Programação Web

Aulas Teóricas – Capítulo 1 – 1.1

1º Semestre - 2023/2024

Departamento de Engenharia Informática e de Sistemas
Instituto Superior de Engenharia de Coimbra/Instituto Politécnico de Coimbra

Programação Web

Introdução ao C# – Conceitos Básicos

Departamento de Engenharia Informática e de Sistemas
Instituto Superior de Engenharia de Coimbra/Instituto Politécnico de Coimbra

C# vs ASP.Net Core

- **C# é uma linguagem de programação**
  - **Actualmente está na Versão 11 (Novembro de 2022)**
- O ***ASP.Net Core*** é um *framework* para desenvolvimento de aplicações para a *Web* e suporta várias linguagens de programação:
  - *C#*
  - *F#*
  - *VB.Net* (parcialmente)…

Programação Web – 2023/2024 – LEI D+CE+PL – ISEC/IPC @ 3

isec Engenharia DEIS - Instituto Superior de Engenharia de Coimbra

.Net

- *CLR* (***C**ommon **L**anguage **R**untime*)

Codificar neste ambiente...

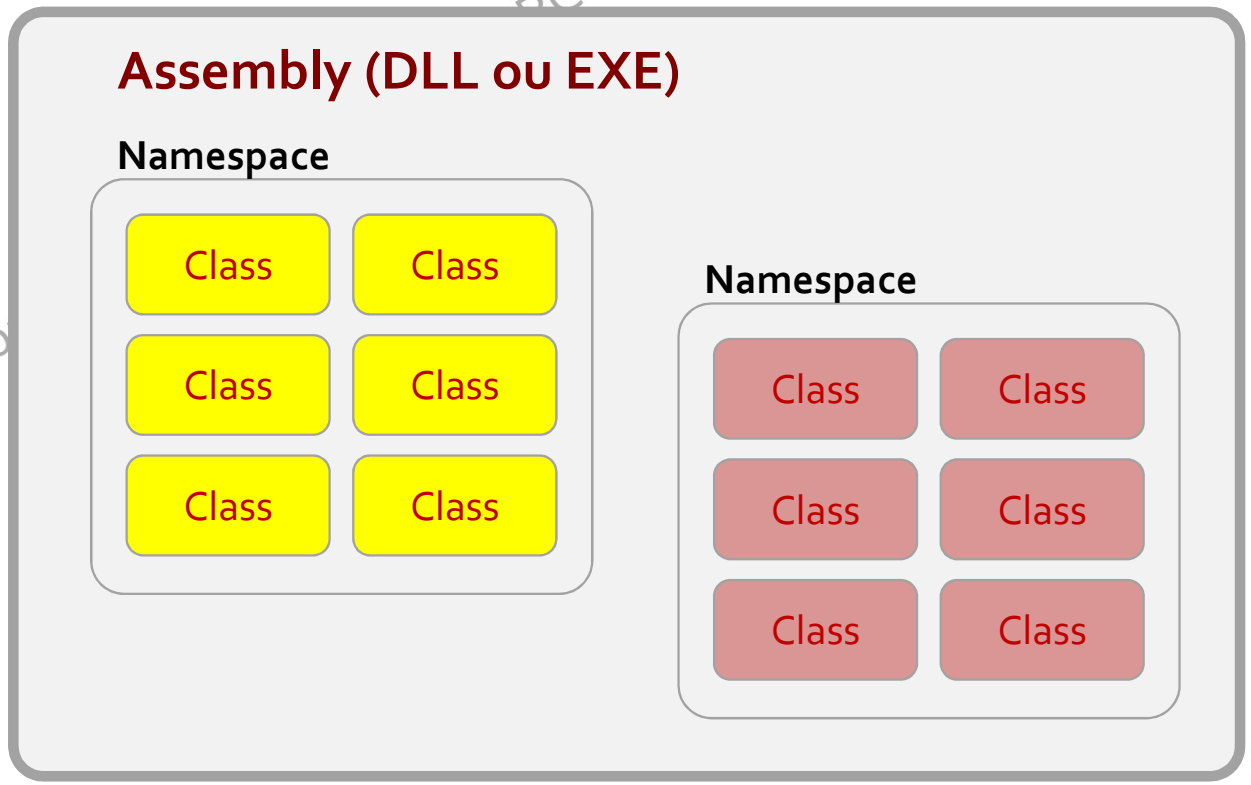
Executar neste....

  - Conversão da Linguagem Intermédia (*IL Code*) para código máquina - *Just-in-time(JIT) compilation.*
  - Integra serviços adicionais como gestão de memória, tratamento de excepções, *garbage colector*, segurança, gestão de *thread*,…
- ***CLS*, *CTS* ...**
- *Class Library*

Programação Web – 2023/2024 – LEI D+CE+PL – ISEC @JB

Arquitetura de aplicações .NET

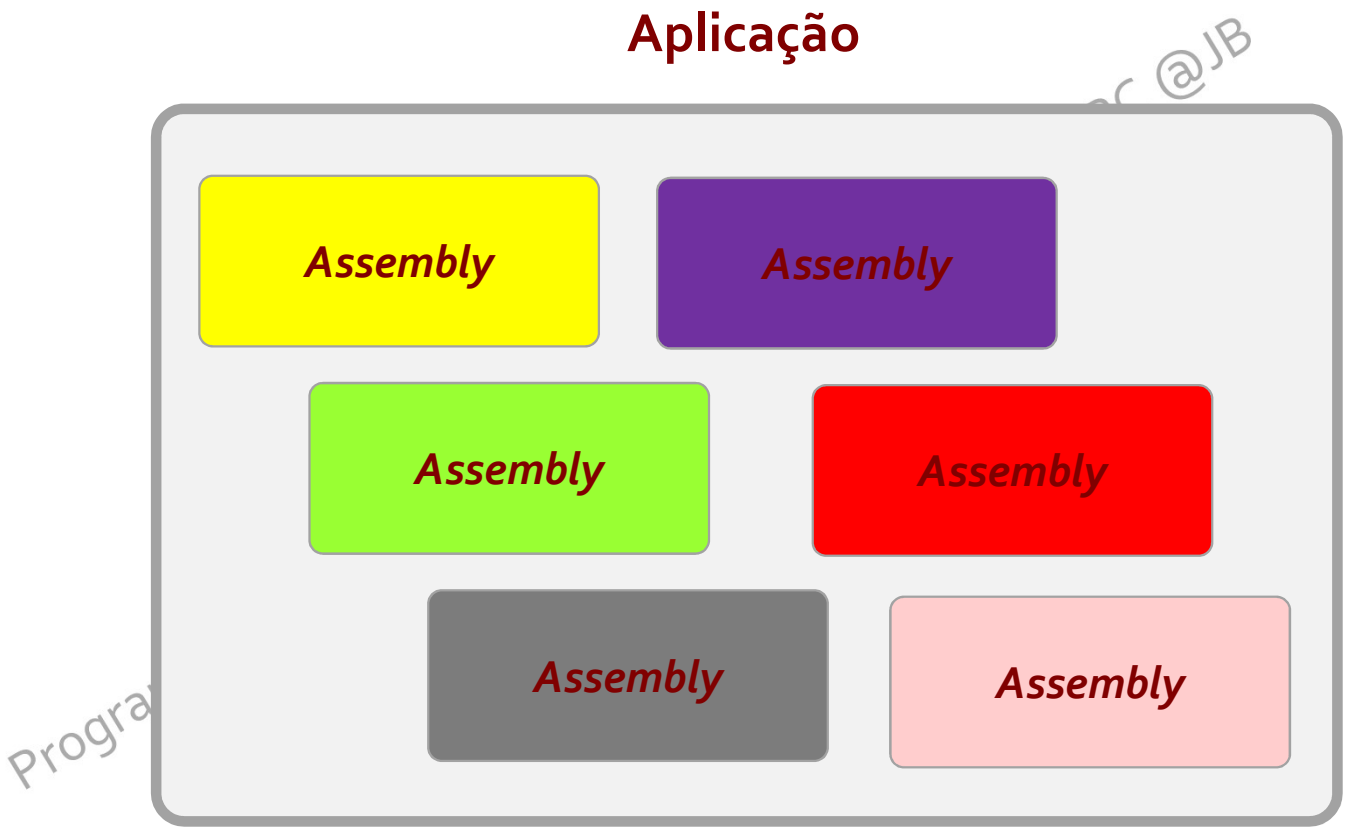
- **Namespaces** permitem organizar as classes
  - Bases de dados
  - Trabalhar com gráficos e imagens
  - Segurança
  - …



Programação Web – 2023/20
PC @

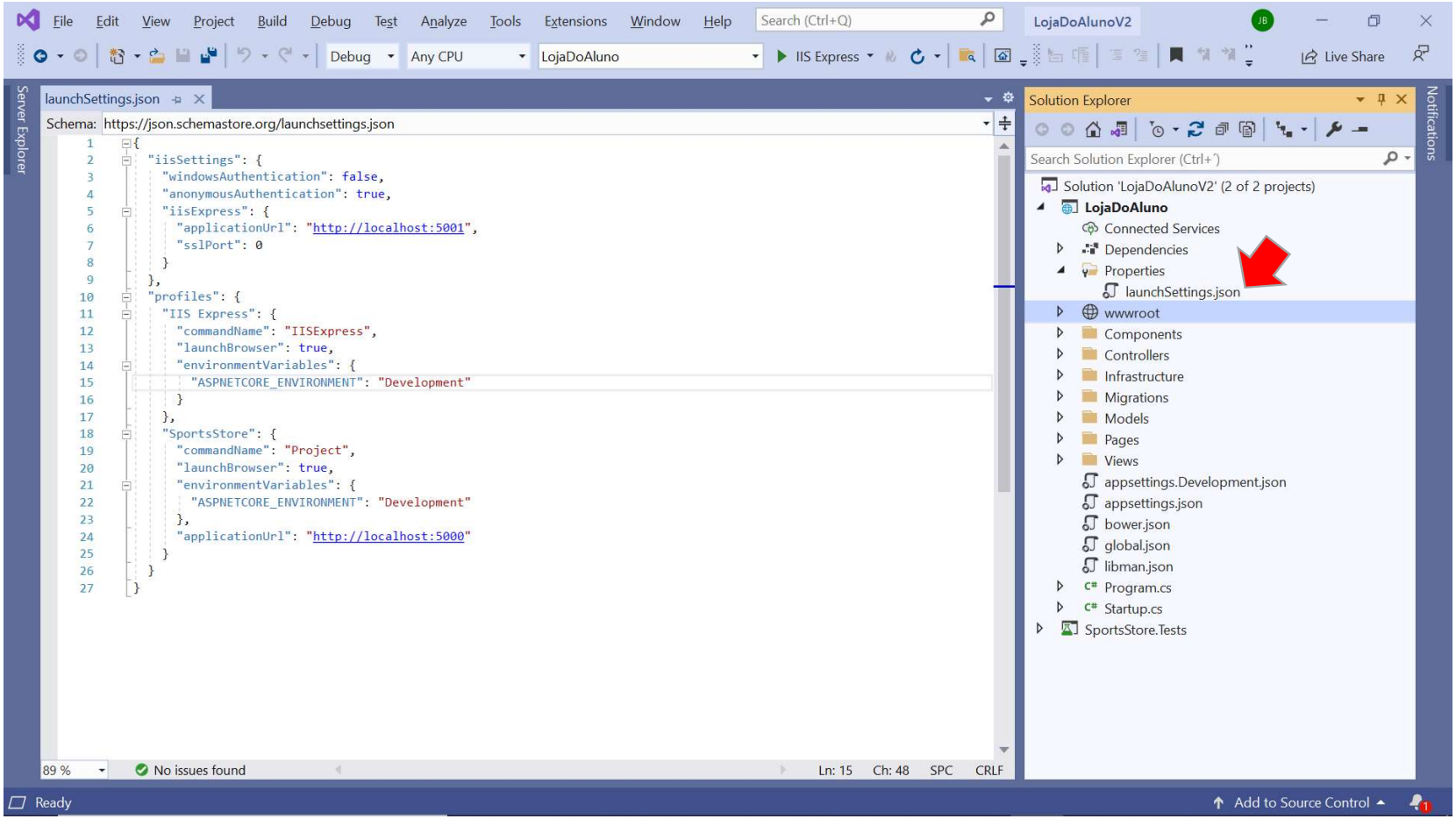
isec Engenharia DEIS - Instituto Superior de Engenharia de Coimbra

Arquitetura de aplicações .NET



isec Engenharia DEIS - Instituto Superior de Engenharia de Coimbra

Arquitetura de aplicações ASP.Net Core



isec Engenharia DEIS - Instituto Superior de Engenharia de Coimbra

Arquitetura de aplicações ASP.Net Core

File Edit View Project Build Debug Test Analyze Tools Extensions Window Help Search (Ctrl+Q) LojaDoAlunoV2

Debug Any CPU LojaDoAluno IIS Express Live Share

Server Explorer

launchSettings.json

Schema: https://json.schemastore.org/launchsettings.json

```
1  {
2    "iisSettings": {
3      "windowsAuthentication": false,
4      "anonymousAuthentication": true,
5      "iisExpress": {
6        "applicationUrl": "http://localhost:5001",
7        "sslPort": 0
8      }
9    },
10   "profiles": {
11     "IIS Express": {
12       "commandName": "IISExpress",
13       "launchBrowser": true,
14       "environmentVariables": {
15         "ASPNETCORE_ENVIRONMENT": "Development"
16       }
17     },
18     "SportsStore": {
19       "commandName": "Project",
20       "launchBrowser": true,
21       "environmentVariables": {
22         "ASPNETCORE_ENVIRONMENT": "Development"
23       },
24       "applicationUrl": "http://localhost:5000"
25     }
26   }
27 }
```

89 % No issues found Ln: 15 Ch: 48 SPC CRLF

Solution Explorer

Search Solution Explorer (Ctrl+')

- Solution 'LojaDoAlunoV2' (2 of 2 projects)
  - LojaDoAluno
    - Connected Services
    - Dependencies
      - Analyzers
        - Microsoft.AspNetCore.Analyzers
        - Microsoft.AspNetCore.Components.Analy
        - Microsoft.AspNetCore.Mvc.Analyzers
        - Microsoft.EntityFrameworkCore.Analyzers
      - Frameworks
        - Microsoft.AspNetCore.App
        - Microsoft.NETCore.App
      - Packages
        - Microsoft.AspNetCore.Identity.EntityFrame
        - Microsoft.EntityFrameworkCore.Design (3.
        - Microsoft.EntityFrameworkCore.SqlServer
        - Microsoft.EntityFrameworkCore.SqlServer.
        - Microsoft.Extensions.Identity.Core (5.0.10)
    - Properties
      - launchSettings.json
    - wwwroot
    - Components
    - Controllers
    - Infrastructure
    - Migrations
    - Models
    - Pages
    - Views

Ready Add to Source Control

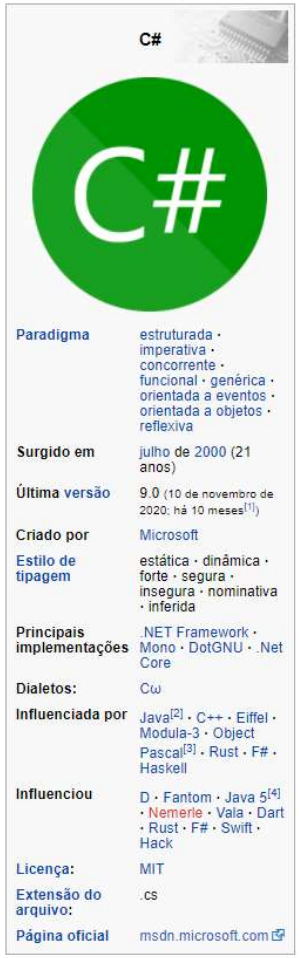
isec Engenharia DEIS - Instituto Superior de Engenharia de Coimbra

Introdução ao C#

- Desenvolvida pela Microsoft
- Baseada no C (Mistura de C++ e Java)
  - Orientada a Objetos
- Principal linguagem de programação para *framework* .NET
- Criador: Anders Hejlsberg
- Introduzido C#1.0 com VS2002, atualmente vai na versão C# 11.0 com VS2022 (ASP.Net Core 6)
- Possui um elevado nível de abstração

Programação Web – 2023/2024 – LEI-D+CE+PL – ISEC/IPC @JB

Introdução ao C#



C#	
Paradigma	estruturada · imperativa · concorrente · funcional · genérica · orientada a eventos · orientada a objetos · reflexiva
Surgido em	julho de 2000 (21 anos)
Última versão	9.0 (10 de novembro de 2020; há 10 meses[1])
Criado por	Microsoft
Estilo de tipagem	estática · dinâmica · forte · segura · insegura · nominativa · inferida
Principais implementações	.NET Framework · Mono · DotGNU · .Net Core
Dialetos:	Cω
Influenciada por	Java[2] · C++ · Eiffel · Modula-3 · Object Pascal[3] · Rust · F# · Haskell
Influenciou	D · Fantom · Java 5[4] · Nemerle · Vala · Dart · Rust · F# · Swift · Hack
Licença:	MIT
Extensão do arquivo:	.cs
Página oficial	msdn.microsoft.com

Programação Web – 2023/2024 +PL – ISEC/IPC @JB

isec Engenharia DEIS - Instituto Superior de Engenharia de Coimbra

Características Gerais

- Todos os ficheiros C# tem extensão *.cs*
- C# é *case sensitive*

```
Console.WriteLine("Programação em C#");

Console.Writeline("Programação em C#");
console.WriteLine("Programação em C#");
```

- O compilador C# ignora espaços adicionais
  - Sequencia de espaços, carateres das tabs ,*cariage return,…*

Programação Web – ISEC/IPC @ JB

Características Gerais

- Cada *statement* C# deve terminar com um ponto e vírgula (;)
  - Podem existir vários *statements* numa só linha
    - Por questões de legibilidade, muda-se de linha no final de cada *statement*
- Um bloco de código C# está delimitado por { e }
  - Pode conter qualquer número de linhas de código ou nenhum
  - Estes não necessitam de ser acompanhados pelo ‘;’

Programação Web – 2023/2024 – LEI D+CE+PL – ISEC/IPC @ JB

isec Engenharia DEIS - Instituto Superior de Engenharia de Coimbra

Características Gerais

- Comentários
  - Comentar uma só linha: // …
  - Comentar mais do que uma linha: /* ….. ….. */

LEI D+CE+M – ISEC/IPC @JB

```
int x; // Isto é um comentário
```

```
/* Isto..
      .....
       ...Também é um comentário! */
```

isec Engenharia DEIS - Instituto Superior de Engenharia de Coimbra

Estrutura Básica de um programa em C#

System é um namespace

Classes organizadas em *namespaces*

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace CSharpExampleConsole
{
    0 references
    class Program
    {
        0 references
        static void Main(string[] args)
        {
            Console.WriteLine("Hello World!");
        }
    }
}
```

Console é uma classe do namespace System

isec Engenharia DEIS - Instituto Superior de Engenharia de Coimbra

Estrutura Básica de um programa em C#

```
using System;
using System.Collections.Generic;
using
using
using

namesp
{
    0
    cl
    {

        static void Main(string[] args)
        {
            Console.WriteLine("Hello World!");
        }
    }
}
```

- **Contém unicamente um método chamado *Main*.**
- ***Main* contém uma linha de código que escreve "Hello, World!"**
- **O método que efetua esta ação chama-se *WriteLine***
- **O método *WriteLine* pertence ao objeto *System.Console***
- **A palavra chave "*static*" especifica que o método *Main* pode ser chamado mesmo que não haja uma instância da classe.**

@JB

isec Engenharia DEIS - Instituto Superior de Engenharia de Coimbra

Características Gerais

- #region
  - Permite comprimir/expandir o código de forma a ocultar/apresentar os detalhes de um bloco

LD+CE+PL – ISEC/IPC @JB

Progra

```
#region Using Diretivas
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
#endregion
```

isec Engenharia DEIS - Instituto Superior de Engenharia de Coimbra

Programação Web

C#: Variáveis

Departamento de Engenharia Informática e de Sistemas
Instituto Superior de Engenharia de Coimbra/Instituto Politécnico de Coimbra

C#: Literais

- ***Reais***
  - Usado para valores do tipo ***float***, ***double*** e ***decimal***
  - Podem ter notação exponencial
  - O sufixo ‘f’ ou ‘F’ significa ***float***
  - O sufixo ‘d’ ou ‘D’ significa ***double***
  - O sufixo ‘m’ ou ‘M’ significa ***decimal***
  - A interpretação por omissão é double

```
float realNumber = 12.5f;
```

Programação Web – 2023/2024 – LEID+CE+PL – ISEC/IPC @JB

C#: Literais - *Escape* carateres

- ***String***

Sequência de *Escape*	Carater produzido
\'	Pelica
\"	Aspas
\\	Barra invertida
\0	Null
\a	Alert (origina um *beep*)
\b	Backspace
\f	*Form feed*
\n	Mudança de linha
\r	*Carriage return*
\t	Tabulação horizontal
\v	Tabulação vertical

Programação Web –

isec Engenharia DEIS - Instituto Superior de Engenharia de Coimbra

C#: Literais

- ***String*** (cont.)
  - Pode conter o prefixo @ (*verbatim strings literals)*, permitindo reduzir as sequências de *espace*

D+CE+PL – ISEC/IPC @JB

