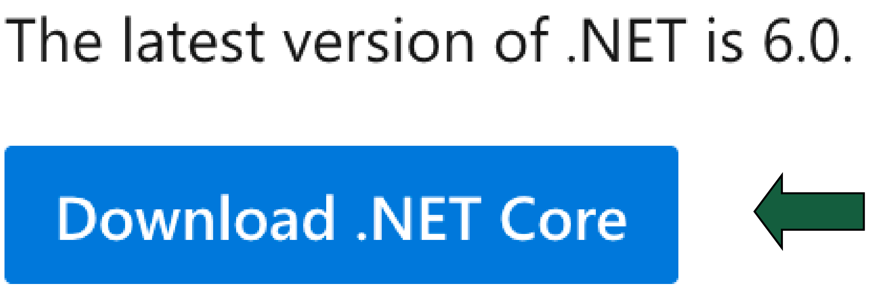
Dezvoltarea Aplicatiilor Web utilizand ASP.NET Core MVC
Curs 5 – Baza de Date - MAC OS

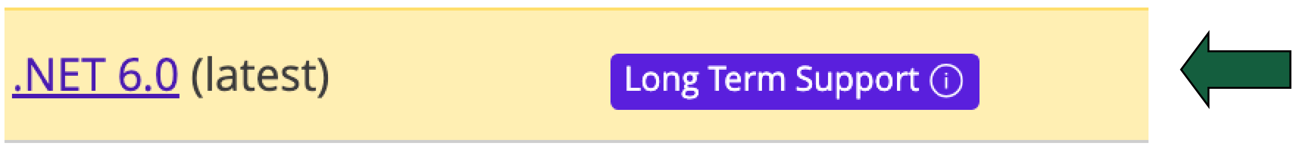
Crearea unui proiect utilizand EF si sistemul de migratii

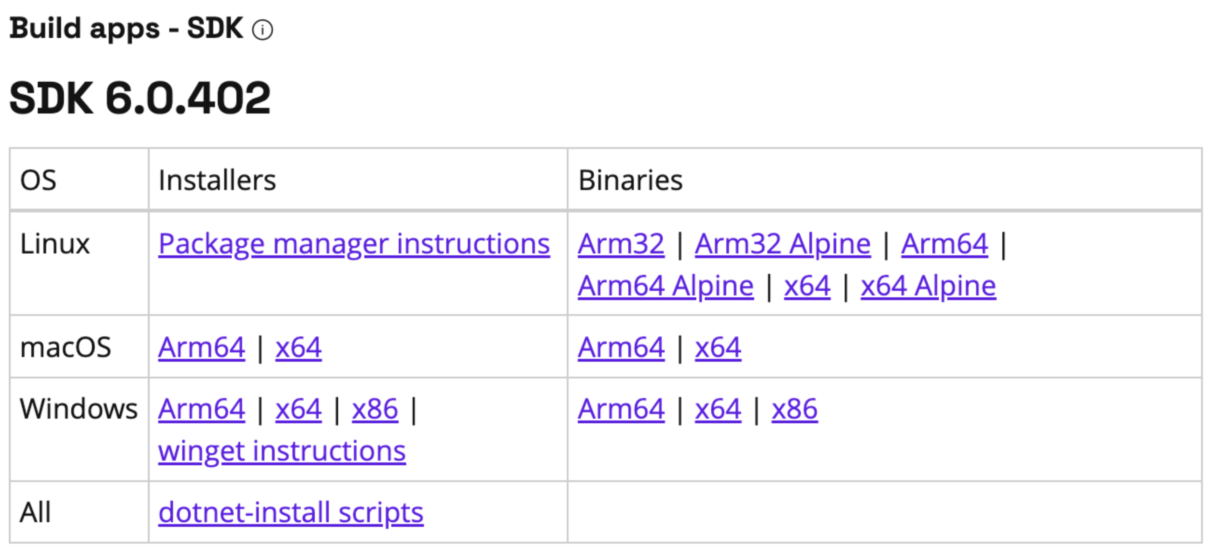
PASUL 1 – Instalare .NET Core

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

https://learn.microsoft.com/enus/dotnet/core/install/macos?tabs=netcore2x#dependencis


The latest version of .NET is 6.0.





Build apps - SDK ⓘ

SDK 6.0.402

OS	Installers	Binaries
Linux	Package manager instructions	Arm32 | Arm32 Alpine | Arm64 | Arm64 Alpine | x64 | x64 Alpine
macOS	Arm64 | x64	Arm64 | x64
Windows	Arm64 | x64 | x86 | winget instructions	Arm64 | x64 | x86
All	dotnet-install scripts	

