**Dokumentacja aplikacji Hotel Management**

**1. Opis projektu**

Użytkownik wchodząc na stronę, może wybrać pokój znajdujący się w jednym z dwóch hoteli, w opisie hotelu możemy dowiedzieć czy posiada on parking oraz zdobyć dane kontaktowe obsługi. Użytkownik ma możliwość rezerwacji pokoju, w tym celu po przejściu do nowego widoku powinien on podać ilość osób które mają zamiar korzystać z pokoju. Następnie zostanie on przeniesiony do widoku ukazującego listę wolnych pokoi. Aby zarezerwować pokój użytkownik musi podać swoje dane kontaktowe, po zatwierdzeniu akcji rezerwacji użytkownik zostanie przeniesiony na stronę główną. Na stronie głównej znajduje się spisy wszystkich pokoi w dwóch hotelach z informacją który pokój jest wolny a który zajęty. Jeżeli pokój jest zajęty użytkownik ma możliwość podglądu danych rezerwatora. W aplikacji istnie opcja dodanie pokoi edycji pokoi oraz usunięcia pokoi. W aplikacji znajdują się 4 encje 3 z nich połączone są kluczem głównym i znajdują się w relacji: Hotel - pokój -> jeden do wielu Hotel - obsługa -> jeden do wielu Pokój - gosc ( rezerwator) -> jeden do jeden

**2. Wymagania systemowe**

* **.NET SDK**: Wersja 5.0 lub wyższa (zaleca się najnowszą wersję).
* **SQL Server**: SQL Server 2019 lub wyższy, w tym przypadku używamy lokalnej bazy danych (**LocalDB**).
* **Visual Studio**: Wersja 2019 lub 2022 z zainstalowanymi komponentami **ASP.NET** i **Entity Framework**.

**3. Instalacja i uruchomienie aplikacji**

**3.1 Kroki instalacji**

1. **Pobranie repozytorium z GitHub**:
   * Skopiuj repozytorium z GitHub:
2. git clone https://github.com/xwisniax96/RoomBooking
3. **Otwórz projekt w Visual Studio**:
   * Otwórz Visual Studio.
   * Wybierz **Open Project** i wskaż folder z projektem **HotelManagement**.
4. **Zainstalowanie paczek NuGet**: W Visual Studio przejdź do **Tools -> NuGet Package Manager -> Manage NuGet Packages for Solution** i zainstaluj poniższe paczki:
   * Microsoft.EntityFrameworkCore
   * Microsoft.EntityFrameworkCore.SqlServer
   * Microsoft.EntityFrameworkCore.Tools
   * Microsoft.AspNetCore.Identity.EntityFrameworkCore
5. **Konfiguracja połączenia z bazą danych**: W pliku **appsettings.json** skonfiguruj łańcuch połączenia do bazy danych **SQL Server**:

{

"AllowedHosts": "\*",

"ConnectionStrings": {

//"DefaultConnection": "Server=DESKTOP-UU0O7QT;Database=RoomBookingDb;Encrypt=False;TrustServerCertificate=True;Trusted\_Connection=True;MultipleActiveResultSets=true",

"ApplicationDbContext": "Data Source=DESKTOP-UU0O7QT;Initial Catalog=RoomBookingDb;Integrated Security=True;Connect Timeout=30;Encrypt=False;Trust Server Certificate=False;Application Intent=ReadWrite;Multi Subnet Failover=False"

},

"Logging": {

"LogLevel": {

"Default": "Information",

"Microsoft.AspNetCore": "Warning"

}

}

}

1. **Migracje bazy danych**: W konsoli **Package Manager Console** uruchom następujące komendy, aby utworzyć bazę danych:
2. Add-Migration InitialCreate
3. Update-Database
4. **Uruchomienie aplikacji**:
   * W Visual Studio kliknij przycisk **Start** (zielona strzałka) lub naciśnij **Ctrl + F5**, aby uruchomić aplikację.

**4. Wymagania konfiguracyjne**

**4.1 Łańcuch połączenia z bazą danych**

Aplikacja wykorzystuje bazę danych **SQL Server** (LocalDB). Łańcuch połączenia znajduje się w pliku **appsettings.json**:

"ConnectionStrings": {

"ApplicationDbContext": "Data Source=DESKTOP-UU0O7QT;Initial Catalog=RoomBookingDb;Integrated Security=True;Connect Timeout=30;Encrypt=False;Trust Server Certificate=False;Application Intent=ReadWrite;Multi Subnet Failover=False"

},

Jeśli chcesz użyć zewnętrznego serwera SQL, zmień odpowiednio adres serwera i dane uwierzytelniające.

