Laborator 4 – Aplicatii Web cu ASP.NET Core si Entity Framework Code First

1. In laboratorul curent, vom dezvolta aplicatia web creata la laboratorul anterior. Deschidem Visual Studio si apoi alegem optiunea **Open a project or solution** si cautam proiectul **Nume_Pren_Lab2,** creat anterior. Pentru a putea fi evaluata activitatea aferenta fiecarui laborator, laboratorul curent il vom dezvolta pe un branch nou, diferit de cel master care se afla deja creat. Utilizand pasii indicati in Lab 1, pct. 22 vom crea un branch nou pe care il vom denumi Laborator4
2. In laboratorul curent vom realiza functionalitatile care ne permit sa citim si sa afisam date relationate. Datele relationate sunt incarcate de EF Core in ceea ce numim navigation properties. Astfel dorim sa putem vizualiza toate cartile editate de fiecare editura in parte.
3. Pagina Views/Publishers/Index.cshtml va trebuie sa afiseze date din doua tabele diferite, astfel vom crea un viewmodel pentru a include cele doua tabele ca si proprietati ale viewmodel-ului. In folderul Models, creati un nou folder cu denumire *ViewModels* si acest folder adaugati o noua clasa cu denumirea *Models/ViewModels/PublisherIndexData.cs* cu urmatorul continut:

```
using Nume_Pren_Lab2.Models;

namespace Nume_Pren_Lab2.ViewModels
{
    public class PublisherIndexData
    {

            public IEnumerable<Publisher> Publishers { get; set; }
            public IEnumerable<Book> Books { get; set; }

    }
}
```

4. Vom actualiza clasa IndexModel din fisierul View/Publishers/Index.cshtml.cs conform codului marcat mai jos:

```
public class IndexModel : PageModel
    {
        private readonly Nume_Pren_Lab2.Data.Nume_Pren_Lab2Context _context;

        public IndexModel(Nume_Pren_Lab2.Data.Nume_Pren_Lab2Context context)
        {
            _context = context;
        }

        public IList<Publisher> Publisher { get;set; } = default!;


        public PublisherIndexData PublisherData { get; set; }
        public int PublisherID { get; set; }
        public int BookID { get; set; }

        public async Task OnGetAsync(int? id, int? bookID)
        {
```

```
            PublisherData = new PublisherIndexData();
            PublisherData.Publishers = await _context.Publisher
                .Include(i => i.Books)
                    .ThenInclude(c => c.Author)
                .OrderBy(i => i.PublisherName)
                .ToListAsync();

            if (id != null)
            {
                PublisherID = id.Value;
                Publisher publisher = PublisherData.Publishers
                    .Where(i => i.ID == id.Value).Single();
                PublisherData.Books = publisher.Books;
            }

             }
}
}
```

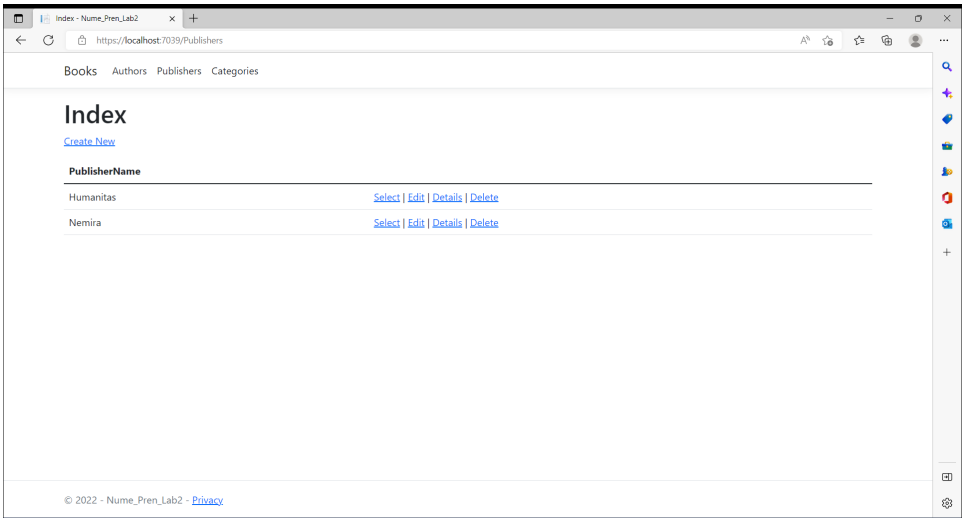
Publisher-ul selectat este preluat din lista de edituri din viewmodel. Proprietatea Books din viewmodel este incarcata cu cartile aflate in navigation property ale editurii (publisher) selectate.
Metoda *Where()* returneaza o colectie, iar pentru a filtra un singur publisher utilizam metoda *Single()*

