Akademia Górniczo-Hutnicza im. Stanisława Staszica w Krakowie
Wydział Elektrotechniki, Automatyki,
Informatyki i Inżynierii Biomedycznej

Inżynieria oprogramowania
Kierunek Informatyka
III rok
2020/2021

Kacper Duda, Adrian Kuśmierek
Maryna Lukachyk, Yauheniya Padbiarozskaya

1. Cel i założenia projektu

Celem systemu jest utworzenie aplikacji webowej pozwalającej użytkownikom na szybką i prostą wymianę dowolnej ilości książek poprzez spotkania w świecie rzeczywistym oraz na prowadzenie własnego zbioru bibliotecznego.

2. Wymagania funkcjonalne

- Użytkownik rejestruje się na stronie podając nazwę użytkownika, adres e-mail oraz hasło do konta.
- Po zalogowaniu użytkownik ma dostęp do wszystkich kluczowych funkcjonalności serwisu: globalnej bazy książek, własnej kolekcji oraz wymian.
- Użytkownik ma możliwość zarządzania kontem, dodawania i modyfikowania informacji takich jak imię, nazwisko, adres, dane kontaktowe, a także zmiana hasła lub języka.
- Kolekcja książek danego użytkownika składa się z dodanych przez niego książek. Książka zawiera takie informacje jak tytuł, autor, kategoria, data wydania, data dodania (w aplikacji), status, opis.
- Możliwe jest *sortowanie* półki względem tytułu, autora, *filtrowanie* względem kategorii, statusu dostępności oraz *wyszukiwanie* konkretnej książki po słowach kluczowych tytułu lub autora.
- Kolekcja użytkownika umożliwia dodawanie nowych książek, usunięcie książki oraz edytowanie już dodanych jednostek.
- Rejestracja książki jest dostępna z poziomu kolekcji użytkownika. Rejestracja artykułu oznacza dodanie go do kolekcji użytkownika a także globalnej bazy książek jeżeli użytkownik ustawi jej status jako publiczny, w tym celu należy wpisać wymagane informacje do odpowiednich pól. Potrzebne są: tytuł, autor, kategoria (wybrać z proponowanej listy), status (dostępna / prywatna).
- Wymiana książek inicjowana jest przez użytkownika z poziomu globalnej listy książek. Użytkownik deklarujący chęć wymiany w zgłoszeniu przesyła użytkownikowi będącemu właścicielem propozycje wymiany (własną książkę lub pulę książek do wyboru).
- Zainicjowana wymiana jest dostępna w zakładce Wymiany dla obu użytkowników. Zakładka ta przechowuje listę wymian danego użytkownika, status, datę rozpoczęcia, datę zakończenia jeżeli wymiana jest już skończona, a także szczegółowe dane o danej transakcji po wybraniu konkretnego pola na liście.
- Użytkownik B otrzymując ofertę wymiany może ją zaakceptować lub odrzucić. W przypadku zaakceptowania transakcja jest aktywna u użytkownika A który teraz może zatwierdzić lub odrzucić wybór użytkownika B spośród jego propozycji na wymianę.
- Po potwierdzeniu obu stron dostępna jest opcja ustalenia daty i lokalizacji w której ma wystąpić wymiana.
- Przez cały czas trwania transakcji dla użytkowników aktualnie decydujących o wymianie dostępne jest pole z dodatkowym komentarzem dla drugiej strony.

3. Przykłady użycia

1. **Logowanie**
    - **Aktorzy:** Użytkownik
    - **Zdarzenie wyzwalające:** użytkownik wybiera funkcję zalogowania
    - **Warunki początkowe:** użytkownik nie może być zalogowany
    - **Warunki końcowe w przypadku zakończenia powodzeniem:** zmiana statusu użytkownika na „zalogowany”
    - **Warunki końcowe w przypadku zakończenia niepowodzeniem:** status użytkownika się nie zmienia, wysyłana jest informacja o niepowodzeniu logowania

    Scenariusz główny:
    - ➢ System wyświetla formularz logowania.
    - ➢ Użytkownik wprowadza e-mail oraz hasło.
    - ➢ Weryfikacja danych przez system kończy się powodzeniem.
    - ➢ Status użytkownika zostaje zmieniony na „zalogowany”.

    Scenariusz alternatywny:
    - ➢ System wyświetla formularz logowania.
    - ➢ Użytkownik wprowadza e-mail oraz hasło.
    - ➢ Weryfikacja danych przez system kończy się niepowodzeniem.
    - ➢ System wyświetla informacje o niepoprawnych danych.
2. **Edycja profilu**
    - **Aktorzy:** Użytkownik
    - **Zdarzenie wyzwalające:** użytkownik wybiera funkcję edycji profilu
    - **Warunki początkowe:** użytkownik musi być zalogowany
    - **Warunki końcowe w przypadku zakończenia powodzeniem:** profil zostaje edytowany.
    - **Warunki końcowe w przypadku zakończenia niepowodzeniem:** profil nie zostaje edytowany, wysyłana jest informacja o niepowodzeniu edytowania.

    Scenariusz główny:
    - ➢ System wyświetla formularz edycji profilu.
    - ➢ Użytkownik zmienia wybrane dane (hasło, e-mail, miasto)
    - ➢ Weryfikacja danych przez system zakończona powodzeniem.
    - ➢ Dane użytkownika zostają zmienione.

