

Platformy programistyczne .Net i Java

Laboratorium 4

Filip Ziolo (272543)

1 Wstęp

Druga część polegała na somodzielnym zaimplementowania pod strony która będzie pozwalala na dodawanie filmów i przechowywanie ich w bazie danych. Dodatkową funkcją którądodałem było logowanie się przez google.

```
1 @code {
2     [Parameter]
3     public int Id { get; set; }
4
5     private Movie movie = new();
6
7     protected override async Task OnInitializedAsync()
8     {
9         movie = await DbContext.Movies.FindAsync(Id);
10    }
11 }
12
13
14 @if (movie != null)
15 {
16     <h3>@movie.Title</h3>
17     <p><strong>Release Date:</strong> @movie.ReleaseDate?.
18         ToShortDateString()</p>
19     <p><strong>Description:</strong> @movie.Description</p>
20     <p><strong>Current Rate:</strong> @movie.Rate/10</p>
21     @if (!string.IsNullOrEmpty(movie.ImageUrl))
22     {
23         
25     }
26 }
```

Ten kod wyświetla szczegóły konkretnego filmu na podstawie przekazanego parametru Id. W metodzie OnInitializedAsync() pobierany jest film z bazy danych przy użyciu DbContext.Movies.FindAsync(Id). Na stronie wyświetlane są tytuł, data premiery, opis, ocena oraz opcjonalnie obrazek filmu, jeśli dostępny jest jego adres URL.

```
1 <h3>Movies List</h3>
2
3 <table class="table">
4     <thead>
```

```

5         <tr>
6             <th>
7                 <button class="btn btn-link" @onclick="@(() => SortBy("
                        Title"))">Title</button>
8             </th>
9             <th>
10                <button class="btn btn-link" @onclick="@(() => SortBy("
                        Rate"))">Rate</button>
11            </th>
12            <th>Actions</th>
13        </tr>
14    </thead>
15    <tbody>
16        @foreach (var movie in movies)
17        {
18            <tr>
19                <td>@movie.Title</td>
20                <td>@movie.Rate</td>
21                <td>
22                    <a href="/moviedetails/@movie.Id" class="btn btn-sm
                        btn-primary">Details</a>
23                </td>
24            </tr>
25        }
26    </tbody>
27</table>
28
29
30@code {
31    private List<Movie> movies = new();
32    private Movie newMovie = new();
33    private bool sortAscending = true;
34    private string? sortColumn = null;
35
36    protected override async Task OnInitializedAsync()
37    {
38        movies = await DbContext.Movies.ToListAsync();
39    }
40
41    private async Task HandleValidSubmit()
42    {
43        var authState = await AuthStateProvider.
44            GetAuthenticationStateAsync();
45        var user = authState.User;
46
47        if (user.Identity != null && user.Identity.IsAuthenticated)
48        {
49            newMovie.UserId = user.FindFirst(c => c.Type == "sub")?.
                Value;
            newMovie.UserName = user.Identity.Name;














































```

```

50     }
51
52     DbContext.Movies.Add(newMovie);
53     await DbContext.SaveChangesAsync();
54     newMovie = new();
55     movies = await DbContext.Movies.ToListAsync();
56 }
57
58 private void SortBy(string columnName)
59 {
60     if (sortColumn == columnName)
61     {
62         sortAscending = !sortAscending;
63     }
64     else
65     {
66         sortColumn = columnName;
67         sortAscending = true;
68     }
69
70     switch (columnName)
71     {
72         case "Title":
73             movies = sortAscending ?
74                 movies.OrderBy(m => m.Title).ToList() :
75                 movies.OrderByDescending(m => m.Title).ToList();
76             break;
77
78         case "Rate":
79             movies = sortAscending ?
80                 movies.OrderBy(m => m.Rate).ToList() :
81                 movies.OrderByDescending(m => m.Rate).ToList();
82             break;
83     }
84 }

```

Ten kod przedstawia listę filmów wyświetlaną w tabeli z możliwością sortowania według tytułu lub oceny. Użytkownik może dodać nowy film, który zostanie zapisany w bazie danych, przy czym identyfikator i nazwa użytkownika są pobierane z aktualnego stanu uwierzytelnienia. Sortowanie działa dynamicznie — kliknięcie nagłówka kolumny przełącza porządek rosnący/malejący.

- ▶  Connected Services
- ▶   Properties
- ▶   wwwroot
- ▶   Zależności
- ▲   Components
 - ▶   Account
 - ▲   Layout
 - ▶   MainLayout.razor
 - ▶   NavMenu.razor
 - ▲   Pages
 -   Auth.razor
 -   Counter.razor
 -   Error.razor
 -   Home.razor
 -   Logout.razor
 -   MovieDetails.razor
 -   Movies.razor
 -   Weather.razor
 -   _Imports.razor
 -   App.razor
 -   Routes.razor
- ▲   Data
 - ▶   ApplicationDbContext.cs