Aplikacja do zarządzania biblioteką filmów

Contents

Zakres projektu

Projekt obejmuje zbudowanie aplikacji WWW do zarządzania biblioteką filmów. Zaimplementowane zostaną funkcjonalności zarówno po stronie serwerowej, jak i przeglądarkowej. Głównym celem jest spełnienie wymagań poziomu 1, a następnie rozszerzenie projektu o funkcjonalności z poziomów 2 i 3.

Wymagania poziomu 1

- ➔ W projekcie wykorzystanych będzie 8 tabel z wyłączeniem tabel asocjacyjnych (w tym takich reprezentujących relacje wiele-do-wiele z dodatkowymi kolumnami). Dodatkowo:
  a) Każda tabela posiadać będzie co najmniej 2 kolumny
  b) Wykorzystane zostaną 3 różne typy danych (int, varchar, date).
- ➔ Zapewnione zostaną operacje CRUD dla wszystkich tabel, w tym operacje:
  a) pozwalające na dodawanie, edycję i usuwanie rekordów do bazy danych,
  b) zwracające listę rekordów z wybranymi kolumnami dla każdej tabeli,
  c) zwracające widoki szczegółowe (wszystkie kolumny + rekordy połączone relacjami).
- ➔ Zaimplementowana zostanie walidacja danych zarówno po stronie klienta, jak i serwera, co ma zapobiec wprowadzaniu niepoprawnych danych.
- ➔ Wykorzystane zostaną następujące technologie:
  a) Frontendu: React
  b) Backendu: ASP.NET
  c) Baza danych: MySQL
- ➔ Sporządzona zostanie instrukcja konfiguracji i uruchomienia aplikacji
- ➔ Przygotowany zostanie krótki dokument opisowy zawierający wykaz zaimplementowanych funkcjonalności ze wskazaniem, czy dana funkcjonalność jest przedmiotem kontroli dostępu.

Wymagania poziomu 2

Zaimplementowane zostaną:

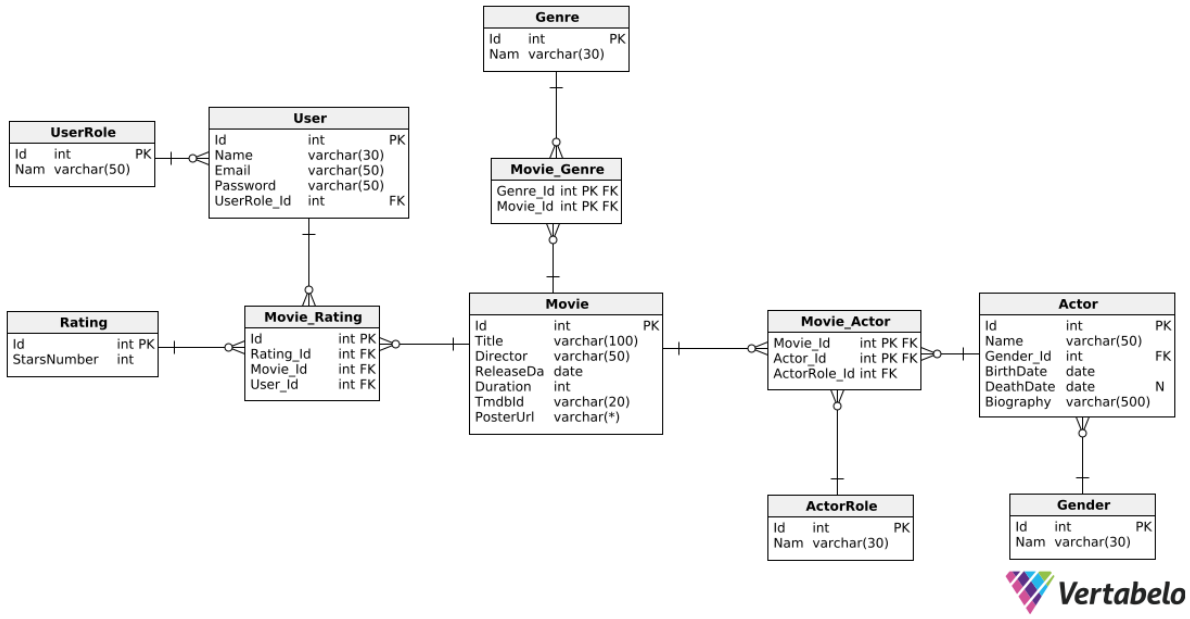
- ➔ Rejestracja i logowanie użytkowników.
- ➔ Różne funkcjonalności w zależności od statusu użytkownika:
  a) Gość: przeglądanie danych.
  b) Zalogowany użytkownik: dodawanie opinii na temat filmów.
- ➔ Paginacja w widokach list rekordów.

