UNIWERSYTET
WSB MERITO
POZNAŃ

Dokumentacja projektowa systemu informatycznego do rezerwowania i zarządzania wizytami w salonie fryzjerskim

Autorzy:

Grzegorz Olejniczak, grupa: K32

Łukasz Molenda, grupa: K32

Spis treści

1. Ogólne informacje o projekcie

System rezerwacji wizyt u fryzjera umożliwia użytkownikom rezerwowanie wizyt u fryzjerów w wybranym salonie fryzjerskim. Klienci mogą rejestrować się, logować, przeglądać dostępne usługi, oraz rezerwować terminy wizyt. System pozwala także na edytowanie danych użytkownika oraz anulowanie lub modyfikowanie istniejących rezerwacji. Salon ma możliwość zarządzania swoimi usługami, przeglądania rezerwacji. Fryzjerzy mają możliwość podglądu przypisanych do nich wizyt, dodania nowego użytkownika z rolą fryzjera, dodawania usług do systemu edytowanie ich oraz usuwanie. Wszystkie dane są przechowywane w bazie danych.

2. Specyfikacja wykorzystanych technologii

- Backend: ASP.NET 8.0
- Frontend: HTML, CSS, JavaScript
- Framework frontendowy: Bootstrap 5.33
- Baza danych: Microsoft SQL Server
- Autentykacja i autoryzacja: ASP.NET Identity
- Inne: Entity Framework Core (do zarządzania bazą danych)

3. Instrukcja pierwszego uruchomienia

- **Pobierz repozytorium:**
  - Skopiuj repozytorium z GitHub: https://github.com/GrzegorzOlejniczak/HairSalonApp.
- **Wypakuj plik ZIP:**
  - Jeśli został pobrany plik ZIP (HairSalonApp-master.zip), wypakuj go w odpowiednim miejscu na swoim komputerze i uruchom plik HairSalon.sln w Visual Studio
- **Migracja bazy danych:**
  - Przed uruchomieniem aplikacji wykonaj migrację bazy danych:
    - Otwórz **Konsolę Menedżera Pakietów** w Visual Studio: **Narzędzia > Menedżer pakietów NuGet > Konsola Menedżera Pakietów**.
    - W konsoli wykonaj następujące polecenia: **Add-Migration Initial2** oraz **Update-Database**
- **Rejestracja i logowanie:**
  - **Dla zwykłych użytkowników (klientów):** Zarejestruj się za pomocą formularza rejestracji, podając swoje dane (email, hasło, itd.).
  - **Aby zalogować się na konto fryzjera:** Użyj poniższych poświadczeń:
    - **Adres email:** jan.kowalski@gmail.com
    - **Hasło:** zaq1@WSX

4. Opis struktury projektu

- **wwwroot** - katalog zawierający pliki statyczne, które są bezpośrednio dostępne dla klienta. Obejmuje:
    - **CSS:** Arkusze stylów do zarządzania wyglądem aplikacji**.**
    - **JS:** Skrypty JavaScript, które odpowiadają za interaktywność aplikacji.
    - **Images:** Obrazy, np. loga, zdjęcia użytkowników itp.
- **Areas -** obszary w aplikacji są używane do organizowania logiki aplikacji w różne sekcje, co pozwala na lepszą modularność i łatwiejsze zarządzanie kodem:
    - **Hairdresser** - odpowiada za rejestrację i zarządzanie fryzjerami. Zawiera widoki, modele i kontrolery dedykowane fryzjerom.
    - **Identity** - obsługuje wszystkie funkcje związane z tożsamością użytkowników, takie jak logowanie, rejestracja, zmiana hasła, potwierdzenie adresu email i zarządzanie rolami użytkowników
- **Controllers** - zawiera logikę aplikacji odpowiedzialną za obsługę żądań HTTP. Każdy kontroler odpowiada za konkretne zasoby (np. wizyty, użytkownicy, usługi) i jest odpowiedzialny za przekazywanie danych do widoków oraz obsługę akcji (np. dodawanie, edytowanie, usuwanie). Przykład kontrolera:
- **Models** - definicje modeli danych, które reprezentują strukturę przechowywanych informacji w aplikacji. Modele są wykorzystywane do mapowania danych na bazę danych oraz do przekazywania danych między warstwami aplikacji**.**
- **Views** - Widoki Razor Pages używane do prezentacji danych użytkownikowi. Są to pliki HTML z kodem C#, które dynamicznie renderują dane na stronie.