    Scenariusz alternatywny:
    - ➢ System wyświetla formularz edycji profilu.
    - ➢ Użytkownik zmienia wybrane dane (hasło, e-mail, miasto)
    - ➢ Weryfikacja danych przez system kończy się niepowodzeniem.
    - ➢ Dane użytkownika nie zostają zmienione. System wyświetla informacje o niepoprawnych danych.
3. **Rejestracja książki**
    - **Aktorzy:** Użytkownik
    - **Zdarzenie wyzwalające:** użytkownik wybiera zakładkę rejestracji książki
    - **Warunki początkowe:** użytkownik musi być zalogowany, otwarta zakładka rejestracji książki

- **Warunki końcowe w przypadku zakończenia powodzeniem:** książka zostaje dodana na półkę użytkownika
- **Warunki końcowe w przypadku zakończenia niepowodzeniem:** książka nie zostaje dodana na półkę użytkownika, system wyświetla informację o niepowodzeniu

Scenariusz główny:

- ➢ System wyświetla formularz rejestracji nowej książki.
- ➢ Użytkownik wpisuje wymagane dane (tytuł, autor, kategoria, status, komentarz)
- ➢ Weryfikacja danych przez system (wszystkie wymagane dane zostały wpisane) zakończona powodzeniem.
- ➢ Nowa książka została zarejestrowana i dodana na półkę użytkownika.

Scenariusz alternatywny:

- ➢ System wyświetla formularz rejestracji nowej książki.
- ➢ Użytkownik wpisuje wymagane dane (tytuł, autor, kategoria, status, komentarz)
- ➢ Weryfikacja danych przez system (wszystkie wymagane dane zostały wpisane) kończy się niepowodzeniem.
- ➢ Nowa książka nie została zarejestrowana i półka użytkownika pozostaje niezmieniona.

4. **Usuwanie książki**
   - **Aktorzy:** Użytkownik
   - **Zdarzenie wyzwalające:** Użytkownik klika na przycisk ‘usuń książkę’.
   - **Warunki początkowe:** Użytkownik musi być zalogowany i znajdować się w zakładce ‘półka użytkownika’
   - **Warunki końcowe w przypadku zakończenia powodzeniem:** Książka zostaje usunięta z półki użytkownika
   - **Warunki końcowe w przypadku zakończenia niepowodzeniem:** Książka nie zostaje usunięta, wyświetlona jest informacja o niepowodzeniu.

   Scenariusz główny:

   - ➢ Użytkownik wybiera książkę z listy książek na jego półce i klika na przycisk ‘usuń książkę’.
   - ➢ Weryfikacja danych przez system zakończona powodzeniem.
   - ➢ Książka zostaje usunięta z półki użytkownika.

   Scenariusz alternatywny:

   - ➢ Użytkownik wybiera książkę z listy książek na jego półce i klika na przycisk ‘usuń książkę’.
   - ➢ Weryfikacja danych przez system zakończona niepowodzeniem.
   - ➢ Książka nie zostaje usunięta, zostaje wysłana informacja o niepowodzeniu.

5. **Edycja opisu książki**
   - **Aktorzy:** Użytkownik
   - **Zdarzenie wyzwalające:** Użytkownik klika na przycisk ’Edytuj Opis Książki’
   - **Warunki początkowe:** Użytkownik musi być zalogowany i znajdować się w zakładce ‘półka użytkownika’
   - **Warunki końcowe w przypadku zakończenia powodzeniem:** Opis książki zostaje zedytowany.

- **Warunki końcowe w przypadku zakończenia niepowodzeniem:** Opis pozostaje bez zmian, wyświetlona jest informacja, że zmiany nie zostały wprowadzone.

Scenariusz główny:

- ➢ System wyświetla formularz edycji opisu książki.
- ➢ Użytkownik wprowadza zmiany w danych książki (pola: tytuł, autor, kategoria, komentarz).
- ➢ Weryfikacja danych przez system zakończona powodzeniem.
- ➢ Dane książki zostają zmienione.

Scenariusz alternatywny:

- ➢ System wyświetla formularz edycji opisu książki.
- ➢ Użytkownik wprowadza zmiany w opisie książki (pola: tytuł, autor, kategoria, komentarz).
- ➢ Weryfikacja danych przez system zakończona niepowodzeniem (pozostawione puste pola).
- ➢ Dane książki nie zostają zmienione. Zostaje wysłana informacja o niepoprawnie wprowadzonych danych.

6. **Wyszukiwanie książek**
   - **Aktorzy:** Użytkownik
   - **Zdarzenie wyzwalające:** Użytkownik wybiera funkcję wyszukiwania książki.
   - **Warunki początkowe:** Użytkownik musi być zalogowany.
   - **Warunki końcowe w przypadku zakończenia powodzeniem:** Książki odpowiadające wyszukiwaniu zostają wyświetlone.
   - **Warunki końcowe w przypadku zakończenia niepowodzeniem:** Książka nie zostaje znaleziona, wyświetlona jest informacja, że dana książka nie istnieje.

   Scenariusz główny:

   - ➢ System wyświetla formularz szukania książek.
   - ➢ Użytkownik wprowadza dane książki, którą chce znaleźć (tytuł, autor, kategoria).
   - ➢ Weryfikacja danych przez system zakończona powodzeniem.
   - ➢ Książki odpowiadające kryteriom zostają znalezione i wyświetlone.

   Scenariusz alternatywny:

   - ➢ System wyświetla formularz szukania książek.
   - ➢ Użytkownik wprowadza dane książki, którą chce znaleźć (tytuł, autor, kategoria).
   - ➢ System nie znajduje książki o podanych parametrach. Zostaje wysłana informacja o tym że taka książka nie istnieje.

7. **Pełny opis książki**
   - **Aktorzy:** Użytkownik
   - **Zdarzenie wyzwalające:** Użytkownik wybiera funkcję wyświetlenia szczegółowych informacji o książce.
   - **Warunki początkowe:** Użytkownik musi być zalogowany.
   - **Warunki końcowe w przypadku zakończenia powodzeniem:** Informacje o książce zostają wyświetlone.