```
string citacao= "\"Caros, Aula de PW!\", ....";
string caminho = "C:\\3Ano\\PW\\exemplo.exe";
citacao = @"""Caros, Aula de PW!"", ....";
caminho = @"C:\3Ano\PW\exemplo.exe";
string str = @"string exemplo";
```

Verbatim strings

Progr

isec Engenharia DEIS - Instituto Superior de Engenharia de Coimbra

Tipos de Dados em C#: Exemplo

```
int meuInteiro;
string minhaString;
```

```
meuInteiro = 30;
minhaString = "\"meuInteiro\" igual a ";
```

```
Console.WriteLine($"{minhaString} {meuInteiro}");
Console.ReadKey();
```

Atribuição de valores fixos
Valores literais

Escape carateres

Característica do C#6
String Interpolation

```
"meuInteiro" igual a  30
```


isec Engenharia DEIS - Instituto Superior de Engenharia de Coimbra

Tipos de Dados em C#: Dinâmicos

- Especificados com a palavra chave *dynamic*
- Podem conter qualquer valor (*string*, *número*, *objeto*, *referência* a uma função/método)
- As operações são avaliadas em *runtime*

```
dynamic val1 = 1;
dynamic val2 = 2;
Console.WriteLine(val1 + val2); // Escreve 3
val1 = "1";
val2 = 2;
Console.WriteLine(val1 + val2); // Escreve 12
```

Programação Web – 2023/2024 – EI D+CE PL – ISEC/IPC @ 23

isec Engenharia DEIS - Instituto Superior de Engenharia de Coimbra

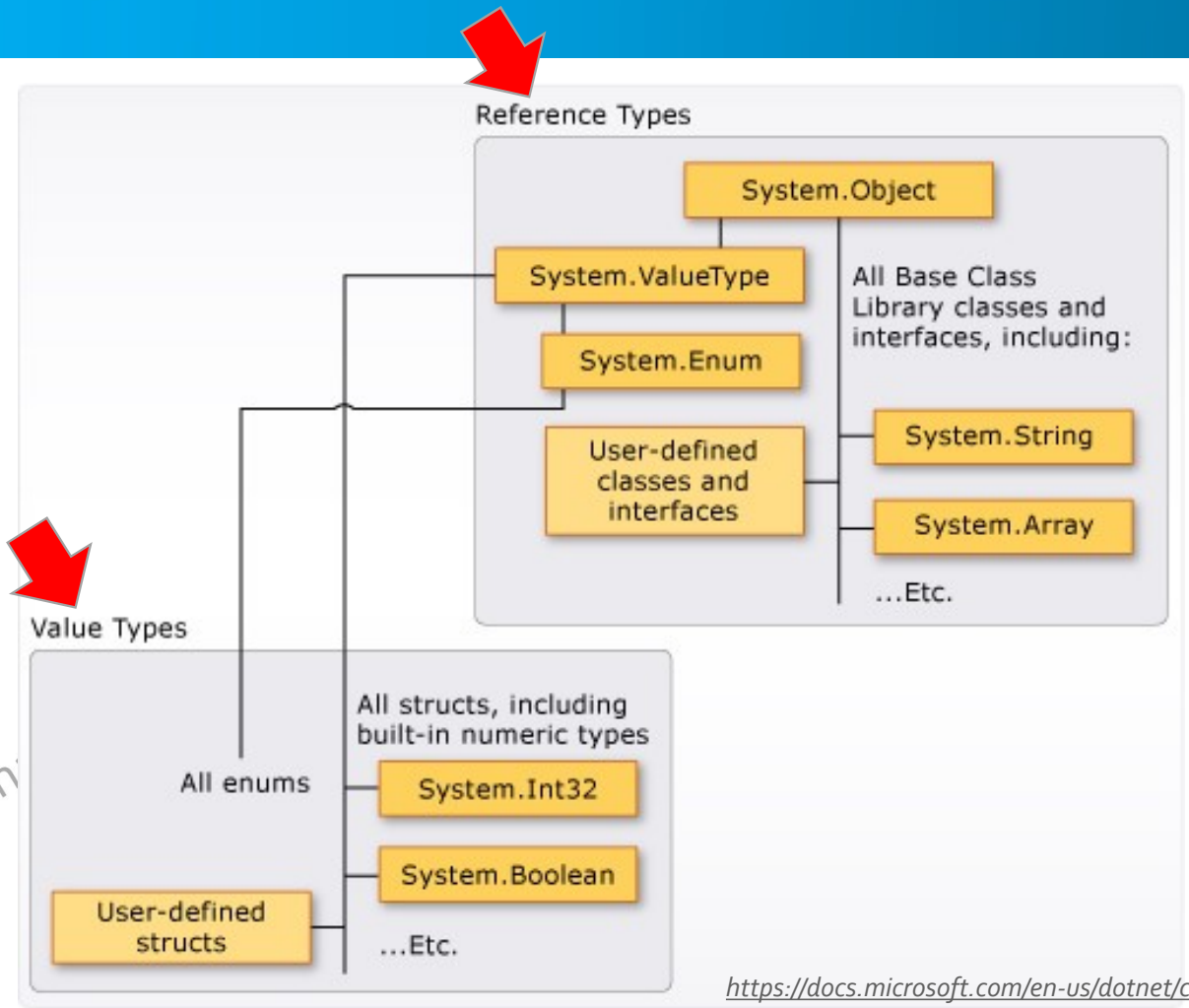
Tipos de Dados em C#

- Tipos de dados em C# são categorizados pela forma como são armazenados em memória. Existem dois tipos:
  - Valor
    - ***bool, byte, char, decimal, double, enum, float, int, long, sbyte, short, struct, uint, ulong, ushort***
  - Referência
    - ***String, Arrays, Classes e Delegates*** *(Veremos mais adiante)*

Programação Web – 2023/2024 – LEI D+CE+PL – ISEC/IPC @ 19

isec Engenharia DEIS - Instituto Superior de Engenharia de Coimbra

Tipos de Dados em C#



https://docs.microsoft.com/en-us/dotnet/csharp/programming-guide/types/

Program

isec Engenharia DEIS - Instituto Superior de Engenharia de Coimbra

Programação Web

C#: *Enums, Structs, Arrays*

Departamento de Engenharia Informática e de Sistemas
Instituto Superior de Engenharia de Coimbra/Instituto Politécnico de Coimbra

C#: *Enumerations*

- Tipo de variável que permite declarar uma lista de constantes, podendo assim ser utilizados de uma forma *human-readable*
- Muitas vezes referidos como ***enums***
- Sintaxe:

```
enum <nome>
{
        lista
};
```

  - *nome*: especifica o nome do tipo de enumeração
  - *lista*: É uma lista de identificadores separados por vírgula

Programação Web – 2023/2024 – LEID+CE+PL – ISEC/IPC @JB

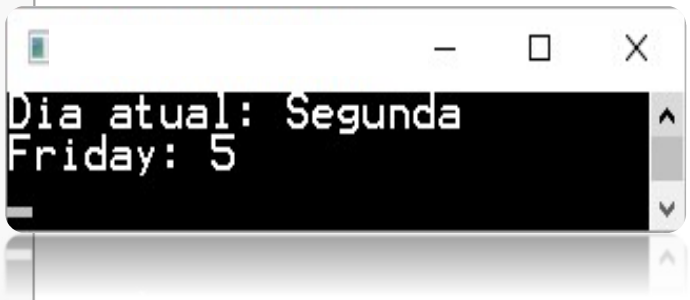
isec Engenharia DEIS - Instituto Superior de Engenharia de Coimbra

C#: *Enumerations* - Exemplo

```
enum DiasSemana : int { Segunda=1,
                        Terca=2,
                        Quarta=3,
                        Quinta=4,
                        Sexta=5,
                        Sabado=7,
                        Domingo=8
                      };
static void Main(string[] args)
{
    DiasSemana diaSemanaAtual = DiasSemana.Segunda;
    int FimSemana = (int)DiasSemana.Sexta;
    Console.WriteLine("Dia atual: {0}", diaSemanaAtual);
    Console.WriteLine("Friday: {0}", FimSemana);
    Console.ReadKey();
}
```

@JB

Saída?



isec Engenharia DEIS - Instituto Superior de Engenharia de Coimbra

C#: *Enumerations* - Exemplo

```
using System;
public class Enum
{
    enum DiasSemana { Domingo, Segunda, Terca, Quarta, Quinta, Sexta, Sabado};

    static void Main()
    {
        int x = (int)DiasSemana.Domingo;
        int y = (int)DiasSemana.Sexta;
        Console.WriteLine("Domingo = {0}", x);
        Console.WriteLine("Sexta = {0}", y);
        Console.ReadKey();
    }
}
```

Execute no VS e veja o resultado!

C#: *Struct*

- ***Class*** é tipo *referência* ≠ ***Struct*** é tipo *valor*
- Estruturas podem conter construtores, constantes, métodos, propriedades, eventos,…
- Usa-se a ***keyword*** `struct`
- Pode ser inicializada com ou sem o ***new***
- Estruturas não podem incluir constructores ou destrutores *default*, mas podem ter construtores parametrizados

Programação Web – 2023/2024 – LEI D+CE – PL – ISEC/IPC @JB

isec Engenharia DEIS - Instituto Superior de Engenharia de Coimbra

C#: *Struct*

- Podem implementar interfaces
- Não podem herdar outra estrutura ou classe
  - Membros das estruturas não podem ser especificados como ***abstract***, ***virtual***, ou ***protected***.
  - Devem ser inicializadas com o ***new*** de forma a poder usar-se as suas propriedades, métodos ou eventos
- Um ***`struct`*** pode ser usado como um tipo que permite valor nulo e receber um valor nulo.

Programação Web – 2023/2024 – LEI D+CE+PL – ISEC/IPC @JB

isec Engenharia DEIS - Instituto Superior de Engenharia de Coimbra

C#: *Struct* - Exemplo

```
using System;

class Program
{
    static void Main(string[] args)
    {
        Carro carro;

        carro = new Carro("Azul");
        Console.WriteLine(carro.Descricao());

        carro = new Carro("Vermelho");
        Console.WriteLine(carro.Descricao());

        Console.ReadKey();
    }
}
```

```
struct Carro  {
    private string cor;

    public Carro(string cor)
    {
        this.cor = cor;
    }
    public string Descricao()
    {
        return "Cor do carro = " + cor;
    }
    public string Cor
    {
        get { return cor; }
        set { cor = value; }
    }
}
```

C#: *Arrays*

Numero[0]	Numero[1]	Numero[2]	Numero[3]	…

```
int[] tabela;
int[] numeros;
numeros = new int[10];
numeros = new int[20];
string[,] saladeaula;
saladeaula = new string[10, 6];
int[,,] cubo;
```

Programação Web – 2023/2024 – LEI D+CE+PL – ISEC

isec Engenharia DEIS - Instituto Superior de Engenharia de Coimbra

C#: *Arrays*

```
int[] numeros= new int[5] { 1, 2, 3, 4, 5 };
string[] nomes= new string[3] { "Joana", "Filipa", "José" };

// ou

int[] numeros = new int[] { 1, 2, 3, 4, 5 };
string[] nomes = new string[] {"Joana", "Filipa", "José" };

// ou

int[] numeros = { 1, 2, 3, 4, 5 };
string[] nomes = {"Joana", "Filipa", "José" };
```

Programação Web – 2023/2024 - LEID+CE+PL – ISEC/IPC @JB

isec Engenharia DEIS - Instituto Superior de Engenharia de Coimbra

C#: *Arrays* – Acesso aos Elementos

```
int[] n = new int[10];
int i, j;

for (i = 0; i < 10; i++)
{
    n[i] = i + 100;
}


for (j = 0; j < 10; j++)
{
    Console.WriteLine("Elemento[{0}] = {1}", j, n[j]);
}
```

Programação Web – 2023/2024 – LEI D+CE+PL – ISEC/IPC @JB

isec Engenharia DEIS - Instituto Superior de Engenharia de Coimbra

C#: Arrays – Acesso aos Elementos

- ***foreach***

```
int[] n = new int[10];

for (int i = 0; i < 10; i++)
{
    n[i] = i + 100;
}

foreach (int j in n)
{
    int i = j - 100;
    Console.WriteLine("Elemento[{0}] = {1}", i, j);
}
Console.ReadKey();
```

Programação Web – 2023/2024 – LEI D+CE+PL – ISEC/IPC @JB

isec Engenharia DEIS - Instituto Superior de Engenharia de Coimbra

Programação Web

C#: Namespaces

Departamento de Engenharia Informática e de Sistemas
Instituto Superior de Engenharia de Coimbra/Instituto Politécnico de Coimbra

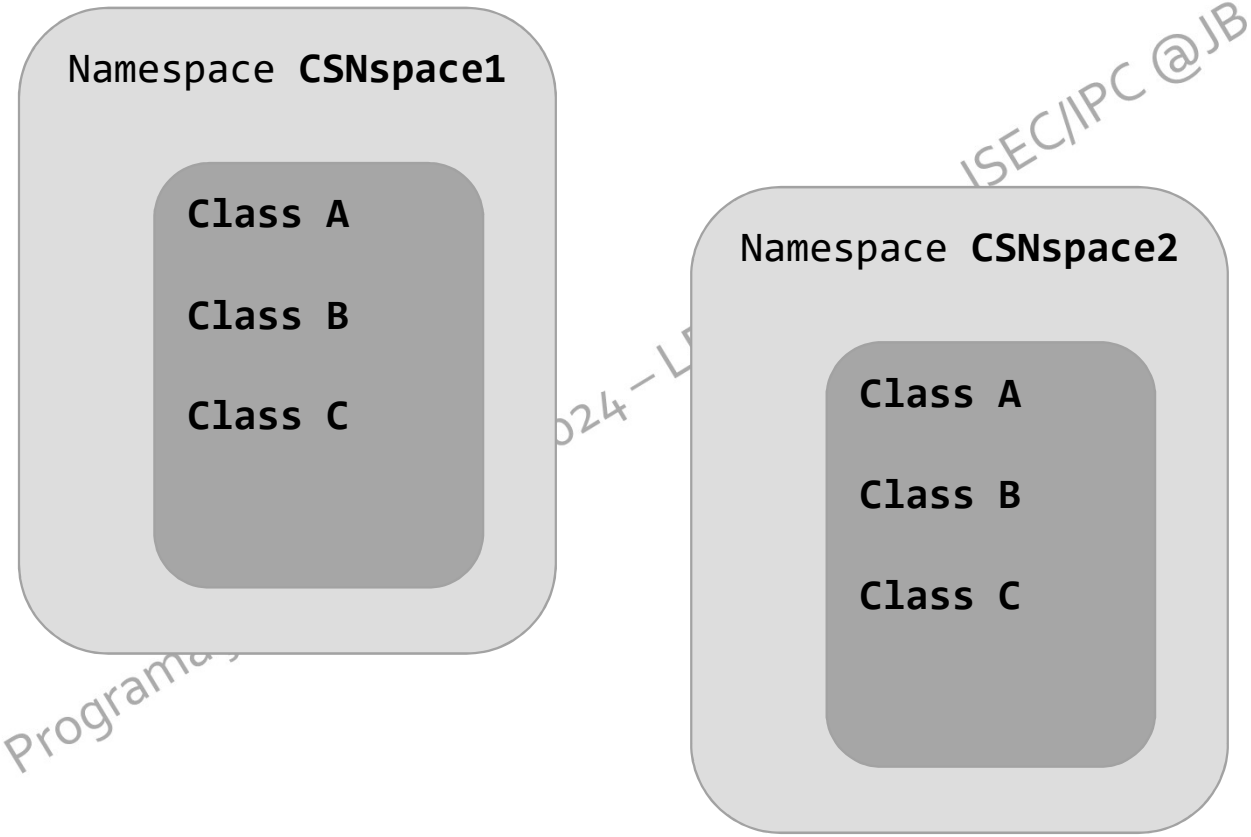
C#: *Namespaces*

- O Código dos programas *C#* estão organizados em ***namespaces***
  - Organiza um conjunto de classes relacionadas entre si
  - As classes podem ter nomes iguais, desde que estejam em *namespaces* diferentes
  - Um *namespace* pode conter outros *namespaces*

Programação Web – 2023/2024 – LEID+CESPL – ISEC/IPC @ JB

isec Engenharia DEIS - Instituto Superior de Engenharia de Coimbra

C#: *Namespaces*




```
CSNspace1.A ...

CSNspace2.A ...
```

ISEC/IPC @JB
Programa 024 – L

isec Engenharia DEIS - Instituto Superior de Engenharia de Coimbra

C#: *Namespaces*

```
using System;
...
Console.Write("Introduza um nome:");
string nome = Console.ReadLine();
```

```
System.Console.Write("Introduza um numero:");
string nome = System.Console.ReadLine();
```

2023/202

C#: *Namespaces*

```
namespace Pweb.Exemplo
{
    class ClasseExemplo
    {

    }
}
```

@JB

Programa

Programação Web

Programação Orientada Objetos com C#

Departamento de Engenharia Informática e de Sistemas
Instituto Superior de Engenharia de Coimbra/Instituto Politécnico de Coimbra

Programação Orientada a Objetos

- Representação de cada elemento em termos de um objeto, ou classe.
- Aproximar o sistema que se cria com o que se observa no mundo real
- Reutilização de código
  - Temporal e número de linhas de código
  - Independência
- Facilidade de leitura e manutenção de código, …

Programação Web – 2023/2024 – LEI D+CE+PL – ISEC/IPC @ 2

POO – Princípios Básicos

Abstração

Encapsulamento

Programação Orientada a Objetos

Herança

Polimorfismo

Programação Web – 202 – LEI D+CE+PL – ISEC/IPC @JB

isec Engenharia DEIS - Instituto Superior de Engenharia de Coimbra

Programação Orientada a Objectos

- Auxilia na solução de problemas aproximando o sistema que se cria com o que se observa do mundo real;
- Dada a complexidade do mundo real, é necessário abstrair conhecimento relevante e encapsular dentro de **objetos**;
- Conjunto de objetos trabalham juntos para realizar uma tarefa;

Programação Web – 2023/2024 – LEI D+CE+PL – ISEC/PC @ ...

Programação Orientada a Objectos

- O mundo é composto por diversos **objetos** que possuem um conjunto de **características** e um **comportamento** bem definido;
- Definir **abstrações** dos objetos reais existentes;
- Todo **objeto** possui as seguintes características:
  - *Estado*: conjunto de propriedades de um objeto;
  - *Comportamento*: conjunto de ações possíveis sobre o objeto;
  - *Unicidade*: todo objeto é único.

Programação Web – 2023/2024 – LEI+CE+PL – ISEC/DEIS @JB

isec Engenharia DEIS - Instituto Superior de Engenharia de Coimbra

Programação Orientada a Objectos

- A comunicação entre objectos é efetuada por mensagens

- **Objecto = Dados + Código**

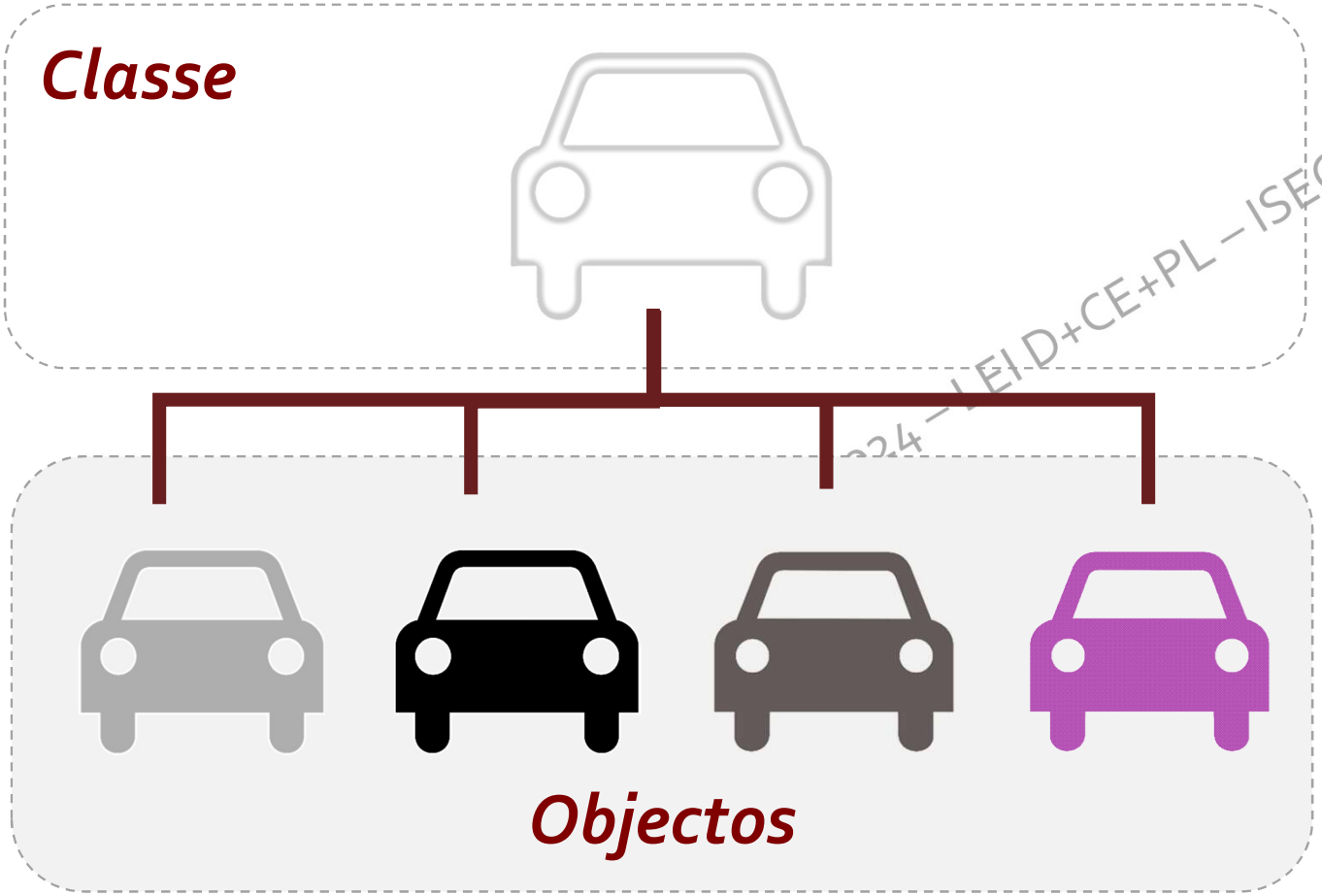
Dados → **atributos**
Código → **métodos**

- **Classes**
  - Definem os tipos de objectos
  - Definem quais os dados e o código dos objectos

Programação Web – 2023/2024 – LEI D+CE+ISEC/IPC @ DEIS

isec Engenharia DEIS - Instituto Superior de Engenharia de Coimbra

Programação Orientada a Objectos



As **classes** são a caracterização abstrata de algo (cliente, empregado, carro,...)

Objeto ≠ Classe

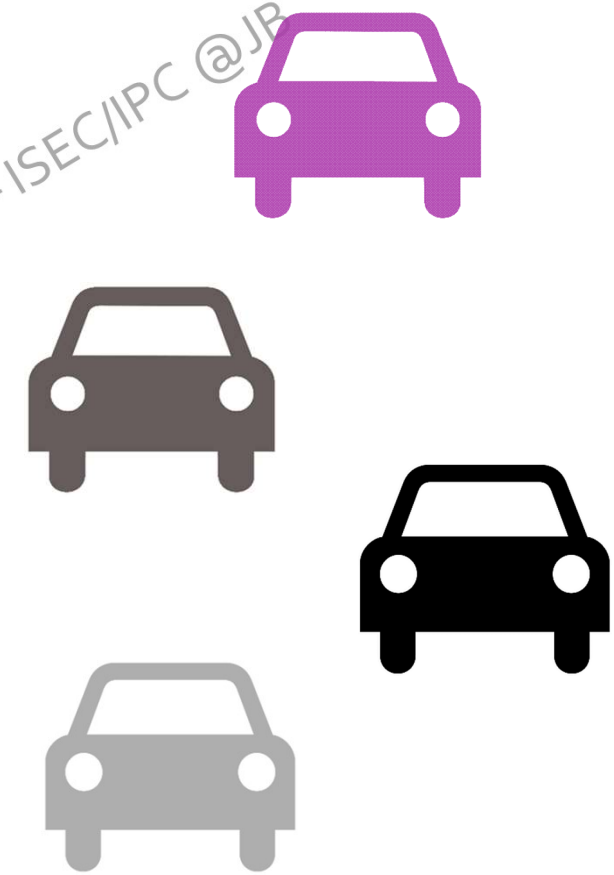
Objeto = Instância de uma Classe

2024 – LEID+CE+PL – ISEC/IPC @JB

isec Engenharia DEIS - Instituto Superior de Engenharia de Coimbra

Programação Orientada a Objectos

