

Universitatea din Bucureşti
Facultatea de Matematică și Informatică

Dezvoltarea aplicațiilor web

CTI Anul IV, Semestrul 1 2022-2023

Introducere

Lect. dr. Petru Soviany

petru.soviany@fmi.unibuc.ro

Curs: 2 ore / săptămână

Laborator: 3 ore / săptămână

Subiecte

- Web. Recapitulare HTML, CSS
- C#
- .Net Core
- Baze de date
- API
- Typescript
- Angular 2+

Evaluare

50% - proiect de laborator (minimum 5)

50% - examen la calculator (minimum 5)

Proiect de laborator

- Echipe de 1-3 studenți
- Temă aleasă de comun acord
- Tehnologiile învățate la curs / laborator
- Prezentarea după vacanța de iarnă până la sfârșitul semestrului

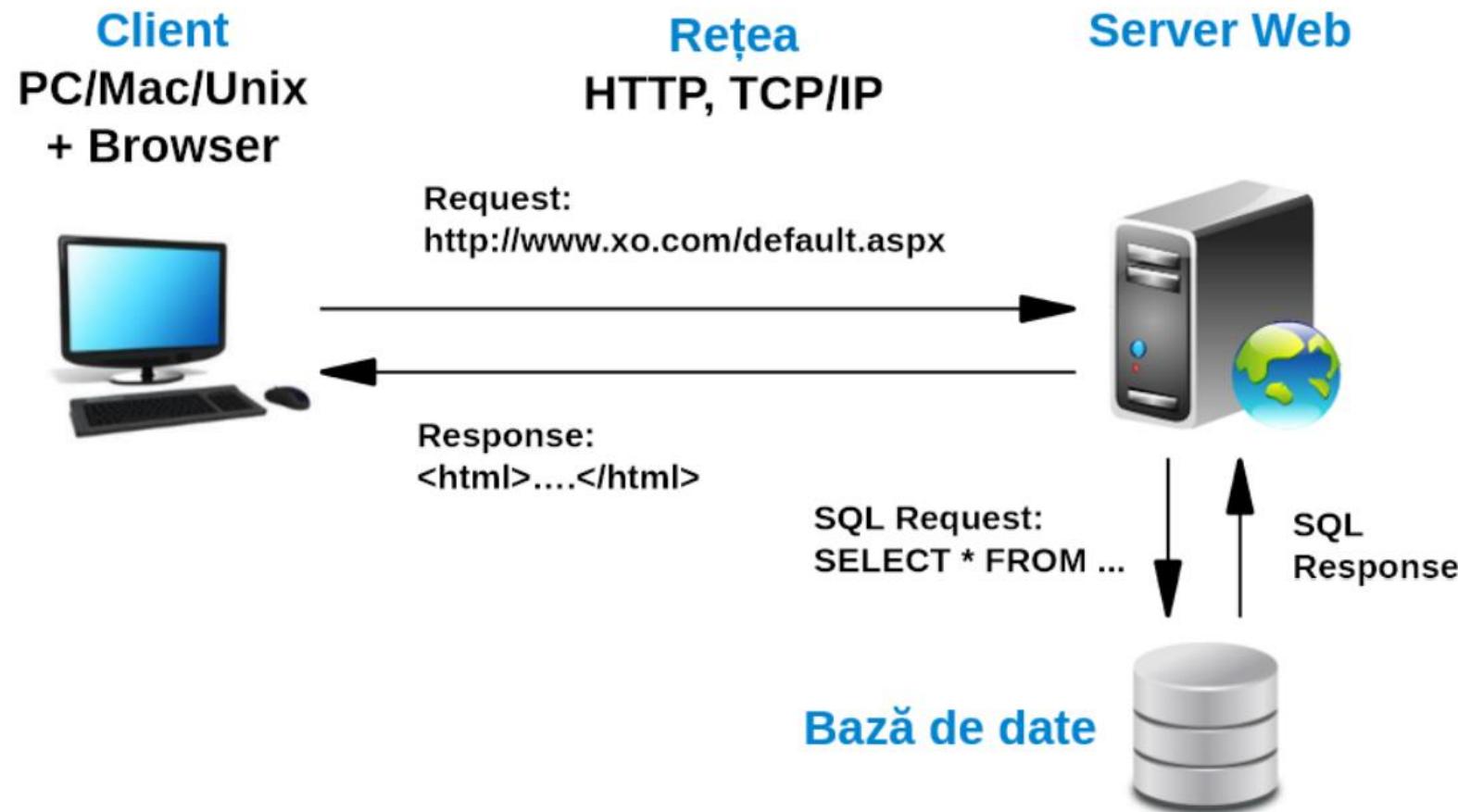
Examen

- Test la calculator în ultimul laborator din semestrul
- 1 ora și 20 de minute, din materia de curs / laborator
- Materiale pe suport fizic sunt permise, fără acces la internet

Aplicație Web

- Client – Server prin URL (uniform resource locator)
- Protocol HTTP
- Limbaj HTML

Arhitectura aplicațiilor web



HTTP

- Request – Response (Solicitare – Răspuns)
- Stateless (nu reține informațiile anterioare)
- Textual (comunicațiile sunt de tip text)

Request

- Solicitare făcută de client la server
- Structură:
 - Request Line: Metodă, Cale, Versiune HTTP
 - Headere: Content type, Content length, Authorization
 - Conținut: HTML, CSS, Javascript

Metode (acțiuni)

- GET (cere resursa)
 - POST (creează resursa)
 - PUT (actualizează resursa)
 - PATCH (actualizează parțial resursa)
 - DELETE (elimină resursa)
- ...și altele: <https://developer.mozilla.org/en-US/docs/Web/HTTP/Methods>

Headere

- Authorization (userul are acces la o anumită resursă?)
 - Content-Type (tipul de date primit)
 - Accept (tipul de date așteptat de la server)
 - Cookie (informații suplimentare despre user)
 - Access-Control-Allow-Origin (CORS - din ce origine pot fi accesate resursele:
<https://developer.mozilla.org/en-US/docs/Web/HTTP/CORS>)
- ...și altele: <https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers>

Exemplu request

```
curl https://reqres.in/api/users?page=2 -v
```

<https://reqres.in/>

Răspuns

- Status code
- Headere
- Conținut

Status codes

- Răspunsuri informative (100–199)
- Răspunsuri de succes (200–299)
- Mesaje de redirectare (300–399)
- Erori la nivel de client (400–499)
- Erori la nivel de server (500–599)

Detalii: <https://developer.mozilla.org/en-US/docs/Web/HTTP/Status>

REST API

- API (Application Programming Interface) – permit comunicarea între componente software diferite
- REST (Representational State Transfer) API – bazate pe metodele HTTP, facilitează manipularea datelor

...mai multe într-un curs viitor.

HTML

- Limbaj de marcat
- Standard pentru pagini web
- Structura și semantica

HTML

```
<!DOCTYPE html>
<html>
    <head>
        <title>Page Title</title>
    </head>
    <body>
        <h1>This is a Heading</h1>
        <p>This is a paragraph.</p>
    </body>
</html>
```

HTML

Pentru reamintire:

[https://developer.mozilla.org/en-US/docs/Learn/Getting started with the web/HTML basics](https://developer.mozilla.org/en-US/docs/Learn/Getting_started_with_the_web/HTML_basics)

HTML

Limbaj semantic: folosim tagurile pentru ce reprezintă, nu pentru aspect

- Motoare de căutare și SEO
- Cititoare de ecrane
- Dezvoltare

...mai multe: https://www.w3schools.com/html/html5_semantic_elements.asp

CSS

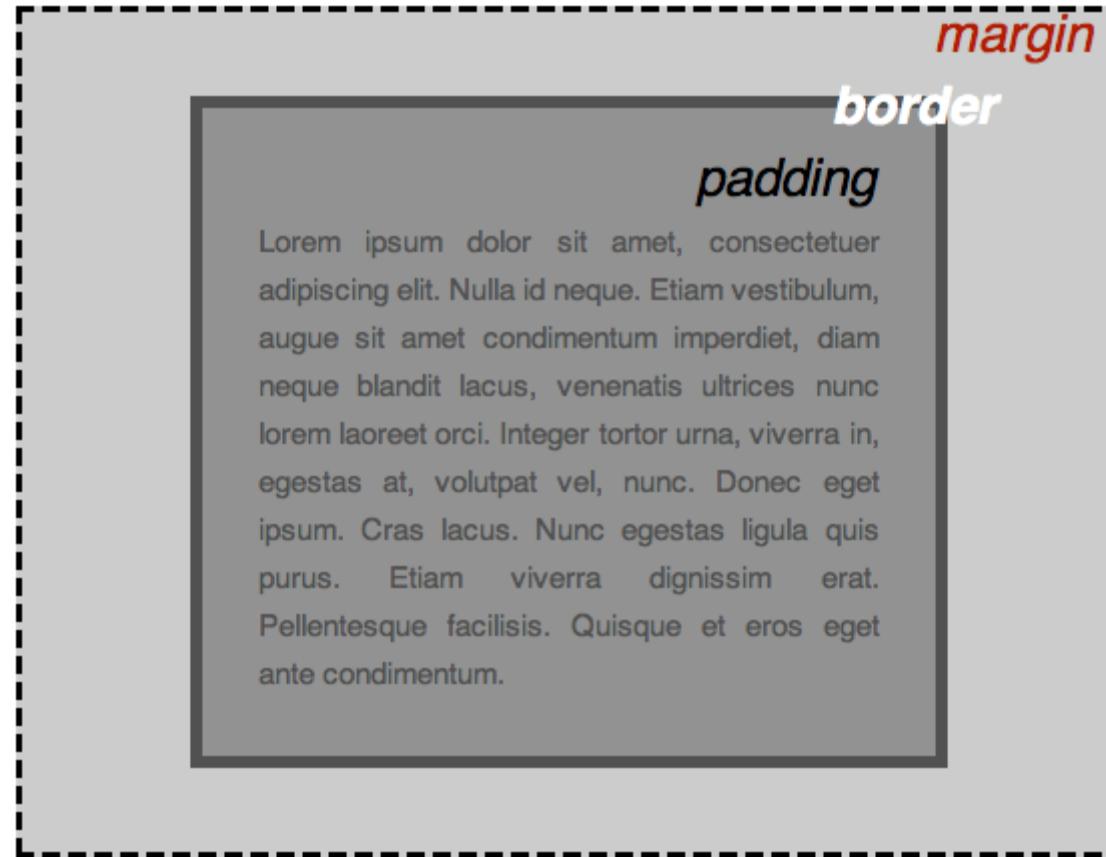
- style sheet language
- selectori: taguri HTML, clase, id-uri

https://www.w3schools.com/cssref/css_selectors.asp

- fișier extern → <style> → inline

CSS

Box model



Sursa: Mozilla Developer

CSS

Display:

Inline

Inline-block

Block

[https://www.samanthaming.com/pictorials/css-inline-vs-
inlineblock-vs-block/](https://www.samanthaming.com/pictorials/css-inline-vs-inlineblock-vs-block/)

CSS

Display:

Flex

<https://css-tricks.com/snippets/css/a-guide-to-flexbox/>

CSS

Display:

None – nu ocupă spațiu (\neq visibility: hidden)

CSS

Position:

Static

Relative

Fixed

Absolute

Sticky

...mai multe: https://www.w3schools.com/css/css_positioning.asp

Javascript

Limbaj de programare

Weakly typed

Var/let/const

Tipuri de date:

- Primitive (valoare): number, string, boolean
- Referință: Object

Mai multe într-un curs viitor...

Universitatea din Bucureşti
Facultatea de Matematică și Informatică

Dezvoltarea aplicațiilor web

CTI Anul IV, Semestrul 1 2022-2023

Curs 2 – 11.10.2022

C#

- Limbaj de programare
- Strongly typed
- OOP
- Limbaj compilat
- Limbaj de nivel înalt
- Management automat al memoriei

Tipuri de date

- Predefinite
- Custom

Tipuri de date predefinite

- bool System.Boolean
- char System.Char
- double System.Double
- float System.Single
- int System.Int32

Tipuri valoare

Tipuri de date predefinite

- object System.Object
- string System.String
- dynamic System.Object

Tipuri referință

Tipuri de date custom

- Struct
- Class
- Interface
- Enum
- Record

Declararea variabilitelor

tip nume_variabila[= initializare];

var nume_variabila = initializare;

Arrays

```
Tip nume[] = new Tip[dim];
```

Arrays

```
double nume[] = new double[3];
```

```
double nume[] = new double[3] = [1,2,3];
```

```
double nume[] = new double[] = [1,2,3];
```

```
double nume[] = new[] = [1,2,3];
```

Liste

```
List<Tip> nume = new List<Tip>();
```

Liste

```
List<double> nume = new List<double>();  
nume.Add(1);  
nume[0];
```

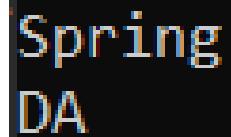
Enum

```
public enum Nume[: tip]
{
    cheie1[= valoare1],
    cheie2[= valoare2],
    ...
}
```

Enum

```
Console.WriteLine(Season.Spring);
if (Season.Spring == 0)
{
    Console.WriteLine("DA");
}
```

```
public enum Season: int
{
    Spring,
    Summer,
    Autumn,
    Winter
}
```

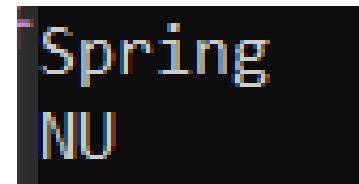
A black terminal window with white text, showing the output of the program. It displays the word "Spring" followed by "DA" on separate lines.

```
Spring
DA
```

Enum

```
Console.WriteLine(Season.Spring);
if (Season.Spring == 0) {
    Console.WriteLine("DA");
} else {
    Console.WriteLine("NU");
}
```

```
public enum Season: int
{
    Spring = 1,
    Summer = 2,
    Autumn = 3,
    Winter = 4
}
```



Struct

```
struct Rectangle {  
    int width;  
    int height;  
    public Rectangle(int width, int height) {  
        this.width = width;  
        this.height = height;  
    }  
  
    public int GetArea()  
    {  
        return this.width * this.height;  
    }  
}
```

Clase

```
class Rectangle {  
    int width;  
    int height;  
    public Rectangle(int width, int height) {  
        this.width = width;  
        this.height = height;  
    }  
  
    public int GetArea()  
    {  
        return this.width * this.height;  
    }  
}
```

```
Rectangle x = new Rectangle(200, 300);  
Console.WriteLine($"Dreptunghiul are suprafata: {x.GetArea()}");
```

Class

```
| Dreptunghiul are suprafata: 60000
```

```
| Dreptunghiul are suprafata: 60000
```

Struct

Struct vs Class

Struct

Nu poți avea constructor fără parametri

Class

Poți avea constructor fără parametri

```
Rectangle x = new Rectangle();  
...  
public Rectangle() {  
    Console.WriteLine("Am apelat constructorul");  
}
```

Class

```
Am apelat constructorul  
Dreptunghiul are suprafata: 0
```

✖ CS0171	Field 'Rectangle.width' must be fully assigned before control is returned to the caller. Consider updating to language version 'preview' to auto-default the field.	ConsoleApp1	Program.cs	8	Active
✖ CS0171	Field 'Rectangle.height' must be fully assigned before control is returned to the caller. Consider updating to language version 'preview' to auto-default the field.	ConsoleApp1	Program.cs	8	Active

Struct

```
int width = 0;
```

```
int height = 0;
```

Class

```
Am apelat constructorul  
Dreptunghiul are suprafata: 0
```

```
Am apelat constructorul  
Dreptunghiul are suprafata: 0
```

Struct

Struct vs Class

Struct

Nu pot avea constructor fără parametri

Începând cu v10, struct poate avea constructori fără parametri. Câmpurile trebuie să aibă o valoare asignată.

Class

Pot avea constructor fără parametri

Struct vs Class

Struct

Nu pot avea constructor fără parametri

Începând cu v10, struct poate avea constructori fără parametri. Câmpurile trebuie să aibă o valoare asignată.

Tip valoare

Class

Pot avea constructor fără parametri

Tip referință

```
Rectangle x = new Rectangle();
x.width = 500;
Rectangle y = x;
y.width = 200;
Console.WriteLine(x.width);
Console.WriteLine(y.width);
```

Class

```
o 200
o 200
```

```
500
200
```

Struct

Struct vs Class

Struct

Nu pot avea constructor fără parametri

Începând cu v10, struct poate avea constructori fără parametri. Câmpurile trebuie să aibă o valoare asignată.

Tip valoare

Nu acceptă moșternire

Class

Pot avea constructor fără parametri

Tip referință

Acceptă moștenire

```
class Square : Rectangle { };  
Rectangle x = new Rectangle();  
Square y = new Square();  
Console.WriteLine(y.GetArea());
```

Class

200

✖ CS0509	'Square': cannot derive from sealed type 'Rectangle' 'Square' does not contain a definition for 'GetArea' and no accessible extension method 'GetArea'	ConsoleApp1	Program.cs	19	Active
✖ CS1061	accepting a first argument of type 'Square' could be found (are you missing a using directive or an assembly reference?)	ConsoleApp1	Program.cs	3	Active

Struct

Modificatori acces

public	Codul este accesibil tuturor claselor
private	Codul este accesibil în aceeași clasă
protected	Codul este accesibil în aceeași clasă sau într-o clasă care moștenește clasa respectivă.

