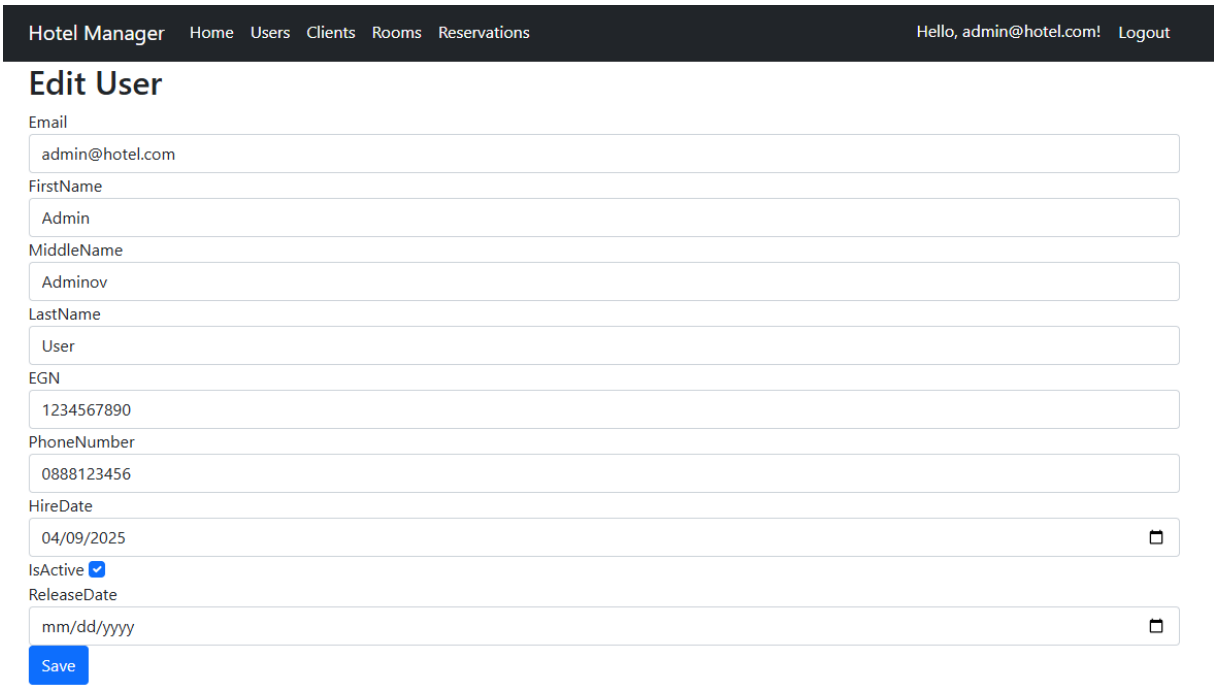
- **Преглед на всички потребители** – изглед с таблица, странициране (10, 25, 50 записа) и филтри по потребителско име, имейл, имена.



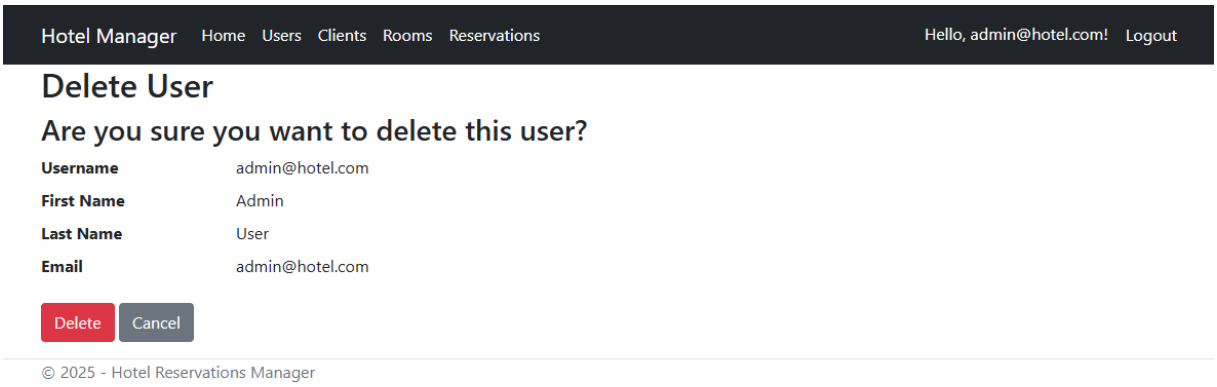
- **Създаване на потребител** – форма с всички задължителни данни и избор на роля.



- **Редакция на потребител** – възможност за промяна на личните данни, без парола.

