Dezvoltarea Aplicațiilor Web utilizând ASP.NET Core MVC
Curs 5 – Baza de Date - MAC OS

Crearea unui proiect utilizând EF și sistemul de migrații

PASUL 1 – Instalare .NET Core

Pe langa instalarea Visual Studio 2022, mai este necesara si instalarea .NET Core:

https://dotnet.microsoft.com/en-us/download/dotnet/8.0

Se selecteaza **x64 pe MAC OS cu INTEL** si **ARM64** pe **MAC OS cu procesor M**.

Se ruleaza in linia de comanda:
ln -s /usr/local/share/dotnet/dotnet /usr/local/bin/

PASUL 2 – Instalare server MySQL

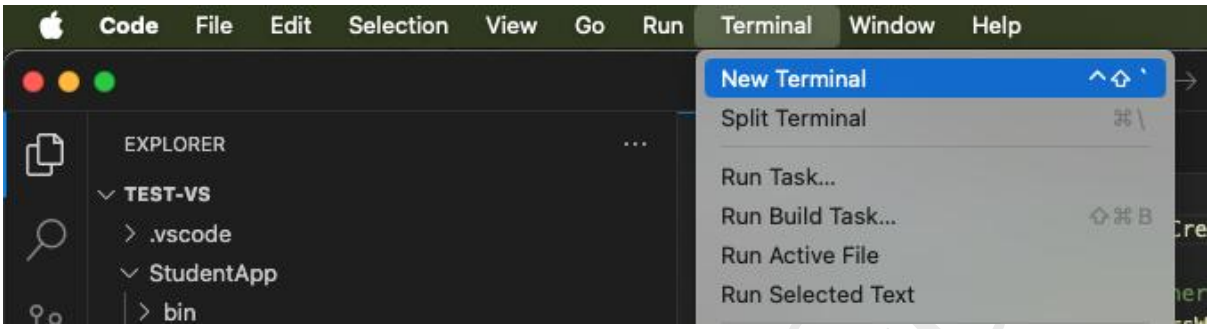
Se instaleaza serverul MySQL urmarind pasii din urmatorul tutorial, pana la comanda **mysql_secure_installation** inclusiv:

https://flaviocopes.com/mysql-how-to-install/

PASUL 3 – Crearea proiectului

Se creeaza un nou proiect, procedand la fel ca in cursurile anterioare. Proiectul o sa se numeasca **Lab5**.

Se deschide Visual Studio Code intr-un folder unde se doreste crearea proiectului. In IDE, se acceseaza consola prin deschiderea unui nou terminal, din meniul “Terminal”:



In terminalul deschis, se ruleaza comanda pentru crearea proiectului:

dotnet new mvc -o **NumeAplicatie**

PASUL 4 – Adaugare Entity Framework Core

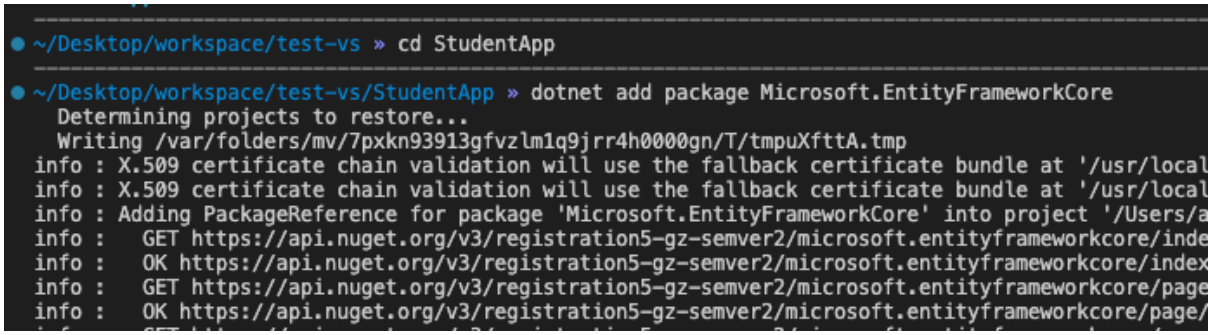
Pentru a adauga pachete in cadrul proiectului, se foloseste consola. Comanda pentru a instala un pachet este:

dotnet add package **NumelePachetului**

In cadrul noului proiect se adauga EF selectandu-se urmatoarele pachete:

Microsoft.EntityFrameworkCore
Microsoft.EntityFrameworkCore.Design
Microsoft.EntityFrameworkCore.SqlServer
Microsoft.EntityFrameworkCore.Tools
Pomelo.EntityFrameworkCore.MySql

Pentru a rula comanda de instalare a pachetelor cu succes, este necesara schimbarea directorului de lucru in acela in care se regaseste aplicatia. De exemplu, daca aplicatia se numeste “StudentApp”, trebuie sa rulam comanda in cadrul folderului, si nu in root-ul proiectului.



