

---

# SPRAWOZDANIE PLATFORMY PROGRAMISTYCZNE .NET I JAVA – LAB4

---



---

POLITECHNIKA WROCŁAWSKA

WYDZIAŁ INFORMATYKI I TELEKOMUNIKACJI

INFORMATYCZNE SYSTEMY AUTOMATYKI

---

MICHAŁ WYRZYKOWSKI

INDEKS 264228

# 1. Kod C# oraz aplikacja okienkowa

lab4

Home

Counter

Weather

About

## Weather

This component demonstrates showing data.

Date	Temp. (C)	Temp. (F)	Summary
22.05.2024	42	107	Cool

Number of warm days (above 15°C): 5

Filter Warm Days

Restore

cool!

lab4\_2

Home

Counter

Weather

Auth Required

Movies

About

## Index

[Create New](#)

Title	ReleaseDate	Rate	
Piraci z karaibów	5/10/2024 12:00:00 AM	9	<a href="#">Details</a>   <a href="#">Edit</a>   <a href="#">Delete</a>
Nemo	1/12/2023 12:00:00 AM	8	<a href="#">Details</a>   <a href="#">Edit</a>   <a href="#">Delete</a>
Dzień świra	7/27/2002 12:00:00 AM	10	<a href="#">Details</a>   <a href="#">Edit</a>   <a href="#">Delete</a>
John Wick 2	5/17/2024 12:00:00 AM	5.6666665	<a href="#">Details</a>   <a href="#">Edit</a>   <a href="#">Delete</a>
Film nad filmami	5/17/2024 12:00:00 AM	3.3333333	<a href="#">Details</a>   <a href="#">Edit</a>   <a href="#">Delete</a>

user@gmail.com

Logout

lab4\_2

About

Home

Counter

Weather

Auth Required


Movies

user@gmail.com

Logout

# Movie Details

## Movie

Title	John Wick 2
Description	Strzelanie
Release Date	5/17/2024 12:00:00 AM
Rate	5.6666665
Image	

### Rate this Movie

New Rate (1-10)

Submit Rating

### Upload Movie Image

Wybierz plik

Nie wybrano pliku

Upload

[Edit](#) | [Back to List](#)

```
45  @code {
46      private WeatherForecast[]? forecasts;
47      private WeatherForecast[]? originalForecasts;
48      private int warmDays = 0;
49
50      protected override async Task OnInitializedAsync()
51      {
52          await Task.Delay(500);
53
54          var startDate = DateOnly.FromDateTime(DateTime.Now);
55          var summaries = new[] { "Freezing", "Bracing", "Chilly", "Cool", "Mild", "Warm", "Balmy", "Hot", "Sweltering", "Scorching" };
56          originalForecasts = Enumerable.Range(1, 10).Select(index => new WeatherForecast
57              {
58                  Date = startDate.AddDays(index),
59                  TemperatureC = Random.Shared.Next(-20, 55),
60                  Summary = summaries[Random.Shared.Next(summaries.Length)]
61              }).ToArray();
62
63          forecasts = originalForecasts.ToArray();
64
65          warmDays = forecasts.Count(forecast => forecast.TemperatureC > 15);
66      }
67
68      private void FilterWarmDays()
69      {
70          forecasts = originalForecasts.Where(forecast => forecast.TemperatureC > 15).ToArray();
71      }
72
73      private void RestoreForecasts()
74      {
75          forecasts = originalForecasts.ToArray();
76      }
77
78      private void Input(ChangeEventArgs arg)
79      {
80          var filterText = arg.Value.ToString();
81          forecasts = originalForecasts
82              .Where(forecast => forecast.Summary.Contains(filterText, StringComparison.OrdinalIgnoreCase))
83              .ToArray();
84      }
85
86      private class WeatherForecast
87      {
88          public DateOnly Date { get; set; }
89          public int TemperatureC { get; set; }
90          public string? Summary { get; set; }
91          public int TemperatureF => 32 + (int)(TemperatureC / 0.5556);
92      }
93  }
```

```

1  @page "/movies"
2  @using Microsoft.AspNetCore.Components.QuickGrid
3  @inject Lab4_2.Data.ApplicationDbContext DB
4  @using System.Linq
5  @using Microsoft.EntityFrameworkCore
6  @rendermode InteractiveServer
7
8  <PageTitle>Index</PageTitle>
9
10 <h1>Index</h1>
11
12 <p>
13     <a href="/movies/create">Create New</a>
14 </p>
15
16 <QuickGrid Class="table" Items="movie" SortBy="@defaultSortProperty" SortDirection="defaultSortDirection">
17     <PropertyColumn Property="movie => movie.Title" Sortable="true" />
18     <PropertyColumn Property="movie => movie.ReleaseDate" Sortable="true" />
19     <PropertyColumn Property="movie => movie.Rate" Sortable="true" />
20
21     <TemplateColumn Context="movie">
22         <a href="@($"movies/details?id={movie.Id}")">Details</a> |
23         <a href="@($"movies/edit?id={movie.Id}")">Edit</a> |
24         <a href="@($"movies/delete?id={movie.Id}")">Delete</a>
25     </TemplateColumn>
26 </QuickGrid>
27
28 @code {
29     private IQueryable<Movie>? movie;
30     private Func<Movie, object>? defaultSortProperty = g => g.Title;
31     private SortDirection defaultSortDirection = SortDirection.Ascending;
32     protected override async Task OnInitializedAsync()
33     {
34
35         movie = DB.Movies.AsQueryable();
36     }
37 }