5. Modele użyte w projekcie

W projekcie wykorzystano następujące modele:

- **ApplicationUser -** model użytkownika, który rozszerza wbudowaną klasę IdentityUser dostarczaną przez ASP.NET Core Identity. Służy do reprezentacji użytkowników w systemie, takich jak klienci oraz pracownicy salonu fryzjerskiego. Oprócz standardowych pól dziedziczonych z IdentityUser, model dodaje dodatkowe właściwości specyficzne dla aplikacji. Pola w modelu:
  - **Firstname:**
    a) **Typ:** string
    b) **Opis:** Imię użytkownika
  - **Lastname:**
    a) **Typ: string**
    b) **Opis:** nazwisko użytkownika
  - **Email:**
    a) **Typ: string**
    b) **Opis:** Ścieżka do zdjęcia profilowego użytkownika
- **Appointment** – model wizyty. Przechowuje informacje o szczegółach wizyty, takich jak powiązania z klientem, fryzjerem, a także datę i godzinę wizyty. Wykorzystane pola w modelu:
  - **Id**:
    a) **Typ**: int
    b) **Opis**: unikalny identyfikator wizyty, automatycznie generowany przez system.
  - **UserId**:
    a) **Typ**: string
    b) **Opis:** Identyfikator użytkownika, który dokonał rezerwacji wizyty
    c) **Walidacja:** [Required] – pole obowiązkowe
  - **User**:
    a) **Typ**: ApplicationUser?
    b) **Opis**: obiekt reprezentujący użytkownika. Jest to pole nawigacyjne umożliwiające dostęp do szczegółowych danych użytkownika.
    c) **Walidacja**: [ValidateNever] – wyłącza walidację tego pola
  - **HairdresserId**:
    a) **Typ:** string
    b) **Opis:** identyfikator fryzjera wykonującego usługę
    c) **Walidacja:** [Required] – pole obowiązkowe
  - **Hairdresser**:
    a) **Typ**: ApplicationUser?
    b) **Opis**: obiekt reprezentujący fryzjera wykonującego usługę. Jest to pole nawigacyjne umożliwiające dostęp do danych fryzjera
    c) **Walidacja**: [ValidateNever] – wyłącza walidację tego pola. [Display(Name = „Fryzjer”)] – wyświetlana nazwa w fomularzach
  - **AppointmentDate**:

a) **Typ**: DateTime
b) **Opis**: Data i godzina wizyty
c) **Walidacja**: [Required(ErrorMessage = "Upewnij się, że został wybrany dzień oraz godzina wizyty!")] - pole jest obowiązkowe z niestandardowym komunikatem błędu.

- **ServiceId**:
  a) **Typ**: int?
  b) **Opis**: identyfikator usługi wybranej podczas rezerwacji
  c) **Walidacja**: [Required(ErrorMessage = "Upewnij się, że została wybrana usługa!")] - Pole jest obowiązkowe z niestandardowym komunikatem błędu.
  d)
- **Service**:
  a) **Typ**: obiekt reprezentujący wybraną usługę. Jest to pole nawigacyjne umożliwiające dostęp do danych o usłudze.
  b) **Walidacja**: [ValidateNever] – wyłącza walidację dla tego pola

- **Service** – reprezentuje usługę oferowaną w salonie fryzjerskim. Model zawiera informacje takie jak nazwa usługi, czas jej trwania oraz koszt. Jest używany w systemie do zarządzania katalogiem usług. Wykorzystuje pola**:**
  - **Id:**
    a) **Typ:** int
    b) **Opis:** unikalny identyfikator usługi. Generowany automatycznie przez system
  - **Name:**
    a) **Typ:** string
    b) **Opis:** nazwa usługi
    c) **Walidacja:** [Required(ErrorMessage = "Nazwa jest wymagana.")] - Pole jest obowiązkowe z niestandardowym komunikatem błędu.
  - **Duration:**
    a) Typ: int
    b) **Opis:** czas trwania usługi w godzinach
    c) Walidacja: [Range(1, 10, ErrorMessage = "Czas trwania musi być liczbą od 1 do 10.")] – wartość musi być w zakresie od 1 do 10, co ogranicza możliwy czas trwania usługi
  - **Price** :
    a) **Typ**: decimal
    b) **Opis**: koszt usługi wyrażony w walucie
    c) **Walidacja**:
      - [Required(ErrorMessage = "Cena jest wymagana.")] - Pole jest obowiązkowe z niestandardowym komunikatem błędu.
      - [Range(0, double.MaxValue, ErrorMessage = "Cena musi być większa lub równa 0.")] - Cena musi być nieujemna.