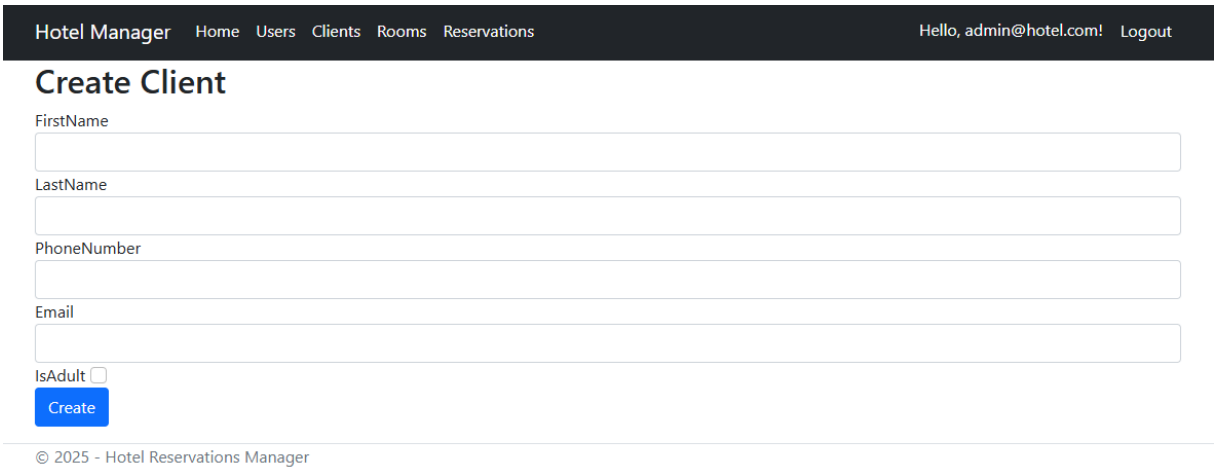


- **Деактивиране на потребител** – чрез чекбокс или задаване на дата на освобождаване.
- **Изтриване на потребител** – възможно само ако потребителят няма активни резервации.

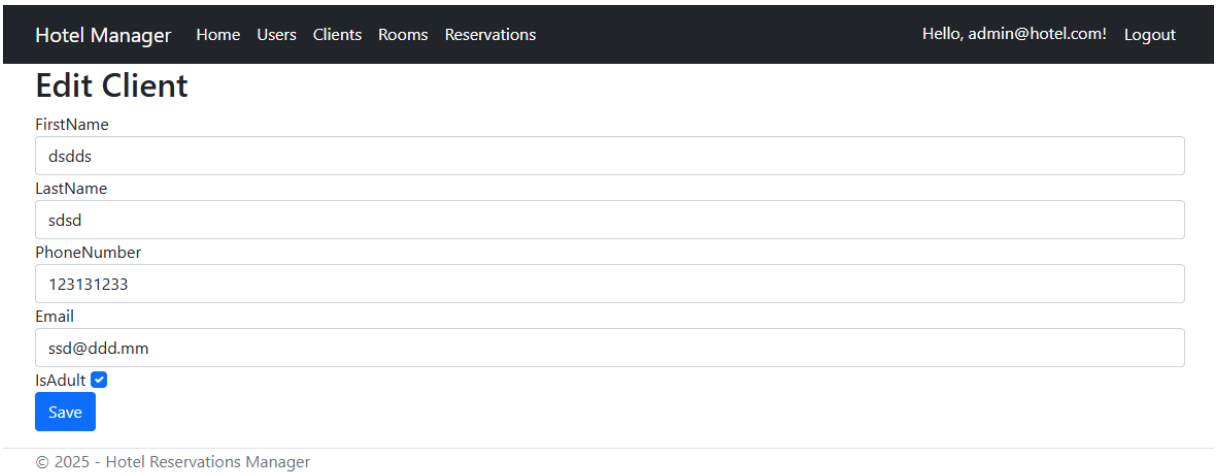


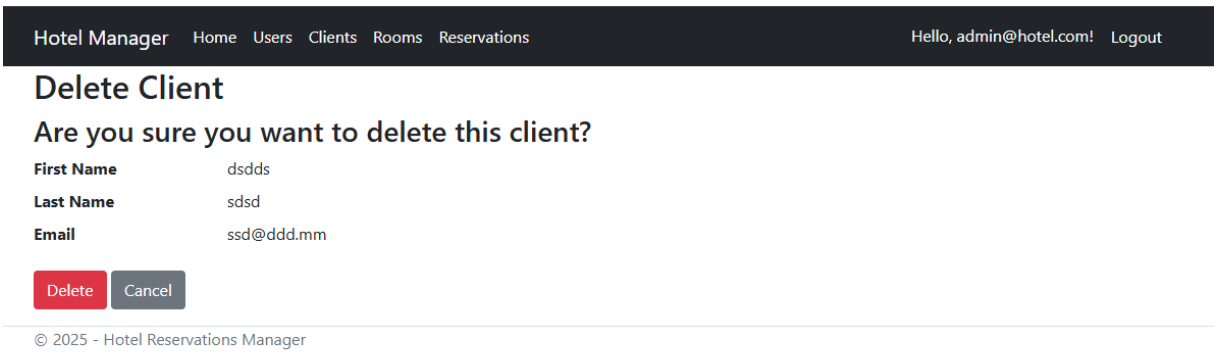
8.3 Работа с клиенти (служители и администратори)

- **Създаване на клиент** – форма с полета за име, имейл, възраст, телефон.

