



Wyższa Szkoła Informatyki i Zarządzania
Kolegium Informatyki Stosowanej, Informatyka

Rafał Wątroba, w65575

Remigiusz Wojak, w65577

Projekt Szkolenie Techniczne 2

Rzeszów, 28.06.2023

1. Opis wybranego tematu, funkcjonalności systemu.

Aplikacja webowa umożliwiająca tworzenie własnych półek w celu dodania książek. Również umożliwia przeglądanie listy książek oraz listy autorów oraz przeglądanie informacji o nich.

2. Opis wybranego stosu technologicznego

- **C#:** Jest to język programowania ogólnego przeznaczenia, rozwijany przez firmę Microsoft. C# jest często wykorzystywany do tworzenia aplikacji dla platformy .NET.
- **.NET Core:** To wieloplatformowy framework open-source, również opracowany przez Microsoft. .NET Core umożliwia tworzenie aplikacji na różne systemy operacyjne, takie jak Windows, macOS i Linux.
- **Angular:** Jest to framework JavaScript stworzony przez firmę Google. Angular umożliwia tworzenie aplikacji internetowych i mobilnych. Jest często wykorzystywany w połączeniu z HTML i CSS.
- **HTML:** Jest to język znaczników wykorzystywany do tworzenia struktur i treści stron internetowych. HTML definiuje elementy strony, takie jak nagłówki, akapity, obrazy itp.
- **CSS:** Jest to język arkuszy stylów, który służy do określania wyglądu i formatowania stron internetowych. CSS definiuje style, takie jak kolory, czcionki, marginesy itp.
- **Bootstrap:** Jest to framework CSS, który ułatwia tworzenie responsywnych i estetycznych stron internetowych. Bootstrap oferuje gotowe komponenty i stylizację, które można łatwo włączyć do projektu.
- **MariaDB:** Jest to system zarządzania bazą danych oparty na MySQL. MariaDB jest rozwinięciem MySQL i oferuje wiele funkcji i ulepszeń w porównaniu z oryginalnym systemem.

Oraz krótki opis większości użytych bibliotek:

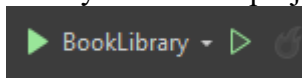
- **FluentValidation:** biblioteka do walidacji danych w aplikacjach .NET. Zapewnia bardziej ekspresywny i deklaratywny sposób definiowania reguł walidacji. Może być używana wraz z różnymi frameworkami, takimi jak ASP.NET Core, MVC itp.
- **MediatR:** biblioteka do implementacji wzorca mediatora w aplikacjach .NET. Wzorzec mediatora pomaga w odseparowaniu logiki biznesowej od komunikacji między komponentami. MediatR ułatwia obsługę żądań, zapytań i zdarzeń, umożliwiając łatwą komunikację między różnymi częściami aplikacji.
- **IdentityServer:** biblioteka do uwierzytelniania i autoryzacji w aplikacjach .NET. IdentityServer to narzędzie open-source, które umożliwia implementację protokołów uwierzytelniania, takich jak OAuth 2.0 i OpenID Connect. Może być wykorzystywane do budowania bezpiecznych systemów uwierzytelniania jednoznakowego (SSO) i zarządzania tożsamością.
- **Diagnostics.EntityFrameworkCore:** zawiera zestaw narzędzi i funkcji diagnostycznych dla Entity Framework Core. Umożliwia monitorowanie wydajności, debugowanie i śledzenie zapytań SQL, rejestrowanie zdarzeń i wiele innych narzędzi, które mogą pomóc w analizie i optymalizacji działania aplikacji opartych na Entity Framework Core.
- **Identity.EntityFrameworkCore:** biblioteka dostarczająca funkcjonalności zarządzania tożsamością, takie jak uwierzytelnianie, autoryzacja, role i uprawnienia w aplikacjach opartych na Entity Framework Core. Umożliwia przechowywanie i zarządzanie użytkownikami, rolami, hasłami i innymi informacjami związanymi z tożsamością w bazie danych.
- **Pomelo.EntityFrameworkCore.MySql:** Biblioteka rozszerzająca funkcjonalności Entity Framework Core w celu umożliwienia komunikacji z bazą danych w systemie MariaDB.