```
public class Carro
{
    private string cor;
    private string modelo;
    private string anoFabrico;
    private string combustivel;
    public void Anda()
    {   ...
    }
    public void Para()
    {   ...
    }
    public void Acelera()
    {   ...
    }
}
```



Programação Web – 2023/2024 – LEI D+CE+PL – ISEC/IPC @JR

isec Engenharia DEIS - Instituto Superior de Engenharia de Coimbra

C#: Objetos

- Para criação de um objeto:
  - **Definição da Classe**
    - Considerada como um *template* do objeto
    - Exemplo: *Template* de um *Carro*
  - **Criação do objeto**
    - Objetos de uma classe criados em *runtime*
    - Exemplo: *Carro* em si! Instância do Carro

Programação Web – 2023/2024 – LEID+CE-PL – ISEC/IPC @JB

isec Engenharia DEIS - Instituto Superior de Engenharia de Coimbra

C#: Classes

- Sintaxe geral:

```
acesso class nome_da_classe
    {
        // Bloco de Instruções
    }
```

- acesso
  - nível de proteção da classe (*internal / public*)
  - Por omissão, assume o nível *internal*
  - Pode ser também *abstract* ou *sealed*

Programação Web – 2023/202 @JB

isec Engenharia DEIS - Instituto Superior de Engenharia de Coimbra

C#: Classes

```
using System;
namespace Exemplo {
    class Program {
        static void Main(string[] args)
        {
            @override x = new @override();
            Console.WriteLine(x.Numero);
            Console.ReadKey();
        }
    }
    class @override
    {
        public int Numero { get; set; }

        public @override()
        {
            Numero = 5;
        }
    }}
```

Programaç

C#: Classes

- *Construtor*
  - Se não for definido um construtor para uma classe, o C# define um por omissão

```
class MinhaClasse {
    public MinhaClasse()
    {
        // Construtor Default
    }

    public MinhaClasse(int meuInt)
    {
        // Construtor com parâmetro
    }
}
```

Programação Web – 2023/2024 – ISEC/IPC @JB

isec Engenharia DEIS - Instituto Superior de Engenharia de Coimbra

C#: Classes

- *Construtor: keyword **this***

@JB

```
class Motociclo {
    public string nomeCondutor;
    public int velocidade;
    public Motociclo() {}
    public Motociclo(int velocidade): this(velocidade, "") { }
    public Motociclo(string nome) : this(0,nome) { }
    public Motociclo(int velocidade, string nome) {
        if(velocidade>20) { velocidade = 20; }
        this.velocidade = velocidade;
        nomeCondutor = nome;
    }
}
```

isec Engenharia DEIS - Instituto Superior de Engenharia de Coimbra

C#: Classes

- *Destrutor*

```
class MinhaClasse
{
        ~MinhaClasse()
        {
                //Destrutor
        }
}
```

- Executado quando o *Garbage Collector* ocorre, permitindo libertar recursos

@JB
Programação W

C#: Classe

```
using System;
1 reference
public class MyClass
{
    public string  myField = string.Empty;
    0 references
    public MyClass()
    {
    }
    0 references
    public void MyMethod(int parameter1, string parameter2)
    {
        Console.WriteLine("First Parameter {0}, second parameter {1}", parameter1, parameter2);
    }

    0 references
    public int MyAutoImplementedProperty { get; set; }

    private int myPropertyVar;

    0 references
    public int MyProperty
    {
        get { return myPropertyVar; }
        set { myPropertyVar = value; }

    }
}
```

Nome da Classe

Campos

Construtor

Método (Função)

Propriedade *Auto-Implemented*

Propriedade

Programa

isec Engenharia DEIS - Instituto Superior de Engenharia de Coimbra

C#: Membros de uma Classe

- Definindo uma classe, fornece-se definições para todos os membros:
  - Variáveis onde são armazenados os dados, designados como ***campos*** ou ***atributos*** *(fields)*;
  - **Métodos** que permitem manipular os dados;
  - **Propriedades**
    - Recorrendo aos *getters* e *setters*

Programação Web – 2023/2024 – LEI+D+CE+DL – ISEC/IPC @ ie

isec Engenharia DEIS - Instituto Superior de Engenharia de Coimbra

C#: Atributos / Campos

- *Atributos* são também designados como *campos* (*member fields*)
- Permitem definir as características/dados que especificam os objetos
- Em C# os atributos de uma classe são definidos no interior da mesma
- Sintaxe: `acesso tipo field;`
  - `acesso`: Nível de proteção do atributo
    private (por omissão), *public, protected, internal, protected internal*

Programação Web – 2023/2024 – LEI D+CEPL – ISEC/IPC @ 13

isec Engenharia DEIS - Instituto Superior de Engenharia de Coimbra

C#: Campos

Nível de proteção	Descrição
private (por omissão)	Só pode ser acedido a partir da classe onde foi declarado
public	Pode ser acedido externamente por outras classes ou métodos (incluindo fora do assembly onde foi definido)
protected	Só pode ser acedido a partir da classe onde foi declarado e a partir de classes derivadas
internal	Pode ser acedido a partir de qualquer classe (exceto fora do assembly onde foi definido
protected internal	Só pode ser acedido a partir da classe onde foi declarado ou partir de classes derivadas (exceto fora do assembly onde foi definido)

C#: Exemplo (1)

Exemplo - Program.cs*

Program.cs* | Exemplo | Program | Main(string[] args)

```
using System;
7 references
public class Circulo
{
    private int raio;
    1 reference
    public Circulo()
    {
        raio = 0;
    }
    1 reference
    public Circulo(int raioInicial)
    {
        raio = raioInicial;
    }
    3 references
    public double Area()
    {
        return Math.PI * raio * raio;
    }
}
0 references
public class Program
{
    0 references
    static void Main(string[] args)
    {
        Circulo x = new Circulo();
        Circulo y;
        y = new Circulo(5);
        Circulo z = y;

        Console.WriteLine("Circulo x = {0} ", x.Area());
        Console.WriteLine("Circulo y = {0} ", y.Area());
        Console.WriteLine("Circulo z = {0} ", z.Area());
    }
}
```

91 %

Criação da classe ***Circúlo*** com o campo *raio*

@JB
+PL
Pro

C#: Exemplo (1)

Exemplo - Program.cs*

Program.cs* | Exemplo | Program | Main(string[] args)

```
using System;
7 references
public class Circulo
{
    private int raio;
    1 reference
    public Circulo()
    {
        raio = 0;
    }
    1 reference
    public Circulo(int raioInicial)
    {
        raio = raioInicial;
    }
    3 references
    public double Area()
    {
        return Math.PI * raio * raio;
    }
}
0 references
public class Program
{
    0 references
    static void Main(string[] args)
    {
        Circulo x = new Circulo();
        Circulo y;
        y = new Circulo(5);
        Circulo z = y;

        Console.WriteLine("Circulo x = {0} ", x.Area());
        Console.WriteLine("Circulo y = {0} ", y.Area());
        Console.WriteLine("Circulo z = {0} ", z.Area());
    }
}
```

91 %

Construtor d*efault* da classe **Circulo**

Construtor da classe **Circulo** passando por parâmetro o raio

Pro
SEC/IPC @JB

C#: Exemplo (1)

Exemplo - Program.cs*

Program.cs* | Exemplo | Program | Main(string[] args)

```
using System;
7 references
public class Circulo
{
    private int raio;
    1 reference
    public Circulo()
    {
        raio = 0;
    }
    1 reference
    public Circulo(int raioInicial)
    {
        raio = raioInicial;
    }
    3 references
    public double Area()
    {
        return Math.PI * raio * raio;
    }
}
0 references
public class Program
{
    0 references
    static void Main(string[] args)
    {
        Circulo x = new Circulo();
        Circulo y;
        y = new Circulo(5);
        Circulo z = y;

        Console.WriteLine("Circulo x = {0} ", x.Area());
        Console.WriteLine("Circulo y = {0} ", y.Area());
        Console.WriteLine("Circulo z = {0} ", z.Area());
    }
}
```

91 %

Criação de uma **instância** da classe **Circulo** (uso da *keyword new*)

Criação de uma **instância** da classe **Circulo**

PL – ISEC/IPC @JB

isec Engenharia DEIS - Instituto Superior de Engenharia de Coimbra

C#: Exemplo (1)

```
using System;
7 references
public class Circulo
{
    private int raio;
    1 reference
    public Circulo()
    {
        raio = 0;
    }
    1 reference
    public Circulo(int raioInicial)
    {
        raio = raioInicial;
    }
    3 references
    public double Area()
    {
        return Math.PI * raio * raio;
    }
}
0 references
public class Program
{
    0 references
    static void Main(string[] args)
    {
        Circulo x = new Circulo();
        Circulo y;
        y = new Circulo(5);
        Circulo z = y;

        Console.WriteLine("Circulo x = {0} ", x.Area());
        Console.WriteLine("Circulo y = {0} ", y.Area());
        Console.WriteLine("Circulo z = {0} ", z.Area());
    }
}
```

Programaçã
ISEC/IPC @JB

Saída?

```
Circulo x = 0
Circulo y = 78,5398163397448
Circulo z = 78,5398163397448
```

isec Engenharia DEIS - Instituto Superior de Engenharia de Coimbra

C#: *Partial Class*

- Pode-se “partir” a classe em vários ficheiros
  - O nome dos ficheiros pode ser qualquer um, importante é definir o tipo (neste caso classe) com o mesmo nome
- Recurso à keyword `partial`
- *Visual Studio* recorre constantemente a classes parciais
- Por exemplo:
  - Métodos e Propriedades num ficheiro e restantes elementos noutro…

Programação Web – 2023/2024 – LEI D+CE+RI – ISEC/IPC @ 2

isec Engenharia DEIS - Instituto Superior de Engenharia de Coimbra

C#: *Partial Class*

```
namespace PartialClasse
{
    partial class MinhaClasse
    {
        // Campos e Construtores
        private int idade;
        public MinhaClasse()
        {
            idade = 0;
        }
        public MinhaClasse(int i)
        {
            idade = i;
        }
    }
}
```

MinhaClasse.cs

```
namespace PartialClasse
{
    partial class MinhaClasse
    {
        // Metodos e Propriedades
        public int Idade
        {
            get { return idade; }
            set  {
                if (value<0||value>100)
                    idade = 0;
                else idade = value;
            }
        }
    }
}
```

MinhaClasse.Core.cs

isec Engenharia DEIS - Instituto Superior de Engenharia de Coimbra

C#: Níveis de Proteção das Classes

Nível de Proteção	Descrição
internal (por omissão)	Classe acessível apenas no projeto corrente
public	Classe acessível de qualquer parte
abstract **internal abstract**	Acessível apenas no projeto corrente e não pode ser instanciada, apenas derivada de
public abstract	Pode ser acessível de qualquer parte e não pode ser instanciada, apenas derivada de
sealed **internal sealed**	Acessível apenas no projeto corrente e não pode ser derivada de, apenas instanciada
public sealed	Acessível de qualquer parte e não pode ser derivada de, apenas instanciada

C#: Encapsulamento

- **Encapsulamento** tem como objetivo a ocultação de pormenores internos da classe
- Fornecida uma *interface pública* constituída por métodos que são a única coisa que o resto do programa se apercebe da classe
- A implementação da classe pode ser modificada sem que isso interfira com o resto do programa, desde que não se altere a interface

Programação Web – 2023/2024 – LEI D+CE+DL – ISEC/IPC @ JB

isec Engenharia DEIS - Instituto Superior de Engenharia de Coimbra

POO: Encapsulamento

- Adicionam segurança na implementação em OO uma vez que escondem elementos internos, criando uma espécie de *black box*
- Maior parte das linguagens OO implementam o encapsulamento com recurso a métodos especiais chamados de *getters* e *setters*
- Em C# as **propriedades** evitam o acesso direto aos campos do objeto

Programação Web – 2023/2024 – LEID+CE+PL – ISEC/IPC @ DEIS

isec Engenharia DEIS - Instituto Superior de Engenharia de Coimbra

C#: Propriedades

- Definidas de forma semelhante aos campos, permitindo desempenhar um processamento adicional
  - Recurso às *keywords* *get* e *set*
- Estrutura básica:

```
public int MinhaPropriedade
    {
        get {
            // código para o get
        }
        set {
            // código para o set
        }
    }
```

Programação Web – 2023/20
D+CE+PL – ISEC-IPC @ 20

isec Engenharia DEIS - Instituto Superior de Engenharia de Coimbra

C#: Propriedades

- Propriedade
  - Encapsula um campo privado
  - Permite especificar o *get* para retribuir valores de um campo e permite especificar o *set* para definir um valor
  - *Exemplo:*

```
private int meuCampo;
public int MeuCampo
    {
        get { return meuCampo;
            }
        set { meuCampo=value;
            }
    }
```

Programação W – DWD+CE PL – ISEC/IPC @JB

isec Engenharia DEIS - Instituto Superior de Engenharia de Coimbra

C#: Propriedades

- É possível também adicionar lógica nas propriedades:

```
class Exemplo
{
    private int propriedade;
    public int Propriedade {
        get
        {
            return propriedade / 2;
        }
        set
        {
            if (value > 100) propriedade = 100;
            else propriedade = value;
        }
    }
}
```

Saída?

```
Exemplo x = new Exemplo();
c.Propriedade = 1;
```

PC @ IB

C#: Propriedades

```
using System;
class IntervaloTempo
{
   private double segundos;

   public double Horas
   {
       get { return segundos / 3600; }
       set {
          if (value < 0 || value > 24)
             throw new ArgumentOutOfRangeException(
                   $"{nameof(value)} deve estar compreendido entre 0 e 24.");
          segundos = value * 3600;
       }
   }
}
```

```
IntervaloTempo t = new IntervaloTempo();
t.Horas = 24;
 Console.WriteLine($"Tempo em horas: {t.Horas}");
```

C#: *Auto Implemented Properties*

- *Auto implemented Properties* em C#
  - Surgiram com o C# 3.0
  - Permitem uma declaração simplificada, quando não é necessária especificar qualquer lógica no *get* ou *set*
  - Não foram declaradas variáveis internas?
    - Criadas pelo compilador! Poupa tempo e torna o código mais limpo.

```
public String MeuNome
    {
        get; set;
    }
```

Programação Web – 2023/2024 – TELD+CE-PL – ISEC/IPC @JB

isec Engenharia DEIS - Instituto Superior de Engenharia de Coimbra

C#: *Auto Implemented Properties*

```
using System;
public class Aluno
{
   public string Nome  { get; set; }
   public decimal Idade { get; set; }
}
class Program
{
   static void Main(string[] args)
   {
      var aluno = new Aluno{ Nome = "xpto", Idade = 17 };
      Console.WriteLine($"{aluno.Nome} tem {aluno.Idade} anos!");
   }
}
```

C#: Propriedades

- Apenas de leitura

```
public int Numero {get;}
```

- Apenas de escrita

```
public int Numero {set;}
```

Programação Web – 2023/2024 – LEI D+CE+PL – ISEC/IPC @JB

isec Engenharia DEIS - Instituto Superior de Engenharia de Coimbra

C#: *Properties*

```
public class Aluno
{
    string nome;
    decimal idade;
    public Aluno(string nome, decimal idade)
    {
        this.nome = nome;
        this.idade = idade;
    }
    public string Nome
    {
        get => nome;
        set => nome = value;
        }
    public decimal Idade
    {
        get => idade;
        set => idade = value;
        }
    }
}
```

```
var aluno = new Aluno{ Nome = "xpto", Idade = 17 };
Console.WriteLine($"{aluno.Nome} tem {aluno.Idade} anos!");
```

C# 7.0
Expression-bodied members (EBM's)

O C# 6 permitiu a implementação de propriedades somente leitura e métodos através das expressões lambdas e isso foi chamado de EBM.

Na versão 7.0, este recurso foi melhorado permitindo a utilização de expressões lambdas em construtores, propriedades com get/set e finalizadores.

O objetivo do recurso é tornar o código conciso e legível fazendo com que o membro do tipo (construtor, métodos, propriedades, destrutor, etc) seja definido em uma única expressão.

ISEC/IPC @ IP

isec Engenharia DEIS - Instituto Superior de Engenharia de Coimbra

C#: Métodos

- Ações que os objetos podem desempenhar, definidos dentro do corpo de uma classe
- Tecnicamente conhecidas por funções de membro de uma classe
  - Sintaxe: `acesso tipo nome_método(argumentos)`
  - Exemplo:

```
public void fazQualquerCoisa() { }

public String ObtemNome() { }
```

- Existem métodos próprios: construtores, …

Programação Web – 2023/2024 – LEI D+CD+PL – ISEC/IPC @ DEIS

isec Engenharia DEIS - Instituto Superior de Engenharia de Coimbra

C#: Exemplo

Exemplo - Program.cs*

Program.cs* | Exemplo | Program | Main(string[] args)

```
using System;
7 references
public class Circulo
{
    private int raio;
    1 reference
    public Circulo()
    {
        raio = 0;
    }
    1 reference
    public Circulo(int raioInicial)
    {
        raio = raioInicial;
    }
    3 references
    public double Area()
    {
        return Math.PI * raio * raio;
    }
}
0 references
public class Program
{
    0 references
    static void Main(string[] args)
    {
        Circulo x = new Circulo();
        Circulo y;
        y = new Circulo(5);
        Circulo z = y;

        Console.WriteLine("Circulo x = {0} ", x.Area());
        Console.WriteLine("Circulo y = {0} ", y.Area());
        Console.WriteLine("Circulo z = {0} ", z.Area());
    }
}
```

91 %

Criação do método **Area()**

PL – ISEC/IPC @JB

isec Engenharia DEIS - Instituto Superior de Engenharia de Coimbra

C#: Exemplo

Exemplo - Program.cs*

Program.cs* | Exemplo | Program | Main(string[] args)

```
using System;
7 references
public class Circulo
{
    private int raio;
    1 reference
    public Circulo()
    {
        raio = 0;
    }
    1 reference
    public Circulo(int raioInicial)
    {
        raio = raioInicial;
    }
    3 references
    public double Area()
    {
        return Math.PI * raio * raio;
    }
}
0 references
public class Program
{
    0 references
    static void Main(string[] args)
    {
        Circulo x = new Circulo();
        Circulo y;
        y = new Circulo(5);
        Circulo z = y;

        Console.WriteLine("Circulo x = {0} ", x.Area());
        Console.WriteLine("Circulo y = {0} ", y.Area());
        Console.WriteLine("Circulo z = {0} ", z.Area());
    }
}
```

91 %

Chamada ao método **Area**

PL – ISEC/IPC @JB

DEIS - Instituto Superior de Engenharia de Coimbra

C#: Exemplo

Exemplo - Program.cs*

Program.cs* | Exemplo | Program | Main(string[] args)

```
using System;
7 references
public class Circulo
{
    private int raio;
    1 reference
    public Circulo()...
    1 reference
    public Circulo(int raioInicial)...
    3 references
    public double Area()...
    public void MostraMensagemRaio()
    {
        Console.WriteLine("Raio =  {0}",raio);
    }
}
0 references
public class Program
{
        0 references
        static void Main(string[] args)
        {
            Circulo x = new Circulo();
            Circulo y;
            y = new Circulo(5);
            Circulo z = y;

            Console.WriteLine("Circulo x = {0} ", x.Area());
            Console.WriteLine("Circulo y = {0} ", y.Area());
            Console.WriteLine("Circulo z = {0} ", z.Area());

            x.MostraMensagemRaio();
            y.MostraMensagemRaio();
        }
}
```

91 %

Chamada do método **MostraMensagemRaio**

PL – ISEC/IPC @JB

DEIS - Instituto Superior de Engenharia de Coimbra

C#: Exemplo

Exemplo - Program.cs*

Program.cs* | Exemplo | Program | Main(string[] args)

```
using System;
7 references
public class Circulo
{
    private int raio;
    1 reference
    public Circulo()...
    1 reference
    public Circulo(int raioInicial)...
    3 references
    public double Area()...
    2 references
    public void MostraMensagemRaio()
    {
        Console.WriteLine("Raio =  {0}",raio);
    }
}
0 references
public class Program
{
        0 references
        static void Main(string[] args)
        {
            Circulo x = new Circulo();
            Circulo y;
            y = new Circulo(5);
            Circulo z = y;

            Console.WriteLine("Circulo x = {0} ", x.Area());
            Console.WriteLine("Circulo y = {0} ", y.Area());
            Console.WriteLine("Circulo z = {0} ", z.Area());

            x.MostraMensagemRaio();
            y.MostraMensagemRaio();
        }
}
```

91 %

PL – ISEC/IPC @JB

Saída?

file:///D:/IBDP/Exemplo/Exempl...

