Dokumentacja techniczna - Aplikacja ToDo List

Autorzy projektu

- **Autor: Sebastian Marek**

Krótki opis projektu

Aplikacja ToDo List to prosta aplikacja webowa, która ułatwia zarządzanie zadaniami. Umożliwia tworzenie, edytowanie, usuwanie, przeglądanie oraz kategoryzowanie zadań według statusów (w toku, ukończone, zaległe itd.). Aplikacja została stworzona z myślą o szybkim i intuicyjnym zarządzaniu pracą, w oparciu o przeglądarkę internetową.

Specyfikacja technologii

- **Backend: C# .NET 6 (wymagana wersja zgodnie z wymaganiami projektu).**
- **Frontend: Razor Pages, Bootstrap do stylizacji komponentów HTML.**
- **Baza danych: MS SQL Server do zarządzania danymi aplikacji.**
- **Środowisko: Visual Studio 2022 do tworzenia kodu i zarządzania projektem.**
- **ORM: Entity Framework Core do komunikacji z bazą danych.**
- **Kontrola dostępu: ASP.NET Identity dla uwierzytelnienia użytkowników.**

Instrukcje pierwszego uruchomienia

1. **Skonfiguruj bazę danych MS SQL Server lub SQLite.**
2. **Zaktualizuj schemat bazy danych, wykonując polecenie Update-Database w konsoli Package Manager Console, aby utworzyć wszystkie tabele.**
3. **Wciśnij przycisk "Run" w Visual Studio, aby uruchomić aplikację lokalnie.**

Struktura projektu

- **Controllers - Zawiera kontrolery aplikacji odpowiedzialne za przetwarzanie żądań HTTP oraz interakcję z modelami i widokami (DashboardController, HomeController, TasksController).**
- **Models - Przechowuje modele reprezentujące dane, takie jak: TaskItem, TaskItemViewModel, AppUser.**
- **Views - Zawiera widoki Razor Pages, odpowiedzialne za generowanie treści HTML.**
- **Data - Odpowiada za konfigurację dostępu do bazy danych, w tym ApplicationDbContext.**
- **Helper – Zawiera dodatkowe usługi pomocnicze takie jak enkodowanie ID każdego z zadań**

Opis modeli

1. **TaskItem - Główny model reprezentujący zadanie. Pola w modelu to:**
    - **Id (int) - unikalny identyfikator zadania (klucz główny).**
    - **Title (string) - tytuł zadania (maks. 100 znaków, z alternatywną wiadomością, że pole jest wymagane).**
    - **Description (string) - opis zadania (z alternatywną wiadomością, że pole jest wymagane).**
    - **DueDate (DateTime) - data zakończenia (z alternatywną wiadomością, że pole jest wymagane).**
    - **PriorityStatus (enum) - status priorytetu (Low, Medium, High, Urgent; z alternatywną wiadomością, że pole jest wymagane).**
    - **Status (enum) - status zadania (NotStarted, InProgress, Completed, Overdue); domyślnie ustawione jako NotStarted.**
    - **UserId (string) - identyfikator użytkownika jako klucz obcy.**
    - **CreatedDate (DateTime) - data utworzenia zadania.**
    - **CompletionDate (DateTime?) - data ukończenia zadania (tylko dla ukończonych).**
2. **AppUser**
    - **ApplicationUser - rozszerzenie wbudowanej klasy IdentityUser, która dostarcza nam autoryzacje użytkowników.**
    - **Tasks - reprezentuje kolekcje obiektów TaskItem. Tworzy połączenie jeden do wielu pomiędzy użytkownikami, a ich zadaniami.**
3. **DashboardViewModel**
    - **DashboardViewModel – przekazuje informacje do widoku ‘Dashboard’**
    - **UpcomingTasks – reprezentuje listę zadań z statusem Upcoming/Nadchodzące**
    - **InProgressTasks – reprezentuje listę zadań z statusem InProgress/W trakcie**
    - **CompletedTasks – reprezentuje listę gotowych zadań (z statusem Completed)**
    - **OverdueTasks -reprezentuje listę zaległych zadań (z statusem Overdue/Zaległe)**
4. **TaskItemViewModel**
    - **TaskItemViewModel – służy jako objekt do przekazywania danych, aby móc obsługiwać input użytkowników oraz wyświetlać go w widoku.**

- **DueDate – zapisany jako string, aby mógł sobie poradzić z wysyłaniem formularza do tworzenia zadania przez użytkownika w zależności od jego preferencji regionalnej.**
- **InProgressTasks – reprezentuje listę zadań z statusem InProgress/W trakcie**
- **CompletedTasks – reprezentuje listę gotowych zadań (z statusem Completed)**
- **OverdueTasks -reprezentuje listę zaległych zadań (z statusem Overdue/Zaległe)**

Kontrolery i ich metody

- **TasksController:**
  - **Index() - GET, wyświetla listę wszystkich zadań.**
  - **Details(int id) - GET, wyświetla szczegóły konkretnego zadania.**
  - **Create() - GET, wyświetla formularz do utworzenia nowego zadania.**
  - **Create(TaskItemViewModel taskItemViewModel) - POST, dodaje nowe zadanie do bazy danych.**
  - **Edit(int id) - GET, wyświetla formularz edycji zadania.**
  - **Edit(TaskItemViewModel taskItemViewModel) - POST, aktualizuje zadanie.**
  - **Delete(int id) - GET, wyświetla stronę potwierdzenia usunięcia zadania.**
  - **DeleteConfirmed(int id) - POST, usuwa zadanie z bazy danych.**

System użytkowników

- **Role w systemie: Obecnie każdy użytkownik jest traktowany jednakowo, brak jest dodatkowych ról typu administrator.**
- **Uwierzytelnienie: Użytkownicy mogą się rejestrować oraz logować do systemu przy użyciu ASP.NET Identity. Każde zadanie jest przypisane do konkretnego użytkownika.**

Najciekawsze funkcjonalności

- **Kategoryzacja zadań: Zadania są podzielone na bloki, takie jak "W toku", "Ukończone", "Zaległe", co pomaga w zarządzaniu listą zadań. Ponadto zadanie automatycznie przeskakuje do sekcji „Zaległych” jeśli minie jego data ukończenia.**
- **Enkodowanie id każdego zadania dla większego bezpieczeństwa.**