**Biblioteka**

|  |  |  |  |
| --- | --- | --- | --- |
| **Wersja** | **Data utworzenia** | **Data ost. modyfikacji** | **Autorzy** |
| 1.0 | 15.01.2025 | 01.02.2025 | Kamil Wieczorek  Nr. Albumu.: 127 901 |

# Ogólne informacje – Opis systemu

Głównym celem było utworzenie aplikacji webowej, która sprawi, że pracownicy biblioteki będą mogli nią cyfrowo zarządzać, natomiast sami czytelnicy – użytkownicy, będą mogli korzystać z bibliotek w przystępniejszy sposób. Konsekwencją stworzenia takiego systemu jest przedstawienie roli administratora systemu. W zależności od roli przypisanej do użytkownika, dostępne są inne funkcjonalności. Po krótce przedstawiając funkcjonalność systemu można wymienić możliwość logowania, rejestracji, zarządzania książkami, filtrowanie książek biblioteki, zarządzanie użytkownikami, zarządzanie rezerwacjami oraz wypożyczeniami.

Dostęp do projektu jest publiczny. Każdy może wejść na podstrony Login oraz Rejestracja.

By móc przeglądać katalog książek, wypożyczać i rezerwować czytelnik powinien się zalogować. Podobnie w scenariuszu kont z innymi rolami, czyli administratora oraz pracownika.

# Specyfikacja

* Język programowania: C#
* Środowisko uruchomieniowe: .NET 8
* Framework: ASP .NET MVC
* ORM: Entity Framework Core
* Baza: Sql Server

# Instrukcje pierwszego uruchomienia projektu

* Otworzenie projektu w Visual Studio
* Wybranie z górnego menu Tools -> NuGetPackage Manager -> Package Manager Console.
  + Albo w inny sposób w oknie Package Manager Console (View -> Other Windows - Package Manager Console)
* Wpisać komendę „Update-Database” – na podstawie migracji w folderze Migrations, utworzy się w bazie struktura bazy danych (tabele, klucze, relacje).
* Uruchomienie aplikacji
* Aby utworzyć konto z rolą Admin przy **rejestracji** należy podać hasło „Admin123@”.

Mała uwaga 1:

Jeżeli pojawi się problem z Updatem bazy, spowodowany właśnie bazą, to należy spróbować kilku potencjalnych rozwiązań: zmienić nazwę bazy w connection string albo usunąć bazę i migracje z folderu migrations potem otworzyć projekt i w (wspomnianym wyżej) package manager console wpisać „**add-migration init**”. Dopiero po tym spróbować jeszcze raz polecenia **Update-Database**

Mała uwaga 2:

Ponieważ GitHub nie mógł zaakceptować wersji większej niż 25MB, to zostałem, niestety, zmuszony do usunięcia zdjęć okładek książek z katalogu biblioteki. Jednakże ów zdjęcia można bez problemu dodać z poziomu konta pracownika. Do repozytorium załączyłem dedykowany do tego folder. Podobnie z zrzutami sprzed usunięcia wspomnianych wyżej zdjęć

#### Dalszy przebieg działania

* Po utworzeniu konta Administratora należy utworzyć konto pracownika.
  + Myszką, zmierzamy do prawego górnego rogu najeżdżamy na „Witaj <nazwa\_admina>” i klikamy -> „Mój panel”
  + Lewy blok „Użytkownicy” tam znajduje się opcja dodania pracownika biblioteki.

Reszta kont, niezależnie od roli ma możliwość korzystania z podstron dla siebie przeznaczonych w podobny sposób jak powyższy przykład

# Opis struktury projektu

* W katalogu Controllers znajdują się kontrolery aplikacji. To tu trafia żądanie od użytkownika. Zostaje ono obsłużone wraz ze sprawdzeniem roli a następnie wynik przekazywany jest do widoku.
* W katalogu DTO znajdują się klasy, które służą do przekazywania danych pobranych w wyniku żądania. Dane pobrane z bazy modyfikowane lub przypisywane są do klas DTO, tylko te pola które są potrzebne.
* W folderze Files zapisywane są obrazy, które wybierane są z pamięci komputera podczas dodawania i edycji książki.
* W folderze Migrations zapisywane są migracje, które są tworzone automatycznie przez Entity Framework Core. Można je modyfikować lub tworzyć własnoręcznie aby modyfikować strukturę bazy danych.
* Klasy zawarte w folderze Models odzwierciedlają tabele w bazie danych.
* W folderze Repositories są pliki które dla każdego modelu, a tym samym tabeli w bazie danych obsługują operacje w bazie danych takie jak dodawanie, edycja, usuwanie i edycja.
* Folder Services zawiera serwisy, które są przekaźnikiem między kontrolerem, a repozytorium. Dane otrzymane w kontrolerze podczas żądania przez użytkownika aplikacji przekazywane są do serwisu. W tym miejscu zachodzą operacje, które modyfikują dane i w gotowej do zapisania formie przekazują do repozytorium, gdzie odbywa się zapis w bazie.
* W katalogu Views, znajdują się widoki, czyli warstwa prezentacji. Pliki te są odpowiedzialne za wyświetlanie danych otrzymanych z kontrolera oraz inne wizualne elementy.
* W pliku appsettings.json przechowywana jest konfiguracja aplikacji. W przypadku tej aplikacji znajduje się tylko ConnectionString do bazy danych.