6. Kontrolery i metody użyte w projekcie

W projekcie wykorzystano następujące kontrolery oraz metody:

- AppointmentController – kontroler odpowiedzialny za zarządzanie wizytami w salonie fryzjerskim. Korzysta z metod:
    - **GetReservedHours**:
        a) **Metoda HTTP**: GET
        b) **Parametry**:
            - String hairdresserId – identyfikator fryzjera
            - Int year: rok daty wizyty
            - Int month: miesiąc daty wizyty
            - Int dayl dzień daty wizyty
            - String hour: wybrany godzina wizyty w formacie ”HH:mm”
        c) **Opis**: Zwraca dostępne usługi oraz listę godzina zarezerwowanych dla danego fryzjera w podanym dniu.
            - Sprawdza, czy wybrana godzina jest poprawna
            - Pobiera wizyty fryzjera w danym dniu i sprawdza, które godziny są zajęte
            - Filtruje usługi, które mogą być wykonane o wskazanej godzinie
        d) **Zwracane dane**: JSON zwracający dostępne usługi (availableServices) i listę zablokowanych godzin (blockedHours)
    - **Index**:
        a) **Metoda HTTP:** GET
        b) **Parametry:** brak
        c) **Opis:** wyświetla listę wizyt dla zalogowanego użytkownika:
            - Fryzjerzy widzą swoje przypisane wizyty
            - Klienci widzą wizyty, które zarezerwowali
        d) **Zwracane dane:** widok z listą wizyt
    - **Details:**
        a) **Metoda HTTP:** GET
        b) **Parametry:** int? Id – identyfikator wizyty
        c) **Opis:** wyświetla szczegóły wizyty o podanym indentyfikatorze. Jeśli wizyta nie istnieje zwraca błąd 404
        d) **Zwracane dane:** widok szczegółowy wizyty
    - **Create (GET):**
        a) **Metoda HTTP:** GET
        b) **Parametry:** brak
        c) **Opis:** wyświetla formularz tworzenia nowej wizyty. Pobiera listę fryzjerów i przypisuje aktualnie zalogowanego użytkownika jako klienta
        d) **Zwracane dane:** widok formularza
    - **Create (POST):**
        a) **Metoda HTTP:** POST
        b) **Parametry:** Appointment appointment – obiekt zawierający szczegóły nowej wizyty
        c) **Opis:** dodaje nową wizytę do systemu. Waliduje datę, godzinę i usługę. Sprawdza konflikty z istniejącymi wizytami

        d) **Zwracane dane:** przekierowanie do listy wizyt Index w przypadku poprawnie przesłanych danych. Widok formularza z komunikatami walidacyjnymi w przypadku błędów.
    - **Edit (GET):**
        a) **Metoda HTTP:** GET
        b) **Parametry:** int? – identyfikator wizyty
        c) **Opis:** wyświetla formularz edycji wizyty. Sprawdza, czy użytkownik jest właścicielem wizyty lub przypisanym fryzjerem. Pobiera listę fryzjerów i usług
        d) **Zwracane dane:** widok formularza
    - **Edit (POST):**
        a) **Metoda HTTP:** POST
        b) **Parametry:**
            - int id – identyfikator wizyty
            - Appointment appointment – zaktualizowane dane wizyty
        c) **Opis:** aktualizuje wizytę o podanym identyfikatorze. Waliduje datę, godzinę i konflikty z innymi wizytami. Sprawdza uprawnienia użytkownika do edycji.
        d) **Zwracace dane:** Przekierowanie do listy wizyt **Index** w przypadku poprawnie przesłanych danych. Widok formularza z błędami walidacji w przypadku niepowodzenia.
    - **Delete (GET):**
        a) **Metoda HTTP**: GET
        b) **Parametry**: int id – identyfikator wizyty
        c) **Opis**: Wyświetla szczegóły wizyty przed jej usunięciem. Sprawdza, czy użytkownik jest właścicielem wizyty lub przypisanym fryzjerem.
        d) **Zwracane dane**: Widok szczegółów wizyty
    - **DeleteConfirmed**:
        a) **Metoda HTTP:** POST
        b) **Parametry:** int id – identyfikator wizyty
        c) **Opis:** usuwa wizytę o podanym identyfikatorze. Sprawdza uprawnienia użytkownika przed usunięciem
        d) **Zwracana dane:** przekierowanie do listy wizyt **Index**
    - **AppointmentExists:**
        a) **Metoda HTTP:** brak
        b) **Parametry: int id –** identyfikator wizyty
        c) **Opis:** sprawdza, czy wizyta o podanym id istnieje w bazie danych
        d) **Zwracane dane:** bool (true, jeśli istnieje; w przeciwnym razie false)
- **ServicesController** – odpowiada za zarządzanie usługami w aplikacji.
    - **Index:**
        a) **Metoda HTTP:** GET
        b) **Parametry:** brak
        c) **Opis:** wyświetla listę wszystkich dostępnych usług
        d) **Zwracane dane:** widok z listą usług

