**Dezvoltarea Aplicatiilor Web utilizand ASP.NET Core MVC**
**Laborator 6**

---

# EXERCITII:

1. Sa se creeze un proiect, in cadrul caruia sa se adauge Entity Framework, baza de date si sa se includa sistemul de migratii (**VEZI Curs 5 – Sectiunea Crearea unui proiect utilizand EF si sistemul de migratii**).

2. Sa se testeze corectitudinea pasilor implementati in cadrul exercitiului 1 adaugand clasa Articles.cs, iar in Controller-ul ArticlesController sa se creeze metoda Index, metoda prin care se afiseaza toate articolele din baza de date. De asemenea, o sa fie nevoie si de un folder Articles in care o sa se creeze View-ul Index.cshtml.

    a. Se adauga Clasa → Article.cs

```
public class Article
    {
        [Key]
        public int ArticleID { get; set; }
        public string Title { get; set; }
        public string Content { get; set; }
        public DateTime Date { get; set; }
    }
```

    b. Se ruleaza sistemul de migratii pentru update-ul bazei de date

    c. Se deschide tabelul si se insereaza 2 intrari in tabel

    d. Se implementeaza metoda Index in cadrul Controller-ului si View-ul asociat in folderul din View.

### Metoda Index din ArticlesController

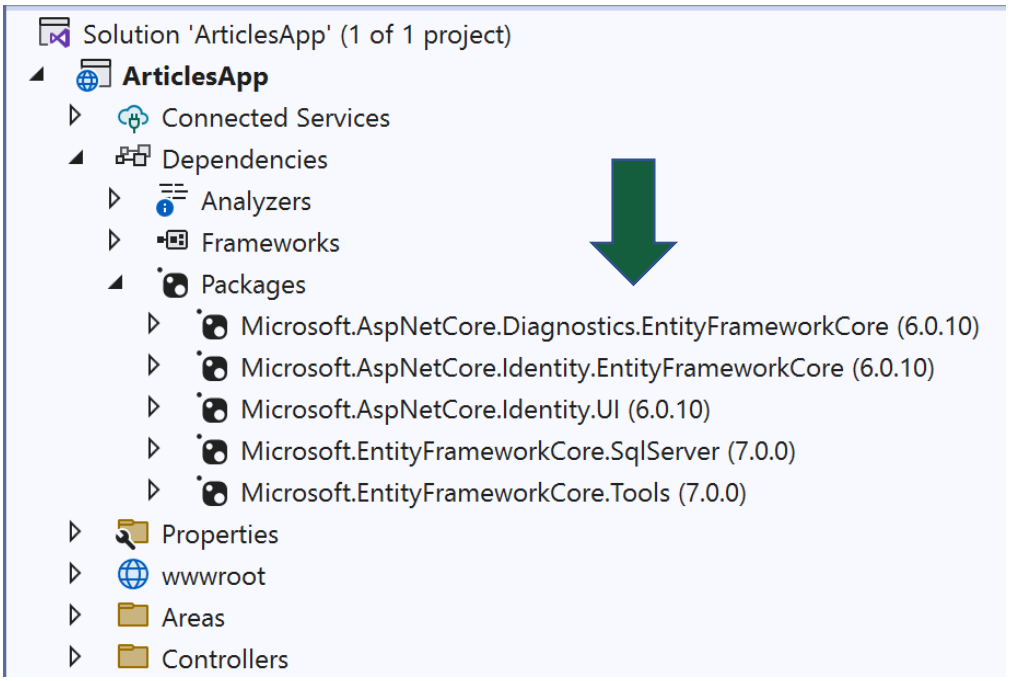
```
public IActionResult Index()
        {
            var articles = from article in db.Articles
                           select article;

            ViewBag.Articles = articles;

            return View();
        }
```

### View-ul Index.cshtml

```
<h2>Afisare articole</h2>
<br />

@foreach (var art in ViewBag.Articles)
{
    <p>@art.Title</p>
    <p>@art.Content</p>
    <p>@art.Date</p>

    <br />
    <a href="/Articles/Show/@art.ArticleID">Afisare
articol</a>
    <br />
    <a href="/Articles/Edit/@art.ArticleID">Editare
articol</a>
    <hr />

}
```

3. Sa se creeze un nou proiect numit **ArticlesApp** de tipul ASP.NET Core Web App (Model-View-Controller). Proiectul o sa contina sistem de autentificare (**VEZI Curs 6 – Sectiunea Adaugarea Sistemului de Autentificare**).

4. Se verifica existenta urmatoarelor pachete:




5. Se ruleaza migratia Update-Database pentru realizarea update-ului in baza de date. Dupa acest pas se pot vedea tabelele in SQL Server Object Explorer. Se ruleaza doar comanda Update-Database deoarece migratia initiala exista.

6. Se ruleaza proiectul si se inregistreaza un cont, dupa care se poate vizualiza noul user in baza de date, tabelul **dbo.AspNetUsers**.

