Dokumentacja Projektowa Systemu Rekomendacji Książek

Anna Grelewska, Bartosz Grabiński, Bartosz Sęga, Jakub Lipiński, Maksymilian Kułakowski

2 Lutego 2026

Spis treści

Wprowadzenie

Cel projektu

Celem projektu jest stworzenie kompleksowego systemu rekomendacji książek, który umożliwia użytkownikom odkrywanie nowych pozycji literackich na podstawie ich historii czytania oraz preferencji. Aplikacja integruje nowoczesne technologie webowe z zaawansowanymi algorytmami uczenia maszynowego, aby dostarczać spersonalizowane sugestie.

Zakres funkcjonalności

System oferuje następujące funkcjonalności:

- Rejestracja i logowanie użytkowników: Bezpieczne uwierzytelnianie oparte na tokenach JWT.
- Zarządzanie półkami z książkami: Możliwość dodawania książek do statusów „Przeczytane", „W trakcie" i „Do przeczytania".
- Wyszukiwanie książek: Integracja z Google Books API do przeszukiwania bazy milionów książek.
- System rekomendacji: Sugestie oparte na treści (wektorowe) oraz preferencjach użytkownika (microserwis Books-rec).
- Recenzje i oceny: Możliwość dodawania opinii o przeczytanych książkach.

Architektura systemu

System składa się z trzech głównych komponentów:

1. Frontend: Aplikacja internetowa typu SPA (Single Page Application) zbudowana w React.
2. Backend: REST API oparte na platformie .NET, obsługujące logikę biznesową i komunikację z bazą danych.
3. Books-rec Microservice: Usługa napisana w Pythonie, odpowiedzialna za generowanie rekomendacji przy użyciu wektorowej bazy danych Qdrant.

Poniższy diagram przedstawia ogólną architekturę systemu:

Backend komunikuje się z bazą danych PostgreSQL oraz z mikroserwisem Books-rec poprzez HTTP. Frontend komunikuje się wyłącznie z Backendem, który pełni rolę bramy (Gateway) dla niektórych zapytań.

Backend

Backend aplikacji został zrealizowany w technologii **.NET 8/9** (C#) przy użyciu frameworka **ASP.NET Core**. Pełni on rolę głównego serwera API, obsługując żądania z Frontendu oraz zarządzając danymi.

Technologie

- **Platforma**: .NET 8/9
- **Framework API**: ASP.NET Core Web API
- **Baza Danych**: PostgreSQL
- **ORM**: Entity Framework Core
- **Uwierzytelnianie**: ASP.NET Core Identity + JWT Bearer
- **Dokumentacja API**: Swagger / OpenAPI

Struktura projektu

Projekt backendowy zorganizowany jest w architekturze warstwowej z podziałem na funkcjonalności:

- **Controllers**: Punkty końcowe API (Endpoints).
- **Services**: Logika biznesowa aplikacji.
- **Data / Models**: Modele danych i konfiguracja Entity Framework (DbContext).
- **DTOs (Data Transfer Objects)**: Obiekty służące do przesyłania danych między klientem a serwerem.

Kluczowe komponenty

Uwierzytelnianie (Auth)

System wykorzystuje standard **JWT (JSON Web Token)** do zabezpieczania endpointów. Użytkownicy logują się, otrzymując token, który następnie jest przesyłany w nagłówku `Authorization` każdego żądania. Rejestracja i logowanie obsługiwane są przez `AuthController`, który korzysta z `UserManager` i `SignInManager` dostarczanych przez ASP.NET Identity.

Integracja z Google Books

Serwis `GoogleBooksService` odpowiada za komunikację z zewnętrznym API Google Books. Pozwala na:

- Wyszukiwanie książek po tytule, autorze lub ISBN.
- Pobieranie szczegółowych informacji o książce.

Zarządzanie Półkami (Shelves)

Użytkownicy mogą zarządzać swoimi książkami poprzez `ShelvesController`. Każda książka na półce ma przypisany status (np. „Read”, „Reading”, „ToRead”). Dane te są kluczowe dla systemu rekomendacji.