- **Identity.UI:** biblioteka dostarczająca gotowe widoki i interfejs użytkownika dla funkcjonalności zarządzania tożsamością dostarczanych przez bibliotekę Identity.EntityFrameworkCore. Umożliwia szybkie wdrożenie panelu administracyjnego, w którym można zarządzać użytkownikami, rolami, hasłami itp.
- **SpaProxy:** biblioteka służąca do tworzenia interfejsu API (Application Programming Interface) dla aplikacji jednostronicowych (Single Page Applications - SPA). Umożliwia komunikację między front-endem aplikacji SPA a backendem poprzez automatyczne generowanie kodu proxy, co upraszcza proces integracji między dwoma warstwami.
- **EntityFrameworkCore.Relational:** część Entity Framework Core, która zapewnia narzędzia i funkcje dla baz danych

3. Opis jak uruchomić aplikację

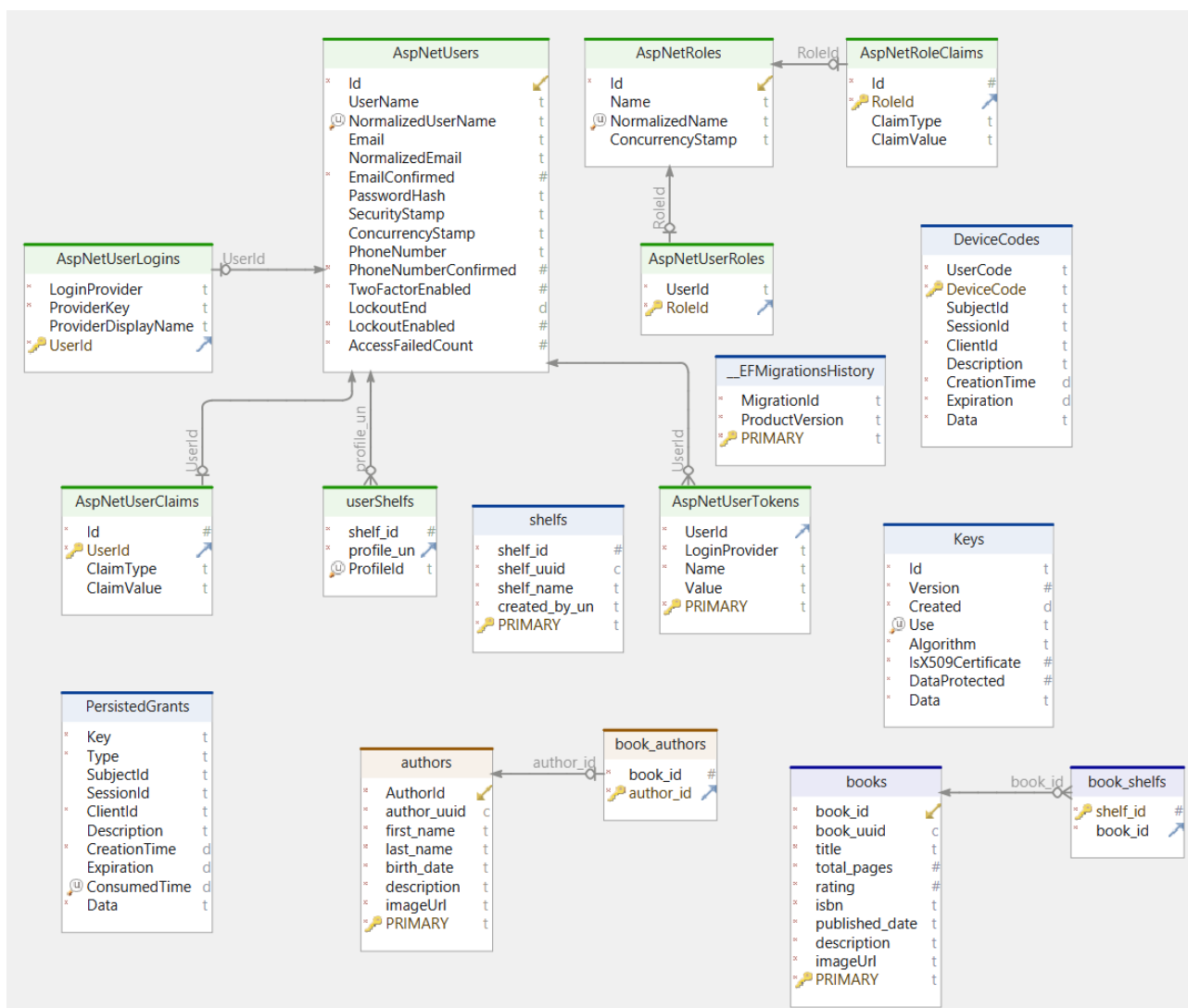
Aby uruchomić projekt należy wykonać następujące kroki:

1. Zainstalować wymagane oprogramowanie:
 - a. Visual Studio
 - b. .Net 7.0
 - c. Node.js
 - d. Angular cli
2. Należy dołączyć do projektu plik appsettings.json w którym znajdują się connection string do bazy danych MariaDb utworzonej na allwaysdata.com.
3. Należy uruchomić projekt klikając kompilację.