```
Circulo z = 78,5398163397448
Raio =  0
Raio =  5
```

isec Engenharia DEIS - Instituto Superior de Engenharia de Coimbra

C#: Métodos - Parâmetros

- Por valor
- Por referência
  - *ref*

Exemplo - Program.cs

Program.cs

ExemploIBDP | PassagemParametros | Quadrado(int x)

```
using System;
0 references
class PassagemParametros
{
    1 reference
    static void Quadrado(int x)
    {
        x *= x;
        System.Console.WriteLine("Valor dentro do metodo: {0}", x);
    }
    0 references
    static void Main(string[] args)
    {
        int n = 5;
        Console.WriteLine("Valor antes de chamar o método: {0}", n);

        Quadrado(n);
        Console.WriteLine("Valor depois de chamar o método: {0}", n);

        Console.ReadKey();
    }
}
```

91 %

Método **com parâmetros**

Programação Web – 20

isec Engenharia DEIS - Instituto Superior de Engenharia de Coimbra

C#: Passagem de Parâmetros por valor

- Por valor
- Por referência
  - *ref*

```
using System;
0 references
class PassagemParametros
{
    1 reference
    static void Quadrado(int x)
    {
        x *= x;
        System.Console.WriteLine("Valor dentro do metodo: {0}", x);
    }
    0 references
    static void Main(string[] args)
    {
        int n = 5;
        Console.WriteLine("Valor antes de chamar o método: {0}", n);

        Quadrado(n);
        Console.WriteLine("Valor depois de chamar o método: {0}", n);

        Console.ReadKey();
    }
}
```

Programação Web -

isec Engenharia DEIS - Instituto Superior de Engenharia de Coimbra

C#: Passagem de Parâmetros por valor

- Por valor
- Por referência
  - *ref*

```
using System;
0 references
class PassagemParametros
{
    1 reference
    static void Quadrado(int x)
    {
        x *= x;
        System.Console.WriteLine("Valor dentro do metodo: {0}", x);
    }
    0 references
    static void Main(string[] args)
    {
        int n = 5;
        Console.WriteLine("Valor antes de chamar o método: {0}", n);

        Quadrado(n);
        Console.WriteLine("Valor depois de chamar o método: {0}", n);

        Console.ReadKey();
    }
}
```

```
Valor antes de chamar o método: 5
Valor dentro do metodo: 25
Valor depois de chamar o método: 5
```

Programação Web -

isec Engenharia DEIS - Instituto Superior de Engenharia de Coimbra

C#: Passagem de Parâmetros *ref*

- Valores
- Referência
  - *ref*
  - Devem ser inicializados antes de serem passados ao método

```
using System;
0 references
class PassagemParametros
{
    1 reference
    static void Quadrado(ref int x)
    {
        x *= x;
        System.Console.WriteLine("Valor dentro do metodo: {0}", x);
    }
    0 references
    static void Main(string[] args)
    {
        int n = 5;
        Console.WriteLine("Valor antes de chamar o método: {0}", n);

        Quadrado(ref n);
        Console.WriteLine("Valor depois de chamar o método: {0}", n);

        Console.ReadKey();
    }
}
```

Programação Web -

isec Engenharia DEIS - Instituto Superior de Engenharia de Coimbra

C#: Passagem de Parâmetros por referência

- Valores
- Referência
  - *ref*

Saída?

```
Valor antes de chamar o método: 5
Valor dentro do metodo: 25
Valor depois de chamar o método: 25
```

```
using System;
0 references
class PassagemParametros
{
    1 reference
    static void Quadrado(ref int x)
    {
        x *= x;
        System.Console.WriteLine("Valor dentro do metodo: {0}", x);
    }
    0 references
    static void Main(string[] args)
    {
        int n = 5;
        Console.WriteLine("Valor antes de chamar o método: {0}", n);

        Quadrado(ref n);
        Console.WriteLine("Valor depois de chamar o método: {0}", n);

        Console.ReadKey();
    }
}
```

isec Engenharia DEIS - Instituto Superior de Engenharia de Coimbra

C#: Métodos – Parâmetros *out*

```
using System;
0 references
```

- out
- Não necessita ser inicializado antes de ser passado ao método

```
0 references
static void Main(string[] args)
{
    int[] meuArray = { 1, 8, 3, 6, 2, 5, 9, 3, 0, 2 };
    int maxIndice;
    Console.WriteLine($"O valor máximo no meuArray é { MaxValor(meuArray, out maxIndice)}");
    Console.WriteLine($"A primeira ocorrencia deste valor está no elemento { maxIndice + 1} );

    Console.ReadKey();
}
}
```

Programação Web

isec Engenharia DEIS - Instituto Superior de Engenharia de Coimbra

C#: Métodos – Parâmetros *out*

- out

```
using System;
0 references
class Parametros
{

    1 reference
    static int MaxValor(int[] intArray, out int maxIndice)
    {
        int maxVal = intArray[0];
        maxIndice = 0;
        for (int i = 1; i < intArray.Length; i++)
        {
            if (intArray[i] > maxVal)
            {
                maxVal = intArray[i];
                maxIndice = i;
            }
        }
        return maxVal;
    }
    0 references
    static void Main(string[] args)
    {
        int[] meuArray = { 1, 8, 3, 6, 2, 5, 9, 3, 0, 2 };
        int maxIndice;
        Console.WriteLine($"O valor máximo no meuArray é { MaxValor(meuArray, out maxIndice)}");
        Console.WriteLine($"A primeira ocorrencia deste valor está no elemento { maxIndice + 1} ");
```

```
O valor máximo no meuArray é 9
A primeira ocorrencia deste valor está no elemento 7
```

```
}
```

Programação Web

isec Engenharia DEIS - Instituto Superior de Engenharia de Coimbra

C#: Parâmetros *ref* vs *out*

```
using System;
namespace RefVSOut {
    class Program
    {
        static void Main(string[] args)
        {
            int val1 = 0;
            int val2;

            MethodRef(ref val1);
            Console.WriteLine(val1);

            MethodOut( out val2);
            Console.WriteLine(val2);
        }
    ...
```

```
static void MethodRef(ref int value)
 {
            value = 1;
 }
static void MethodOut(out int value)
  {
     value = 2;
  }
```

@JB

C#: *Params Arrays*

- C# suporta ***parameter arrays***
- Permite a passagem de um número variável de parâmetros de <u>tipo idêntico</u> (ou classes relacionadas por herança) como um único parâmetro lógico
- Argumentos especificados com a <u>*keyword params*</u> podem ser processados, se chamado através do envio de um *array* fortemente "tipado" ou uma lista de itens separados por vírgula

Programação Web – 2023/2024 – LEI D+CE+PL – ISEC/IPC @JB

C#: *Params Arrays*

```
using System;
namespace ArrayApplication {
    class ParamArray {
        public int AddElements(params int[] arr) {
            int sum = 0;
            foreach (int i in arr)
                sum += i;
            return sum;
        }
    }
    class TestClass {
        static void Main(string[] args) {
            ParamArray app = new ParamArray();
            int sum = app.AddElements(512, 720, 250, 567, 889);
            Console.WriteLine("Soma: {0}", sum);
            Console.ReadKey();
        }
    }
}
```

Programação Web - 2023/2024 – LEI D+CE+PL – ISEC/IPC @JB

isec Engenharia DEIS - Instituto Superior de Engenharia de Coimbra

C#: *Params Arrays*

```
using System;
namespace ArrayApplication {
    class ParamArray {
        public int AddElements(params int[] arr) {
            int sum = 0;
            foreach (int i in arr)
                sum += i;
            return sum;
        }
    }
    class TestClass {
        static void Main(string[] args) {
            ParamArray app = new ParamArray();
            int sum = app.AddElements();
            Console.WriteLine("Soma: {0}", sum);
            Console.ReadKey();
        }
    }
}
```

Programação Web – 2023/2024 – LEI D+CE+PL – ISEC/IPC @JB

isec Engenharia DEIS - Instituto Superior de Engenharia de Coimbra

C#: Params Arrays

```
using System;
namespace ArrayApplication {
    class ParamArray {
        public int AddElements(params int[] arr) {
            int sum = 0;
            foreach (int i in arr)
                sum += i;
            return sum;
        }
    }
    class TestClass {
        static void Main(string[] args) {
            ParamArray app = new ParamArray();
            int sum = app.AddElements
            Console.WriteLine(
            Console.ReadKey();
        }
    }
}
```

```
int[] dados={1, 4, 5, 6 };
AddElements(dados);
```

Programação Web – 2023/2024 – LEI D+CE+PL – ISEC/IPC @JB

isec Engenharia DEIS - Instituto Superior de Engenharia de Coimbra

C#: Parâmetros Opcionais

- O C# permite criar métodos que passam **argumentos <u>opcionais</u>**
- Permite invocação do método omitindo alguns argumentos "desnecessários"

Programação Web – 2023/2024 – LEI+CE+PL – ISEC/IPC @m

isec Engenharia DEIS - Instituto Superior de Engenharia de Coimbra

C#: Parâmetros Opcionais - Exemplo

```
class Program {
    static void Main() {
        // Omite os parâmetros opcionais
        Metodo();
        // Omite segundo parametro
        Metodo(4);
        // Sintaxe clássica
        Metodo(4, "Jose");
        // ERRO:  Metodo("Filomena");  Necessário especificar o nome do parâmetro
        Metodo(nome: "Filipa");
        // Especifica ambos os parâmetros
        Metodo(nome: "Maria“, numero: 5);
    }
    static void Metodo(int numero = 0, string nome = "Indefinido") {
        System.Console.WriteLine("Numero = {0}, Nome = {1}", numero, nome);
    }
}
```

C#: Parâmetros Opcionais

CE+PL – ISEC/IPC @JB

```
static void Metodo(int dia = System.DateTime.Now.Day ,
                   string nome = "Indefinido")
{

        System.Console.WriteLine("Dia = {0}, Nome = {1}",
                                 dia, nome);
}
```


isec Engenharia DEIS - Instituto Superior de Engenharia de Coimbra

Programação Web

C#: Polimorfismo, Herança, ...

Departamento de Engenharia Informática e de Sistemas
Instituto Superior de Engenharia de Coimbra/Instituto Politécnico de Coimbra

C#: Polimorfismo

- O **Polimorfismo** permite ao programador definir **métodos genéricos** em classes ou interfaces, que podem ser implementados de **diversas formas** nas respetivas subclasses
- Polimorfismo aplica-se apenas aos métodos da classe e, por definição, exige a utilização de herança
- Vantagens:
  - Reutilização de código;
  - Tempo de desenvolvimento;
  - Manutenção.

Programação Web – 2023/2024 – LEI D+CE+PL – ISEC/IPC @JB

isec Engenharia DEIS - Instituto Superior de Engenharia de Coimbra

C#: Herança

- A **Herança** é um dos principais conceitos da POO
- Permite reutilização de código e redução da complexidade
- Permite **construção de classes baseadas noutras classes** já existentes (neste caso, a primeira classe diz-se *classe base*)
- A subclasse:
  - Herda todas as propriedades e métodos da classe existente
  - Pode incluir ou sobrepor novas propriedades e novos métodos

Programação Web – 2023/2024 – LEI D+OF+PL – ISEC/IPC @ DEIS

isec Engenharia DEIS - Instituto Superior de Engenharia de Coimbra

C#: Herança

- Em *C#* apenas é permitido **uma classe base!**
- Mecanismo recursivo, permitindo criar uma hierarquia de classes

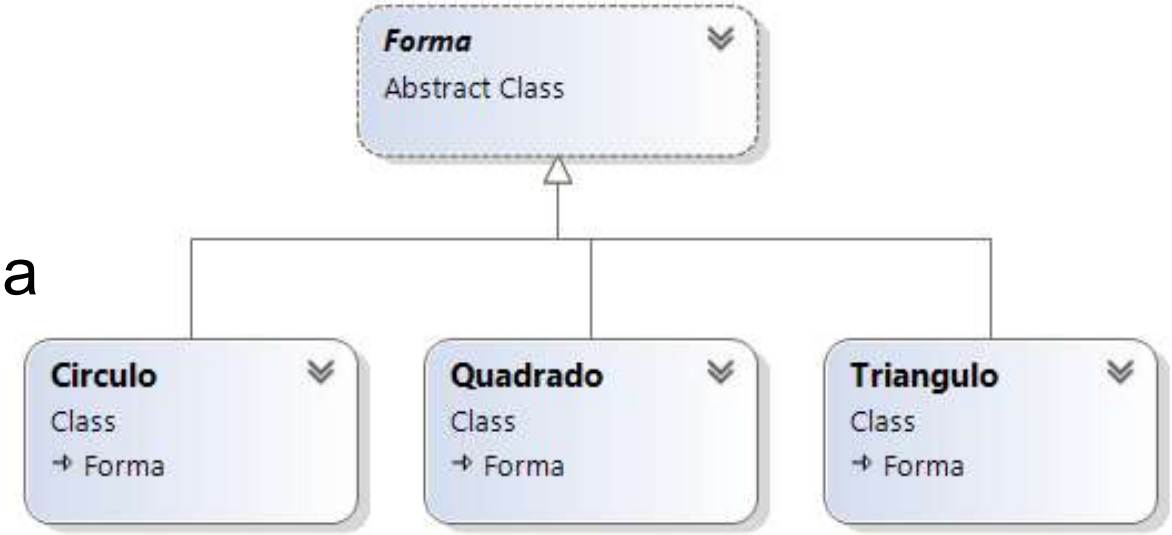
- Exemplo:

Programação Web – 2023/2024 – LEI D+CE+PL – ISEC/IPC @JP

isec Engenharia DEIS - Instituto Superior de Engenharia de Coimbra

C#: Herança (3)

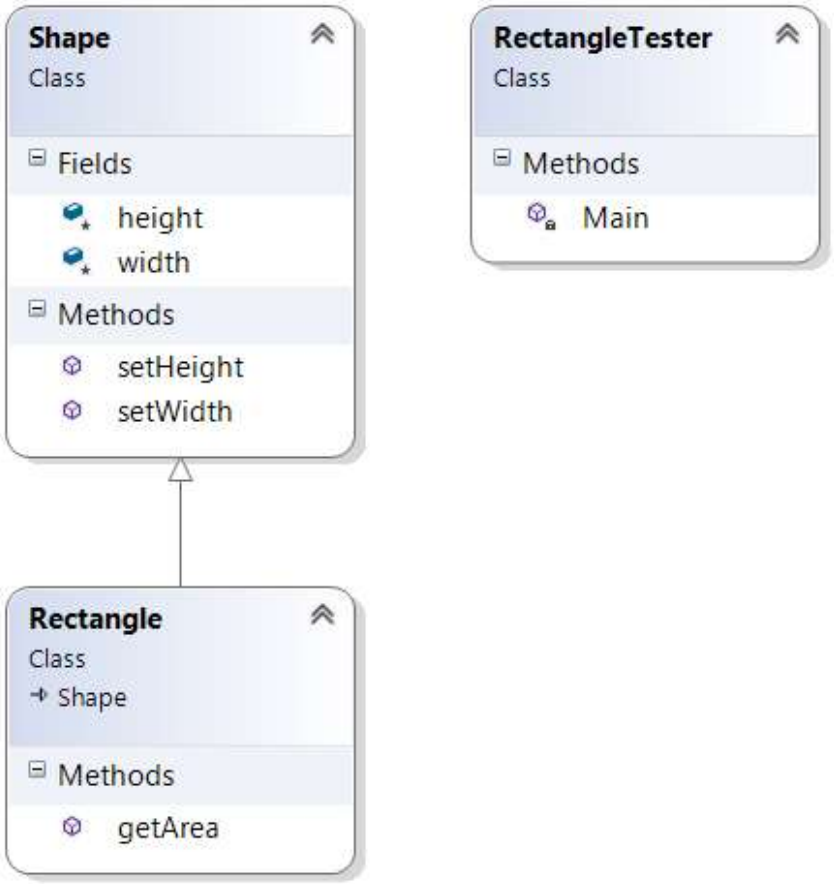
- Exemplo 2:
  - Um “quadrado” é uma “forma”
  - Uma “forma” define uma propriedade comum “cor”
  - Um “quadrado” herda a propriedade “cor”



Programação Web – 2023/2024 – JB

isec Engenharia DEIS - Instituto Superior de Engenharia de Coimbra

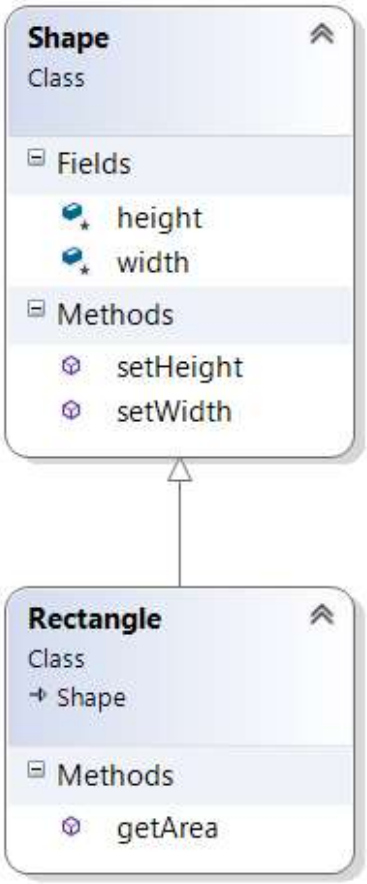
C#: Herança – Exemplo (1)

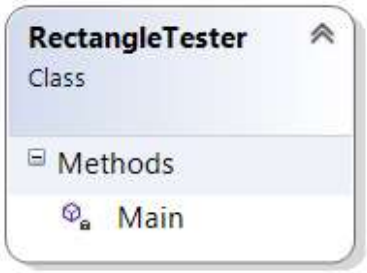



```
class Shape
    {
        public void setWidth(int w)
        {
            width = w;
        }
        public void setHeight(int h)
        {
            height = h;
        }
        protected int width;
        protected int height;
    }
```


isec Engenharia DEIS - Instituto Superior de Engenharia de Coimbra

C#: Herança – Exemplo (1)






```
class Rectangle : Shape
    {
        public int getArea()
        {
            return (width * height);
        }
    }
```