- **Details:**
    a) **Metoda HTTP:** GET
    b) **Parametry:** int? id – opcjonalny, identyfikator usługi
    c) **Opis:** wyświetla szczegóły pojedynczej usługi na podstawie jej identyfikatora. Jeśli identyfikator nie jest podany lub usługa nie istnieje, zwraca NotFound.
    d) **Zwracane dane:** widok z detalami usługi (Service).
- **Create (GET):**
    a) **Metoda HTTP:** GET
    b) **Parametry:** brak
    c) **Opis:** Wyświetla formularz do tworzenia nowej usługi. Dostępne wyłącznie dla użytkowników z rolą **Hairdresser**.
    d) **Zwracane dane:** widok formularza tworzenia usługi
- **Create (POST):**
    a) **Metoda HTTP:** POST
    b) **Parametry:** Service service – wiązany model zawierający dane usługi Id, Name, Duration, Price
    c) **Opis:** dodaje nową usługę do bazy danych, po wcześniejszej walidacji. Duration jest konwertowany z minut na sekundy przed zapisaniem.
    d) **Zwracane dane:** przekierowanie do akcji Index w przypadku sukcesu lub ten sam widok z błędami walidacji.
- **Edit (GET):**
    a) **Metoda HTTP:** GET
    b) **Parametry:** int? id (opcjonalny, identyfikator usługi)
    c) **Opis:** 2yświetla formularz edycji istniejącej usługi. Jeśli identyfikator nie jest podany lub usługa nie istnieje, zwraca NotFound. Dostępne wyłącznie dla użytkowników z rolą **Hairdresser**.
    d) **Zwracane dane:** Widok formularza edycji usługi (Service).
- **Edit (POST):**
    a) **Metoda HTTP:** POST
    b) **Parametry:** int id (identyfikator usługi), Service service (model zaktualizowanych danych)
    c) **Opis:** aktualizuje dane istniejącej usługi w bazie danych po walidacji.
    d) **Zwracane dane:** Przekierowanie do akcji Index w przypadku sukcesu lub ten sam widok z błędami walidacji.
- **Delete (GET):**
    a) **Metoda HTTP:** GET
    b) **Parametry:** int? id (opcjonalny, identyfikator usługi)
    c) **Opis:** wyświetla stronę z potwierdzeniem usunięcia usługi. Jeśli identyfikator nie jest podany lub usługa nie istnieje, zwraca NotFound. Dostępne wyłącznie dla użytkowników z rolą Hairdresser.
    d) **Zwracane dane:** widok potwierdzenia usunięcia usługi (Service).
- **Delete (POST):**
    a) **Metoda HTTP:** POST
    b) **Parametry:** int id (identyfikator usługi)

