

SPRAWOZDANIE PLATFORMY PROGRAMISTYCZNE .NET I JAVA – LAB4



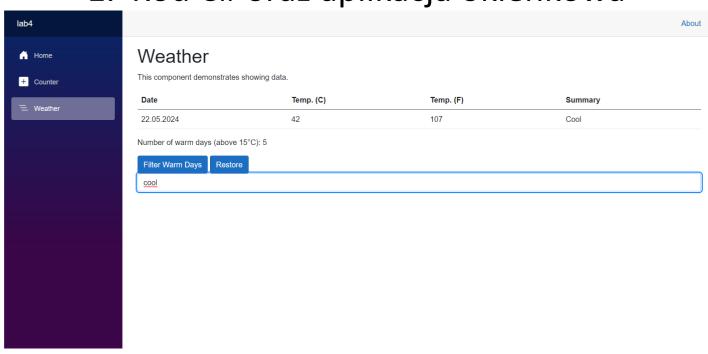
POLITECHNIKA WROCŁAWSKA

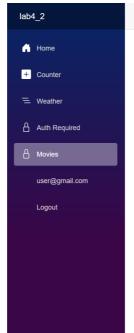
WYDZIAŁ INFORMATYKI I TELEKOMUNIKACJI

INFORMATYCZNE SYSTEMY AUTOMATYKI

MICHAŁ WYRZYKOWSKI INDEKS 264228

1. Kod C# oraz aplikacja okienkowa

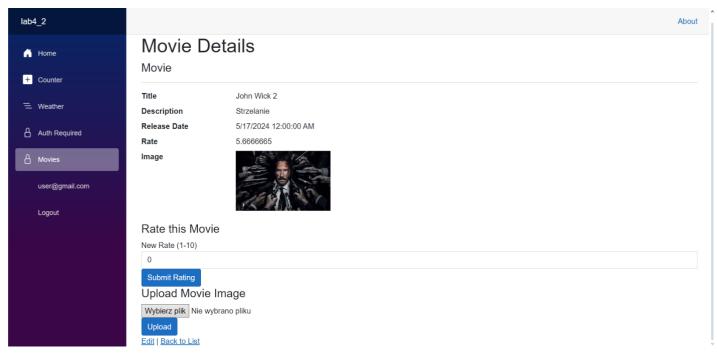




Index

Create New

Title	RelaseDate	Rate	
Piraci z karaibów	5/10/2024 12:00:00 AM	9	Details Edit Delete
Nemo	1/12/2023 12:00:00 AM	8	Details Edit Delete
Dzień świra	7/27/2002 12:00:00 AM	10	Details Edit Delete
John Wick 2	5/17/2024 12:00:00 AM	5.6666665	Details Edit Delete
Film nad filmami	5/17/2024 12:00:00 AM	3.3333333	Details Edit Delete



