

# Dokumentacja projektu – MvcAppMovies (ASP.NET MVC)

Autor: Patryk Figas

Data: 02.01.2026

## SPIS TREŚCI:

1. Informacje ogólne.....	2
2. Architektura aplikacji (MVC) .....	3
3. Struktura projektu.....	3
4. Model danych.....	3
5. Warstwa dostępu do danych – Entity Framework Core.....	4
6. Kontroler MoviesController.....	5
7. Widoki aplikacji .....	6
8. Routing.....	9
9. Baza danych.....	9
10. Funkcjonalności aplikacji .....	9
11. Wnioski końcowe.....	12

# 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

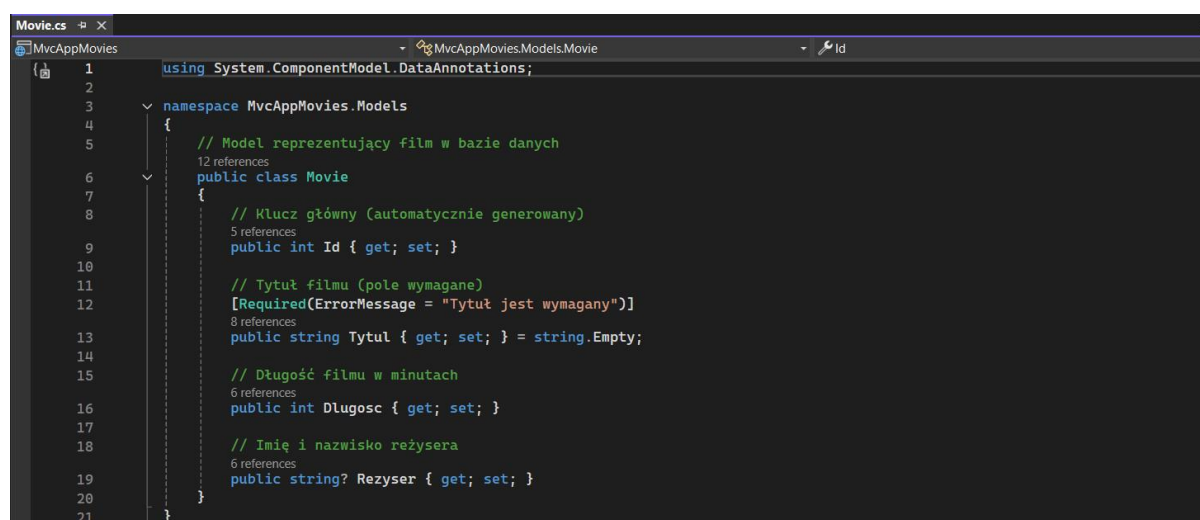
Projekt składa się z następujących katalogów:

- **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 Tytul { 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:

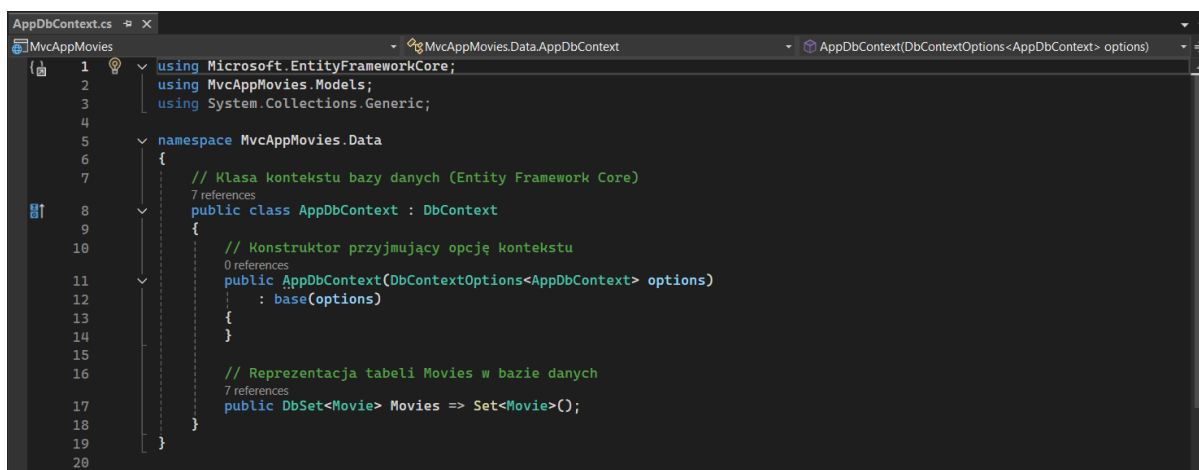
- Id – klucz główny (automatycznie generowany)
- Tytul – tytuł filmu
- Dlugosc – długość filmu w minutach
- Rezyser – imię i nazwisko reżysera

