

Dokumentacja projektu – MvcAppMovies (ASP.NET MVC)

Autor: Patryk Figas

Data: 02.01.2026

SPIS TREŚCI:

1. Informacje ogólne.....	3
2. Architektura aplikacji (MVC)	3
3. Struktura projektu.....	4
4. Model danych.....	4
5. Warstwa dostępu do danych – Entity Framework Core.....	5
6. Kontroler MoviesController.....	5
7. Widoki aplikacji	7
8. Routing.....	9
9. Baza danych.....	10
10. Funkcjonalności aplikacji	10
11. Wnioski końcowe.....	12
12. Aktualizacje i rozszerzenia projektu (Update 1.0)	13
12.1 Ulepszenia interfejsu użytkownika (UI).....	13
12.2 Ulepszenia tabeli z listą filmów	14
12.3 Modal potwierdzenia usuwania.....	14
12.4 Ulepszenia formularzy.....	15
13. Podsumowanie końcowe	17

1. Informacje ogólne

Nazwa projektu: MvcAppMovies

Technologie:

- C#
- ASP.NET Core MVC
- Entity Framework Core
- SQLite

Opis projektu:

Projekt MvcAppMovies to aplikacja webowa umożliwiająca zarządzanie listą filmów.

Użytkownik może dodawać nowe filmy, edytować istniejące, usuwać je oraz przeglądać szczegóły wybranego filmu.

Dane są zapisywane w relacyjnej bazie danych SQLite.

2. Architektura aplikacji (MVC)

Aplikacja została wykonana zgodnie ze wzorcem architektonicznym **MVC (Model–View–Controller)**.

Model

Odpowiada za reprezentację danych oraz logikę biznesową aplikacji.

View

Odpowiada za prezentację danych użytkownikowi (interfejs graficzny).

Controller

Pośredniczy pomiędzy modelem a widokiem, obsługuje żądania użytkownika.

3. Struktura projektu

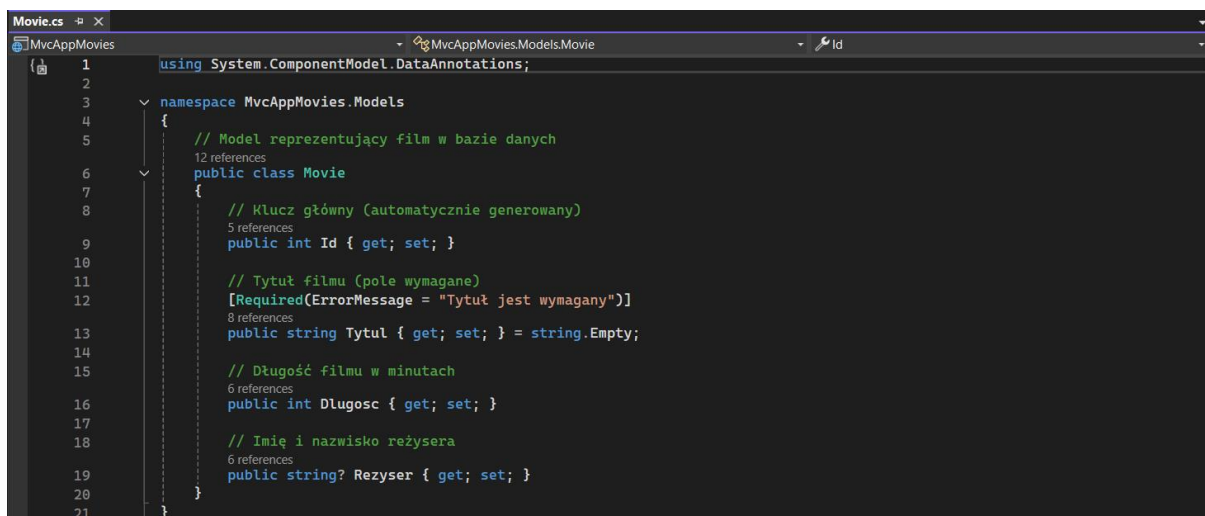
Projekt składa się z następujących katalogów:

- **Controllers** – zawiera kontrolery aplikacji
- **Models** – zawiera klasy modeli danych
- **Data** – zawiera konfigurację bazy danych
- **Views** – zawiera widoki Razor (.cshtml)
- **wwwroot** – pliki statyczne (CSS, JS, biblioteki)
- **Migrations** – migracje Entity Framework Core

4. Model danych

Klasa Movie

Model Movie reprezentuje pojedynczy film w bazie danych.



```
1 using System.ComponentModel.DataAnnotations;
2
3 namespace MvcAppMovies.Models
4 {
5     // Model reprezentujący film w bazie danych
6     public class Movie
7     {
8         // Klucz główny (automatycznie generowany)
9         public int Id { get; set; }
10
11         // Tytuł filmu (pole wymagane)
12         [Required(ErrorMessage = "Tytuł jest wymagany")]
13         public string Tytuł { get; set; } = string.Empty;
14
15         // Długość filmu w minutach
16         public int Dlugosc { get; set; }
17
18         // Imię i nazwisko reżysera
19         public string? Rezyser { get; set; }
20     }
21 }
```

Pola modelu:

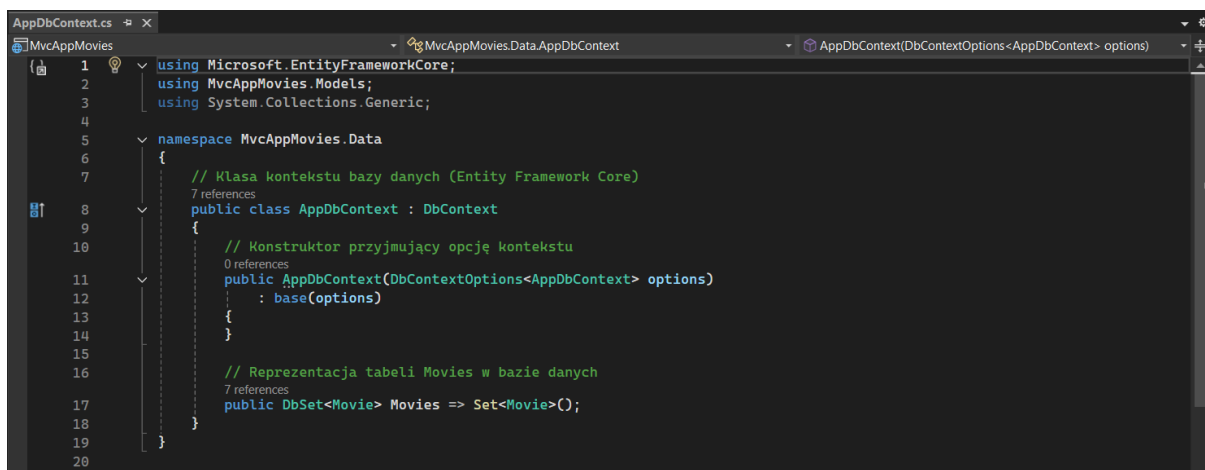
- Id – klucz główny (automatycznie generowany)
- Tytuł – tytuł filmu
- Dlugosc – długość filmu w minutach
- Rezyser – imię i nazwisko reżysera

Model wykorzystuje atrybuty walidacji, np. [Required], aby zapewnić poprawność danych wprowadzanych przez użytkownika.

5. Warstwa dostępu do danych – Entity Framework Core

Aplikacja korzysta z **Entity Framework Core** jako warstwy dostępu do danych.

AppDbContext



```

AppDbContext.cs
MvcAppMovies
1 using Microsoft.EntityFrameworkCore;
2 using MvcAppMovies.Models;
3 using System.Collections.Generic;
4
5 namespace MvcAppMovies.Data
6 {
7     // Klasa kontekstu bazy danych (Entity Framework Core)
8     public class AppDbContext : DbContext
9     {
10         // Konstruktor przyjmujący opcję kontekstu
11         public AppDbContext(DbContextOptions<AppDbContext> options)
12             : base(options)
13         {
14         }
15
16         // Reprezentacja tabeli Movies w bazie danych
17         public DbSet<Movie> Movies => Set<Movie>();
18     }
19
20

```

Klasa AppDbContext dziedziczy po DbContext i odpowiada za:

- konfigurację połączenia z bazą SQLite,
- mapowanie modelu Movie na tabelę w bazie danych.

