Dokumentacja techniczna projektu

Tytuł projektu:

BookWorm

Krótki opis:

BookWorm to aplikacja wspierająca zarządzanie książkami i użytkownikami, umożliwiająca przeglądanie, dodawanie do koszyka. Można znaleźć różną literaturę za pomocy filtrów, przeczytać informację i „kupić". Posiada także panel administratora dla dodawania, edycji i usuwania książek.

Autor:

Yaroslav Fedorov

Wersja .NET: Aplikacja została zbudowana przy użyciu frameworka .NET Core 8.0, który umożliwia tworzenie nowoczesnych, wydajnych i wieloplatformowych aplikacji webowych.

Baza danych:

- **W projekcie wykorzystano Entity Framework Core jako ORM (Object-Relational Mapping) do zarządzania bazą danych.**
- **Domyślną bazą danych jest Microsoft SQL Server, która zapewnia skalowalność i niezawodność przy przechowywaniu danych.**

Autoryzacja i uwierzytelnianie:

- **Mechanizm autoryzacji oparty na Identity Framework umożliwia zarządzanie użytkownikami, rolami oraz logowaniem (np. role Admin i User).**

Frontend:

- **Widoki zostały stworzone przy użyciu Razor Pages, które pozwalają na dynamiczne generowanie treści HTML w połączeniu z backendem.**

Stylizacja i interfejs użytkownika:

- **Do stylizacji użyto frameworka Bootstrap 5, który zapewnia responsywność oraz estetyczny wygląd aplikacji.**

Instrukcja pierwszego uruchomienia

1) **Jeżeli SQL Server nie jest zainstalowany lub jest z nim błąd – patrz pkt. 2. Jeżeli nie ma problemów – patrz pkt. 5.**
2) **Zainstalować SQL Servera na urządzeniu, za potrzeby skonfigurować go dla pomyślnego korzystania. Pobrać SQL Server: https://www.microsoft.com/en-us/sql-server/sql-server-downloads SQL Server Express – zalecany. Podczas instalacji, wybierz opcję LocalDB.**
3) **Jeżeli jest pobrany i skonfigurowany do działania SQL Server, ale wciąż pokazuję się błąd po typu „A connection was successfully established with the server, but then an error occurred during the login process. (provider: SSL Provider, error: 0 - The certificate chain was issued by an authority that is not trusted.)” – proszę edytować zawartość pliku appsettings.json (parameter “ConnectionString”):**

   "ConnectionStrings": {
   "DefaultConnection":"
   Server=(localdb)\\MSSQLLocalDB;Database=BookWormDb;Trusted_Connection=True;MultipleActiveResultSets=true;TrustServerCertificate=True" }

   Lub

   "ConnectionStrings": {
   "DefaultConnection":"Server=localhost\\SQLEXPRESS;Database=master;Trusted_Connection=True;MultipleActiveResultSets=true;TrustServerCertificate=True"

   ↑ Jeżeli korzystasz z SQL Server Express (zalecane) – „DefaultConnection” musi wyglądać tak jak jest napisane nad tą (czerwoną) wskazówką.

   Zależnie od konfiguracji SQL Server.

   Jeżeli wciąż nie działa – proszę dodać do oryginalnej treści parametr „TrustServerCertificate=True” do „DefaultConnection”.

}

4) **Jeżeli wciąż nie działa, a skorzystałeś z tych rad – proszę uprzejmie zapytać ChatGPT, bo na pewno znajdzie dla ciebie szybki sposób rozwiązania** 😊
5) **Po załatwieniu wszystkich spraw związanych z SQL Server, proszę wpisać polecenie „Update-Database” w Package Manager Console**

Struktura projektu:

Projekt BookWorm jest oparty na technologii C# .NET 8 i korzysta z podejścia MVC (Model-View-Controller). Poniżej znajduje się opis głównych elementów struktury:

1. **Areas/Identity**
   Obsługuje funkcje związane z autoryzacją i zarządzaniem tożsamością użytkowników.
2. **Controllers**
   Przechowuje kontrolery obsługujące logikę aplikacji:
   - **BooksController.cs – obsługuje operacje związane z książkami.**
   - **CartController.cs – zarządza logiką koszyka.**
   - **HomeController.cs – obsługuje stronę główną aplikacji.**
   - **ProfilesController.cs – odpowiedzialny za zarządzanie profilami użytkowników.**
3. **Data**
   - **ApplicationDbContext.cs – definiuje kontekst bazy danych, umożliwiając komunikację między aplikacją a bazą danych.**
4. **Migrations**
   Folder zawiera pliki migracji używane do zarządzania schematem bazy danych.
5. **Models**
   Modele reprezentujące dane w aplikacji:
   - **Book.cs – reprezentuje książki.**
   - **Cart.cs – obsługuje dane koszyka.**
   - **ErrorViewModel.cs – przechowuje informacje o błędach.**
   - **PaymentViewModel.cs – obsługuje dane związane z płatnościami.**
   - **Profile.cs – reprezentuje profil użytkownika.**

6. Services

Folder Services zawiera klasy, które zapewniają różne usługi wykorzystywane w całej aplikacji. Usługi te obejmują klasy pomocnicze, zarządzanie danymi, funkcjonalności związane z e-mailem i inne. Usługi są zazwyczaj wstrzykiwane do kontrolerów, stron lub innych komponentów aplikacji, aby obsługiwały określone zadania lub zapewniały wspólne funkcje.

o NoOpEmailSender

Implementacja interfejsu IEmailSender jako no-op (brak operacji). Klasa ta służy do wyłączenia funkcji wysyłania e-maili w aplikacji, zachowując jednocześnie wymóg wstrzykiwania zależności. Zapobiega wykonywaniu operacji związanych z e-mailem, takich jak wysyłanie e-maili potwierdzających rejestrację lub resetowanie hasła, poprzez dostarczenie implementacji metody SendEmailAsync, która nie wykonuje żadnych działań.

Cel: Usługa ta zapobiega wyjątkom związanym z brakiem zależności IEmailSender, gdy funkcjonalność e-mailowa nie jest potrzebna w aplikacji.

Metody:

SendEmailAsync: Metoda przyjmująca parametry e-maila, tematu i wiadomości HTML, ale nie wykonująca żadnych działań i zwracająca ukończone zadanie. Dzięki temu aplikacja nie próbuje wysyłać e-maili, gdy funkcjonalność ta jest wyłączona.

6. **Views**
   Folder widoków zawierający widoki dla różnych funkcjonalności aplikacji:
   - **Books – widoki związane z książkami.**
   - **Cart – widoki koszyka.**
   - **Home – widoki strony głównej.**
   - **Profiles – widoki profilu użytkownika.**
   - **Shared – współdzielone widoki, np. dla layoutu.**
7. **Pliki konfiguracyjne**
   - **appsettings.json – konfiguracja aplikacji, np. dla połączenia z bazą danych.**
   - **Program.cs – punkt wejścia do aplikacji.**

Modele:

Model: Book

Reprezentuje książkę w systemie.

- **Id** *(int)*
  Unikalny identyfikator książki.
- **Title** *(string)*
  Tytuł książki. Domyślna wartość to pusty ciąg znaków.
- **Author** *(string)*
  Autor książki. Domyślna wartość to pusty ciąg znaków.
- **Price** *(float)*
  Cena książki. Domyślnie ustawiona na 0f.
- **Description** *(string)*
  Opis książki. Domyślnie ustawiona na pusty ciąg znaków.
- **ImageUrl** *(string)*
  URL do obrazu okładki książki. Wyświetlany w widoku z użyciem atrybutu [Display(Name = "Image")]. Domyślnie ustawiony na pusty ciąg znaków.

Model: Cart

Reprezentuje wpis w koszyku użytkownika.

- **Id** *(int)*
  Unikalny identyfikator wpisu w koszyku.
- **UserId** *(string)*
  Klucz obcy wskazujący użytkownika, który dodał książkę do koszyka.
- **BookId** *(int)*
  Klucz obcy wskazujący książkę dodaną do koszyka.
- **BookName** *(string)*
  Nazwa książki w momencie dodania do koszyka.
- **Price** *(decimal)*
  Cena książki w momencie dodania do koszyka.
- **User** *(IdentityUser) (właściwość nawigacyjna)*
  Powiązanie z użytkownikiem (model IdentityUser).
- **Book** *(Book) (właściwość nawigacyjna)*
  Powiązanie z modelem Book.