PASUL 5 – Conexiunea cu Baza de Date

In continuare se creeaza baza de date impreuna cu un user care trebuie sa aiba drepturi asupra bazei de date. Acest lucru se face pentru fiecare aplicatie noua.

// se creeaza baza de date

mysql> **create database lab5;**

Query OK, 1 row affected (0.01 sec)

// se creeaza un user cu username daw_example si parola Password1!

mysql> **CREATE USER 'daw_example'@'localhost' IDENTIFIED WITH mysql_native_password BY 'Password1!';**

Query OK, 0 rows affected (0.01 sec)

// userul primeste drepturi asupra bazei de date numita lab5

mysql> **grant all privileges on lab5.* to 'daw_example'@'localhost';**

Query OK, 0 rows affected (0.00 sec)

// se inchide sesiunea cu serverul MySQL
mysql> **exit**

PASUL 6 – Configurarea Stringului de conexiune

Varianta fără dependency injection:

Se creeaza in Model clasa AppDbContext pentru adaugarea conexiunii la baza de date.

```
public class AppDBContext : DbContext
{
   public AppDBContext() : base ()
   {

   }

   protected override void OnConfiguring
(DbContextOptionsBuilder options)

   {
        var connectionString =
"server=localhost;database=lab5;uid=daw_example;password=Passwor
d1!";

       var serverVersion = new MySqlServerVersion(new
Version(8, 0, 31));

     options.UseMySql(connectionString, serverVersion);

   }
   public DbSet<Student> Students { get; set; }
}
```

Valorile pe care trebuie sa le configuram in codul de mai sus sunt:

- **database** – numele bazei de date creata in pasul anterior
- **uid** – numele de utilizator creat in pasul anterior
- **password** – parola utilizatorului creat in pasul anterior

Versiunea serverului de baze de date se poate afla conectandu-ne din terminal la baza de date cu utilizatorul creat la pasul anterior.

```
~/Projects/Lab5/Lab5 » mysql -uroot -p
Enter password:
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 29
Server version: 8.0.31 Homebrew

Copyright (c) 2000, 2022, Oracle and/or its affiliates.
```

Varianta cu dependency injection:

În **Program.cs** – se adaugă serviciul pentru conexiunea cu baza de date

```
// Add services to the container.
var connectionString =
builder.Configuration.GetConnectionString("DefaultConnection") ??
throw new InvalidOperationException("Connection string
'DefaultConnection' not found.");

builder.Services.AddDbContext<ApplicationDbContext>(options =>
    options.UseMySql(connectionString,
ServerVersion.Parse("8.0.31")));

builder.Services.AddDatabaseDeveloperPageExceptionFilter();
```

În **appsettings.json** – se adaugă stringul de conexiune (utilizând configurația din cadrul pasului anterior)

```
{
  "ConnectionStrings": {
    "DefaultConnection":
"Server=localhost;database=lab5;uid=daw_example;password=Password1!"
  },

  "Logging": {
    "LogLevel": {
      "Default": "Information",
      "Microsoft.AspNetCore": "Warning"
    }
  },
  "AllowedHosts": "*"
}
```

Se adaugă clasa, numită sugestiv, **AppDbContext**. Aceasta o sa aibă doar constructorul, astfel:

```
public class AppDbContext : DbContext
{
      public AppDbContext(DbContextOptions<AppDbContext>
options)
         : base(options)
    {
    }

}
```

În final, în cadrul fiecărui Controller se realizează conexiunea cu baza de date astfel:

```
public class ArticlesController : Controller
{
        private readonly AppDbContext db;

        public ArticlesController(AppDbContext context)
        {
             db = context;
        }
...}
```