Dane są przechowywane w pliku **movies.db**.

6. Kontroler MoviesController

Kontroler MoviesController odpowiada za obsługę operacji CRUD na filmach.

Dostępne akcje:

- Index – wyświetla listę filmów

```
18 // Wyświetlenie listy filmów
19 3 references
20 public async Task<IActionResult> Index()
21 {
22     return View(await _context.Movies.ToListAsync());
23 }
24
```

- Details – wyświetla szczegóły filmu

```
70 // Wyświetlenie szczegółów filmu
71 0 references
72 public async Task<IActionResult> Details(int id)
73 {
74     var movie = await _context.Movies.FindAsync(id);
75     if (movie == null)
76         return NotFound();
77     return View(movie);
78 }
79
```

- Add – dodawanie nowego filmu

```
24 // Formularz dodawania nowego filmu
25 0 references
26 public IActionResult Add() => View();
27
28 // Obsługa dodawania filmu
29 [HttpPost]
30 [ValidateAntiForgeryToken]
31 0 references
32 public async Task<IActionResult> Add(Movie movie)
33 {
34     if (!ModelState.IsValid) return View(movie);
35     _context.Movies.Add(movie);
36     await _context.SaveChangesAsync();
37     return RedirectToAction(nameof(Index));
38 }
```

- Edit – edycja istniejącego filmu

```
39 // Formularz edycji filmu
40 0 references
41 public async Task<IActionResult> Edit(int id)
42 {
43     var movie = await _context.Movies.FindAsync(id);
44     if (movie == null) return NotFound();
45     return View(movie);
46 }
47
48 // Obsługa edycji filmu
49 [HttpPost]
50 [ValidateAntiForgeryToken]
51 0 references
52 public async Task<IActionResult> Edit(Movie movie)
53 {
54     if (!ModelState.IsValid) return View(movie);
55     _context.Movies.Update(movie);
56     await _context.SaveChangesAsync();
57     return RedirectToAction(nameof(Index));
58 }
```

- Delete – usuwanie filmu

```

59
60 // Usunięcie filmu
0 references
61 public async Task<IActionResult> Delete(int id)
62 {
63     var movie = await _context.Movies.FindAsync(id);
64     if (movie == null) return NotFound();
65
66     _context.Movies.Remove(movie);
67     await _context.SaveChangesAsync();
68     return RedirectToAction(nameof(Index));
69 }

```

Kontroler korzysta bezpośrednio z kontekstu ApplicationDbContext.

7. Widoki aplikacji

Aplikacja zawiera następujące widoki:

- **Index.cshtml** – lista filmów w formie tabeli

```

Index.cshtml  x
MvcAppMovies
1  @*
2      Widok prezentujący listę filmów zapisanych w bazie danych.
3      Umożliwia przejście do szczegółów, edycji oraz usuwania filmu.
4      *@
5      @model List<MvcAppMovies.Models.Movie>
6
7      <h1>Lista filmów</h1>
8
9      <p>
10         <a asp-action="Add" class="btn btn-primary">Dodaj nowy film</a>
11     </p>
12
13     @{
14         int lp = 1;
15     }
16
17     <table class="table">
18     <tbody>
19
20         <tr style="font-weight:bold; color:#555; background:#f2f2f2;">
21             <td>Lp.</td>
22             <td>Tytuł</td>
23             <td style="text-align:center;">Czas (min)</td>
24             <td>Reżyser</td>
25             <td>Akcje</td>
26         </tr>
27
28         @foreach (var movie in Model)
29         {
30             <tr>
31                 <td>@lp</td>
32                 <td>@movie.Tytul</td>

```

```

Index.cshtml  X
MvcAppMovies
22      <td>Lp.</td>
23      <td>Tytuł</td>
24      <td style="text-align:center;">Czas (min)</td>
25      <td>Reżyser</td>
26      <td>Akcje</td>
27    </tr>
28
29    @foreach (var movie in Model)
30    {
31      <tr>
32        <td>@Lp</td>
33        <td>@movie.Tytuł</td>
34        <td style="text-align:center;">@movie.Dlugosc</td>
35        <td>@movie.Rezyser</td>
36        <td>
37          <a asp-action="Details" asp-route-id="@movie.Id">Szczegóły</a> |
38          <a asp-action="Edit" asp-route-id="@movie.Id">Edytuj</a> |
39          <a asp-action="Delete"
40            asp-route-id="@movie.Id"
41            onclick="return confirm('Na pewno usunąć ten film?');">
42            Usun
43          </a>
44        </td>
45      </tr>
46    }
47    Lp++;
48  }
49 </tbody>
50 </table>
51