Model: ErrorViewModel

Służy do obsługi błędów w aplikacji.

- **RequestId** *(string?)*
  Identyfikator żądania, który pomaga w identyfikacji błędu.
- **ShowRequestId** *(bool)*
  Wskazuje, czy należy wyświetlić RequestId. Zwraca true, jeśli RequestId nie jest pusty.

Model: PaymentViewModel

Obsługuje dane niezbędne do realizacji płatności.

- **CartItems** *(IEnumerable<Cart>)*
  Lista pozycji znajdujących się w koszyku użytkownika.
- **TotalAmount** *(decimal)*
  Całkowita kwota do zapłaty za wszystkie pozycje w koszyku.

Model: Profile

Reprezentuje profil użytkownika.

- **Id** *(int)*
  Unikalny identyfikator profilu.
- **UserId** *(string)*
  Klucz obcy do użytkownika (IdentityUser).
- **FullName** *(string?)*
  Pełne imię użytkownika. Wartość opcjonalna.
- **Surname** *(string?)*
  Nazwisko użytkownika. Wartość opcjonalna.
- **PhoneNumber** *(string?)*
  Numer telefonu użytkownika. Opcjonalne pole.
- **PostAddress** *(string?)*
  Adres pocztowy użytkownika. Opcjonalne pole.
- **ProfilePictureUrl** *(string?)*
  URL do zdjęcia profilowego użytkownika.

- **User** *(IdentityUser) (właściwość nawigacyjna)*
  Powiązanie z użytkownikiem (model IdentityUser).

Kontrolery:

BooksController (Kontroler Książek)

1. Index

- **Metody HTTP: GET**
- **Parametry:**
  - **sortOrder (string, opcjonalny): Określa porządek sortowania książek (np. według tytułu, autora lub ceny).**
  - **searchString (string, opcjonalny): Fraza wyszukiwania do filtrowania książek według tytułu.**
- **Opis działania:**
  Pobiera listę książek z bazy danych, stosuje opcjonalne sortowanie i filtrowanie, a następnie przekazuje wynik do widoku.
- **Zwracane dane: Widok wyświetlający przefiltrowaną i posortowaną listę książek.**

2. Details

- **Metody HTTP: GET**
- **Parametry:**
  - **id (int, wymagany): Identyfikator książki, dla której mają być wyświetlone szczegóły.**
- **Opis działania:**
  Pobiera szczegóły książki na podstawie jej identyfikatora. Jeśli książka nie istnieje, zwraca błąd 404.
- **Zwracane dane: Widok wyświetlający szczegółowe informacje o książce.**

3. Create

- **Metody HTTP:**
  - **GET: Wyświetla formularz do utworzenia nowej książki.**
  - **POST: Przetwarza przesłany formularz w celu dodania nowej książki.**
- **Parametry (POST):**
  - **book (Book, wymagany): Obiekt książki do dodania.**
- **Opis działania:**
  - **GET: Renderuje formularz tworzenia książki.**
  - **POST: Waliduje dane i zapisuje książkę w bazie danych.**

- **Zwracane dane:**
    - **GET: Widok formularza tworzenia książki.**
    - **POST: Przekierowanie do widoku AdminIndex w przypadku sukcesu lub ponowne wyświetlenie formularza z błędami walidacji.**

4. Edit

- **Metody HTTP:**
    - **GET: Wyświetla formularz do edycji istniejącej książki.**
    - **POST: Przetwarza przesłany formularz w celu aktualizacji książki.**
- **Parametry:**
    - **GET:**
        - **id (int, wymagany): Identyfikator książki do edycji.**
    - **POST:**
        - **id (int, wymagany): Identyfikator książki do aktualizacji.**
        - **book (Book, wymagany): Zaktualizowany obiekt książki.**
- **Opis działania:**
    - **GET: Pobiera dane książki do edycji.**
    - **POST: Aktualizuje książkę w bazie danych po walidacji.**
- **Zwracane dane:**
    - **GET: Widok formularza edycji.**
    - **POST: Przekierowanie do widoku AdminIndex w przypadku sukcesu lub ponowne wyświetlenie formularza z błędami walidacji.**