5. Actualizam pagina Views/Publishers/Index.cshtml conform codului marcat mai jos:

```
@page "{id:int?}"
@model Nume_Pren_Lab2.Pages.Publishers.IndexModel

@{
    ViewData["Title"] = "Index";
}

<h1>Index</h1>

<p>
    <a asp-page="Create">Create New</a>
</p>
<table class="table">
    <thead>
        <tr>
            <th>
                @Html.DisplayNameFor(model =>
model.Publisher[0].PublisherName)
            </th>
            <th></th>
        </tr>
    </thead>
    <tbody>
        @foreach (var item in Model.PublisherData.Publishers)
        {
            string selectedRow = "";
            if (item.ID == Model.PublisherID)
            {
                selectedRow = "table-success";
            }
```

```
            <tr class="@selectedRow">
            <td>
                @Html.DisplayFor(modelItem => item.PublisherName)
            </td>

                <td>
                <a asp-page="./Index" asp-route-id="@item.ID">Select</a> |
                <a asp-page="./Edit" asp-route-id="@item.ID">Edit</a> |
                <a asp-page="./Details" asp-route-id="@item.ID">Details</a> |
                <a asp-page="./Delete" asp-route-id="@item.ID">Delete</a>
            </td>
        </tr>
}
    </tbody>
</table>
```

Rulam aplicatia si verificam ca aplicatia functioneaza navigand la pagina Publishers/Index



6. Modificati clasa Author astfel incat sa avem o proprietate care contine numele si prenumele.

```
public class Author
    {
        public int ID { get; set; }

        public string FirstName { get; set; }

        public string LastName { get; set; }

        [Display(Name = "Full Name")]
        public string FullName
        {
            get
            {
                return FirstName + " " + LastName;
            }
        }
```

```
        public ICollection<Book>? Books { get; set; }
    }
```

7. Pentru a fi afisate cartile editate de o editura selectata (cand apasam pe Select) vom adauga dupa tagul </table>, in fisierul Views/Publishers/Index.cshtml

```
</tbody>
</table>

@if (Model.PublisherData.Books != null)
{
    <h3>Books Edited by Selected Publisher</h3>
    <table class="table">


        @foreach (var item in Model.PublisherData.Publishers)
        {
            string selectedRow = "";
            if (item.ID == Model.PublisherID)
            {
                selectedRow = "table-success";
            }
            <tr class="@selectedRow">

                <td>
                    @{
                        foreach (var b in item.Books)
                        {

                            @b.Title @:  @b.Author.FullName

                        <br />
                        }
                    }
            </td>
            </tr>
        }

    </table>
}
```

Rulati aplicatia, navigati la Publishers/Index si selectati una dintre edituri. In partea de jos a paginii veti putea vizualiza cartile editate de editura selectata.

8. In continuare vom realiza functionalitati legate de sortarea cartilor in pagina Books/Index.cshtml. Astfel in clasa IndexModel din fisierul Books/Index.cshtml.cs adaugam urmatoarele proprietati:

```
public IndexModel(Nume_Pren_Lab2.Data.Nume_Pren_Lab2Context context)
        {
            _context = context;
```

```
        }

        public IList<Book> Book { get;set; } = default!;

        public BookData BookD { get; set; }
        public int BookID { get; set; }
        public int CategoryID { get; set; }


        public string TitleSort { get; set; }
        public string AuthorSort { get; set; }

…….
```