Wymagania poziomu 3

- ➔ Frontend zaimplementowany będzie jako SPA (Single Page Application) z wykorzystaniem frameworka React.
- ➔ Dostępne będą trzy role użytkowników:
  a) Gość: przeglądanie danych.
  b) Zalogowany użytkownik: dodawanie opinii na temat filmów.
  c) Administrator: zarządzanie danymi.
- ➔ Użytkownicy będą mieć ograniczony dostęp do zasobów w zależności od ich roli (np. zalogowany użytkownik będzie mógł modyfikować tylko dodane przez siebie opinie).
- ➔ Projekt wspierać będzie co najmniej dwa języki (w tym język polski i angielski).

Model danych

Diagram związków encji



Opisy tabel

Tabela Actor

Nazwa pola	Typ pola	Opis
Id	int	Unikalny identyfikator aktora.
Name	varchar(50)	Imię i nazwisko aktora.
Gender_Id	Int	Klucz obcy wskazujący na płeć aktora w tabeli **Gender**.
BirthDate	date	Data urodzenia aktora.
DeathDate	date (nullable)	Data śmierci aktora (jeśli aktor nie żyje).
Biography	varchar(500)	Biografia aktora.

Tabela Gender

Nazwa pola	Typ pola	Opis
Id	int	Unikalny identyfikator płci.
Name	varchar(30)	Nazwa płci (np. "Male", "Female").

Tabela ActorRole

Nazwa pola	Typ pola	Opis
Id	int	Unikalny identyfikator roli aktora.
Name	varchar(30)	Nazwa roli aktora.

Tabela Movie

Nazwa pola	Typ pola	Opis
Id	int	Unikalny identyfikator filmu.
Title	varchar(100)	Tytuł filmu.
Director	varchar(50)	Imię i nazwisko reżysera filmu.
ReleaseDate	date	Data premiery filmu.
Duration	int	Czas trwania filmu w minutach.
TmdbId	varchar(20)	Identyfikator filmu w bazie TMDB (The Movie Database).
PosterUrl	varchar(*)	Adres URL wskazujący na plakat filmu.

Tabela Genre

Nazwa pola	Typ pola	Opis
Id	int	Unikalny identyfikator gatunku filmu.
Name	varchar(30)	Nazwa gatunku filmu (np. "Action", "Drama").

Tabela Rating

Nazwa pola	Typ pola	Opis
Id	int	Unikalny identyfikator oceny.
StartNumber	int	Liczba gwiazdek przyznanych filmowi (np. od 1 do 5).

Tabela Movie_Actor

Nazwa pola	Typ pola	Opis
Movie_Id	int	Klucz obcy wskazujący na tabelę **Movie**.
Actor_Id	int	Klucz obcy wskazujący na tabelę **Actor**.
ActorRole_Id	int	Klucz obcy wskazujący na tabelę **ActorRole**.

Tabela Movie_Genre

Nazwa pola	Typ pola	Opis
Movie_Id	int	Klucz obcy wskazujący na tabelę **Movie**.
Genre_Id	int	Klucz obcy wskazujący na tabelę **Genre**.