   Scenariusz główny:

- ➢ Użytkownik wyszukuje książki i wybiera jedną, aby zobaczyć informacje o niej.
- ➢ System wyświetla informacje o książce.

8. **Wymiana (Transakcja)**
   - **Aktorzy:** Użytkownik
   - **Zdarzenie wyzwalające:** Użytkownik wybiera funkcję wymiany książek.
   - **Warunki początkowe:** Użytkownik musi być zalogowany.
   - **Warunki końcowe w przypadku zakończenia powodzeniem:** Transakcja powiodła się. Użytkownicy są umówieni na spotkanie w celu wymiany książek.
   - **Warunki końcowe w przypadku zakończenia niepowodzeniem:** Wymiana nie powiodła się. Użytkownicy otrzymują informację o niepowodzeniu transakcji z powodu anulowania jej przez jedną ze stron.

   Scenariusz główny:
   - ➢ System wyświetla funkcję transakcji książek.
   - ➢ Użytkownik A tworzy ofertę wymiany. Wybiera pulę książek, które może zaoferować w zamian za książki, które chce otrzymać od użytkownika B.
   - ➢ Użytkownik B dostaje ofertę od użytkownika A. Wybiera dostępne książki z puli od użytkownika A, które chce otrzymać w zamian i wysyła informacje o wybranych pozycjach.
   - ➢ Użytkownik A akceptuje ofertę i wysyła do użytkownika B propozycję czasu i miejsca wymiany.
   - ➢ Użytkownik B akceptuje czas oraz miejsce wymiany.
   - ➢ Transakcja zakończona powodzeniem.

   Scenariusz alternatywny 1:
   - ➢ System wyświetla funkcję transakcji książek.
   - ➢ Użytkownik A tworzy ofertę wymiany. Wybiera pulę książek, które może zaoferować w zamian za książki, które chce otrzymać od użytkownika B.
   - ➢ Użytkownik B dostaje ofertę od użytkownika A. Wybiera dostępne książki z puli od użytkownika A, które chce otrzymać w zamian i wysyła informacje o wybranych pozycjach.
   - ➢ Użytkownik A akceptuje ofertę i wysyła do użytkownika B propozycję czasu i miejsca wymiany.
   - ➢ Użytkownik B nie akceptuje miejsca i/lub czasu wymiany. Proponuje inny czas i/lub miejsce wymiany.
   - ➢ Użytkownik A akceptuje wymianę.
   - ➢ Transakcja zakończona powodzeniem.

   Scenariusz alternatywny 2:
   - ➢ System wyświetla funkcję transakcji książek.
   - ➢ Użytkownik A tworzy ofertę wymiany. Wybiera pulę książek, które może zaoferować w zamian za książki, które chce otrzymać od użytkownika B.
   - ➢ Użytkownik B dostaje ofertę od użytkownika A. Nie znajduje jednak pozycji za które mógłby się zamienić, więc odrzuca transakcję.
   - ➢ Transakcja zakończona niepowodzeniem.

