Dezvoltarea Aplicatiilor Web utilizand ASP.NET Core MVC
Curs 5

Cuprins

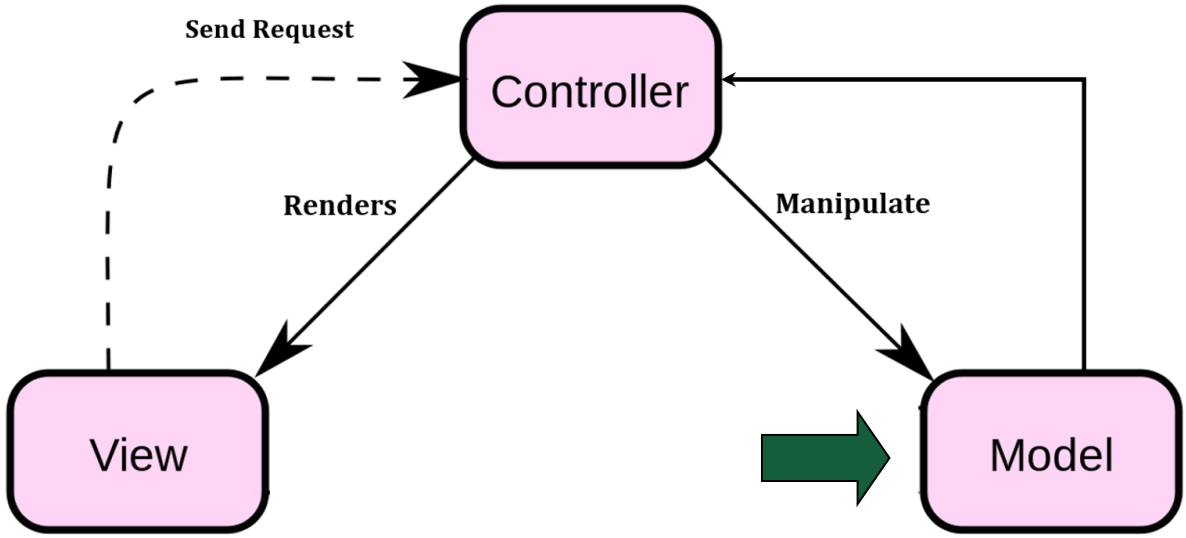
Model (Stratul business – prelucrarea datelor) 2
Ce este Modelul 2
Entity Framework Core 3
Entity Framework 3
Entity Framework Core 4
Instalare Entity Framework Core 4
Entity Framework Core – Migratii 11
Ce sunt migratiile? 11
Crearea unui proiect utilizand EF si sistemul de migratii 12
Crearea proiectului 12
Adaugare Entity Framework Core 12
Adaugarea Modelului 12
Conexiunea cu Baza de Date 15
Adaugarea unei baze de date SQL Server 16
Crearea migratiilor in baza de date 21
C.R.U.D. utilizand Entity Framework 22
Index 22
Show 24
New 24
Model Binding 26
Edit 27
Delete 28

CEZARA BENEGUI

Model (Stratul business – prelucrarea datelor)

Ce este Modelul

Modelul este responsabil cu gestionarea datelor din aplicatie si manipularea acestora. Acesta raspunde cererilor care vin din View prin intermediul Controller-ului, modelul comunicand doar cu Controller-ul. Este cel mai de jos nivel care se ocupa cu **procesarea** si **manipularea** datelor, reprezentand nucleul aplicatiei, fiind cel care realizeaza legatura cu baza de date. **Modelul** ofera accesul la date prin intermediul atributelor publice ale claselor.



Entity Framework Core

Entity Framework

Pentru stocarea datelor in ASP.NET MVC se utilizeaza o tehnologie numita open source numita **Entity Framework (EF)**.

Entity Framework este un **ORM** (**Object Relational Mapper**) pentru .NET, si anume este o colectie de librarii care coreleaza fiecare clasa dintr-un model cu o baza de date. Scopul utilizarii EF este acela de a permite dezvoltatorilor sa se focuseze pe dezvoltarea propriu-zisa a aplicatiei si nu pe baza de date.

Procesarea datelor se poate realiza si prin metode clasice, de exemplu utilizand ADO.NET, dar EF ofera posibilitatea implementarii facile a operatiilor de tip CRUD (Create, Read, Update, Delete).

De asemenea, in cadrul EF se poate utiliza **LINQ (Language Integrated Query)** ajutand la integrarea oricarui **RDBMS** (**Relational Database Management System**) → Oracle SQL, SQL Server, etc. Un RDBMS stocheaza date in tabele, pe care ulterior le acceseaza si prelucreaza cu ajutorul unui limbaj **SQL (Structured Query Language)**. Un RDBMS asigura securitatea, integritatea si consistenta datelor.

LINQ permite integrarea query-urilor SQL in cadrul codului C#.

CEZARA BENEGUI

Entity Framework Core

Entity Framework Core este versiunea mai light a EF, cross-platform (Linux, Windows) care functioneaza foarte bine impreuna cu ASP.NET Core. Entity Framework Core suporta, la fel ca EF, atat tehnica **database-first** cat si **code-first**.

EF Core contine atat posibilitatea integrarii unui RDBMS (Oracle SQL, Microsoft SQL Server, MySQL), cat si integrarea unor baze de date non-relationale (MongoDB, Redis, CassandraDB).

Instalare Entity Framework Core

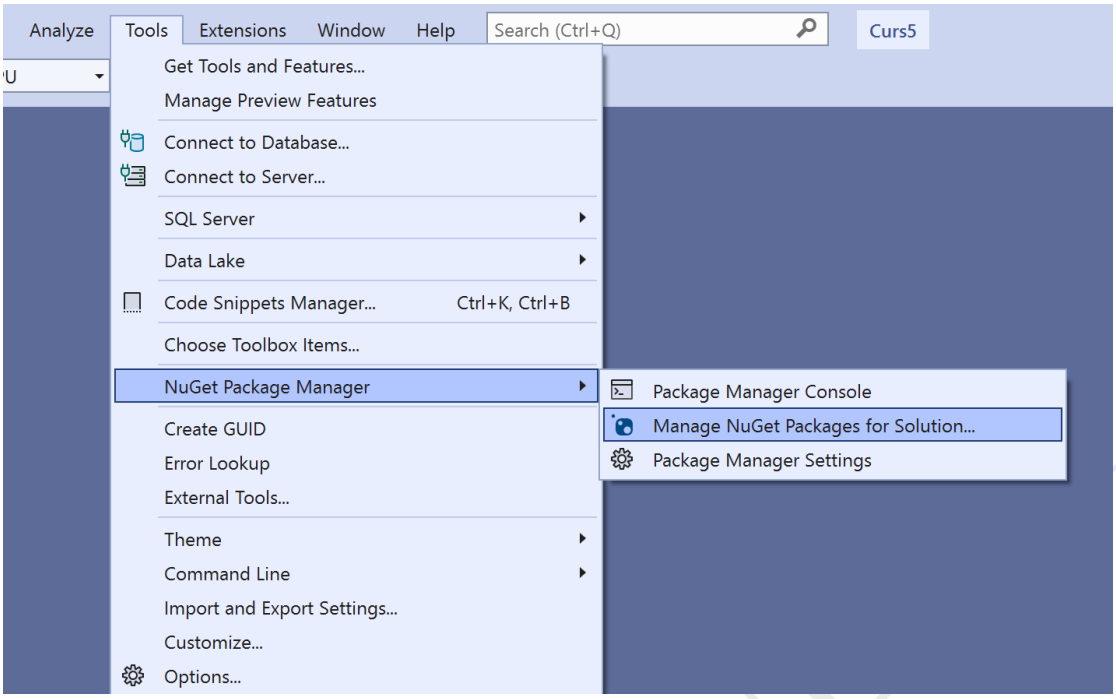
Entity Framework este un pachet care poate fi instalat folosind **NuGet** si care suporta tehnica **code-first**. Tehnica code-first ofera posibilitatea dezvoltatorilor de a scrie clase prin intermediul carora baza de date va fi generata automat. Acest lucru duce la o dezvoltare curata si rapida a aplicatiilor cu baze de date.