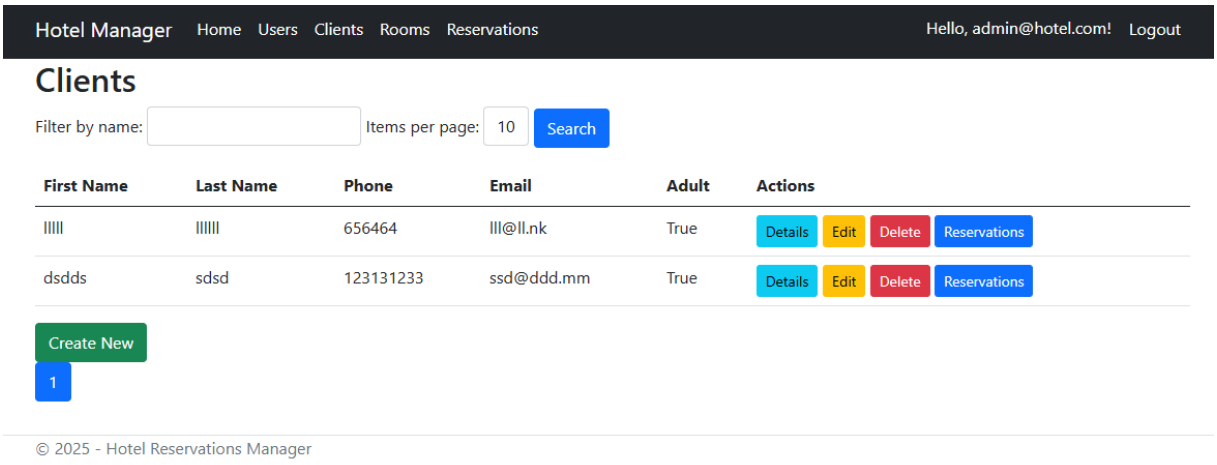


- **Редакция и изтриване** – позволени, ако клиентът няма текуща резервация.





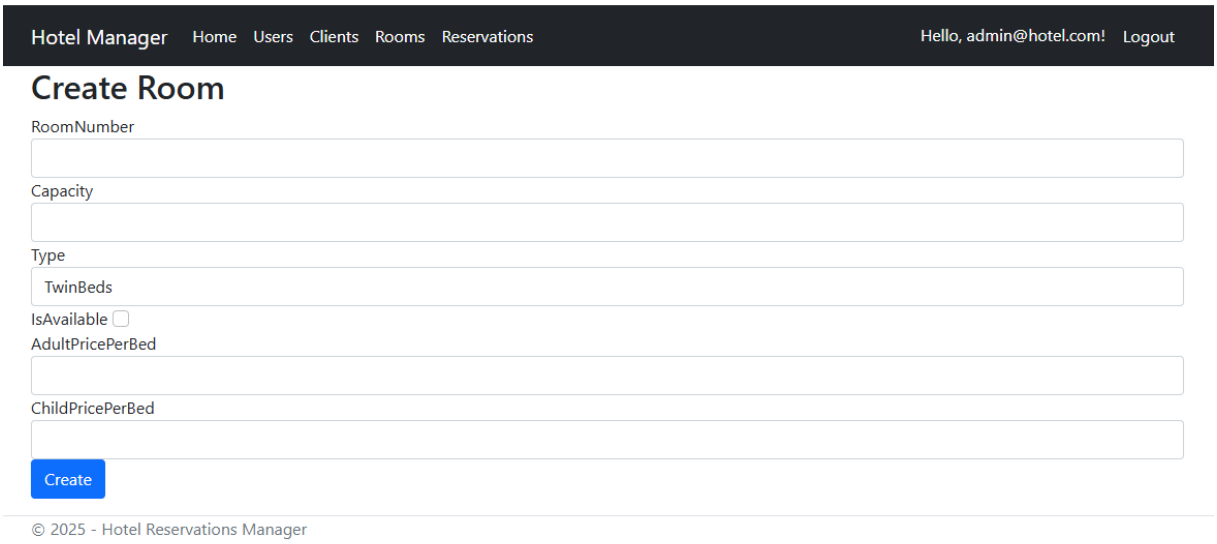
- **Преглед на клиенти** – таблица с филтри и странициране, както и бутон за “Резервации на клиента”.



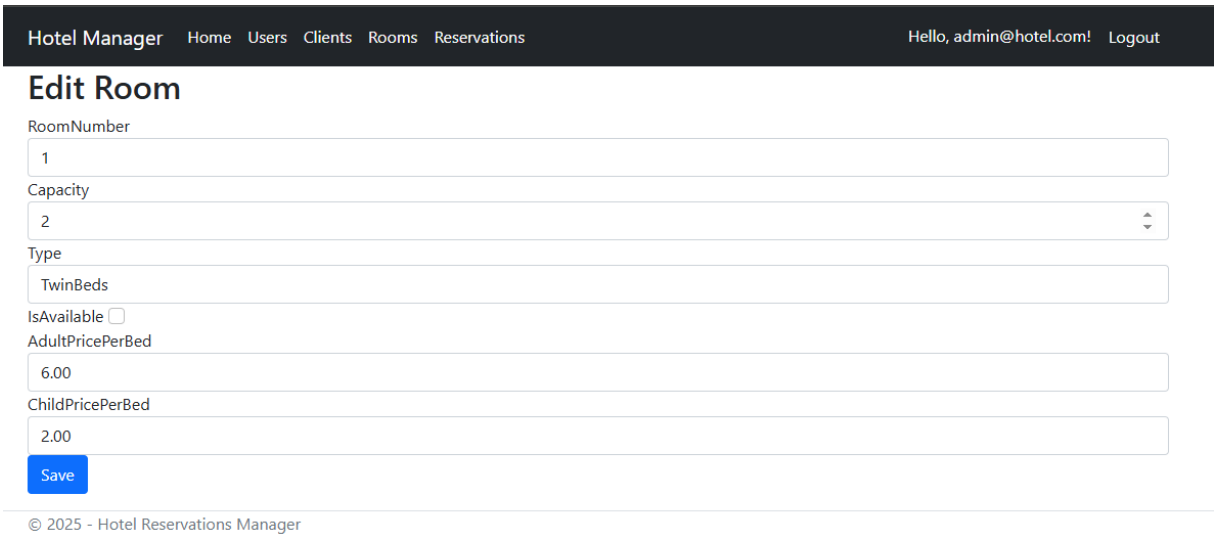
- **История на резервациите** – преглед на всички резервации, свързани с клиента.