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/

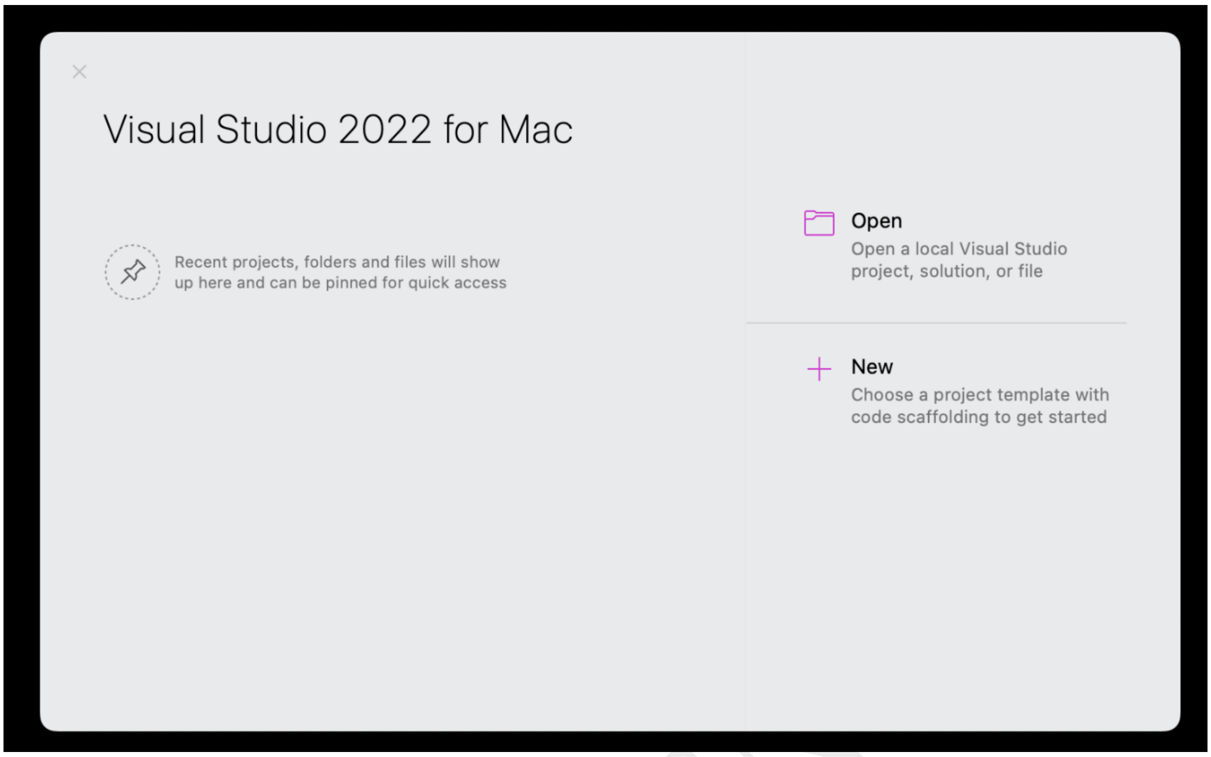
PASUL 2 – Instalare server MySQL

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

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

PASUL 3 – Crearea proiectului

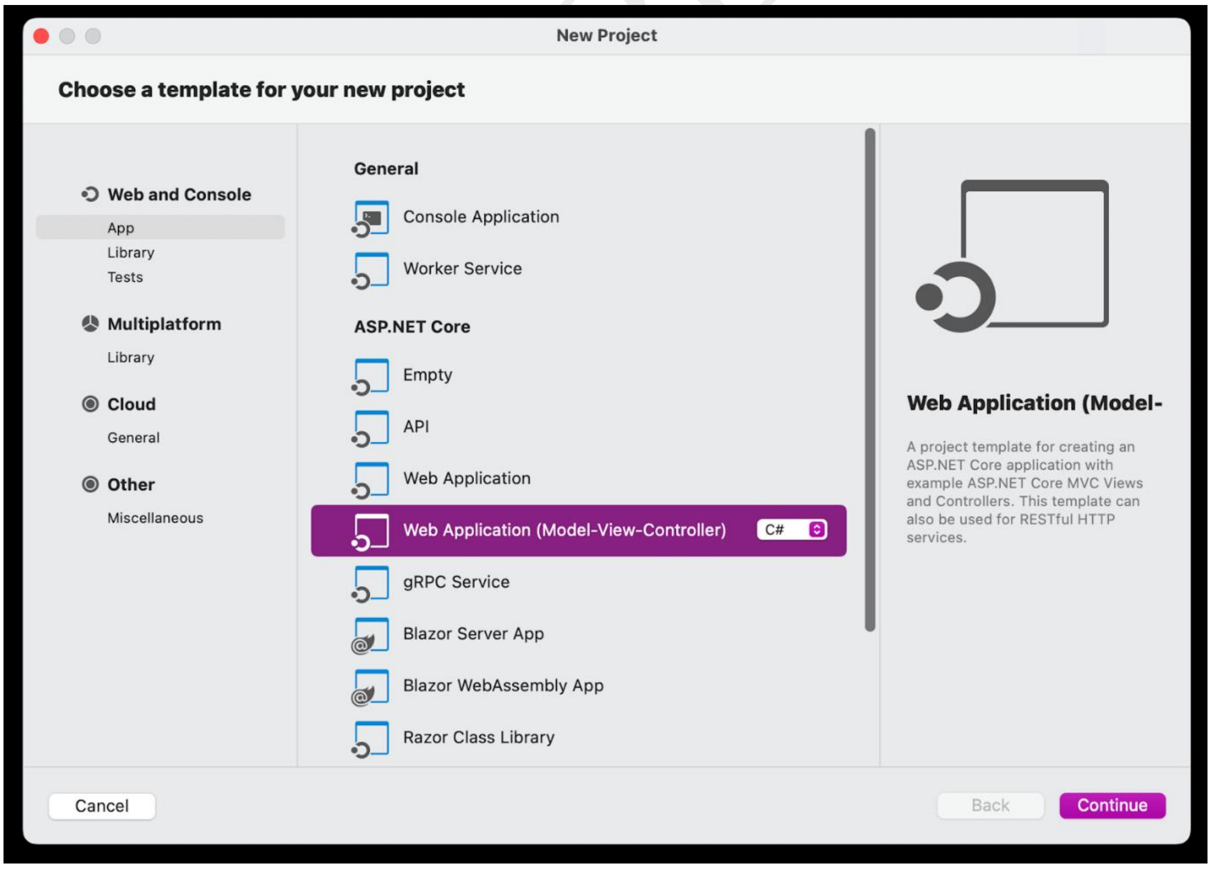
Se creeaza un nou proiect, procedand la fel ca in cursurile anterioare. Proiectul o sa se numeasca **Lab5**.