5. Delete

- **Metody HTTP:**
    - **GET: Wyświetla widok potwierdzenia usunięcia książki.**
    - **POST: Usuwa książkę po potwierdzeniu.**
- **Parametry:**
    - **GET:**
        - **id (int, wymagany): Identyfikator książki do usunięcia.**
    - **POST:**

        - **id (int, wymagany): Identyfikator książki do usunięcia (z potwierdzenia).**

- **Opis działania:**
    - **GET: Wyświetla widok potwierdzający usunięcie książki.**
    - **POST: Usuwa książkę z bazy danych.**
- **Zwracane dane:**
    - **GET: Widok potwierdzenia usunięcia.**
    - **POST: Przekierowanie do widoku AdminIndex.**

6. AdminIndex

- **Metody HTTP: GET**
- **Parametry:**
    - **sortOrder (string, opcjonalny): Określa porządek sortowania książek.**
    - **searchString (string, opcjonalny): Fraza wyszukiwania do filtrowania książek według tytułu.**
- **Opis działania:**
  Podobna do Index, ale dostępna wyłącznie dla użytkowników z uprawnieniami administratora. Wyświetla listę książek z opcjami sortowania i filtrowania.
- **Zwracane dane: Widok wyświetlający przefiltrowaną i posortowaną listę książek.**

7. AddToCart

- **Metody HTTP: POST**
- **Parametry:**
    - **bookId (int, wymagany): Identyfikator książki do dodania do koszyka.**
- **Opis działania:**
  Dodaje określoną książkę do koszyka użytkownika.
- **Zwracane dane: Przekierowanie do akcji ViewCart.**

8. ViewCart

- **Metody HTTP: GET**
- **Parametry: Brak**
- **Opis działania:**
  Pobiera i wyświetla elementy znajdujące się w koszyku użytkownika.

- **Zwracane dane: Widok wyświetlający listę przedmiotów w koszyku.**

9. RemoveFromCart

- **Metody HTTP: POST**
- **Parametry:**
    - **id (int, wymagany): Identyfikator elementu koszyka do usunięcia.**
- **Opis działania:**
  Usuwa określony element z koszyka użytkownika.
- **Zwracane dane: Przekierowanie do akcji ViewCart.**

CartController (Kontroler Koszyka)

1. AddToCart

- **Metody HTTP: POST**
- **Parametry:**
    - **bookId (int, wymagany): Identyfikator książki do dodania do koszyka.**
- **Opis działania:**
  Dodaje książkę do koszyka.
- **Zwracane dane: Przekierowanie do akcji ViewCart.**

2. ViewCart

- **Metody HTTP: GET**
- **Parametry: Brak**
- **Opis działania:**
  Wyświetla koszyk użytkownika.
- **Zwracane dane: Widok wyświetlający elementy koszyka.**

3. RemoveFromCart

- **Metody HTTP: POST**
- **Parametry:**
    - **id (int, wymagany): Identyfikator elementu koszyka do usunięcia.**
- **Opis działania:**
  Usuwa element z koszyka.
- **Zwracane dane: Przekierowanie do akcji ViewCart.**

4. Payment

- **Metody HTTP: GET**
- **Parametry: Brak**
- **Opis działania:**
  Wyświetla stronę płatności z podsumowaniem koszyka użytkownika i całkowitą kwotą.
- **Zwracane dane: Widok strony płatności.**

5. ConfirmPayment

- **Metody HTTP: POST**
- **Parametry: Brak**
- **Opis działania:**
  Przetwarza symulację płatności i opróżnia koszyk użytkownika po udanej płatności.
- **Zwracane dane: Przekierowanie do akcji ViewCart z komunikatem o sukcesie.**

HomeController (Kontroler Strony Głównej)

1. Index

- **Metody HTTP: GET**
- **Parametry:**
  - **searchString (string, opcjonalny): Fraza wyszukiwania do filtrowania książek.**
- **Opis działania:**
  Wyświetla stronę główną z listą książek lub przekierowuje do akcji BooksController.Index w przypadku podania frazy wyszukiwania.
- **Zwracane dane: Widok wyświetlający książki lub przekierowanie do BooksController.Index.**

