**Dokumentacja techniczna projektu EventBookingApp**

**Tytuł projektu**

EventBookingApp

**Autor projektu**

Piotr Chwałczyński

**Krótki opis działania projektu**

EventBookingApp to prosta aplikacja do zarządzania wydarzeniami, pozwalająca użytkownikom przeglądać listę wydarzeń, filtrować je i wyszukiwać po nazwie. Dla użytkowników zalogowanych dostępna jest opcja dodawania nowych wydarzeń i eksportowania listy wydarzeń do pliku CSV.

**Specyfikacja wykorzystanych technologii**

* **.NET wersja 6.0** - platforma do budowy aplikacji webowych.
* **Baza danych SQLite** - lekka baza danych do zarządzania danymi wydarzeń.

**Instrukcje pierwszego uruchomienia projektu**

1. Otworzyć konsolę cmd/terminal i przejść do folderu projektowego.
2. Wykonać polecenie "dotnet ef database update” w celu zainicjalizowania bazy danych.
3. Wykonać polecenie „dotnet build” w celu zbudowania projektu.
4. Wykonać polecenie „dotnet run” w celu uruchomienia serwera startowego.
5. Przejść pod adres wskazany w konsoli cmd/terminalu.

**Opis struktury projektu**

Projekt składa się z następujących głównych elementów:

* **Controllers**: Kontrolery obsługujące logikę aplikacji.
* **Data:** Plik kontekstu bazy danych.
* **Migrations:** Pliki migracji bazy danych.
* **Models**: Definicje modeli danych używanych w aplikacji.
* **ViewModels**: Modele pomocnicze do przekazywania danych pomiędzy kontrolerami a widokami.
* **Views**: Widoki generujące interfejs użytkownika.

**Modele**

**Event - reprezentuje wydarzenie w projekcie.**

**Pola:**

* **Id**: Klucz główny, unikalny identyfikator wydarzenia.
* **Title**: Nazwa wydarzenia (wymagane, maksymalnie 100 znaków).
* **Date**: Data i czas wydarzenia (wymagane, musi wskazywać przyszłą datę).
* **Description**: Opis wydarzenia (wymagane).
* **AvailableSeats**: Liczba dostępnych miejsc (wymagane, wartość musi być między 1 a 1000).
* **UserId**: Identyfikator użytkownika przypisanego do wydarzenia.

**ErrorViewModel - jest wykorzystywany do obsługi błędów w aplikacji.**

**Pola:**

* **RequestId**: Identyfikator żądania, który pomaga w identyfikacji problemów.
* **ShowRequestId**: Właściwość logiczna, która zwraca true, jeśli RequestId nie jest pusty.

**Kontrolery**

**AccountController**

Ten kontroler obsługuje funkcje związane z rejestracją, logowaniem i wylogowaniem użytkowników.

* **Login (GET)**  
  **Parametry**:
  + returnUrl (opcjonalny, typ string, domyślna wartość /)  
    **Opis**:  
    Wyświetla formularz logowania. Jeśli użytkownik jest już zalogowany, zostaje przekierowany do strony głównej.
* **Login (POST)**  
  **Parametry**:
  + username (wymagany, typ string)
  + password (wymagany, typ string)
  + returnUrl (opcjonalny, typ string, domyślna wartość /)  
    **Opis**:  
    Próbuje zalogować użytkownika przy użyciu podanych danych. W przypadku sukcesu przekierowuje użytkownika na returnUrl lub stronę główną.
* **Register (GET)**  
  **Opis**:  
  Wyświetla formularz rejestracji dla nowych użytkowników.
* **Register (POST)**  
  **Parametry**:
  + RegisterViewModel model (wymagany, zawiera dane wprowadzone w formularzu rejestracji)  
    **Opis**:  
    Rejestruje nowego użytkownika. Jeśli proces rejestracji się powiedzie, przekierowuje na stronę logowania. W przypadku błędów walidacji wyświetla je na ekranie.
* **Logout (POST)**  
  **Opis**:  
  Wylogowuje obecnie zalogowanego użytkownika i przekierowuje go na stronę logowania.

**ErrorController**

Ten kontroler obsługuje strony błędów, takie jak "Nie znaleziono" i "Brak autoryzacji".

* **NotFound (GET)**  
  **Opis**:  
  Wyświetla stronę błędu o kodzie **404** ("Nie znaleziono"). Ustawia kod statusu odpowiedzi HTTP na **404** i renderuje widok dedykowany dla tego błędu.
* **Unauthorized (GET)**  
  **Opis**:  
  Wyświetla stronę błędu o kodzie **403** ("Brak autoryzacji"). Ustawia kod statusu odpowiedzi HTTP na **403** i renderuje widok dedykowany dla tego błędu.

**EventsController**

Ten kontroler obsługuje zarządzanie wydarzeniami, w tym ich wyświetlanie, tworzenie, edytowanie, usuwanie oraz eksportowanie do pliku CSV. Wszystkie akcje są dostępne wyłącznie dla zalogowanych użytkowników (**[Authorize]**).