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



```
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.

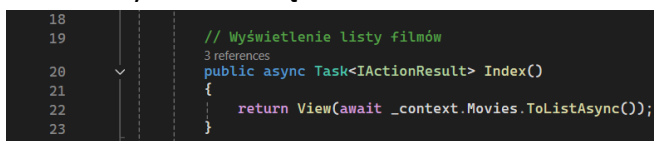
Dane są przechowywane w pliku **movies.db**.

## 6. Kontroler MoviesController

Kontroler MoviesController odpowiada za obsługę operacji CRUD na filmach.

Dostępne akcje:

- Index – wyświetla listę filmów



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

- Details – wyświetla szczegóły filmu

```

70
71 // Wyświetlenie szczegółów filmu
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
78     return View(movie);
79 }

```

- Add – dodawanie nowego filmu

```

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

```

- Edit – edycja istniejącego filmu

```

39
40 // Formularz edycji filmu
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 public async Task<IActionResult> Edit(Movie movie)
52 {
53     if (!ModelState.IsValid) return View(movie);
54
55     _context.Movies.Update(movie);
56     await _context.SaveChangesAsync();
57     return RedirectToAction(nameof(Index));
58 }

```

- Delete – usuwanie filmu

```

59
60 // Usunięcie filmu
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 }

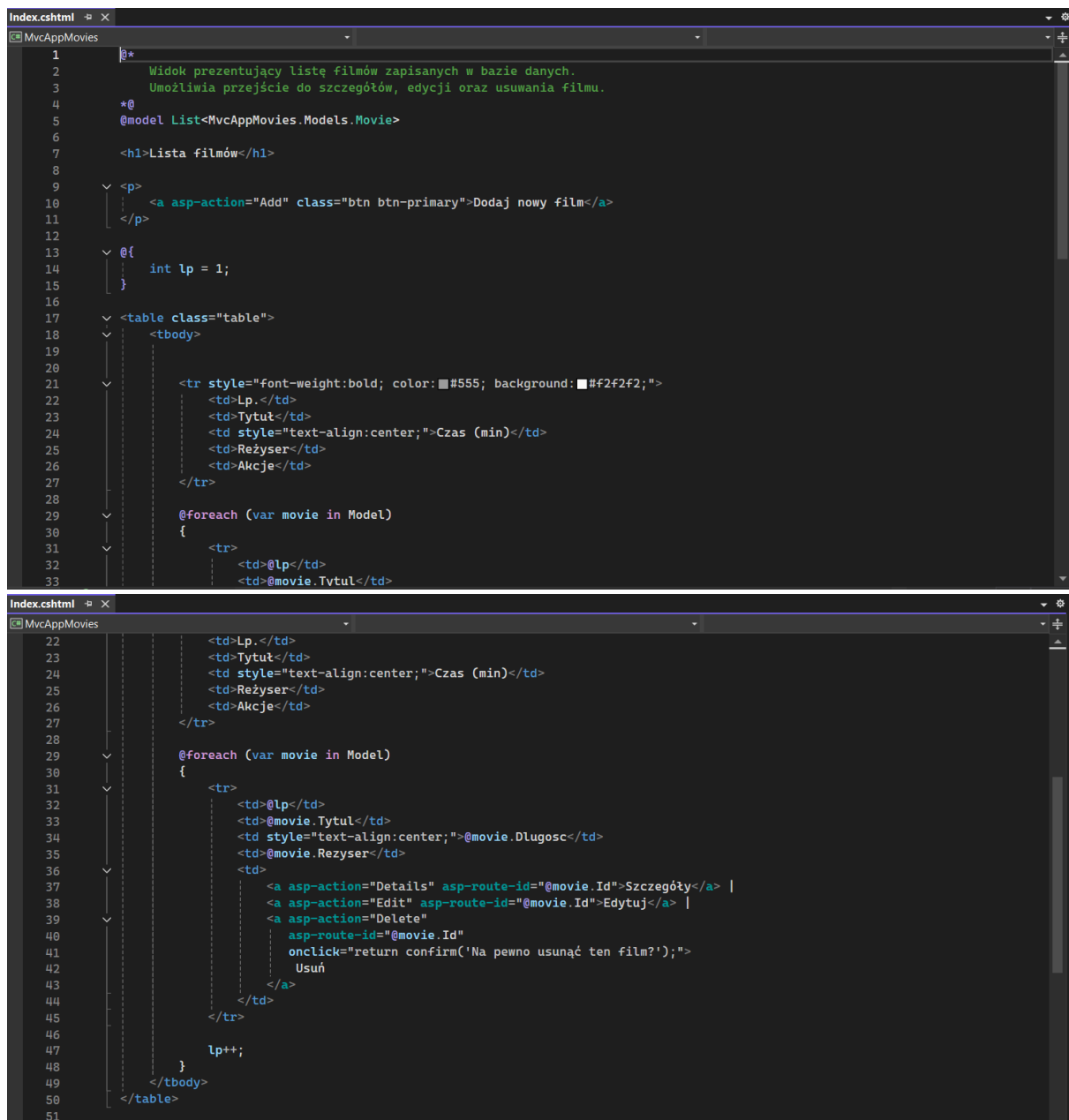
```

Kontroler korzysta bezpośrednio z kontekstu ApplicationDbContext.

## 7. Widoki aplikacji

Aplikacja zawiera następujące widoki:

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



```
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>
33
34                  <td>Lp.</td>
35                  <td>Tytuł</td>
36                  <td style="text-align:center;">Czas (min)</td>
37                  <td>Reżyser</td>
38                  <td>Akcje</td>
39              </tr>
40
41              @foreach (var movie in Model)
42              {
43                  <tr>
44                      <td>@lp</td>
45                      <td>@movie.Tytul</td>
46                      <td style="text-align:center;">@movie.Dlugosc</td>
47                      <td>@movie.Rezyser</td>
48                      <td>
49                          <a asp-action="Details" asp-route-id="@movie.Id">Szczegóły</a> |
50                          <a asp-action="Edit" asp-route-id="@movie.Id">Edytuj</a> |
51                          <a asp-action="Delete"
52                             asp-route-id="@movie.Id"
53                             onclick="return confirm('Na pewno usunąć ten film?');">
54                              Usun
55                          </a>
56                      </td>
57                  </tr>
58              }
59              lp++;
60          }
61      </tbody>
62  </table>
```

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

```
Add.cshtml
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="Tytul"></label><br />
8         <input asp-for="Tytul" />
9         <span asp-validation-for="Tytul"></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
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.Reżyser
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