Mai multe... <https://learn.microsoft.com/en-us/dotnet/csharp/programming-guide/classes-and-structs/access-modifiers>

Modificatori acces

```
Rectangle x = new Rectangle();
Console.WriteLine(x.width);
```

3 references

```
class Rectangle {
    int width = 10;
}
```

1 reference

```
class Square : Rectangle {
    Square()
    {
        Console.WriteLine(width);
    }
};
```

Modificatori acces

```
Rectangle x = new Rectangle();
Console.WriteLine(x.width);

3 references
class Rectangle {
    protected int width = 10;
}

1 reference
class Square : Rectangle {
    Square()
    {
        Console.WriteLine(width);
    }
};
```

Modificatori acces

```
Rectangle x = new Rectangle();
Console.WriteLine(x.width);
```

3 references

```
class Rectangle {
    public int width = 10;
}
```

1 reference

```
class Square : Rectangle {
    Square()
    {
        Console.WriteLine(width);
    }
};
```

Operatori

- Arimetici
- De comparație
- Booleeni
- Pe biți
- De egalitate

Majoritatea
operatorilor pot
fi supraîncărcăți

Instructiuni condiționale

```
if (condition) {  
    Statement  
}  
  
else {  
    Statement  
}
```

Instructiuni condiționale

```
switch (expression) {  
    case expression_value1:  
        Statement  
        break;  
    case expression_value2:  
        Statement  
        break;  
    case expression_value3:  
        Statement  
        break;  
    default:  
        Statement  
        break;  
}
```

Operatorul ?

condition ? consequent : alternative

Instructiuni repetitive – for

```
for (s1; c; s2) {  
    // cod  
}
```

s1 – se execută înainte de blocul de cod

c – condiția pentru executarea blocului de cod

s2 – se execută după fiecare execuție a blocului de cod

```
for (int i = 0; i < 5; i++)  
{  
    Console.WriteLine(i);  
}
```

Instructiuni repetitive – while

```
while (c)
{
    // cod
}
```

```
int i = 0;
while (i < 5)
{
    Console.WriteLine(i);
    i++;
}
```

Instructiuni repetitive – do while

```
do {  
    // cod  
} while (c)
```

```
int i = 0;  
do {  
    Console.WriteLine(i);  
    i++;  
} while (i < 5)
```

Instructiuni repetitive – foreach

```
foreach (type variabila in array)
{
    // cod
}
string[] cars = {"Volvo", "BMW", "Ford", "Mazda"};
foreach (string i in cars)
{
    Console.WriteLine(i);
}
```

Universitatea din Bucureşti
Facultatea de Matematică și Informatică

Dezvoltarea aplicațiilor web

CTI Anul IV, Semestrul 1 2022-2023

Curs 3 – 18.10.2022

Tipuri de UI în ASP.NET Core

- La nivel de server: HTML și CSS generate dinamic la nivelul serverului ca răspuns la un request.
 - ✓ Necessită resurse limitate la nivel de client
 - ✓ Acces facil la resurse protejate
 - ✓ SEO
- Costul resurselor este agregat la nivel de server
- Interacțiunile la nivel de client necesită accesarea serverului

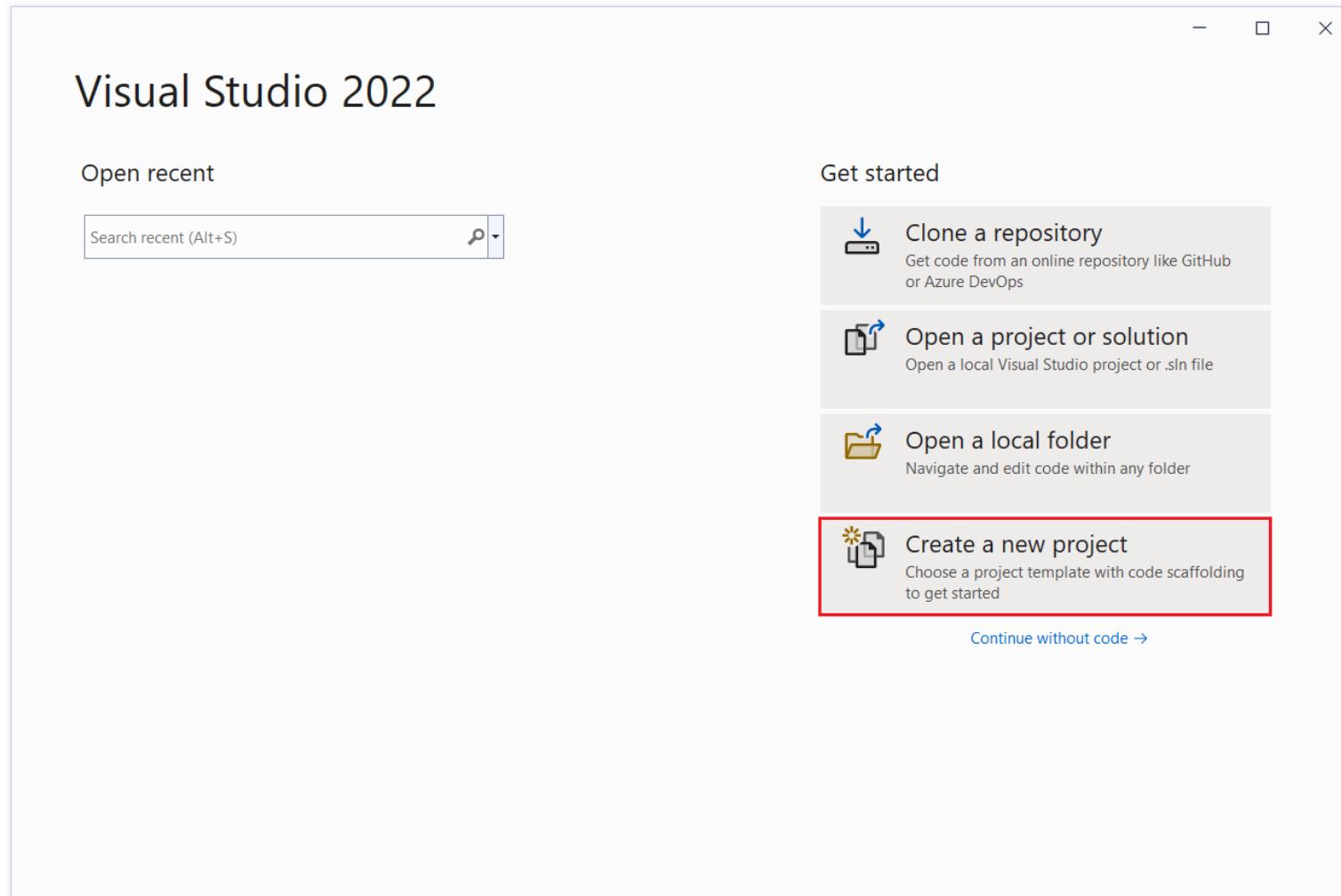
Tipuri de UI în ASP.NET Core

- La nivel de client: UI generat dinamic la nivel de client prin actualizarea directă a DOM.
 - ✓ Interactivitate
 - ✓ Permite salvarea parțială a formularelor
 - ✓ Reduce supraîncarcarea serverului
- Logica este executată la nivel de client
- Utilizatorii de dispozitive low-end vor avea probleme cu utilizarea aplicației

ASP.NET Core Razor Pages

- ASP.NET Core UI
- La nivel de server
- Alternativă la MVC
- Model bazat pe pagini
- UI și logică separate, dar în aceeași pagină

ASP.NET Core Razor Pages



ASP.NET Core Razor Pages

Create a new project

Search for templates (Alt+S)

All languages All platforms All project types

C# Console Application
A project for creating a command-line application that can run on .NET Core on Windows, Linux and macOS
C# Linux macOS Windows Console

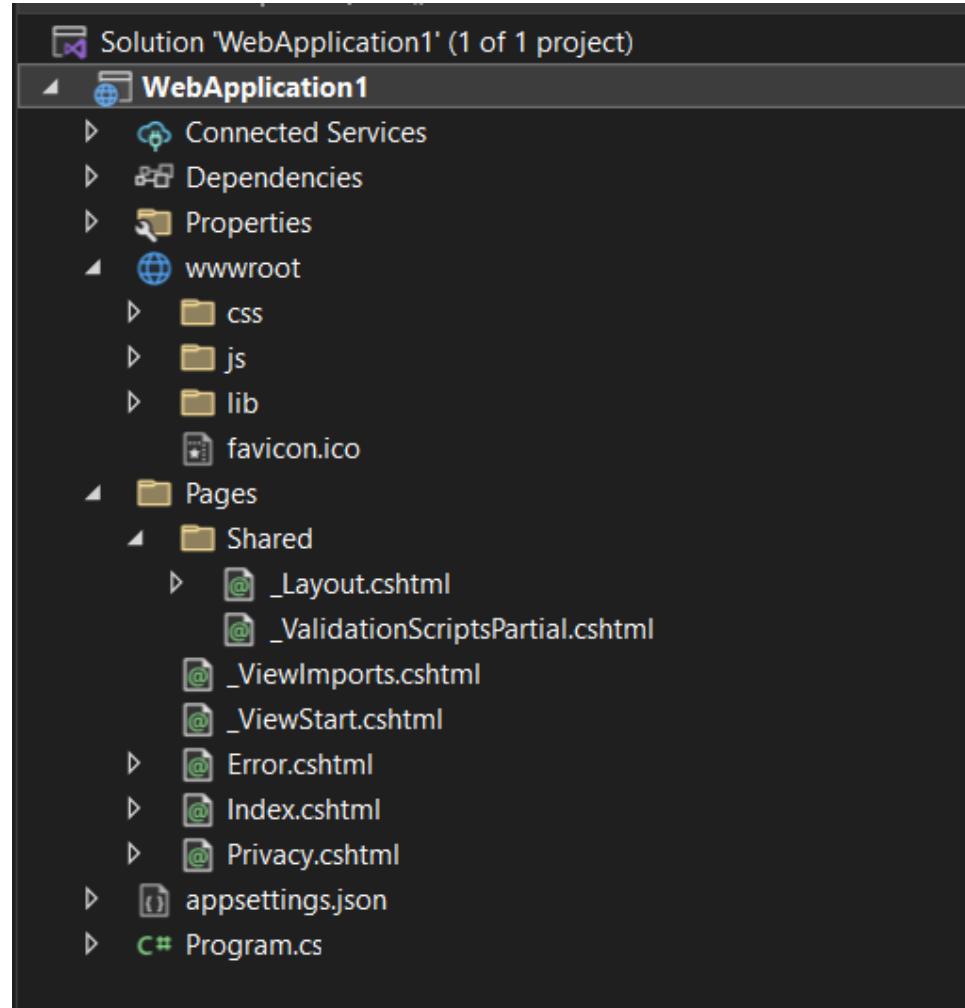
VB Console Application
A project for creating a command-line application that can run on .NET Core on Windows, Linux and macOS
Visual Basic Linux macOS Windows Console

ASP.NET Core Web App
A project template for creating an ASP.NET Core application with example ASP.NET Razor Pages content.
C# Linux macOS Windows Cloud Service Web

Blazor WebAssembly App
A project template for creating a Blazor app that runs on WebAssembly and is optionally hosted by an ASP.NET Core app. This template can be used for web apps with rich dynamic user interfaces (UIs).
C# Linux macOS Windows Cloud Web

Back Next

ASP.NET Core Razor Pages



ASP.NET Core Razor Pages

- Directorul Pages: conține pagini Razor și fișiere suport
- Paginile razor sunt alcătuite din două fișiere:
 - Un fișier .cshtml ce conține HTML și cod C# cu sintaxa Razor
 - Un fișier .cshtml.cs ce conține cod C# și tratează evenimentele din pagină
- Fișerele suport încep cu `__Layout.cshtml` este un template comun tuturor paginilor din aplicație.

ASP.NET Core Razor Pages

- Directorul wwwroot: conține asseturi statice:
 - Fișiere HTML
 - Fișiere CSS
 - Fișiere Javascript
 - Imagini
 - Fonturi

ASP.NET Core Razor Pages

- Fișierul `appsettings.json`: conține informații legate de configurare precum *connection strings*.
- Proprietățiile sunt accesibile din pagini la runtime.

ASP.NET Core Razor Pages

Fișierul Program.cs

Se creează o aplicație nouă cu suport pentru Razor Pages și se construiește aplicația.

```
var builder = WebApplication.CreateBuilder(args);
// Add services to the container.
builder.Services.AddRazorPages();
var app = builder.Build();
```

ASP.NET Core Razor Pages

Fișierul Program.cs

În producție este dezactivat modul de raportare al excepțiilor al developerilor și utilizatorii sunt redirectionați la pagina „Error”. Se activează [HSTS](#).

```
// Configure the HTTP request pipeline.  
if (!app.Environment.IsDevelopment())  
{  
    app.UseExceptionHandler("/Error");  
    app.UseHsts();  
}
```

ASP.NET Core Razor Pages

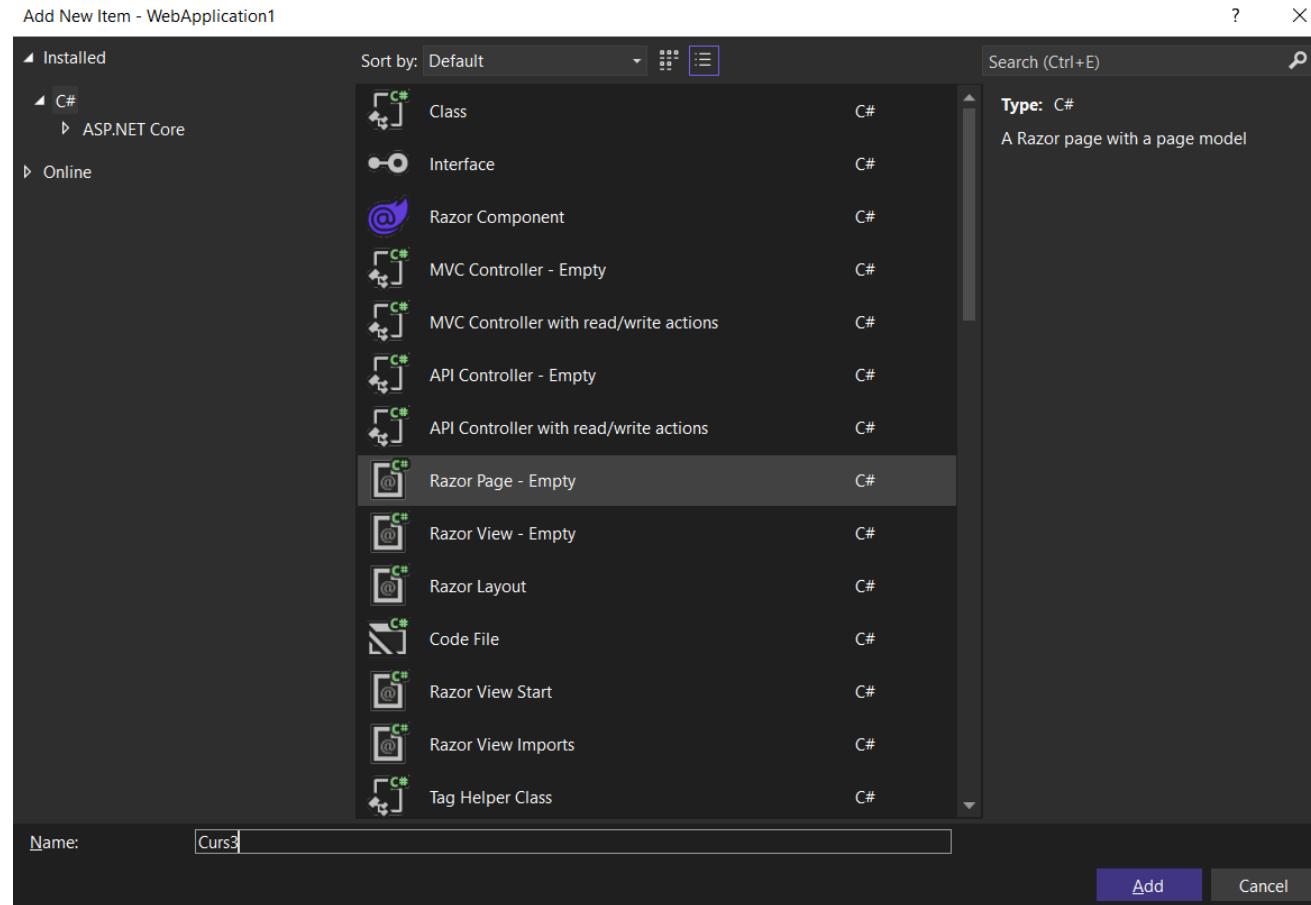
Fișierul Program.cs

- `app.UseHttpsRedirection()` – Redirecționează HTTP la HTTPS.
- `app.UseStaticFiles()` – Activează fișierele statice
- `app.UseRouting()` – Adaugă mecanismul de routing
- `app.MapRazorPages()` – Configurează rutarea endpointurilor pentru Razor Pages.
- `app.UseAuthorization()` – Activează mecanismul de autorizare
- `app.Run()` – Rulează aplicația

ASP.NET Core Razor Pages

- Paginile Razor sunt o combinație de cod HTML și C#.
- Au extensia .cshtml
- Primul element din document este directiva @page

ASP.NET Core Razor Pages



ASP.NET Core Razor Pages

```
@page  
@model WebApplication1.Pages.Curs3Model  
{@  
}
```

```
using Microsoft.AspNetCore.Mvc.RazorPages;  
  
namespace WebApplication1.Pages  
{  
    5 references  
    public class Curs3Model : PageModel  
    {  
        0 references  
        public void OnGet()  
        {  
        }  
    }  
}
```

ASP.NET Core Razor Pages

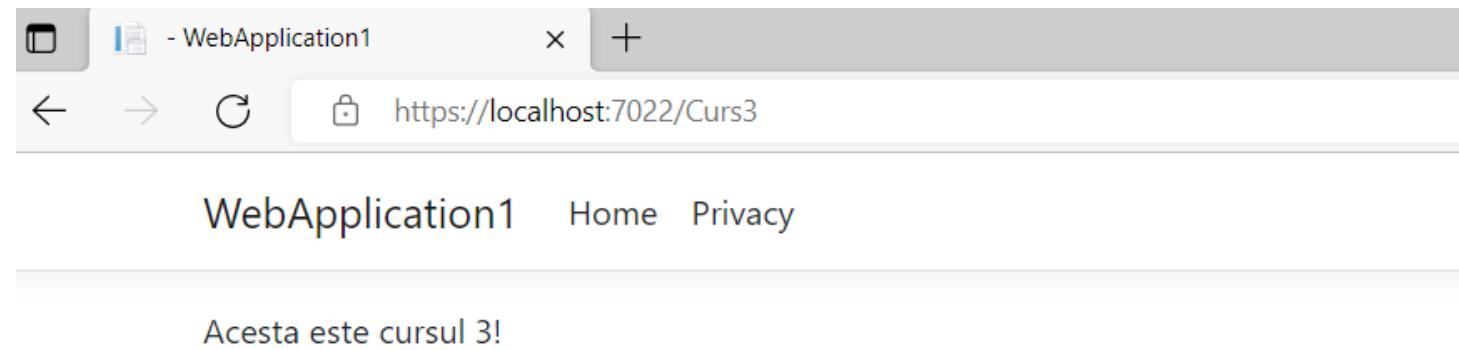
```
@page
@model WebApplication1.Pages.Curs3Model
@{
}

@Model.mesaj
```

```
using Microsoft.AspNetCore.Mvc.RazorPages;

namespace WebApplication1.Pages
{
    5 references
    public class Curs3Model : PageModel
    {
        2 references
        public string? mesaj { get; set; }
        0 references
        public void OnGet()
        {
            mesaj = "Acesta este cursul 3!";
        }
    }
}
```

ASP.NET Core Razor Pages



Tag helpers

- Permit codului la nivel de server să participe la generarea structurii HTML
- Structură mai simplă și suport mai bun decât HTML Helpers

```
<label class="caption" asp-for="FirstName"></label>
```

vs

```
@Html.Label("FirstName", "First Name:", new {@class="caption"})
```

Tag helpers

Anchor Tag Helper

Cache Tag Helper

Component Tag Helper

Distributed Cache Tag Helper

Environment Tag Helper

Form Tag Helper

Form Action Tag Helper

Image Tag Helper

Input Tag Helper

Label Tag Helper

Link Tag Helper

Partial Tag Helper

Script Tag Helper

Select Tag Helper

Textarea Tag Helper

Validation Message Tag Helper

Validation Summary Tag Helper

Tag helpers

Anchor Tag Helper: adaugă funcționalități suplimentare tagului HTML

<a>

```
<a asp-page="/Attendee" asp-route-attendeeid="10">View  
Attendee</a>
```

```
    <a href="/Attendee?attendeeid=10">View Attendee</a>
```

Tag helpers

Input Tag Helper: folosit pentru introducerea datelor într-un formular.

```
<input asp-for="<Expression Name>">
```

- Generează proprietățiile id și name în funcție de valoarea dată în asp-for.
- Setează automat atributul type în funcție de tipul modelului.
- Generează validări HTML5 în funcție de model.