PASUL 7 – Adaugarea migratiilor in baza de date

În folderul proiectului, în linia de comandă, se rulează comenzile prin care se creează migrația și se realizează update-ul bazei de date.

```
~/Projects/Lab5/Lab5 » dotnet ef migrations add
InitialMigration

Build started...
Build succeeded.
Done. To undo this action, use 'ef migrations remove'

~/Projects/Lab5/Lab5 » dotnet ef database
update

Build started...
Build succeeded.
Applying migration '20221106130348_InitialMigration'.
Done.
```

Pasul 8 – Managementul bazei de date

Managementul bazei de date se poate realiza in doua moduri:

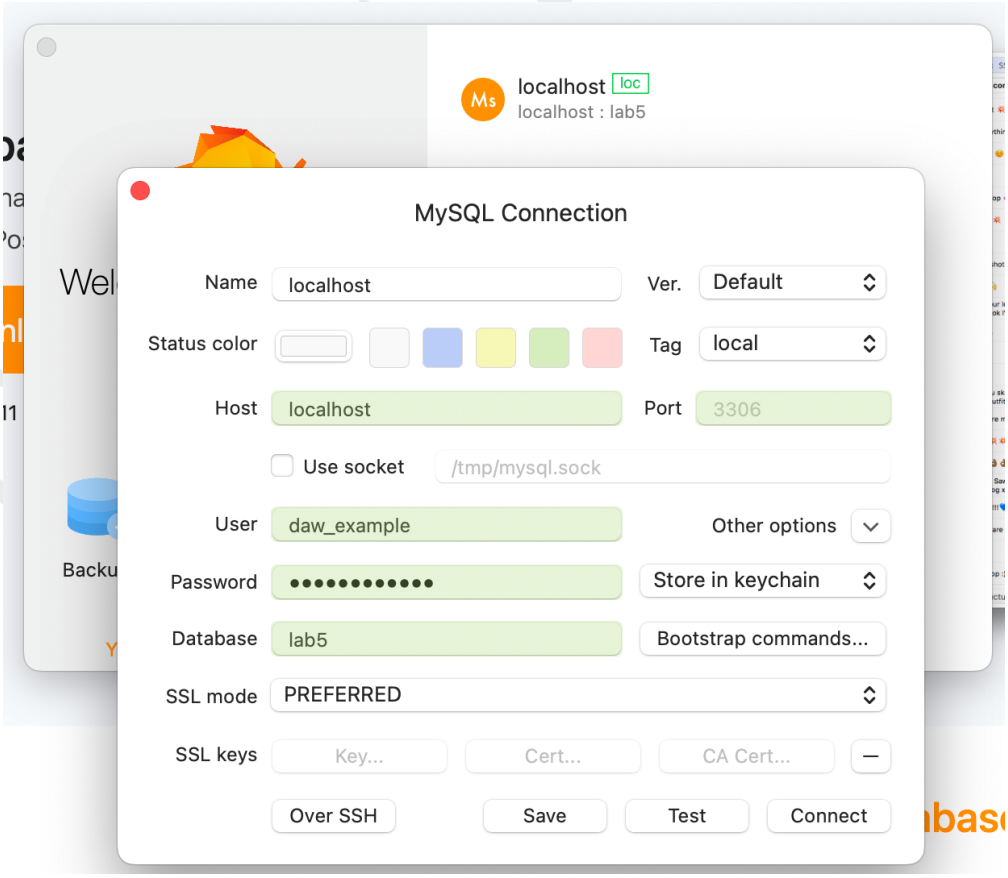
1. **Din terminal**

```
mysql> use lab5;
Reading table information for completion of table and column
names
You can turn off this feature to get a quicker startup with -A
Database changed

mysql> show tables;
+----------------------+
| Tables_in_lab5       |
+----------------------+
| __EFMigrationsHistory |
| Articles             |
mysql> desc Articles;
```