Visual Studio 2022 for Mac

Recent projects, folders and files will show up here and can be pinned for quick access

Open
Open a local Visual Studio project, solution, or file

New
Choose a project template with code scaffolding to get started


New Project

Choose a template for your new project

Web and Console
- App
- Library
- Tests

Multiplatform
- Library

Cloud
- General

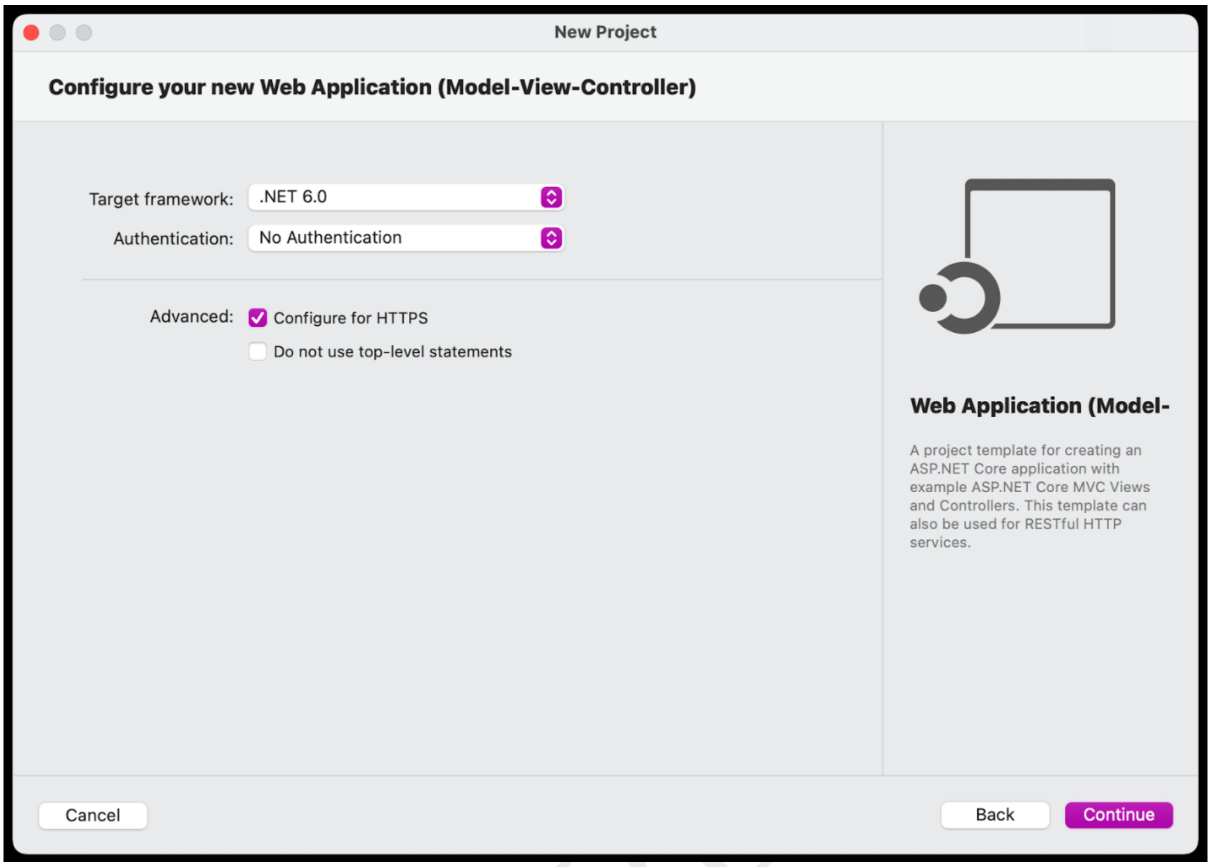
Other
- Miscellaneous

General
- Console Application
- Worker Service

ASP.NET Core
- Empty
- API
- Web Application
- Web Application (Model-View-Controller) C#
- gRPC Service
- Blazor Server App
- Blazor WebAssembly App
- Razor Class Library

Web Application (Model-

A project template for creating an ASP.NET Core application with example ASP.NET Core MVC Views and Controllers. This template can also be used for RESTful HTTP services.