Tag helpers

Partial Tag Helper: permite utilizarea unor view-uri parțiale pentru împărțirea UI-ului în subcomponente.

```
<partial name="_PartialName" />
```

Un view parțial este un template Razor cu extensia .cshtml care nu începe cu directiva @page.

Model și binding

- Proprietățile declarate în model sunt utilizate în view
- Modelul unei pagini Razor este dat de proprietățile din fișierul .cshtml.cs
- Pentru conectarea datelor dintr-un model cu cele din view se utilizează mecanismul de binding.

[BindProperty]

```
public string? mesaj { get; set; }
```

Model și validare

- Validarea se adaugă în model, deasupra proprietății
- Implicit, validarea se efectuează la nivel de client

[MaxLength(5), Required]

[BindProperty]

```
public string? nume { get; set; }
```

ModelState.IsValid

Model și validare

- [ValidateNever]
- [CreditCard]
- [Compare]
- [EmailAddress]
- [Phone]
- [Range]
- [RegularExpression]
- [Required]
- [StringLength]
- [Url]
- [Remote]

Model și validare

```
<form method="post">
    <label asp-for="name">Name:</label>
    <input asp-for="name" />
    <input type="submit" />
    <br />
    <span asp-validation-for="name" class="text-danger"></span>
</form>
@(!String.IsNullOrEmpty(Model.name) ? "Hello, " + Model.name : "Hello, world!")
```

[Required, StringLength(10)]
[BindProperty]
5 references
public string? name { get; set; }

Model și validare

Nume: Submit

The name field is required.

Hello, world!

Nume: Submit

Hello, Curs3

Model și validare

```
▼ <main b-g6ltozs93r role="main" class="pb-3"> == $0
  ▼ <form method="post">
    <label for="name">Nume:</label>
    <input type="text" data-val="true" data-val-length="The field name must be a string with
      a maximum length of 10." data-val-length-max="10" data-val-required="The name field is r
      equired." id="name" maxlength="10" name="name" value>
    <input type="submit">
    <br>
    <span class="text-danger field-validation-valid" data-valmsg-for="name" data-valmsg-
      replace="true"></span>
    <input name="__RequestVerificationToken" type="hidden" value="CfDJ8Av8R6Roz2dCrIG5Sp9wXd
      43o_25VSLxGSbSqnmoX-8APqU6TdGi0VSe-yPofwlBZm9vi-9ceytF-BDq8AWZo2NIeXZU-CR_RFeiyFLmcIWYz_
      7Fi_6aV1uzqSILa1YB99UiZrLY901SwxsVnxrnweQ">
  </form>
  " Hello, world! "
</main>
```

Universitatea din Bucureşti
Facultatea de Matematică și Informatică

Dezvoltarea aplicațiilor web

CTI Anul IV, Semestrul 1 2022-2023

Curs 4 – 25.10.2022

Entity Framework Core

- Entity Framework Core – object-relational mapper (ORM)
- Versiune lightweight, open-source, cross-platform
- Acceptă mai multe tipuri de baze de date

Sql Server

MySql

Oracle

PostgresSQL

[Mai multe...](#)

Entity Framework Core

- Datele sunt accesate folosind un model
- Modelul este compus din clase de entități și un obiect context
- Obiectul context:
 - Sesiune a bazei de date
 - Permite accesarea și salvarea datelor

Entity Framework Core – Model

- Generat dintr-o bază de date existentă
- Dezvoltat manual pentru a se potrivi cu baza de date4
- Se creează întâi modelul, apoi se utilizează migrări pentru a crea / actualiza baza de date

Entity Framework Core – Model

```
public class Blog
{
    0 references
    public int BlogId { get; set; }
    0 references
    public string Url { get; set; }
    0 references
    public int Rating { get; set; }
    0 references
    public List<Post> Posts { get; set; }
}
```

```
public class Post
{
    0 references
    public int PostId { get; set; }
    0 references
    public string Title { get; set; }
    0 references
    public string Content { get; set; }
    0 references
    public int BlogId { get; set; }
    0 references
    public Blog Blog { get; set; }
}
```

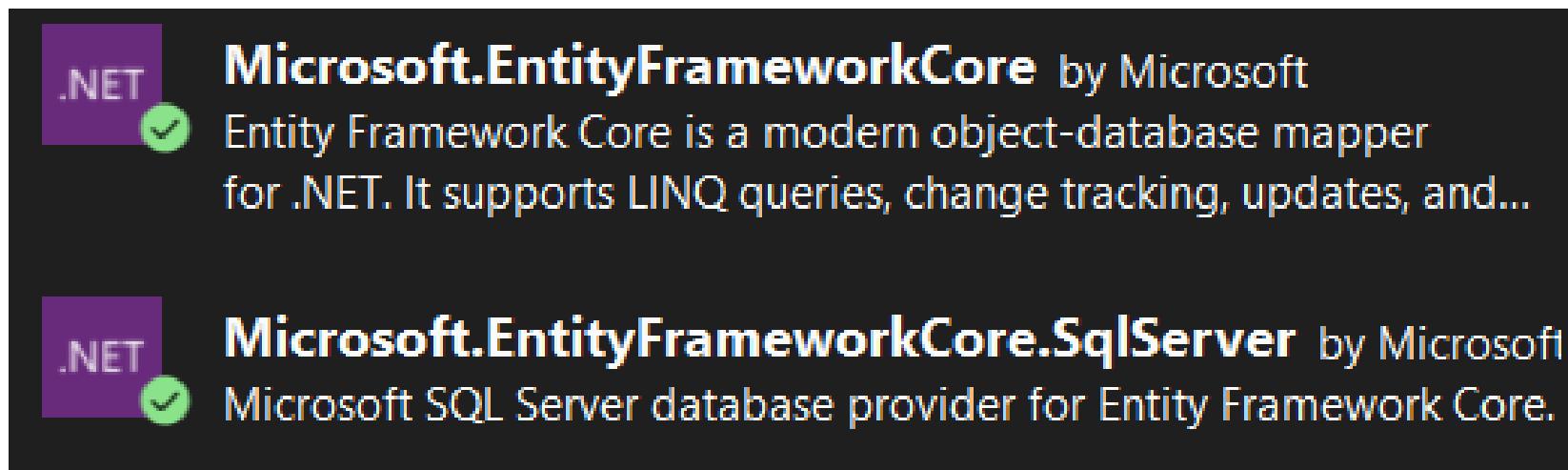
Entity Framework Core – Model

```
public class BloggingContext : DbContext
{
    public DbSet<Blog> Blogs { get; set; }
    public DbSet<Post> Posts { get; set; }
}
```

```
protected override void OnConfiguring(DbContextOptionsBuilder optionsBuilder)
{
    optionsBuilder.UseSqlServer(
        @"Server=(localdb)\mssqllocaldb;Database=Blogging;Trusted_Connection=True");
}
```

Entity Framework Core

- Folosind NuGet package manager, instaläm:



Entity Framework Core

- În Package Manager Console rulăm:

Install-Package Microsoft.EntityFrameworkCore.Tools

Add-Migration InitialCreate

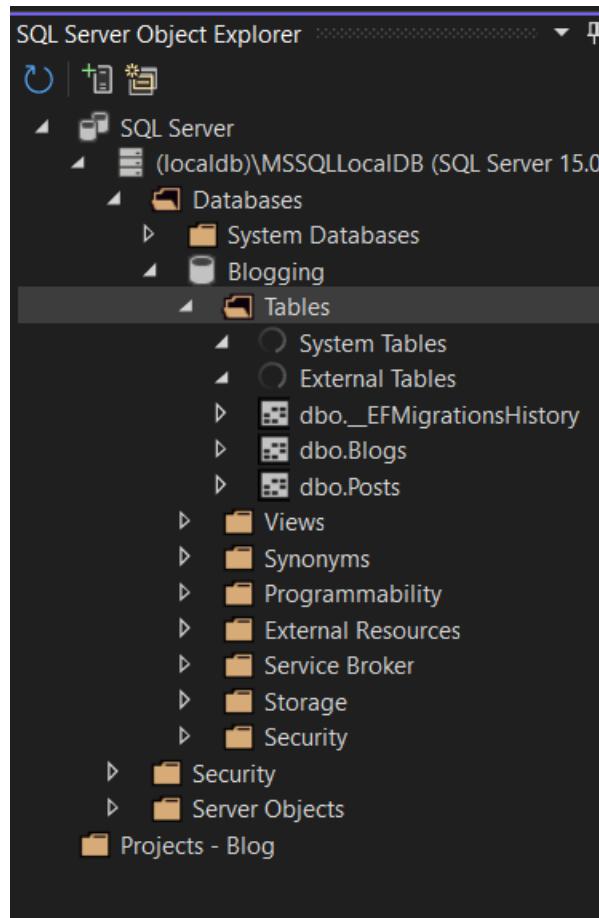
Update-Database

Entity Framework Core

```
namespace Blog.Migrations
{
    public partial class InitialCreate : Migration
    {
        protected override void Up(MigrationBuilder migrationBuilder)
        {
            migrationBuilder.CreateTable(
                name: "Blogs",
                columns: table => new
                {
                    BlogId = table.Column<int>(type: "int", nullable: false)
                        .Annotation("SqlServer:Identity", "1, 1"),
                    Url = table.Column<string>(type: "nvarchar(max)", nullable: true),
                    Rating = table.Column<int>(type: "int", nullable: false)
                },
                constraints: table =>
                {
                    ...
                }
            );
        }

        protected override void Down(MigrationBuilder migrationBuilder)
        {
            migrationBuilder.DropTable(
                name: "Blogs");
        }
    }
}
```

Entity Framework Core



Entity Framework Core

The screenshot shows the Entity Framework Core database design interface. At the top, the title bar displays "dbo.Blogs [Design]" and "Script File: dbo.Blogs.sql". The main area is divided into two sections: a table structure on the left and a list of constraints on the right.

Table Structure:

Name	Data Type	Allow Nulls	Default
BlogId	int	<input checked="" type="checkbox"/>	
Url	nvarchar(MAX)	<input checked="" type="checkbox"/>	
Rating	int	<input checked="" type="checkbox"/>	

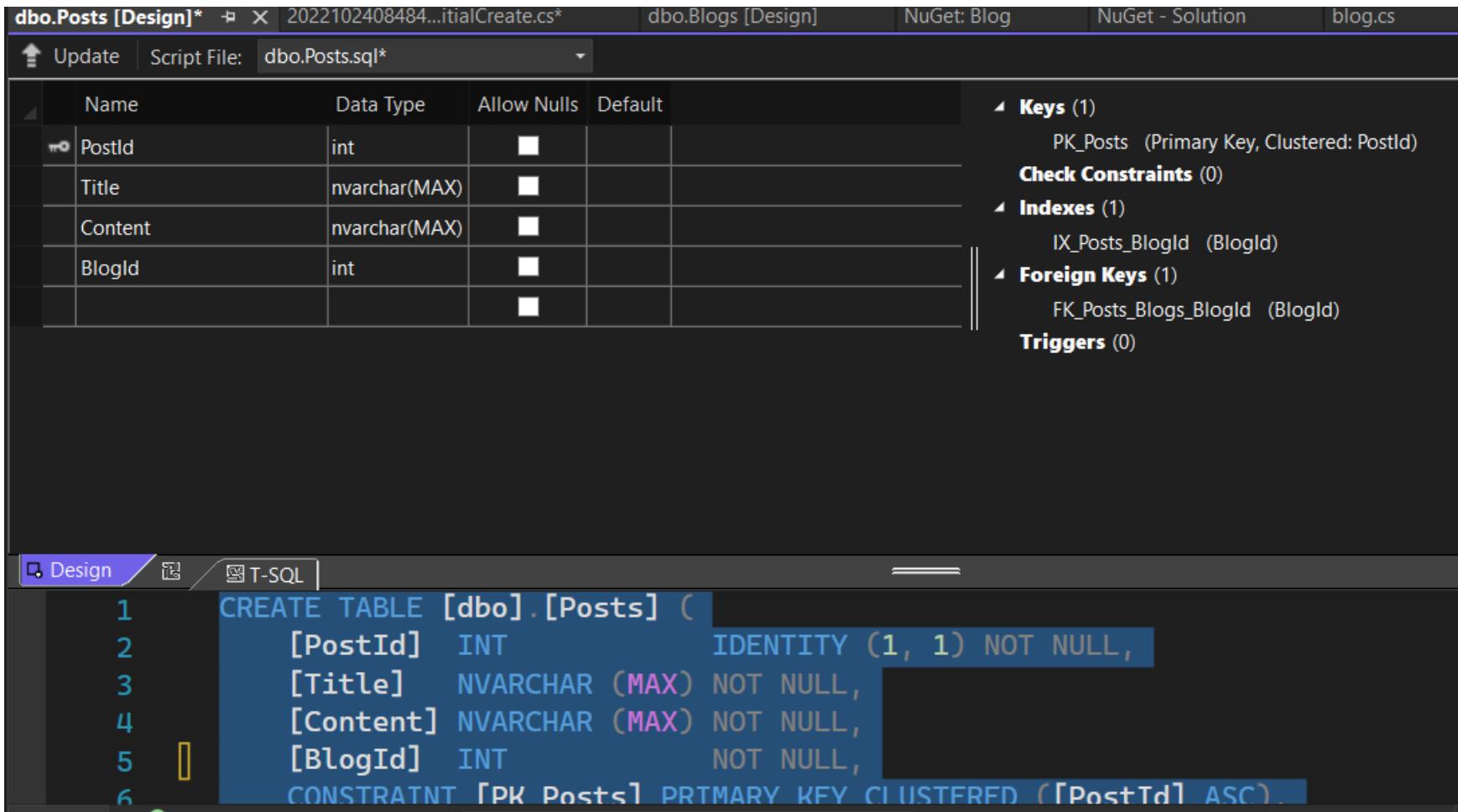
Constraints:

- Keys (1)**
 - PK_Blogs (Primary Key, Clustered: BlogId)
- Check Constraints (0)**
- Indexes (0)**
- Foreign Keys (0)**
- Triggers (0)**

T-SQL Script:

```
1 CREATE TABLE [dbo].[Blogs] (
2     [BlogId] INT           IDENTITY (1, 1) NOT NULL,
3     [Url]    NVARCHAR (MAX) NOT NULL,
4     [Rating] INT           NOT NULL,
5     CONSTRAINT [PK_Blogs] PRIMARY KEY CLUSTERED ([BlogId] ASC)
6 );
```

Entity Framework Core



The screenshot shows the Entity Framework Core tooling interface. The top navigation bar includes tabs for 'dbo.Posts [Design]', '2022102408484...italCreate.cs*', 'dbo.Blogs [Design]', 'NuGet: Blog', 'NuGet - Solution', and 'blog.cs'. Below the navigation bar, there's a toolbar with 'Update' and 'Script File: dbo.Posts.sql*' buttons.