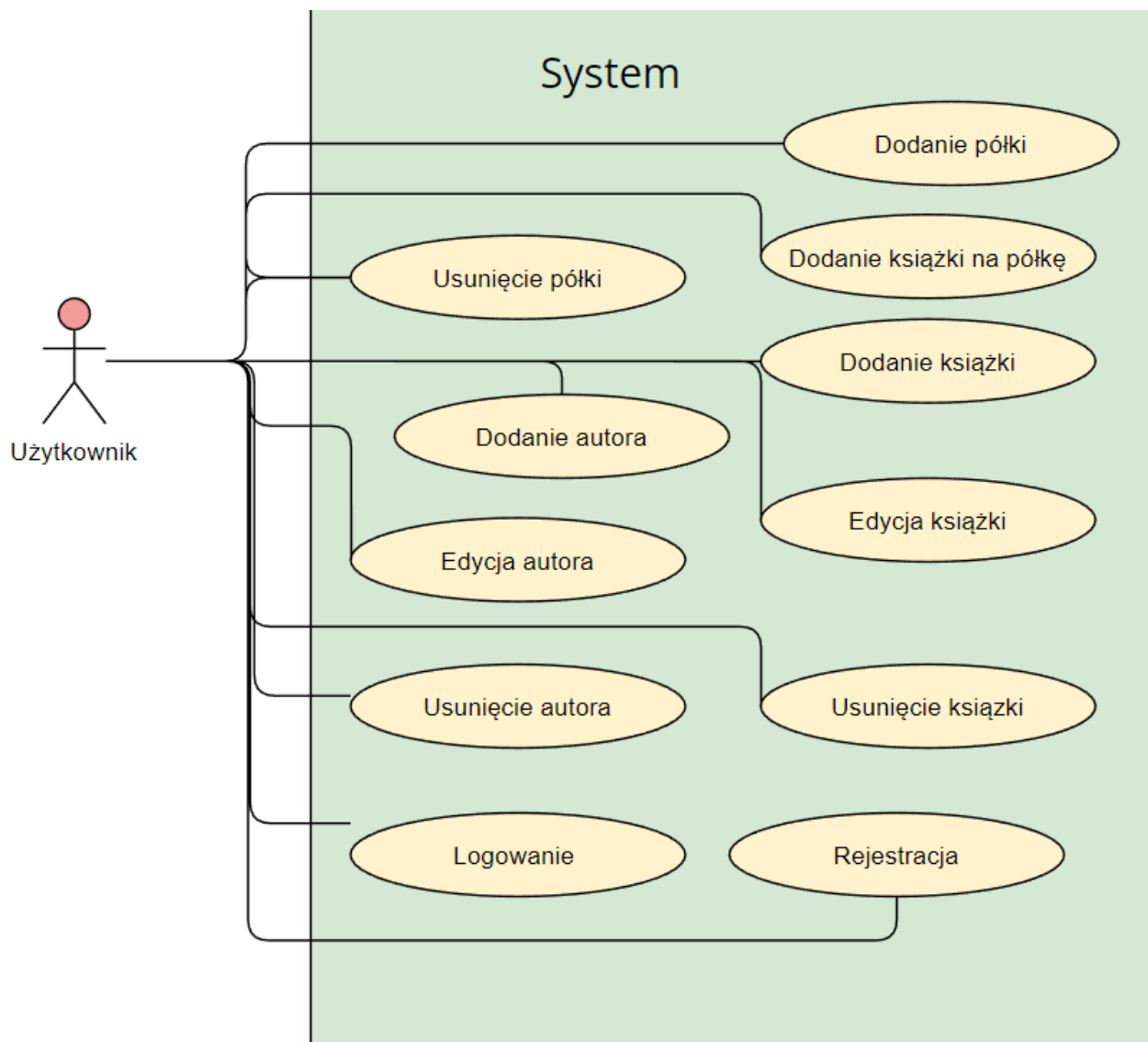
4. Zostanie uruchomiony backend z proxy które będzie oczekiwało na włączenie frontendu który uruchomi się po chwili.
5. Aplikacja jest gotowa do działania.