New Project

Configure your new Web Application (Model-View-Controller)

Target framework: .NET 6.0

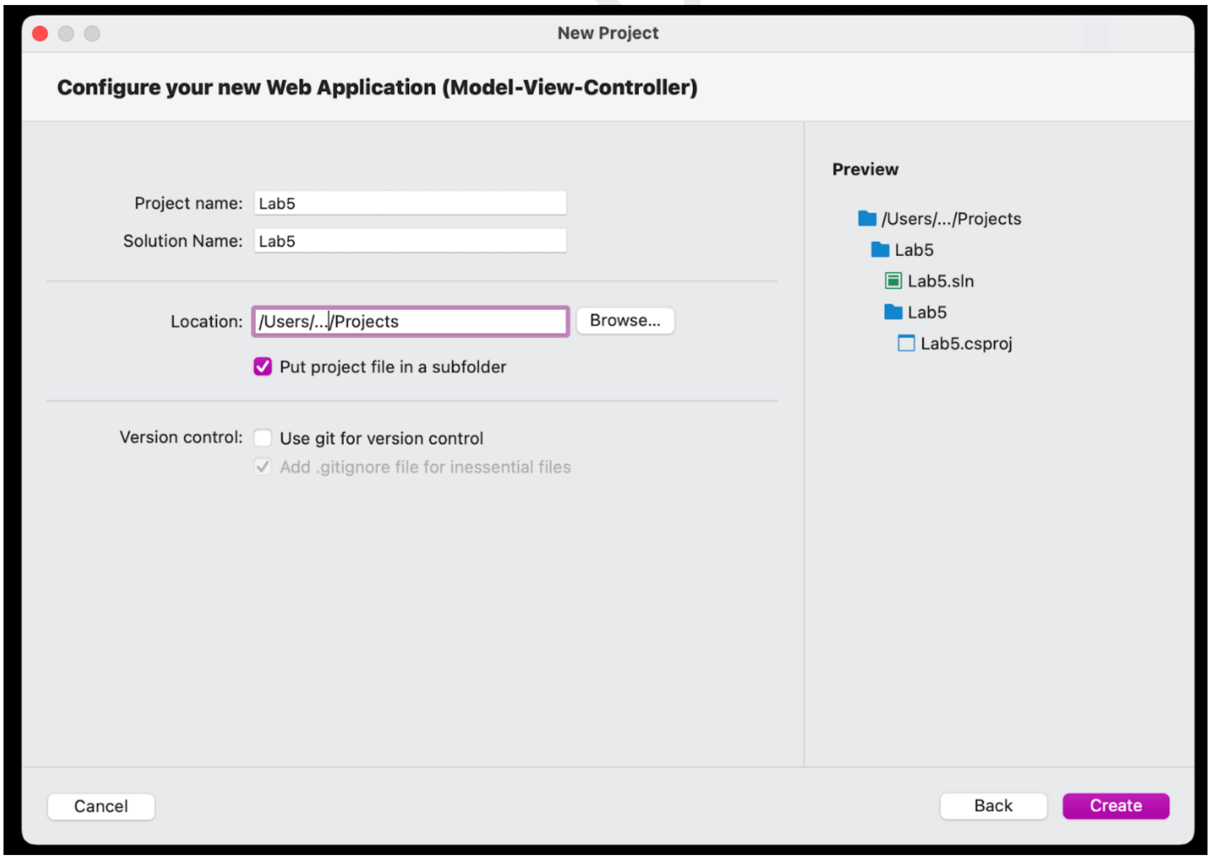
Authentication: No Authentication

Advanced: [✓] Configure for HTTPS

[ ] Do not use top-level statements

Web Application (Model-

A project template for creating an ASP.NET Core application with example ASP.NET Core MVC Views and Controllers. This template can also be used for RESTful HTTP services.


New Project

Configure your new Web Application (Model-View-Controller)