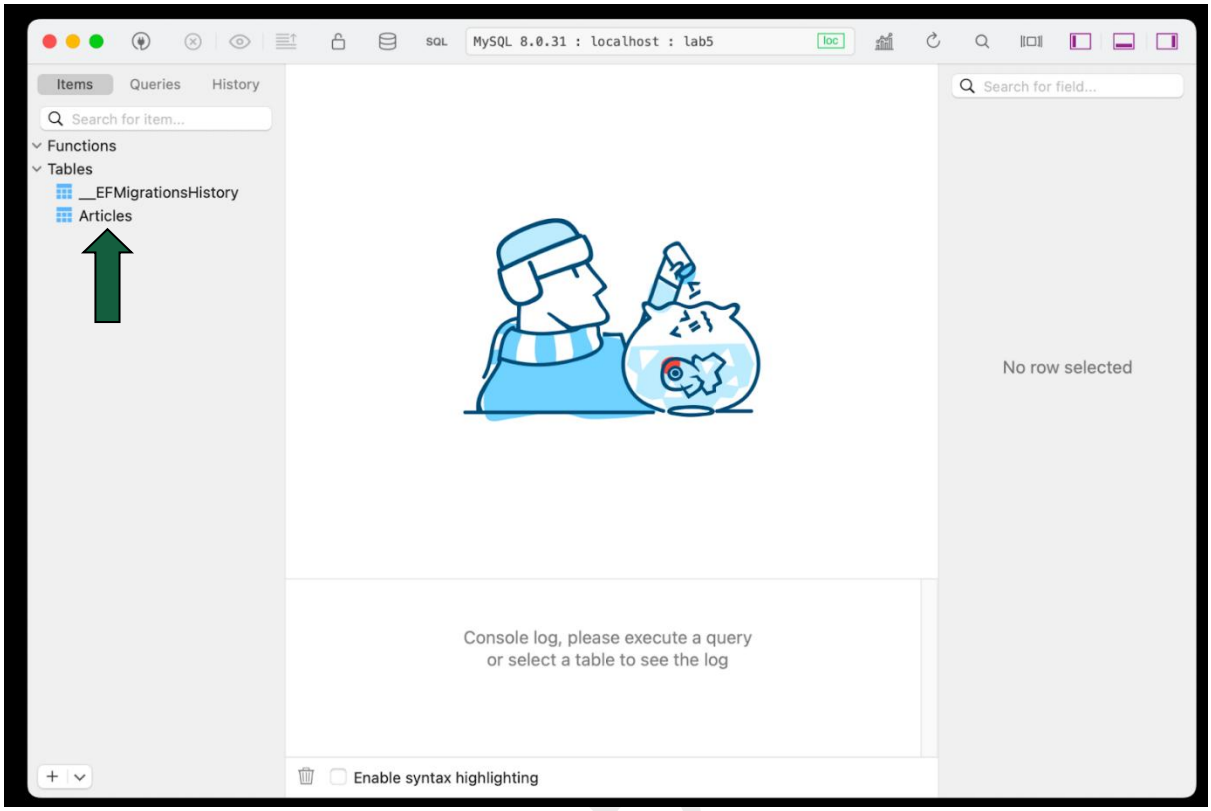
```
+-----------+-------------+------+-----+---------+--------------
-+
| Field     | Type        | Null | Key | Default |
Extra          |
+-----------+-------------+------+-----+---------+--------------
-+
| ArticleId | int         | NO   | PRI | NULL    | auto_increment
|
| Title     | longtext    | NO   |     |
NULL    |                |
| Content   | longtext    | NO   |     |
NULL    |                |
| Date      | datetime(6) | NO   |     |
NULL    |                |
+-----------+-------------+------+-----+---------+--------------
-+
   4  rows in set (0.01 sec)
```