```

```

1  using System.ComponentModel.DataAnnotations;
2
3  namespace lab4_2.Components
4  {
5      public class Movie
6      {
7          Odwołania: 10
8          public int Id { get; set; }
9          Odwołania: 15
10         public string? Title { get; set; }
11         Odwołania: 12
12         public string? Description { get; set; }
13         Odwołania: 13
14         [DataType(DataType.Date)]
15         public DateTime? ReleaseDate { get; set; }
16         Odwołania: 17
17         public float? Rate { get; set; }
18         Odwołania: 3
19         public string? ImagePath { get; set; }
20     }
21 }

```

```

1  @page "/movies/details"
2  @inject lab4_2.Data.ApplicationDbContext DB
3  @inject NavigationManager NavigationManager
4  @using Microsoft.EntityFrameworkCore
5  @using Microsoft.AspNetCore.Components.Forms
6  @rendermode InteractiveServer
7
8  <PageTitle>Details</PageTitle>
9
10 <h1>Movie Details</h1>
11
12 <div>
13     <h4>Movie</h4>
14     <hr />
15     @if (movie is null)
16     {
17         <p><em>Loading...</em></p>
18     }
19     else
20     {
21         <dl class="row">
22             <dt class="col-sm-2">Title</dt>
23             <dd class="col-sm-10">@movie.Title</dd>
24             <dt class="col-sm-2">Description</dt>
25             <dd class="col-sm-10">@movie.Description</dd>
26             <dt class="col-sm-2">Release Date</dt>
27             <dd class="col-sm-10">@movie.ReleaseDate</dd>
28             <dt class="col-sm-2">Rate</dt>
29             <dd class="col-sm-10">@movie.Rate</dd>
30             @if (!string.IsNullOrEmpty(movie.ImagePath))
31             {
32                 <dt class="col-sm-2">Image</dt>
33                 <dd class="col-sm-10"></dd>
34             }
35         </dl>
36
37         <h4>Rate this Movie</h4>
38         <EditForm Model="@newRate" OnValidSubmit="@SubmitRating" FormName="AddMovieRating">
39             <DataAnnotationsValidator />
40             <ValidationSummary />
41             <div class="form-group">
42                 <label for="newRate">New Rate (1-10)</label>
43                 <InputNumber id="newRate" class="form-control" @bind-Value="@newRate" min="1" max="10" />
44             </div>
45             <button type="submit" class="btn btn-primary">Submit Rating</button>
46         </EditForm>
47
48         <h4>Upload Movie Image</h4>
49         <EditForm EditContext="@editContext" OnValidSubmit="@UploadImage" FormName="UploadImage">
50             <DataAnnotationsValidator />
51             <ValidationSummary />
52             <div class="form-group">
53                 <InputFile OnChange="OnFileChange" />
54             </div>
55             <button type="submit" class="btn btn-primary">Upload</button>
56         </EditForm>
57     }

```

```

57
58
59     <a href="@($"/movies/edit?id={movie.Id})">Edit</a> |
60     <a href="@($"/movies)">Back to List</a>
61 </div>
62 }
63 </div>
64
65 @code {
66     Movie? movie;
67     IBrowserFile? selectedFile;
68     EditContext editContext;
69     [SupplyParameterFromQuery]
70     public int Id { get; set; }
71     int newRate;
72     int srednia = 1;
73     protected override async Task OnInitializedAsync()
74     {
75         movie = await DB.Movies.FirstOrDefaultAsync(m => m.Id == Id);
76
77         if (movie is null)
78         {
79             NavigationManager.NavigateTo("notfound");
80         }
81         editContext = new EditContext(new object());
82     }
83
84     private async Task SubmitRating()
85     {
86         if (movie is not null)
87         {
88             if (movie.Rate.HasValue)
89             {
90                 movie.Rate = ((movie.Rate.Value * srednia) + newRate) / (1 + srednia);
91                 srednia++;
92             }
93             else
94             {
95                 movie.Rate = newRate;
96             }
97
98             await DB.SaveChangesAsync();
99             await OnInitializedAsync();
100         }
101     }
102
103     private void OnFileChange(InputFileChangeEventArgs e)
104     {
105         selectedFile = e.File;
106     }
107