Project name: Lab5

Solution Name: Lab5

Location: /Users/.../Projects

[✓] Put project file in a subfolder

Version control: [ ] Use git for version control

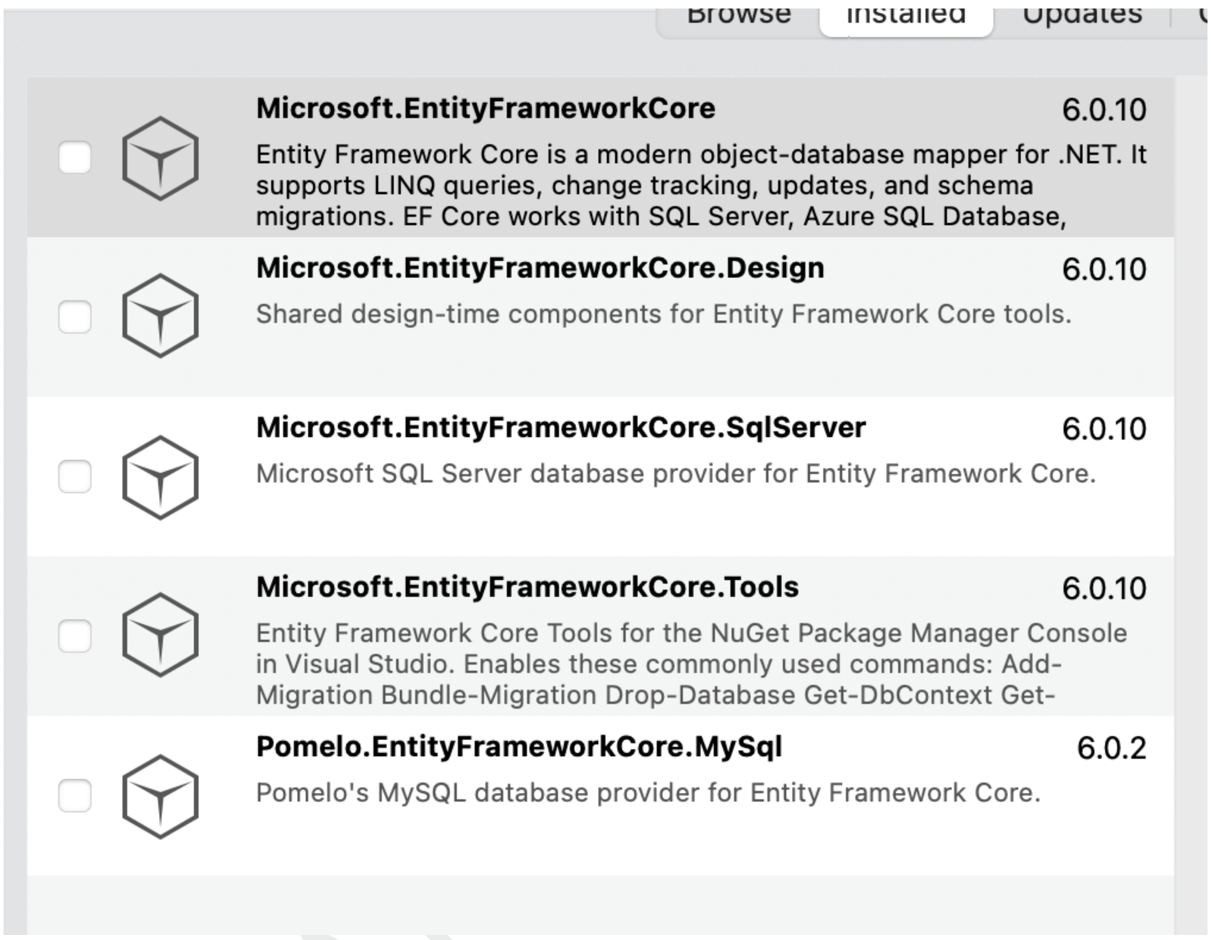
[✓] Add .gitignore file for inessential files

Preview

- /Users/.../Projects
  - Lab5
    - Lab5.sln
    - Lab5
      - Lab5.csproj

PASUL 4 – Adaugare Entity Framework Core

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



Microsoft.EntityFrameworkCore 6.0.10
Entity Framework Core is a modern object-database mapper for .NET. It supports LINQ queries, change tracking, updates, and schema migrations. EF Core works with SQL Server, Azure SQL Database,

Microsoft.EntityFrameworkCore.Design 6.0.10
Shared design-time components for Entity Framework Core tools.

Microsoft.EntityFrameworkCore.SqlServer 6.0.10
Microsoft SQL Server database provider for Entity Framework Core.

Microsoft.EntityFrameworkCore.Tools 6.0.10
Entity Framework Core Tools for the NuGet Package Manager Console in Visual Studio. Enables these commonly used commands: Add-Migration Bundle-Migration Drop-Database Get-DbContext Get-

Pomelo.EntityFrameworkCore.MySql 6.0.2
Pomelo's MySQL database provider for Entity Framework Core.

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
Bye
```

PASUL 6 – Configurarea Stringului de conexiune

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

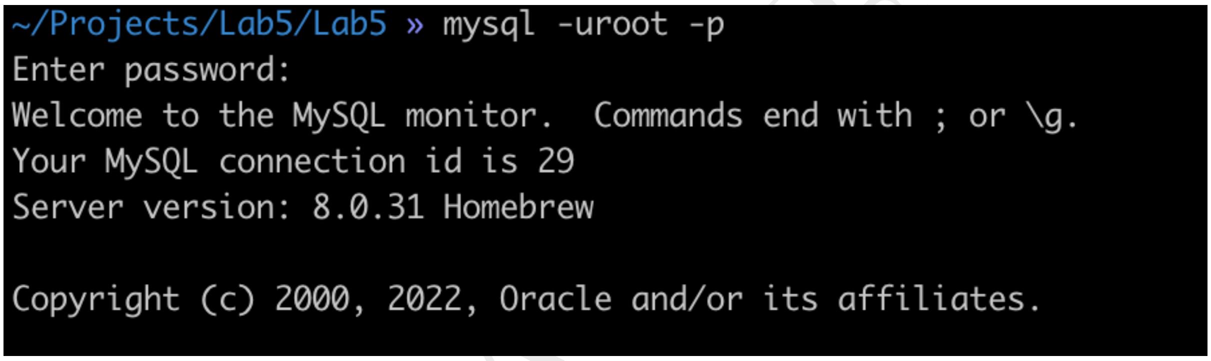
```
public class AppDBContext : DbContext
{
   public AppDBContext() : base ()
   {