Tabela Movie_Rating

Nazwa pola	Typ pola	Opis
Id	int	Unikalny identyfikator oceny filmu.
Movie_Id	int	Klucz obcy wskazujący na tabelę **Movie**.

Rating_Id	int	Klucz obcy wskazujący na tabelę **Rating**.
User_Id	int	Klucz obcy wskazujący na tabelę **User**.

Tabela User

Nazwa pola	Typ pola	Opis
Id	int	Unikalny identyfikator użytkownika.
Name	varchar(30)	Nazwa użytkownika.
Email	varchar(50)	Adres e-mail użytkownika.
Password	varchar(50)	Hasło użytkownika.
UserRole_Id	int	Klucz obcy wskazujący na tabelę **UserRole**.

Tabela UserRole

Nazwa pola	Typ pola	Opis
Id	int	Unikalny identyfikator roli aktora.
Name	varchar(50)	Nazwa roli (np. "Admin", "User").

Wykaz zaimplementowanych funkcjonalności

Ogólna zasada: dodawać recenzje może tylko użytkownik z rolą **User**, a dodawać, modyfikować i usuwać filmy, aktorów oraz gatunki filmów może tylko użytkownik z rolą **Admin**. Pozostałe końcówki są dostępne dla wszystkich.

ActorsController

1. **GetAllGenders**
    - Endpoint: GET /api/actors/get-all-genders
    - Dostęp: Publiczny (brak wymagań).
2. **GetAllActorRoles**
    - Endpoint: GET /api/actors/get-all-actor-roles
    - Dostęp: Publiczny (brak wymagań).
3. **AddNewActorRole**
    - Endpoint: POST /api/actors/add-new-actor-role
    - Dostęp: Wymaga roli **Admin**.
4. **GetActorById**
    - Endpoint: GET /api/actors/get-actor-by-id
    - Dostęp: Publiczny (brak wymagań).

5. **GetActorByName**
    - Endpoint: GET /api/actors/get-actor-by-name
    - Dostęp: Publiczny (brak wymagań).
6. **GetAllActors**
    - Endpoint: GET /api/actors/get-all-actors
    - Dostęp: Publiczny (brak wymagań).
7. **AddNewActor**
    - Endpoint: POST /api/actors/add-new-actor
    - Dostęp: Wymaga roli **Admin**.
8. **UpdateActor**
    - Endpoint: PUT /api/actors/update-actor-by-id
    - Dostęp: Wymaga roli **Admin**.
9. **DeleteActor**
    - Endpoint: DELETE /api/actors/delete-actor-by-id
    - Dostęp: Wymaga roli **Admin**.

MoviesController

1. **GetAllGenres**
    - Endpoint: GET /api/movies/get-all-genres
    - Dostęp: Publiczny (brak wymagań).
2. **AddNewGenre**
    - Endpoint: POST /api/movies/add-new-genre
    - Dostęp: Wymaga roli **Admin**.
3. **UpdateGenreById**
    - Endpoint: PUT /api/movies/update-genre-by-id
    - Dostęp: Wymaga roli **Admin**.
4. **DeleteGenreById**
    - Endpoint: DELETE /api/movies/delete-genre-by-id
    - Dostęp: Wymaga roli **Admin**.
5. **GetMovieById**
    - Endpoint: GET /api/movies/get-movie-by-id
    - Dostęp: Publiczny (brak wymagań).