The main area displays the 'Posts' table design. The table has four columns: 'Name', 'Data Type', 'Allow Nulls', and 'Default'. The columns are:

Name	Data Type	Allow Nulls	Default
PostId	int	<input checked="" type="checkbox"/>	
Title	nvarchar(MAX)	<input checked="" type="checkbox"/>	
Content	nvarchar(MAX)	<input checked="" type="checkbox"/>	
BlogId	int	<input checked="" type="checkbox"/>	

To the right of the table, there are several navigation links:

- Keys (1) → PK_Posts (Primary Key, Clustered: PostId)
- Check Constraints (0)
- Indexes (1) → IX_Posts_BlogId (BlogId)
- Foreign Keys (1) → FK_Posts_Blogs_BlogId (BlogId)
- Triggers (0)

At the bottom, there are tabs for 'Design' (selected) and 'T-SQL'. The T-SQL tab contains the generated CREATE TABLE script:

```
1 CREATE TABLE [dbo].[Posts] (
2     [PostId] INT IDENTITY (1, 1) NOT NULL,
3     [Title] NVARCHAR (MAX) NOT NULL,
4     [Content] NVARCHAR (MAX) NOT NULL,
5     [BlogId] INT NOT NULL,
6     CONSTRAINT [PK_Posts] PRIMARY KEY CLUSTERED ([PostId] ASC)
```

Entity Framework Core

Modificând clasa Post prin adăugarea: public string Autor { get; set; }

Rulăm comanda Add-Migration UpdatePostAddAutor

Se creează fișierul 20221024093752_UpdatePostAddAutor.cs

Entity Framework Core

```
namespace Blog.Migrations
{
    1 reference
    public partial class UpdatePostAddAutor : Migration
    {
        0 references
        protected override void Up(MigrationBuilder migrationBuilder)
        {
            migrationBuilder.AddColumn<string>(
                name: "Autor",
                table: "Posts",
                type: "nvarchar(max)",
                nullable: false,
                defaultValue: "");
        }
    }
}
```

Entity Framework Core

Rulăm comanda Update-Database

The screenshot shows the 'dbo.Posts [Design]' tab in SQL Server Management Studio. The table structure is defined with the following columns:

Name	Data Type	Allow Nulls	Default
PostId	int	<input type="checkbox"/>	
Title	nvarchar(MAX)	<input type="checkbox"/>	
Content	nvarchar(MAX)	<input type="checkbox"/>	
BlogId	int	<input type="checkbox"/>	
Autor	nvarchar(MAX)	<input type="checkbox"/>	(N'')

On the right side of the interface, there are navigation links for related entities and constraints:

- Keys (1)**
 - PK_Posts (Primary Key, Clustered: PostId)
- Check Constraints (0)**
- Indexes (1)**
 - IX_Posts_BlogId (BlogId)
- Foreign Keys (1)**
 - FK_Posts_Blogs_BlogId (BlogId)
- Triggers (0)**

Entity Framework Core

```
using var db = new BloggingContext();

// Create
Console.WriteLine("Inserting a new blog");
db.Add(new Blog { Url = "http://google.com", Rating = 4 });
db.SaveChanges();
```

Entity Framework Core

```
// Read  
Console.WriteLine("Querying for a blog");  
var blog = db.Blogs  
    .OrderBy(b => b.BlogId)  
    .First();
```

Entity Framework Core

```
// Update  
Console.WriteLine("Updating the blog and adding a post");  
blog.Url = "https://devblogs.microsoft.com/dotnet";  
blog.Posts.Add(  
    new Post { Title = "Hello World", Content = "I wrote an app using EF  
Core!" });  
db.SaveChanges();
```

Entity Framework Core

```
// Delete  
Console.WriteLine("Delete the blog");  
db.Remove(blog);  
db.SaveChanges();
```

Entity Framework Core

- Pentru query-uri folosim LINQ

```
var queryAllCustomers = from cust in customers  
                        select cust;
```

```
var queryLondonCustomers = from cust in customers  
                            where cust.City == "London"  
                            select cust;
```

Entity Framework Core

```
var queryLondonCustomers3 = from cust in customers  
    where cust.City == "London"  
    orderby cust.Name ascending  
    select cust;
```

Entity Framework Core

```
// queryCustomersByCity is an IEnumerable<IGrouping<string, Customer>>
var queryCustomersByCity =
    from cust in customers
    group cust by cust.City;

// customerGroup is an IGrouping<string, Customer>
foreach (var customerGroup in queryCustomersByCity)
{
    Console.WriteLine(customerGroup.Key);
    foreach (Customer customer in customerGroup)
    {
        Console.WriteLine("  {0}", customer.Name);
    }
}
```

Entity Framework Core

```
// custQuery is an IEnumerable<IGrouping<string, Customer>>
var custQuery =
    from cust in customers
    group cust by cust.City into custGroup
    where custGroup.Count() > 2
    orderby custGroup.Key
    select custGroup;
```

Entity Framework Core

```
var innerJoinQuery =  
    from cust in customers  
    join dist in distributors on cust.City equals dist.City  
    select new { CustomerName = cust.Name, DistributorName =  
dist.Name };
```

Entity Framework Core

Operatori LINQ vs Metode LINQ

```
IEnumerable<int> numQuery1 =  
    from num in numbers  
    where num % 2 == 0  
    orderby num  
    select num;
```

```
IEnumerable<int> numQuery2 = numbers.Where(num => num % 2 ==  
    0).OrderBy(n => n);
```

Entity Framework Core

Conexiunea la baza de date se realizează printr-un `ConnectionString`.

Aceasta se stochează, de obicei, în fișierul `appsettings.json`.

Ex:

```
"ConnectionStrings": {  
    "SchoolContext": "Server=(localdb)\\mssqllocaldb;Database=  
    SchoolContext0e9;Trusted_Connection=True;MultipleActiveResultSets=  
    true"  
}
```

Entity Framework Core

În fișierul Program.cs se înregistrează contextul bazei de date.

```
builder.Services.AddDbContext<SchoolContext>(  
    options =>  
        options.UseSqlServer(  
            builder.Configuration  
                .GetConnectionString("SchoolContext")  
        ));
```

Entity Framework Core

Tutorial: <https://learn.microsoft.com/en-us/aspnet/core/data/ef-rp/intro?view=aspnetcore-6.0&tabs=visual-studio>

Universitatea din Bucureşti
Facultatea de Matematică și Informatică

Dezvoltarea aplicațiilor web

CTI Anul IV, Semestrul 1 2022-2023

Curs 5 – 01.11.2022

ASP.NET Core Identity

- Sistem de management al utilizatorilor
- Nativ în ASP.NET Core
- Logică pentru UI
- Provideri externi
- Scaffolding
- Suport nativ pentru SqlServer

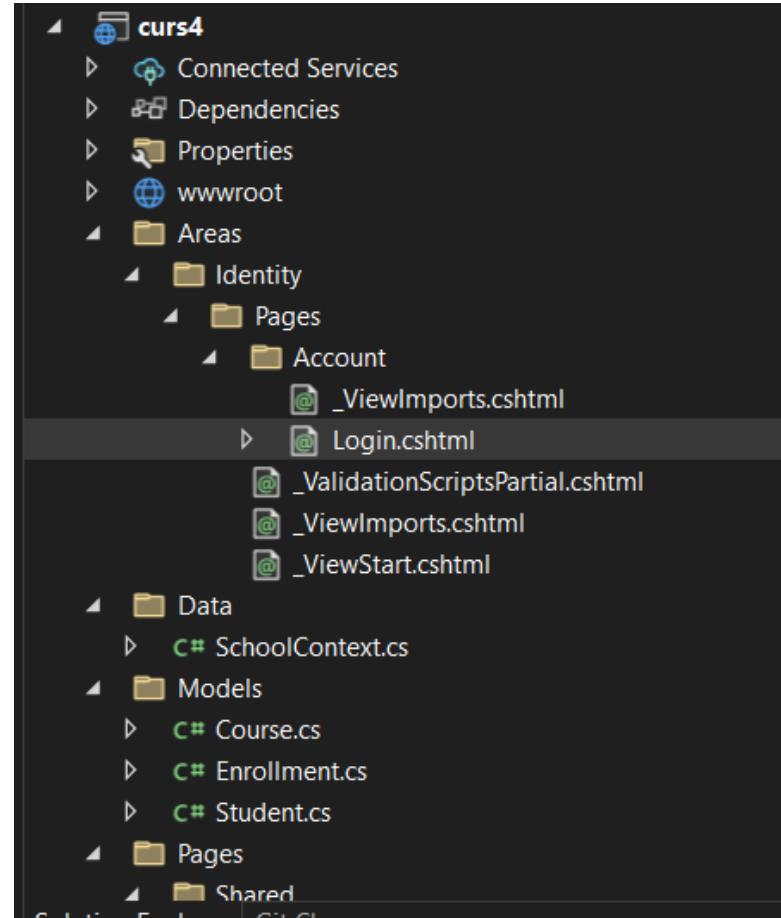
ASP.NET Core Identity

- Autentificare: cine e utilizatorul?
- Autorizare: ce poate face utilizatorul?
 - Roluri
 - Permisii

ASP.NET Core Identity

- Microsoft.AspNetCore.Identity.EntityFrameworkCore
- IdentityDbContext
- Identity Razor Class Library
 - /Identity/Account/Login
 - /Identity/Account/Logout
 - /Identity/Account/Manage
- Add -> New scaffolded item -> Identity

ASP.NET Core Identity



ASP.NET Core Identity

Log in

Use a local account to log in.

Remember me?

Log in

[Forgot your password?](#)

[Register as a new user](#)

[Resend email confirmation](#)

Use another service to log in.

There are no external authentication services configured. See this [article about setting up this ASP.NET application to support logging in via external services](#).

ASP.NET Core Identity

Program.cs:

- app.UseAuthentication();
 - app.UseAuthorization();
-
- Add-migration AddIdentity
 - Update-database

ASP.NET Core Identity

▷	 dbo.AspNetRoleClaims
▷	 dbo.AspNetRoles
▷	 dbo.AspNetUserClaims
▷	 dbo.AspNetUserLogins
▷	 dbo.AspNetUserRoles
▷	 dbo.AspNetUsers
▷	 dbo.AspNetUserTokens

ASP.NET Core Identity

- User: utilizatorii
- Role: rolurile
- UserClaim: permisiunile unui utilizator
- UserToken: token de autentificare al unui user
- UserLogin: autentificările unui utilizator
- RoleClaim: permisiunile unui rol
- UserRole: tabel asociativ între utilizatori și roluri

ASP.NET Core Identity

```
builder.Services.Configure<IdentityOptions>(options =>
{
    // Password settings.
    options.Password.RequireDigit = true;
    options.Password.RequireLowercase = true;
    options.Password.RequireNonAlphanumeric = true;
    options.Password.RequireUppercase = true;
    options.Password.RequiredLength = 6;
    options.Password.RequiredUniqueChars = 1;

    // Lockout settings.
    options.Lockout.DefaultLockoutTimeSpan = TimeSpan.FromMinutes(5);
    options.Lockout.MaxFailedAccessAttempts = 5;
    options.Lockout.AllowedForNewUsers = true;

    // User settings.
    options.User.AllowedUserNameCharacters =
        "abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789-_@";
    options.User.RequireUniqueEmail = false;
});
```

ASP.NET Core Identity

- [Facebook and Google authentication in ASP.NET Core | Microsoft Learn](#)

ASP.NET Core Identity

```
C# Copy

using Microsoft.AspNetCore.Authorization;
using Microsoft.AspNetCore.Mvc.RazorPages;
using Microsoft.Extensions.Logging;

namespace WebApp1.Pages
{
    [Authorize]
    public class PrivacyModel : PageModel
    {
        private readonly ILogger<PrivacyModel> _logger;

        public PrivacyModel(ILogger<PrivacyModel> logger)
        {
            _logger = logger;
        }

        public void OnGet()
        {
        }
    }
}
```

ASP.NET Core Identity

- Pentru autorizare folosim roluri sau permisiuni (claims)
- Putem crea oricâte roluri
- Identity RoleManager

ASP.NET Core Identity

- `CreateAsync(role)` – creează un rol nou
- `DeleteAsync(role)` – sterge un rol
- `FindByIdAsync(id)` – găsește un rol după id
- `FindByNameAsync(name)` – găsește un rol după nume
- `RoleExistsAsync(name)` – verifică dacă un rol există
- `UpdateAsync(name)` – actualizează un rol
- `Roles` – listează toate rolurile

ASP.NET Core Identity

UserManager

- `AddToRoleAsync(AppUser user, string name)`
- `RemoveFromRoleAsync(AppUser user, string name)`
- `GetRolesAsync(AppUser user)`
- `IsInRoleAsync(AppUser user, string name)`

ASP.NET Core Identity

La nivel de PageModel în Razor Pages.

- [Authorize(Roles = "Administrator")]
- [Authorize(Roles = "HRManager,Finance")]
- [Authorize(Roles = "PowerUser")]
[Authorize(Roles = "ControlPanelUser")]

ASP.NET Core Identity

- Permisii sunt perechi nume, valoare
- Pentru a verifica permisiunile se creează un policy

```
builder.Services.AddAuthorization(options => {
    options.AddPolicy("EmployeeOnly", policy =>
        policy.RequireClaim("EmployeeNumber")));
});
```
- [Authorize(Policy = "EmployeeOnly")]

ASP.NET Core Identity

```
builder.Services.AddAuthorization(options => {
    options.AddPolicy("Founders", policy =>
        policy.RequireClaim("EmployeeNumber", "1", "2", "3", "4", "5"));
});
```

```
[Authorize(Policy = „Founders“)]
```

Universitatea din Bucureşti
Facultatea de Matematică și Informatică

Dezvoltarea aplicațiilor web

CTI Anul IV, Semestrul 1 2022-2023

Curs 6 – 08.11.2022

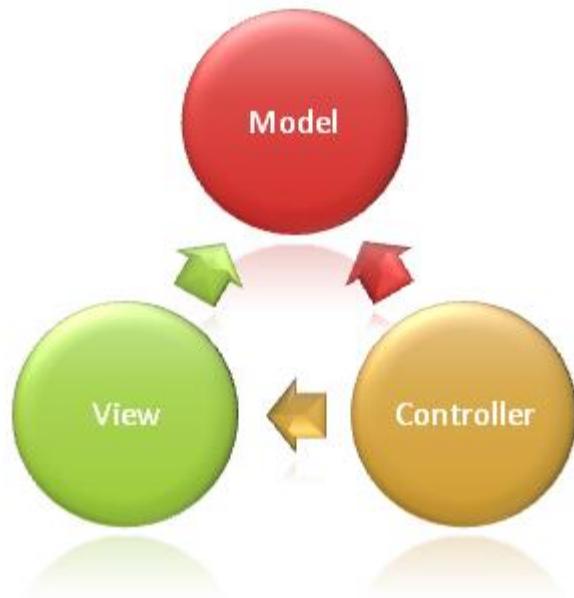
Tipuri de UI în ASP.NET Core

- La nivel de server: HTML și CSS generate dinamic la nivelul serverului ca răspuns la un request.
 - ✓ Necessită resurse limitate la nivel de client
 - ✓ Acces facil la resurse protejate
 - ✓ SEO
- Costul resurselor este agregat la nivel de server
- Interacțiunile la nivel de client necesită accesarea serverului

ASP.NET Core MVC

- ASP.NET Core UI
- La nivel de server
- Arhitectura MVC:
 - Model
 - View
 - Controller
- Complex, dar robust

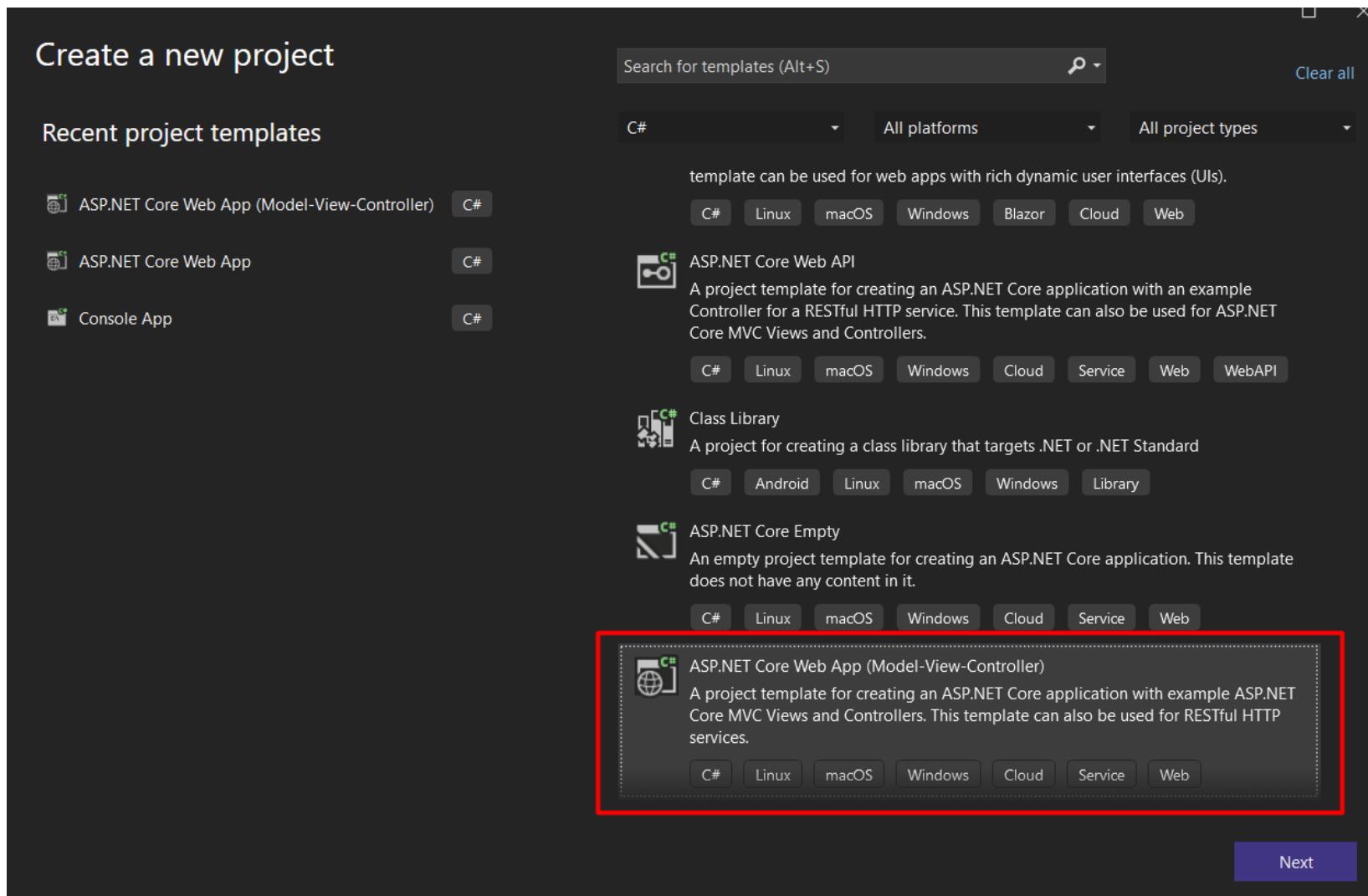
ASP.NET Core MVC



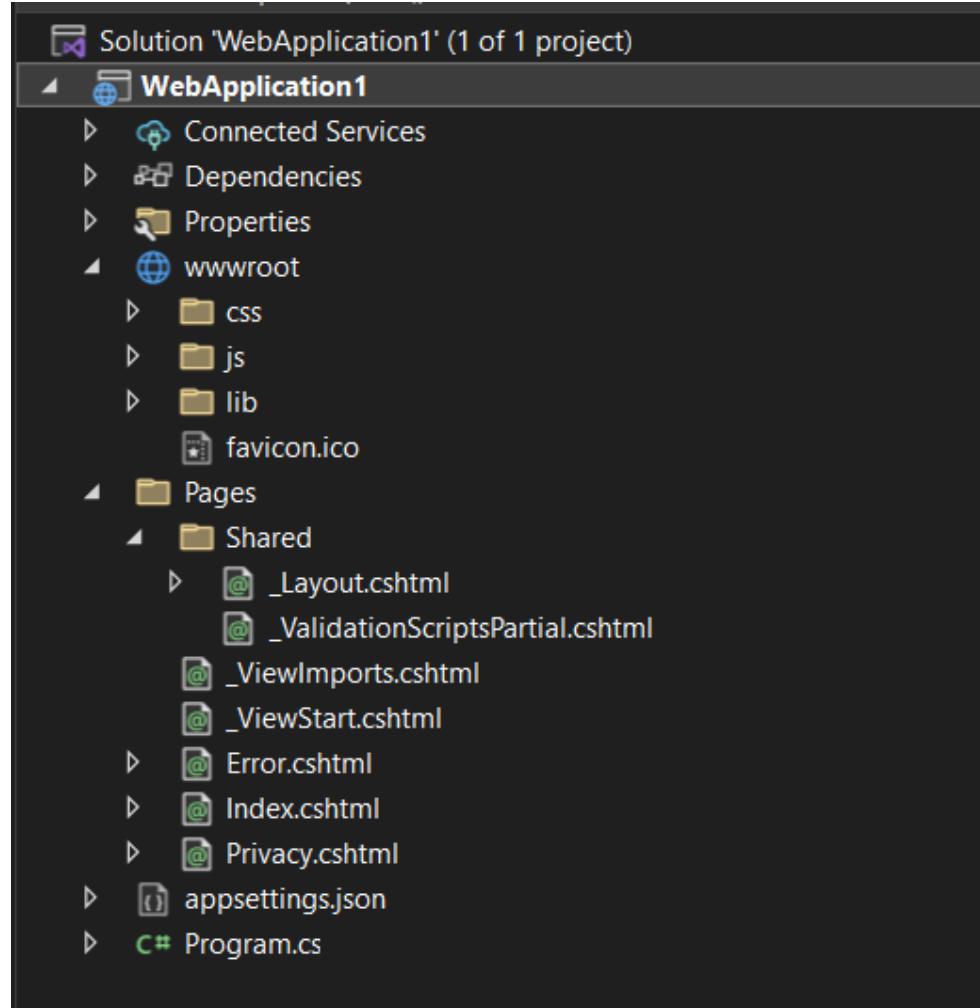
Separation of concerns:

- Model: reprezentare a logicii de business cu operațiile necesare, păstrează starea curentă a aplicației, gestionează datele
- View: prezentarea conținutului prin interfață
- Controller: gestionează requesturile, se ocupă de interacțiunile utilizatorului, comunică cu modelul și selectează view-ul care trebuie afișat

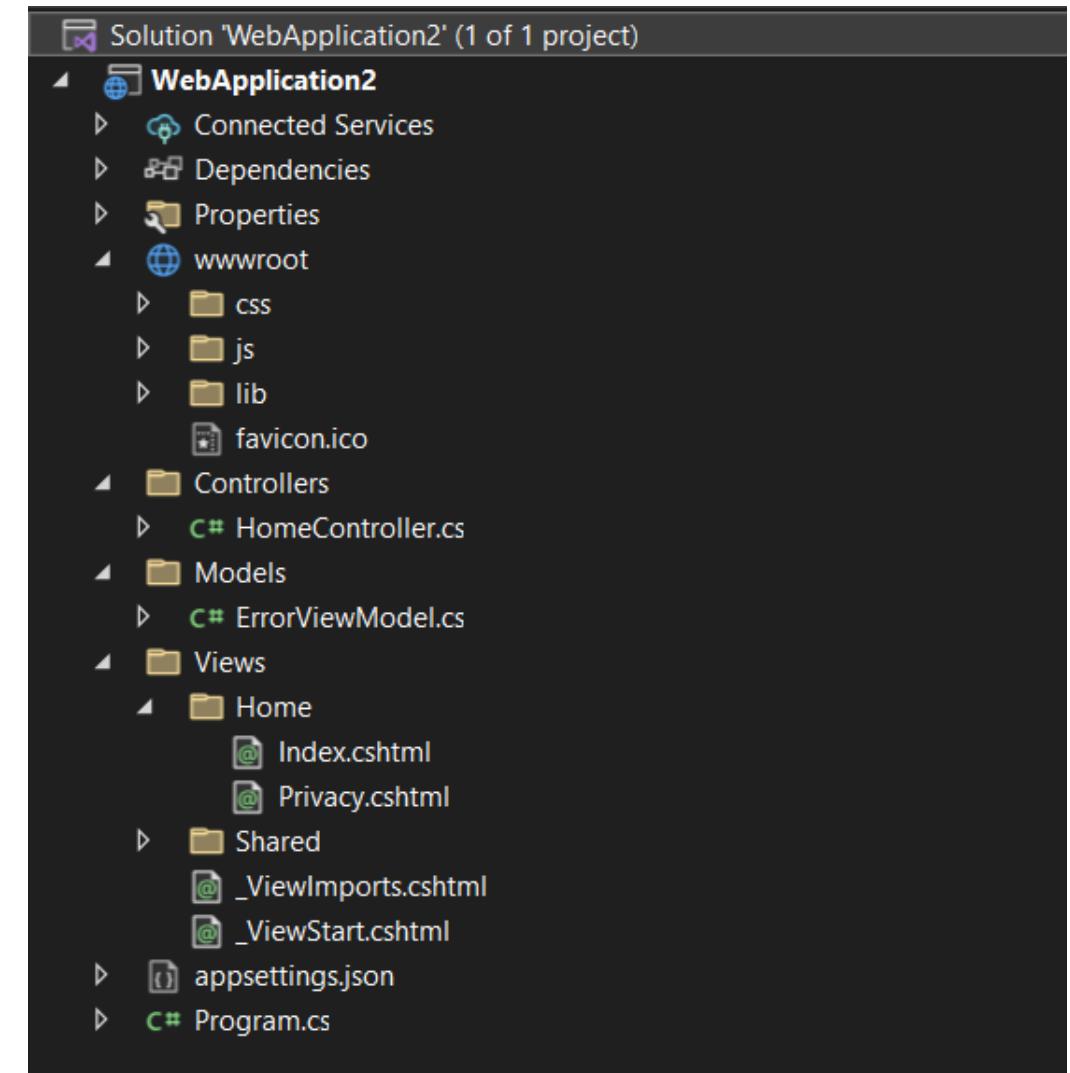
ASP.NET Core MVC



ASP.NET Core MVC



Razor pages



MVC

ASP.NET Core MVC

- Directorul Controllers conține controllerele – fișiere cs
- Directorul Models conține modelele – fișiere cs
- Directorul Views conține viewurile – fișiere cshtml

ASP.NET Core MVC

- În fișierul appsettings.json se țin detalii de configurare
- Ca în cazul Razor Pages, directorul wwwroot: conține asseturi statice:
 - Fișiere HTML
 - Fișiere CSS
 - Fișiere Javascript
 - Imagini
 - Fonturi

ASP.NET Core MVC

Fișierul Program.cs

Se creează o aplicație nouă cu suport pentru MVC și se construiește aplicația.

```
var builder = WebApplication.CreateBuilder(args);
// Add services to the container.
builder.Services.AddRazorPages();
builder.Services.AddControllersWithViews();
var app = builder.Build();
```

ASP.NET Core MVC

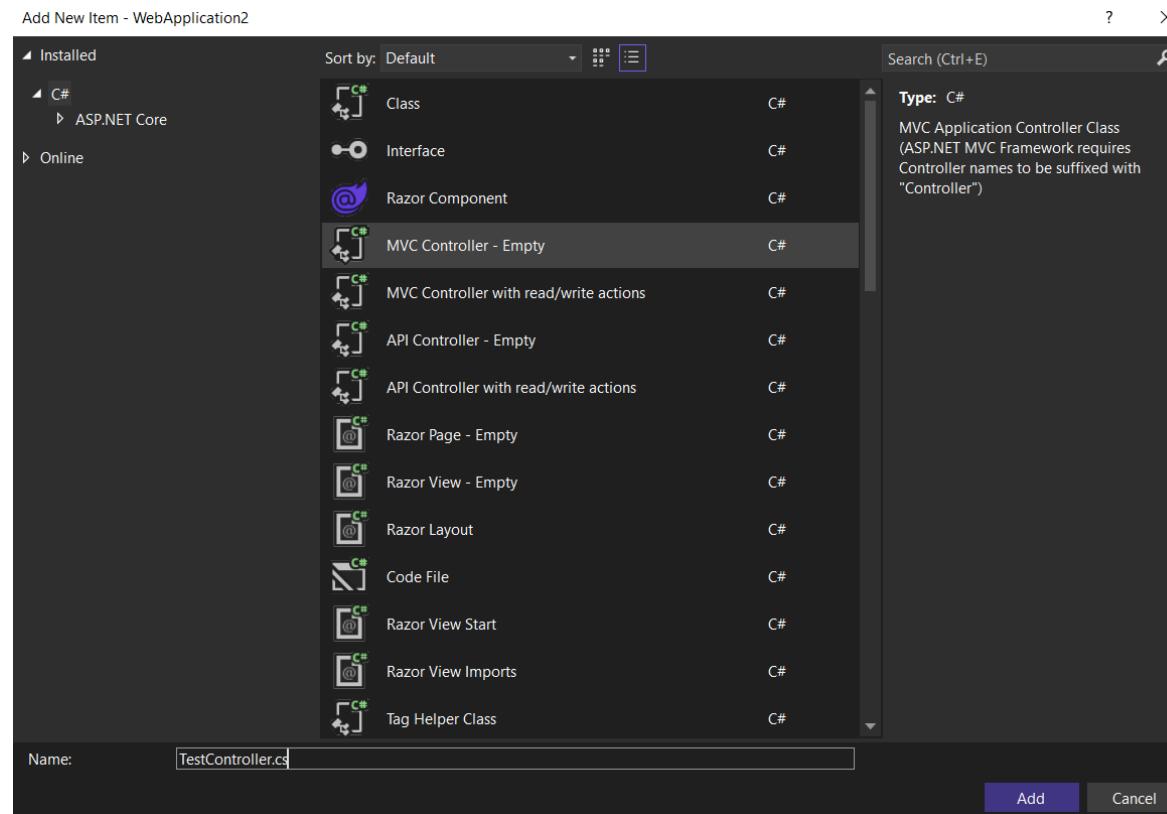
Fișierul Program.cs

Rutarea implicită este de tipul [adresa]/[controller]/[actiune]/[ID], unde ID-ul este optional. Dacă nu se specifică acești parametri, se va apela acțiunea Index a controllerului Home.

```
app.MapControllerRoute  
    name: "default",  
    pattern: "{controller=Home}/{action=Index}/{id?}";
```

ASP.NET Core MVC

- Adăugare controller: click dreapta pe Controllers, Add, Controller..., MVC Controller - Empty



ASP.NET Core MVC

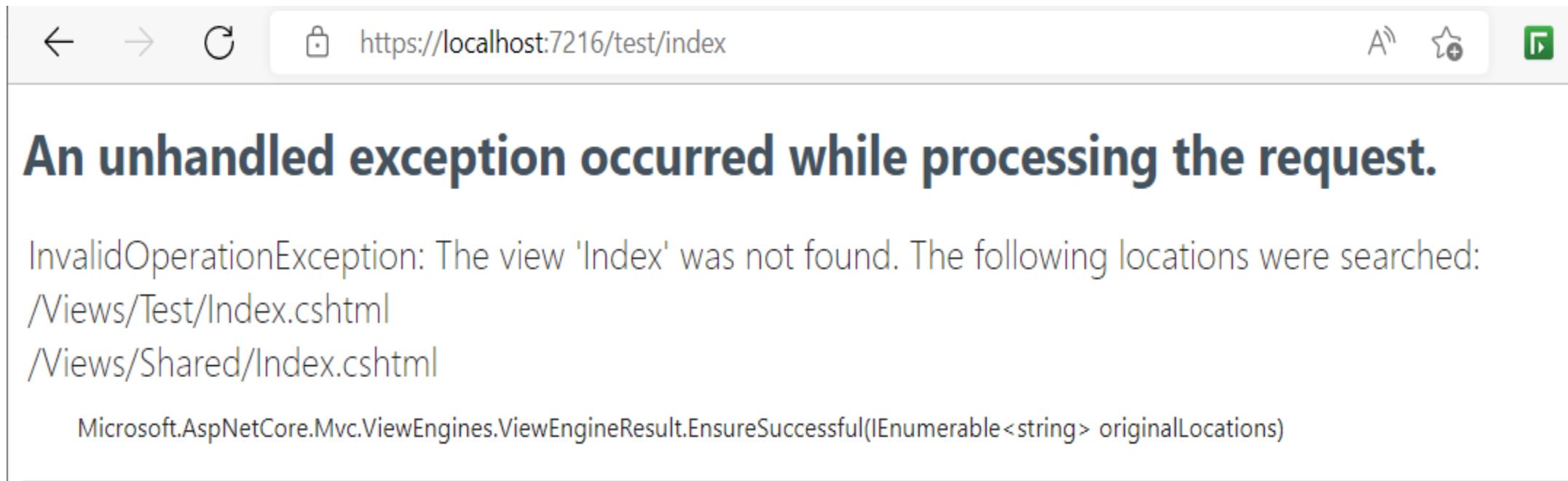
```
using Microsoft.AspNetCore.Mvc;

namespace WebApplication2.Controllers
{
    0 references
    public class TestController : Controller
    {
        0 references
        public IActionResult Index()
        {
            return View();
        }
    }
}
```

- Clasă care moștenește clasa Controller
- Numele se încheie cu sufixul Controller
- Metodele din controller: acțiuni
- View-ul returnat implicit este Views/Test/Index.cshtml

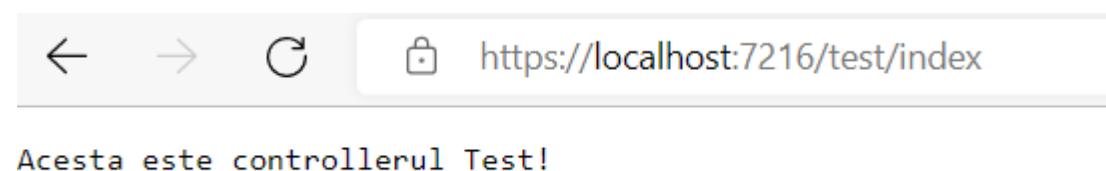
ASP.NET Core MVC

- Nu există view-ul Index



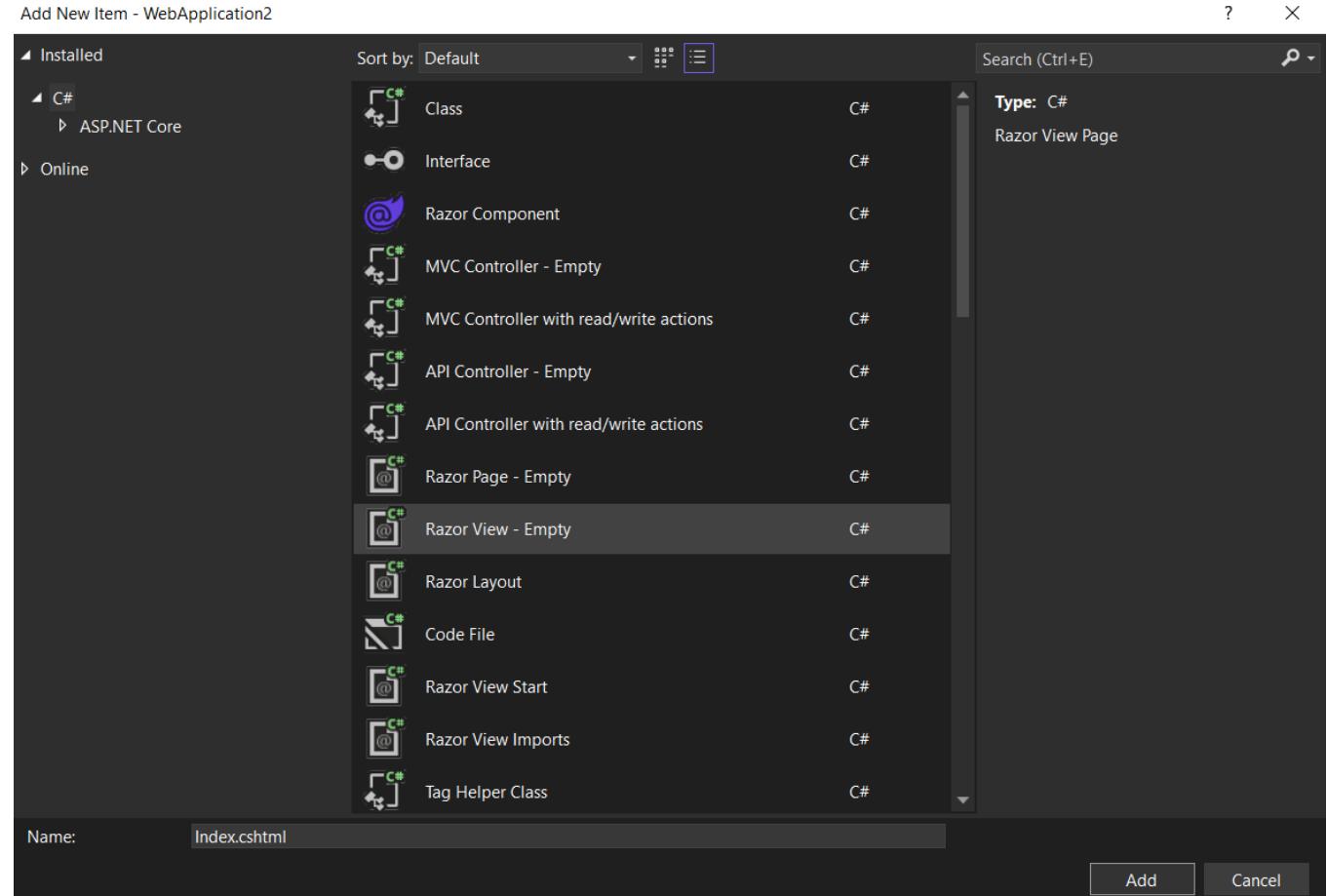
ASP.NET Core MVC

```
public class TestController : Controller
{
    0 references
    public string Index()
    {
        return "Acesta este controllerul Test!";
    }
}
```



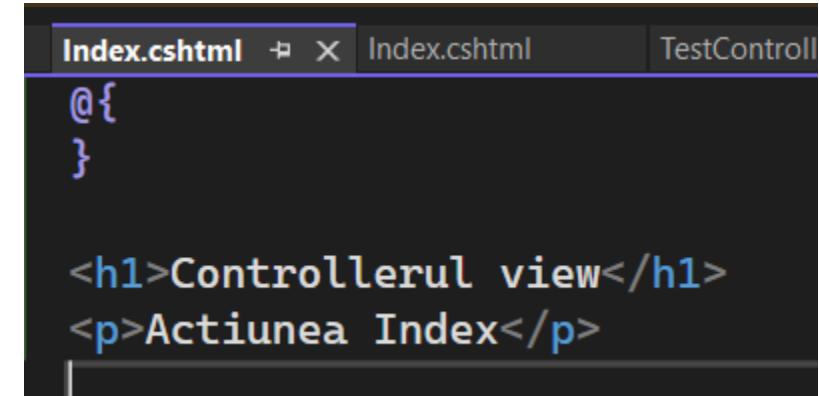
ASP.NET Core MVC

- Adăugare view: pentru a adăuga view-ul pentru acțiunea Index a controllerului Test, creăm mai întâi directorul Test, în directorul Views. Apoi dăm click dreapta pe directorul nou adăugat, selectăm Add, View, Razor View – Empty.