Pentru instalarea Entity Framework (EF) se procedeaza astfel:

PASUL 1:

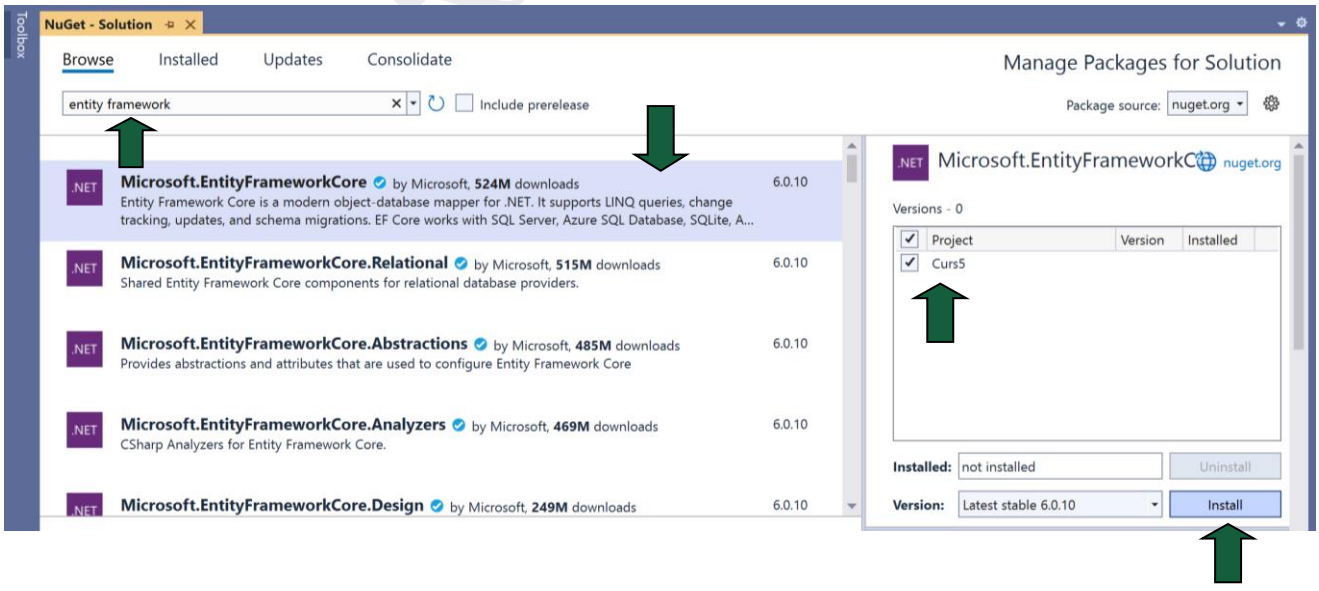
Tools → NuGet Package Manager → Manage NuGet Packages for Solution…

CEZARA BENEGUI

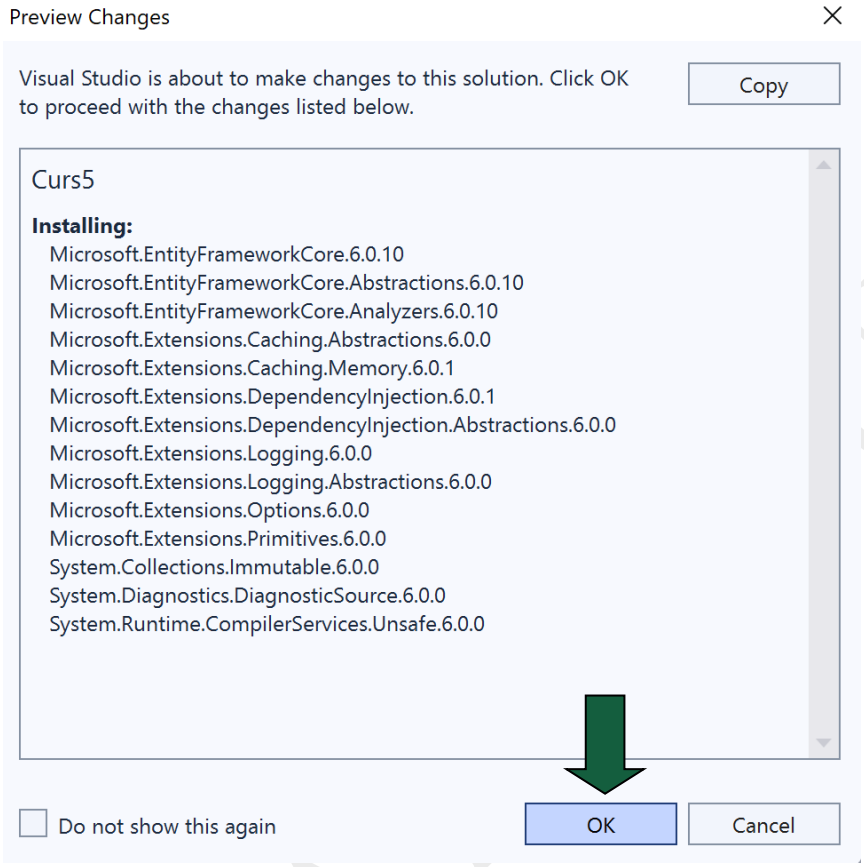


PASUL 2:

Se acceseaza optiunea **Browse** si se cauta **Microsoft.EntityFrameworkCore** dupa cum urmeaza:

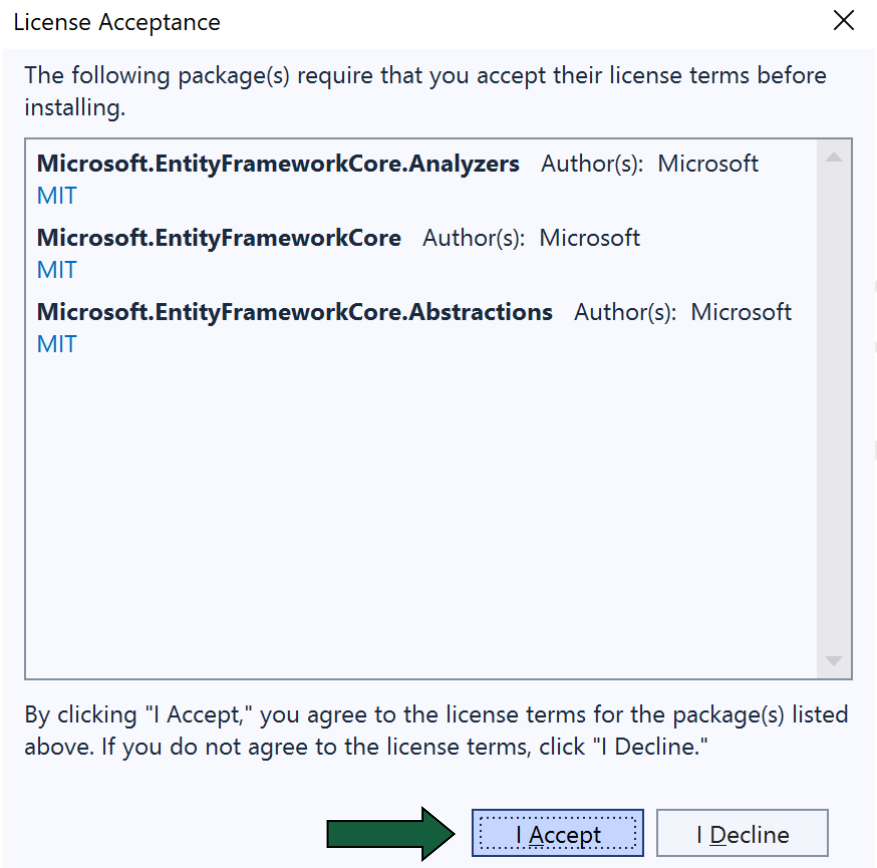


PASUL 3:



CEZARA BENEGUI

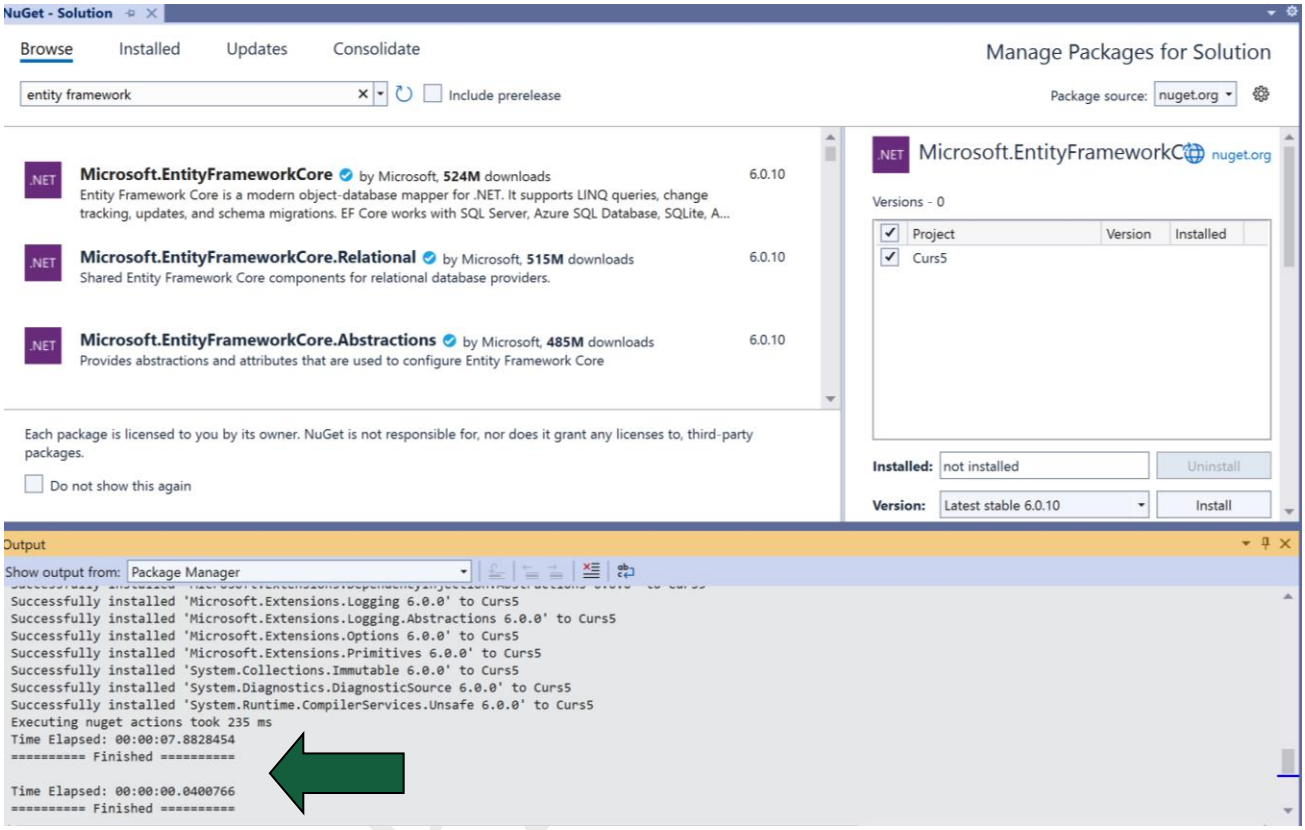
PASUL 4:



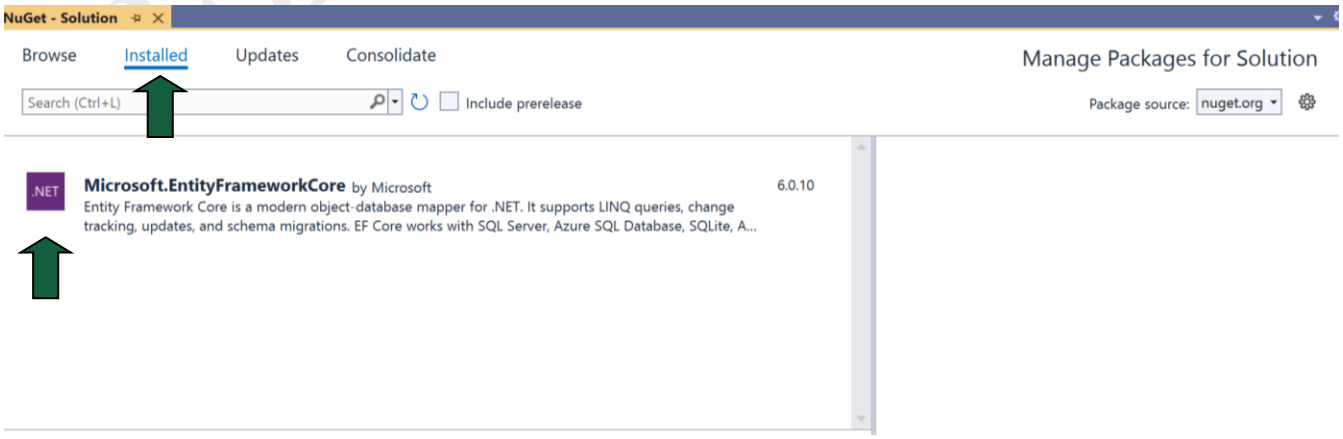
CEZARA BENEGUI

PASUL 5:

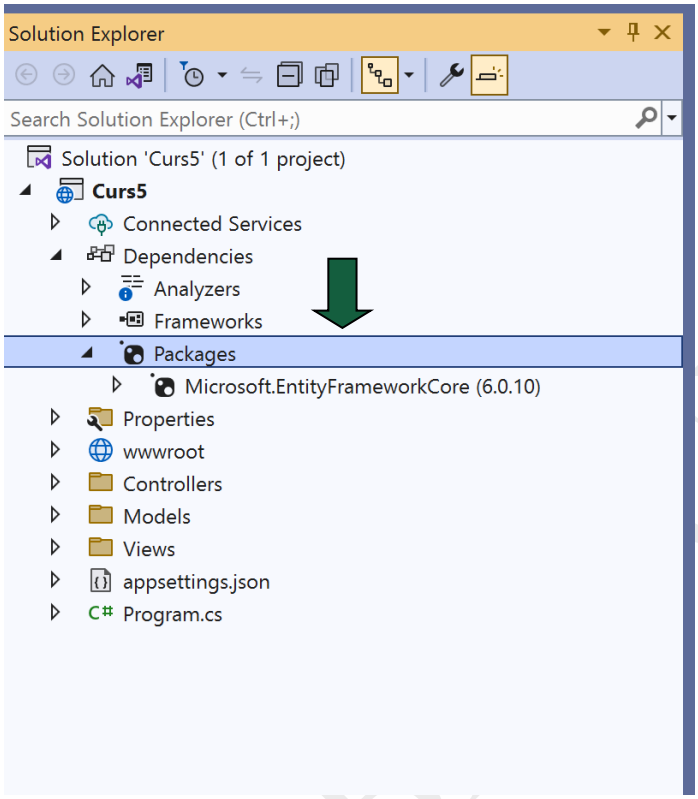
Dupa instalare se poate verifica in consola daca EF a fost adaugat cu succes in cadrul proiectului.



De asemenea, se poate verifica si in sectiunea **Installed**.