6. **GetMovieByTitle**
    - Endpoint: GET /api/movies/get-movie-by-title
    - Dostęp: Publiczny (brak wymagań).
7. **GetMovieByTmdbId**
    - Endpoint: GET /api/movies/get-movie-by-tmdb-id
    - Dostęp: Publiczny (brak wymagań).
8. **GetAllMovies**
    - Endpoint: GET /api/movies/get-all-movies
    - Dostęp: Publiczny (brak wymagań).
9. **GetAllMoviesSegmented**
    - Endpoint: GET /api/movies/get-all-movies-segmented
    - Dostęp: Publiczny (brak wymagań).
10. **GetAllMoviesSegmentedWithCount**
    - Endpoint: GET /api/movies/get-all-movies-segmented-with-count
    - Dostęp: Publiczny (brak wymagań).
11. **Search**
    - Endpoint: GET /api/movies/search
    - Dostęp: Publiczny (brak wymagań).
12. **GetAverageMovieRating**
    - Endpoint: GET /api/movies/get-average-movie-rating
    - Dostęp: Publiczny (brak wymagań).
13. **AddNewMovie**
    - Endpoint: POST /api/movies/add-new-movie
    - Dostęp: Wymaga roli **Admin**.
14. **UpdateMovie**
    - Endpoint: PUT /api/movies/update-movie-by-id
    - Dostęp: Wymaga roli **Admin**.
15. **DeleteMovie**
    - Endpoint: DELETE /api/movies/delete-movie-by-id
    - Dostęp: Wymaga roli **Admin**.
16. **GetMovieRatings**

- Endpoint: GET /api/movies/get-movie-ratings
- Dostęp: Publiczny (brak wymagań).

17. **GetAllMovieRatingsAddedBy**
    - Endpoint: GET /api/movies/get-all-movie-ratings-added-by
    - Dostęp: Publiczny (brak wymagań).
18. **AddMovieRating**
    - Endpoint: POST /api/movies/add-movie-rating
    - Dostęp: Wymaga roli **User**.
19. **UpdateMovieRatingAddedBy**
    - Endpoint: PUT /api/movies/update-movie-rating-added-by
    - Dostęp: Publiczny (brak wymagań).
20. **DeleteMovieRatingAddedBy**
    - Endpoint: DELETE /api/movies/delete-movie-rating-added-by
    - Dostęp: Publiczny (brak wymagań).

UsersController

1. **AddNewUserRole**
    - Endpoint: POST /api/users/add-new-user-role
    - Dostęp: Wymaga roli **Admin**.
2. **DeleteUserRole**
    - Endpoint: DELETE /api/users/delete-user-role
    - Dostęp: Publiczny (brak wymagań).
3. **GetUserById**
    - Endpoint: GET /api/users/get-user-by-id
    - Dostęp: Publiczny (brak wymagań).
4. **GetUserByEmail**
    - Endpoint: GET /api/users/get-user-by-email
    - Dostęp: Publiczny (brak wymagań).
5. **GetAllUsers**
    - Endpoint: GET /api/users/get-all-users
    - Dostęp: Publiczny (brak wymagań).
6. **AddNewUser**

- Endpoint: POST /api/users/add-new-user
- Dostęp: Publiczny (brak wymagań).

7. **DeleteUser**
   - Endpoint: DELETE /api/users/delete-user
   - Dostęp: Publiczny (brak wymagań).

SecurityController

1. **Register**
   - Endpoint: POST /api/security/register
   - Dostęp: Publiczny (brak wymagań).
2. **Login**
   - Endpoint: POST /api/security/login
   - Dostęp: Publiczny (brak wymagań).
3. **Refresh**
   - Endpoint: POST /api/security/refresh
   - Dostęp: Publiczny (brak wymagań).

Wykaz kroków niezbędnych do uruchomienia aplikacji

1. Przygotowanie bazy danych i backend-u:

- Aplikacja wykorzystuje lokalną bazę danych, która jest tworzona za pomocą Entity Framework.
- Upewnij się, że masz zainstalowany SQL Server Management Studio 19 oraz .NET SDK w wersji kompatybilnej z projektem (np. .NET 8).

Kroki:

1. Utwórz migrację w projekcie backendowym:
   ```
   >> dotnet ef migrations add Init
   ```
2. Zaktualizuj bazę danych:
   ```
   >> dotnet ef database update
   ```
3. Uruchom aplikację

2. Frontend

- Frontend jest zbudowany w React i używa npm do zarządzania paczkami.

Kroki:

1. Aby zainstalować wymagane paczki:
   ```
   >> npm install
   ```
2. Następnie uruchom frontend:
   ```
   >> npm start
   ```