4. Diagram bazy danych:



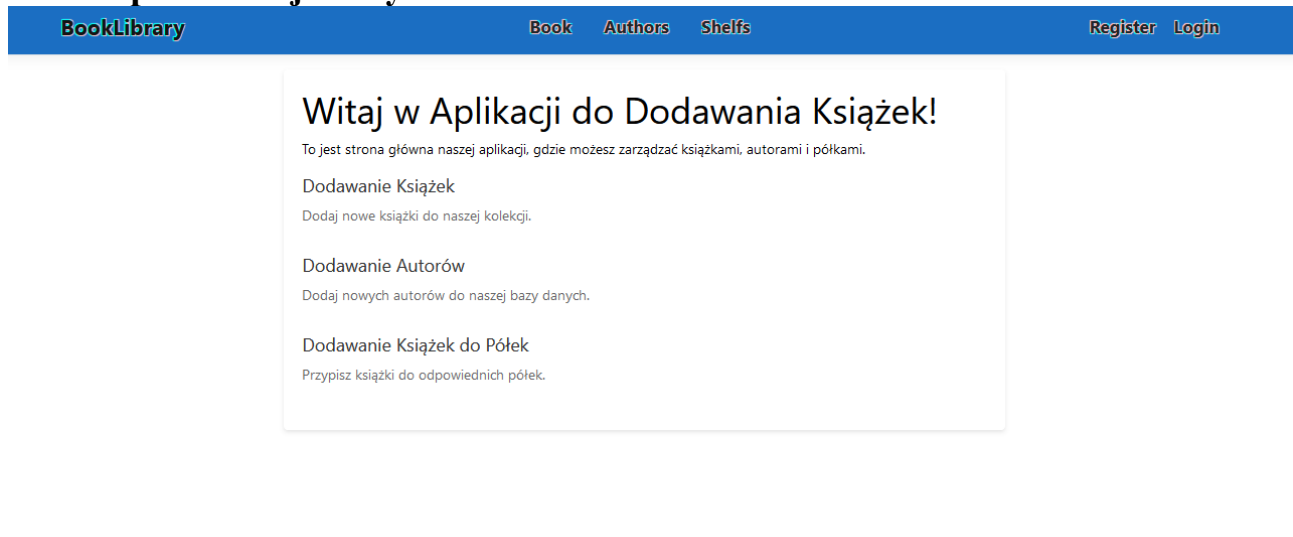
Rys. 1 Diagram Bazy Danych

5. Diagram UML Przypadków użycia

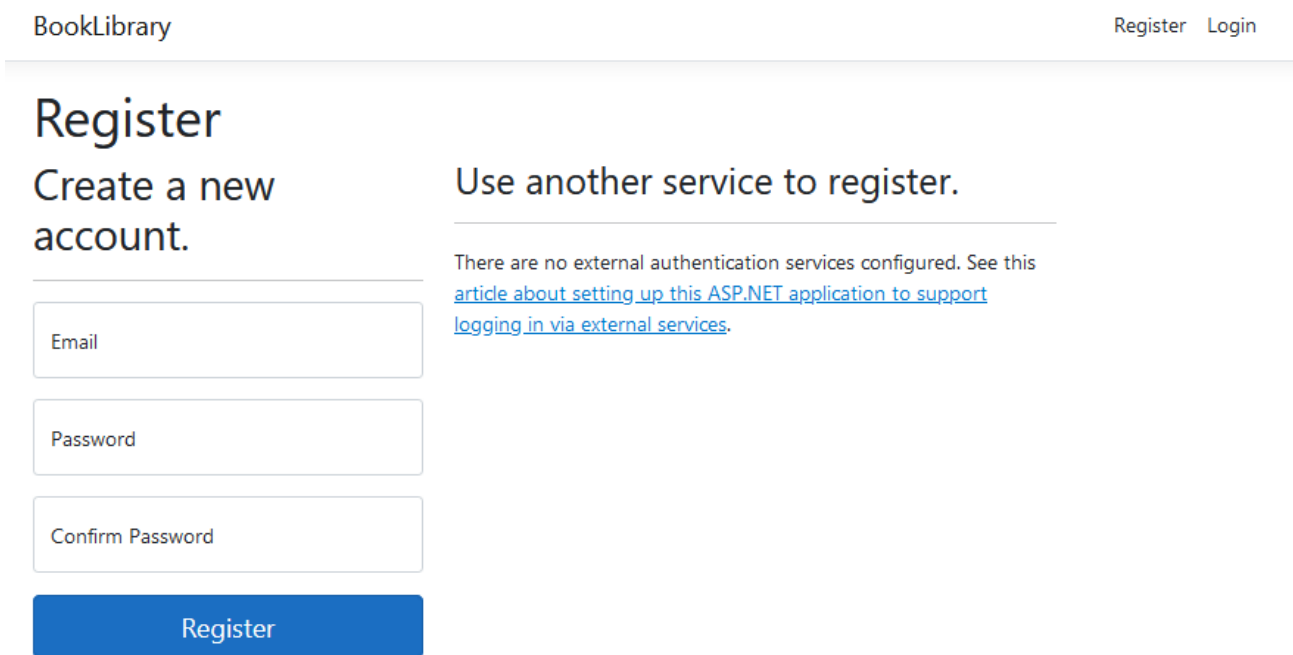


Rys. 2 Diagram UML Przypadków Użycia

6. Opis interfejsu użytkownika



Rys. 3 Strona Startowa



Rys. 4 Strona Rejestracji

Log in

Use a local account
to log in.

The Password field is required.

☐ Remember me?

Log in

[Forgot your password?](#)

[Register as a new user](#)

[Resend email confirmation](#)

Use another service to log in.

There are no external authentication services configured. See this [article about setting up this ASP.NET application to support logging in via external services](#).

Rys. 5 Strona Logowania

Dodaj Autora

LISTA AUTORÓW

Delete Edytuj



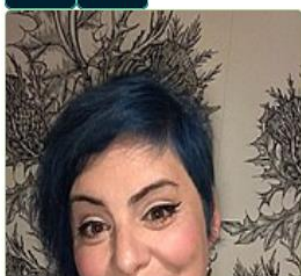
Delete Edytuj



Delete Edytuj



Delete Edytuj



Delete Edytuj



Delete Edytuj



Rys. 6 Strona Listy Autorów

Dodaj Książkę

LISTA KSIĄŻEK

Delete Edytuj

Delete Edytuj

Ostatnie Życzenie

Rating: 999/100

Potężny wiedźmin wyrusza odnaleźć swoją przybraną córkę. Podczas podróży będzie musiał zmierzyć się z groźnymi przeciwnikami, takimi jak: uzależnienie od gwinta, te baby, widły.

Więcej

Delete Edytuj

Delete Edytuj