```
class RectangleTester
{
    static void Main(string[] args)
    {
        Rectangle Rect = new Rectangle();
        Rect.setWidth(5); Rect.setHeight(7);
        Console.WriteLine("Total area: {0}", Rect.getArea());}
}
```

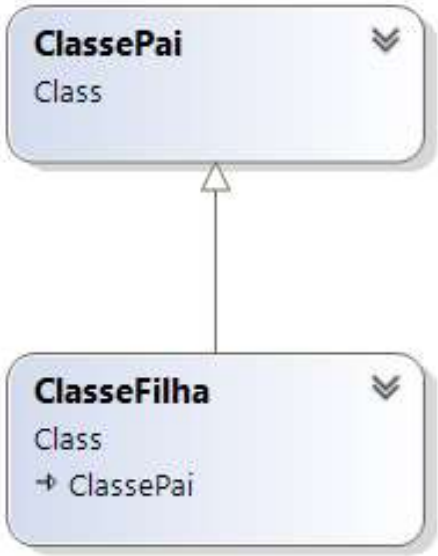
isec Engenharia DEIS - Instituto Superior de Engenharia de Coimbra

C#: Herança

- Herdar uma classe

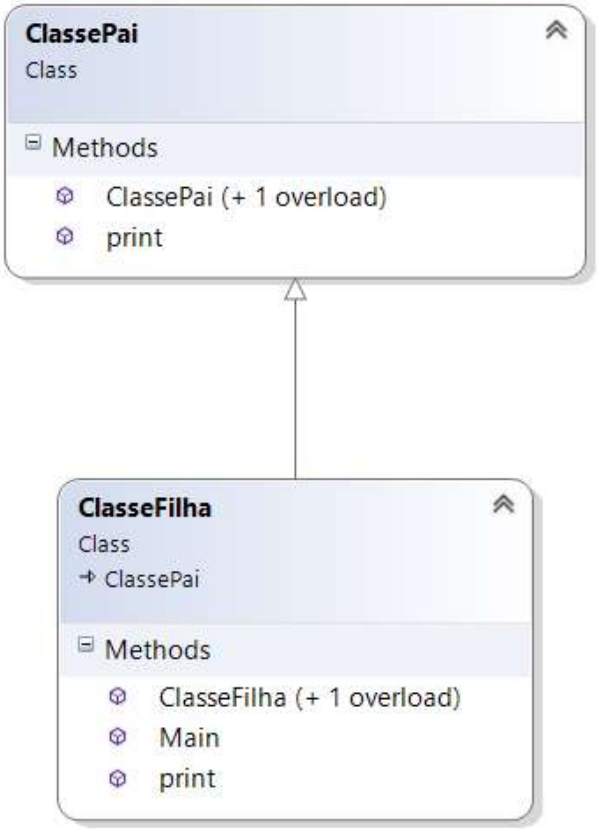
IPC @JB

```
public class ClassePai
{
    // Membros da classe
}
internal class ClasseFilha: ClassePai
{
    // Membros da classe
}
```



isec Engenharia DEIS - Instituto Superior de Engenharia de Coimbra

C#: Herança

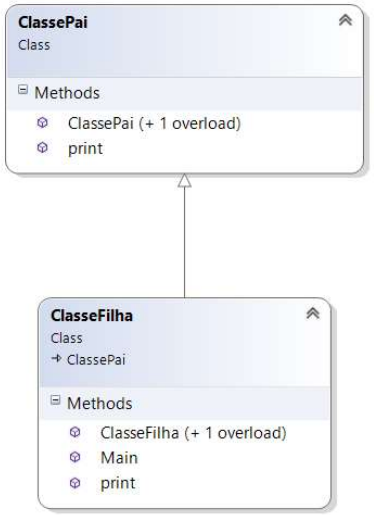



```
public class ClassePai
    {
        public ClassePai()
        {
            Console.WriteLine("Construtor Pai");
        }
        public ClassePai(string texto)
        {
            Console.WriteLine("Construtor Pai " + texto);
        }
        public void print()
        {
            Console.WriteLine("Print() da classe Pai.");
        }
    }
```

isec Engenharia DEIS - Instituto Superior de Engenharia de Coimbra

C#: Herança

Ocultar qualquer implementação do método print() acima dele




```
public class ClasseFilha : ClassePai     {
        public ClasseFilha() : base("Mensagem Ola...")  {
            Console.WriteLine("Construtor Filha");
        }
        public ClasseFilha(string x) : base(x) {
            Console.WriteLine("Construtor Filha com string");
        }
        public new void print()  {
            base.print();
            Console.WriteLine("Print() da classe filha");
        }
        public static void Main()  {
            ClasseFilha filha = new ClasseFilha();
            ClasseFilha filha2 = new ClasseFilha("String Parametro");
            filha.print();
            ((ClassePai)filha).print();
        }
    }
```


isec Engenharia DEIS - Instituto Superior de Engenharia de Coimbra

C#: Herança

- Membros da classe **base** com acesso ***private*** não são acessíveis da classe derivada
  - Nível de proteção ***protected*** permite que as classes derivadas tenham acesso ao membro, sem o tornar público
- Membros da classe base podem ser virtuais (***virtual***)
  - A classe que herda uma classe com membros virtuais, pode ser ***overridden*** pela classe que a herda -> i.e. altera o método original existente na classe pai
  - Isso, permite que a classe derivada possa ter uma implementação alternativa para o membro.

Programação Web – 2023/2024 – LEI, LCE+M – ISEC/IPC @ 3

isec Engenharia DEIS - Instituto Superior de Engenharia de Coimbra

C#: Classes e Métodos Abstratos

- A Classe *Base* pode ser **Abstrata**
  - Uma Classe Abstrata é uma classe conceptual no qual se define funcionalidades para que as subclasses (que a herdam) possam implementá-las de forma **não obrigatória**
- Uma *classe abstrata* <u>pode ser somente herdada</u> e <u>não instanciada</u>
- Classes abstratas podem ter membros abstratos, os quais não tem implementação na classe base, portanto, tem de ser fornecida uma implementação na classe derivada caso sejam usados na classe derivada

Programação Web – 2023/2024 – LEI D+CG+PL – ISEC/IPC @JB

isec Engenharia DEIS - Instituto Superior de Engenharia de Coimbra

C#: Classes e Métodos Abstratos(2)

- Quando uma classe possui pelo menos um método abstrato esta deve também ser declarada como abstrata
- Classes bases abstratas podem já fornecer implementação dos membros
- Uma classe abstrata pode herdar de outra classe abstrata

```
public abstract class nome_da_classe
    {
        // Membros da classe, podem ser abstratos
    }
```

Web – 2023/2024 – LEI/LDSE+PL – ISEC/JRC @JR

isec Engenharia DEIS - Instituto Superior de Engenharia de Coimbra

C#: Classes e Métodos Abstratos(3)

- Um método abstrato é um método que ***não possui implementação*** na classe abstrata, apenas possui a definição da sua **assinatura**
- A sua implementação deve ser feita na **classe derivada**
- Os métodos podem ou não ser abstratos, mas quando estes são abstratos, a sua **implementação** é **obrigatória**!

Programação Web – 2023/2024 – LEID+CE+PL – ISEC/IPC @ JR

isec Engenharia DEIS - Instituto Superior de Engenharia de Coimbra

C#: Membros virtuais

- Membro ***virtual***
  - É um membro na classe base que define uma **implementação por omissão** que pode ser alterada (*overridden*) pela classe derivada
  - **Por contraste, um método abstrato**, é um membro na classe base que não fornece qualquer implementação por omissão, mas fornece a sua assinatura.

Programação Web – 2023/2024 – LEI+CET+PL – ISEC/IPC @JB

isec Engenharia DEIS - Instituto Superior de Engenharia de Coimbra

Classes Abstratas e *Sealed*

- A palavra chave **abstract** permite criar classes e membros de classes incompletas que têm de ser implementados na classe derivada

- A palavra chave ***sealed*** evita que a classe ou membros virtuais sejam herdadados

Programação Web – 2023/2024 – LEI DAE+PL – ISEC/IPC @JB

C#: *Interfaces*

- Para a definição de ***Interfaces***, utiliza-se a palavra chave ***interface***

```
public interface IMeuInterface
{
    // Membros da Interface
}
```

- Todas as interfaces são **públicas**
- Na versão do C# 9, os métodos declarados numa interface <u>podem possuir implementação</u> e são sempre públicos
- Quando uma interface é referenciada tem de se implementar a funcionalidade dos seus métodos

Programação Web – 2023/2024 – ISEC/IPC @ JR

isec Engenharia DEIS - Instituto Superior de Engenharia de Coimbra

C#: Interfaces (1)

```
public class MyClass : IUmaInterface
{
      // Membros da class
}
```

```
public class MyClass : MyBase, IUma1Interface, IUma2Interface
{
      // Membros da class
}
```


isec Engenharia DEIS - Instituto Superior de Engenharia de Coimbra

C#: Herança e Interfaces

- Herdar uma *Interface*

```
public class MyClass : IMeuInterface
 {
     // Membros da class
 }
```

```
public class MyClass : MyBase, IMeuInterface, IMeu2Interface
   {
       // Membros da class
   }
```

BC @JB

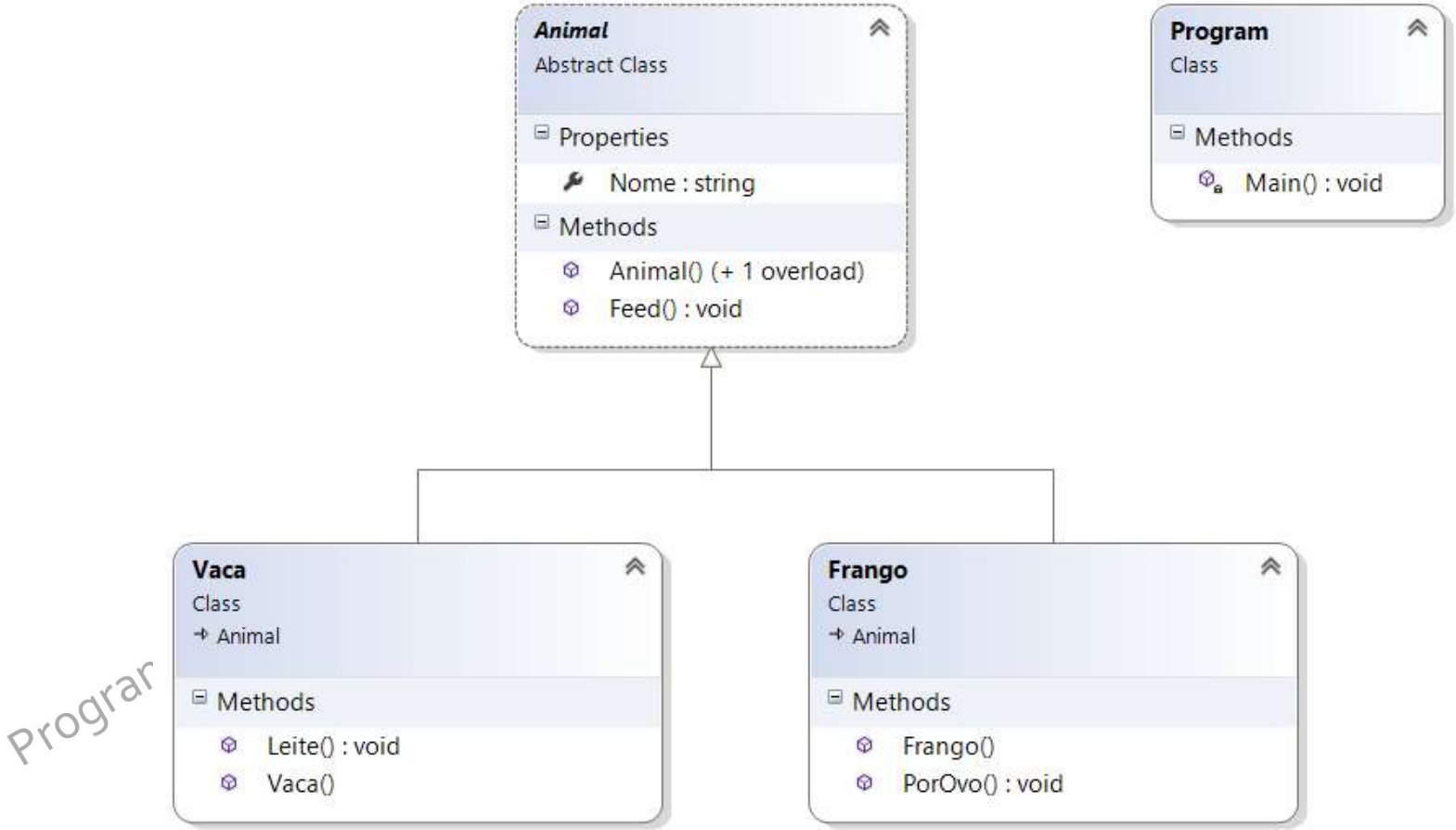
C#: Interfaces

Implicitamente public e abstract

```
public interface IMeuInterface
{
        // Membros da Interface
        int método();
}
```

Programação Web – 2023/2024 – LEI D+GE+PL – ISEC/IPC @JB

C#: Classes e Métodos Abstratos - Exemplo



Prograr

isec Engenharia DEIS - Instituto Superior de Engenharia de Coimbra

C#: Classes e Métodos Abstratos - Exemplo

```
using System;
namespace CSharpExemplo
{
    public abstract class Animal
    {
        public string Nome { get; set; }
        public Animal()
        {
            Nome = "Animal sem nome";
        }
        public Animal(string novoNome)
        {
            Nome = novoNome;
        }
        public void Feed()
        {
            Console.WriteLine($"{Nome} foi alimentado!");
        }
    }
}
```

Prog

C#: Classes e Métodos Abstratos - Exemplo

```
using System;
namespace CSharpExemplo
{
    public abstract class Animal
    {
        public string Nome { get; set; }
        public Animal()
        {
            Nome = "Animal sem nome";
        }
        public Animal(string novoNome)
        {
            Nome = novoNome;
        }
        public void Feed()
        {
            Console.WriteLine($"{Nome} foi
alimentado!");
        }
    }
}
```

```
using System;
namespace CSharpExemplo {
    public class Frango : Animal {
        public void PorOvo() {
            Console.WriteLine($"{nome} pôs um ovo.");
        }
        public Frango(string novoNome) : base(novoNome)  { }
    }
}
```

```
using System;
namespace CSharpExemplo {
  public class Vaca : Animal
  {
    public void Leite() {
      Console.WriteLine($"A vaca {nome} foi ordenhada! ");
    }
    public Vaca(string novoNome) : base(novoNome) { }
  }
}
```

Programa

C#: Classes e Métodos Abstratos - Exemplo

```
static void Main(string[] args)
        {
            Vaca minhaVaca = new Vaca("LeaVaca");
            Frango meuFrango2 = new Frango("ZetaFrango");
            Animal meuFrango = new Frango("NoaFrango");
            Console.WriteLine($"Criei 3 objectos: {minhaVaca.Nome},
                        {meuFrango.Nome} e {meuFrango2.Nome}");

            meuFrango2.PorOvo();
            ((Frango)meuFrango).PorOvo();
            meuFrango.Feed();
            minhaVaca.Feed();
        }
```

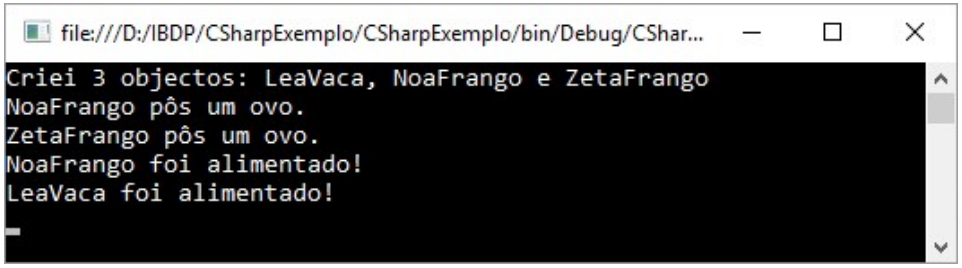
Programação Web – 2023/2024 – LEI+D+CE+PL - ISEC/IPC @JB

isec Engenharia DEIS - Instituto Superior de Engenharia de Coimbra

C#: Classes e Métodos Abstratos - Exemplo

```
static void Main(string[] args)
        {
            Vaca minhaVaca = new Vaca("LeaVaca");
            Frango meuFrango2 = new Frango("ZetaFrango");
            Animal meuFrango = new Frango("NoaFrango");
            Console.WriteLine($"Criei 3 objectos: {minhaVaca.Nome},
                        {meuFrango.Nome} e {meuFrango2.Nome}");

            meuFrango2.PorOvo();
            ((Frango)meuFrango).PorOvo();
            meuFrango.Feed();
            minhaVaca.Feed();
        }
```

Saída?



Programação Web - 2023/2024 – LEID+CE+PL ISEC/IPC @JB

DEIS - Instituto Superior de Engenharia de Coimbra

C#: Classes e Métodos Abstratos - Exemplo

```
namespace CSharpExemplo {
    public abstract class Animal
    {
        protected string nome;
        public string Nome
        {
            get { return nome; }
            set { nome = value; }
        }
        public Animal()
        {
            nome = "Animal sem nome";
        }
        public Animal(string novoNome)
        {
            nome = novoNome;
        }

        public virtual void Feed();

    }
}
```

Programação Web – 2023/2024 – LEI D+CE+PL – ISEC/IPC @JB

isec Engenharia DEIS - Instituto Superior de Engenharia de Coimbra

C#: Classes e Métodos Abstratos - Exemplo

```
namespace CSharpExemplo {
    public abstract class Animal
    {
        protected string nome;
        public string Nome
        {
            get { return nome; }
            set { nome = value; }
        }
        public Animal()
        {
            nome = "Animal sem nome";
        }
        public Animal(string novoNome)
        {
            nome = novoNome;
        }

        public virtual void Feed() {
          Console.WriteLine($"{Nome} foi alimentado!");
        }
    }
}
```

Programação Web – 2023/2024 – LEI D+CE+PL – ISEC/IPC @JB

isec Engenharia DEIS - Instituto Superior de Engenharia de Coimbra

C#: Classes e Métodos Abstratos - Exemplo

```
using System;
namespace CSharpExemplo {
    public class Frango : Animal
    {
        public void PorOvo() {
            Console.WriteLine($"{Nome} pôs um ovo.");
        }
        public Frango(string novoNome) : base(novoNome){ }

        public override void Feed()
         {
           Console.WriteLine($"{Nome} foi alimentado com milho!");
         }
    }
}
```


isec Engenharia DEIS - Instituto Superior de Engenharia de Coimbra

C#: Classes e Métodos Abstratos - Exemplo

```
using System;
namespace CSharpExemplo {
    public cl
    {
        publi
        }
        publi
```

```
Criei 3 objectos: LeaVaca, NoaFrango e ZetaFrango
ZetaFrango pôs um ovo.
NoaFrango pôs um ovo.
NoaFrango foi alimentado com milho!
LeaVaca foi alimentado!
```

```
        public override void Feed()
         {
            Console.WriteLine($"{Nome} foi alimentado com milho!");
         }
    }
}
```


isec Engenharia DEIS - Instituto Superior de Engenharia de Coimbra

C#: Classes e Métodos Abstratos - Exemplo

```
namespace CSharpExemplo {
    public abstract class Animal
    {
        protected string nome;
        public string Nome
        {
            get { return nome; }
            set { nome = value; }
        }
        public Animal()
        {
            nome = "Animal sem nome";
        }
        public Animal(string novoNome)
        {
            nome = novoNome;
        }

        public abstract void Feed();

    }
}
```

É OBRIGATÓRIO!

Programação Web – 2023/2024 – LEI D+CE+PL – ISEC/IPC @JB

isec Engenharia DEIS - Instituto Superior de Engenharia de Coimbra

C#: Classes e Métodos Abstratos - Exemplo

```
using System;
namespace CSharpExemplo {
    public class Frango : Animal
    {
        public void PorOvo() {
            Console.WriteLine($"{Nome} pôs um ovo.");
        }
        public Frango(string novoNome) : base(novoNome){ }

        public override void Feed()
         {
           Console.WriteLine($"{Nome} foi alimentado com milho!");
         }
    }
}
```


isec Engenharia DEIS - Instituto Superior de Engenharia de Coimbra

C#: Classes e Métodos Abstratos - Exemplo

```
using System;
namespace CSharpExemplo {
    public class Vaca : Animal
    {
        public void Leite() {
            Console.WriteLine($"A vaca {nome} foi ordenhada! ");
        }
        public Vaca(string novoNome) : base(novoNome) { }

        public override void Feed()
        {
            Console.WriteLine($"{Nome} foi alimentado com erva!");
        }
    }
}
```

C#: Classes e Métodos Abstratos - Exemplo

```
static void Main(string[] args)
        {
            Vaca minhaVaca = new Vaca("LeaVaca");
            Frango meuFrango = new Frango("ZetaFrango");
             Animal meuAnimal = new Animal("NovoAnimal");
             object meuObjecto = new Frango("FrangoObjecto");
             Animal meuFrango2 = new Frango("NoaFrango");
             meuFrango.PorOvo();
             meuFrango2.PorOvo();
         }
```

Programação Web – 2023/2024 – LEI/LDCE+PL – ISEC/IPC @JB

isec Engenharia DEIS - Instituto Superior de Engenharia de Coimbra

C#: Classes e Métodos Abstratos - Exemplo

```
static void Main(string[] args)
        {
            Vaca minhaVaca = new Vaca("LeaVaca");
            Frango meuFrango = new Frango("ZetaFrango");
              Animal meuAnimal = new Animal("NovoAnimal");
              object meuObjecto = new Frango("FrangoObjecto");
              Animal meuFrango2 = new Frango("NoaFrango");
              meuFrango.PorOvo();
             meuFrango2.PorOvo(); ((Frango)meuFrango2).PorOvo();

          }
```

Programação Web – 2023/2024 – LEILD+CET+PL – ISEC/IPC @JB

isec Engenharia DEIS - Instituto Superior de Engenharia de Coimbra

C#: Será que isto compila?