Rekomendacje

Backend działa jako pośrednik w pobieraniu rekomendacji. `Services/BooksRecService.cs` (lub odpowiednik) komunikuje się z mikroserwisem Pythonowym, przekazując historię użytkownika i odbierając listę sugerowanych książek.

Baza Danych

Aplikacja korzysta z relacyjnej bazy danych **PostgreSQL**. Migracje bazy danych są zarządzane przez Entity Framework Core (`dotnet ef migrations`). Główne tabele to:

- `AspNetUsers`: Informacje o użytkownikach.
- `Books`: Lokalne kopie książek (cache’owane z Google Books lub dodane przez użytkowników).
- `Shelves`: Powiązania użytkowników z książkami.

- `Reviews`: Opinie użytkowników.

Frontend

Frontend aplikacji to nowoczesna aplikacja webowa typu SPA (Single Page Application), zapewniająca płynne i responsywne doświadczenie użytkownika.

Technologie

- Framework: React 19
- Build Tool: Vite
- Język: TypeScript
- Stylowanie: TailwindCSS + Material UI (MUI)
- Routing: React Router Dom
- Klient HTTP: Axios

Struktura projektu

Kod źródłowy znajduje się w katalogu `src` i jest podzielony na:

- `components/`: Reużywalne komponenty UI (np. Navbar, BookCard, Shelves).
- `pages/`: Główne widoki aplikacji (np. HomePage, LoginPage, BookDetails).
- `context/`: Zarządzanie stanem globalnym (np. AuthContext dla sesji użytkownika).
- `hooks/`: Własne hooki Reactowe (np. `useBooks`, `useAuth`).

Główne funkcjonalności UI

Strona Główna (Dashboard)

Prezentuje spersonalizowane rekomendacje w stylu „Netflixa". Książki są pogrupowane w poziome listy (karuzele) według kategorii, np. „Ponieważ przeczytałeś...", „Popularne w Twoim gatunku".

Wyszukiwarka

Interfejs umożliwiający wpisywanie zapytań do Google Books. Wyniki są prezentowane w formie siatki kart, z możliwością natychmiastowego dodania książki na półkę.

Widok Szczegółów Książki

Wyświetla pełne informacje o książce (opis, autor, data wydania, okładka) oraz sekcję recenzji. Użytkownik może zmienić status książki na półce bezpośrednio z tego widoku.

Profil Użytkownika i Półki

Dedykowana strona, gdzie użytkownik widzi swoje książki podzielone na kategorie (Przeczytane, W trakcie, Do przeczytania). Umożliwia filtrowanie i sortowanie własnej biblioteczki.

Stylowanie

Aplikacja wykorzystuje TailwindCSS do układu (layout) i responsywności oraz Material UI do gotowych komponentów interaktywnych (przyciski, pola tekstowe, modale), co przyspiesza development i zapewnia spójny wygląd.

Mikroserwis Rekomendacji (books-rec)

Mikroserwis `books-rec` jest „mózgiem" systemu rekomendacji. Jest to niezależna usługa napisana w języku Python, która wykorzystuje techniki przetwarzania języka naturalnego (NLP) i wektorowe bazy danych.

Technologie

- Język: Python 3.10+
- Framework API: FastAPI
- Baza Wektorowa: Qdrant
- Serwer Aplikacji: Uvicorn
- Biblioteki ML: Sentence-Transformers (do generowania embeddingów)

Zasada działania

1. Indeksowanie Książek

Książki (tytuły, opisy, gatunki) są przetwarzane przez model językowy (np. wielojęzyczny model sentence-transformers), który zamienia tekst na wektory liczbowe (embeddingi). Te wektory są zapisywane w bazie danych Qdrant. Dzięki temu podobne semantycznie książki znajdują się blisko siebie w przestrzeni wektorowej.