Delete Edytuj

Delete Edytuj

Rys. 7 Strona Listy Książek

Dodaj Półkę

LISTA PÓŁEK

Delete

Do Przeczytania


Więcej

Rys. 8 Strona Listy Półek

BookLibrary

Book
Authors
Shelfs

admin@admin.pl
Logout



Delete
Edytuj

Ostatnie Życzenie

Total Pages: 332

Rating: 999

ISBN: 9788375780

Published Date:

Description: Potężny wiedźmin wyrusza odnaleźć swoją przybraną córkę. Podczas podróży będzie musiał zmierzyć się z groźnymi przeciwnikami, takimi jak: uzależnienie od gwinta, te baby, widły.

Powrót

Shelf:

Do Przeczytania

Select a shelf

Do Przeczytania


Rys. 9 Strona Szczegółów Książki

BookLibrary

Book
Authors
Shelfs

admin@admin.pl
Logout

Delete
Delete From Shelf
Edytuj



Book ID:

Book Title: Ostatnie Życzenie

Book Rating: 999

Book Description: Potężny wiedźmin wyrusza odnaleźć swoją przybraną córkę. Podczas podróży będzie musiał zmierzyć się z groźnymi przeciwnikami, takimi jak: uzależnienie od gwinta, te baby, widły.

Więcej

Rys. 10 Strona Szczegółów Półki

Oraz formularze dodawania i edycji.

7. Opis kluczowych elementów back-endu

Obsługa zapytań do api:

- MediatR: Wykorzystanie biblioteki w celu implementacji wzorca mediatora w projekcie, do obsługi zdarzeń oraz komunikacji między komponentami.