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

```
@page "/movies"
       Qusing Microsoft.AspNetCore.Components.QuickGrid
       @inject lab4_2.Data.ApplicationDbContext DB
      ⊘@using System.Ling
      Qusing Microsoft.EntityFrameworkCore
       @rendermode InteractiveServer
       <PageTitle>Index</PageTitle>
       <hl>Index</hl>
           <a href="/movies/create">Create New</a>
      v<QuickGrid Class="table" Items="movie" SortBy="@defaultSortProperty" SortDirection="defaultSortDirection">
           <PropertyColumn Property="movie => movie.Title" Sortable="true" />
           <PropertyColumn Property="movie => movie.RelaseDate" Sortable="true" />
           <PropertyColumn Property="movie => movie.Rate" Sortable="true" />
           <TemplateColumn Context="movie">
22
23
24
                <a href="@($"movies/details?id={movie.Id}")">Details</a> |
               <a href="@($"movies/edit?id={movie.Id}")">Edit</a> |
<a href="@($"movies/delete?id={movie.Id}")">Delete</a>
26
27
28
      </QuickGrid>
      √@code {
           private IQueryable<Movie>? movie;
           private Func<Movie, object>? defaultSortProperty = g => g.Title;
30
31
           private SortDirection defaultSortDirection = SortDirection.Ascending;
           protected override async Task OnInitializedAsync()
                movie = DB.Movies.AsQueryable();
```

```
using System.ComponentModel.DataAnnotations;

namespace lab4_2.Components

public class Movie

public int Id { get; set; }

Odwolania: 15

public string? Title { get; set; }

Odwolania: 12

public string? Description { get; set; }

[DataType(DataType.Date)]

Odwolania: 13

public DateTime? RelaseDate { get; set; }

Odwolania: 17

public float? Rate { get; set; }

Odwolania: 3

public string? ImagePath { get; set; }

It |

It |
```

```
@inject lab4_2.Data.ApplicationDbContext DB
  @inject NavigationManager NavigationManager
  ∨@using Microsoft.EntityFrameworkCore
Qusing Microsoft.AspNetCore.Components.Forms
Prendermode InteractiveServer
  <PageTitle>Details</PageTitle>
  <h1>Movie Details</h1>
      <h4>Movie</h4>
      @if (movie is null)
           <em>Loading...</em>
      else
           <dl class="row">
               <dt class="col-sm-2">Title</dt>
               <dd class="col-sm-10">@movie.Title</dd>
               <dt class="col-sm-2">Description</dt>
               <dd class="col-sm-10">@movie.Description</dd>
               <dt class="col-sm-2">Release Date</dt>
               <dd class="col-sm-10">@movie.RelaseDate</dd>
               <dt class="col-sm-2">Rate</dt
               <dd class="col-sm-10">@movie.Rate</dd>
              @if (!string.IsNullOrEmpty(movie.ImagePath))
                   <dt class="col-sm-2">Image</dt>
                   <dd class="col-sm-10"><img src="@movie.ImagePath" alt="@movie.Title Image" style="max-width:8θθpx;" /></dd>
           <h4>Rate this Movie</h4>
           <EditForm Model="@newRate" OnValidSubmit="@SubmitRating" FormName="AddMovieRating">
               <div class="form-group"
                  <label for="newRate">New Rate (1-10)</label>
                   <InputNumber id="newRate" class="form-control" @bind-Value="@newRate" min="1" max="10" />
               <button type="submit" class="btn btn-primary">Submit Rating</button>
           <h4>Upload Movie Image</h4>
           <EditForm EditContext="@editContext" OnValidSubmit="@UploadImage" FormName="UploadImage">
               <DataAnnotationsValidator />
<ValidationSummary />
               <div class="form-group">
                   <InputFile OnChange="OnFileChange" />
               <button type="submit" class="btn btn-primary">Upload</button>
```

```
<a href="@($"/movies/edit?id={movie.Id}")">Edit</a> |
                   <a href="@($"/movies")">Back to List</a>
       </div>
      ∨@code {
           Movie? movie;
           IBrowserFile? selectedFile;
           EditContext editContext;
           [SupplyParameterFromQuery]
           public int Id { get; set; }
           int newRate;
           int srednia = 1;
           protected override async Task OnInitializedAsync()
               movie = await DB.Movies.FirstOrDefaultAsync(m => m.Id == Id);
               if (movie is null)
                   NavigationManager.NavigateTo("notfound");
                editContext = new EditContext(new object());
           private async Task SubmitRating()
                if (movie is not null)
                   if (movie.Rate.HasValue)
                       movie.Rate = ((movie.Rate.Value * srednia) + newRate) / (1 + srednia);
                       srednia++;
                   else
                       movie.Rate = newRate;
                   await DB.SaveChangesAsync();
                   await OnInitializedAsync();
100
           private void OnFileChange(InputFileChangeEventArgs e)
                selectedFile = e.File;
```

```
private void OnFileChange(InputFileChangeEventArgs e)
            {
104
                selectedFile = e.File;
107
           private async Task UploadImage()
                if (selectedFile is not null)
                    var uploadsFolder = Path.Combine("wwwroot", "uploads");
                    if (!Directory.Exists(uploadsFolder))
                       Directory.CreateDirectory(uploadsFolder);
                   var filePath = Path.Combine(uploadsFolder, selectedFile.Name);
                   using (var stream = new FileStream(filePath, FileMode.Create))
                        await selectedFile.OpenReadStream().CopyToAsync(stream);
                    if (movie is not null)
                       movie.ImagePath = $"/uploads/{selectedFile.Name}";
                        await DB.SaveChangesAsync();
                        await OnInitializedAsync();
```

2. Opis działania

W pierwszej części zadania jedyna zmiana 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.