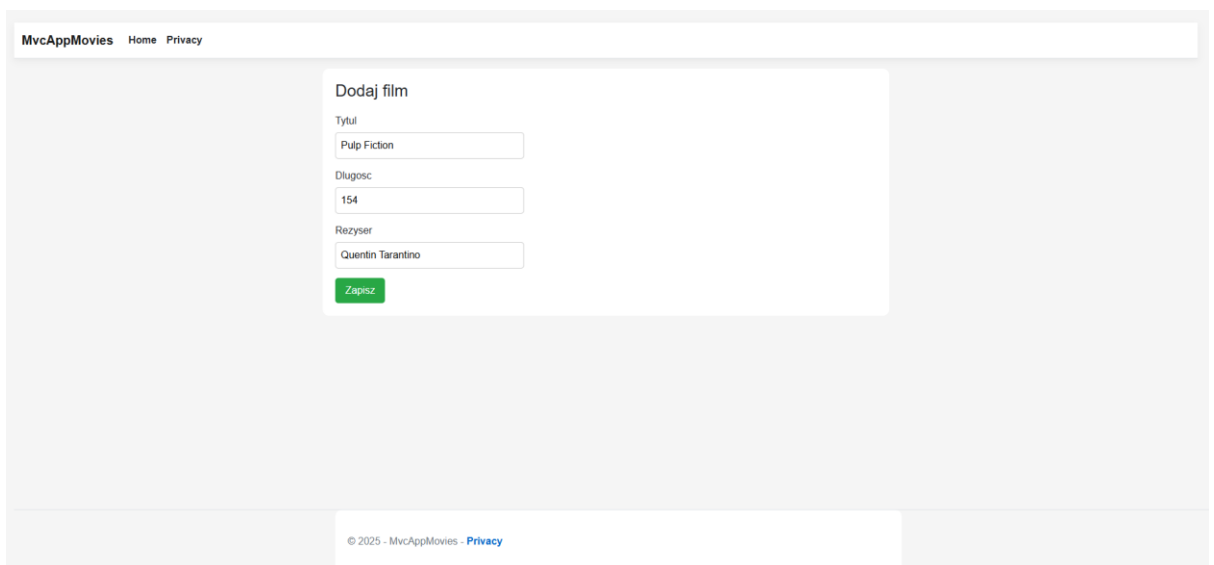
Aplikacja wykorzystuje **SQLite** jako relacyjną bazę danych. Dane są zapisywane trwale w pliku movies.db

Struktura bazy jest zarządzana za pomocą migracji EF Core.

## 10. Funkcjonalności aplikacji

Aplikacja umożliwia:

- dodawanie nowych filmów,



The screenshot shows a web application interface for adding a new movie. At the top, there is a navigation bar with the text 'MvcAppMovies' and links for 'Home' and 'Privacy'. The main content area features a form titled 'Dodaj film'. The form contains three input fields: 'Tytuł' (Title) with the value 'Pulp Fiction', 'Długość' (Length) with the value '154', and 'Reżyser' (Director) with the value 'Quentin Tarantino'. Below these fields is a green button labeled 'Zapisz' (Save). At the bottom of the page, there is a footer with the text '© 2025 - MvcAppMovies - [Privacy](#)'.