```

- **Add.cshtml** – formularz dodawania filmu

```

Add.cshtml  X
MvcAppMovies
1  @model MvcAppMovies.Models.Movie
2
3  <h1>Dodaj film</h1>
4
5  <form asp-action="Add" method="post">
6    <div>
7      <label asp-for="Tytuł"></label><br />
8      <input asp-for="Tytuł" />
9      <span asp-validation-for="Tytuł"></span>
10    </div>
11
12    <div>
13      <label asp-for="Dlugosc"></label><br />
14      <input asp-for="Dlugosc" type="number" />
15    </div>
16
17    <div>
18      <label asp-for="Rezyser"></label><br />
19      <input asp-for="Rezyser" />
20    </div>
21
22    <button type="submit">Zapisz</button>
23  </form>
24

```

- **Edit.cshtml** – formularz edycji filmu

```
Edit.cshtml MvcAppMovies
1 @model MvcAppMovies.Models.Movie
2
3 <h1>Edytuj film</h1>
4
5 <form asp-action="Edit" method="post">
6     <input type="hidden" asp-for="Id" />
7
8     <div>
9         <label asp-for="Tytul"></label><br />
10        <input asp-for="Tytul" />
11        <span asp-validation-for="Tytul"></span>
12    </div>
13
14    <div>
15        <label asp-for="Dlugosc"></label><br />
16        <input asp-for="Dlugosc" type="number" />
17    </div>
18
19    <div>
20        <label asp-for="Rezyser"></label><br />
21        <input asp-for="Rezyser" />
22    </div>
23
24    <button type="submit">Zapisz zmiany</button>
25 </form>
26
```

- **Details.cshtml** – szczegóły wybranego filmu

```
Details.cshtml MvcAppMovies
1 @model MvcAppMovies.Models.Movie
2
3 <h1>Szczegóły filmu</h1>
4
5 <hr />
6
7 <h2>@Model.Tytul</h2>
8
9 <p>
10     <strong>Reżyser:</strong> @Model.Rezyser
11 </p>
12
13 <p>
14     <strong>Czas trwania:</strong> @Model.Dlugosc min.
15 </p>
16
17 <hr />
18
19 <p>
20     <a asp-action="Edit" asp-route-id="@Model.Id">Edytuj</a> |
21     <a asp-action="Index">Powrót do listy</a>
22 </p>
23
```

Widoki zostały wykonane przy użyciu Razor Pages oraz HTML.

8. Routing

Domyślna trasa aplikacji została skonfigurowana w pliku Program.cs:

```
{controller=Movies}/{action=Index}/{id?}
```

```

38
39
40 // Domyślna trasa: Movies/Index
41 app.MapControllerRoute(
42     name: "default",
43     pattern: "{controller=Movies}/{action=Index}/{id?}");
44
45 app.Run();
46
47
48

```

Oznacza to, że po uruchomieniu aplikacji użytkownik zostaje automatycznie przekierowany do listy filmów.

9. Baza danych

Aplikacja wykorzystuje **SQLite** jako relacyjną bazę danych. Dane są zapisywane trwale w pliku movies.db

Struktura bazy jest zarządzana za pomocą migracji EF Core.

10. Funkcjonalności aplikacji

Aplikacja umożliwia:

- dodawanie nowych filmów,

MvcAppMovies Home Privacy

Dodaj film

Tytuł
Pulp Fiction

Długość
154

Reżyser
Quentin Tarantino

Zapisz

© 2025 - MvcAppMovies - Privacy

- edytowanie istniejących filmów,

The screenshot shows the 'Edytuj film' form in the MvcAppMovies application. The form is titled 'Edytuj film' and contains the following fields:

- Tytuł: A text input field containing 'Gladiator'.
- Długość: A dropdown menu showing '155'.
- Reżyser: A text input field containing 'Ridley Scott'.

