PROGRAMOWANIE ZAAWANSOWANE

Dokumentacja techniczna projektu zaliczeniowego

Tytuł projektu: System oceny filmów

Autorzy projektu:

- Maria Blandzi 138451
- Daniel Urbanowicz 138434

Opis projektu

Proponowana aplikacja umożliwi jej użytkownikom zarządzanie personalną listą filmów oraz dokonywanie ich oceny.

Ocena filmu możliwa będzie w skali od 1 (nie polecam) do 5 (polecam).
Oprócz oceny liczbowej możliwe będzie również pozostawienie tekstowego opisu (zwanego dalej recenzją), stanowiącego z założenia opis wrażeń po obejrzeniu filmu.

Niezalogowani użytkownicy będą mogli przeglądać listę filmów oraz zapoznać się z ocenami innych osób.
Zalogowani użytkownicy mogą dodawać filmy do własnej listy filmów „Do obejrzenia” oraz oznaczać je jako „Obejrzane”.

Dla filmów oznaczonych jako “Obejrzane” dostępna będzie funkcjonalność zarządzania opiniami i recenzjami, w ramach której możliwe będzie pozostawienie oceny i opcjonalnej recenzji, ich edycja oraz usuwanie.

1. Specyfikacja wykorzystanych technologii

1) Wersja platformy

Aplikacja została zbudowana w oparciu o C# i .NET 8.0, co zapewnia najnowsze funkcje i wysoką wydajność w kontekście aplikacji webowych.

2) Wykorzystane biblioteki NuGet

W projekcie wykorzystano następujące biblioteki:

a) **Microsoft.AspNetCore.Diagnostics.EntityFrameworkCore** (8.0.8) → Obsługuje diagnostykę błędów w kontekście bazy danych.
b) **Microsoft.AspNetCore.Identity.EntityFrameworkCore** (8.0.8) → Odpowiada za integrację systemu tożsamości (Identity) z bazą danych, zarządzaną przez Entity Framework Core.
c) **Microsoft.AspNetCore.Identity.UI** (8.0.8) → Dostarcza gotowe komponenty interfejsu użytkownika dla systemu tożsamości, takie jak formularze logowania czy rejestracji.
d) **Microsoft.EntityFrameworkCore.SqlServer** (8.0.8) → Umożliwia korzystanie z SQL Server jako bazy danych dla aplikacji.
e) **Microsoft.EntityFrameworkCore.Tools** (8.0.11) - Zapewnia narzędzia wspierające zarządzanie migracjami i bazą danych w Entity Framework Core.
f) **Microsoft.VisualStudio.Web.CodeGeneration.Design** (8.0.7) → Narzędzia do generowania kodu, w tym automatycznego tworzenia kontrolerów i widoków w Visual Studio.

3) Baza danych

Aplikacja korzysta z **SQL Server[**SQL Server Express LocalDB**]** jako systemu zarządzania bazą danych. Połączenie z bazą danych definiowane jest w pliku konfiguracyjnym aplikacji (appsettings.json), a migracje i zarządzanie schematem danych odbywają się za pomocą **Entity Framework Core**.

2. instrukcje pierwszego uruchomienia projektu

1) Uruchom Visual Studio.
2) Wybierz opcję “Klonuj repozytorium”.

3) W polu adresu URL repozytorium wklej link: http://github.com/Basilisk-404/MoviesReviewer
4) Wybierz pusty folder w którym projekt zostanie zapisany i potwierdź.
5) Po zakończeniu klonowanie, otwórz konsole NuGet (z narzędzi) i wykonaj polecenie 'Update-Database'
6) Uruchom aplikację, klikając zielony przycisk uruchom na pasku narzędzi.

3. Struktura projektu

Projekt **MoviesReviewer** to aplikacja webowa napisana w ASP.NET Core MVC. Umożliwia użytkownikom zarządzanie filmami, oznaczanie ich preferencji oraz tworzenie recenzji. Struktura projektu opiera się na:

1) **Modelach**: Reprezentują dane aplikacji oraz zawierają walidację.
2) **Kontrolerach**: Realizują logikę biznesową i odpowiadają za obsługę żądań HTTP.
3) **Widokach**: Odpowiadają za generowanie interfejsu użytkownika.
4) **Baza danych**: Obsługiwana przez **Entity Framework**, definiująca relacje między tabelami.

Główne foldery:

- Models: Zawiera klasy reprezentujące strukturę danych w aplikacji.
- Controllers: Zawiera logikę odpowiadającą za obsługę zapytań i interakcje użytkownika.
- Views: Przechowuje pliki Razor odpowiadające za interfejs użytkownika.
- Enums: Przechowuje klasę PreferenceType, zawierającą możliwe typy preferencji
- Dtos: Przechowuje klasę PreferenceTypeDto służącą do przekazywania danych dot. preferencji pomiędzy kontrolerem a widokami

Głównym elementem projektu są filmy (Movie). Na ich podstawie tworzona jest lista preferencji (Preference). Dzięki współistnieniu filmów i preferencji możliwe jest tworzenie opinii i recenzji (Reviews).