```
object meuObjecto= new Frango("NoaFrango");
Quadrado quadrado = (Quadrado)meuObjecto;
```

Programação Web – 2023/2024 – LEI+LCE+PL – ISEC/IPC @JB

isec Engenharia DEIS - Instituto Superior de Engenharia de Coimbra

C#: Será que isto compila?

```
object m                                  NoaFrango");
Quadrado                                  euObjecto;
```

Programaç
JB

Palavra chave *as* em C#

- Os *cast* explícitos são avaliados em *runtime* e não em tempo de compilação
- ***as*** tenta converter um objeto para um tipo específico e retorna ***null*** se a conversão falhar.

Programação Web – 2023/2024 – LEI D+CE+PL – ISEC/IPC @

```
object meuObjecto = new Frango("FrangoObjecto");
Quadrado quadrado = meuObjecto as Quadrado;
if (quadrado==null)
{
    Console.WriteLine("O objecto não é um quadrado...");
}
```

isec Engenharia DEIS - Instituto Superior de Engenharia de Coimbra

Palavra-chave *is* em C#

- ***is*** permite de forma rápida determinar se um determinado **objecto** é compatível com um **tipo** e devolve *false* caso não sejam

```
object meuObjecto = new Frango("FrangoObjecto");
Quadrado quadrado = meuObjecto as Quadrado;
if (quadrado is Frango)
{
    Console.WriteLine("O objecto é um Frango...");
}
```

Programação Web – 2023/2024 – LEI D+CE+PL – ISEC/IPC @ JB

isec Engenharia DEIS - Instituto Superior de Engenharia de Coimbra

Programação Web

C#: Manipulação de Excepções

Departamento de Engenharia Informática e de Sistemas
Instituto Superior de Engenharia de Coimbra/Instituto Politécnico de Coimbra

C#: Manipulação de Excepções

- ***try***
- ***catch***
- ***finally***
- ***throw***

```
try
{
    // instruções que podem causar excepção
}
catch( ExceptionName e1 )
{
    // Codigo que manipula a excepção
}
catch( ExceptionName e2 )
{
    // Codigo que manipula a excepção
}
catch( ExceptionName eN )
{
    // Codigo que manipula a excepção
}
finally
{
    // Codigo a ser executado
}
```

Programação Web – 2023/2024 – LEID+CE+PL – ISEC/IPC @JB

isec Engenharia DEIS - Instituto Superior de Engenharia de Coimbra

C#: Manipulação de Excepções

- E o **Throw ?**
  - ***throw***
  - ***throw ex***

```
try
{
    //algum código
}
catch
{
    throw;
}
```

```
try
{
    //algum código
}
catch (Exception ex)
{
    throw ex;
}
```

- ***throw*** **-** quando se utiliza o ***throw*** passamos a mesma excepção "para a frente" e com isso outro trecho de código pode capturá-la e tratar essa excepção original retendo todas as informações necessárias com o stack trace.

- ***throw ex*** **- q**uando se utiliza o ***throw ex***, para-se a excepção ali e após se fazer alguma operação, é lançada outra excepção a partir desse ponto. Com isso, a informação de onde veio a excepção original é perdida, como se tivesse ocorrido uma nova excepção.

- De preferência devemos usar a primeira opção.

Programação Web 2023/2024 – D

isec Engenharia DEIS - Instituto Superior de Engenharia de Coimbra

C# Manipulação de Excepções - throw

```
private int idade;
public int Idade { get { return idade; }
                   set {
                       if (value< 10 || value> 30)
                             throw new Exception("Idade Inválida!");
                       else
                          idade = value;
            }
        }
```

```
Aluno aluno = new Aluno(12342, "NomeAluno", Tipo.Normal);
try
{
    aluno.Idade = 50;
} catch(Exception e)
{
    Console.WriteLine(e.Message);
}
```

Programação W

isec Engenharia DEIS - Instituto Superior de Engenharia de Coimbra

C# Manipulação de Exceçpões – Ex1

```
Aluno al = null;
try {
    Console.WriteLine(al.ToString());
}
catch (ArgumentNullException ex) {
    Console.WriteLine("ArgumentNullException: " + ex.Message);
}
catch (ArgumentException ex) {
    Console.WriteLine("ArgumentException: " + ex.Message);
}
catch (Exception ex) {
    Console.WriteLine("Exception: " + ex.Message);
}
finally {
    Console.WriteLine("Executa sempre…");
}
```

C# Manipulação de Excepções – Ex1

```
class DivNumeros
{
    int result;
    DivNumeros() {
        result = 0;
    }
    public void Divisao(int num1, int num2)
    {
        try {  result = num1 / num2;
        }
        catch (DivideByZeroException e)
            { Console.WriteLine("{0}", e);
        }
        finally {
            Console.WriteLine("Resultado: {0}", result);
        }
    }
}
```

@JB

O que aconteceria se se fizesse esta chamada?

```
static void Main(string[] args)
 {
  DivNumeros d = new DivNumeros();
  d.Divisao(25, 0);
  Console.ReadKey();
 }
```

Iria tentar executar uma divisão por Zero
O que é impossível e como tal geraria uma excepção

isec Engenharia DEIS - Instituto Superior de Engenharia de Coimbra

C# Manipulação de Excepções – Ex2

```
public class Idade
    {
        int idade = 12;

        public void mostraIdade()
        {
            if (idade < 18)
            {
                throw (new MaiorIdadeException("Idade inferior a 18 anos!"));
            }
            else
            {
                Console.WriteLine("Idade: {0}", idade);
            }
        }
    }
```

```
public class MaiorIdadeException : Exception
    {
        public MaiorIdadeException(string message) : base(message)
        { }
    }
```


isec Engenharia DEIS - Instituto Superior de Engenharia de Coimbra

C# Manipulação de Excepções – Ex2

```
class VerificaIdade
{
    static void Main(string[] args)
    {
        Idade idade = new Idade();
        try
        {
            idade.mostraIdade();
        }
        catch (MaiorIdadeException e)
        {
            Console.WriteLine("MaiorIdadeException: {0}", e.Message);
        }
        Console.ReadKey();
    }
}
```

```
public class MaiorIdadeException : Exception
  {
      public MaiorIdadeException(string message) : base(message)
      { }
  }
```

isec Engenharia DEIS - Instituto Superior de Engenharia de Coimbra

Programação Web

Aulas Teóricas – Capítulo 1 – 1.2

1º Semestre - 2023/2024

Departamento de Engenharia Informática e de Sistemas
Instituto Superior de Engenharia de Coimbra/Instituto Politécnico de Coimbra

Estes Slides foram reformulados para o presente Ano Lectivo.
No entanto para a elaboração dos mesmos foram incluídos partes de conteúdos utilizados nesta Unidade Curricular em Anos Lectivos anteriores os quais tiveram contribuições de diversos docentes, nomeadamente dos docentes Jorge Barbosa, Cristiana Areias, Francisco Leite!
A matéria e alguns exemplos seguem as orientações existentes nos livros indicados na Bibliografia!

Programação Web - 2022/2023 – LEI DMEF+C – ISEC/DC @JB

Programação Web

C# – Conceitos Avançados - 2ª Parte

Departamento de Engenharia Informática e de Sistemas
Instituto Superior de Engenharia de Coimbra/Instituto Politécnico de Coimbra

Evolução do C#

- **C# é uma linguagem de programação em Constante Evolução**
  - Actualmente está na Versão 11
- Ao contrário de muitas outras linguagens de programação, o C# a cada nova Release apresenta bastantes inovações
  - Em muitos aspectos, a actual versão "nada tem a ver" com a versão original
  - Do mesmo modo paradigmas das linguagens "mães" do C# foram alteradas nas novas versões

Programação Web – 2023/2024 – LEI D+CE+PL – ISEC/IPC @ 3

isec Engenharia DEIS - Instituto Superior de Engenharia de Coimbra

Evolução do C#

- Ela é hoje a principal linguagem de desenvolvimento da Microsoft para *desktop*, *web* e *mobile*

- Muitas das novas funcionalidades que tem incorporado rapidamente são copiadas noutras linguagens
  - Também tem incorporado muitas inovações que têm aparecido noutras linguagens

Programação Web – 2023/2024 – LEI D+CE+PL – ISEC/IPC @ 2

isec Engenharia DEIS - Instituto Superior de Engenharia de Coimbra

Evolução do C#

- Esta constante evolução do C# faz com que existem novas formas de se resolver velhos problemas
- Desse modo com o C# tem de se ter consciência de que só porque algo até hoje foi feito de determinada maneira e "funcionava", não significa que seja a melhor forma de implementar pois pode haver um modo melhor
  - Obriga a uma constante actualização de conhecimentos
  - Consegue-se com "menos fazer mais"

Programação Web – 2023/2024 – LEI/LD+CE/PL – ISEC/IPC @ JP

isec Engenharia DEIS - Instituto Superior de Engenharia de Coimbra

Evolução do C#

- A contínua evolução dos frameworks .NET e .NET Core no sentido da sua unificação, que já é em parte uma realidade a partir do .NET 5 e a recente nova release .NET 6, LTS, (a 7 é a atual mas é STS) "obrigam" a ter-se em consideração na programação, em particular para a Web, de novos paradigmas
  - O conhecimento do C# na versão 11 (em desenvolvimento a V12 e prevista para ser lançada até ao final do corrente ano) permite acompanhar as necessárias mudanças que a unificação dos frameworks .NET e .NET Core impõe

Programação Web – 2023/2024 – LEI-CE+PL – ISEC/IPC @ DEIS

top level statments do C#

- As novas instruções do C# também são designadas por Instruções de Nível Superior - *top level statements*
  - Nota: Estas novas instruções não aparecerem necessariamente só na versão 9 mas podem ter aparecido <u>até</u> esta versão
- Não há necessidade de se ter um ficheiro com a designação *Program.cs*
  - Assim, o *main* pode estar num ficheiro com outro nome

Programação Web – 2023/2024 – LEI+CE+RI – ISEC/UPC @ JB

isec Engenharia DEIS - Instituto Superior de Engenharia de Coimbra

top level statments do C#

- Não há a necessidade de inserir o esqueleto de código – *scaffolding* - como é obrigatório em Java, C++ e em versões anteriores do C#
  - De certo modo isto permite que o C# 9 se comporte como Python, Ruby, etc
- Se por exemplo no Visual Studio 2022 (ou superior) criarmos um projecto em modo *Console*, o VS automaticamente criará um ficheiro *Program.cs* onde estará o *main* com o código mostrado de seguida

Programação Web – 2023/2024 – LEID+CE+PL – ISEC/IPC @ JS

isec Engenharia DEIS - Instituto Superior de Engenharia de Coimbra

top level statments do C#

```
using System;
namespace TesteConsoleCore
{
    class Program
    {
        static void Main(string[] args)
        {
            Console.WriteLine("Hello World!");
        }
    }
}
```

Programação Web – 2023/2024 – LEI D+CE+PL – ISEC/IPC @JB

isec Engenharia DEIS - Instituto Superior de Engenharia de Coimbra

top level statments do C#

- O código apresentado é o “tradicional que o VS cria automaticamente como *scaffolding* dos ficheiros C# de um projecto e estará correcto e completamente funcional e permite exibir a tradicional mensagem do 1º programa desenvolvido numa nova linguagem que se esteja a apreender:

Hello World!

- No entanto, usando os novos recursos inerentes às *top level statements* bastaria o código que se apresenta a seguir

Programação Web – 2023/2024 – LEI+LEI-D+CE+PL – ISEC/MC@JB

isec Engenharia DEIS - Instituto Superior de Engenharia de Coimbra

top level statments do C#

```
using System;
Console.WriteLine("Hello World!");
```

Ou até mesmo usando somente:

```
System.Console.WriteLine("Hello World!");
```

- Experimente esta possibilidade criando no VS um projecto em modo Console selecionado o *framework .NET 5* e introduzindo uma das variantes acima e de seguida corra essa aplicação (Ctrl+F5)
- Depois de executar, experimente nas Propriedades do projecto do VS mudar o framework para o *.Net Core 3.1* (o anterior ao 5) e verificará que já não compilará, dando erro

Programação Web – 2023/2024 – LEI D+CG+PL – ISEC/IPC @JB

top level statments do C#

- As top *level statments*, ***tls***, dispensam então a habitual utilização de código "excessivo" e permitem por isso escrever scripts simples e concisos que podem ser utilizados em qualquer classe do *framework .NET*, retornar valores e executar código assíncrono

  - Actualmente ainda existe a restrição de só se poder ter um único ficheiro de código no projeto que utilize as ***tls***

Programação Web – 2023/2024 – LEI+LCE+mc+ISEC+RC @ JS

isec Engenharia DEIS - Instituto Superior de Engenharia de Coimbra

Strings no C#

- Uma *string* ou cadeia de caracteres é uma coleção sequencial de caracteres usada para representar texto
- Nas plataformas *.NET* e *.NET Core*, o texto de uma *string* é representado como uma sequência de unidades de código UTF-16
  - Desse modo, é possível nestas plataformas armazenar em memória até 2GB (aproximadamente mil milhões de caracteres)

Programação Web – 2023/2024 – LEI+LEI-PL – ISEC/RC @JB

Strings no C#

- **Interpolação de strings**
- No C# 6.0 foi introduzido um recurso chamado interpolação de strings.
  - Nas versões anteriores do C# a concatenação de strings era mais trabalhosa
  - Exemplo com o método *Format* da classe *String* :

```
var aluno = new Aluno();
var mensagem = string.Format("{0} é o curso do aluno {1}", aluno.Curso,aluno.Nome);
```

Programação Web – 2023/2024 – LEI+LD+CE+PL – ISEC/IPC @JB

Strings no C#

- **Interpolação de strings**

- Nas versões actuais do C#, podemos utilizar a seguinte alternativa, que é mais simples

```
var aluno = new Aluno();

var mensagem = $"{aluno.Curso} é o curso do aluno {aluno.Nome}";
```

- Para usarmos a interpolação de *strings*, é necessário preceder a *string* com o caractere $

Programação Web – 2023/2024 – LEI D+LE+PL – ISEC/IPC @JB

isec Engenharia DEIS - Instituto Superior de Engenharia de Coimbra

Strings no C#

- **Caractere @**
  - Já referimos a utilização do @ para a construção de *strings* mais facilmente sem recorrer tanto às sequências de *escape*, usando antes *verbatim strings*
    - Assim, em vez de se utilizar

```
string myFileName = "C:\\myfolder\\myfile.txt";
```

pode-se utilizar

```
string myFileName = @"C:\myfolder\myfile.txt";
```

Programação Web – 2023/2024 – LEID+CET PL – ISEC/IPC @JB

isec Engenharia DEIS - Instituto Superior de Engenharia de Coimbra

Strings no C#

```
using System;
namespace TesteComStrings
{
    class Program
    {
        static void Main(string[] args)
        {
            var mensagem = "Caro aluno,\n\n\tA matéria tratada nesta secção respeita à
construção de strings de um modo mais simplificado. \nAs facilidades oferecidas permitirão
a escrita de um código mais legível.\n\nUtilize estas novas caracteristicas!\nBom estudo!";

            Console.WriteLine(mensagem);
        }
    }
}
```

Programação Web – 2023/2024 – LEI D+CE+PL – ISEC/IPC @JB

isec Engenharia DEIS - Instituto Superior de Engenharia de Coimbra

Strings no C#

- Pelo contrário, no exemplo seguinte a *string* é construída utilizando-se o @ para facilitar a formatação da mesma

  - Observe que a formatação, espaços, mudanças de linha, etc, está incorporada na própria *string*

Programação Web – 2023/2024 – LEI D+CE+RL – ISEC/IPC @JB

Strings no C#

```
using System;
namespace TesteComStrings
{
    class Program
    {
        static void Main(string[] args)
        {
            var mensagem = @"Caro aluno,

        A matéria tratada nesta secção respeita à construção de strings de um modo mais
simplificado.
As facilidades oferecidas permitirão a escrita de um código mais legível.

Utilize estas novas caracteristicas!
Bom estudo!";
            Console.WriteLine(mensagem);
        }
    }
}
```

Programação Web – 2023/2024 – LEI D+CE+PL – ISEC/IPC @JB

isec Engenharia DEIS - Instituto Superior de Engenharia de Coimbra

Strings no C#

- A saída, quer de uma versão quer da outra, será exactamente a mesma e é a mostrada na figura

- Eventualmente formatar a *string* usando o caracter @ como foi feito na 2ª versão tornará essa formatação mais fácil de ser feita e de manutenção futura também mais fácil

23/2024 – LEI-D+CE-PL – ISEC/IPC @JB

Selecionar Microsoft Visual Studio Debug Console

```
Caro aluno,

        A matéria tratada nesta secção respeita à construção de strings de um modo mais simplificado.
As facilidades oferecidas permitirão a escrita de um código mais legível.

Utilize estas novas caracteristicas!
Bom estudo!
```

Strings no C#

- A utilização do @ poderá ser útil quando se criam *queries SQL*

  var alunocursosql = @"SELECT *

  FROM Aluno

  WHERE curso = 'Eng.ª Informática'";

- O @ permite que essas *queries* sejam mais facilmente legíveis

Programação Web – 2023/2024 – LEI D+CE+TL – ISEC/IPC @

Strings no C#

- A utilização do @ também permite simplificar a passagem para um método de uma *string* de conexão de uma base de dados

optionsBuilder.UseSqlServer(@"Server=(localdb)\mssqllocaldb;Database=Loja;Trusted_Connection=True;");

Programação Web – 2023/2024 – LEI D+CE+PL – ISEC@DEIS

isec Engenharia DEIS - Instituto Superior de Engenharia de Coimbra

Strings no C#

- Conversão de *Strings*
- A conversão de *strings* em *arrays* de *bytes* ou em *array* de *strings* é algo habitualmente feito
  - O novo *framework* dispõe de uma panóplia de métodos específicos para isso e que facilitam bastante a programação
- Esta facilidade, é bastante útil quando seja necessário consumir métodos das classes dos *frameworks .NET* e *.NET Core* ou de bibliotecas de terceiros pois alguns métodos de classes destes frameworks obrigam a que uma *string* seja passada como um *array* de *bytes*

Programação Web – 2023/2024 – LEI DACE+PL – ISEC/IPC @JB

Strings no C#

- Conversão de *strings* em *arrays* de *bytes*
- Para essa conversão utiliza-se o método estático *GetBytes*, presente na classe *Unicode* e na classe *UTF8* do namespace *System.Text.Encoding*
  - Exemplo de conversão de *string* para *byte[]*

```
string mensagem = "Programação Web com C# e ASP.Net Core";
byte[] conteudo =
System.Text.Encoding.Unicode.GetBytes(mensagem) ;
```

Programação Web – 2023/2024 – LEI D+CE+PL – ISEC/IPC @JB

isec Engenharia DEIS - Instituto Superior de Engenharia de Coimbra

Strings no C#

- Conversão de *arrays* de *bytes* em *strings*
- Para essa conversão utiliza-se o método estático *GetString*, presente na classe *Unicode* e na classe *UTF8* do namespace *System.Text.Encoding*

string strfinal = Encoding.Unicode.GetString(strconvertida, 0, strconvertida.Length);

- Para ter uma melhor percepção de como estas conversões são feitas, nomeadamente de *array* de *bytes* para *string,* analise e execute o código exemplo que se segue

Programação Web – 2023/2024 – LEI D+CE+PL – ISEC/IPC @JB

Strings no C#

```
using System;
using System.Text;
namespace TesteComStrings
{
    class Program
    {
        static void Main(string[] args)
        {
            string strinicial = "Programação Web com C# e ASP.Net Core";
            Console.Write("String Inicial:");
            Console.WriteLine(strinicial);
            byte[] strconvertida = System.Text.Encoding.Unicode.GetBytes(strinicial);
            Console.WriteLine("\nConcatenando todos os Bytes em uma string de bytes: ");
            StringBuilder builder = new StringBuilder();
            for (int i = 0; i < strconvertida.Length; i++)
            {
                builder.Append(strconvertida[i].ToString("x2"));
            }
            Console.WriteLine(builder.ToString());
            Console.WriteLine("\nUsando a classe BitConverter para colocar todos os bytes numa string:");
            string bitString = BitConverter.ToString(strconvertida);
            Console.WriteLine(bitString);
            Console.Write("\nObtendo a string inicial em Unicode a partir dos Bytes: ");
            string strfinal = Encoding.Unicode.GetString(strconvertida, 0, strconvertida.Length);
            Console.WriteLine(strfinal);

            Console.ReadKey();
        }
    }
}
```

Programação Web – 2023/2024 – LEI D+CE+PL – ISEC/IPC @JB

Strings no C#

- Execute o código e interprete-o
  - A saída deste código de conversão é a seguinte:

LEI D CE+PL – ISEC/IPC @JB

```
C:\Users\Jorge\source\repos\TesteConsoleCore\bin\Debug\net5.0\TesteConsoleCore.exe
String Inicial:Programação Web com C# e ASP.Net Core