Sau chiar in Solution Explorer → Dependencies → Packages

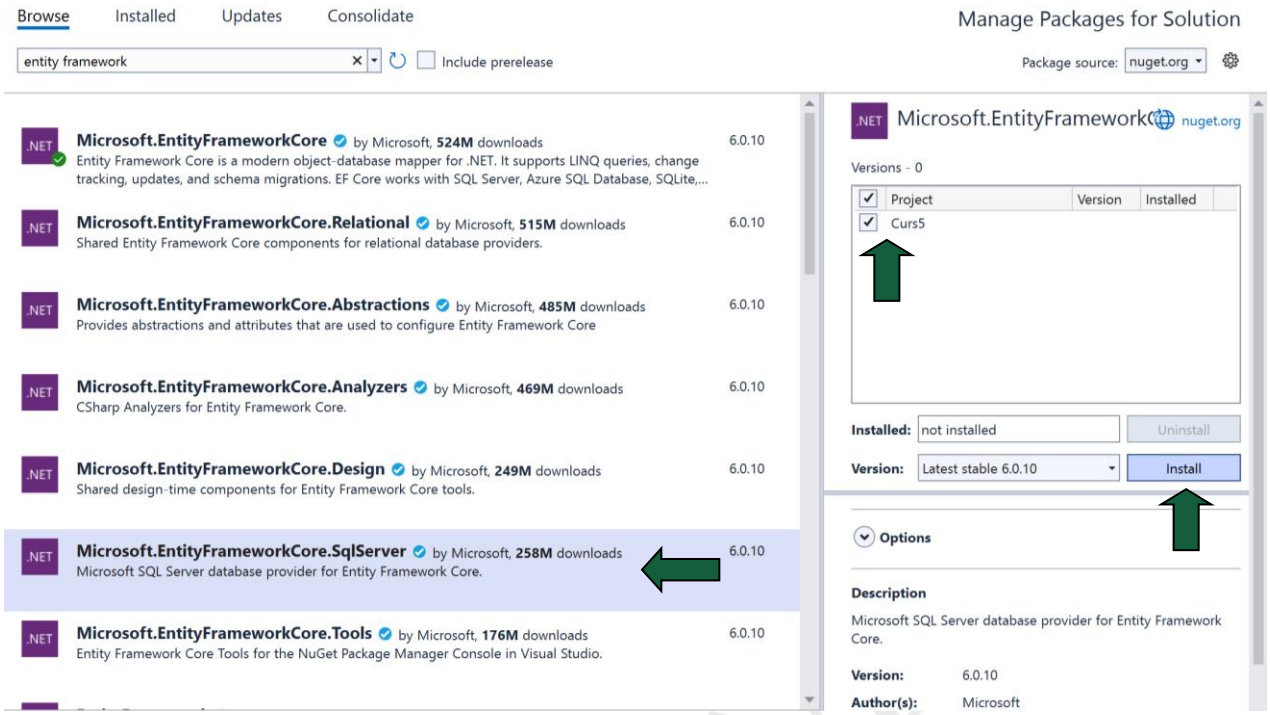


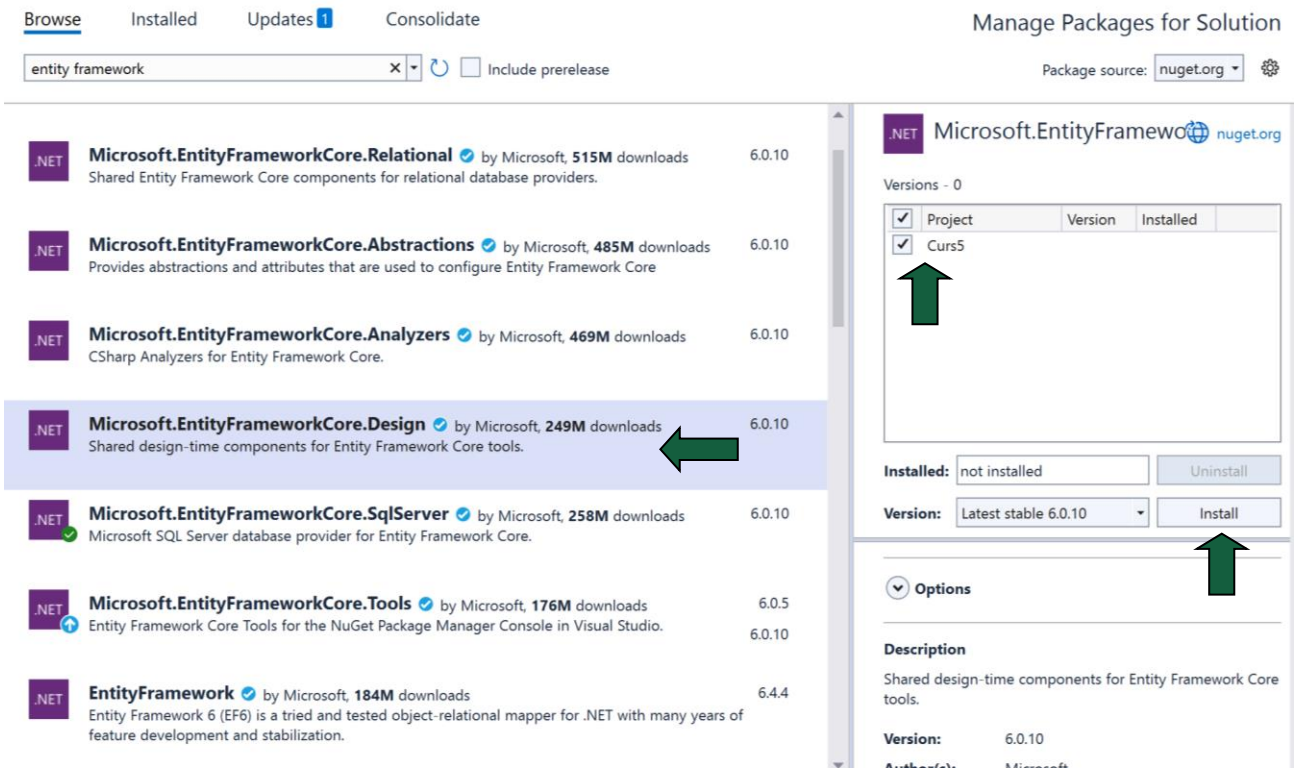
In continuare este necesara includerea pachetului **SqlServer** (pentru baza de date) si **Design** (pachet care contine tool-urile necesare pentru rularea comenzilor de migrare).

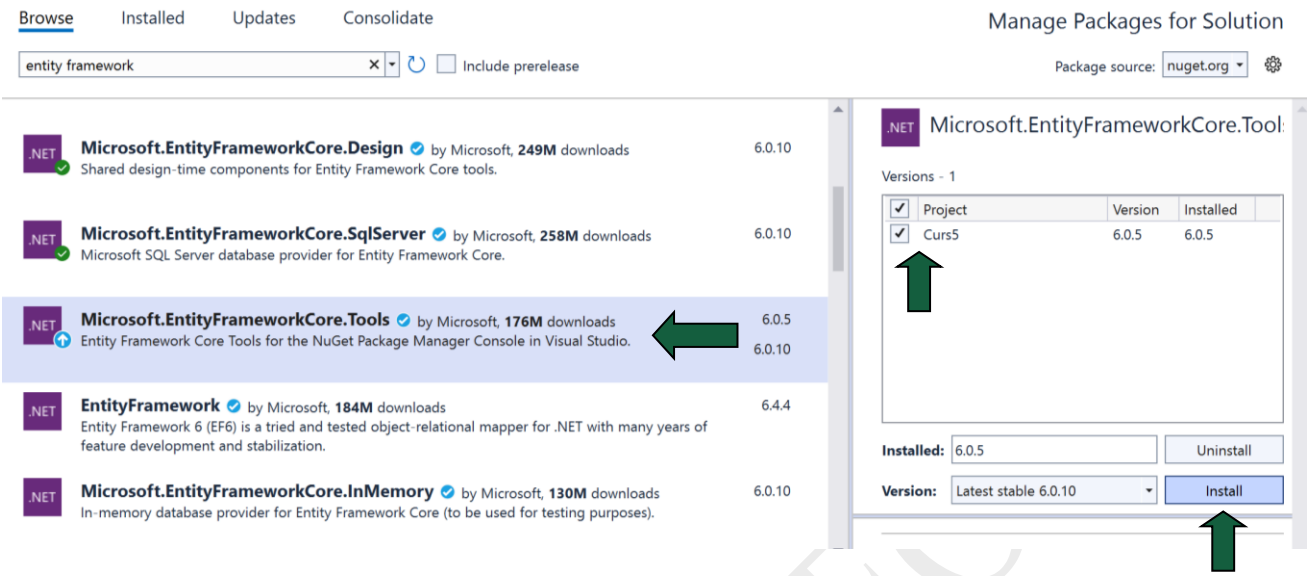
Se instaleaza si pachetul Tools care include comenzile comezi precum: Add-Migration, Drop-Database, Get-DbContext, Remove-Migration , etc.