4. Modele

1) ErrorViewModel.cs

CEL: Model używany do wyświetlania szczegółów błędów w aplikacji.
POLA:

a) RequestId: *(string?)* — ID zapytania, które spowodowało błąd.

b) ShowRequestId: *(bool)* — Wskazuje, czy RequestId powinno być wyświetlone (gdy nie jest null ani puste).

2) Movie.cs

CEL: Reprezentuje film w aplikacji.
POLA:

a) Id: *(int)* — Klucz główny.
b) Year: *(int)* — Rok produkcji filmu; walidacja: zakres od 1888 do 2100.
c) Title: *(string)* — Tytuł filmu; walidacja: minimalna długość 2, maksymalna 300 znaków.
d) Author: *(string)* — Reżyser filmu; walidacja: minimalna długość 2, maksymalna 300 znaków.
e) UserId: *(string?)* — ID użytkownika, który dodał film.
f) User: *(IdentityUser?)* — Powiązanie z użytkownikiem.

3) Preference.cs

CEL: Przechowuje preferencje użytkownika wobec filmów.
POLA:

a) Id: *(int)* — Klucz główny.
b) Type: *(string)* — Typ preferencji - możliwy spośród dostępnych w PreferenceType.cs
Sprawdzanie poprawności wyboru odbywa się na etapie tworzenia lub edycji preferencji
c) UserId: *(string?)* — ID użytkownika.
d) User: *(IdentityUser?)* — Powiązanie z użytkownikiem.
e) MovieId: *(int?)* — ID filmu.
f) Movie: *(Movie?)* — Powiązanie z filmem.

4) Review.cs

CEL: Reprezentuje recenzję filmu stworzoną przez użytkownika.
POLA:

a) Id: *(int)* — Klucz główny.
b) Value: *(int)* — Ocena filmu (1–5); walidacja: zakres od 1 do 5.
c) Comment: *(string?)* — Komentarz do recenzji; walidacja: długość od 3 do 1000 znaków.
d) CreatedAt: *(DateTime?)* — Data utworzenia recenzji.
e) UserId: *(string?)* — ID użytkownika tworzącego recenzję.

f) User: *(IdentityUser?)* — Powiązanie z użytkownikiem.
g) MovieId: *(int?)* — ID recenzowanego filmu.
h) Movie: *(Movie?)* — Powiązanie z filmem.

5. Kontrolery

1) ReviewsController

CEL → Zarządza recenzjami użytkowników

METODA	HTTP	PARAMETRY	OPIS	ZWRACANE DANE
Index	GET	Brak	wyświetla listę wszystkich recenzji, ładuje dane recenzji, filmów i użytkowników z bazy danych	Widok z listą recenzji (View)
Details	GET	int? id	Wyświetla szczegóły recenzji o podanym ID. Jeśli brak ID lub recenzji, zwraca NotFound.	Szczegóły recenzji (View)
My	GET	Brak	Wyświetla listę recenzji zalogowanego użytkownika.	Widok z listą recenzji użytkownika (View)
Create	GET	int? mov	Formularz tworzenia recenzji dla filmu o ID mov. Sprawdza, czy użytkownik obejrzał film i czy już go ocenił. Jeśli nie, zwraca błąd.	Formularz recenzji (View)
Create	POST	Review, review	Zapisuje nową recenzję w bazie danych. Sprawdza warunki: film istnieje, użytkownik obejrzał, nie ma już recenzji tego filmu.	Przekierowanie do My

Edit	GET	int? id	Formularz edycji recenzji o ID id, dostępny tylko dla autora recenzji. Jeśli recenzja lub film nie istnieją, zwraca błąd.	Formularz edycji (View)
Edit	POST	int id, Review r	Zapisuje zmiany w recenzji. Sprawdza właściciela i integralność danych. Jeśli recenzja nie istnieje, zwraca błąd.	Przekierowanie do My
Delete	GET	int? id	Wyświetla stronę potwierdzenia usunięcia recenzji. Sprawdza właściciela recenzji.	Widok potwierdzenia usunięcia (View)
DeleteConfirmed	POST	int id	Usuwa recenzję o ID id, jeśli użytkownik jest jej autorem.	Przekierowanie do My

2) **MoviesController**

CEL → Zarządza filmami w aplikacji

METODA	HTTP	PARAMETRY	OPIS	ZWRACANE DANE
Index	GET	brak	Wyświetla listę wszystkich filmów w aplikacji. Ładuje dane filmów z bazy danych i umożliwia filtrowanie i sortowanie.	Widok z listą filmów (View)
Details	GET	int? id	Wyświetla szczegóły filmu o podanym ID. Jeśli brak filmu, zwraca NotFound.	Szczegóły filmu (View)
Create	GET	brak	Formularz dodawania nowego filmu. Umożliwia wprowadzenie danych filmu, takich jak tytuł, rok i autor.	Formularz dodawania filmu (View)