Below the fields is a green button labeled 'Zapisz zmiany' (Save changes). The application header shows 'MvcAppMovies', 'Home', and 'Privacy' links. The footer shows '© 2025 - MvcAppMovies - Privacy'.

- usuwanie filmów,

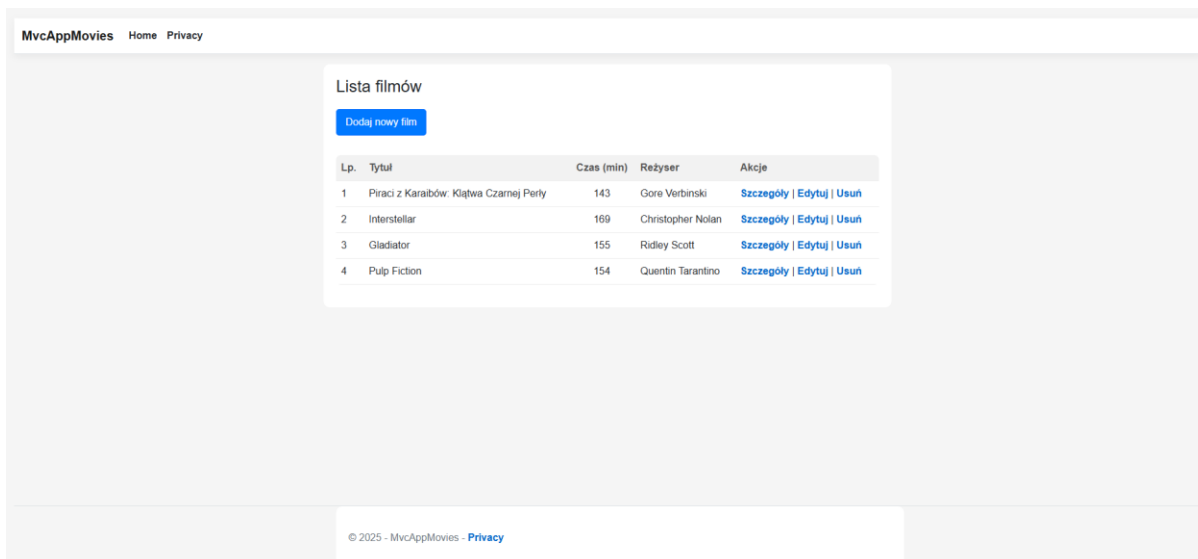
The screenshot shows the 'Lista filmów' (List of movies) page in the MvcAppMovies application. A confirmation dialog is displayed over the table, asking 'Komunikat ze strony localhost:7145' and 'Na pewno usunąć ten film?' (Are you sure you want to delete this movie?). The dialog has 'OK' and 'Anuluj' (Cancel) buttons.

The table below the dialog lists the movies:

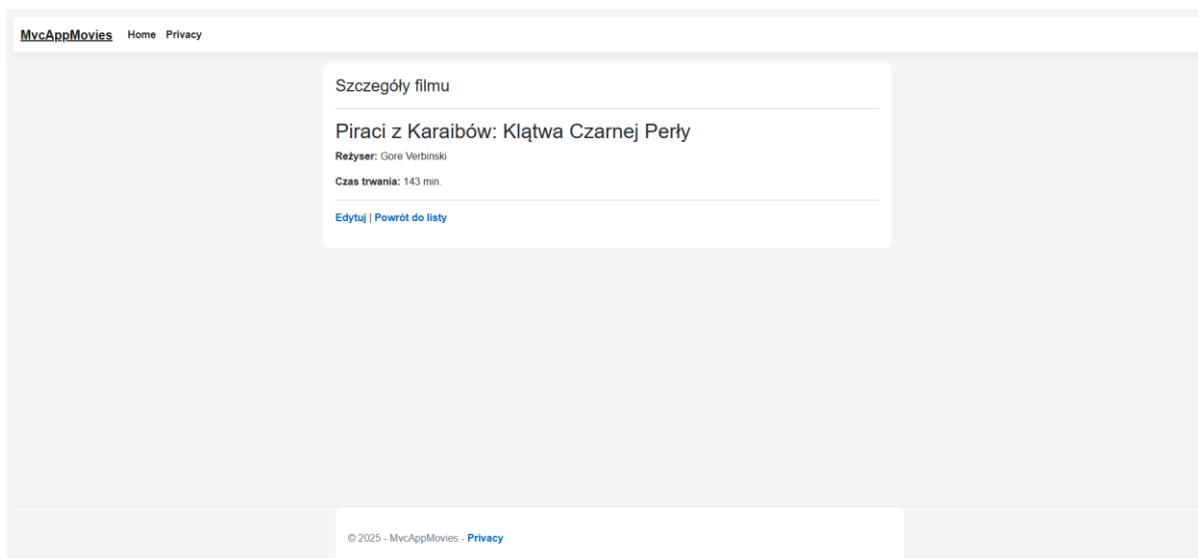
Lp.	Tytuł	Czas (min)	Reżyser	Akcje
1	Piraci z Karaibów: Kłajwa Czarnej Perły	143	Gore Verbinski	Szczegóły Edytuj Usuń
2	Interstellar	169	Christopher Nolan	Szczegóły Edytuj Usuń
3	Gladiator	155	Ridley Scott	Szczegóły Edytuj Usuń
4	Pulp Fiction	154	Quentin Tarantino	Szczegóły Edytuj Usuń

The application header shows 'MvcAppMovies', 'Home', and 'Privacy' links. The footer shows '© 2025 - MvcAppMovies - Privacy'.

- wyświetlanie listy filmów,



- podgląd szczegółów filmu.



11. Wnioski końcowe

Projekt MvcAppMovies został zrealizowany zgodnie z założeniami architektury MVC. Zastosowanie Entity Framework Core oraz bazy danych SQLite umożliwia trwałe przechowywanie danych oraz łatwe zarządzanie strukturą bazy.

12. Aktualizacje i rozszerzenia projektu (Update 1.0)

W trakcie realizacji projektu aplikacja **MvcAppMovies** została rozbudowana o dodatkowe elementy, które poprawiają estetykę interfejsu oraz komfort użytkowania.

Technologie:

- HTML5,
- CSS3,
- JavaScript,
- Bootstrap 5.

12.1 Ulepszenia interfejsu użytkownika (UI)

W projekcie wprowadzono własny motyw graficzny, który:

- wykorzystuje zmienne CSS do zarządzania kolorami,
- zapewnia spójny wygląd wszystkich widoków,
- poprawia czytelność treści oraz estetykę aplikacji.

Zastosowano m.in.:

- ciemne tło aplikacji,
- jasny kontrastowy tekst,
- wyraźne kolory akcji (dodawanie, edycja, usuwanie).

MvcAppMovies

Lista filmów [Dodaj nowy film](#)

Lp.	Tytuł	Czas (min)	Reżyser	Akcje
1	Piraci z Karaibów: Kłytwa Czarnej Perły	143	Gore Verbinski	Szczegóły Edytuj Usuń
2	Interstellar	169	Christopher Nolan	Szczegóły Edytuj Usuń
3	Gladiator	155	Ridley Scott	Szczegóły Edytuj Usuń
4	Pulp Fiction	154	Quentin Tarantino	Szczegóły Edytuj Usuń
5	Incepcja	148	Christopher Nolan	Szczegóły Edytuj Usuń
6	Skazani na Shawshank	142	Frank Darabont	Szczegóły Edytuj Usuń
7	Matka	136	Lana i Lilly Wachowski	Szczegóły Edytuj Usuń
8	Ojciec chrestny	175	Francis Ford Coppola	Szczegóły Edytuj Usuń
9	Fight Club	139	David Fincher	Szczegóły Edytuj Usuń
10	Forrest Gump	142	Robert Zemeckis	Szczegóły Edytuj Usuń
11	Zielona miła	189	Frank Darabont	Szczegóły Edytuj Usuń
12	Django	165	Quentin Tarantino	Szczegóły Edytuj Usuń
13	Leon zawodowiec	110	Luc Besson	Szczegóły Edytuj Usuń
14	Siedem	127	David Fincher	Szczegóły Edytuj Usuń
15	Avatar	162	James Cameron	Szczegóły Edytuj Usuń
16	Titanic	195	James Cameron	Szczegóły Edytuj Usuń
17	Joker	122	Todd Phillips	Szczegóły Edytuj Usuń
18	Władza Pierścieni: Drużyna Pierścienia	178	Peter Jackson	Szczegóły Edytuj Usuń
19	Władza Pierścieni: Dwie Wieże	179	Peter Jackson	Szczegóły Edytuj Usuń
20	Dark Knight	152	Christopher Nolan	Szczegóły Edytuj Usuń