➔ Microsoft.EntityFrameworkCore.SqlServer
➔ Microsoft.EntityFrameworkCore.Design

CEZARA BENEGUI


Browse
Installed
Updates
Consolidate
Manage Packages for Solution
entity framework
Include prerelease
Package source: nuget.org
Microsoft.EntityFrameworkCore by Microsoft, 524M downloads 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, SQLite,...
Microsoft.EntityFrameworkCore.Relational by Microsoft, 515M downloads 6.0.10
Shared Entity Framework Core components for relational database providers.
Microsoft.EntityFrameworkCore.Abstractions by Microsoft, 485M downloads 6.0.10
Provides abstractions and attributes that are used to configure Entity Framework Core
Microsoft.EntityFrameworkCore.Analyzers by Microsoft, 469M downloads 6.0.10
CSharp Analyzers for Entity Framework Core.
Microsoft.EntityFrameworkCore.Design by Microsoft, 249M downloads 6.0.10
Shared design-time components for Entity Framework Core tools.
Microsoft.EntityFrameworkCore.SqlServer by Microsoft, 258M downloads 6.0.10
Microsoft SQL Server database provider for Entity Framework Core.
Microsoft.EntityFrameworkCore.Tools by Microsoft, 176M downloads 6.0.10
Entity Framework Core Tools for the NuGet Package Manager Console in Visual Studio.
Microsoft.EntityFrameworkCore nuget.org
Versions - 0
Project
Version
Installed
Curs5
Installed: not installed
Uninstall
Version: Latest stable 6.0.10
Install
Options
Description
Microsoft SQL Server database provider for Entity Framework Core.
Version: 6.0.10
Author(s): Microsoft


Browse
Installed
Updates 1
Consolidate
Manage Packages for Solution
entity framework
Include prerelease
Package source: nuget.org
Microsoft.EntityFrameworkCore.Relational by Microsoft, 515M downloads 6.0.10
Shared Entity Framework Core components for relational database providers.
Microsoft.EntityFrameworkCore.Abstractions by Microsoft, 485M downloads 6.0.10
Provides abstractions and attributes that are used to configure Entity Framework Core
Microsoft.EntityFrameworkCore.Analyzers by Microsoft, 469M downloads 6.0.10
CSharp Analyzers for Entity Framework Core.
Microsoft.EntityFrameworkCore.Design by Microsoft, 249M downloads 6.0.10
Shared design-time components for Entity Framework Core tools.
Microsoft.EntityFrameworkCore.SqlServer by Microsoft, 258M downloads 6.0.10
Microsoft SQL Server database provider for Entity Framework Core.
Microsoft.EntityFrameworkCore.Tools by Microsoft, 176M downloads 6.0.5
Entity Framework Core Tools for the NuGet Package Manager Console in Visual Studio. 6.0.10
EntityFramework by Microsoft, 184M downloads 6.4.4
Entity Framework 6 (EF6) is a tried and tested object-relational mapper for .NET with many years of feature development and stabilization.
Microsoft.EntityFramewo nuget.org
Versions - 0
Project
Version
Installed
Curs5
Installed: not installed
Uninstall
Version: Latest stable 6.0.10
Install
Options
Description
Shared design-time components for Entity Framework Core tools.
Version: 6.0.10



Entity Framework Core – Migratii

Ce sunt migratiile?

In timpul dezvoltarii unei aplicatii baza de date se modifica constant, fiind necesare entitati noi, proprietati noi sau chiar eliminarea unor proprietati existente. Pentru a sincroniza aceste modificari cu baza de date existenta, sunt necesare **migratii**.

In momentul in care apare o modificare in baza de date, Entity Framework Core, prin sistemul de migratii, compara modelul curent cu cel anterior pentru a detecta diferentele dintre cele doua versiuni. Ulterior este generat un fisier, continand codul asociat migratiei.

CEZARA BENEGUI

Crearea unui proiect utilizand EF si sistemul de migratii

Crearea proiectului

Se creeaza un nou proiect, procedand la fel ca in cursurile anterioare. Proiectul o sa se numeasca **Curs5**.

Adaugare Entity Framework Core

In cadrul noului proiect se adauga EF (**VEZI** Sectiunea – Instalare Entity Framework Core – din cadrul cursului curent).

Adaugarea Modelului

Pentru adaugarea unui model se porneste de la crearea in cadrul acestuia a tuturor entitatilor. Prin **entitate** ne referim la un tabel din baza de date.

Entitate = un loc, o actiune, o persoana, etc.

Exemplu de entitati dintr-o baza de date care gestioneaza o Universitate → Studenti, Cursuri, Note, MergeLa – tabel asociativ intre Studenti si Cursuri

In continuare vom adauga clasa **Student** cu urmatoarele atribute:

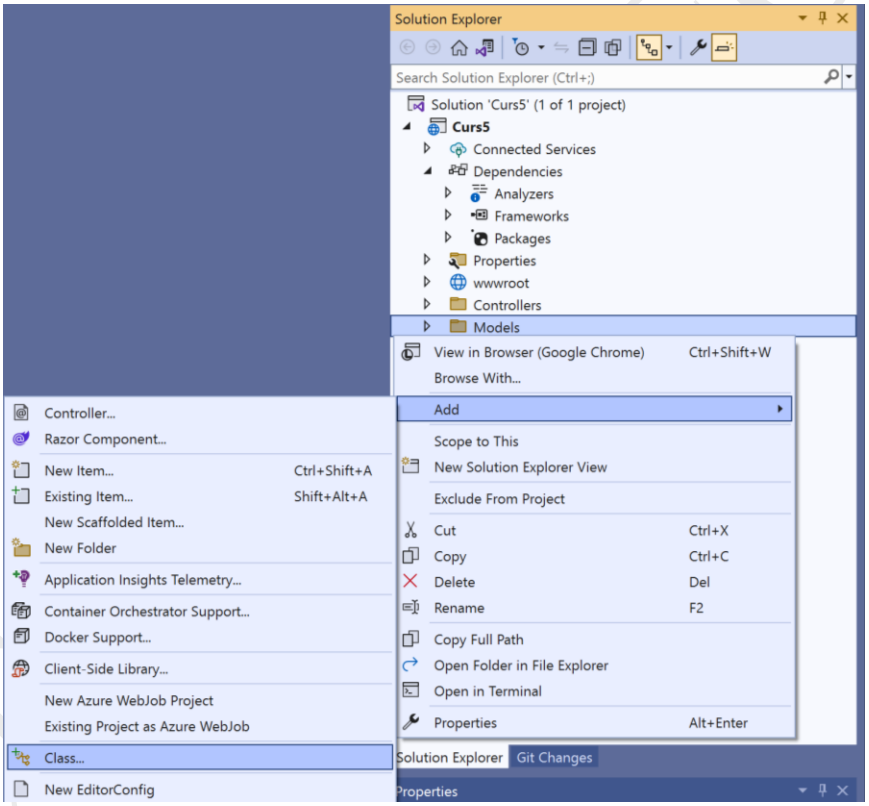
- **StudentID** – de tip int care reprezinta ID-ul studentului
- **Name** – de tip string care reprezinta numele studentului
- **Email** – de tip string care reprezinta o adresa de e-mail a studentului