ASP.NET Core MVC

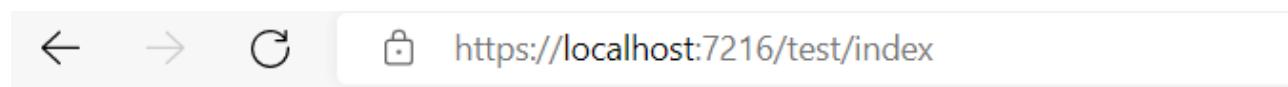
```
public class TestController : Controller
{
    0 references
    public IActionResult Index()
    {
        return View();
    }
}
```



The screenshot shows a code editor with three tabs: Index.cshtml, Index.cshtml, and TestController. The active tab is Index.cshtml, which contains the following code:

```
@{
}

<h1>Controllerul view</h1>
<p>Actiunea Index</p>
```



WebApplication2 Home Privacy

Controllerul view

Actiunea Index

ASP.NET Core MVC

Pentru a încărca un anumit view:

- return View("Orders");
- return View("Views/Home/About.cshtml");
- return View("../Manage/Index");
- return View("./About");

ASP.NET Core MVC

Transmiterea datelor între controller și view:

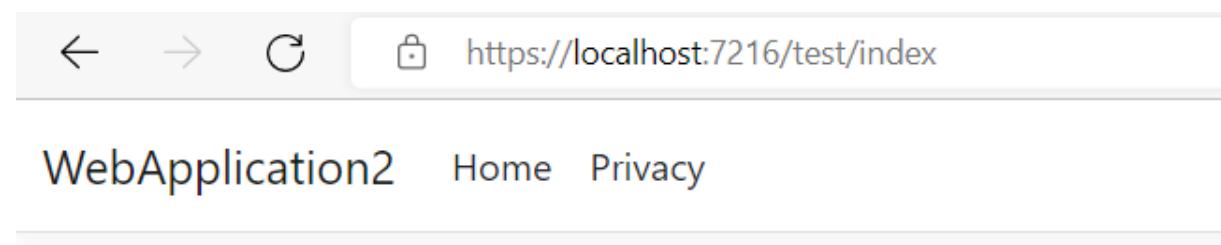
- ViewData
- ViewBag
- ViewModel

ASP.NET Core MVC

```
public class TestController : Controller
{
    0 references
    public IActionResult Index()
    {
        ViewData["Text"] = "Acesta este textul din ViewData";
        return View();
    }
}
```

```
@{
}

<h1>Controllerul view</h1>
<p>Actiunea Index</p>
<p>Text: @ViewData["Text"]</p>
```



Controllerul view

Actiunea Index

Text: Acesta este textul din ViewData

ASP.NET Core MVC

```
public class TestController : Controller
{
    [ViewData]
    1 reference
    public string Text { get; set; }
    0 references
    public IActionResult Index()
    {
        Text = "Acesta este textul din ViewData";
        return View();
    }
}
```

```
@{
}

<h1>Controllerul view</h1>
<p>Actiunea Index</p>
<p>Text: @ViewData["Text"]</p>
```



WebApplication2 Home Privacy

Controllerul view

Actiunea Index

Text: Acesta este textul din ViewData

ASP.NET Core MVC

```
public class TestController : Controller
{
    0 references
    public IActionResult Index()
    {
        ViewData["Text"] = "Acesta este textul din ViewData";
        ViewBag.Text = "Acesta este textul din ViewBag";
        return View();
    }
}
```

```
@{
}

<h1>Controllerul view</h1>
<p>Actiunea Index</p>
<p>Text: @ViewData["Text"]</p>
<p>Text: @ViewBag.Text</p>
```

The screenshot shows a browser window with the URL `https://localhost:7216/test/index`. The page title is "WebApplication2". The main content area displays the text "Controllerul view" in a large font, followed by "Actiunea Index" and two paragraphs of text: "Text: Acesta este textul din ViewBag" and "Text: Acesta este textul din ViewBag".

← → ⌂ https://localhost:7216/test/index

WebApplication2 Home Privacy

Controllerul view

Actiunea Index

Text: Acesta este textul din ViewBag

Text: Acesta este textul din ViewBag

ASP.NET Core MVC

```
public class Student
{
    1 reference
    public string Nume { get; set; }
    1 reference
    public int Grupa { get; set; }
    1 reference
    public Student(string nume, int grupa)
    {
        Nume = nume;
        Grupa = grupa;
    }
}
```

ASP.NET Core MVC

```
public class TestController : Controller
{
    0 references
    public IActionResult Index()
    {
        ViewData["Student"] = new Student("Popescu Ion", 451);
        return View();
    }
}
```

```
@using WebApplication2.Controllers
 @{
    var student = ViewData["Student"] as Student;
}

<p>Nume student: @student.Nume</p>
<p>Grupa student: @student.Grupa</p>
```



WebApplication2 Home Privacy

Nume student: Popescu Ion

Grupa student: 451

ASP.NET Core MVC

```
public class TestController : Controller
{
    public IActionResult Index()
    {
        var student = new Student("Popescu Ion", 451);
        return View(student);
    }
}
```

```
@model WebApplication2.Controllers.Student

<p>Nume student: @Model.Nume</p>
<p>Grupa student: @Model.Grupa</p>
```

← → ⌂ https://localhost:7216/test/index

WebApplication2 Home Privacy

Nume student: Popescu Ion

Grupa student: 451

ASP.NET Core MVC

Pentru blocurile din view care se repetă se pot utiliza:

- PartialView
- ViewComponent

ASP.NET Core MVC

PartialView

```
@foreach(var stire in Model.Stiri) {  
    <partial name="_StirePartial" model="stire">  
}
```

- Nu are logică proprie

ASP.NET Core MVC

- ViewComponent: clasă + template
- Clasa poate fi creată:
 - ❑ Moștenind clasa ViewComponent
 - ❑ Decorând clasa cu atributul [ViewComponent]
 - ❑ Denumind clasa cu sufixul ViewComponent

ASP.NET Core MVC

Logica unui ViewComponent este definită în metoda Invoke (InvokeAsync)

```
public async Task<IViewComponentResult> InvokeAsync(int maxPriority, bool isDone)
{
    string MyView = "Default";
    if (maxPriority > 3 && isDone == true)
    {
        MyView = "PVC";
    }
    var items = await GetItemsAsync(maxPriority, isDone);
    return View(MyView, items);
}
```

ASP.NET Core MVC

Un ViewComponent poate fi încărcat:

- Din template:

```
<vc:priority-list max-priority=maxPriority is-done=isDone>
</vc:priority-list>
```

- Din controller:

```
public IActionResult IndexVC(int maxPriority = 2, bool isDone = false) {
    return ViewComponent("PriorityList", new { maxPriority =
        maxPriority, isDone = isDone });
}
```

ASP.NET Core MVC

- Adăugare model: pentru a adăuga un model, click dreapta pe directorul Models și Add > New class.

```
public class Student
{
    0 references
    public int Id { get; set; }
    4 references
    public string Nume { get; set; }
    4 references
    public int Grupa { get; set; }
    0 references
    public Student(string nume, int grupa)
    {
        Nume = nume;
        Grupa = grupa;
    }
    0 references
    public Student() { }
}
```

ASP.NET Core MVC

- Pentru conexiunea la baza de date, trebuie efectuați pașii învățați la Razor Pages. Se creează un Context și apoi o migrație.

```
"ConnectionStrings": {  
    "Student": "Data Source=(localdb)\\MSSQLLocalDB;Initial Catalog=student;Integrated Security=True"  
}
```

```
builder.Services.AddDbContext<StudentContext>(options =>  
    options.UseSqlServer(builder.Configuration.GetConnectionString("Student")));
```

```
public class StudentContext : DbContext  
{  
    public DbSet<Student> students { get; set; }  
  
    public StudentContext(DbContextOptions options) : base(options) { }  
}
```

```
PM> add-migration InitialMigration
```

```
PM> update-database
```

SQL Server Object Explorer

dbo.students [Design] 2022111321370...IMigration.cs NuGet - Solution StudentContext.cs

Update Script File: dbo.students.sql

Name	Data Type	Allow Nulls	Default
Id	int	■	
Nume	nvarchar(MAX)	■	
Grupa	int	■	

Keys (1)
PK_students (Primary Key, Clustered: Id)

Check Constraints (0)

Indexes (0)

Foreign Keys (0)

Triggers (0)

Design T-SQL

```
CREATE TABLE [dbo].[students] (
    [Id] INT IDENTITY (1, 1) NOT NULL,
    [Nume] NVARCHAR (MAX) NOT NULL,
    [Grupa] INT NOT NULL,
    CONSTRAINT [PK_students] PRIMARY KEY CLUSTERED ([Id] ASC)
```

133 % No issues found

Connection Ready

L:\localdb\MSQLLocalDB\LUVOETPS\svany\student

Solution Explorer

Search Solution Explorer (Ctrl+;)

- Controllers
 - C# HomeController.cs
 - C# TestController.cs
- Data
 - C# StudentContext.cs
- Migrations
- Models
 - C# ErrorViewModel.cs
 - C# Student.cs
- Views
 - Home
 - Index.cshtml
 - Privacy.cshtml
 - Shared
 - Test
 - Index.cshtml
 - _ViewImports.cshtml
 - _ViewStart.cshtml
 - Index.cshtml
- appsettings.json
- C# Program.cs

ASP.NET Core MVC

```
Adauga.cshtml ✘ X dbo.students [Design] Index.cshtml TestController.cs Student.cs 2022111321370...IM
1 @model WebApplication2.Models.Student
2
3 <form method="post">
4   <div class="form-group">
5     <label asp-for="@Model.Nume">Nume</label>
6     <input class="form-control" asp-for="@Model.Nume" />
7   </div>
8   <div class="form-group">
9     <label asp-for="@Model.Grupa">Grupa</label>
10    <input class="form-control" asp-for="@Model.Grupa" />
11  </div>
12  <input type="submit" />
13 </form>
```

ASP.NET Core MVC

```
public StudentContext _studentContext { get; set; }
0 references
public TestController(StudentContext studentContext)
{
    _studentContext = studentContext;
}
0 references
public IActionResult Index()
{
    var student = _studentContext.students.FirstOrDefault();
    return View(student);
}
0 references
public IActionResult Adauga()
{
    return View();
}
[HttpPost]
0 references
public IActionResult Adauga(Student student)
{
    _studentContext.Add(student);
    _studentContext.SaveChanges();
    return RedirectToAction("Index");
}
```

ASP.NET Core MVC

← → ⌂ https://localhost:7216/test/adauga

WebApplication2 Home Privacy

Nume

Grupa

← → ⌂ https://localhost:7216/Test

WebApplication2 Home Privacy

Nume student: ion

Grupa student: 232

ASP.NET Core MVC

- <https://learn.microsoft.com/en-us/aspnet/core/tutorials/first-mvc-app/start-mvc?view=aspnetcore-7.0&tabs=visual-studio>

Universitatea din Bucureşti
Facultatea de Matematică și Informatică

Dezvoltarea aplicațiilor web

CTI Anul IV, Semestrul 1 2022-2023

Curs 7

Aplicațiile web

- Front-end
 - Interfața aplicației (HTML, CSS, JS, ...)
- Back-end
 - Funcționalitate, date (C#, Java, PHP, ...)

Aplicațiile web

- ASP.Net Core Razor Pages, ASP.Net Core MVC – frameworkuri complete, atât BE, cât și FE
- ASP.Net Core Web API – doar BE

Aplicațiile web

- ASP.Net Core Razor Pages, ASP.Net Core MVC – frameworkuri complete, atât BE, cât și FE
- ASP.Net Core Web API – doar BE

Web API

- API (Application programming interface) – permite comunicarea între componente software diferite
- Web API – interfață web ce poate fi accesată folosind protocolul HTTP: Request – Response (Solicitare – Răspuns)
- Putem implementa operațiile CRUD folosind Web API-uri

Request

- Solicitare făcută de client la server
- Structură:
 - Request Line: Metodă, Cale, Versiune HTTP
 - Headere: Content type, Content length, Authorization
 - Conținut: HTML, CSS, Javascript

Metode (acțiuni)

- GET (cere resursa)
 - POST (creează resursa)
 - PUT (actualizează resursa)
 - PATCH (actualizează parțial resursa)
 - DELETE (elimină resursa)
- ...și altele: <https://developer.mozilla.org/en-US/docs/Web/HTTP/Methods>

Headere

- Authorization (userul are acces la o anumită resursă?)
 - Content-Type (tipul de date primit)
 - Accept (tipul de date așteptat de la server)
 - Cookie (informații suplimentare despre user)
 - Access-Control-Allow-Origin (CORS - din ce origine pot fi accesate resursele:
<https://developer.mozilla.org/en-US/docs/Web/HTTP/CORS>)
- ...și altele: <https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers>

Exemplu request

```
curl https://reqres.in/api/users?page=2 -v
```

<https://reqres.in/>

Răspuns

- Status code
- Headere
- Conținut

Status codes

- Răspunsuri informative (100–199)
- Răspunsuri de succes (200–299)
- Mesaje de redirectare (300–399)
- Erori la nivel de client (400–499)
- Erori la nivel de server (500–599)

Detalii: <https://developer.mozilla.org/en-US/docs/Web/HTTP/Status>

ASP.NET Core Web API

Create a new project

Recent project templates

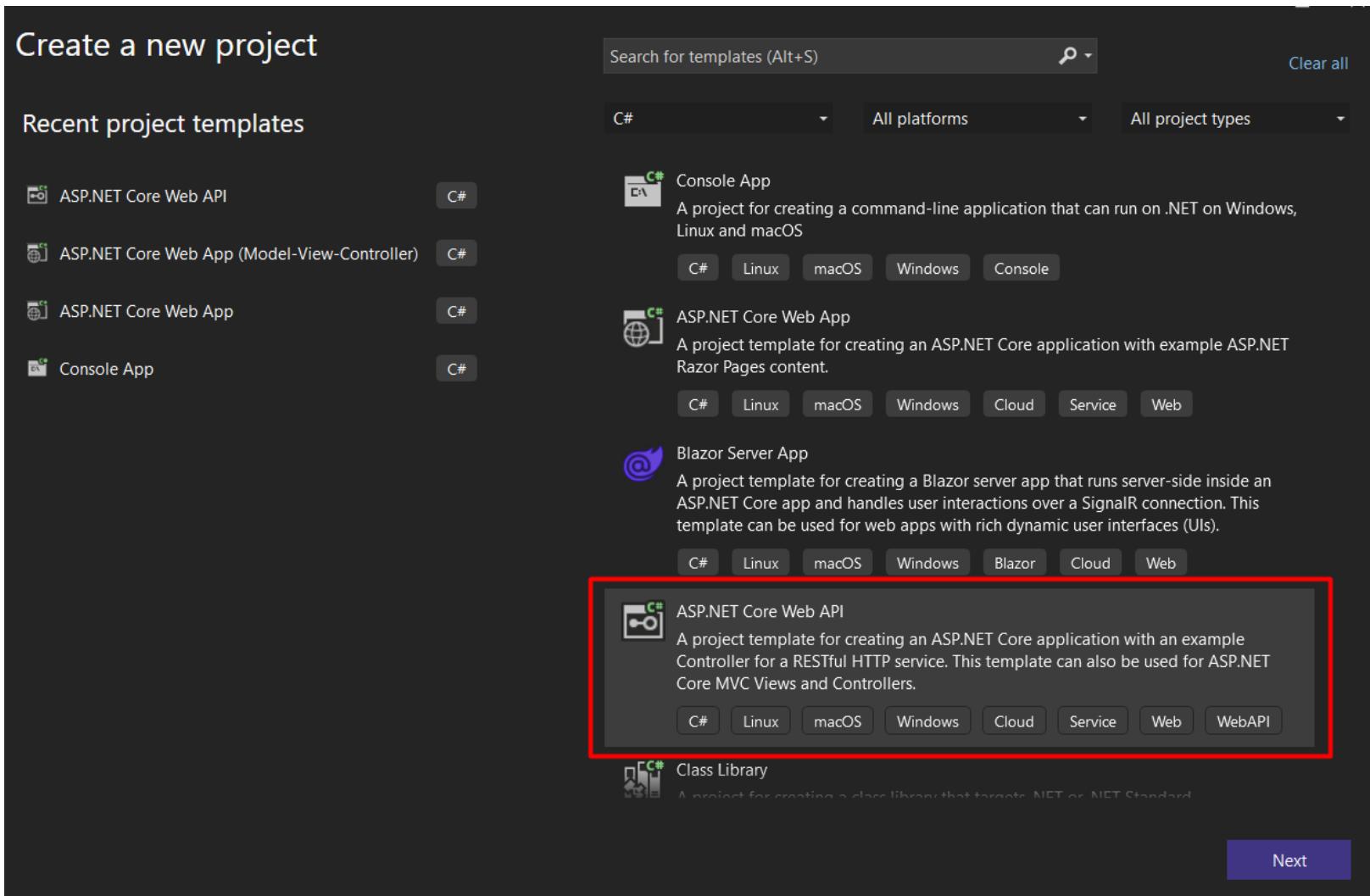
Search for templates (Alt+S)