Walidacja Danych:

- FluentValidation: Wykorzystanie biblioteki w celu walidacji wprowadzanych danych

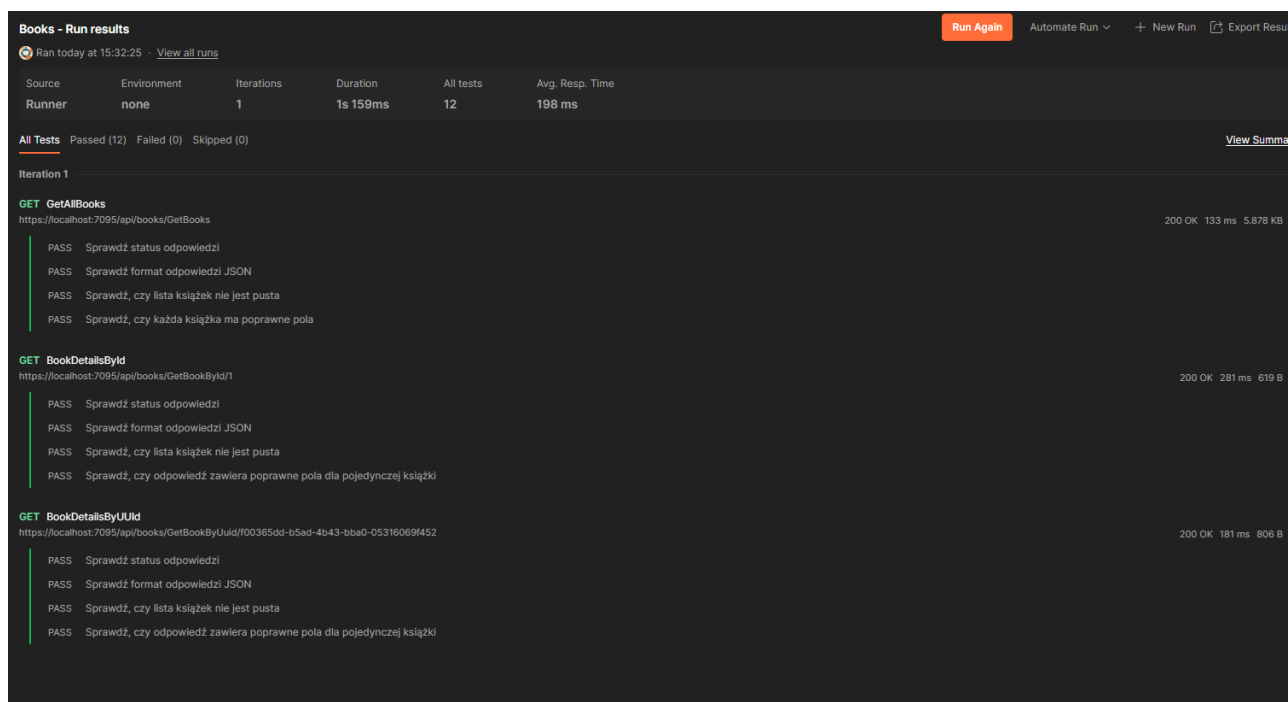
Autoryzacja:

- Identity Framework: Wykorzystanie biblioteki w celu obsługi rejestracji oraz logowania do aplikacji.