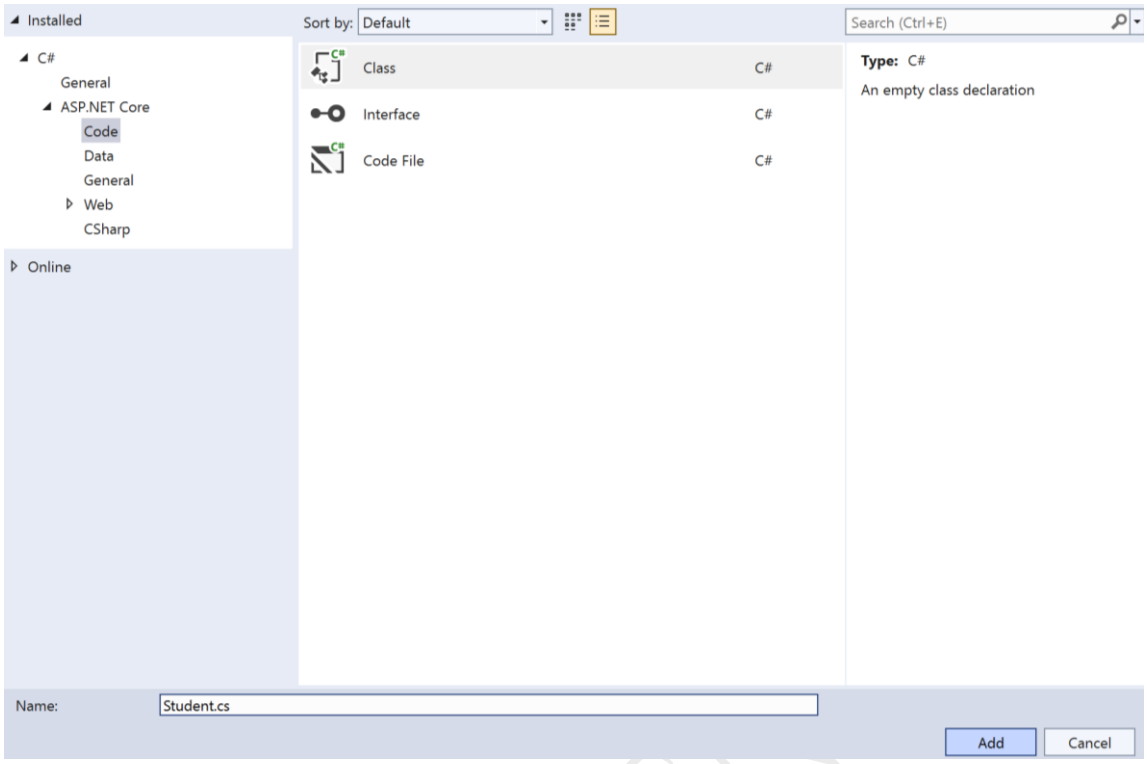
CEZARA BENEGUI

- **CNP** – de tip string care reprezinta CNP-ul studentului. Aceasta proprietate a fost definita de tip string deoarece spatiul alocat pentru int nu suporta valori de 13 caractere. De asemenea, definit ca string se poate procesa caracter cu caracter pentru calcule ulterioare (ex: extragere data de nastere).

Pentru adaugarea clasei **Student** se parcurg urmatorii pasi:

Click dreapta Model → Add → Class → ASP.NET Core → Class → se completeaza numele clasei Student.cs





Clasa **Student** din fisierul Student.cs:

```
public class Student

{
    public int StudentID { get; set; }
    public string Name { get; set; }
    public string Email { get; set; }
    public string CNP { get; set; }

}
```

Aceasta clasa reprezinta tabelul **Student** din baza de date, pe care in varianta clasica il adaugam prin intermediul comenzii **CREATE TABLE**.

Atributele / proprietatile clasei sunt: StudentID, Name, Email, CNP. Acestea reprezinta coloanele din tabelul Student, coloane care sunt create in momentul in care se creeaza si tabelul → utilizand comanda **CREATE TABLE**

Conexiunea cu Baza de Date

Pentru a putea adauga layer-ul de conexiune cu baza de date in cadrul unui model, este necesara adaugarea unei noi clase.

Se adauga o clasa, numita sugestiv **AppDbContext**.

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

   }


   protected override void OnConfiguring
(DbContextOptionsBuilder options)
   {
       options.UseSqlServer(
           @"Stringul de Conexiune");
   }

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

Clasa **AppDBContext** mosteneste clasa de baza **DbContext** din Entity Framework. Clasa de baza realizeaza in mod automat conexiunea cu baza de date, crearea tabelului daca acesta nu exista si contine o proprietate **DbSet**, care trebuie sa primeasca tipul modelului (Student in cazul curent) si numele pluralizat al modelului.

DbSet <Student> Students { get; set; } – prin intermediul acestei secvente de cod vom avea acces la intrarile din baza de date; se pot interoga si stoca instante de tip Student.

Stringul de conexiune la baza de date este preluat de parametrul **options** din metoda `OnConfiguring` prin intermediul urmatoarei secvente de cod:

CEZARA BENEGUI

```
options.UseSqlServer(@"Stringul de Conexiune");
```

`UseSqlServer()` → metoda prin intermediul careia se configureaza contextul pentru conectarea la baza de date. Primeste ca argument stringul de conectare la baza de date. Stringul de conectare se obtine urmand sectiunea urmatoare din curs.

In final se adauga in Program.cs urmatoarea secventa de cod pentru initializarea bazei de date.

```
builder.Services.AddDbContext<AppDBContext>();
```

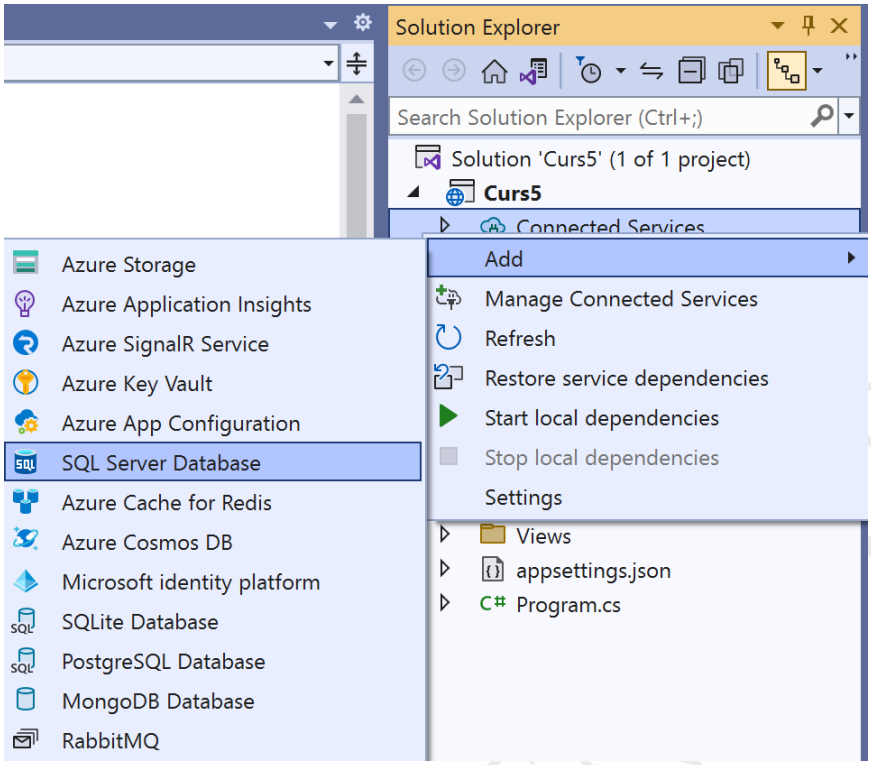
Adaugarea unei baze de date SQL Server

Pentru adaugarea baze de date se parcurg urmatorii pasi:

PASUL 1:

In Solution Explorer → click dreapta pe sectiunea Connected Services → Add → SQL Server Database

