#### ****Dokumentacja projektu CalendarOfVisits****

#### ****Tytuł Projektu****

System do raportowania i rejestrowania wizyt przedstawicieli handlowych u klientów

**Opis działania projektu**

Aplikacja internetowa zbudowana w technologii ASP.NET MVC, która umożliwia przedstawicielom handlowym raportowanie i rejestrowanie wizyt u klientów. System wspiera zarządzanie relacjami z klientami poprzez:

* **Rejestrację wizyt:** możliwość dodawania informacji o odwiedzanych klientach, daty, czasu i celu wizyty.
* **Integrację z kalendarzem:** użytkownicy mogą planować wizyty z wykorzystaniem kalendarza, który pozwala na:
  + Dodawanie nowych spotkań
  + Wyświetlanie harmonogramu na dzień, tydzień lub miesiąc.
* **Raportowanie:** generowanie raportów o odbytych wizytach na podstawie raportu przedstawiciela.
* **Panel administracyjny:** Panel dla menedżerów, pozwalający na przeglądanie wizyt oraz ocenę działań przedstawicieli handlowych.
* **Konta użytkowników:** każdy użytkownik posiada indywidualne konto dostępowe .

Aplikacja działa w przeglądarce internetowej, obsługując zarówno komputery stacjonarne, jak i urządzenia mobilne. System korzysta z relacyjnej bazy danych do przechowywania informacji o wizytach, danych użytkowników oraz szczegółów związanych z planowanymi spotkaniami.

**Użyte technologie:**

* - .NET 8
* - ASP.NET
* - Entity Framework
* - iText7
* - Microsoft SQL Server

**Instrukcja uruchomienia projektu:**

* Otwórz plik CalendarOfVisits.sln.
* Wpisz komendę Update-Database w konsoli menedżera pakietów.
* Po zmianie roli przez administratora, wszystkie wcześniej otwarte sesje użytkownika muszą zostać zresetowane. Należy wylogować się i zalogować ponownie, aby zmiany zaczęły obowiązywać.

**Struktura projektu:**

* Controllers – Zawiera kontrolery, które obsługują logikę aplikacji
* Models – Przechowuje dane aplikacji
* Views – Zawiera widoki, które odpowiadają za wygląd strony
* Views/Shared – Przechowuje wspólne elementy widoków, jak nagłówek
* wwwroot – Zawiera pliki statyczne, jak CSS, JavaScript, obrazy.
* Program.cs – Konfiguracja serwisów i aplikacji.

**Modele w projekcie:**

* WebAPIEvent – DTO używany w warstwie REST API do przesyłania danych o wydarzeniu w kalendarzu:
  + id – identyfikator wydarzenia.
  + text – tekst wydarzenia.
  + purpose – cel wydarzenia.
  + created\_by – osoba, która stworzyła wydarzenie.
  + rating – ocena wydarzenia.
  + start\_date – data rozpoczęcia wydarzenia.
  + end\_date – data zakończenia wydarzenia.
* UserRoleViewModel – DTO do wyświetlania użytkowników i ich przypisanych ról:
  + UserId – identyfikator użytkownika.
  + UserName – nazwa użytkownika.
  + Role – rola przypisana użytkownikowi.
* SchedulerEvent – Encja reprezentująca wydarzenie, zapisywana w bazie danych:
  + Id – identyfikator wydarzenia.
  + Description – opis wydarzenia.
  + Purpose – cel wydarzenia.
  + CreatedBy – osoba, która stworzyła wydarzenie.
  + Rating – ocena wydarzenia.
  + StartDate – data rozpoczęcia wydarzenia.
  + EndDate – data zakończenia wydarzenia.

**Szczegółowy opis funkcjonalności:**

Aplikacja umożliwia rejestrację dla każdego. Każdy nowo utworzony użytkownik ma początkowo dostęp tylko do strony głównej. W ramach procesu inicjalizacji aplikacji tworzony jest użytkownik o roli admin, który ma pełne uprawnienia do zarządzania użytkownikami i przypisywania im ról. Istnieją trzy role w systemie: admin, menedżer, oraz przedstawiciel handlowy.

* **Rola Admin** - Admin ma również możliwość przypisywania ról do użytkowników, w tym ról managera i przedstawiciela handlowego.
* **Rola Sales Representative** - Użytkownicy o roli przedstawiciel handlowy mają dostęp do własnego kalendarza, w którym mogą dodawać, edytować i usuwać swoje wizyty. Każda wizyta dodana przez przedstawiciela handlowego zawiera szczegóły, takie jak data, godzina, tytuł oraz cel wizyty.
* **Rola Manager** - Użytkownicy o roli manager mają możliwość przeglądania kalendarzy przedstawicieli handlowych oraz oceniania dodanych wizyt. Manager może również wyświetlać listę przedstawicieli handlowych, którzy dodali co najmniej jedną wizytę do systemu, w sekcji Sales Representatives. Ponadto, manager ma możliwość generowania raportów w formacie PDF, które zawierają szczegółowe informacje na temat wizyt przedstawicieli handlowych.