2. **Utilizand un utilitar extern cum este TablePlus (https://tableplus.com/)**

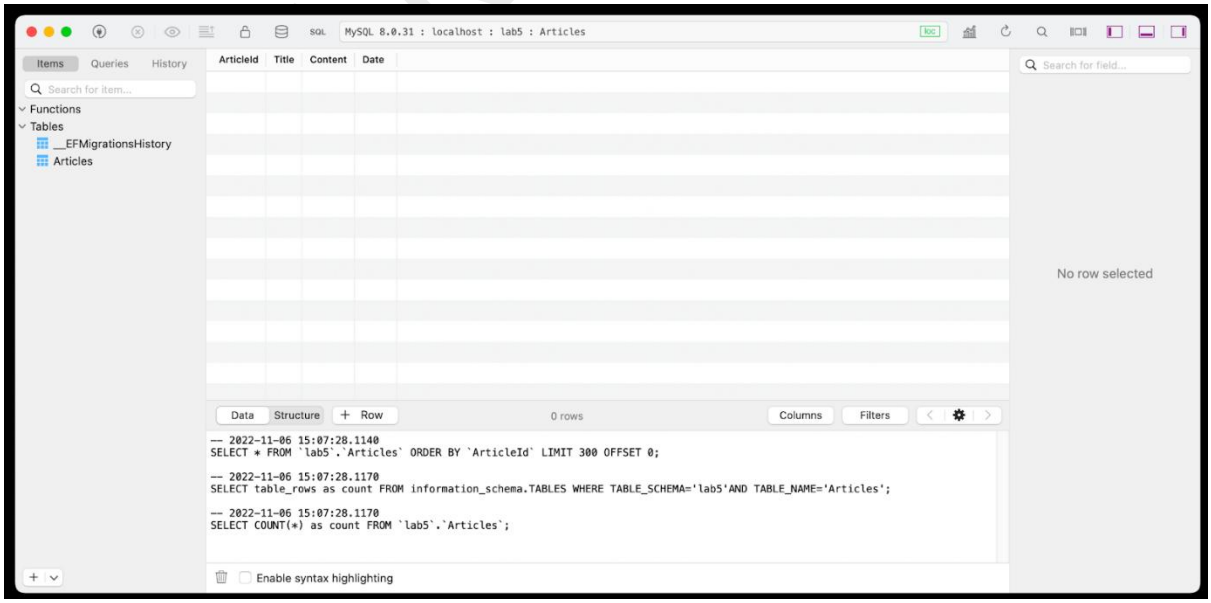
Dupa instalare se porneste si se creeaza o conexiune:



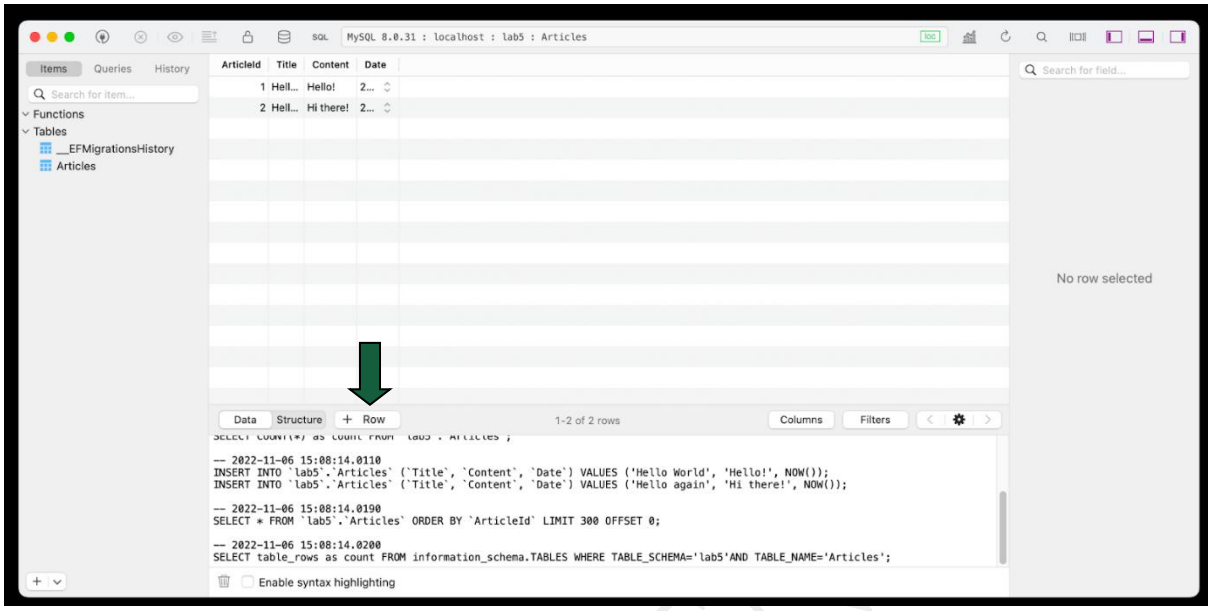
In acest moment avem acces la tabele



Se face click pe tabele pentru a vizualiza/adauga



Se adauga o intrare in baza de date apasand + Row.
Pentru salvare se folosesc comenzile CMD + S.