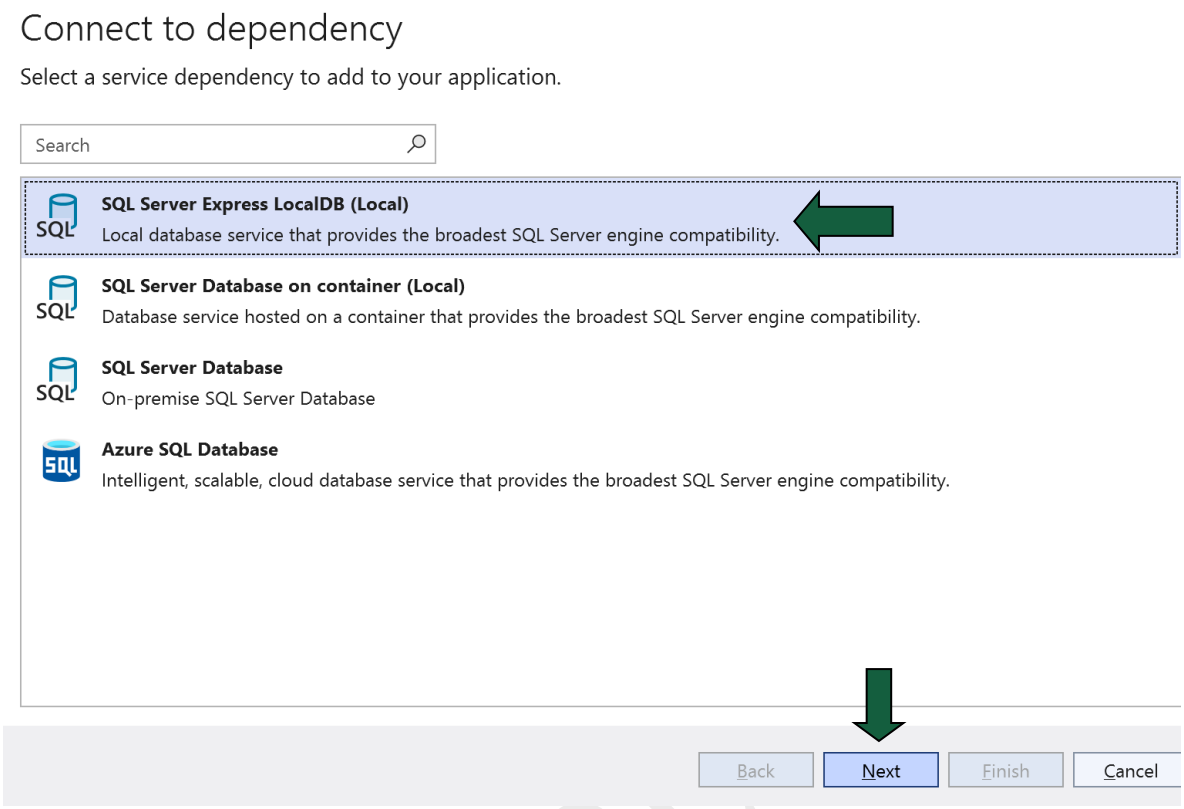
CEZARA BENEGUI



PASUL 2:

Se selecteaza optiunea SQL Server Espress LocalDB (Local)

CEZARA BENEGUI



PASUL 3:

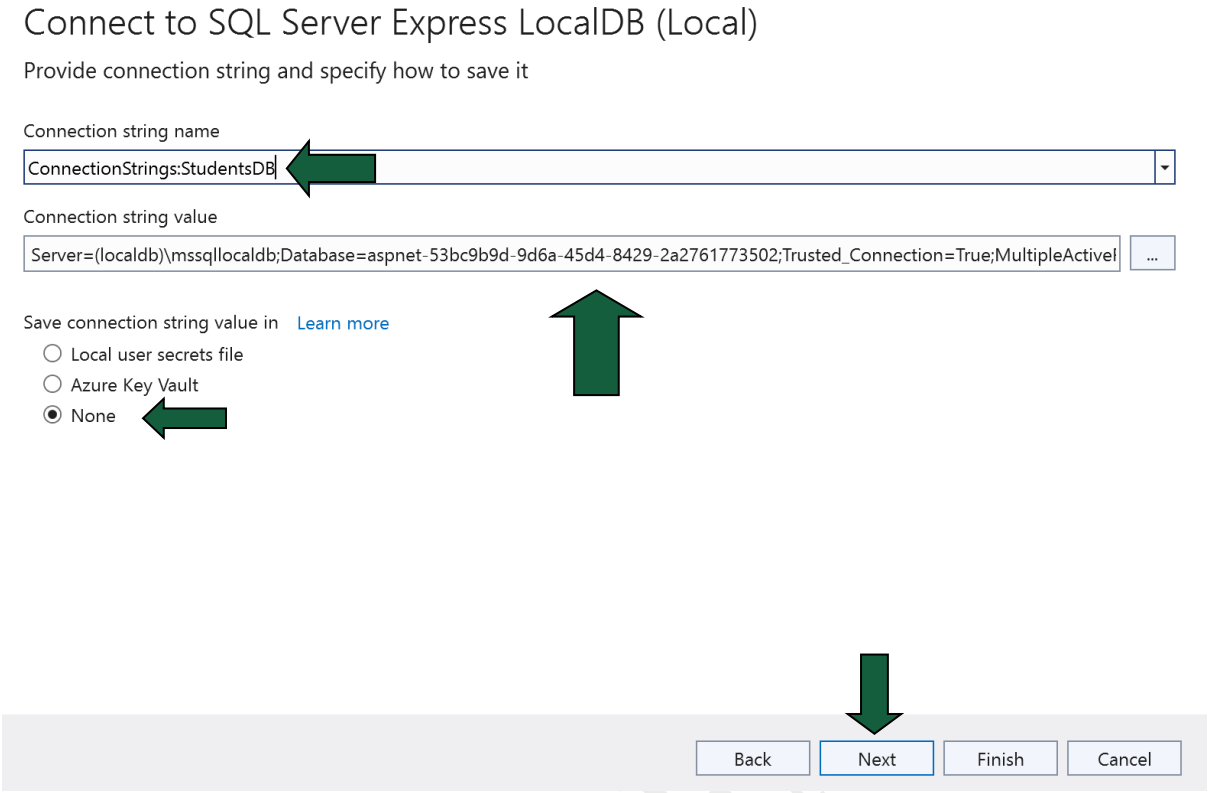
Se adauga un nume conexiunii la baza de date (ex: StudentsDB).

Se bifeaza optiunea **None**.

Se copiaza stringul de conexiune aflat in casuta Connection string value.

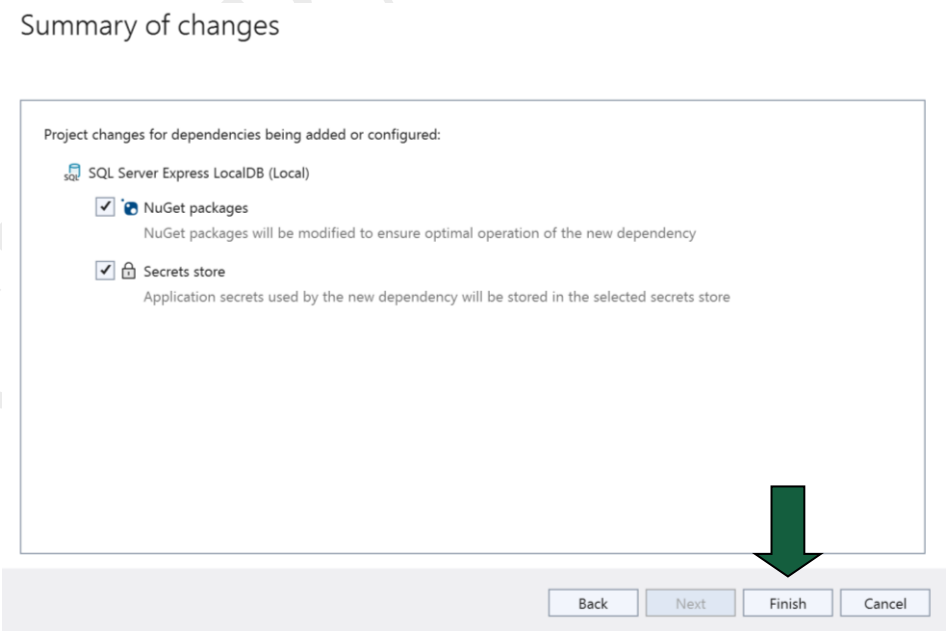
Se apasa butonul Next.