Concatenando todos os Bytes em uma string de bytes:
500072006f006700720061006d006100e700e3006f002000570065006200200063006f006d002000430023002000650020004100530050002e004e00
650074002000430006f0072006500

Usando a classe BitConverter para colocar todos os bytes numa string:
50-00-72-00-6F-00-67-00-72-00-61-00-6D-00-61-00-E7-00-E3-00-6F-00-20-00-57-00-65-00-62-00-20-00-63-00-6F-00-6D-00-20-00-
43-00-23-00-20-00-65-00-20-00-41-00-53-00-50-00-2E-00-4E-00-65-00-74-00-20-00-43-00-6F-00-72-00-65-00

Obtendo a string inicial em Unicode a partir dos Bytes: Programação Web com C# e ASP.Net Core
```

isec Engenharia DEIS - Instituto Superior de Engenharia de Coimbra

Strings no C#

- **Partição de uma *string* num *array* de *strings***
- Para se efectuarem partições de *strings* em *arrays* de *strings* utiliza-se o método *Split* da classe *String*

  - Utiliza-se uma lista de delimitadores parametrizada: os *split delimiters,* que podem ser um único caractere, um *array* de caracteres ou até mesmo um *array* de *strings*.

Programação Web – 2023/2024 – LEI D+CD PL – ISEC/IPC @JR

isec Engenharia DEIS - Instituto Superior de Engenharia de Coimbra

Strings no C#

- Utilização da forma mais simples de uso do método Split:

```
using System;
namespace TeseComStrings
{
    class Program
    {
        static void Main(string[] args)
        {
            Console.WriteLine("Unidades Curriculares:");
            Console.WriteLine();
            //Separação entre cada UC é feita apenas por vírgulas
            string unidadesCurriculares = "Programação Web, Ética e Deontologia, Projecto ou Estágio, Introdução às Bases de
Dados";
        string[] lista = unidadesCurriculares.Split(", ");
            foreach (string uc in lista)
                Console.WriteLine(uc);

            Console.ReadKey();
        }
    }
}
```

Programação Web – 2023/2024 – LEI D+CE+PL – ISEC/IPC @JB

isec Engenharia DEIS - Instituto Superior de Engenharia de Coimbra

Strings no C#

- Utilização da forma mais simples de uso do método Split:

```
using System;
namespace TeseComStrings
{
    class Program
    {
        static void Main(string[] args)
        {
            Console.WriteLine("Unidades Curriculares:");
            Console.WriteLine();
            //Separação entre cada UC é feita apenas por vírgulas
            string unidadesCurriculares = "Programação Web, Ética e Deontologia, Projecto ou Estágio, Introdução às Bases de
Dados";
        string[] lista = unidadesCurriculares.Split(", ");
            foreach (string uc in lista)
                Console.WriteLine(uc);

            Console.ReadKey();
        }
    }
}
```

Programação Web – 2023/2024 – LEI D+CE+PL – ISEC/IPC @JB

isec Engenharia DEIS - Instituto Superior de Engenharia de Coimbra

Strings no C#

- Execute o código e interprete-o

  - A saída deste código utilizando o método *Split* é a seguinte:

23/2024 – LEI DGE+PL – ISEC/IPC @JB

Selecionar C:\Users\Jorge\source\repos\TesteConsoleCore\bin\Debug\net5.0\TesteConsoleCore.exe

```
Unidades Curriculares:

Programação Web
Ética e Deontologia
Projecto ou Estágio
Introdução às Bases de Dados
```

isec Engenharia DEIS - Instituto Superior de Engenharia de Coimbra

Strings no C#

- No exemplo mostrado no código seguinte aborda-se uma situação mais complexa dado que se pretende partir, com pouco código, uma *string* complexa numa lista de palavras
  - Para o efeito, utiliza-se o seguinte *array* de delimitadores:

```
string[] lista = pensamento.Split(new Char[] { ' ', ',', '.', ':','-', '\n', '\t' });
```

  - Este array vai ser como que o filtro que vai permitir "partir" as palavras separadas por cada um dos elementos constantes deste array de delimitadores

Programação Web – 2023/2024 – LEI DAE+PL – ESEC/IPC @JB

isec Engenharia DEIS - Instituto Superior de Engenharia de Coimbra

Strings no C#

```
using System;
namespace TesteComStrings
{
    class Program
    {
        static void Main(string[] args)
        {
            string pensamento = @"Nosso cérebro é o melhor brinquedo já criado:
Nele se encontram todos os segredos, inclusive o da felicidade.
A vida é maravilhosa se você não tem medo dela.";
            Console.WriteLine("Pensamento de Charles Chaplin:");
            Console.WriteLine();
            Console.WriteLine(pensamento);
            Console.WriteLine();
            Console.WriteLine("Palavras:");
            Console.WriteLine();
            string[] lista = pensamento.Split(new Char[] { ' ', ',', '.', ':','-', '\n', '\t' });

            foreach (string palavra in lista)
            {
                if (palavra.Trim() != "")
                    Console.WriteLine(palavra);
            }
            Console.ReadKey();
        }
    }
}
```

Programação Web – 2023/2024 – LEI D+CE+PL – ISEC/IPC @JB

isec Engenharia DEIS - Instituto Superior de Engenharia de Coimbra

Strings no C#

Selecionar C:\Users\Jorge\source\repos\TesteConsoleCore\bin\Debug\net5.0\TesteConsoleCore.exe

```
Pensamento de Charles Chaplin:

Nosso cérebro é o melhor brinquedo já criado:
Nele se encontram todos os segredos, inclusive o da felicidade.
A vida é maravilhosa se você não tem medo dela.

Palavras:

Nosso
cérebro
é
o
melhor
brinquedo
já
criado
Nele
se
encontram
todos
os
segredos
inclusive
o
da
felicidade
A
vida
é
maravilhosa
se
você
não
tem
medo
dela
```


isec Engenharia DEIS - Instituto Superior de Engenharia de Coimbra

Strings no C#

- **Verificar se uma *string* é nula ou vazia**
- A abordagem normal para se fazer esta verificação seria habitualmente a seguinte:

```
string nome = string.Empty;
if (nome == null || nome == string.Empty)
{
Console.WriteLine("variável nome está vazia ou contém null");
}
else
{
Console.WriteLine("nome: {nome}");
}
```

Programação Web - 2023/2024 - LEI D+CE+PL – ISEC/IPC @JB

isec Engenharia DEIS - Instituto Superior de Engenharia de Coimbra

Strings no C#

- Podemos realizar a mesma validação com menos código
  Caso utilizemos o método IsNullOrEmpty da classe String, podemos realizar essa validação de um modo muito simples:

```
string nome = string.Empty;
if (string.IsNullOrEmpty(nome))
{
Console.WriteLine("variável nome está vazia ou contém null");
}
Else
{
Console.WriteLine("nome: {nome}");
}
```

Programação Web – 2023/2024 – LEI D+CE+RC – ISEC/PC @JP

Strings no C#

- **Outras Formas de Formatar *Strings***

- No C# existem outras formas de formatar *strings* para além da interpolação de *strings*

- Para a formatação num valor específico, existem o método estático *Format* da classe *String* e o método *ToString* da classe base *Object*

Programação Web – 2023/2024 – LEI DTSE+PL – ISEC/IPC @JB

isec Engenharia DEIS - Instituto Superior de Engenharia de Coimbra

Strings no C#

- **Utilizar *String.Format para* separar data e hora**

```
using System;
namespace TesteComStrings
{
    class Program
    {
        static void Main(string[] args)
        {
            // Formatação de data e hora
            DateTime agora = DateTime.Now;
            string dtStr = String.Format("Hoje é dia {0:d} e são {0:t}", agora);
            Console.WriteLine(dtStr);
        }
    }
}
```

Programação Web – 2023/2024 – LEI D+CE+PL – ISEC/IPC @JB

isec Engenharia DEIS - Instituto Superior de Engenharia de Coimbra

Strings no C#

- **Utilizar *String.Format para* separar data e hora – V 2**
- **Caso esteja a usar o C# 10, em vez do código mostrado atrás, em *Program.cs* basta colocar <u>só</u> o seguinte:**

```
DateTime agora = DateTime.Now;
Console.WriteLine($"Hoje é dia {agora:d} e são {agora:t}");
```

Programação Web – 2023/2024 – LEI D+CE+IC – ISEC/IPC @JB

isec Engenharia DEIS - Instituto Superior de Engenharia de Coimbra

Strings no C#

- Execute o código e interprete-o
  - A saída deste código utilizando o ***String.Format*** é a seguinte:

2023/2024 – LEI DSGE+PL – ISEC/IPC @JB

isec Engenharia DEIS - Instituto Superior de Engenharia de Coimbra

Strings no C#

- **Utilizar *String.Format para* valores monetários**

```
using System;
namespace TesteComStrings
{
    class Program
    {
        static void Main(string[] args)
        {
            //Formatação de valores monetários
            decimal gasolina = 1.634m;
            string precoGasolina = String.Format("Gasolina: {0:C3}", gasolina);
            Console.WriteLine(precoGasolina);
        }
    }
}
```

Programação Web_ 2023/2024 _ LEI D+CE+PL – ISEC/IPC @JB

isec Engenharia DEIS - Instituto Superior de Engenharia de Coimbra

Strings no C#

- **Utilizar *String.Format baseado numa "máscara"***

```
using System;
namespace TesteComStrings
{
    class Program
    {
        static void Main(string[] args)
        {
            //Formatação usando uma máscara
            string mascara = "0000-000";
            int codigopostal = 3030190;
            Console.WriteLine("O Código Postal da Morada do ISEC é: " +
codigopostal.ToString(mascara));
        }
    }
}
```

Programação Web – 2023/2024 – LEI D+CE+PL – ISEC/IPC @JB

isec Engenharia DEIS - Instituto Superior de Engenharia de Coimbra

Strings no C#

- Execute o código e interprete-o

  - A saída deste código utilizando o ***String.Format*** com máscara é a seguinte:

Web – 2023/2024 – LEI DCE+PL – ISEC/IPC @JB

isec Engenharia DEIS - Instituto Superior de Engenharia de Coimbra

Strings no C#

- **Validação de Dados Utilizando Expressões Regulares**
- As expressões regulares são usadas como ferramentas para executar com pouco código tarefas complexas, como localizar e extrair múltiplas ocorrências de sequências de caracteres específicas dentro de um texto e validar se os dados fornecidos respeitam um formato pré-definido
  - A classe *Regex* do *namespace System.Text.RegularExpressions* é usada para este tipo de validações com base em expressões regulares e contém métodos para executar todo tipo de operações envolvendo expressões regulares
  - O método *IsMatch* é usado para verificar se a *string* passada como parâmetro combina com a expressão regular definida

Programação Web – 2023/2024 – LEIDCE+PL – MEC/pc @JB

Strings no C#

- **Validação de Dados Utilizando Expressões Regulares**
- Uma expressão regular (ER) nem sempre é algo simples de ser construída pois pode ser complexo condensar numa única sequência de metacaracteres várias regras que compõem o padrão
- Para facilitar a construção deste tipo de expressões regulares em casos mais complexos e trabalhosos deve-se seguir uma abordagem de ir construindo a expressão aos "poucos"
  - Significa dividir a ER nas partes que a compõe e documentar cada uma delas no momento da sua criação, após efetuar uma bateria de testes

Programação Web – 2023/2024 – LEI DSGE+PL+ISEC/IPC @JB

isec Engenharia DEIS - Instituto Superior de Engenharia de Coimbra

Strings no C#

- **Validação de Dados Utilizando Expressões Regulares**

```
using System;
using System.Text.RegularExpressions;
namespace TesteComStrings {
    class Program {
        static void Main(string[] args) {
            string formato = @"^\d{4}\-\d{3}$";
            string strCodigoPostal = string.Empty;
            Regex regex = new Regex(formato);
            while (true) {
                Console.Write("Código Postal: ");
                strCodigoPostal = Console.ReadLine().Trim();
                if (strCodigoPostal.ToUpper() == "FIM") break;
                string resultado = regex.IsMatch(strCodigoPostal) ? "válido" : "inválido";
                Console.WriteLine($"{strCodigoPostal} é um formato de Código Postal {resultado} !\n");
            }
        }
    }
}
```

Programação Web – 2023/2024 – LEID+CE+PL – ISEC/IPC @JB

Strings no C#

- Execute o código e interprete-o
  - A saída deste código de validação recorrendo-se a expressões regulares é a seguinte:

LEI D+CE+PL - ISEC/IPC @JB

```
C:\Users\Jorge\source\repos\TesteConsoleCore\bin\Debug\net5.0\TesteConsoleCore.exe

Código Postal: 3030-199
3030-199 é um formato de Código Postal válido !

Código Postal: 300-199
300-199 é um formato de Código Postal inválido !

Código Postal: 2320-122
2320-122 é um formato de Código Postal válido !

Código Postal: 3030-200
3030-200 é um formato de Código Postal válido !

Código Postal:
```

Strings no C#

- **Validação de Dados Utilizando Expressões Regulares**
- Na tabela mostram-se algumas das partes habituais em expressões regulares, nomeadamente as que foram usadas no exemplo anterior

LEI D+CE+PL - ISEC/PC @JB

Parte	Significado
^	Inicio de linha
\d	Algarismos
{n}	n é o número de ocorrências do algarismo
\.	Ponto obrigatório
\-	Hífen obrigatório
$	Fim da Linha

Strings no C#

- **Validação de Dados Utilizando Expressões Regulares**
- Normalmente números como o NIF, o CC, Número de Cartão de Crédito existem dígitos verificadores no final do número que também precisam ser validados para além da validação com a máscara.
- Nestes casos, é preciso combinar expressões regulares com código de validação dos dígitos verificadores.
  - Exemplifica-se a validação do NIF

Programação Web – 2023/2024 – LEIS+CE+MC – ISEC/DEIS @JB

Strings no C#

- **Algoritmo de Validação do NIF (Ver 2013)**
- O algoritmo para validação do NIF é bastante simples: é um número composto por 9 dígitos, sendo o último um digito de controlo, é este digito que iremos calcular para verificar se o NIF está correcto.
- Antes de procedermos ao cálculo do digito de controlo, temos de apurar algumas condições, uma bastante importante tem a ver com o 1º digito, este terá que ser válido, o primeiro dígito do NIF tem a ver com o tipo de entidade que o possui, que pode ser uma das seguintes:

- NIF e Número de Contribuinte – Número de Identificação Fiscal
  - 1 – pessoa singular
  - 2 – pessoa singular

Programação Web – 2023/2024 – LEI DEF+PL – ISEC/IPC @JB

isec Engenharia DEIS - Instituto Superior de Engenharia de Coimbra

Strings no C#

- NIPC – Número de Identificação de Pessoa Colectiva
  - 5 (pessoa colectiva),
  - 6 (pessoa colectiva pública),
  - 8 (empresário em nome individual),
  - 9 (pessoa colectiva irregular ou número provisório).
- Passado com sucesso esta condição, calcula-se o digito de controle
- Para calcular o digito de controlo iremos multiplicar individualmente e por ordem os restantes 8 algarismos, fazendo-o da seguinte forma:
  - O 1.º digito multiplicamos por 9, o 2.º dígito por 8, 3.º dígito por 7, 4.º dígito por 6, 5.º dígito por 5, 6.º dígito por 4, 7.º dígito por 3 e o 8.º dígito por 2.
- Iremos somar todos os resultados da operação anterior.
- Com o resultado da soma anterior iremos dividir por 11 e aproveitar o resto dessa divisão. Se o resto for 0 ou 1, o dígito de controle será 0, caso seja outro algarismo, o dígito de controle será o resultado de 11 – resto.

Programação Web – 2023/2024 – LEI D+CE+PL – ISEC/IPC @JB

Strings no C#

- **Algoritmo de Validação do NIF (Ver 2019)**
  - É constituído por nove dígitos, sendo os oito primeiros sequenciais e o último um dígito de controlo.
  - O NIF pode pertencer a uma de várias gamas de números, definidas pelos dígitos iniciais, com as seguintes interpretações:
  - 1 a 3: Pessoa singular, a gama 3 começou a ser atribuída em junho de 2019;[3]
  - 45: Pessoa singular. Os algarismos iniciais "45" correspondem aos cidadãos não residentes que apenas obtenham em território português rendimentos sujeitos a retenção na fonte a título definitivo;
  - 5: Pessoa colectiva obrigada a registo no Registo Nacional de Pessoas Colectivas;

Programação Web – 2023/2024 – EDS CE+PL – ISEC/IPC @JB

Strings no C#

- **Algoritmo de Validação do NIF (Ver 2019)**
  - 5: Pessoa colectiva obrigada a registo no Registo Nacional de Pessoas Colectivas;[5]
  - 6: Organismo da Administração Pública Central, Regional ou Local;
  - 70, 74 e 75: Herança Indivisa, em que o autor da sucessão não era empresário individual, ou Herança Indivisa em que o cônjuge sobrevivo tem rendimentos comerciais;
  - 71: Não residentes colectivos sujeitos a retenção na fonte a título definitivo;
  - 72: Fundos de investimento;
  - 77: Atribuição Oficiosa de NIF de sujeito passivo (entidades que não requerem NIF junto do RNPC);
  - 78: Atribuição oficiosa a não residentes abrangidos pelo processo VAT REFUND;
  - 79: Regime excepcional - Expo 98;

Programação Web – 2023/2024 – LEI DTCE+PL – ISEC/IPC @JB

Strings no C#

- **Algoritmo de Validação do NIF (Ver 2019)**
  - 8: "empresário em nome individual" (actualmente obsoleto, já não é utilizado nem é válido);
  - 90 e 91: Condomínios, Sociedade Irregulares, Heranças Indivisas cujo autor da sucessão era empresário individual;
  - 98: Não residentes sem estabelecimento estável;
  - 99: Sociedades civis sem personalidade jurídica.
  - O nono e último dígito é o dígito de controlo. É calculado utilizando o algoritmo módulo 11.
  - Obter dígito de controlo
  - editar
  - O NIF tem 9 dígitos, sendo o último o digito de controlo. Para ser calculado o digito de controlo:

Programação Web – 2023/2024 – LEI D+CE+RL – ISEC/IPC @JB

isec Engenharia DEIS - Instituto Superior de Engenharia de Coimbra

Strings no C#

- **Algoritmo de Validação do NIF (Ver 2019)**
  - Multiplique o 8.º dígito por 2, o 7.º dígito por 3, o 6.º dígito por 4, o 5.º dígito por 5, o 4.º dígito por 6, o 3.º dígito por 7, o 2.º dígito por 8 e o 1.º dígito por 9;
  - Some os resultados;
  - Calcule o resto da divisão do número por 11;
  - Se o resto for 0 (zero) ou 1 (um) o dígito de controlo será 0 (zero);
  - Se for outro qualquer algarismo X, o dígito de controlo será o resultado da subtracção 11 - X.

- Ver:
- https://pt.wikipedia.org/wiki/N%C3%BAmero_de_identifica%C3%A7%C3%A3o_fiscal

Programação Web – 2023/2024 – LEID+CE+PL – ISEC/IPC @JB

Strings no C# - Validar NIF V2019

- **Validação do NIF com C# - V 2019**

```
public static bool Nif(string nifNumber)
{
    int tamanhoNumero = 9; // Tamanho do número NIF

    string filteredNumber = Regex.Match(nifNumber, @"[0-9]+").Value; // extrair Número

    if (filteredNumber.Length != tamanhoNumero || int.Parse(filteredNumber[0].ToString()) == 0) { return false; } // Verificar Tamanho, e zero no inicio

    int calculoCheckSum = 0;
    // Calcular check sum
    for (int i = 0; i < tamanhoNumero - 1; i++)
    {
        calculoCheckSum += (int.Parse(filteredNumber[i].ToString()))*(tamanhoNumero - i);
    }

    int digitoVerificacao = 11-(calculoCheckSum % 11);

    if (digitoVerificacao > 9) { digitoVerificacao = 0; }
    // retornar validação
    return digitoVerificacao == int.Parse(filteredNumber[tamanhoNumero - 1].ToString());
}
```

Programação Web – 2023/2024 – LEID+CE+PL – ISEC/IPC @JB

isec Engenharia DEIS - Instituto Superior de Engenharia de Coimbra

Strings no C# - Validar NIF V2013

- **Validação do NIF com Expressões Regulares + Checkdigit V 2013**

```
using System;
using System.Text.RegularExpressions;
namespace TesteComStrings
{
   class Program    {
      public static bool IsValidContrib(string Contrib)       {
         string[] s = new string[9];
         string C = null;
         int i = 0;
         long checkDigit = 0;

         s[0] = Convert.ToString(Contrib[0]);
         s[1] = Convert.ToString(Contrib[1]);
         s[2] = Convert.ToString(Contrib[2]);
         s[3] = Convert.ToString(Contrib[3]);
         s[4] = Convert.ToString(Contrib[4]);
         s[5] = Convert.ToString(Contrib[5]);
```

Programação Web – 2023/2024 – LEI D+CE+PL – ISEC/IPC @JB

isec Engenharia DEIS - Instituto Superior de Engenharia de Coimbra

Strings no C# - Validar NIF V2013

```
    s[6] = Convert.ToString(Contrib[6]);
    s[7] = Convert.ToString(Contrib[7]);
    s[8] = Convert.ToString(Contrib[8]);
    C = s[0];
  if (s[0] == "1" || s[0] == "2" || s[0] == "5" || s[0] == "6" || s[0] == "9")
  {
      checkDigit = Convert.ToInt32(C) * 9;

      for (i = 2; i <= 8; i++)
      {
          checkDigit = checkDigit + (Convert.ToInt32(s[i - 1]) * (10 - i));
      }
      checkDigit = 11 - (checkDigit % 11);
      if ((checkDigit >= 10))
          checkDigit = 0;
      if ((checkDigit == Convert.ToInt32(s[8])))
          return true;
  }
  return false;
}
```

Programação Web – 2023/2024 – LEI D+CE+PL – ISEC/IPC @JB

isec Engenharia DEIS - Instituto Superior de Engenharia de Coimbra

Strings no C# - Validar NIF V2013

```
        static void Main(string[] args)
        {
            string padrao = @"^\d{9}$";
            string nif = String.Empty;
            Regex regex = new Regex(padrao);
            while (true)
            {
                Console.Write("NIF: ");
                nif = Console.ReadLine().Trim();
                if (nif.ToUpper() == "FIM") break;
                string resultado = (regex.IsMatch(nif) && IsValidContrib(nif)) ? "válido" : "inválido";
                Console.WriteLine($"{nif} é um Número de Contribuinte {resultado}!\n");
            }
        }
    }
}
```

Programação Web – 2023/2024 – LEI D+CE+PL – ISEC/IPC @JB

isec Engenharia DEIS - Instituto Superior de Engenharia de Coimbra

Strings no C#

- Execute o código e interprete-o
  - A saída deste código de validação do NIF recorrendo-se a expressões regulares e ao Checkdigit é a seguinte:

LEI DAE+PL - ISEC/IPC @JB

Selecionar C:\Users\Jorge\source\repos\TesteConsoleCore\bin\Debug\net5.0\TesteConsoleCore.exe

```
NIF: 600027350
600027350 é um Número de Contribuinte válido!