* **Index (GET)**  
  **Parametry**: Brak.  
  **Opis**:  
  Wyświetla listę wydarzeń przypisanych do aktualnie zalogowanego użytkownika na podstawie identyfikatora użytkownika (**UserId**). Jeśli użytkownik nie jest zalogowany, zwraca błąd **401 Unauthorized**.
* **Create (GET)**  
  **Parametry**: Brak.  
  **Opis**:  
  Wyświetla formularz do tworzenia nowego wydarzenia. Wymaga autoryzacji użytkownika.
* **Create (POST)**  
  **Parametry**:
  + title (typ string) – Tytuł wydarzenia.
  + description (typ string) – Opis wydarzenia.
  + date (typ DateTime) – Data i czas wydarzenia.
  + availableSeats (typ int) – Liczba dostępnych miejsc.  
    **Opis**:  
    Tworzy nowe wydarzenie przypisane do zalogowanego użytkownika. Waliduje dane wejściowe, a w przypadku błędów zwraca widok formularza z wprowadzonymi danymi. W przypadku poprawnych danych zapisuje wydarzenie w bazie danych i przekierowuje do widoku **Index**.
* **Edit (GET)**  
  **Parametry**:
  + id (typ int) – Identyfikator wydarzenia.  
    **Opis**:  
    Wyświetla formularz edycji dla konkretnego wydarzenia przypisanego do zalogowanego użytkownika. Jeśli wydarzenie nie istnieje lub użytkownik nie ma do niego dostępu, zwraca błąd **404 Not Found** lub **403 Unauthorized**.
* **Edit (POST)**  
  **Parametry**:
  + id (typ int) – Identyfikator wydarzenia.
  + editedEvent (typ Event) – Zmodyfikowany model wydarzenia.  
    **Opis**:  
    Zapisuje zmiany w istniejącym wydarzeniu przypisanym do zalogowanego użytkownika. Jeśli wydarzenie nie istnieje lub użytkownik nie ma do niego dostępu, zwraca błąd **404 Not Found** lub **403 Unauthorized**. W przypadku poprawnych danych przekierowuje do widoku **Index**.
* **Delete (GET)**  
  **Parametry**:
  + id (typ int) – Identyfikator wydarzenia.  
    **Opis**:  
    Wyświetla potwierdzenie usunięcia dla wydarzenia przypisanego do zalogowanego użytkownika. Jeśli wydarzenie nie istnieje lub użytkownik nie ma do niego dostępu, zwraca błąd **404 Not Found**.
* **DeleteConfirmed (POST)**  
  **Parametry**:
  + id (typ int) – Identyfikator wydarzenia.  
    **Opis**:  
    Usuwa wydarzenie przypisane do zalogowanego użytkownika. Jeśli wydarzenie istnieje i należy do użytkownika, usuwa je z bazy danych i przekierowuje do widoku **Index**.
* **ExportToCsv (GET)**  
  **Parametry**: Brak.  
  **Opis**:  
  Eksportuje listę wszystkich wydarzeń (niezależnie od użytkownika) do pliku **CSV**. Plik zawiera kolumny: Id, Name, Date oraz AvailableSeats. Plik jest generowany dynamicznie z aktualnymi danymi i oferowany do pobrania z unikalną nazwą pliku (zawierającą znacznik czasu).

**HomeController**

Ten kontroler obsługuje główną stronę aplikacji, umożliwia przeglądanie listy wydarzeń, filtrowanie, wyświetlanie szczegółów wydarzeń oraz obsługę błędów.

* **Privacy (GET)**  
  **Parametry**: Brak.  
  **Opis**:  
  Wyświetla statyczną stronę informacyjną dotyczącą prywatności.
* **Index (GET)**  
  **Parametry**:
  + searchTerm (typ string, opcjonalny) – Fraza do filtrowania wydarzeń na podstawie tytułu lub opisu.  
    **Opis**:  
    Wyświetla listę wszystkich wydarzeń z opcjonalną funkcją filtrowania. Jeśli parametr **searchTerm** jest podany, zwracane są tylko wydarzenia, których tytuł lub opis zawiera podaną frazę. Wyniki są przekazywane do widoku za pomocą modelu **EventFilterViewModel**, który zawiera listę pasujących wydarzeń oraz użyty termin wyszukiwania.
* **Details (GET)**  
  **Parametry**:
  + id (typ int, wymagany) – Identyfikator konkretnego wydarzenia.  
    **Opis**:  
    Wyświetla szczegóły wybranego wydarzenia na podstawie identyfikatora. Jeśli wydarzenie o podanym identyfikatorze nie istnieje, zwracany jest błąd **404 Not Found**.
* **Error (GET)**  
  **Parametry**: Brak.  
  **Opis**:  
  Wyświetla stronę błędu. Tworzy model **ErrorViewModel**, który zawiera identyfikator bieżącej operacji (**RequestId**) dla celów diagnostycznych. Ustawia również odpowiedź HTTP na brakujące dane buforowania.

**System użytkowników**

System obsługuje dwie role użytkowników, które de facto dzielą użytkowników na zalogowanych i gości:

* **Użytkownik zalogowany**:
  + Może dodawać nowe wydarzenia.
  + Może eksportować wydarzenia do pliku CSV.
* **Gość**:
  + Może przeglądać listę wydarzeń.
  + Może wyszukiwać wydarzenia po nazwie.
  + Może zobaczyć szczegóły wydarzeń.

**Najciekawsze funkcjonalności**

* **Wyszukiwanie wydarzeń**: Użytkownicy mogą wyszukiwać wydarzenia po nazwie, co ułatwia znalezienie interesującego wydarzenia.
* **Eksport do CSV**: Użytkownikcy zalogowani mogą pobierać listę dodanych przez siebie wydarzeń w formacie CSV, co jest przydatne do raportów lub dalszego przetwarzania.