CEZARA BE


Connect to SQL Server Express LocalDB (Local)
Provide connection string and specify how to save it
Connection string name
ConnectionStrings:StudentsDB
Connection string value
Server=(localdb)\mssqllocaldb;Database=aspnet-53bc9b9d-9d6a-45d4-8429-2a2761773502;Trusted_Connection=True;MultipleActiveR
...
Save connection string value in
Learn more
Local user secrets file
Azure Key Vault
None

Back
Next
Finish
Cancel

PASUL 4:


Summary of changes
Project changes for dependencies being added or configured:
SQL Server Express LocalDB (Local)
NuGet packages
NuGet packages will be modified to ensure optimal operation of the new dependency
Secrets store
Application secrets used by the new dependency will be stored in the selected secrets store
Back
Next
Finish
Cancel

PASUL 5:

Se adauga stringul de conectare la baza de date ca parametru al metodei `UseSqlServer()` in `OnConfiguring`

```
options.UseSqlServer(@"Server=(localdb)\mssqllocaldb;Database=aspnet-53bc9b9d-9d6a-45d4-8429
2a2761773502;Trusted_Connection=True;MultipleActiveResultSets=true");
```

/!\ OBSERVATIE

Stringul dat ca parametru in metoda de mai sus UseSqlServer este unic in cadrul fiecarei baze de date. Asadar, nu trebuie sa utilizati stringul din acest exemplu, ci trebuie sa preluati stringul de conexiune la baza voastra de date urmand pasii anteriori.

In acest moment proiectul contine Entity Framework si are creata o baza de date si o conexiune cu aceasta. **Urmeaza sa integram sistemul de migratii.**

CEZARA BENEGUI

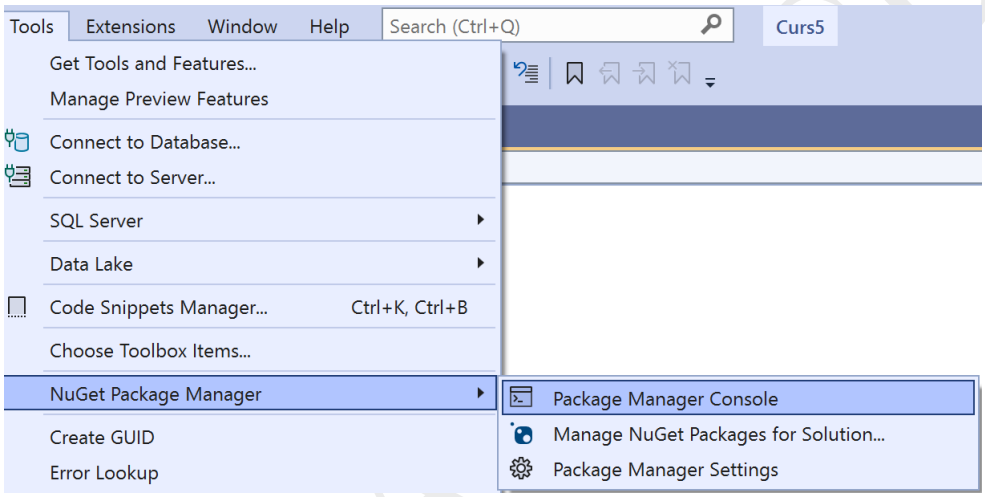
Crearea migratiilor in baza de date

Pentru integrarea sistemului de migratii se parcurg pasii urmatori:

PASUL 1:

Pentru adaugarea migratiilor o sa se utilizeze consola.

Tools → NuGet Package Manager → Package Manager Console



PASUL 2:

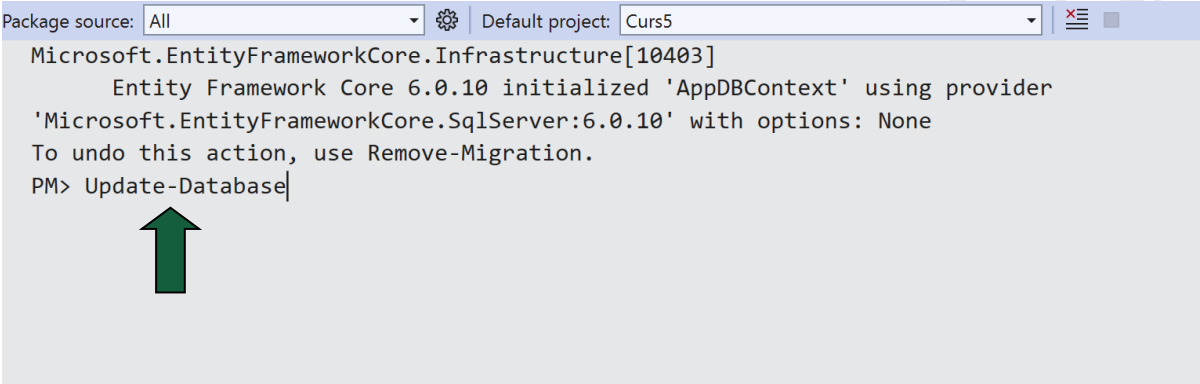
Se adauga migratia utilizand comanda `Add-Migration` urmata de o denumire pe care o dam acestei migratii.

```
Each package is licensed to you by its owner. NuGet is not responsible for, nor does it grant any licenses to, third-party
packages. Some packages may include dependencies which are governed by additional licenses. Follow the package source
(feed) URL to determine any dependencies.

Package Manager Console Host Version 6.3.0.131

Type 'get-help NuGet' to see all available NuGet commands.

PM> Add-Migration CreateStudent
```

Dupa executarea migratiei, in Solution Explorer o sa se creeze un folder numit **Migrations** unde o sa se genereze fisierele specifice.

PASUL 3:

Se executa comanda `Update-Database` care modifica baza de date, aducand-o la versiunea finala.



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>
```

A BENEGUI

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

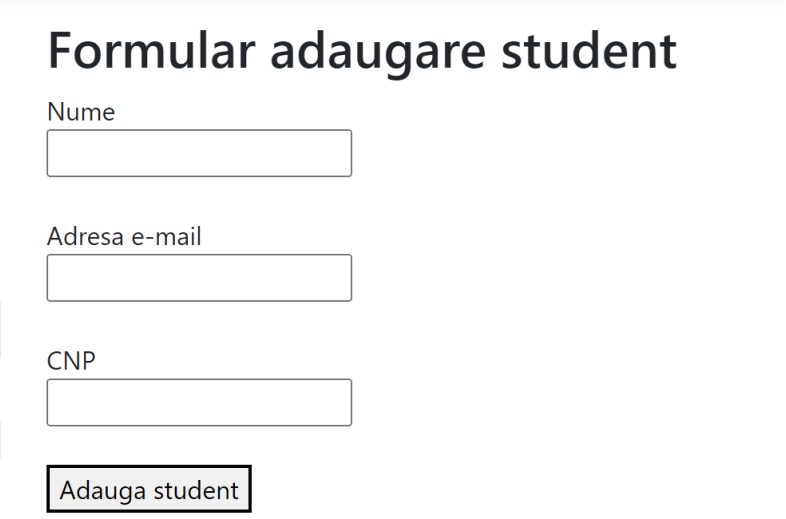
```
public IActionResult New()
   {
       return View();
   }

   [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

CEZARA BENEGUI

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>
```



CEZARA BENEGUI

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.

CEZARA BENEGUI

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 />
```

CEZARA BENEGUI

```
    <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`