7. Se considera entitatile **Article**, **Category** si **Comment** cu proprietatile de mai jos. De asemenea, se considera cerintele din cadrul **Cursului 6 – sectiunea Exemplu practic pentru proiectarea Diagramei E/R.**

**Article**

- Id (int – primary key)
- Title (string – titlul este obligatoriu)
- Content (string – continutul este obligatoriu)
- Date (DateTime)
- CategoryId (int – cheie externa – categoria din care face parte articolul)

**Category:**

- Id (int – primary key)
- CategoryName (string – numele este obligatoriu)

**Comment:**

- Id (int – primary key)
- Content (string – continutul comentariului este obligatoriu)
- Date (DateTime – data la care a fost postat comentariul)
- Date (DateTime)
- ArticleId (int – cheie externa – articolul caruia ii apartine comentariul)

- sa existe cel putin 4 tipuri de utilizatori: vizitator neinregistrat, utilizator inregistrat, editor si administrator;
- orice utilizator poate vizualiza stirile aparute pe site. Pe pagina principala vor aparea stirile cele mai recente;
- stirile vor fi impartite pe categorii (create dinamic): stiinta, tehnologie, sport, etc, existand posibilitatea de adaugare a noi categorii (administratorul poate face CRUD pe categorii);
- stirile dintr-o anumita categorie sunt afisate intr-o pagina separata unde pot fi sortate dupa diferite criterii: data aparitiei si alfabetic;
- editorii se ocupa de publicarea stirilor noi si pot vizualiza, edita, sterge propriile stiri;

- utilizatorii pot adauga comentarii la stirile aparute, isi pot sterge si edita propriile comentarii;
- stirile pot fi cautate prin intermediul unui motor de cautare propriu;
- administratorii se ocupa de buna functionare a intregii aplicatii (ex: pot face CRUD pe stiri, pe categorii, etc.) si pot activa sau revoca drepturile utilizatorilor si editorilor;

Sa se implementeze cele trei clase in Models, dupa care sa se ruleze migratiile → in acest caz Add-Migration NumeMigratie si Update-Database (**VEZI Pasul urmator – exercitiul 9**).

**Implementarea claselor:**

```
public class Article
{
   [Key]
   public int Id { get; set; }

   [Required(ErrorMessage = "Titlul este obligatoriu")]
   public string Title { get; set; }

   [Required(ErrorMessage = "Continutul articolului este
obligatoriu")]
   public string Content { get; set; }

   public DateTime Date { get; set; }

   [Required(ErrorMessage = "Categoria este obligatorie")]
   public int CategoryId { get; set; }

   public virtual Category Category { get; set; }

   public virtual ICollection<Comment> Comments { get; set; }
}
```

```
public class Category
{
   [Key]
   public int Id { get; set; }

   [Required(ErrorMessage = "Numele categoriei este
obligatoriu")]
   public string CategoryName { get; set; }

   public virtual ICollection<Article> Articles { get; set; }
}
```

```
public class Comment
{
   [Key]
   public int CommentId { get; set; }

   [Required(ErrorMessage = "Continutul este obligatoriu")]
   public string Content { get; set; }

   public DateTime Date { get; set; }

   public int ArticleId { get; set; }

   public virtual Article Article { get; set; }
}
```

8. Se adauga proprietatile in contextul bazei de date pentru realizarea ulterioara a migratiilor.

```
public DbSet<Article> Articles { get; set; }
public DbSet<Category> Categories { get; set; }
public DbSet<Comment> Comments { get; set; }
```

9. Sa se ruleze sistemul de migratii, dupa care sa se insereze manual in fiecare tabel 2-3 intrari (in acest caz se ruleaza ambele comenzi).

10. Sa se adauge cate un Controller pentru fiecare clasa → **ArticlesController**, **CategoriesController** si **CommentsController** in care se vor implementa operatiile CRUD asupra entitatilor.

11. Sa se adauge cate un folder in Views pentru fiecare Controller.

12. Implementati operatii CRUD asupra entitatilor, urmand pasii urmatori.

- Sa existe posibilitatea realizarii operatiilor **C.R.U.D. pentru entitatea Article** astfel:
    - **Index** – afisarea tuturor articolelor, impreuna cu denumirea categoriei din care fac parte – se poate utiliza din Bootstrap -> Cards: https://getbootstrap.com/docs/5.2/components/card/#about
    - **Show** – afisarea unui singur articol (intr-o pagina separata) impreuna cu denumirea categoriei din care face parte articolul respectiv
    - **New** – posibilitatea adaugarii unui nou articol. In momentul in care se adauga articolul, categoria se va selecta dintr-o lista existenta de categorii, folosind un  element de tipul dropdown
    - **Edit** – posibilitatea editarii unui articol. In momentul in care se editeaza articolul, categoria se va selecta dintr-o lista existenta de categorii (element de tipul dropdown)
    - **Delete** – posibilitatea stergerii unui articol

- Sa existe posibilitatea realizarii operatiilor **C.R.U.D. pentru entitatea Category** (Atentie! In momentul in care se doreste stergerea unei categorii, automat se vor sterge si toate articolele care fac parte din categoria respectiva).
- Sa existe posibilitatea realizarii operatiilor **C.R.U.D. pentru entitatea Comment** astfel:
    - **Afisarea tuturor comentariilor** corespunzatoare unui articol. Fiecare articol o sa aiba un buton “Afisare articol” (Show) care redirectioneaza catre o pagina in care vom avea articolul impreuna cu toate comentariile corespunzatoare articolului respectiv

Pentru afisarea articolelor se poate utiliza din Bootstrap > Cards: https://getbootstrap.com/docs/5.2/components/card/#about

- **New** – posibilitatea adaugarii unui nou comentariu
- **Edit** – posibilitatea editarii unui comentariu existent
- **Delete** – posibilitatea stergerii unui comentariu

**Surse utile:**

**Bootstrap – v5.2** → https://getbootstrap.com/docs/5.2/getting-started/introduction/

**Bootstrap – icons** → https://icons.getbootstrap.com/

**Bootstrap – buttons** → https://getbootstrap.com/docs/5.2/components/buttons/

**Bootstrap – spacing** → https://getbootstrap.com/docs/5.0/utilities/spacing/

**Flexbox** → https://css-tricks.com/snippets/css/a-guide-to-flexbox/