   }

   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.
```

PASUL 7 – Adaugarea migratiilor in baza de date

In folderul proiectului, in linia de comanda, se ruleaza comenzile prin care se creeaza migratia si se realizeaza 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:

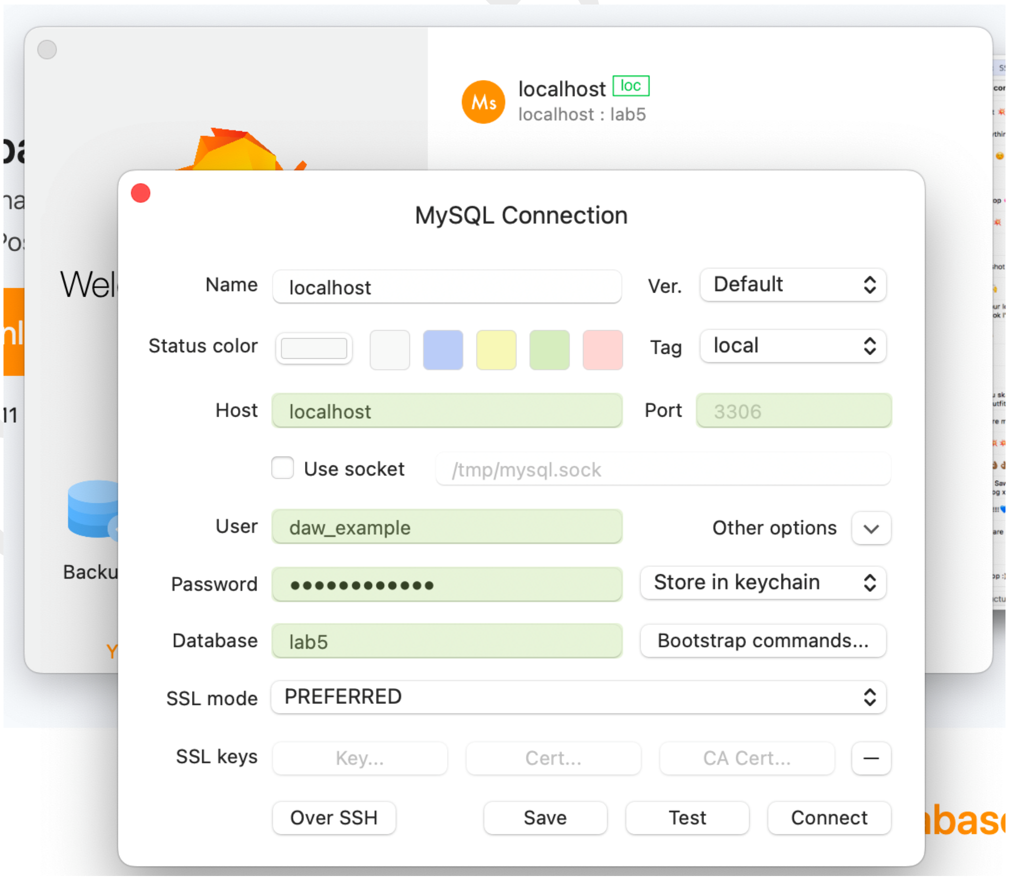
1. Din terminal

```
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:

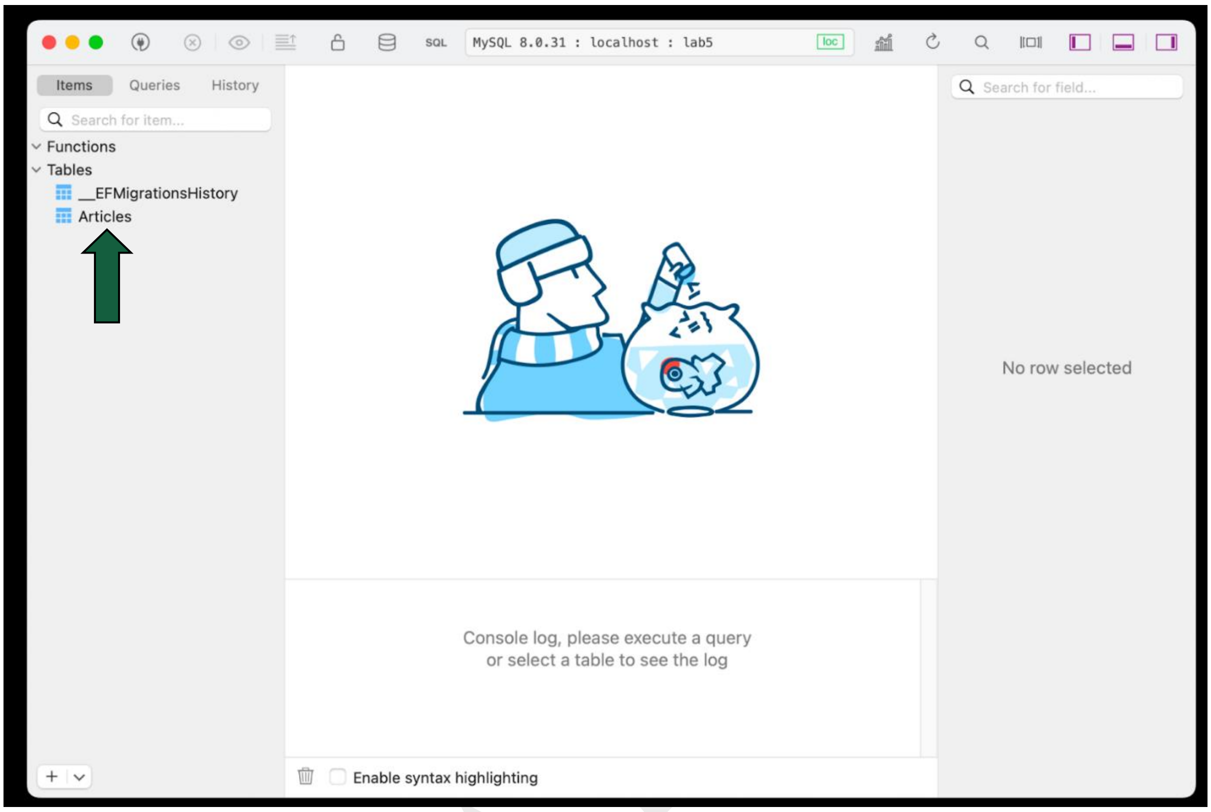


localhost loc
localhost : lab5

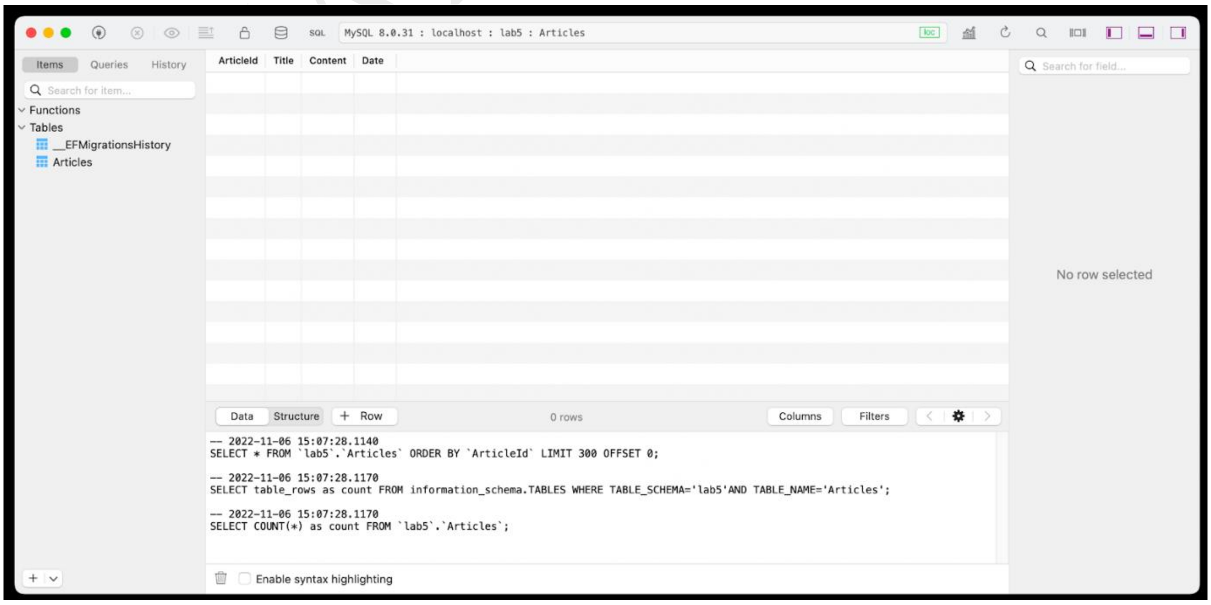
MySQL Connection

Name: localhost
Ver.: Default
Status color
Tag: local
Host: localhost
Port: 3306
Use socket: /tmp/mysql.sock
User: daw_example
Other options
Password: ●●●●●●●●●●●●
Store in keychain
Database: lab5
Bootstrap commands...
SSL mode: PREFERRED
SSL keys: Key... Cert... CA Cert...

In acest moment avem acces la tabele

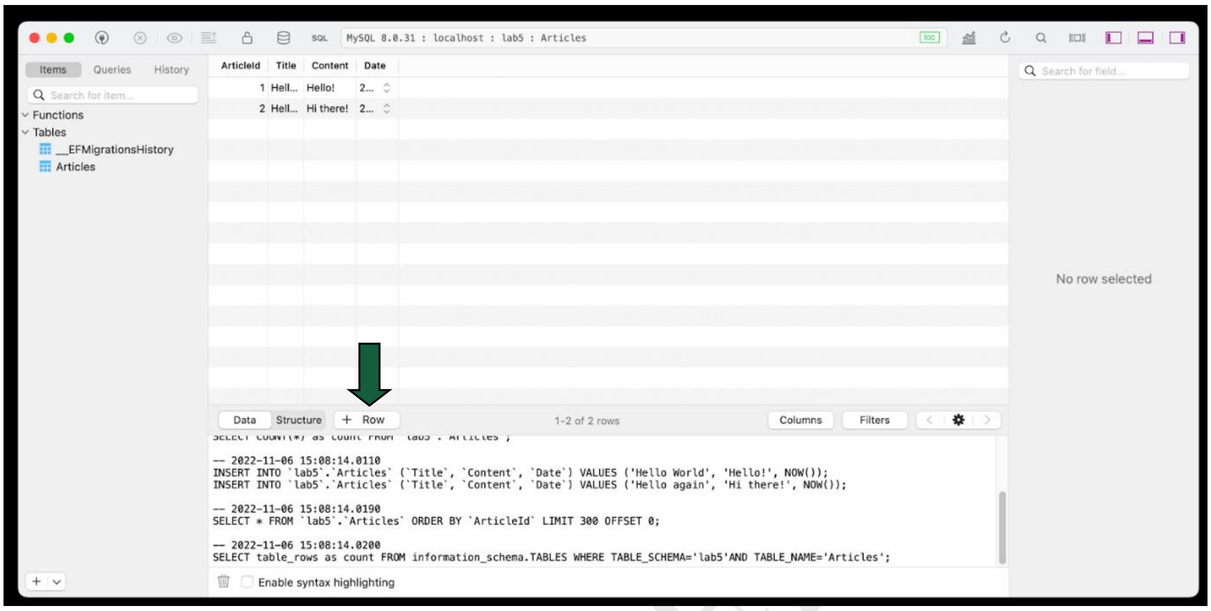

Items
Queries
History
Search for item...
Functions
Tables
__EFMigrationsHistory
Articles
MySQL 8.0.31 : localhost : lab5
Search for field...
No row selected
Console log, please execute a query
or select a table to see the log
Enable syntax highlighting