2. Generowanie Rekomendacji

Gdy Backend prosi o rekomendacje dla użytkownika, mikroserwis:

1. Analizuje historię czytania użytkownika (książki ocenione pozytywnie lub przeczytane).
2. Wyszukuje w bazie Qdrant wektory, które są „blisko" wektorów książek z historii użytkownika.
3. Filtruje wyniki, aby nie polecać książek już przeczytanych.
4. Grupuje wyniki w kategorie (np. „Podobne do X").

Kluczowe Endpointy

- `POST /recommend`: Główny endpoint. Przyjmuje listę przeczytanych książek, zwraca listę rekomendacji.
- `POST /index`: Dodaje nowe książki do indeksu wektorowego.
- `GET /health`: Sprawdza stan usługi i połączenie z bazą Qdrant.
- `GET /google-books/search`: Proxy do wyszukiwania i buforowania wyników z Google Books.

Baza Wektorowa Qdrant

Qdrant działa jako kontener Dockerowy. Przechowuje ona nie tylko wektory, ale też metadane książek (tytuł, autor, okładka), co pozwala na szybkie zwrócenie pełnych informacji o rekomendowanych pozycjach bez konieczności dodatkowych zapytań do głównej bazy danych.

Uruchomienie i Instalacja

Aby uruchomić system lokalnie, wymagane jest posiadanie zainstalowanych narzędzi: Docker oraz Docker Compose.

Wymagania wstępne

- Docker Desktop / Docker Engine
- Git
- (Opcjonalnie) .NET 8 SDK, Node.js 20+, Python 3.10+ (do uruchamiania bez Dockera)

Uruchomienie za pomocą Docker Compose

Najprostszym sposobem na uruchomienie całej aplikacji (Backend, Frontend, Mikroserwis, Bazy danych) jest użycie orkiestracji kontenerów.

1. Sklonuj repozytorium:

   ```
   git clone <adres-repozytorium>
   cd ProgramowanieZespolowe2025
   ```

2. Uruchom kontenery: Upewnij się, że jesteś w głównym katalogu projektu (lub tam, gdzie znajduje się główny `docker-compose.yml`, jeśli taki istnieje - w przeciwnym razie należy uruchomić usługi indywidualnie lub stworzyć główny plik compose).

   W strukturze projektu pliki `docker-compose.yml` znajdują się w podkatalogach backendu i innej konfiguracji. Poniżej instrukcja dla typowej konfiguracji developerskiej:

   Można uruchomić usługi niezależnie.

   Baza danych i Backend:

   ```
   cd backend
   docker-compose up -d --build
   ```

   To uruchomi bazę PostgreSQL oraz API backendowe.

   Mikroserwis Rekomendacji:

   ```
   cd ../books-rec
   # Upewnij się, że posiadasz plik .env lub odpowiednią konfigurację
   docker run -p 6333:6333 -d qdrant/qdrant
   # Następnie uruchom aplikację Python (lokalnie lub przez Dockerfile)
   pip install -r requirements.txt
   python run_app.py
   ```

   Frontend:

   ```
   cd ../frontend
   npm install
   npm run dev
   ```

Konfiguracja zmiennych środowiskowych

Przed uruchomieniem należy upewnić się, że pliki `.env` w katalogach `backend/` i `books-rec/` są poprawnie uzupełnione kluczami (np. klucz JWT, parametry bazy danych). Przykładowy `.env` dla backendu:

```
Jwt__Key=TwojBardzoTajnyKlucz123!
ConnectionStrings__DefaultConnection=Host=localhost;Database=library_db;Username=postgres;Password=
```

Dostęp do aplikacji

Po poprawnym uruchomieniu, serwisy powinny być dostępne pod adresami:

- Frontend: `http://localhost:5173`
- Backend API (Swagger): `http://localhost:5000/swagger`
- Books-Rec API (Docs): `http://localhost:8000/docs`
- Qdrant Dashboard: `http://localhost:6333/dashboard`