Clear all

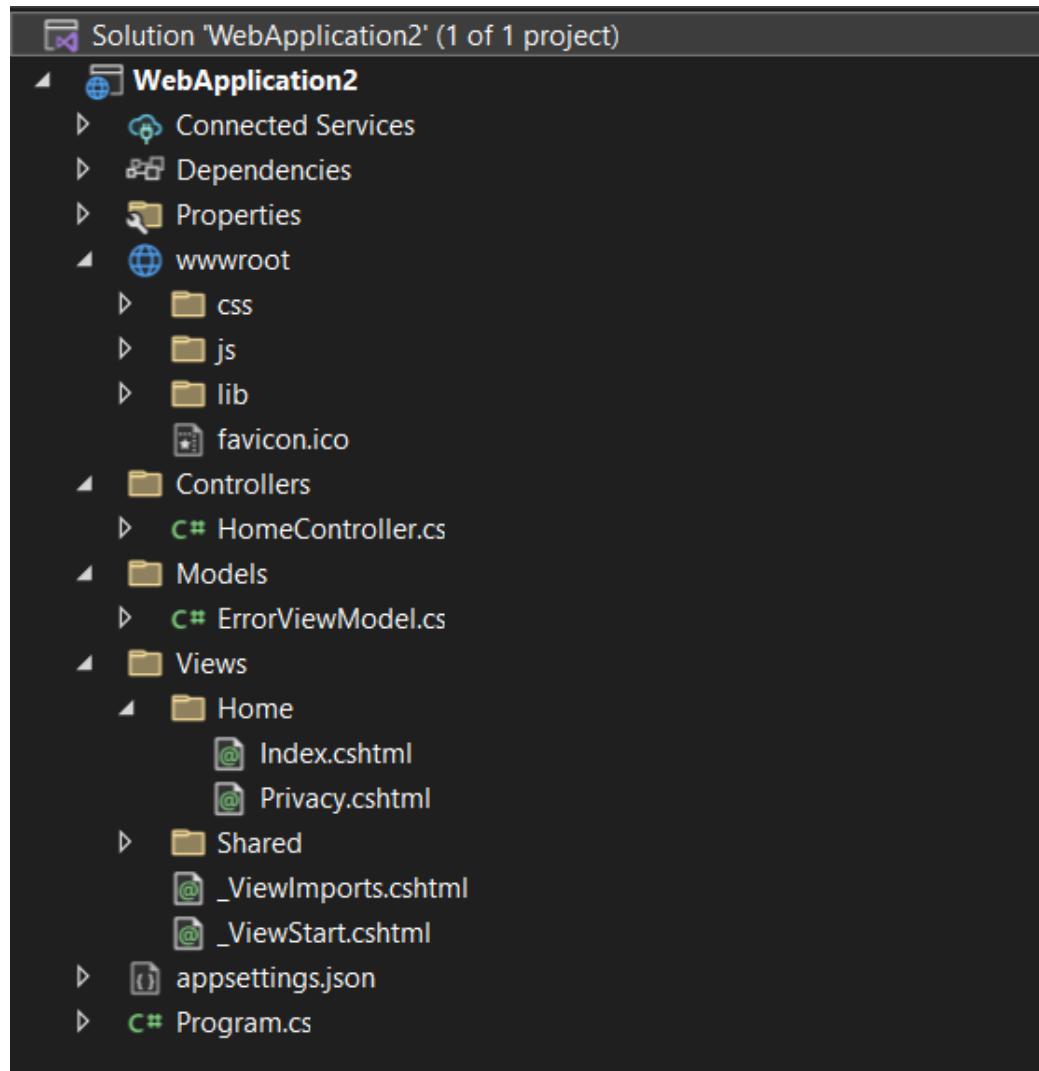
C# All platforms All project types

ASP.NET Core Web API	C#	Console App	A project for creating a command-line application that can run on .NET on Windows, Linux and macOS	C# Linux macOS Windows Console
ASP.NET Core Web App (Model-View-Controller)	C#	ASP.NET Core Web App	A project template for creating an ASP.NET Core application with example ASP.NET Razor Pages content.	C# Linux macOS Windows Cloud Service Web
ASP.NET Core Web App	C#	Blazor Server App	A project template for creating a Blazor server app that runs server-side inside an ASP.NET Core app and handles user interactions over a SignalR connection. This template can be used for web apps with rich dynamic user interfaces (UIs).	C# Linux macOS Windows Blazor Cloud Web
Console App	C#	ASP.NET Core Web API	A project template for creating an ASP.NET Core application with an example Controller for a RESTful HTTP service. This template can also be used for ASP.NET Core MVC Views and Controllers.	C# Linux macOS Windows Cloud Service Web WebAPI
		Class Library	A project for creating a class library that targets .NET or .NET Standard	C# Linux macOS Windows Cloud Service Web

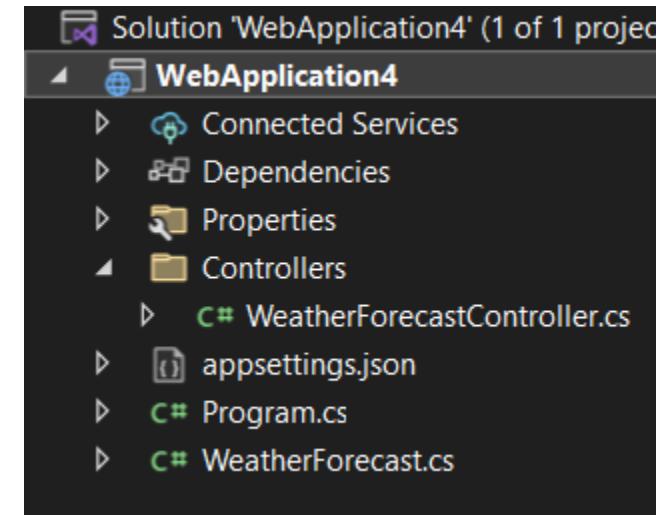
Next



ASP.NET Core Web API



MVC



Web API

ASP.NET Core Web API

- Structură simplificată a arhitecturii MVC – lipsesc vizualizările
- Directorul Controllers conține controllerele – fișiere cs
- Fișierul WeatherForecast.cs este un model (clasă); vom lucra cu un director Models

ASP.NET Core Web API

Fișierul Program.cs

Se creează o aplicație nouă cu suport pentru Web API și Swagger și se construiește aplicația.

```
var builder = WebApplication.CreateBuilder(args);
// Add services to the container.
builder.Services.AddControllersWithViews();
builder.Services.AddControllers();
builder.Services.AddSwaggerGen();
var app = builder.Build();
```

ASP.NET Core Web API

Swagger – instrumente built-in pentru testarea și documentarea API-urilor

The screenshot shows the Swagger UI interface for an ASP.NET Core Web API. The URL in the browser is `https://localhost:7053/swagger/index.html`. The main title is "Responses".

Curl

```
curl -X 'GET' \
  'https://localhost:7053/WeatherForecast' \
  -H 'accept: text/plain'
```

Request URL

```
https://localhost:7053/WeatherForecast
```

Server response

Code	Details
200	Response body [{ "date": "2022-11-26T14:21:21.6750866+02:00", "temperatureC": 34, "temperatureF": 93, "summary": "Scorching" }, { "date": "2022-11-26T14:21:21.6750866+02:00", "temperatureC": 34, "temperatureF": 93, "summary": "Scorching" }]

ASP.NET Core Web API

- Controllerele din WebAPI moștenesc ControllerBase (Controller conține suport pentru Views, care nu este necesar în cazul API-urilor)
- Se decorează clasa cu [ApiController]
- Se poate da o rută generală. În exemplul ruta este:
hostname:port/WeatherForecast

```
[ApiController]
[Route("[controller]")]
public class WeatherForecastController : ControllerBase
```

ASP.NET Core Web API

- Metodele sunt decorate cu verbul HTTP cu care se dorește efectuarea acțiunii: `HttpGet`, `HttpPost`, `HttpDelete`, `HttpPut`...

```
[HttpGet]
0 references
public async Task<ActionResult<IEnumerable<TodoItem>>> GetTodoItems()
{
    return await _context.TodoItems.ToListAsync();
}
```

ASP.NET Core Web API

- Ruta fiecărei acțiuni poate fi dată ca parametru al decoratorului
- Metoda din exemplu va fi apelată la [adresa dată în controller]/id

```
[HttpDelete("{id}")]
0 references
public async Task<IActionResult> DeleteTodoItem(long id)
{
    var todoItem = await _context.TodoItems.FindAsync(id);
    if (todoItem == null)
    {
```

ASP.NET Core Web API

- Este important să tratăm toate cazurile posibile și să returnăm codul de status corespunzător.

```
public async Task<IActionResult> PutTodoItem(long id, TodoItem todoItem)
{
    if (id != todoItem.Id){
        return BadRequest();
    }
    _context.Entry(todoItem).State = EntityState.Modified;
    try {
        await _context.SaveChangesAsync();
    }
    catch (DbUpdateConcurrencyException) {
        if (!TodoItemExists(id))
        {
            return NotFound();
        }
        else
        {
            throw;
        }
    }
    return NoContent();
}
```

ASP.NET Core Web API

Acțiunile pot returna:

- Tip de date specific (primitiv sau custom)
- IActionResult
- ActionResult

<https://learn.microsoft.com/en-us/aspnet/core/web-api/action-return-types?view=aspnetcore-6.0>

Swagger

TodoItems

^

GET

/api/TodoItems

▼

POST

/api/TodoItems

▼

GET

/api/TodoItems/{id}

▼

PUT

/api/TodoItems/{id}

▼

DELETE

/api/TodoItems/{id}

▼

Swagger

POST /api/TodoItems ^

Parameters

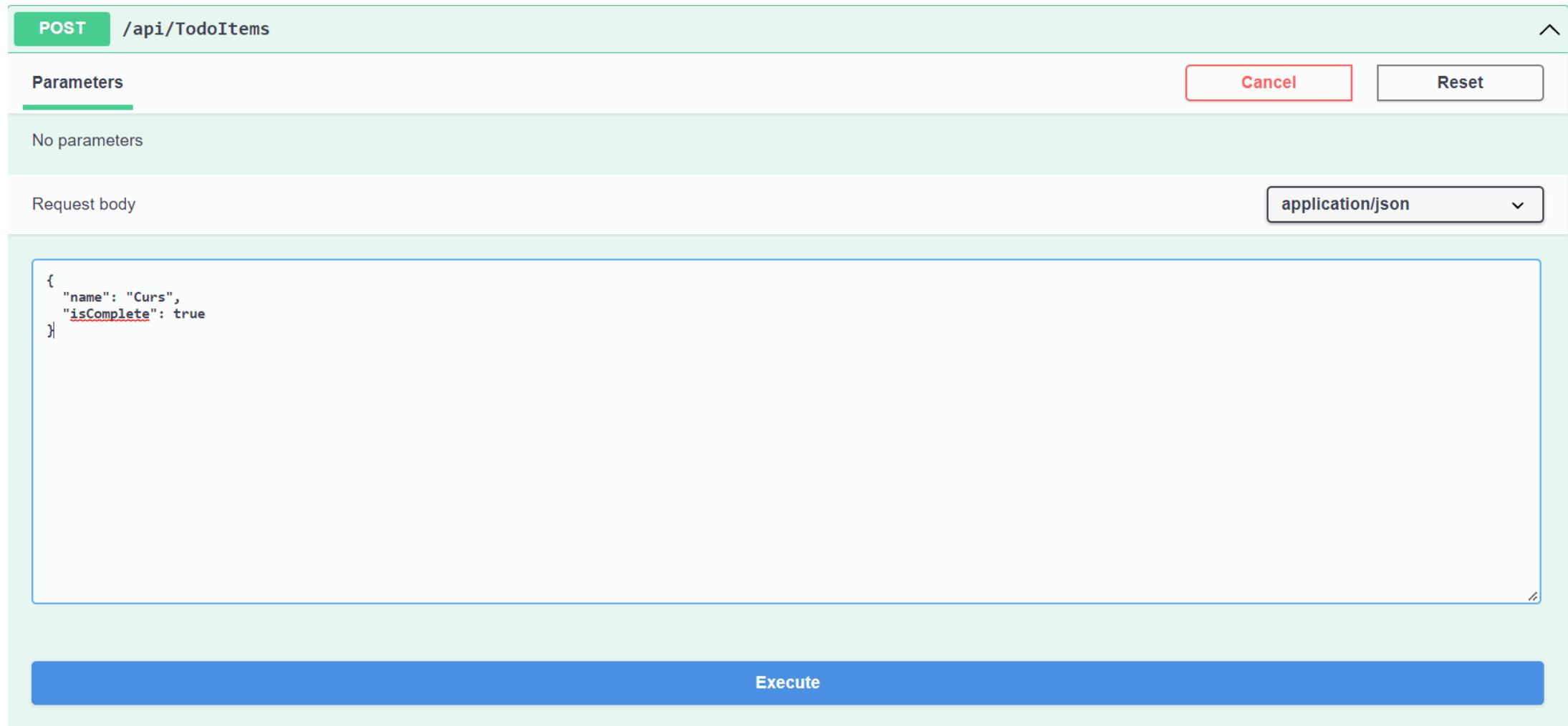
No parameters

Cancel Reset

Request body application/json ▾

```
{  
    "name": "Curs",  
    "isComplete": true  
}
```

Execute



Swagger

Curl

```
curl -X 'POST' \
  'https://localhost:7145/api/TodoItems' \
  -H 'accept: text/plain' \
  -H 'Content-Type: application/json' \
  -d '{
    "name": "Curs",
    "isComplete": true
}'
```



Request URL

```
https://localhost:7145/api/TodoItems
```

Server response

Code Details

201

Undocumented

Response body

```
{
  "id": 1,
  "name": "Curs",
  "isComplete": true
}
```



Download

Response headers

```
content-type: application/json; charset=utf-8
date: Fri,25 Nov 2022 13:35:57 GMT
location: https://localhost:7145/api/TodoItems/1
server: Kestrel
```

ASP.NET Core Web API

Documentația completă: <https://learn.microsoft.com/en-us/aspnet/core/web-api/?view=aspnetcore-6.0>

Universitatea din Bucureşti
Facultatea de Matematică și Informatică

Dezvoltarea aplicațiilor web

CTI Anul IV, Semestrul 1 2022-2023

Curs 8

Angular

- Platformă de dezvoltare aplicații web (FE)
- Folosește Typescript

TypeScript

- Limbaj de programare
- Toate funcționalitățile din Javascript
- Type checking – ajută la identificarea erorilor

TypeScript

- JS: `let titlu = "DAW";`
- TS: `let titlu: string = "DAW";`

TypeScript

- JS:

```
const student = {  
    nume: "Vasile",  
    nota: "10"  
};
```

- TS:

```
interface Student {  
    nume: string;  
    nota: number;  
}  
const student: Student = {  
    nume: "Vasile",  
    nota: 10  
};
```

TypeScript

```
const student: Student = {  
    nume: "Vasile",  
    nota: "10"  
};
```

Type 'string' is not assignable to type 'number'.

```
const student: Student = {  
    nume: "Vasile",  
    nota: 10,  
    grupa: "431"  
};
```

Type '{ nume: string; nota: number; grupa: string; }' is not assignable to type 'Student'.

Object literal may only specify known properties, and 'grupa' does not exist in type 'Student'.

TypeScript

- OOP

```
interface User {  
    name: string;  
    id: number;  
}
```

```
const user: User = new UserAccount("Murphy", 1);
```

```
class UserAccount {  
    name: string;  
    id: number;  
  
    constructor(name: string, id: number) {  
        this.name = name;  
        this.id = id;  
    }  
}
```

TypeScript

- boolean
- bigint
- null
- number
- string
- symbol
- undefined
- any
- unknown
- void
- never

TypeScript

- Tipurile custom pot fi făcute utilizând interfețe sau tipuri
- <https://www.typescriptlang.org/play?e=83#example/types-vs-interfaces>

TypeScript

- Uniunea de tipuri
 - type WindowStates = "open" | "closed" | "minimized";
 - function getLength(obj: string | string[]) {
 return obj.length;
}

TypeScript

- Generics - şabloane, adăugare de variabile la tipuri

```
interface Backpack<Type> {
```

```
    add: (obj: Type) => void;
```

```
    get: () => Type;
```

```
}
```

```
const backpack: Backpack<string>;
```

```
const object = backpack.get();
```

```
backpack.add(23);
```

```
//Argument of type 'number' is not assignable to parameter of type 'string'.
```

TypeScript

- Mai multe...

<https://www.typescriptlang.org/docs/handbook/intro.html>

Angular

Instalare:

- NodeJS

<https://nodejs.org/download/release/latest-v18.x/>

```
PS C:\Windows> node -v  
v16.13.1
```

- Npm

```
PS C:\Windows> npm -v  
8.1.2
```

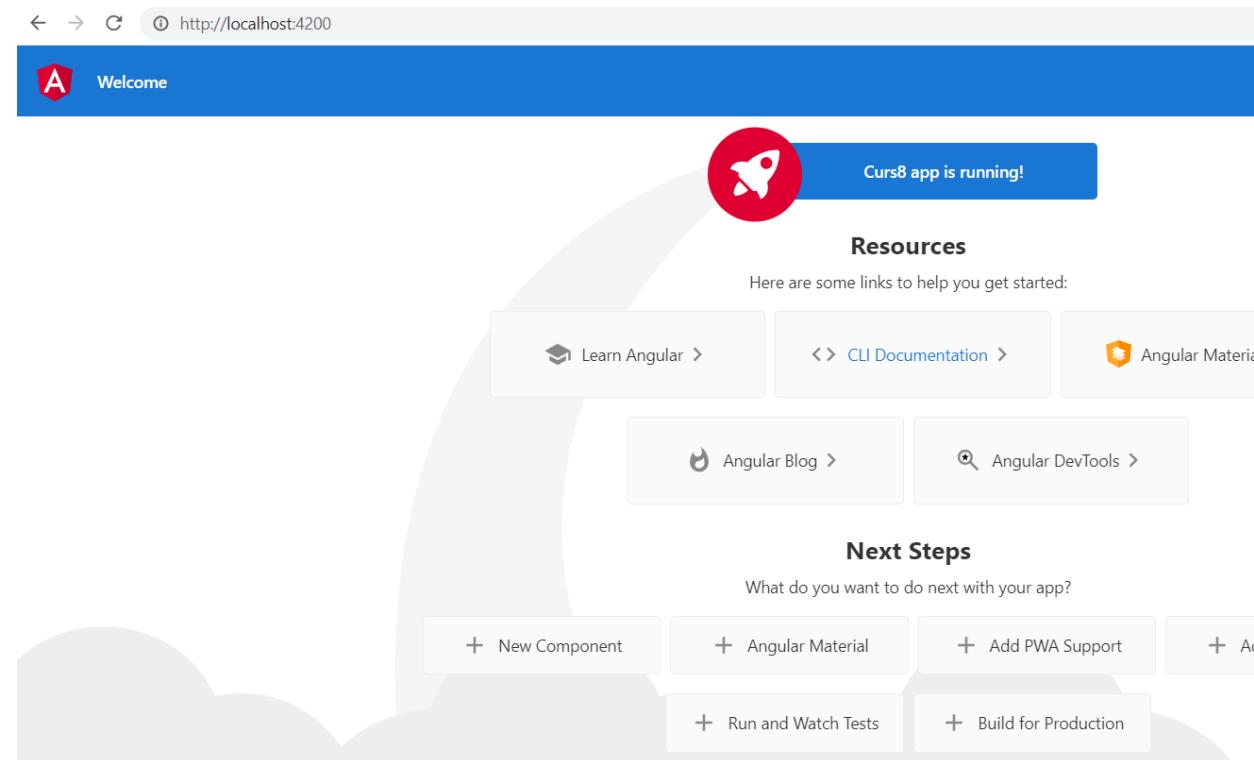
- Angular CLI: npm install -g @angular/cli