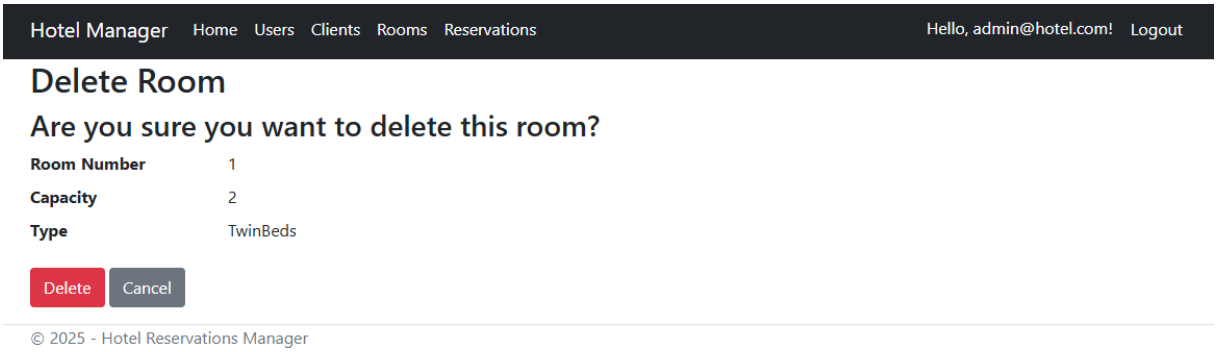
8.4 Работа със стаи (само администратор)

- **Създаване на стая** – задаване на капацитет, тип, цена на легло (възрастен и дете), номер.

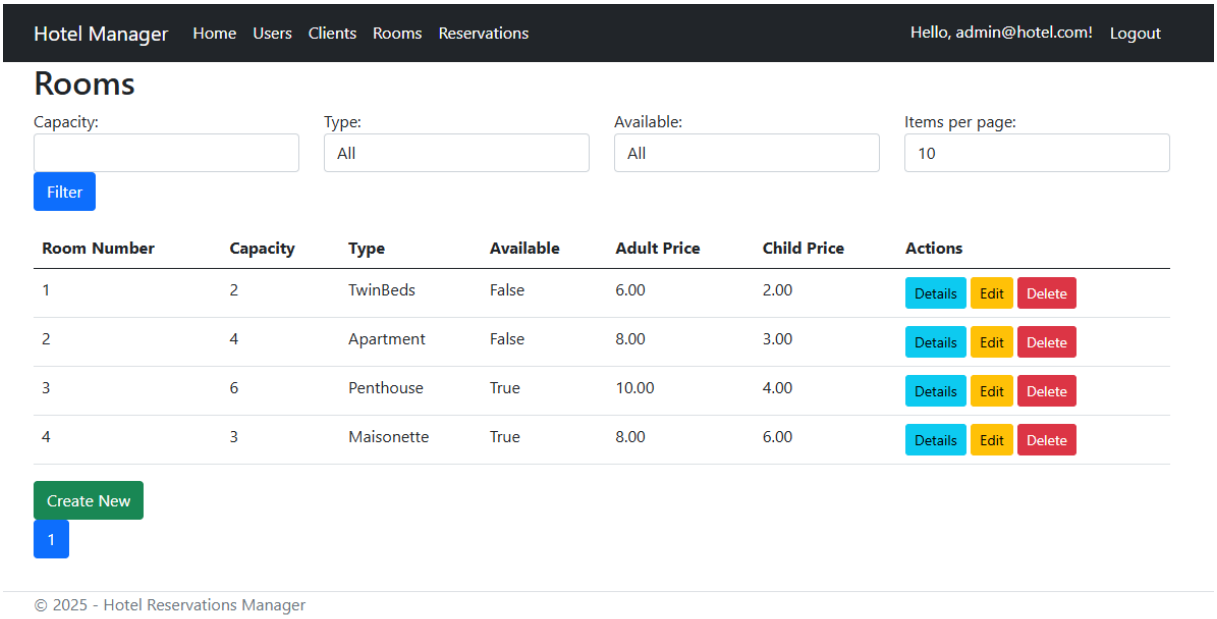


- **Редакция и изтриване** – само ако няма активна резервация за стаята.