# Opis systemu użytkowników ze wszystkimi jego komponentami

System obsługuje trzy role zalogowanych użytkowników oraz gościa. Użytkownik niezalogowany ma dostęp jedynie do widoku rejestracji konta oraz logowania.

* Zalogowany czytelnik z rolą User może przeglądać książki i je wypożyczać oraz przeglądać swoją historię wypożyczeni i rezerwacji.
* Uprawnienia pracownika z rolą Employee, którego kontro może zostać utworzone jedynie przez administratora systemu, pozwalają mu na zarządzanie książkami.
* Administrator natomiast ma rolę Admin i może zarządzać wszystkimi kontami w systemie. Aby utworzyć konto admina należy podczas rejestracji ustawić hasło „Admin123@”.

# Modele

* User – tabela użytkownika
* Book – tabela książki
* BookingData – tabela łącząca książkę z użytkownikiem (relacja wiele do wielu)
* Rental – tabela wypożyczenia
* Reservation – tabela rezerwacji

Warto też wspomnieć, że w bazie danych utworzy się więcej tabel niż tylko powyżej wylistowane. Dzieje się tak ponieważ biblioteka Identity, która została zainstalowana w projekcie sama tworzy dodatkowe tabele, które pomagają obsłużyć autoryzację użytkownika. Jedną z ważniejszych tabel jest tabela Role odpowiedzialna za przechowywanie roli użytkownika w systemie.

# Wylistowane kontrolery z metodami

AccountController – kontroler odpowiedzialny za autoryzacje użytkownika

* Register
* Login
* GetUserHistory
* Logout

BookController – kontroler odpowiedzialny za obsługę książki

* ViewBook
* DeleteBook

Dashboard – kontroler odpowiedzialny za panel użytkownika, admina, pracownika

* Index
* CreateBook
* GetUsers
* DeleteUser
* CreateEmployee
* UpdateUser
* GetUserHistory
* GetReservations
* GetRentals
* UpdateBook

HomeController – kontroler obsługujący widok strony głównej oraz metod obsługi plików

* Index
* UploadFile
* DisplayImage

RentalController – kontroler zmieniający status wypożyczeni

* ChangeStatus

ReservationController – kontroler obsługujący zdarzenia związane z rezerwacją książki

* ChangeStatus

# Najciekawsze funkcjonalności

* Filtrowanie książek po tytule, autorze i gatunku.
  + Filtrowanie odbywa się po stronie frontu przy pomocy skryptów javascript. Wyszukiwana fraza sprawdzana jest w elementach wyświetlonych na stronie. Jeśli zawiera się ona, np. w tytule to nic się nie zmienia, a jeśli wyszukiwana fraza nie pasuje do żadnego elementu jest ona chowana poprzez ustawienie na danym elemencie html stylu display = ‘none’.
* Dodanie zdjęcia do książki
  + Wybór okładki książki odbywa się poprzez kliknięci pola z linkiem do książki. Po tej czynności otwiera się okno z plikami na komputerze, aby wskazać żądany plik. Następnie plik ten zapisuje się na serwerze, a tak naprawdę w folderze Files. Do nazwy pliku doklejany jest na początku unikalny tekst typu Guid oraz nazwa folderu Files. Wartość ta jest ścieżką pod, którą znajduje się dany plik. Zachowywana jest ona w tabeli książki. Dzięki temu po pobraniu książek z bazy, aplikacja może od razu pobrać ścieżkę do pliku i wyświetlić go.
* Walidacja możliwości rezerwacji/wypożyczenia książki
  + Aby zapobiec możliwości wypożyczeniu książki, która została w danym okresie wypożyczona, napisano mechanizm walidacji. Pobiera on wszystkie rezerwacje danej książki oraz jej wypożyczenia. Przy czym, wypożyczenia filtrowane są po statusie. Brane pod uwagę są tylko te wypożyczenia, których status jest Zakończona lub Anulowana.
  + Następnie sprawdzane jest czy daty, które podał czytelnik nie pokrywają się lub nie zachodzą na siebie z tymi które znajdują się w połączonej liście wybranych wypożyczeni oraz wszystkich rezerwacji.