Se face click pe tabele pentru a vizualiza/adauga



```
-- 2022-11-06 15:07:28.1140
SELECT * FROM `lab5`.`Articles` ORDER BY `ArticleId` LIMIT 300 OFFSET 0;

-- 2022-11-06 15:07:28.1170
SELECT table_rows as count FROM information_schema.TABLES WHERE TABLE_SCHEMA='lab5'AND TABLE_NAME='Articles';

-- 2022-11-06 15:07:28.1170
SELECT COUNT(*) as count FROM `lab5`.`Articles`;
```

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



ArticleId	Title	Content	Date
1	Hell...	Hello!	2...
2	Hell...	Hi there!	2...

```
SELECT COUNT(*) as count FROM `lab5`.`Articles`;

-- 2022-11-06 15:08:14.0110
INSERT INTO `lab5`.`Articles` (`Title`, `Content`, `Date`) VALUES ('Hello World', 'Hello!', NOW());
INSERT INTO `lab5`.`Articles` (`Title`, `Content`, `Date`) VALUES ('Hello again', 'Hi there!', NOW());

-- 2022-11-06 15:08:14.0190
SELECT * FROM `lab5`.`Articles` ORDER BY `ArticleId` LIMIT 300 OFFSET 0;

-- 2022-11-06 15:08:14.0200
SELECT table_rows as count FROM information_schema.TABLES WHERE TABLE_SCHEMA='lab5'AND TABLE_NAME='Articles';
```

C.R.U.D. utilizand Entity Framework

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

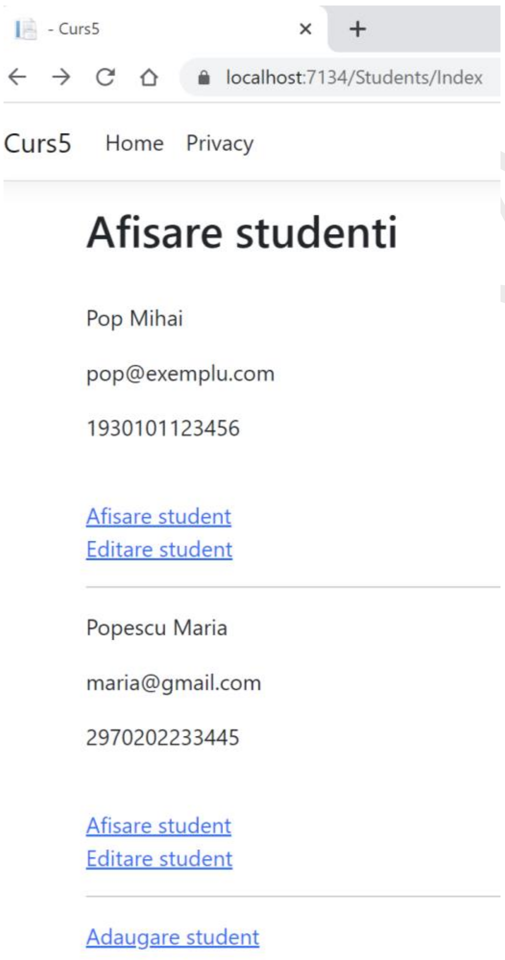
Index

```
private AppDBContext db = new AppDBContext();

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>
```


- Curs5
localhost:7134/Students/Index
Curs5 Home Privacy

Afisare studenti

Pop Mihai

pop@exemplu.com

1930101123456

Afisare student
Editare student

Popescu Maria

maria@gmail.com

2970202233445

Afisare student
Editare student

Adaugare student

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>
```

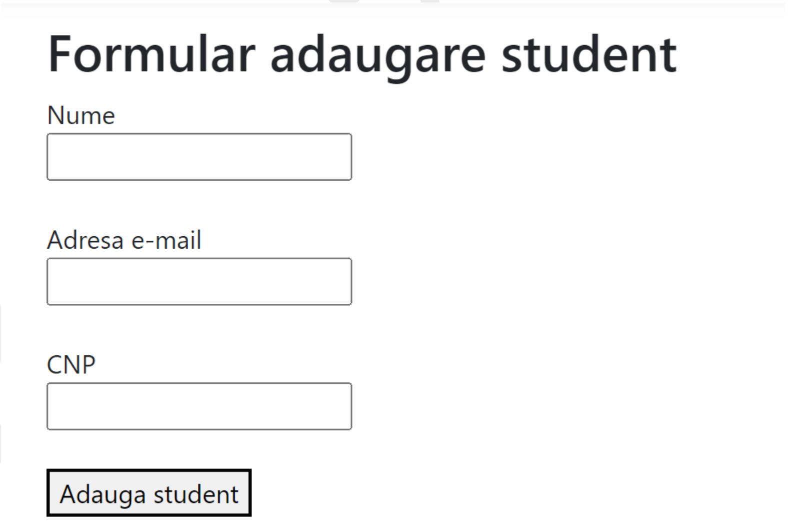
New

```
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`