        c) **Opis:** usuwa usługę z bazy danych po potwierdzeniu. Jeśli usługa nie istnieje, nic nie zostaje usunięte.
        d) **Zwracane dane:** przekierowanie do akcji Index.
    - **DeleteConfirmed:**
        a) **Metoda HTTP:** POST
        b) **Parametry:** int id (identyfikator usługi)
        c) **Opis:** funkcja obsługująca faktyczne usunięcie usługi
        d) **Zwracane dane:** przekierowanie do akcji Index.
    - **ServiceExists:**
        a) **Metoda HTTP:** brak
        b) **Parametry:** int id – identyfikator usługi
        c) **Opis:** sprawdza, czy usługa o podanym identyfikatorze istnieje w bazie danych
        d) **Zwracane dane:** bool (true, jeśli istnieje; w przeciwnym razie false)
- **HomeController** - odpowiedzialny za obsługę podstawowych widoków strony, takich jak strona główna oraz obsługa błędów.
    - **Index:**
        a) **Metoda HTTP: GET**
        b) **Opis:** renderuje główną stronę aplikacji. Jest to domyślny widok, który jest wyświetlany, gdy użytkownik, odwiedza stronę aplikacji
        c) **Zwracace dane:** widok Index
    - **Error:**
        a) **Metoda HTTP:** GET
        b) **Parametry:** int id – identyfikator usługi
        c) **Opis:** renderuje stronę błędu, zawierającą informacje o problemie, takie jak identyfikator żądania (RequestId). Używane, gdy wystąpi błąd w aplikacji.
        d) **Zwracane dane:** Widok Error, który zawiera model ErrorViewModel z identyfikatorem żądania.
- **AboutController:**
    - **Index:**
        a) **Metoda HTTP: GET**
        b) **Opis:** renderuje widok z listą użytkowników, którzy mają przypisaną rolę "Hairdresser". Wykorzystuje UserManager<ApplicationUser> do pobrania wszystkich użytkowników o tej roli.
        c) **Zwracane dane:** widok Index, w którym przekazywana jest lista fryzjerów (hairdressersList), czyli użytkowników przypisanych do roli "Hairdresser".

7. Opis systemu użytkowników

W aplikacji zarządzanie użytkownikami oraz ich rolami odbywa się przy pomocy systemu tożsamości ASP.NET Identity. System pozwala na tworzenie użytkowników, przypisywanie im ról oraz kontrolowanie dostępu do różnych zasobów aplikacji na podstawie ról.

Role w systemie

W aplikacji występują dwie role:

a) User (domyślna rola)
   - **Przypisanie roli**: nadawana jest użytkownikowi automatycznie podczas rejestracji
   - **Uprawnienia**:
     - możliwość zapisywania się na wizyty w salonie
     - możliwość edytowania swoich zapisów np. zmiana daty usługi, czy fryzjera
     - możliwość anulowania wcześniej zaplanowanej wizyty
   - **Ograniczenia**: użytkownicy tej roli nie mają uprawnień do edytowania usług, zarządzania pracownikami, czy przeglądania danych innych użytkowników.

b) **Hairdresser**:
   - **Przypisanie roli**: rola może zostać przydzielana przez innego użytkownika z rolą HAIRDRESSER
   - **Uprawnienia**:
     - użytkownicy z tą rolą mogą zarządzać wizytami przypisanymi do nich
     - możliwość dodawania usług do oferty salonu
     - dodawanie nowych osób do zespołu

8. Krótka charakterystyka najciekawszych funkcjonalności

Dynamiczny kalendarz rezerwowania wizyt w salonie:

- **Dni i godziny:**
  - Kalendarz wyświetla dni bieżącego miesiąca.
  - Blokowanie weekendów i przeszłych dni:
    - W soboty i niedziele dni są automatycznie zablokowane, aby nie można było ich wybrać.
    - Przeszłe dni są również zablokowane, aby uniemożliwić rezerwację wizyt w przeszłości.
  - Dostępne godziny:
    - Godziny dostępne do rezerwacji są wyświetlane w kalendarzu. Zajęte godziny są wykreślone, aby użytkownik wiedział, które terminy są już zajęte.
- **Wybór Usługi:**
  - Po dokonaniu wyboru dnia i godziny, system automatycznie ładuje dostępne usługi w danym terminie.
- **Aktualizacja daty spotkania:**
  - Po wybraniu odpowiedniego dnia i godziny, system automatycznie uzupełnia pole z datą spotkania.
- **Integracja z Backendem:**
  - Kalendarz jest zintegrowany z backendem, który zarządza danymi o rezerwacjach.
  - Backend pobiera dane o już zarezerwowanych godzinach i wprowadza je w system, blokując je w kalendarzu.
- **Wybór Fryzjera:**
  - Po dokonaniu wyboru, system automatycznie dostosowuje dostępność godzin do preferencji fryzjera, blokując godziny, które nie są dostępne w jego grafiku.