9. In fisierul Books/Index.cshtml.cs, in metoda OnGetAsync adauagam un nou parametru sortOrder. Acest parametru il vom folosi pentru a sorta descrescator fie dupa titlul cartiilor, fie dupa autorul cartilor

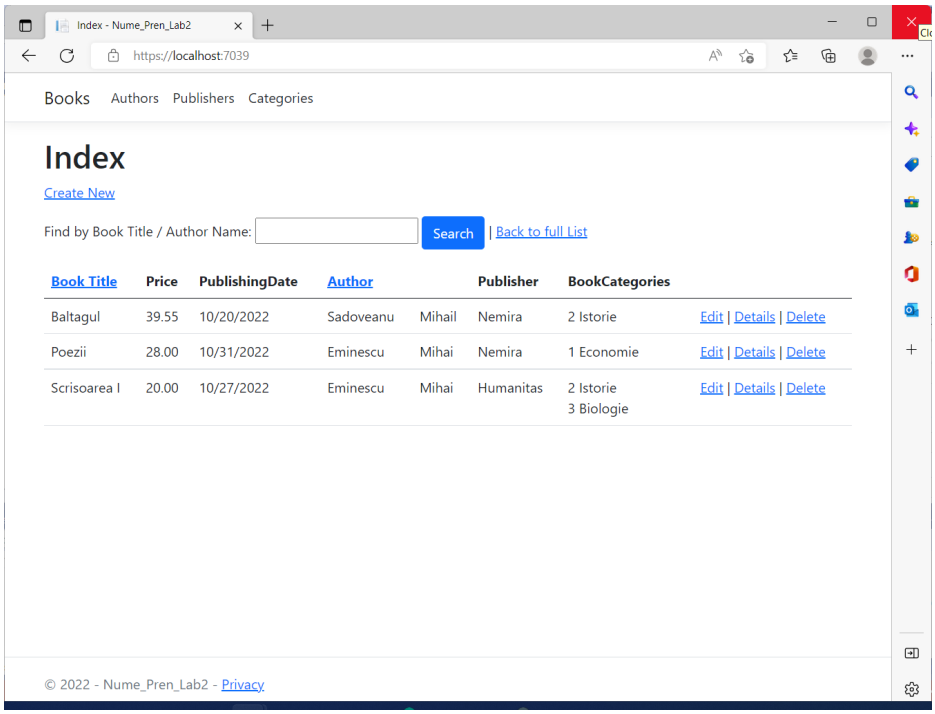
```
public async Task OnGetAsync(int? id, int? categoryID, string sortOrder)
        {
       BookD = new BookData();


       TitleSort = String.IsNullOrEmpty(sortOrder) ? "title_desc" : "";
       AuthorSort =  sortOrder == "author" ? "author_desc" : "author";
            BookD.Books = await _context.Book
                  .Include(b => b.Author)
                  .Include(b => b.Publisher)
                  .Include(b => b.BookCategories)
                  .ThenInclude(b => b.Category)
                  .AsNoTracking()
                  .OrderBy(b => b.Title)
                  .ToListAsync();

            if (id != null)
            {
                BookID = id.Value;
                Book book = BookD.Books
                    .Where(i => i.ID == id.Value).Single();
                BookD.Categories = book.BookCategories.Select(s =>
s.Category);
            }


            switch (sortOrder)
            {
                case "title_desc":
                    BookD.Books =  BookD.Books.OrderByDescending(s =>
s.Title);
                    break;
                case "author_desc":
                    BookD.Books = BookD.Books.OrderByDescending(s =>
s.Author.FullName);
                    break;
                case "author":
                        BookD.Books = BookD.Books.OrderBy(s =>
             s.Author.FullName);
```

```
                    break;
                default:
                    BookD.Books =  BookD.Books.OrderBy (s => s.Title);
                break;


            }
```