Create	POST	Movie movie	Zapisuje nowy film w bazie danych, sprawdzając walidację danych. Jeśli wystąpią błędy, wyświetla formularz z komunikatami.	Przekierowanie do Index

3) PreferencesController

CEL → Zarządza preferencjami użytkowników względem filmów

METODA	HTTP	PARAMETRY	OPIS	ZWRACANE DANE
Index	GET	brak	Wyświetla preferencje użytkownika dla filmów (obejrzane filmy, ulubione itd.).	Widok z listą preferencji użytkownika (View)
Details	GET	int? id	Wyświetla szczegóły wybranej preferencji (powiązanej z konkretnym filmem).	Widok z informacjami o preferencji (View)
Edit	GET	int? id	Wyświetla formularz edycji preferencji dla danego filmu	Widok z formularzem edycji preferencji
Edit	POST	int id, Preference preference	Zapisuje zmiany w edytowanej preferencji	W przypadku sukcesu, przekierowanie do Index, w przeciwnym razie błąd
Delete	GET	int? id	Wyświetla potwierdzenie usunięcia preferencji	Widok potwierdzający usunięcie
Delete	POST	int id	Usuwa preferencję użytkownika o podanym ID	Przekierowanie do Index
Create	POST	Preference preference	Dodaję nową preferencję użytkownika (np. oznaczenie filmu jako “obejrzany”).	W przypadku sukcesu przekierowanie do Index, w

				przeciwnym razie błąd

4) HomeController

CEL → Obsługuje stronę główną aplikacji oraz strony błędów

METODA	HTTP	PARAMETRY	OPIS	ZWRACANE DANE
Index	GET	brak	Strona główna aplikacji. Wyświetla ogólne informacje o aplikacji oraz przydatne linki	Widok strony głównej (View)
Privacy	GET	brak	Strona z polityką prywatności	Widok polityki prywatności (View)
Error	GET	brak	Wyświetla szczegóły błędów aplikacji, takich jak RequestId oraz komunikaty o błędach	Widok błędów (View)

6. System Użytkowników

1) Role w systemie

W systemie funkcjonuje podział na dwie grupy użytkowników:

a) Zalogowani użytkownicy
b) Niezalogowani użytkownicy (goście)

Nie zdefiniowano bardziej szczegółowych ról w systemie, takich jak administrator czy moderator. Funkcjonalność zależy jedynie od stanu logowania użytkownika.

2) Funkcjonalności użytkowników

→ Zalogowani użytkownicy mogą:

a) Dodawać filmy do swojej listy, oznaczać je jako *„do obejrzenia”* lub *„obejrzane”*.
b) Dodawać recenzje do filmów, zawierające zarówno ocenę (w skali od 1 do 5), jak i treść tekstową.

c) Zarządzać swoimi listami filmów, tj. dodawać i usuwać filmy z tych list.

→ Niezalogowani użytkownicy mogą:

a) Przeglądać listę dostępnych filmów.
b) Wyświetlać szczegółowe informacje o filmach, w tym recenzje i średnie oceny.

3) Ograniczenia

a) Funkcje dodawania recenzji, ocen oraz zarządzania listami filmów są dostępne wyłącznie dla zalogowanych użytkowników.
b) Próba dostępu do tych funkcji przez niezalogowanych użytkowników przekierowuje ich na stronę logowania.

4) Powiązane informacje

a) Dane użytkownika są obsługiwane przez bibliotekę ASP.NET Identity, co zapewnia gotowe mechanizmy rejestracji, logowania i zarządzania tożsamością.
b) Każdy zalogowany użytkownik ma powiązaną listę filmów, które oznaczył jako *„do obejrzenia”* lub *„obejrzane”*.
c) Informacje o recenzjach i ocenach są również przypisane do konkretnego zalogowanego użytkownika, co pozwala na ich późniejszą modyfikację lub usunięcie.
d) Dane globalne, takie jak lista filmów, są dostępne dla wszystkich użytkowników, niezależnie od ich stanu logowania.

7) Krótka charakterystyka najciekawszych funkcjonalności

1) **Customowa strona z błędami**
W ramach aplikacji dodana została obsługa błędów w ramach spersonalizowanej strony błędu, wyświetlającej stosowny komunikat, informację o przyczynie niepowodzenia akcji oraz umożliwiającej powrót w odpowiednie miejsce w aplikacji.
Mimo możliwości zablokowania przeniesienia na tę stronę poprzez zastosowanie odpowiednich filtrowań, niektóre funkcjonalności zostały celowo na nią skierowane, w celu wykorzystania i zaobserwowania w prosty sposób.