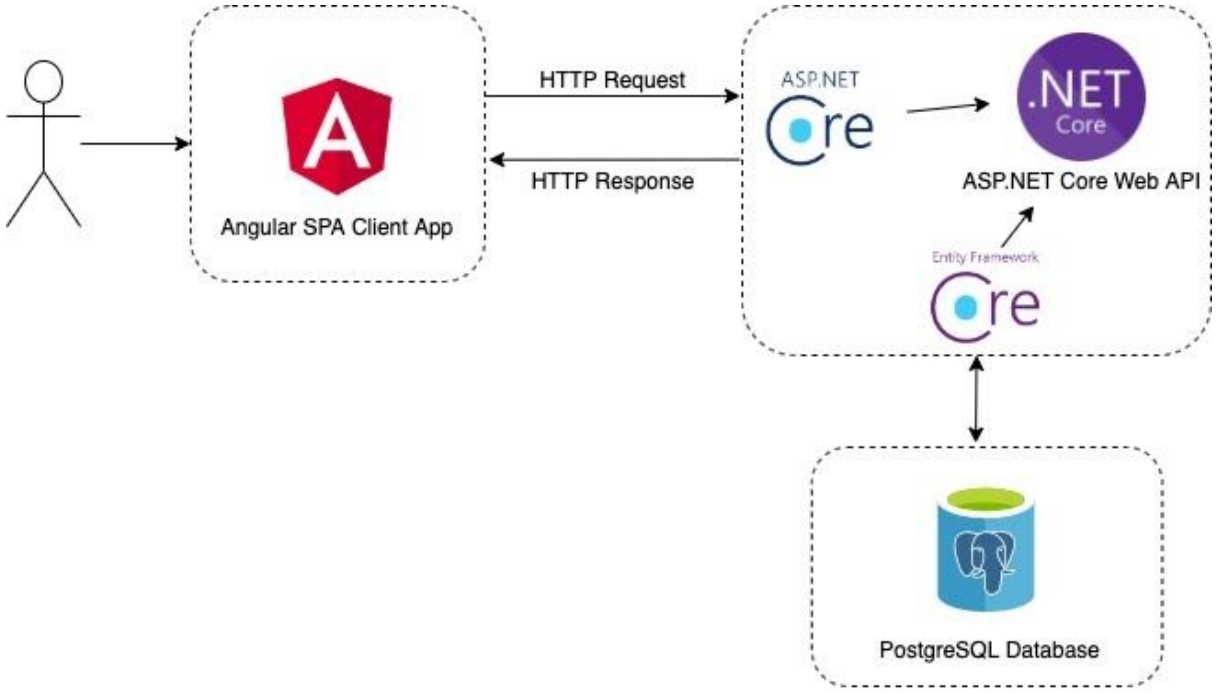
4. Wymagania niefunkcjonalne

- Aplikacja ma być responsywna, przynajmniej na komputerach stacjonarnych i laptopach.
- Aplikacja musi być modularna aby umożliwić większą wydajność poprzez efektywny lazy loading.
- Aplikacja musi posiadać możliwość moderowania i administrowania zasobami przez konta z rozszerzonymi uprawnieniami.
- Aplikacja powinna być kompatybilna z wszystkimi nowoczesnymi wersjami popularnych przeglądarek.
- UI jest responsywny, intuicyjny i oparty na nowoczesnym designie stron internetowych.
- Aplikacja musi być bezpieczna i odporna na niezautoryzowany dostęp do danych.
- Aplikacja musi być odporna na awarie spowodowane nieprawidłowym zachowaniem użytkownika lub błędami systemu, możliwie zapobiegać lub obsługiwać w sposób nieblokujący.
- Aplikacja powinna być łatwo konfigurowalna pod kątem przenoszenia danych, np. użycia nowej bazy danych.
- Aplikacja musi działać szybko i niezawodnie przy dostępie przez wielu użytkowników jednocześnie.
- Aplikacja musi oferować łatwe rozszerzenie (dodanie nowych) wersji językowych.

5. Organizacja projektu

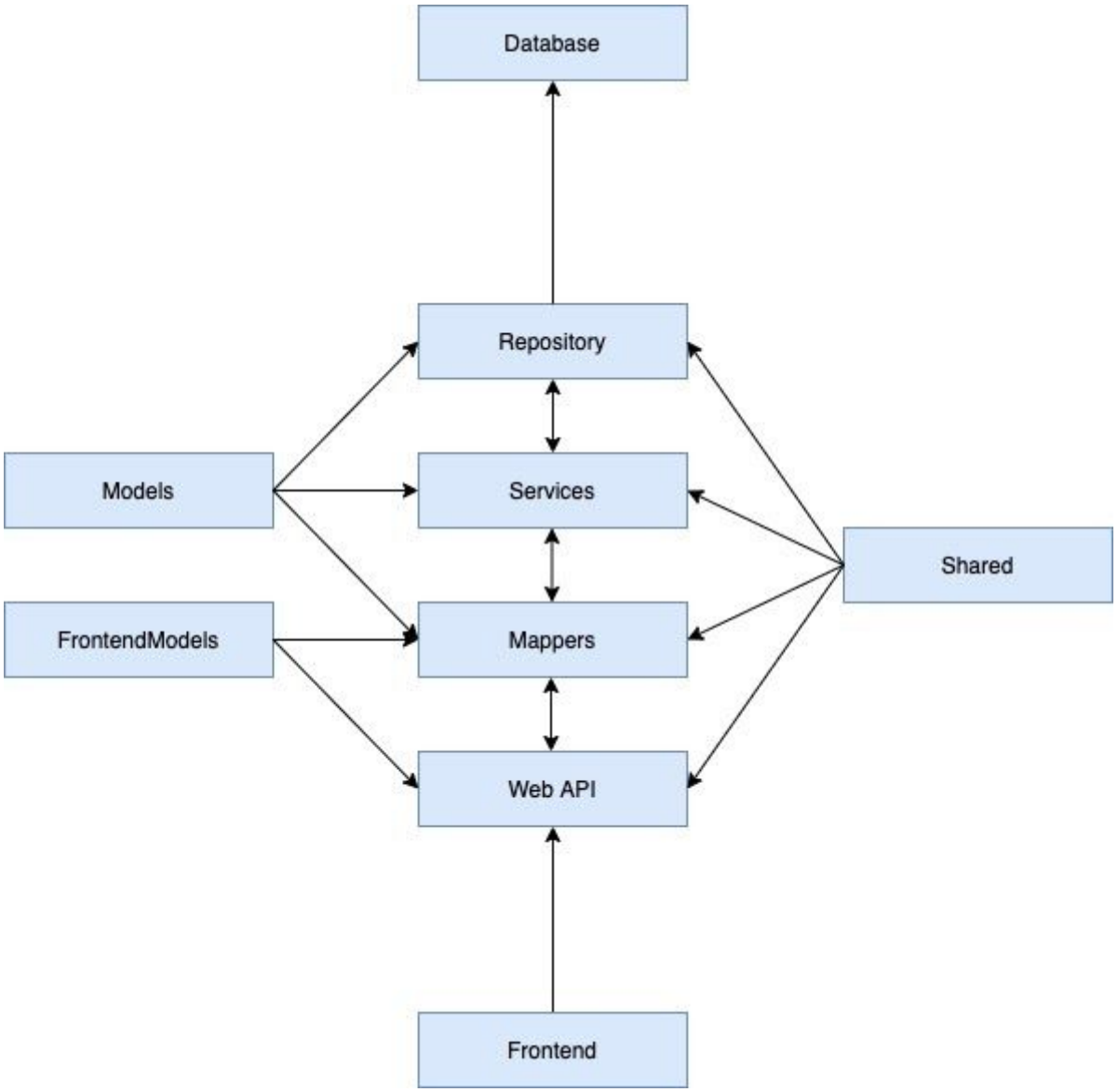
5.1. Architektura fizyczna

Projekt będzie w postaci w pełni kompletnej aplikacji webowej na którą składać się będzie aplikacja kliencka zrealizowana przy pomocy frameworka Angular oraz ASP.NET Core Web API odpowiedzialne za obsługę zapytań z aplikacji webowej, logikę biznesową oraz komunikację z bazą danych.