```

```

103     private void OnFileChange(InputFileChangeEventArgs e)
104     {
105         selectedFile = e.File;
106     }
107
108     private async Task UploadImage()
109     {
110         if (selectedFile is not null)
111         {
112             var uploadsFolder = Path.Combine("wwwroot", "uploads");
113             if (!Directory.Exists(uploadsFolder))
114             {
115                 Directory.CreateDirectory(uploadsFolder);
116             }
117
118             var filePath = Path.Combine(uploadsFolder, selectedFile.Name);
119             using (var stream = new FileStream(filePath, FileMode.Create))
120             {
121                 await selectedFile.OpenReadStream().CopyToAsync(stream);
122             }
123
124             if (movie is not null)
125             {
126
127                 movie.ImagePath = $"/uploads/{selectedFile.Name}";
128                 await DB.SaveChangesAsync();
129                 await OnInitializedAsync();
130             }
131         }
132     }
133

```

## 2. Opis działania

W pierwszej części zadania jedyną zmianą jaką wprowadzono to opcja filtrowania danych. Dostępne są dwa przyciski i pole tekstowe do filtrowania danych. Pierwszy przycisk "Filter Warm Days" filtruje prognozy, pozostawiając tylko te dni, gdzie temperatura jest powyżej 15°C. Drugi przycisk "Restore" przywraca oryginalną listę prognoz. Pole tekstowe umożliwia filtrowanie prognoz na podstawie wprowadzonego przez użytkownika tekstu opisu pogody.

W drugiej części zadania dodane zostały nowe podstrony, te w których zaszły zmiany podczas implementacji swojego pomysłu na stronę internetową to `details.razor` oraz `index.razor`.

### Strona główna (`index.razor`):

- Na stronie głównej wyświetlana jest lista filmów w formie tabeli. Użytkownik może wybierać filmy do oglądania szczegółów, edycji lub usunięcia.
- `OnInitializedAsync()`: W tej metodzie wczytywana jest lista wszystkich filmów z bazy danych.
- Wykorzystano `QuickGrid` do wygodnego wyświetlania listy filmów w formie tabeli. Umożliwia on sortowanie kolumn oraz akcje takie jak edycja, usuwanie i przeglądanie szczegółów filmu.

### Strona szczegółów filmu (`details.razor`):

- Na tej stronie wyświetlane są szczegóły wybranego filmu. Głównie skupia się na wyświetlaniu informacji o filmie, takich jak tytuł, opis, data premiery, ocena oraz obrazek (jeśli istnieje). Oprócz tego użytkownik ma możliwość dodania nowej oceny filmu oraz przesłania nowego obrazka.
- `OnInitializedAsync()`: Ta metoda jest wywoływana przy inicjalizacji komponentu. Wczytuje ona szczegóły wybranego filmu na podstawie identyfikatora (Id) przekazanego przez adres URL.
- `SubmitRating()`: Obsługuje dodawanie nowej oceny filmu. Jeśli film już posiada ocenę, oblicza nową średnią z aktualną oceną oraz nową oceną użytkownika.
- `UploadImage()`: Obsługuje przesyłanie nowego obrazka. Pobiera wybrany plik obrazka, zapisuje go w katalogu `wwwroot/uploads`, aktualizuje ścieżkę obrazka w rekordzie filmu w bazie danych i odświeża widok.

### Połączenie z bazą danych:

Aplikacja wykorzystuje `Entity Framework Core`, aby łączyć się z bazą danych. Za pomocą klasy `ApplicationDbContext` komunikuje się z bazą danych, a dokładniej z tabelą `Movies`. Metody `OnInitializedAsync()` pobierają odpowiednie rekordy z bazy danych, a metody `SaveChangesAsync()` zapisują zmiany (takie jak dodanie oceny lub zmiana ścieżki obrazka) do bazy danych.

### Publikacja na Azure:

Strona została również opublikowana na Azure. Publikacja na platformie Azure umożliwia łatwe udostępnienie aplikacji w Internecie, co pozwala użytkownikom na korzystanie z jej funkcji z dowolnego miejsca i urządzenia. Dodatkowo, korzystając z usług Azure, można skorzystać z zaawansowanych funkcji, takich jak skalowanie automatyczne, zabezpieczenia i zapasowe kopie danych.