C.R.U.D. utilizand Entity Framework

In urmatoarea parte a cursului vom implementa operatiile de tip CRUD asupra entitatii Student, utilizand Entity Framework.

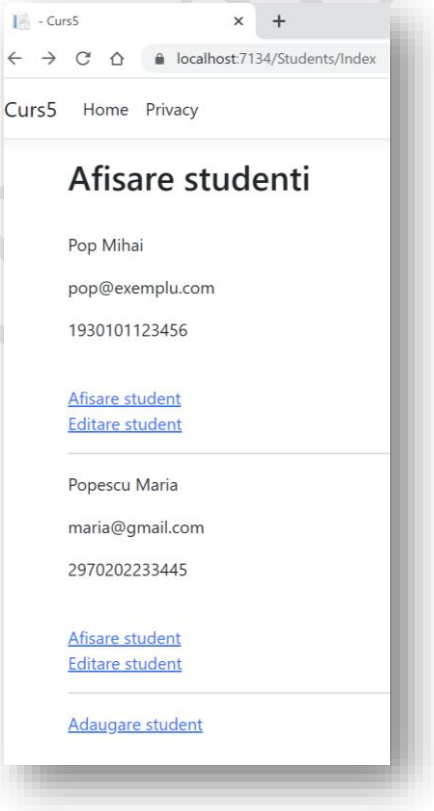
Index

```
private readonly AppDbContext db;

public ArticlesController(AppDbContext context)
{
    db = context;
}

public IActionResult Index()
  {
      var students = from student in db.Students
                     orderby student.Name
                     select student;

      ViewBag.Students = students;

      return View();
  }
```

Preluam toti studentii din baza de date, ordonati dupa nume prin intermediul db.Students

Index.cshtml

```
<h2>Afisare studenti</h2>
<br />

@foreach (var student in ViewBag.Students)
{
      <p>@student.Name</p>
      <p>@student.Email</p>
      <p>@student.CNP</p>

      <br />
      <a href="/Students/Show/@student.StudentID">Afisare
student</a>
      <br />
      <a href="/Students/Edit/@student.StudentID">Editare
student</a>
      <hr />

}

<a href="/Students/New">Adaugare student</a>
```



Show

```
public ActionResult Show(int id)
   {
       Student student = db.Students.Find(id);
       ViewBag.Student = student;
       return View();
    }
```

Metoda Find() primeste ca parametru o valoare pentru coloana care este cheie primara

Show.cshtml

```
<h2>Afisare student</h2>

<br />

<p>@ViewBag.Student.Name</p>
<p>@ViewBag.Student.Email</p>
<p>@ViewBag.Student.CNP</p>

<br />

<a href="/Students/Index">Afisare studenti</a>
```

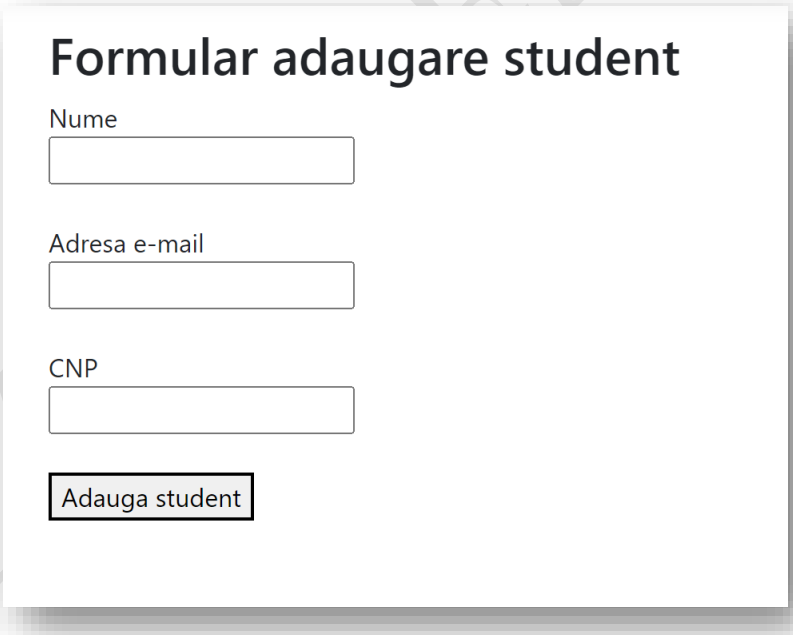
New

```
public IActionResult New()
   {
       return View();
   }

   [HttpPost]
   public IActionResult New(Student s)
   {
       try
       {
           db.Students.Add(s);
           db.SaveChanges();
           return RedirectToAction("Index");
       }
       catch (Exception)
       {
           return View();
       }

   }
```