5.2 Architektura logiczna

5.2.1 Struktura modułów backendu



Architektura backendu składać się będzie ze ściśle współpracujących ze sobą pomniejszych modułów w ramach głównego API w technologii ASP.NET Core.

- **Web API** - główny punkt aplikacji. W nim znajdą się kontrolery obsługujące konkretne endpointy, pliki konfiguracyjne, między innymi dla autoryzacji i autentykacji oraz wstrzykiwania zależności (dependency injection).

- **Frontend Models** - modele odpowiadające tym po stronie frontendu. To na nie będą mapowane dane które otrzymają odpowiednie kontrolery z zapytań HTTP, a także te zwracane w odpowiedziach do aplikacji klienckiej.

- **Models** - modele właściwe po stronie backendu pokrywające się z tymi w bazie danych. To na nich opierać się będzie cała logika biznesowa.

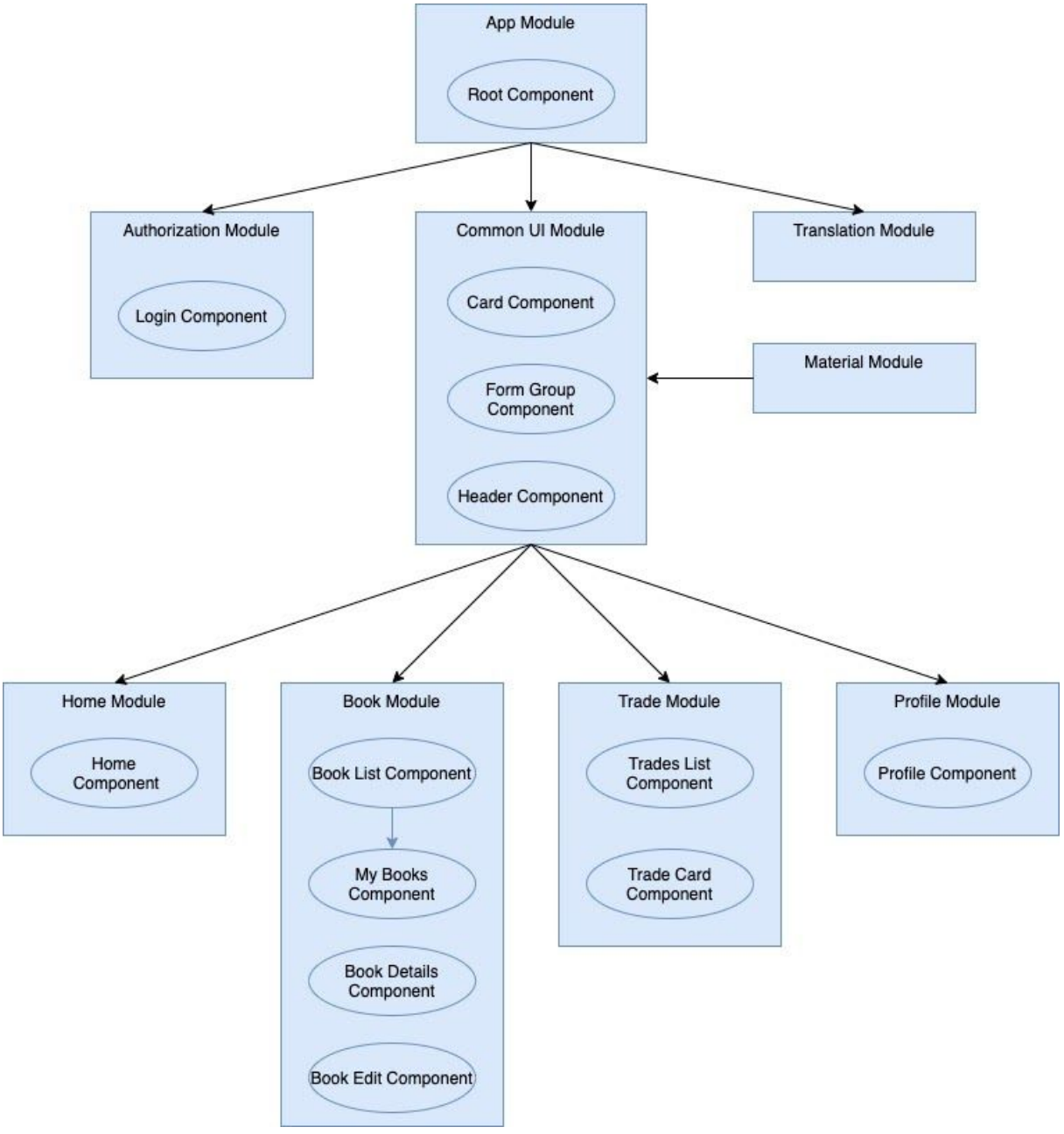
- **Mappers** - tutaj następować będzie konwersja pomiędzy modelami otrzymanymi z modułu Frontend Models, a modelami z modułu Models odpowiadającymi encjom w bazie danych (modele te powinny się pokrywać jednak dobrą praktyką jest ich wyraźne rozdzielenie w celu łatwiejszej implementacji ewentualnych zmian w modelach).

- **Shared** - wszystkie współdzielone zachowania i helpery przydatne w różnych częściach aplikacji powinny się tutaj znaleźć.