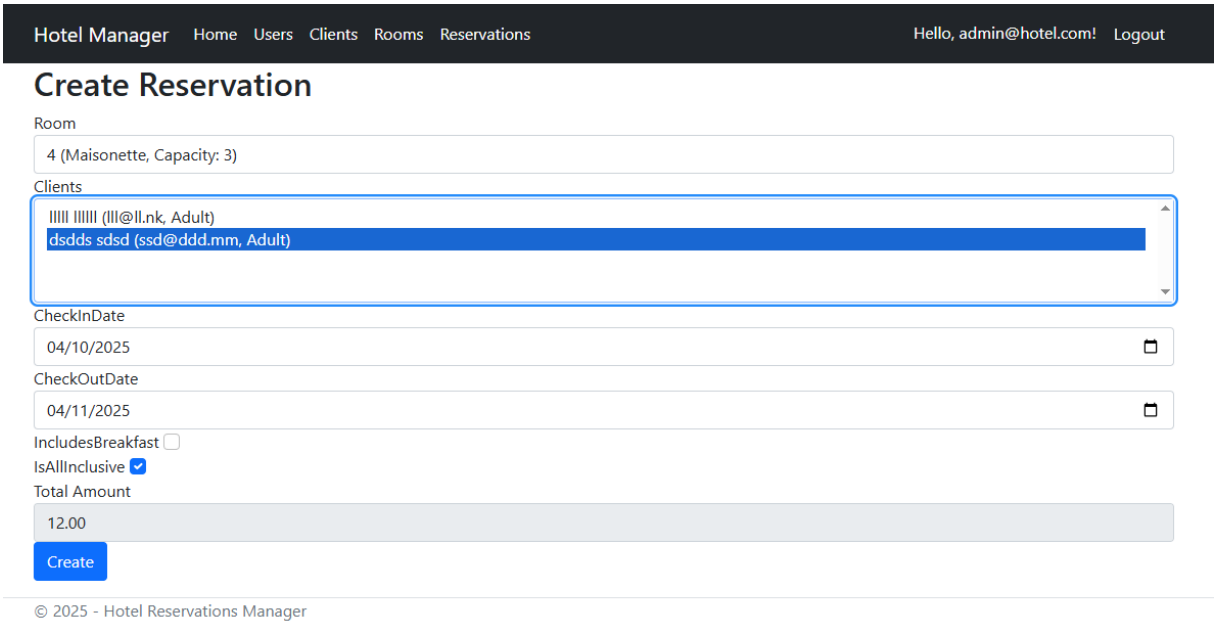


- **Преглед на стаи** – филтриране по капацитет, тип и заетост. Включен е индикатор дали е свободна.

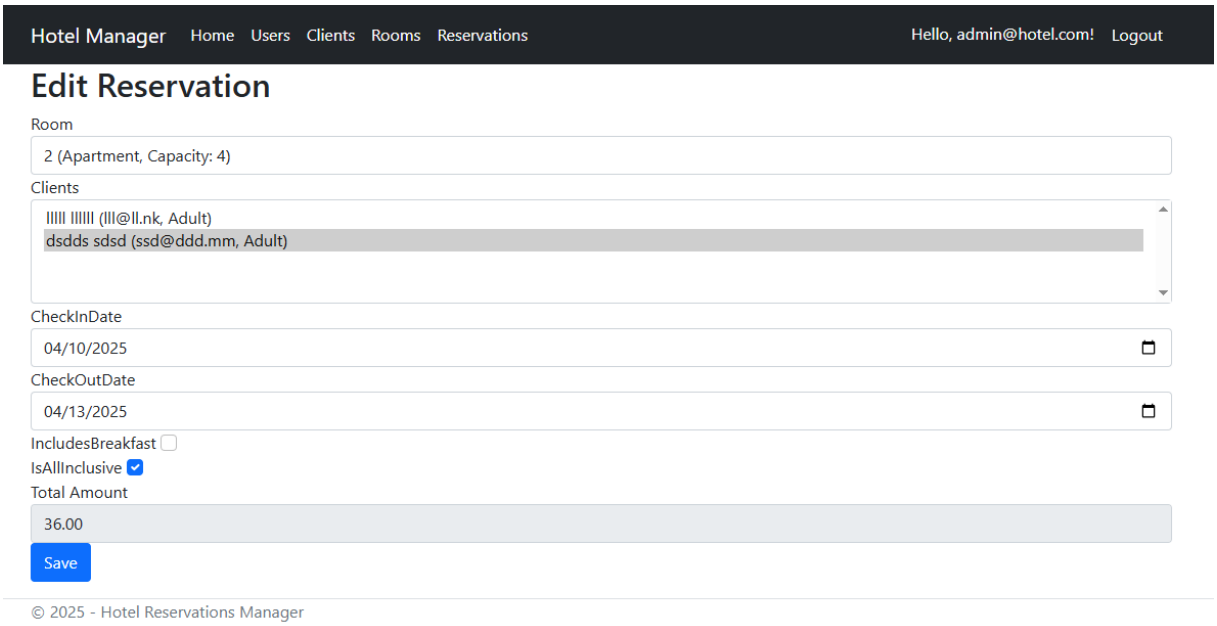


8.5 Работа с резервации (всички активни потребители)