2. Privacy

- **Metody HTTP: GET**
- **Parametry: Brak**
- **Opis działania:**
  Wyświetla stronę z polityką prywatności.
- **Zwracane dane: Widok.**

3. Error

- **Metody HTTP: GET**
- **Parametry: Brak**
- **Opis działania:**
  Wyświetla stronę błędu obsługującą wyjątki.
- **Zwracane dane: Widok z informacjami o błędzie.**

ProfilesController (Kontroler Profili)

1. Index

- **Metody HTTP: GET**
- **Parametry: Brak**
- **Opis działania:**
  Pobiera listę wszystkich profili użytkowników i wyświetla je.
- **Zwracane dane: Widok wyświetlający listę profili.**

2. Details

- **Metody HTTP: GET**
- **Parametry:**
  - **id (int, wymagany): Identyfikator profilu do wyświetlenia.**
- **Opis działania:**
  Wyświetla szczegóły wybranego profilu użytkownika.
- **Zwracane dane: Widok pokazujący szczegóły profilu.**

3. Create

- **Metody HTTP:**
  - **GET: Wyświetla formularz tworzenia nowego profilu.**
  - **POST: Przetwarza formularz w celu utworzenia nowego profilu.**
- **Parametry (POST):**
  - **profile (Profile, wymagany): Obiekt profilu do utworzenia.**
- **Opis działania:**
  - **GET: Renderuje formularz tworzenia profilu.**
  - **POST: Zapisuje profil w bazie danych.**

- **Zwracane dane:**
    - **GET: Widok formularza.**
    - **POST: Przekierowanie do widoku Index.**

4. Edit

- **Metody HTTP:**
    - **GET: Wyświetla formularz edycji istniejącego profilu.**
    - **POST: Przetwarza formularz w celu zaktualizowania profilu.**
- **Parametry:**
    - **GET:**
        - **id (int, wymagany): Identyfikator profilu do edycji.**
    - **POST:**
        - **profile (Profile, wymagany): Zaktualizowany obiekt profilu.**
- **Opis działania:**
    - **GET: Pobiera profil do edycji.**
    - **POST: Aktualizuje profil w bazie danych.**
- **Zwracane dane:**
    - **GET: Widok formularza.**
    - **POST: Przekierowanie do widoku Index.**

5. Delete

- **Metody HTTP:**
    - **GET: Wyświetla potwierdzenie usunięcia profilu.**
    - **POST: Usuwa profil po potwierdzeniu.**
- **Parametry:**
    - **GET:**
        - **id (int, wymagany): Identyfikator profilu do usunięcia.**
    - **POST:**
        - **id (int, wymagany): Identyfikator profilu do usunięcia (z potwierdzenia).**
- **Opis działania:**

    - **GET: Wyświetla widok potwierdzenia.**
    - **POST: Usuwa profil z bazy danych.**
- **Zwracane dane:**
    - **GET: Widok potwierdzenia.**
    - **POST: Przekierowanie do widoku Index.**

Opis systemu użytkowników

Role w systemie

System obsługuje różne role użytkowników, które określają ich uprawnienia i dostęp do funkcji systemu. Role w systemie to:

1. **Gość**
    - Niezalogowany użytkownik.
    - Ma ograniczony dostęp do funkcjonalności (np. przeglądanie książek, ale brak możliwości edycji danych lub dostępu do koszyka).
2. **Użytkownik zalogowany**
    - Użytkownik posiadający konto w systemie i zalogowany.
    - Dodatkowe możliwości:
        - Dostęp do koszyka i funkcji zakupowych.
        - Możliwość dodawania książek do koszyka i przechodzenia przez proces płatności.
3. **Administrator**
    - Użytkownik posiadający rozszerzone uprawnienia administracyjne.
    - Możliwości:
        - Zarządzanie książkami (tworzenie, edytowanie, usuwanie).
        - Dostęp do widoku AdminIndex, który pozwala na zarządzanie danymi.

Nadawanie roli użytkownikowi

Role są przypisane do użytkowników w momencie tworzenia konta.