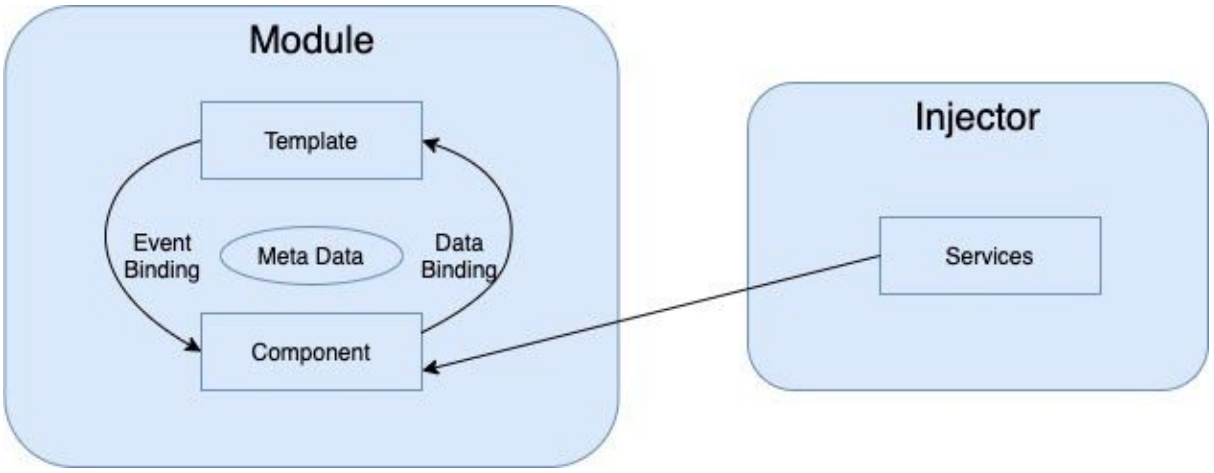
- **Services** - moduł w którym znajdzie się główna logika biznesowa aplikacji.

- **Repository** - data access layer, czyli miejsce w którym znajdą się wszystkie niezbędne funkcjonalności związane z konfiguracją i komunikacją z bazą danych.

5.2.2 Struktura modułów frontendu



Aplikacja kliencka to Single Page Application złożona z luźno powiązanych ze sobą modułów. Poniższy diagram prezentuje schemat budowy większości modułów w aplikacji; składają się na nie liczne komponenty sterujące bezpośrednio lub pośrednio za pomocą wstrzykniętych serwisów UI, który prezentowany jest użytkownikowi poprzez szablony (Template) interaktywne dzięki mechanizmom Data Binding oraz Event Binding.



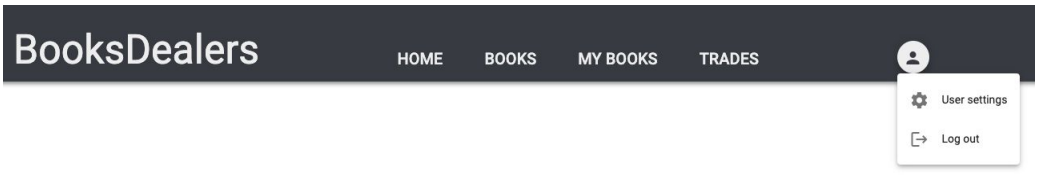
- **Moduł główny (App Module)** - jest punktem startowym aplikacji, zawiera wszystkie potrzebne konfiguracje i główny komponent (Root Component), który zawiera wszystkie aktualnie wyświetlane w aplikacji widoki.

- **Authorization Module** - zawiera komponenty odpowiedzialne za logowanie oraz rejestracje, a także serwis zajmujący się autentykacją i autoryzacją użytkownika poprzez pozyskiwanie i zarządzanie ważnym JWT (JSON Web Token).

- **Translation Module** - konfiguracja biblioteki umożliwiającej sprawne obsługiwanie dodatkowych wersji językowych aplikacji .

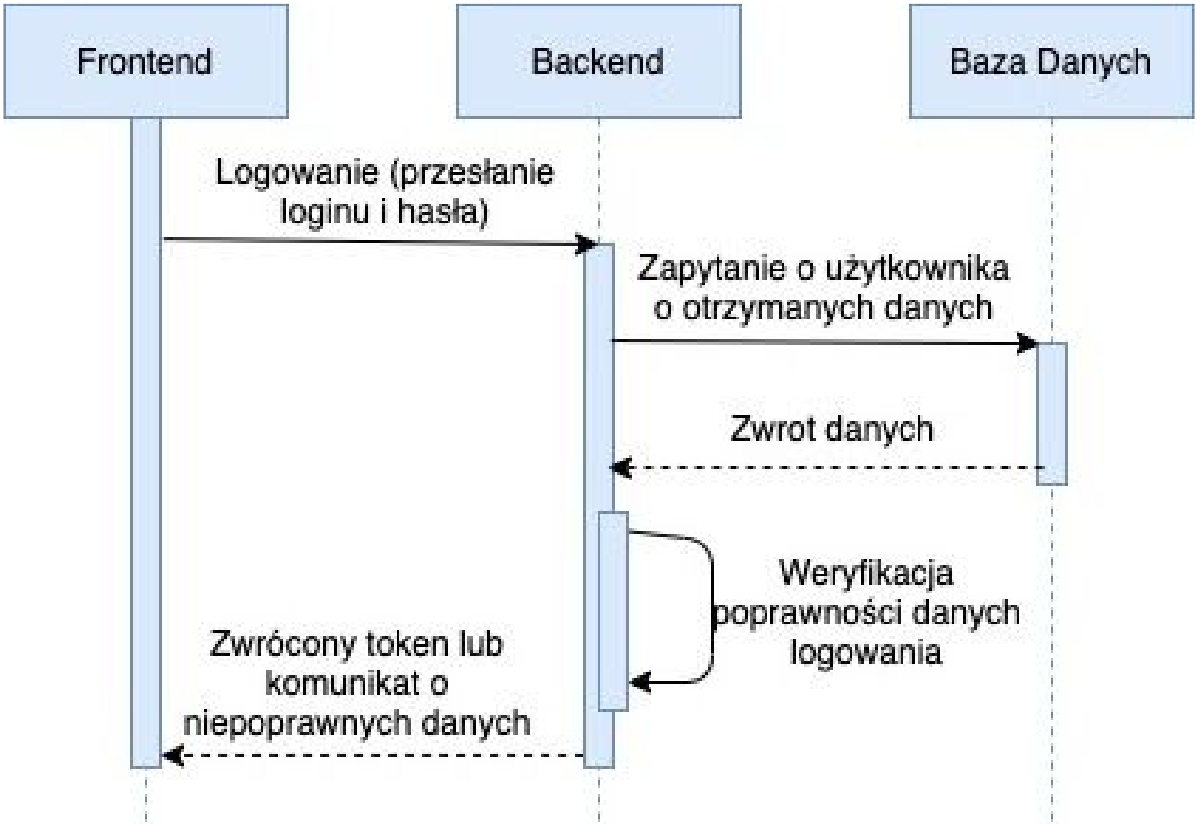
- **Material Module** - osobny moduł służący do przechowywania w jednym miejscu używanych w aplikacji komponentów z biblioteki graficznej Angular Material.