**4.2 Testowi użytkownicy i ich hasła**

Do testowania aplikacji utwórz kilku użytkowników. Poniżej znajdują się przykłady testowych użytkowników:

* **Admin**:
  + **Username**: admin
  + **Password**: Admin123!
  + **Role**: Administrator
* **Zwykły użytkownik**:
  + **Username**: user
  + **Password**: User123!
  + **Role**: Zwykły użytkownik

Testowi użytkownicy mogą zostać dodani za pomocą panelu administracyjnego lub za pomocą skryptu w bazie danych.

**4.3 Autoryzacja użytkowników**

Aplikacja wykorzystuje **ASP.NET Identity** do zarządzania użytkownikami i ich rolami. Użytkownicy mają dwie role:

* **Administrator** – pełen dostęp do aplikacji (może zarządzać hotelami, pokojami i rezerwacjami).
* **Zwykły użytkownik** – dostęp do tworzenia rezerwacji, przeglądania dostępnych pokoi, ale bez możliwości zarządzania danymi.

**5. Opis działania aplikacji z punktu widzenia użytkownika**

Aplikacja **HotelManagement** zapewnia różne funkcjonalności w zależności od roli użytkownika:

**5.1 Rejestracja i logowanie użytkowników**

1. **Rejestracja**: Użytkownicy mogą założyć konto poprzez formularz rejestracji dostępny na stronie głównej aplikacji. Wymagane pola to:
   * **Email**
   * **Hasło**

Po rejestracji użytkownik zostaje przekierowany do strony logowania.

1. **Logowanie**: Użytkownicy logują się, wprowadzając nazwę użytkownika oraz hasło. Po poprawnym zalogowaniu użytkownik zostaje przekierowany na stronę główną aplikacji, gdzie mogą przejść do zarządzania rezerwacjami (dla użytkowników) lub zarządzania hotelami/pokojami (dla administratorów).

**5.2 Działania użytkownika (Zwykły użytkownik)**

Zwykły użytkownik może wykonać następujące operacje:

1. **Przeglądanie dostępnych pokoi**:
   * Użytkownik może wybrać ilość gości i sprawdzić czy dla danej liczby można zamówić pokój. Np. dla 3 gości nie można zamówić pokoju dwuosobowego.
2. **Tworzenie rezerwacji**:
   * System sprawdza dostępność pokoju, a następnie umożliwia dokonanie rezerwacji.
3. **Przeglądanie historii rezerwacji**:
   * Użytkownik może zobaczyć swoje rezerwacje i ich szczegóły.

**5.3 Działania administratora**

Administrator ma dostęp do pełnej funkcjonalności aplikacji, w tym:

1. **Zarządzanie hotelami**:
   * Dodawanie, edytowanie i usuwanie hoteli.
2. **Zarządzanie pokojami**:
   * Dodawanie nowych pokoi, edytowanie istniejących pokoi i usuwanie pokoi.
3. **Zarządzanie rezerwacjami**:
   * Podgląd rezerwacji dokonanych przez użytkowników, anulowanie rezerwacji.
4. **Zarządzanie użytkownikami**:
   * Możliwość nadawania ról użytkownikom (np. zmiana roli zwykłego użytkownika na administratora).

**5.4 Web API**

Aplikacja oferuje również interfejs **Web API** do wykonywania operacji CRUD na encjach takich jak **Hotel**, **Room**, i **Booking**. Poniżej znajduje się przykład, jak wywołać API do zarządzania pokojami:

* **GET /api/rooms** – pobiera listę wszystkich pokoi.
* **POST /api/rooms** – dodaje nowy pokój.
* **PUT /api/rooms/{id}** – edytuje pokój.
* **DELETE /api/rooms/{id}** – usuwa pokój.

**6. Wymagania do uruchomienia aplikacji**

Aby uruchomić aplikację na lokalnej maszynie, należy spełnić następujące wymagania:

1. Zainstalowane **.NET SDK 8.0 lub wyższe**.
2. Zainstalowany **SQL Server** lub użycie **LocalDB** w systemie Windows.
3. Skonfigurowany **łańcuch połączenia do bazy danych** w pliku **appsettings.json**.
4. Przeprowadzenie migracji bazy danych za pomocą poleceń:
5. Add-Migration InitialCreate
6. Update-Database

**7. Podsumowanie**

Aplikacja **HotelManagement** to pełnoprawny system zarządzania hotelami i rezerwacjami, umożliwiający łatwe zarządzanie danymi o hotelach, pokojach i rezerwacjach. Dzięki zastosowaniu technologii ASP.NET MVC i Entity Framework aplikacja jest skalowalna, a interfejs użytkownika zapewnia wygodne narzędzia do pracy.