- Nadanie roli może odbywać się przez:
    - **Bezpośrednią zmianę w bazie danych**: rola może być przypisana na poziomie bazy danych w tabeli użytkowników.

Możliwości użytkowników

1. **Goście**
    - Mogą:
        - Przeglądać listę książek w systemie.

        - Wyszukiwać książki za pomocą fraz wyszukiwania.
    - Nie mogą:
        - Dodawać książek do koszyka.
        - Przechodzić do procesu płatności.
        - Zarządzać danymi książek lub profili użytkowników.
2. **Zalogowani użytkownicy**
    - Mogą:
        - Wszystko, co goście.
        - Dodawać książki do koszyka.
        - Przeglądać i edytować swój koszyk.
        - Przechodzić do strony płatności i potwierdzać zakup.
    - Nie mogą:
        - Zarządzać książkami (dodawać, edytować, usuwać).
        - Zarządzać profilami innych użytkowników.
3. **Administratorzy**
    - Mają pełny dostęp do systemu.
    - Mogą:
        - Wszystko, co zalogowani użytkownicy.
        - Tworzyć, edytować i usuwać książki w systemie.

Informacje powiązane z użytkownikami

1. **Informacje powiązane z konkretnym użytkownikiem**
    - **Profil użytkownika**:
        - Identyfikator użytkownika (ID).
        - Dane osobowe i kontaktowe (np. imię, nazwisko, e-mail).
        - Rola użytkownika (np. gość, użytkownik, administrator).
    - **Koszyk użytkownika**:
        - Przechowuje listę książek dodanych do koszyka przez użytkownika.

        - Koszyk jest specyficzny dla zalogowanego użytkownika i przypisany do jego sesji.

2. **Informacje globalne**
    - Lista książek dostępnych w systemie:
        - Dane o książkach (np. tytuł, autor, cena) są wspólne dla wszystkich użytkowników, niezależnie od ich roli.
    - Polityka prywatności i widok strony głównej:
        - Są one ogólnodostępne i nie zależą od użytkownika.
    - Dane administracyjne:
        - Funkcjonalności, takie jak zarządzanie książkami, są widoczne tylko dla administratorów, ale dane same w sobie (np. książki) są częścią globalnego systemu.

Podsumowanie różnic

Funkcja	**Gość**	**Zalogowany użytkownik**	**Administrator**
Przeglądanie książek	Tak	Tak	Tak
Wyszukiwanie książek	Tak	Tak	Tak
Dodawanie do koszyka	Nie	Tak	Tak
Przeglądanie koszyka	Nie	Tak	Tak
Płatność	Nie	Tak	Tak
Tworzenie książek	Nie	Nie	Tak
Edytowanie książek	Nie	Nie	Tak
Usuwanie książek	Nie	Nie	Tak

Ciekawe funkcjonalności

Filtrowanie książek:
Aplikacja umożliwia użytkownikom przeglądanie dostępnych książek z wykorzystaniem zaawansowanego systemu filtrowania. Użytkownik może filtrować książki według nazwy, autora lub ceny. Dzięki temu proces wyszukiwania staje się bardziej intuicyjny i efektywny.

Obsługiwanie koszyka:
Funkcjonalność koszyka pozwala użytkownikom dodawać wybrane książki do wirtualnego koszyka i usuwać przedmioty. Koszyk przechowuje dane nawet po odświeżeniu strony i jest osobny dla każdego użytkownika. System pozwala także na podgląd całkowitego kosztu zamówienia i przykładowe przejście do realizacji zakupu.

Panel administratora:
Panel administratora umożliwia zarządzanie książkami. Administrator może dodawać, edytować lub usuwać książki. Panel jest zabezpieczony przed nieautoryzowanym dostępem i otwarty wyłącznie dla użytkowników z rolą Admin.

Wyszukiwarka książek:
Wyszukiwarka zapewnia szybkie i precyzyjne znajdowanie książek na podstawie wprowadzonej frazy. Mechanizm obsługuje wyszukiwanie po tytule. Wyniki są prezentowane dynamicznie, z możliwością dalszego filtrowania lub sortowania według preferencji użytkownika.