Angular

```
PS C:\Users\PSoviany\Documents> ng new Curs8
? Would you like to add Angular routing? Yes
? Which stylesheet format would you like to use? SCSS  [ https://sass-lang.com/documentation/syntax#scss
]
CREATE Curs8/angular.json (2869 bytes)
CREATE Curs8/package.json (1036 bytes)
CREATE Curs8/README.md (1059 bytes)
CREATE Curs8/tsconfig.json (901 bytes)
CREATE Curs8/.editorconfig (274 bytes)
CREATE Curs8/.gitignore (548 bytes)
CREATE Curs8/tsconfig.app.json (263 bytes)
CREATE Curs8/tsconfig.spec.json (273 bytes)
CREATE Curs8/.vscode/extensions.json (130 bytes)
CREATE Curs8/.vscode/launch.json (474 bytes)
CREATE Curs8/.vscode/tasks.json (938 bytes)
CREATE Curs8/src/favicon.ico (948 bytes)
CREATE Curs8/src/index.html (291 bytes)
CREATE Curs8/src/main.ts (214 bytes)
CREATE Curs8/src/styles.scss (80 bytes)
CREATE Curs8/src/assets/.gitkeep (0 bytes)
CREATE Curs8/src/app/app-routing.module.ts (245 bytes)
CREATE Curs8/src/app/app.module.ts (393 bytes)
CREATE Curs8/src/app/app.component.html (23115 bytes)
CREATE Curs8/src/app/app.component.spec.ts (1070 bytes)
CREATE Curs8/src/app/app.component.ts (210 bytes)
CREATE Curs8/src/app/app.component.scss (0 bytes)
/ Packages installed successfully.
```

Angular

```
PS C:\Users\PSoviany\Documents> cd .\Curs8\  
PS C:\Users\PSoviany\Documents\Curs8> ng s
```



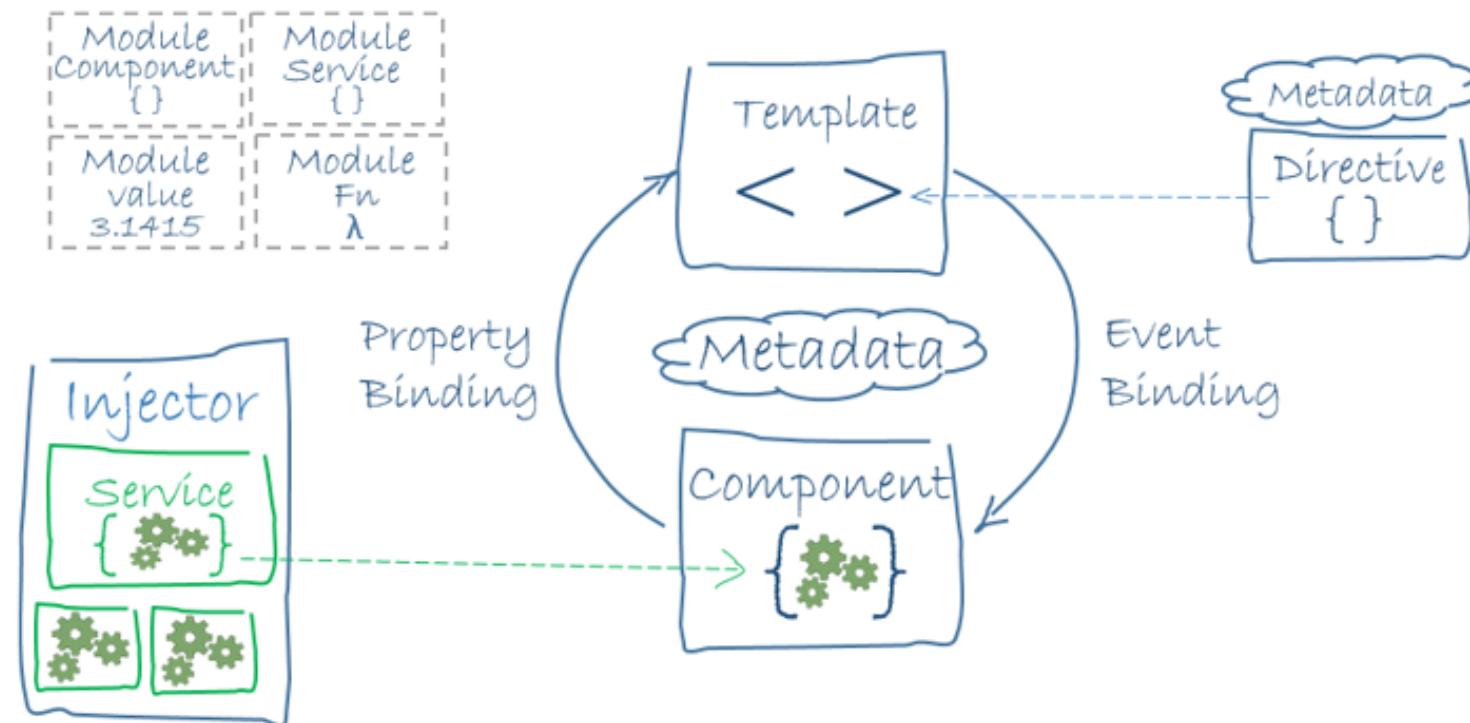
Angular

- Module – unitate logică, câte un modul pentru fiecare funcționalitate, AppModule este modulul inițial care pornește aplicația
- Componente – logică și template, ce sunt combinate pentru a genera DOM-ul paginii; pot fi reutilizate
- Template – combinație de cod HTML și logică, informațiile pot fi transmise între template și codul componentei prin binding

Angular

- Servicii – utilizate pentru date și logică ce nu sunt specifice unui singur view; sunt injectate folosind dependency injection.
- Rutarea – maparea dintre codul dezvoltat și URL-urile la care se regăsește informația se face folosind RouterModule. Mecanismul de rutare permite încărcarea modelelor necesare folosind lazy loading.

Angular



Universitatea din Bucureşti
Facultatea de Matematică și Informatică

Dezvoltarea aplicațiilor web

CTI Anul IV, Semestrul 1 2022-2023

Curs 9

Module Angular

- Asigură modularitate
 - Pot fi încărcate doar când este necesar (lazy loaded)
 - Permit adăugarea funcționalităților din biblioteci externe
-
- Feature Modules – module grupate după funcționalitate
 - Root Module – modulul principal, rădăcina aplicației

Clasă decorată cu
NgModule și care
conține metadate
corespunzătoare.

ng generate module
NumeModul

```
import { NgModule } from '@angular/core';
import { FormsModule } from '@angular/forms';
import { BrowserModule } from '@angular/platform-browser';

import { AppRoutingModule } from './app-routing.module';
import { AppComponent } from './app.component';
import { HelloComponent } from './hello/hello.component';

@NgModule({
  declarations: [
    AppComponent,
    HelloComponent
  ],
  imports: [
    BrowserModule,
    AppRoutingModule,
    FormsModule
  ],
  exports: [HelloComponent],
  providers: [],
  bootstrap: [AppComponent]
})
export class AppModule { }
```

Module Angular – Declarații

- Elementele (componente, directive, pipe-uri) care aparțin unui modul trebuie adăugate în vectorul Declarations
- Orice element poate apartine unui singur modul
- Elementele sunt private – pot fi accesate doar din interiorul modulului; pentru utilizarea în afara modulului, ele trebuie exportate

```
declarations: [
  AppComponent,
  HelloComponent
],
```

Module Angular – Importuri

- Modulele (Angular, proprii, externe) necesare pentru afişarea corespunzătoare a informaţiei de către componente trebuie adăugate în vectorul Imports
- Prin importarea unui Modul se dobândeşte acces doar asupra elementelor sale exportate

```
imports: [
  BrowserModule,
  AppRoutingModule,
  FormsModule
],
```

Module Angular – Exporturi

- Elementele (componente, directive, pipe-uri, module) care vrem să fie disponibile în exteriorul modulului trebuie adăugate în vectorul Exports
- Pot fi exportate și elemente neimportate

```
exports: [HelloComponent],
```

Module Angular – Bootstrap

- Componența de pornire a aplicației trebuie adăugată în vectorul Bootstrap
- Vectorul Bootstrap trebuie populat doar în modulul rădăcină (AppModule)

```
bootstrap: [AppComponent]
```

Module Angular – Providers

- Vectorul Providers este folosit pentru înregistrarea serviciilor utilizate. Această abordare nu este recomandată.

```
providers: [],
```

Componente Angular

- Pagini sau părți de pagini web
- Sunt alcătuite din template (UI), cod (typescript) și metadate (date suplimentare)

Componente Angular

Clasă decorată cu Component și care conține metadate corespunzătoare.

Conține proprietăți pentru manevrarea informațiilor și metode pentru diferite funcționalități.

ng generate component NumeComponent

<https://angular.io/api/core/Component>

```
import { Component } from '@angular/core';

@Component({
  selector: 'app-root',
  templateUrl: './app.component.html',
  styleUrls: ['./app.component.scss']
})
export class AppComponent {
```

Componente Angular - Template

- cod HTML
- data binding
- directive
- sub-componente

Componente Angular - Binding

Interpolare

- Conexiune unidirecțională (one-way binding), din clasă spre template
- {{ expression }}
- Utilizat pentru afișarea de informații
- Permite efectuarea de operații simple, precum concatenarea, iar contextul este componența

Componente Angular - Binding

```
public valoare: string = "Gheorghe";
```

```
<p class="hello">Bun venit, {{ valoare }}</p>
```

← → C ⓘ http://localhost:4200

Acesta este primul nostru proiect in Angular!

Bun venit, Gheorghe

Componente Angular - Binding

Property binding

- Conexiune unidirecțională (one-way binding), din clasă spre template
- [proprietate] = “expresie”
- Utilizat pentru setarea proprietăților elementelor cu o expresie
- Spre deosebire de interpolare care funcționează doar ca string, prin property binding se păstrează tipul datelor

Componente Angular - Binding

```
public imgLink: string = "https://angular.io/assets/images/logos/angular/logo-nav@2x.png";
```

```
<img [src]="imgLink">  
Submit</button>
```

Submit

localhost:4200 says

Buna tuturor!

OK

Componente Angular - Binding

Two-way binding

- Conexiune bidirectională între template și cod
- Folosește directiva ngModel – importați FormsModule în Modul
- [(ngModel)] = “variabila”
- Ajuta la preluarea datelor dintr-un formular

Componente Angular - Binding

```
public valoare: string = "Ion";  
  
public trimite(): void {  
  alert("Am apasat butonul " + this.valoare);  
}
```

```
<input [(ngModel)]="valoare" type="text" /><br/>  
<button (click)="trimite()">Submit</button>
```


localhost:4200 says

Am apasat butonul Ion Modificat



Componente Angular - Pipe-uri

- Customizează afişarea informației
- Pipe-uri predefinite: uppercase, date, currency, ...
- Pot fi construite pipe-uri custom

Componente Angular - Pipe-uri

```
public premiu = "1000";
```

```
<p>Aveti sansa sa castigati un premiu in valoare de  
{{ premiu | currency:'USD':'symbol':'4.2-2' }}</p>
```

← → ⌂ ⓘ http://localhost:4200

Acesta este primul nostru proiect in Angular!

Aveti sansa sa castigati un premiu in valoare de \$1,000.00

Componente Angular - Pipe-uri

```
import { Pipe, PipeTransform } from "@angular/core";

@Pipe({
  name: 'convertToSpaces'
})

export class ConvertToSpacesPipe implements PipeTransform {
  transform(value: string, char: string): string {
    return value.replaceAll(char, ' ')
  }
}
```

```
public mesaj = "Acesta-este-un-test.";
```

```
<p>{{ mesaj | convertToSpaces:'-' }}</p>
```

Acesta este un test.

Componente Angular - Directive

- Component
- Directive structurale (*ngFor, *ngIf – modifică DOMul)
- Directive atribut (NgModel, NgClass – modifică aspectul sau comportamentul unui element)
- <https://angular.io/guide/attribute-directives>

Componente Angular - Directive

- Se pot include componente (subcomponente) cu ajutorul selectorului acestora

```
@Component({
  selector: 'app-hello',
  templateUrl: './hello.component.html',
  styleUrls: ['./hello.component.scss']
})
```

```
<app-hello></app-hello>
```

Componente Angular - Directive

- Se pot afișa elemente HTML în funcție de anumite condiții

```
<p *ngIf="premiu">Aveti sansa sa castigati un premiu in valoare de  
{{ premiu | currency:'USD':'symbol':'4.2-2' }}</p>
```

```
public premiu = "1000";
```

Acesta este primul nostru proiect in Angular!

Aveti sansa sa castigati un premiu in valoare de \$1,000.00

```
public premiu = "|";
```

Acesta este primul nostru proiect in Angular!

Componente Angular - Directive

- Se poate modifica DOM-ul, iterând prin iterables

```
public mesaje = [  
    "Buna ziua",  
    "Acesta este un test",  
    "Iteram cu ngFor",  
    "Prinr-o lista de mesaje"  
];
```

```
<p *ngFor="let mesaj of mesaje; let index = index">  
    # {{index + 1}}. {{ mesaj }}  
</p>
```

Acesta este primul nostru proiect in Angular!

1. Buna ziua
2. Acesta este un test
3. Iteram cu ngFor
4. Prinr-o lista de mesaje

Componente Angular

- Ciclul de viață al unei componente Angular începe când este instanțiată clasa componentei, continuă cu mecanismul de detectie al modificărilor și se încheie când componenta este distrusă.
- Lifecycle hooks – metode prin care se poate executa cod la anumite momente din ciclul de viață al componentei.

Componente Angular

Lifecycle Hooks:

- **ngOnChanges()** – la modificare unui Input()
- **ngOnInit()** – după crearea componentei și primul OnChanges, dacă este cazul
- ngDoCheck()
- ngAfterContentInit()
- ngAfterContentChecked()
- ngAfterViewInit()
- ngAfterViewChecked()
- **ngOnDestroy()** – înainte de distrugerea componentei

<https://angular.io/guide/lifecycle-hooks>

Componente Angular

- Ciclul de viață al unei componente Angular începe când este instantiată clasa componentei, continuă cu mecanismul de detectie al modificărilor și se încheie când componenta este distrusă.
- Lifecycle hooks – metode prin care se poate executa cod la anumite momente din ciclul de viață al componentei.

Componente Angular

```
export class HelloComponent implements OnInit, OnChanges {
  ngOnInit(): void {
    console.log("Mesaj din OnInit");
  }

  ngOnChanges(changes: SimpleChanges): void {
    console.log("Mesaj din OnChanges");
  }
}
```

```
-----  
disabled, Live Reloading enabled, Progress disabled, Overlay enabled.  
Mesaj din OnInit                                     hello.component.ts:10  
Angular is running in development mode. Call enableProdMode() to core.mjs:25811  
enable production mode.  
⚠ DevTools failed to load source map: Could not load content for chrome-extension://  
cfhdoihkhhkhlhkdaihdcddilifddh/browser-polyfill.js.map. System error.
```

Componente Angular

- Subcomponentele pot fi introduse folosind selectorul componentei copil.

```
<app-hello></app-hello>
```

- Prin @Input() anuntăm componentă că proprietatea decorată poate primi valori de la părinte.

```
@Input() public mesaje = [  
    "Buna ziua",  
    "Acesta este un test",  
    "Iteram cu ngFor",  
    "Printr-o lista de mesaje"  
];
```

- Transmiterea datelor de la părinte la copil se poate face folosind @Input() și property binding.

Componente Angular

```
<app-hello></app-hello>
```

```
@Input() public mesaje = [  
    "Buna ziua",  
    "Acesta este un test",  
    "Iteram cu ngFor",  
    "Prinr-o lista de mesaje"  
];
```

Acesta este primul nostru proiect in Angular!

1. Buna ziua

2. Acesta este un test

3. Iteram cu ngFor

4. Prinr-o lista de mesaje

Componente Angular

```
export class AppComponent {  
  public mensajeDinParinte: any[] = [  
    "Buna seara",  
    "Acesta este un test nou"  
  ];  
}
```

```
<app-hello [mensaje]="mensajeDinParinte"></app-hello>
```

Acesta este primul nostru proiect in Angular!

1. Buna seara

2. Acesta este un test nou

Componente Angular

```
export class HelloComponent implements OnInit, OnChanges {
  ngOnInit(): void {
    console.log("Mesaj din OnInit");
  }

  ngOnChanges(changes: SimpleChanges): void {
    console.log("Mesaj din OnChanges");
  }
}
```

```
[webpack-dev-server] Server started: Hot Module Replacement      polyfills.js:1
disabled, Live Reloading enabled, Progress disabled, Overlay enabled.
Mesaj din OnChanges                                hello.component.ts:14
Mesaj din OnInit                                    hello.component.ts:10
Angular is running in development mode. Call enableProdMode() to  core.mjs:25811
enable production mode.
```

Componente Angular

- Pentru a transmite informații de la componenta copil la părinte, se folosește directiva @Output().
- În copil se creează un eveniment de tip EventEmitter și decorat cu Output, pentru a trimite (emite) informații.
- În template-ul părintelui se adaugă evenimentul declarat în copil folosind event binding.

Componente Angular

```
@Output() public mesajEvent = new EventEmitter<string>

public alertaCopil(mesaj: string): void {
  this.mesajEvent.emit(mesaj);
}

<button (click)="alertaCopil(mesaj)">Alerta mesaj</button>
```

Copil

```
public mesajParinte(mesaj: string) {
  alert(mesaj);
}
```

Părinte

```
<app-hello [mesaje]="mesajeDinParinte"
(mesajEvent)="mesajParinte($event)"></app-hello>
```

Componente Angular

Acesta este primul nostru proiect in Angular!

1. Buna seara Alerta mesaj

2. Acesta este un test nou Alerta mesaj

localhost:4200 says

Acesta este un test nou



Componente Angular

- Proprietățile pot fi declarate cu getere și setere
- Ele vor fi folosite cu aceeași sintaxă ca o variabilă obișnuită, dar se va executa cod suplimentar la fiecare acțiune de get și set

Componente Angular

```
private _exemplu: string = "";
public get exemplu(): string {
  console.log("Am apelat get exemplu");
  return this._exemplu;
}
public set exemplu(val: string) {
  console.log("Am apelat set exemplu");
}

ngOnInit(): void {
  console.log("Mesaj din OnInit");
  this.exemplu = "test";
  let ex = this.exemplu;
}
```

Mesaj din OnChanges	hello.component.ts:25
Mesaj din OnInit	hello.component.ts:19
Am apelat set exemplu	hello.component.ts:15
Am apelat get exemplu	hello.component.ts:11

Angular is running in development mode. Call enableProdMode() to [core.mjs:25811](#) enable production mode.