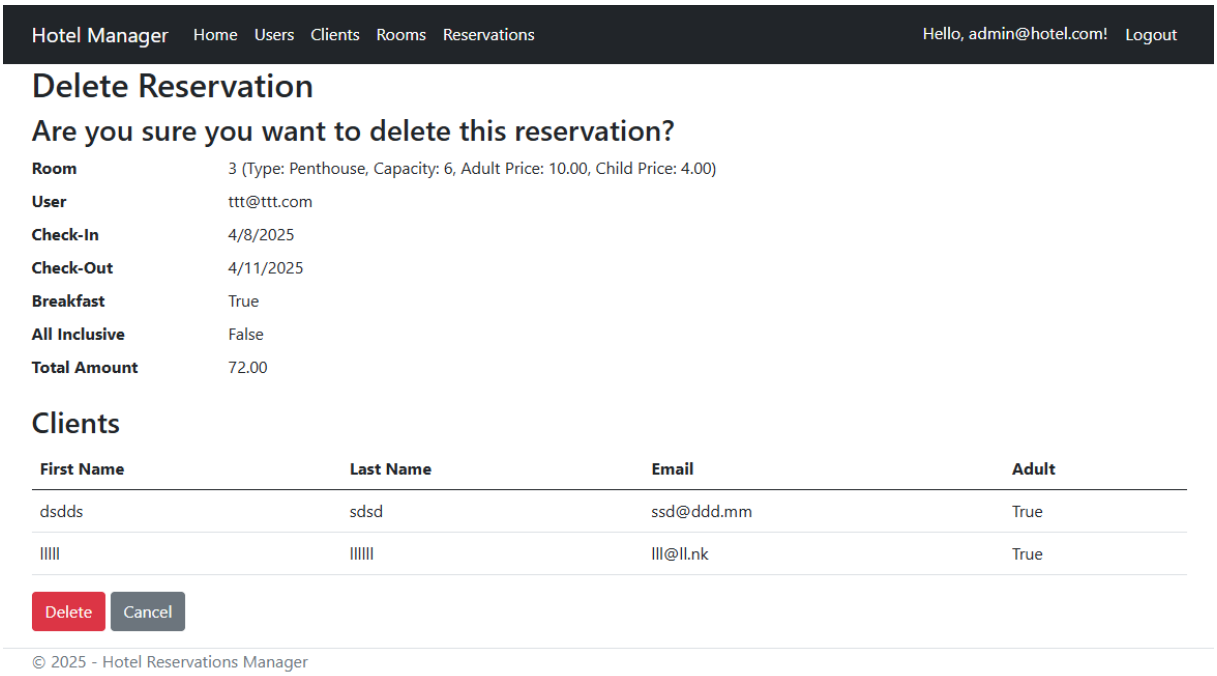
- **Създаване на резервация**:
    - Избор на свободна стая по дати и тип.
    - Избор на съществуващи клиенти (или добавяне на нов).
    - Въвеждане на дата за настаняване/освобождаване, закуска и all inclusive.
    - Автоматично изчисление на дължимата сума спрямо възрастни/деца, брой нощувки и екстри.



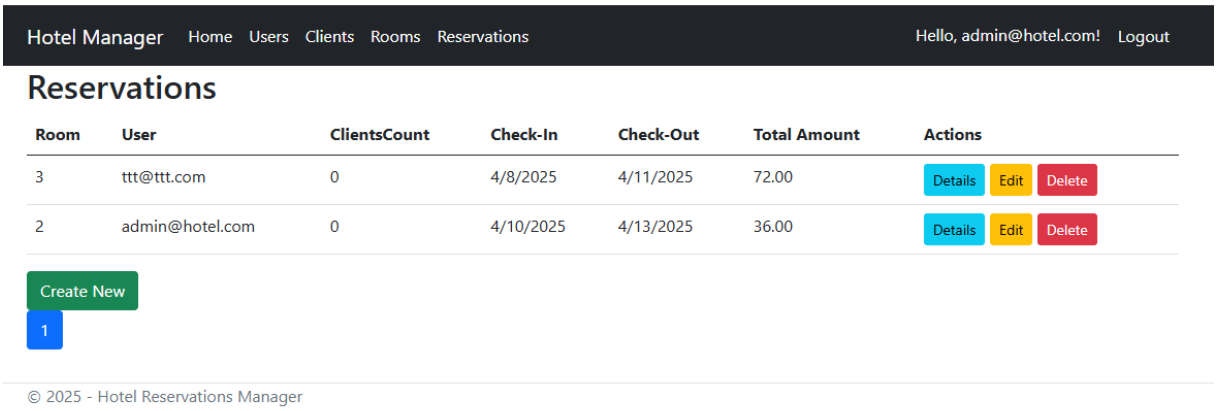
- **Редакция на резервация**.