- **Common UI Module** - elementy interfejsu użytkownika współdzielone w całej aplikacji, znajduje się tam również Header Component który dodatkowo służy jako menu do poruszania się po aplikacji.

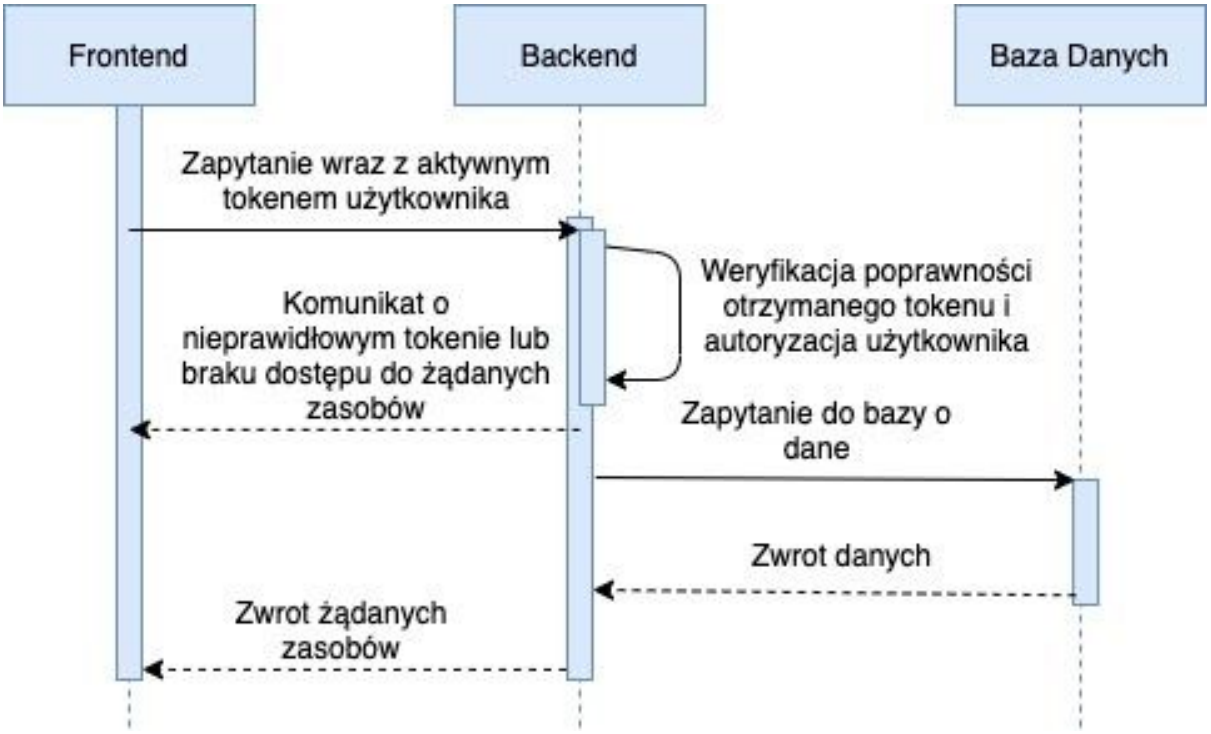


- Pozostałe moduły (**Home, Book, Trade, Profile**) zawierają komponenty tworzące poszczególne strony aplikacji (główna zawartość interfejsu użytkownika), a także dedykowane serwisy do obsługi logiki niezbędnej do poprawnego funkcjonowania aplikacji, np. komunikacja z API w celu uzyskania zasobów.

Diagram obrazujący mechanizm logowania:



Po pomyślnym uwierzytelnieniu schemat zapytań o zasoby do bazy danych będzie wyglądał następująco:



6. Projekt testów

6.1. Testy jednostkowe

Do testów zostaną użyte po stronie frontendu Jasmine test framework, a dla backendu xUnit. Obie technologie zostały wybrane przez wzgląd na swoją popularność i szerokie zastosowanie. Kluczowe będzie pokrycie testami jednostkowymi wszelkiego rodzaju mapperów oraz walidatorów i na to będzie kładziony największy nacisk.

6.2. Testy integralne

W przypadku testów integralnych testowana będzie komunikacja pomiędzy poszczególnymi modułami zarówno w przypadku API jak i frontendu.