**Kontrolery w projekcie:**

**1. CalendarController**

**Index (GET)**

* **Parametry**: Brak
* **Opis działania**: Metoda ta umożliwia użytkownikowi (z rolą "Manager" lub "Sales Representative") wyświetlenie widoku. Wewnątrz metody pobierane są role użytkownika, a następnie przypisywana jest rola do ViewBag, co umożliwia wyświetlenie jej w widoku.

**2. HomeController**

**Index (GET)**

* **Parametry**: Brak
* **Opis działania**: Wyświetla główny widok aplikacji. Prosty kontroler bez logiki biznesowej.

**3. SalesRepController**

**Index (GET)**

* **Parametry**: Brak
* **Opis działania**: Metoda ta pobiera unikalnych użytkowników, którzy stworzyli wydarzenia w kalendarzu, i zwraca je w widoku.
* **Zwracane dane**: Lista użytkowników (sprzedawców) na podstawie danych w bazie (pobrane z tabeli SchedulerEvent).

**GeneratePdf (GET)**

* **Parametry**:
  + username (string): Nazwa użytkownika, dla którego mają być wygenerowane dane.
  + fromDate (string): Data początkowa zakresu.
  + toDate (string): Data końcowa zakresu.
* **Opis działania**: Metoda generuje plik PDF zawierający raport z wydarzeniami dla podanego użytkownika w określonym przedziale czasowym. Wydarzenia są wyświetlane z informacjami o kliencie, celu wizyty, czasie rozpoczęcia i zakończenia wizyty.
* **Zwracane dane**: Plik PDF zawierający dane o wydarzeniach. Typ odpowiedzi to application/pdf.

**4. SchedulerController**

**Get (GET)**

* **Parametry**:
  + from (DateTime): Data początkowa zakresu.
  + to (DateTime): Data końcowa zakresu.
* **Opis działania**: Pobiera wydarzenia z kalendarza dla użytkowników z rolą "Manager" lub "Sales Representative" w podanym zakresie dat. Jeśli użytkownik ma rolę "Manager", widzi wszystkie wydarzenia, natomiast użytkownik z rolą "Sales Representative" może zobaczyć tylko wydarzenia, które stworzył.
* **Zwracane dane**: Lista obiektów typu WebAPIEvent odpowiadająca wydarzeniom w bazie danych.

**Get(int id) (GET)**

* **Parametry**:
  + id (int): Identyfikator wydarzenia.
* **Opis działania**: Pobiera szczegóły jednego wydarzenia z kalendarza na podstawie jego identyfikatora.
* **Zwracane dane**: Obiekt WebAPIEvent z danymi o wydarzeniu.

**EditSchedulerEvent (PUT)**

* **Parametry**:
  + id (int): Identyfikator wydarzenia.
  + webAPIEvent (WebAPIEvent): Dane aktualizowanego wydarzenia.
* **Opis działania**: Zaktualizowanie szczegółów wydarzenia w bazie danych (np. opis, cel wizyty, daty).
* **Zwracane dane**: Określa, że operacja została zakończona pomyślnie i wydarzenie zostało zaktualizowane.

**CreateSchedulerEvent (POST)**

* **Parametry**:
  + webAPIEvent (WebAPIEvent): Dane nowego wydarzenia.
* **Opis działania**: Tworzy nowe wydarzenie w kalendarzu. Wydarzenie jest przypisane do aktualnie zalogowanego użytkownika.
* **Zwracane dane**: Identyfikator nowo utworzonego wydarzenia oraz status operacji.

**DeleteSchedulerEvent (DELETE)**

* **Parametry**:
  + id (int): Identyfikator wydarzenia do usunięcia.
* **Opis działania**: Usuwa wydarzenie z kalendarza o podanym identyfikatorze.
* **Zwracane dane**: Status operacji (wydarzenie zostało usunięte).

**5. UserController**

**Index (GET)**

* **Parametry**: Brak
* **Opis działania**: Pobiera listę wszystkich użytkowników z bazy danych i ich przypisane role. Wyświetla je w widoku.
* **Zwracane dane**: Lista użytkowników z przypisanymi rolami.

**AssignRole(string userId) (GET)**

* **Parametry**:
  + userId (string): Identyfikator użytkownika, którego rolę chcemy przypisać lub zmienić.
* **Opis działania**: Wyświetla formularz przypisania roli dla użytkownika. Pobiera szczegóły użytkownika z bazy danych oraz aktualnie przypisaną rolę.
* **Zwracane dane**: Widok z formularzem przypisania roli.

**AssignRole(UserRoleViewModel model) (POST)**

* **Parametry**:
  + model (UserRoleViewModel): Obiekt zawierający informacje o użytkowniku i roli, którą chcemy przypisać.
* **Opis działania**: Przypisuje nową rolę użytkownikowi, usuwając poprzednią rolę.
* **Zwracane dane**: W zależności od rezultatu operacji, zwraca komunikat o powodzeniu lub błędzie.

Autorzy:

Hubert Jagodziński

Piotr Herzog