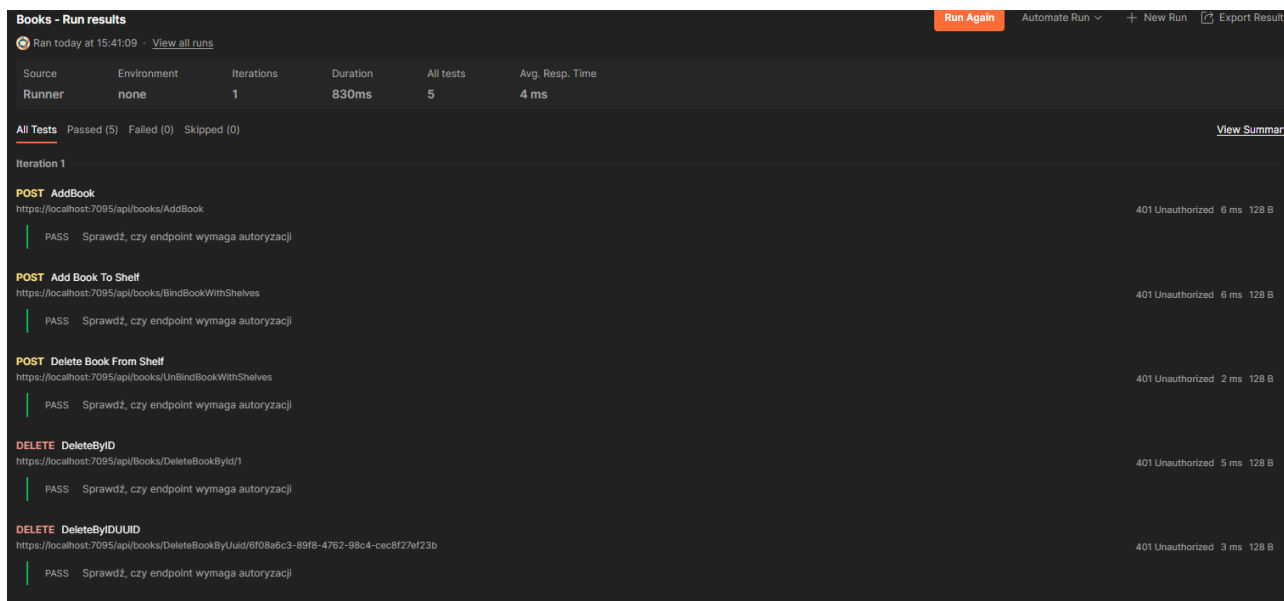
Przekierowywanie:

- SPA proxy: Wykorzystanie biblioteki w celu poprawnego przekierowywania zapytań do api

8. Opis przypadków testów endpointów.



Rys. 11 Testy Api w programie Postman



Rys. 12 Testy Api w programie Postman

9. Opis przypadków testowych

PRZYPADEK NISKOPOZIOMOWY 1

Tytuł: Logowanie do aplikacji za pomocą konta użytkownika.

Cel: Weryfikacja możliwości zalogowania do aplikacji za pomocą poprawnych danych.

Warunki początkowe:

- W aplikacji istnieje aktywne konto użytkownika.
- Użytkownik znajduje się na ekranie logowania do aplikacji.

KROK

REZULTAT

1. Wprowadź poprawne dane w pole Email i Hasło

1. Pola zostały uzupełnione.

2. Naciśnij przycisk Zaloguj

2. Użytkownik został zalogowany do aplikacji.

Priorytet: Wysoki

Wykonanie: Manualne

Estymowany czas: Minuta

Tab. 1 Przypadek Testowy 1

PRZYPADEK NISKOPOZIOMOWY 2

Tytuł: Rejestracja do aplikacji.

Cel: Weryfikacja możliwości rejestracji do aplikacji za pomocą poprawnych danych.

Warunki początkowe:

- W aplikacji nie istnieje aktywne konto użytkownika.
- Użytkownik znajduje się na ekranie rejestracji do aplikacji.

KROK	REZULTAT
1. Wprowadź poprawne dane w pole Email i Hasło	1. Pola zostały uzupełnione.
2. Naciśnij przycisk Zarejestruj	2. Użytkownik został zarejestrowany do aplikacji.
Priorytet: Wysoki	
Wykonanie: Manualne	
Estymowany czas: Minuta	

Tab. 2 Przypadek Testowy 2

PRZYPADEK NISKOPOZIOMOWY 3

Tytuł: Dodanie półki dla użytkownika

Cel: Weryfikacja możliwości dodania półki dla użytkownika za pomocą poprawnych danych.

Warunki początkowe:

- Użytkownik jest zalogowany w aplikacji.
- Użytkownik znajduje się na ekranie listy półek do aplikacji.

KROK	REZULTAT
1. Wprowadź poprawne dane w pole Nazwa Półki	1. Pole zostało uzupełnione.
2. Naciśnij przycisk Dodaj Półkę	2. Półka o podanej nazwie została dodana do aplikacji.
Priorytet: Średni	
Wykonanie: Manualne	
Estymowany czas: Minuta	

Tab. 3 Przypadek Testowy 3

Literatura:

Entity framework:

<https://learn.microsoft.com/en-us/aspnet/core/data/ef-mvc/crud?view=aspnetcore-7.0>

MediatR:

<https://cezarywalenciuk.pl/blog/programing/mediatr-cqrs-i-wzorzec-projektowy-mediator-w-aspnet-core>

Identity:

<https://learn.microsoft.com/pl-pl/aspnet/core/security/authentication/identity?view=aspnetcore-7.0&tabs=visual-studio>