6.3. Testy wydajnościowe

Testy wydajnościowe będą polegały na odtworzeniu wszystkich możliwych przypadków użycia aplikacji i zapisaniu plików HAR (HTTP Archive) na podstawie których będzie możliwa analiza. Takie podejście pozwoli na łatwe zlokalizowanie i wyeliminowanie punktów w których aplikacja działa nieprawidłowo lub jej wydajność spada (niepotrzebne requesty, długi czas oczekiwania na odpowiedź z endpointów itp.).

7. Analiza ryzyka

7.1. Macierz ryzyka użyta do klasyfikacji potencjalnych zagrożeń

			SKUTEK				
			Bardzo niski	Niski	Średni	Wysoki	Bardzo wysoki
			1	2	3	4	5
PRAWDOPODOBIEŃSTWO	Prawie pewne	5	Ś	W	K	K	K
	Prawdopodobne	4	Ś	W	W	K	K
	Możliwe	3	N	Ś	W	W	K
	Mało prawdopodobne	2	N	Ś	Ś	W	W
	Rzadkie	1	N	N	Ś	W	W

	Poziom ryzyka	Opis działania
	Niski **(N)**	Poziom ryzyka akceptowany – działania podejmowane w zależności od wymaganych nakładów
	Średni **(Ś)**	Poziom ryzyka nieakceptowany – działanie może zostać przesunięte w czasie, ale wymaga okresowego monitorowania
	Wysoki **(W)**	Poziom ryzyka nieakceptowany – działanie może zostać przesunięte w czasie, ale wymaga stałego monitorowania
	Krytyczny **(K)**	Poziom ryzyka nietolerowany – wymaga natychmiastowego działania

7.2. Lista ryzyk

Potencjalne ryzyko	Prawdopodobieństwo	Skutek	Poziom ryzyka
Atak hakerski	1	5	Wysoki
Błąd w oprogramowaniu	2	2	Średni
Nieautoryzowana modyfikacja danych	1	3	Średni
Nieautoryzowany dostęp do danych	2	2	Średni
Phishing	3	1	Niski
Awaria techniczna komponentów fizycznych	1	4	Wysoki
Spadek wydajności	3	1	Niski
Niepoprawne działanie na urządzeniach mobilnych	2	3	Średni

8. Stos technologiczny

8.1 Użyte technologie

- **Angular** - aktualnie jeden z najpopularniejszych frameworków do tworzenia nowoczesnych aplikacji webowych. Rozbudowana społeczność oraz ciągłe wsparcie i rozwój przez Googla sprawia, że jest to aktualnie jedna z najrozsądniejszych technologii do tworzenia wysokiej klasy aplikacji webowych. Korzystanie z Typescripta i stosunkowo sztywna architektura projektów wymusza na programiście tworzenie oprogramowania w oparciu o dobre nawyki programistyczne, a wbudowane wsparcie dla modularności, zorientowanie na komponenty i server-side rendering gwarantują wysoką wydajność aplikacji tworzonych przy użyciu Angulara.

- **Angular Material** - dedykowana biblioteka UI dla aplikacji w Angularze dostarczająca nowoczesne i funkcjonalne komponenty.

- **Typescript** - aplikacje pisane w Angularze 2+ są nastawione na używanie właśnie tego języka.

- **RxJS** - biblioteka dostarczająca wiele funkcjonalności dla reaktywnego, asynchronicznego programowania pod które Angular w wielu miejscach jest nastawiony, przykładowo HttpClient (komunikacja HTTP w Angularze jest asynchroniczna).

- **HTML/CSS** - nieodłączny element każdej aplikacji webowej.

- **Sass (SCSS)** - jeden z najpopularniejszych preprocesorów CSS znacznie rozszerzający możliwości zwykłego CSS.

- **.NET Core** - flagowy framework do tworzenia oprogramowania wspierany i mocno rozwijany przez Microsoft. .NET Core w odróżnieniu do .NET frameworku jest nastawiony na wieloplatformowość i większą modularność, jest także szybszy i bezpieczniejszy od oprogramowania dedykowanego na Windowsa w związku z czym aktualnie preferowane jest tworzenie aplikacji właśnie przy użyciu .NET Core

- **ASP.NET Core** - opensourcowy następca ASP.NET działający na platformie .NET Core. ASP.NET Core dostarcza rozwiązania umożliwiające tworzenie nowoczesnych i bezpiecznych aplikacji webowych.

- **C#** - wieloparadygmatowy język programowania najczęściej używany i proponowany do tworzenia aplikacji na platformie .NET Core.

- **Entity Framework Core** - object-relational mapper rekomendowany przez Microsoft w aplikacjach opartych na .NET Core.

- **PostgreSQL** - jeden z najpopularniejszych systemów zarządzania relacyjnymi bazami danych; sprawdzone i polecane rozwiązanie.

8.2 Użyte narzędzia

- **Github** - repozytorium dla projektu
- **Visual Studio Code** - IDE do pisania frontendu
- **Visual Studio** - IDE do pisania backendu
- **TSLint** - rozszerzenie do VSC będące kompletnym narzędziem do ciągłego śledzenia i poprawiania kodu pisanego w Typescripcie
- **ReSharper** - analogicznie jak w przypadku TSLint'a, lecz dla kodu pisanego w C#, rozszerzenie dla Visual Studio
- **Swagger** - narzędzie służące do generowania gotowej specyfikacji API
- **Postman** - jedna z najpopularniejszych aplikacji do testowania API, umożliwia szybkie i łatwe wykonywanie dowolnych zapytań w ramach protokołu HTTP

- **Angular CLI** - command line interface dla Angulara, umożliwia między innymi szybkie generowanie komponentów, serwisów, modułów itd.
- **Debugger for Chrome** - narzędzie do debuggowania aplikacji webowych przy użyciu narzędzi programistycznych Chrome