Students.Add primeste ca parametru un obiect de tip Student iar SaveChanges va face commit in baza de date

New.cshtml

```
<h2>Formular adaugare student</h2>

<form method="post" action="/Students/New">
    <label>Nume</label>
    <br />
    <input type="text" name="Name" />
    <br /><br />
    <label>Adresa e-mail</label>
    <br />
    <input type="text" name="Email" />
    <br /><br />
    <label>CNP</label>
    <br />
    <input type="text" name="CNP" />
    <br />
    <br />
    <button type="submit">Adauga student</button>
</form>
```



Model Binding

In ASP.NET MVC **model binding** ne permite sa facem legatura intre request-urile de tip HTTP si un Model. Model binding este procesul de creare a obiectelor folosind datele trimise de browser printr-un request HTTP (prin intermediul formularelor din View).

Model binding este o legatura intre request-urile HTTP si metodele unui Controller (Actiuni). Deoarece datele trimise prin POST sau GET ajung intotdeauna la Controller, acest mecanism de binding leaga in mod automat variabilele de request cu atributele publice ale modelului. Aceasta mapare se va face dupa **numele atributelor modelului**.

```
<label>Nume</label>
<input type="text" name="Name" />
<label>Adresa e-mail</label>
<input type="text" name="Email" />
<label>CNP</label>
<input type="text" name="CNP" />
```

Parametrii care se vor trimite prin request la controller

/!\ OBSERVATIE

Este necesar ca numele campurilor din View sa coincida cu numele atributelor pentru ca binding-ul sa functioneze.

Edit

```
public IActionResult Edit(int id)
   {
       Student student = db.Students.Find(id);
       ViewBag.Student = student;
       return View();
   }

   [HttpPost]
   public ActionResult Edit(int id, Student requestStudent)
   {
       Student student = db.Students.Find(id);

       try
       {
           student.Name = requestStudent.Name;
           student.Email = requestStudent.Email;
           student.CNP = requestStudent.CNP;
           db.SaveChanges();

           return RedirectToAction("Index");
       }
       catch (Exception)
       {
           return RedirectToAction("Edit", student.StudentID);
       }
    }
```

Edit.cshtml

```
<h2>Editare student</h2>

<br />

<form method="post"
action="/Students/Edit/@ViewBag.Student.StudentID">

    <label>Nume</label>
    <br />
    <input type="text" name="Name" value="@ViewBag.Student.Name" />
    <br /><br />
    <label>Adresa e-mail</label>
    <br />
    <input type="text" name="Email" value="@ViewBag.Student.Email" />
    <br /><br />
    <label>CNP</label>
    <br />
```

```
    <input type="text" name="CNP" value="@ViewBag.Student.CNP" />
    <br />
    <button type="submit">Modifica student</button>

</form>
```

Delete

```
[HttpPost]
public ActionResult Delete(int id)
{
    Student student = db.Students.Find(id);
    db.Students.Remove(student);
    db.SaveChanges();
    return RedirectToAction("Index");
}
```

Remove primeste ca parametru un obiect de tip Student. SaveChanges salveaza modificarile

Show.cshtml (se va utiliza view-ul show)

```
<form method="post"
action="/Students/Delete/@ViewBag.Student.StudentID">

    <button type="submit">Sterge studentul</button>

</form>
```

/!\ OBSERVATIE

In momentul in care sunt necesare in baza de date, fie adaugari sau stergi de tabele, fie adaugari sau stergi de coloane sau proprietati, este nevoie de o noua migratie in baza de date.

De exemplu: daca se doreste adaugarea atributului **Address** in clasa Student → `public string Address { get; set; }`

Se adauga proprietatea, dupa care se executa o noua migratie

→ `Add-Migration AddAddressToStudent`

→ `Update-Database`