12.2 Ulepszenia tabeli z listą filmów

Widok listy filmów (Index.cshtml) został rozbudowany o:

- tabelę z ustalonymi szerokościami kolumn (colgroup),
- numerację wierszy (Lp.),
- poprawione wyrównanie nagłówków i danych,
- efekt hover zwiększający czytelność,
- komunikat informujący o pustej bazie danych.

Dzięki temu tabela jest bardziej czytelna i wygląda profesjonalnie.

12.3 Modal potwierdzania usuwania

Proces usuwania filmu został zabezpieczony poprzez zastosowanie **modala Bootstrap**:

- użytkownik musi potwierdzić operację usunięcia,
- w modalu wyświetlany jest tytuł usuwanego filmu, co zapobiega przypadkowemu usunięciu danych.

Rozwiązanie to poprawia bezpieczeństwo i UX aplikacji.

Gladiator	155	Ridley Scott	Szczegóły	Edytuj	Usuń
Pulp Fiction	154	Quentin Tarantino	Szczegóły	Edytuj	Usuń
Incepcja	148	Christopher Nolan	Szczegóły	Edytuj	Usuń
Skazani na Shawshank			Szczegóły	Edytuj	Usuń
Matrix			Szczegóły	Edytuj	Usuń
Dzieci chrzestny			Szczegóły	Edytuj	Usuń
Fight Club			Szczegóły	Edytuj	Usuń
Forrest Gump			Szczegóły	Edytuj	Usuń
Zielona mila	189	Frank Darabont	Szczegóły	Edytuj	Usuń
Django	165	Quentin Tarantino	Szczegóły	Edytuj	Usuń
Leon zawodowiec	110	Luc Besson	Szczegóły	Edytuj	Usuń
Siedem	127	David Fincher	Szczegóły	Edytuj	Usuń

Potwierdzenie usunięcia

Czy na pewno chcesz usunąć ten film?

Tytuł: Piraci z Karaibów: Kłątwa Czarnej Perły

AnulujUsuń

12.4 Ulepszenia formularzy

Formularze dodawania i edycji filmu zostały dopracowane poprzez:

- walidację danych po stronie klienta i serwera,
- zabezpieczenie CSRF (AntiForgeryToken),
- ograniczenie szerokości formularzy dla lepszej ergonomii,
- spójny styl zgodny z motywem aplikacji.

- Dodawanie filmu

MvcAppMovies

Dodaj film

Tytuł

Czas (min)

Reżyser

Zapisz

Anuluj

© 2023 - MvcAppMovies

- Edycja filmu

MvcAppMovies

Edytuj film

Tytuł
Piraci z Karaibów: Klątwa Czarnej Perły

Czas (min)
143

Reżyser
Gore Verbinski

[Zapisz zmiany](#) [Anuluj](#)

© 2025 - MvcAppMovies

- Szczegóły

MvcAppMovies

Szczegóły filmu

[Edytuj](#) [Wróć do listy](#)

Piraci z Karaibów: Klątwa Czarnej Perły

Czas trwania 143 min

Reżyser Gore Verbinski

© 2025 - MvcAppMovies

Dodano również dostęp do edycji z poziomu okna szczegółów filmu.

13. Podsumowanie końcowe

Projekt **MvcAppMovies** został rozwinięty o dodatkowe funkcje oraz poprawiony został wygląd aplikacji.

Kod aplikacji został:

- uporządkowany zgodnie z architekturą MVC,
- opatrzony czytelnymi komentarzami,
- podzielony logicznie na komponenty (Model, View, Controller),
- przygotowany w sposób umożliwiający dalszą rozbudowę projektu.