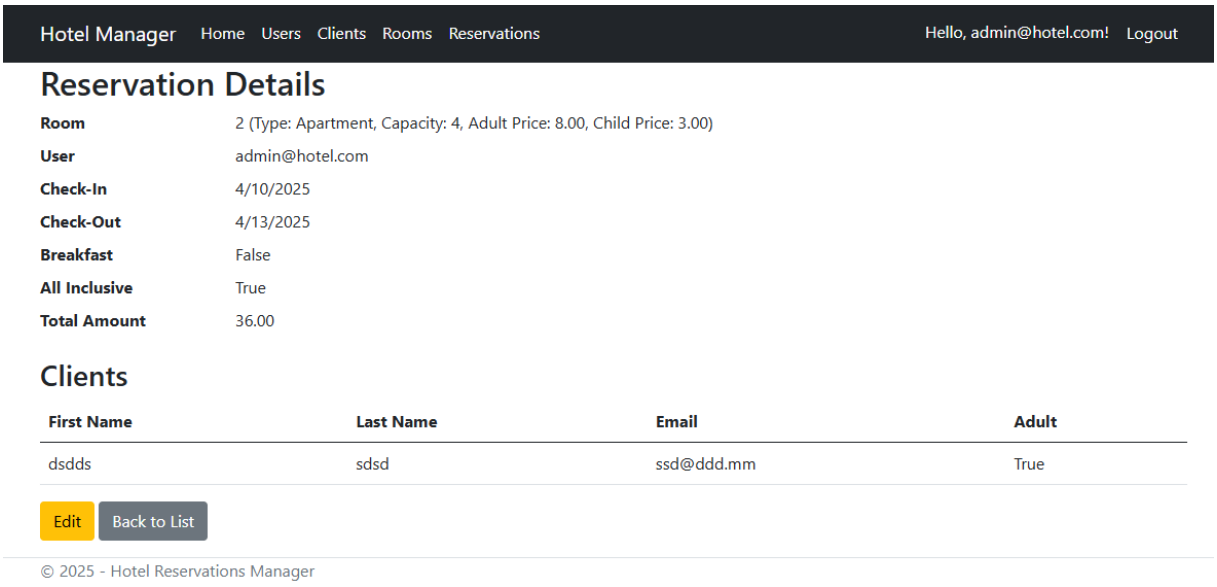


- **Изтриване.**



- **Преглед на резервации** – с детайли за клиенти, дати, услуги и сума.





- **Завършване на резервация** – при освобождаване на стаята, тя автоматично се отбелязва като свободна.

8.6 Търсене и филтриране

Всеки основен изглед (потребители, клиенти, стаи, резервации) поддържа:

- **Филтри по основни полета** – текстови и селекторни.
- **Странициране** – потребителят може да избира колко записа да се показват.

8.7 Валидация

Формите имат **както клиентска, така и сървърна валидация**. Някои примери:

- Телефонните номера се валидират да са точно 10 цифри.

- Дата на освобождаване не може да е преди датата на настаняване.
- Стая не може да бъде резервирана, ако вече е заета за избрания период.
- При създаване на клиент или потребител – всички задължителни полета са валидирани с DataAnnotations.

---

9. Сигурност и защита на данните

9.1 Въведение

Сигурността на потребителските данни и защита от неоторизиран достъп са основни приоритети на системата „Hotel Reservations Manager“. Приложението използва комбинация от вградени механизми на **ASP.NET Core**, както и допълнителни мерки за осигуряване на надеждност и защита.

9.2 Аутентикация и авторизация

- **Аутентикация** се реализира чрез **ASP.NET Core Identity** и **Cookie Authentication**.
- **Авторизацията** е базирана на роли – достъпът до функционалности и изгледи се контролира чрез Authorize атрибути.
- Всеки потребител трябва да премине през вход с валидни данни, преди да получи достъп до която и да е част от системата.

```
[Authorize(Roles = "Administrator")]
public IActionResult ManageRooms() { ... }
```

9.3 Защита на чувствителни данни

- Паролите се съхраняват **само в хеширан вид** чрез механизма на ASP.NET Identity – дори системният администратор няма достъп до тях.
- ЕГН и други чувствителни полета се предават **само по защитена HTTPS връзка**.
- Достъпът до данни за клиенти, потребители и резервации е ограничен според ролята на текущия потребител.

9.4 CSRF защита (Cross-Site Request Forgery)

Всички форми използват **анти-CSRF токени**, които се валидират при всяко POST действие. Това предотвратява атаки от външни източници, които се опитват да изпратят фалшиви заявки от името на логнат потребител.

```
<form asp-action="Create">
  @Html.AntiForgeryToken()
  ...
</form>
```

9.5 Валидация и защита от некоректни данни

- Всички входни данни минават през сървърна валидация (на базата на DataAnnotations), както и клиентска валидация чрез Razor.
- Системата отхвърля заявки с липсващи, некоректни или опасни стойности.
- При опит за достъп до неразрешени ресурси, потребителят получава HTTP статус 403 (Forbidden).

9.6 Ограничения на достъп

- Ако потребител е маркиран като неактивен (IsActive == false), той не може да влезе в системата.
- При зададена ReleaseDate, системата автоматично блокира акаунта след тази дата.
- Администраторите нямат право да изтриват активни служители с активни резервации – това гарантира историята и проследимостта на действията.

9.7 Други мерки

- Защита срещу **SQL Injection** е осигурена чрез Entity Framework, който използва параметризирани заявки.
- **XSS (Cross-site scripting)** се предотвратява чрез Razor Engine, който автоматично енкодира HTML съдържанието.
- Всички логини и опити за неуспешен достъп могат да бъдат логвани.

10. Използвани технологии и инструменти

10.1 ASP.NET Core 8.0 MVC

Приложението е създадено с помощта на **ASP.NET Core 8.0 MVC**, която е една от най-силните и гъвкави платформи за разработка на уеб приложения. ASP.NET Core е кросплатформен, високопроизводителен фреймуърк, който предоставя много възможности за изграждане на мащабируеми, сигурни и поддържани уеб решения.

Основните предимства на използването на ASP.NET Core 8.0 MVC включват:

- Поддръжка на **Model-View-Controller (MVC)** архитектурата, която осигурява ясно разграничение между логиката на приложението, представянето на данни и потребителския интерфейс.
- Вградена поддръжка за **Dependency Injection (DI)**, което улеснява инжектирането на зависимости и подобрява тестируемостта на кода.
- Поддръжка за **middleware**, което позволява лесно добавяне на функционалности като сигурност, логиране, обработка на грешки и др.

10.2 Entity Framework Core

Entity Framework (EF) Core е използван за управление на данните и създаване на база данни чрез **Code-First** подход. EF Core е обектно-ориентиран достъп до базата данни, който предоставя значителни предимства при изграждането и поддържането на бази данни.