10. In fisierul Books/Index.cshtml adaugam hiperlink-uri pentru coloanele pe care dorim sa le sortam (Title si Author), adaugand codul marcat

```
.....

<p>
    <a asp-page="Create">Create New</a>
</p>
<table class="table">
    <thead>
        <tr>
            <th>
                <a asp-page="./Index" asp-route-sortOrder="@Model.TitleSort">
                @Html.DisplayNameFor(model => model.Book[0].Title)
                </a>
            </th>
            <th>
                @Html.DisplayNameFor(model => model.Book[0].Price)
            </th>
            <th>
                @Html.DisplayNameFor(model => model.Book[0].PublishingDate)
            </th>
            <th colspan="2">
                <a asp-page="./Index" asp-route-sortOrder="@Model.AuthorSort">
                @Html.DisplayNameFor(model => model.Book[0].Author)
                </a>
            </th>
            <th>
                @Html.DisplayNameFor(model => model.Book[0].Publisher)
            </th>
            <th>
                @Html.DisplayNameFor(model => model.Book[0].BookCategories)
            </th>

            <th></th>
        </tr>
    </thead>

.................
```

11. Dorim ca in aplicatia realizata sa putem cauta carti fie dupa titlu fie dupa numele autorului similar cu imaginea de mai jos:



Astfel in clasa IndexModel din fisierul Books/Index.cshtml.cs adaugam o noua proprietate:

```
public IndexModel(Nume_Pren_Lab2.Data.Nume_Pren_Lab2Context context)
        {
            _context = context;
        }

        public IList<Book> Book { get;set; } = default!;

        public BookData BookD { get; set; }
        public int BookID { get; set; }
        public int CategoryID { get; set; }


        public string TitleSort { get; set; }
        public string AuthorSort { get; set; }

       public string CurrentFilter { get; set; }


      .......
```

12. In fisierul Books/Index.cshtml.cs modificam metoda OnGetAsync adaugam parametrul searchString , ca carei valoare o vom utiliza pentru a filtra rezultatul cartilor afisate

```
public async Task OnGetAsync(int? id, int? categoryID, string sortOrder, string
searchString)
        {
```

```
            BookD = new BookData();

            // using System;
            TitleSort = String.IsNullOrEmpty(sortOrder) ? "title_desc" : "";
            AuthorSort = String.IsNullOrEmpty(sortOrder) ? "author_desc" : "";

            CurrentFilter = searchString;


            BookD.Books = await _context.Book
                  .Include(b => b.Author)
                  .Include(b => b.Publisher)
                  .Include(b => b.BookCategories)
                  .ThenInclude(b => b.Category)
                  .AsNoTracking()
                  .OrderBy(b => b.Title)
                  .ToListAsync();


            if (!String.IsNullOrEmpty(searchString))
            {
      BookD.Books = BookD.Books.Where(s => s.Author.FirstName.Contains(searchString)

                                          ||s.Author.LastName.Contains(searchString)
                                          || s.Title.Contains(searchString));
            }

            if (id != null)
            {
                BookID = id.Value;
                Book book = BookD.Books
                    .Where(i => i.ID == id.Value).Single();
                BookD.Categories = book.BookCategories.Select(s => s.Category);
            }
……..
```

13. In fisierul Books/Index.cshtml adaugam un formular care va utiliza metoda GET pentru a trimite search string-ul si pentru a filtra rezultatele afisate conform search string-ului

```
@page "/"
@model Nume_Pren_Lab2.Pages.Books.IndexModel

@{
    ViewData["Title"] = "Index";
}

<h1>Index</h1>

<p>
    <a asp-page="Create">Create New</a>
</p>

<form asp-page="./Index" method="get">
    <div class="form-actions no-color">
        <p>
            Find by Book Title / Author Name:
            <input type="text" name="SearchString" value="@Model.CurrentFilter"
```

```
    />
                <input type="submit" value="Search" class="btn btn-primary" /> |
                <a asp-page="./Index">Back to full List</a>
            </p>
        </div>
    </form>

    <table class="table">
        <thead>

    ….
```

Sarcina laborator: Realizati toate modificarile necesare astfel incat navigand la pagina Views/Categories/Index, pentru fiecare categorie afisata, la selectarea acesteia (click pe link) sa se afiseze toate cartile (titlu si autor) care fac parte din acea categorie*.

*vezi exemplu referitor la cartile editate de fiecare editura din laboratorul curent