- edytowanie istniejących filmów,

MvcAppMovies Home Privacy

### Edytuj film

Tytuł  
Gladiator

Długość  
155

Reżyser  
Ridley Scott

Zapisz zmiany

© 2025 - MvcAppMovies - Privacy

- usuwanie filmów,

MvcAppMovies Home Privacy

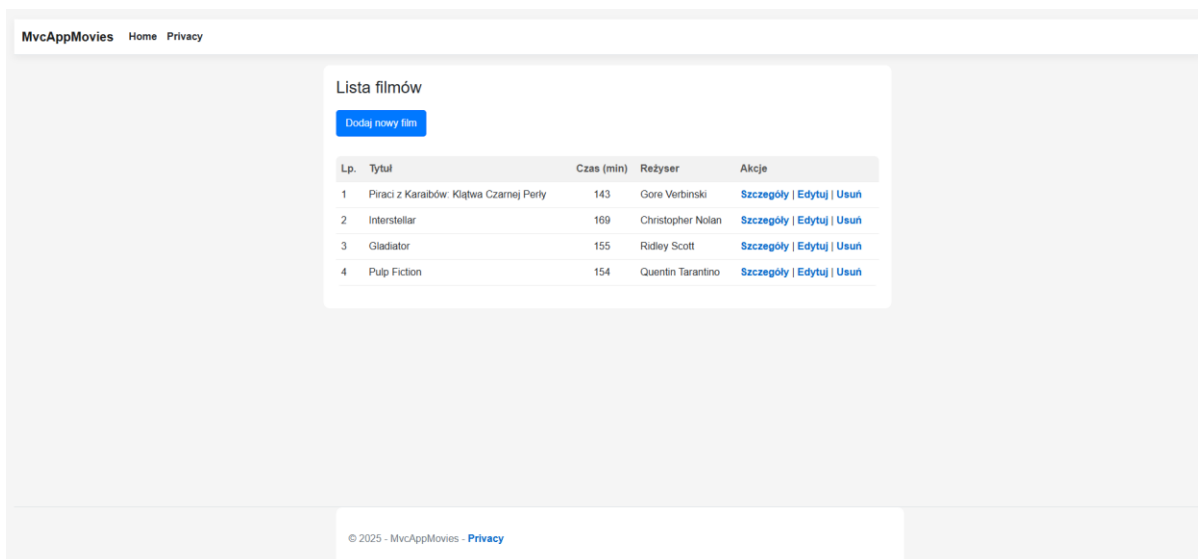
### Lista filmów

Dodaj nowy film

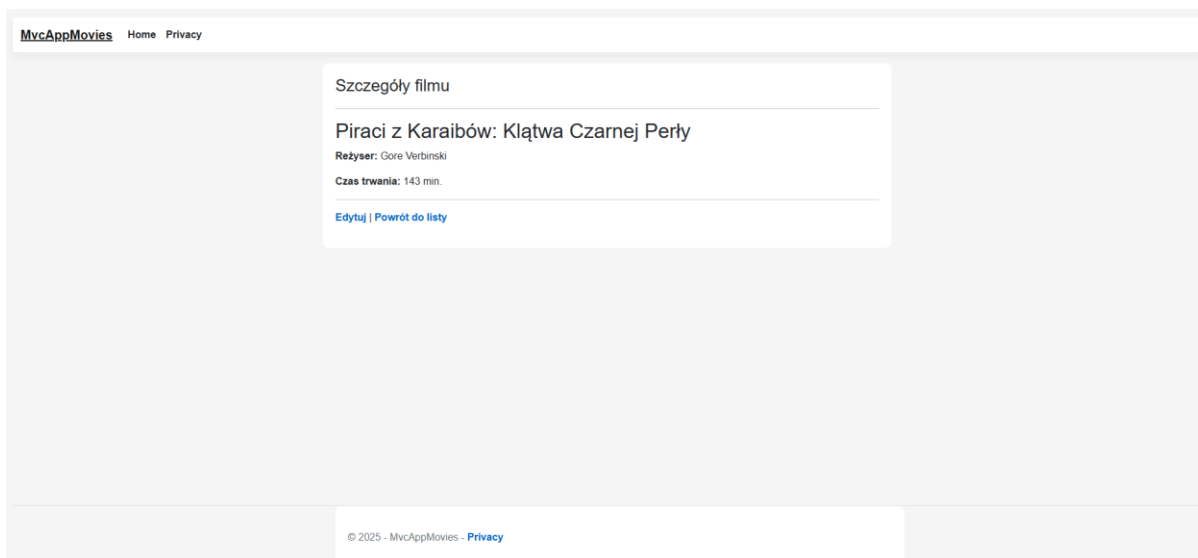
Lp.	Tytuł	Czas (min)	Reżyser	Akcje
1	Piraci z Karaibów: Kłajwa Czarnej Perły	143	Gore Verbinski	<a href="#">Szczegóły</a>   <a href="#">Edytuj</a>   <a href="#">Usun</a>
2	Interstellar	169	Christopher Nolan	<a href="#">Szczegóły</a>   <a href="#">Edytuj</a>   <a href="#">Usun</a>
3	Gladiator	155	Ridley Scott	<a href="#">Szczegóły</a>   <a href="#">Edytuj</a>   <a href="#">Usun</a>
4	Pulp Fiction	154	Quentin Tarantino	<a href="#">Szczegóły</a>   <a href="#">Edytuj</a>   <a href="#">Usun</a>

© 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.