Ключови характеристики на EF Core в проекта:

- **Code-First миграции** – позволяват създаването и актуализирането на базата данни чрез дефиниране на моделите в C# и миграции.
- Автоматично генериране на SQL заявки и оптимизация на изпълнението.
- Използване на **LINQ** за достъп до данни, което осигурява високоефективни и типови заявки.
- Осигурява **многоплатформеност** и възможност за работа с различни бази данни (SQL Server, PostgreSQL, MySQL и др.).

10.3 ASP.NET Core Identity

ASP.NET Core Identity е библиотека за управление на потребители и роли, използвана за аутентикация и авторизация в проекта. Тя предоставя готови механизми за:

- Регистрация и вход на потребители
- Управление на пароли
- Поддръжка за ролево базирана авторизация

Системата използва **IdentityUser** клас, който е разширен със специфични за проекта полета (например, ЕГН, дата на назначаване и др.).

10.4 Razor Pages и Views

Razor Pages се използват за изграждане на динамични уеб страници и представяне на данни. С помощта на **Razor синтаксиса** се интегрират C# код и HTML, за да се изградят персонализирани и динамични интерфейси.

- **Views** са основните компоненти за визуализация на данни, включващи списъци с резервации, клиенти, стаи и потребители.
- **Partial Views** и **Layouts** се използват за споделяне на общи елементи на страниците (например, хедър и футър).

10.5 JavaScript и jQuery

Приложението използва **JavaScript** и **jQuery** за:

- Манипулиране на DOM (Document Object Model) за динамично обновяване на съдържанието на страниците.
- Работа с асинхронни заявки (AJAX) за изпращане на данни без презареждане на страницата.
- Добавяне на допълнителна интерактивност в потребителския интерфейс (например, валидация на форми в реално време).

10.6 Bootstrap

Bootstrap е използван за създаване на отзивчив и мобилно оптимизиран интерфейс. Той предоставя предварително дефинирани стилове и компоненти, които ускоряват разработката на потребителски интерфейс с минимални усилия.

Основни компоненти на Bootstrap:

- **Grid System** – за създаване на адаптивни, многоколонни макети.
- **Form Controls** – за изграждане на стилизирани форми и бутонни елементи.
- **Modals** и **Tooltips** – за показване на допълнителна информация и потвърждения без презареждане на страницата.

10.7 SQL Server

Проектът използва **Microsoft SQL Server** като база данни за съхранение на всички данни, включително потребители, резервации, стаи и клиенти. SQL Server е изключително мощна и стабилна база данни, която предоставя висока производителност, сигурност и надеждност.

Основни функции на SQL Server:

- **Транзакции** – за осигуряване на консистентност и целостта на данните.
- **Поддръжка за индекси** и **оптимизация на заявки** за бързо изпълнение на заявки.
- **Миграции с EF Core** за автоматично генериране и прилагане на промени в схемата на базата данни.

10.8 Git и GitHub

Git е използван за версия на кода, а **GitHub** за хостинг на репозитория и сътрудничество. Системата за контрол на версиите е важна за следене на промените в кода и съвместната работа между екипа.

10.9 Visual Studio

Проектът е разработен в **Visual Studio**, което предоставя пълна интеграция с ASP.NET Core и други инструменти. Visual Studio осигурява:

- Изключителна поддръжка за дебъгване и тестиране на кода.
- Интегрирани инструменти за миграции с Entity Framework.
- Силни функции за интелигентно редактиране на код (например, IntelliSense и Code Navigation).

11. Заключение

Проектът „Мениджър за хотелски резервации“ представлява ефективно решение за управление на хотелски резервации, като използва най-новите технологии и подходи за разработка на уеб приложения. Чрез интегрирането на **ASP.NET Core**, **Entity Framework Core**, и **ASP.NET Core Identity**, системата осигурява висока сигурност, надеждност и лесна разширяемост.

Основните характеристики на системата включват:

- Управление на потребители, роли и достъп;
- Удобен интерфейс за резервации на стаи;
- Система за управление на клиенти и тяхната информация;
- Възможност за изчисляване на дължими суми за резервации и обработка на плащания;
- Пълна поддръжка за CRUD операции върху потребители, стаи, клиенти и резервации;
- Интуитивно уеб приложение с динамичен потребителски интерфейс, базиран на **Razor Pages** и **Bootstrap**.

Проектът е разработен с акцент върху сигурността, като са използвани съвременни практики за защита на данните, включително **аутентикация и авторизация чрез ASP.NET Core Identity**, **валидация на входните данни**, и **защита от атаки като SQL инжекции и XSS**.

Системата е лесна за поддръжка и разширяване благодарение на използването на **Code First** подход за работа с базата данни чрез **Entity Framework**. Възможностите за миграция на данни и поддръжка на различни бази данни гарантират гъвкавост и дългосрочна поддръжка на системата.

Проектът е подходящ за интеграция в различни хотели и хотелски вериги, като осигурява автоматизация и ефективност в управлението на хотелския бизнес.

11.1 Препоръки и бъдещи подобрения

Въпреки че проектът покрива основните изисквания, в бъдеще може да се добавят нови функционалности, като:

- Поддръжка за **мобилни приложения** за по-лесен достъп до системата от страна на клиентите.

- Интеграция с **платежни системи** за автоматизирано плащане на резервации.
- Подобрена **анализ на данни и отчети** за по-добро управление на бизнеса.

Тези допълнения биха увеличили функционалността и конкурентоспособността на системата, като я направят още по-привлекателна за различни търговски обекти в сферата на хотелиерството.