NIF: 344566784
344566784 é um Número de Contribuinte inválido!

NIF: 60002735
60002735 é um Número de Contribuinte inválido!

NIF:
```

isec Engenharia DEIS - Instituto Superior de Engenharia de Coimbra

Strings no C#

- **Validação do Cartão de Cidadão:**
  - https://www.autenticacao.gov.pt/documents/20126/0/Valida%C3%A7%C3%A3o+de+N%C3%BAmero+de+Documento+do+Cart%C3%A3o+de+Cidad%C3%A3o+%281%29.pdf/7d5745ba-2bcc-e861-3954-bafe9f7591a0?version=1.0&t=1658411665319&download=true
- **Validação do NIF (Ver 2019):**
  - https://pt.wikipedia.org/wiki/N%C3%BAmero_de_identifica%C3%A7%C3%A3o_fiscal
- **Validação de Cartões de Crédito:**
  - https://cleilsontechinfo.netlify.app/jekyll/update/2019/12/08/um-guia-completo-para-validar-e-formatar-cartoes-de-credito.html

Programação Web – 2023/2024 – LEI D+CL, PL – ISEC/IPC @JB

Strings no C#

- **Obter uma *String* aleatoriamente de uma lista de *strings***

```
using System;
namespace TesteComStrings
{
    class Program
    {
        static void Main(string[] args)
        {
            string[] alunos = { "João Marques", "Patrícia Ferreira", "Ana Maria Vicente","Carlos Renato", "Vera Santos",
"Paulo Luz", "Jorge Miguel","Mafalda Bernardo", "Diogo Costa", "Julia Pires"};
            Random rand = new Random();
            for (int i = 0; i < alunos.Length; i++)
            {
                int indice = rand.Next(alunos.Length);
                Console.WriteLine($"Aluno: {alunos[indice]}");
                Console.ReadKey();
            }
        }
    }
}
```

Programação Web - 2024/2025 - LEI D+CE+PL – ISEC/IPC @JB

isec Engenharia DEIS - Instituto Superior de Engenharia de Coimbra

Strings no C#

- **UTILIZANDO COMENTÁRIOS ESPECIAIS NO CÓDIGO**
- O *Visual Studio* reconhece um tipo especial de comentário que utiliza *tokens*, como TODO , HACK ou outros *tokens* criados pelo próprio programador.
- Esse tipo de comentário é usado como atalho para sinalizar que um trecho de código não está finalizado, possibilitando ao programador retornar rapidamente a esse ponto do código-fonte e dar seguimento ao trabalho.
- Para sinalizar que algo precisa ser continuado depois, basta adicionar uma linha de comentário contendo o *token* TODO

Programação Web – 2023/2024 – LEI D+CE PL – ISEC/IPC @JB

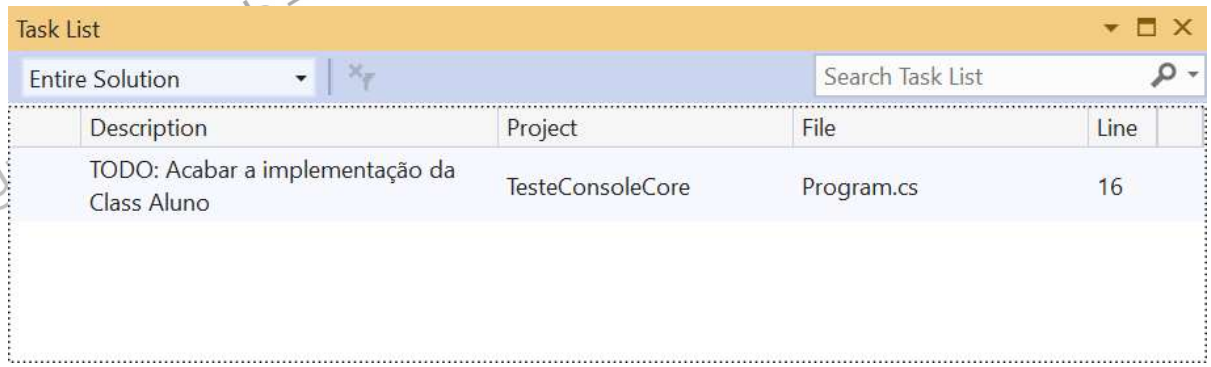
isec Engenharia DEIS - Instituto Superior de Engenharia de Coimbra

Strings no C#

- **EXEMPLO UTILIZANDO O TODO**

  //TODO: Acabar a implementação da Class Aluno

- Pode-se voltar a esta zona do código utilizando a opção Lista de Tarefas (Task List) do Visual Studio do menu View e selecionar de seguida a opção Task List.
- Veja que o comentário aparece na lista:



Prog – 2023/2024 – LEI-CE-L – ISEC/IPC @JB

isec Engenharia DEIS - Instituto Superior de Engenharia de Coimbra

Operadores no C#

- **Para além dos Operadores mais comuns, o C# possui alguns operadores que pelas suas particularidades iremos referir mais em pormenor**
  - **Esses operadores são:**
    - Operador condicional nulo ?
    - Operador de coalescência nula ??
    - Operador ternário
    - Operador *is* e *is not*

Programação Web – 2023/2024 – LEI D+CE, PL – ISEC/IPC G.B

isec Engenharia DEIS - Instituto Superior de Engenharia de Coimbra

Operadores no C#

- **Operador condicional nulo - *null-conditional operator ?***
  - A necessidade de escrever código para testar explicitamente a condição nula antes de poder usar um objeto ou suas propriedades para evitar que uma exceção fosse disparada é maçadora e isto também faz com que o código possa ter várias estruturas condicionais
  - Com a introdução do operador condicional nulo, *null-conditional operator*, o ? (ponto de interrogação) utilizado como nos tipos anuláveis (*nullabe types*), facilitou bastante a escrita de código
    - **Basta colocar o sinal ? depois da instância e antes de chamar a propriedade**

Programação Web – 2023/2024 – LEI & CEPL – ISEC/IPC GJB

isec Engenharia DEIS - Instituto Superior de Engenharia de Coimbra

Operadores no C#

Assim, em vez de

```
if (aluno != null && aluno.Endereco != null)
{
  Console.WriteLine($"{aluno.Nome}\n{aluno.Curso} - aluno.Idade}\n{aluno.Endereco.Rua} -
  {aluno.Endereco.CodigoPostal} - {aluno.Endereco.Pais}\nMail: {aluno.Mail}");
}
```

Pode-se ter

```
Console.WriteLine($"{aluno?.Nome}\n{aluno?.Curso} - {aluno?.Idade}\n{aluno?.Endereco?.Rua} -
{aluno?.Endereco?.CodigoPostal} - {aluno?.Endereco?.Pais}\nMail: {aluno?.Mail}");
```

Sem haver o risco de ocorrer alguma excepção!

Programação Web – 2023/2024 – LEED+CE+PL – ISEC/IPC @JB

isec Engenharia DEIS - Instituto Superior de Engenharia de Coimbra

Operadores no C#

- **Operador de Coalescência Nula ??**
  - O operador de coalescência nula ?? Permite-nos verificar se há um valor nulo numa variável, campo ou propriedade e caso exista atribuir um valor *default*
  - Este operador retornará o operando esquerdo se o operando não for nulo; caso contrário retornará o operando direito
    - Isto é feito numa única linha de código.

Programação Web – 2023/2024 – LEI D+CI +PL – ISEC/IPC @JB

isec Engenharia DEIS - Instituto Superior de Engenharia de Coimbra

Operadores no C#

- **Operador de Coalescência Nula ??**
  - Em vez de

```
static void Main(string[] args)
{
    Aluno aluno = null;
    string nome;
    aluno!=null && aluno.Nome!=null)
    {
      nome = aluno.Nome;
    }
    Else
    {
      nome = "O nome do aluno não foi indicado!";
    }
    Console.WriteLine(nome);
}
```

Programação Web – 2023/2024 – LEI D+CE+PL – ISEC/IPC @JB

isec Engenharia DEIS - Instituto Superior de Engenharia de Coimbra

Operadores no C#

- **Operador de Coalescência Nula ??**

  Pode-se fazer

```
Aluno aluno = null;
var nome = aluno?.Nome ?? "O nome do aluno não foi indicado!";
Console.WriteLine(nome);
```

  - Basta colocar o sinal ?? depois da instância e antes de chamar a propriedade
  - Isto é feito numa única linha de código.

Programação Web – 2023/2024 – LEI D+CE+PL – ISEC/IPC @JB

isec Engenharia DEIS - Instituto Superior de Engenharia de Coimbra

Operadores no C#

- **Operador Ternário**

  - O operador condicional ternário ?: retorna um de dois valores dependendo do valor de uma expressão booleana, cuja sintaxe é:

condição ? primeira_expressão : segunda_expressão;

Programação Web – 2023/2024 – LEIDI+CE+PL – ISEC/IPC @JB

isec Engenharia DEIS - Instituto Superior de Engenharia de Coimbra

Operadores no C#

- A condição é avaliada como *true* ou *false*: se for verdadeira a *primeira_expressão* será avaliada e se tornará o resultado; se for falsa a *segunda_expressão* será avaliada e se tornará o resultado

```
using System;
namespace ProdutividadeEmCSharp
{
    class Program
    {
        static void Main(string[] args)
        {
            string estatuto;
            Console.Write("indique a sua idade:");
            int idade = Convert.ToInt32(Console.ReadLine());
            estatuto = (idade > 18) ? "Maior de idade" : "Menor de idade";
            Console.WriteLine(estatuto);
        }
    }
}
```

Programação Web – 2023/2024 – LEI D+CE+PL – ISEC/IPC @JB

isec Engenharia DEIS - Instituto Superior de Engenharia de Coimbra

Operadores no C#

- **Operador Lógico *is not***
- Este operador *is not* do C# 9.0 permite tornar mais legíveis testes que antes eram feitos usando o operador *is*
  - *Assim, em vez de termos:*

```
if (!(valor is null))
{
   //Código a executar
}
if (!(valor is string))
{
    //Código a executar
}
```

Programação Web – 2023/2024 – LEI DTPE+PL – ISEC/IPC @JB

Operadores no C#

- **Operador Lógico *is not***
  - *Podemos ter:*

```
if (valor is not null)
{
   //Código a executar
}
if (valor is not string)
{
    //Código a executar
}
```

Programação Web – 2023/2024 – LEI D+CE+PL – ISEC/IPC @JB

isec Engenharia DEIS - Instituto Superior de Engenharia de Coimbra

Operadores no C#

- **Sobrecarga de Operadores -** *operator override*
  - O C# permite que alguns dos operadores existentes sejam sobrecarregados, alterando seu significado quando aplicados a tipos pré-definidos pelo programador
    - Basta criar métodos estáticos e preceder o operador a sobrecarregar pela palavra-chave ***operator***
  - Para se perceber o modo de proceder analise o exemplo

Programação Web – 2023/2024 – EI D+CE+PL – ISEC/IPC @JB

Operadores no C#

```
using System;
namespace TesteComStrings {
    class Distancia {
        public int Metros { get; set; }
        public Distancia(int metros) {
            Metros = metros;
        }
        public static Distancia operator +(Distancia a, Distancia b) {
            Distancia resultado = new Distancia(0);
            resultado.Metros = a.Metros + b.Metros;
            return resultado;
        }
        public static Distancia operator -(Distancia a, Distancia b) {
            Distancia resultado = new Distancia(0);
            resultado.Metros = a.Metros - b.Metros;
            return resultado;
        }
```

Programação Web – 2023/2024 – LEI D+CE+PL – ISEC/IPC @JB

isec Engenharia DEIS - Instituto Superior de Engenharia de Coimbra

Operadores no C#

```
public static Distancia operator ++(Distancia a) {
    a.Metros++;
    return a;
}
public static Distancia operator --(Distancia a) {
    a.Metros--;
    return a;
}
public static bool operator ==(Distancia a, Distancia b)
    => a.Metros == b.Metros;
public static bool operator !=(Distancia a, Distancia b)
    => a.Metros != b.Metros;
public static bool operator <(Distancia a, Distancia b)
    => a.Metros < b.Metros;
public static bool operator >(Distancia a, Distancia b)
    => a.Metros > b.Metros;
```

Programação Web – 2023/2024 – LEI D+CE+PL – ISEC/IPC @JB

isec Engenharia DEIS - Instituto Superior de Engenharia de Coimbra

Operadores no C#

```
public static bool operator <=(Distancia a, Distancia b)
        => a.Metros <= b.Metros;
    public static bool operator >=(Distancia a, Distancia b)
        => a.Metros >= b.Metros;
}
class Program {
    static void Main(string[] args) {
        Distancia distancia1 = new Distancia(10);
        Distancia distancia2 = new Distancia(15);
        distancia1++;
        distancia2--;
        Distancia distancia3 = distancia1 + distancia2;
        Console.WriteLine($"Distancia 1: {distancia1.Metros}") ;
        Console.WriteLine($"Distancia 2: {distancia2.Metros}");
        Console.WriteLine($"Distancia 3: {distancia3.Metros}");
```

Programação Web – 2023/2024 – LEI D+CE+PL – ISEC/IPC @JB

isec Engenharia DEIS - Instituto Superior de Engenharia de Coimbra

Operadores no C#

```
if (distancia1 == distancia2)
            Console.WriteLine("A distancia 1 é igual à distancia 2!");
        else
            Console.WriteLine("A distancia 1 é diferente da distancia 2!");
            if (distancia1 >= distancia2)
                Console.WriteLine("A distancia 1 é maior ou igual à distancia 2!");
            else
                Console.WriteLine("A distancia 1 é menor do que a distancia 2!");
        }
    }
}
```

Programação Web – 2023/2024 – LEI+D+CE+PL – ISEC/IPC @JB

isec Engenharia DEIS - Instituto Superior de Engenharia de Coimbra

Code Snippets para o C# no VS

- ***Code Snippets*** **para o C# no** ***Visual Studio***
  - Os *Code Snippets* é um conjunto de códigos pré-definidos com a finalidade de ajudar a inserir trechos de códigos já prontos pressionando algumas teclas ou inserir com o rato
    - O *"Code Snippet Inserter"* tem características similares ao *IntelliSense*
  - Há várias maneiras de utilizar *Code Snippets nos* projectos do *Visual Studio*:
    - No editor pressione as teclas de atalho ***Ctrl+K+X***
    - No *Menu*, selecione *Edit, IntelliSense e Insert Snippet*
    - *Através dos atalhos constantes da tabela – escreva o atalho e aparece a opção e então clique na tecla Tab*

Consultar: http://www.linhadecodigo.com.br/artigo/1374/tempo-e-dinheiro-use-code-snippets-com-csharp.aspx#ixzz79siHrGZC

Programação Web – 2023/2024 – LEI+LICE+PL – ISEC/IPC @

isec Engenharia DEIS - Instituto Superior de Engenharia de Coimbra

Code Snippets para o C# no VS

- ***Atalhos Code Snippets*** **para o C# no** ***Visual Studio***

Atalho	Descrição
do	Criar um ciclo
else	Criar um bloco else
for	Criar um ciclo for
foreach	Criar um ciclo
forr	Cria um ciclo for que decrementa a variável do ciclo
if	Cria um bloco if
switch	Cria um bloco switch
while	Cria um ciclo while

Ao digitar estas palavras chave aparece o menu de contexto e para inserir o código respectivo clique duas vezes em Tab

isec Engenharia DEIS - Instituto Superior de Engenharia de Coimbra

Estrutura Switch no C#

- ***ESTRUTURAS SWITCH***
  - A estrutura condicional *switch* até a versão 6 do *C# tinha* algumas limitações e não era tão flexível como as estruturas equivalentes do Visual Basic e do Delphi
  - Com a versão 7 o *switch* passou a suportar *pattern matching* e estas limitações deixaram de existir e o código gerado com a estrutura condicional *switch* ficou mais conciso e legível
    - Estas alterações ao *switch* permitem utilizações menos triviais
    - No exemplo seguinte mostra-se a utilização da cláusula ***when*** no ***switch***

Programação Web – 2023/2024 – LEI D + CE + RL – ISEC/IPC @JB

isec Engenharia DEIS - Instituto Superior de Engenharia de Coimbra

Estrutura Switch no C#

- **Exemplo 1**

```
using System;
namespace TesteComStrings {
    class Program    {
        private static void IdentificaCaracter(char c)
        {
            switch (c)
            {
                case char l when (c >= 'a' && c <= 'z') || (c >='A' && c <= 'Z'):
                    Console.WriteLine($"{c} é um caracter alfabético.");
                break;
                case char n when c >= '0' && c <= '9':
                    Console.WriteLine($"{c} é um caracter numérico.");
                break;
```

Programação Web – 2023/2024 – LEI D+CE+PL – ISEC/IPC @JB

isec Engenharia DEIS - Instituto Superior de Engenharia de Coimbra

Estrutura Switch no C#

```
            case char n when c >= '!' && c <= '/':
                Console.WriteLine($"{c} é um caracter simbolo.");
                break;
            default:
                break;
        }
    }
    static void Main(string[] args)
    {
        IdentificaCaracter('a');
        IdentificaCaracter('Z');
        IdentificaCaracter('9');
        IdentificaCaracter('*');
    }
}
}
```

Programação Web – 2023/2024 – LEI D+CE+PL – ISEC/IPC @JB

isec Engenharia DEIS - Instituto Superior de Engenharia de Coimbra

Estrutura Switch no C#

- Execute o código e interprete-o
  - A saída deste código onde se utiliza when no switch para classificar os caracteres é a seguinte:

2023/2024 – LEI DAGE+PL – ISEC/IPC @JB

Microsoft Visual Studio Debug Console

```
a é um caracter alfabético.
Z é um caracter alfabético.
9 é um caracter numérico.
* é um caracter simbolo.
```

Estrutura Switch no C#

- **Exemplo 2 – Classificar as Fases da Vida**

```
using System;
namespace TesteComStrings
{
    class Program
    {
        private static void ClassificaFaseDaVida(int idade)
        {
            string fase = String.Empty;
            switch (idade)
            {
                case int i when i <= 11:
                    fase = "Infância";
                    break;
```

Programação Web – 2023/2024 – LEI D+CE+PL – ISEC/IPC @JB

isec Engenharia DEIS - Instituto Superior de Engenharia de Coimbra

Estrutura Switch no C#

- **Exemplo 2 – Classificar as Fases da Vida**

```
case int i when i >= 12 && i <= 20:
            fase = "Adolescência";
            break;
        case int i when i >= 21 && i <= 74:
            fase = "Idade Adulta";
            break;
        case int i when i >= 75:
            fase = "Velhice";
            break;
    }
    Console.WriteLine($"{idade} anos = {fase}");
}
```

Programação Web – 2023/2024 – LEI D+CE+PL – ISEC/IPC @JB

isec Engenharia DEIS - Instituto Superior de Engenharia de Coimbra

Estrutura Switch no C#

- **Exemplo 2 – Classificar as Fases da Vida**

```
        static void Main(string[] args)
        {
            ClassificaFaseDaVida(1);
            ClassificaFaseDaVida(18);
            ClassificaFaseDaVida(46);
            ClassificaFaseDaVida(84);
        }
    }
}
```

Programação Web – 2023/2024 – LEI D+CE+PL – ISEC/IPC @JB

isec Engenharia DEIS - Instituto Superior de Engenharia de Coimbra

Estrutura Switch no C#

- Execute o código e interprete-o
  - A saída deste código onde se utiliza *when* no *switch* para classificar as fases da vida é a seguinte:

Programa – LEI D+CE+PL – ISEC/IPC @JB

Microsoft Visual Studio Debug Console

```
1 anos = Infância
18 anos = Adolescência
46 anos = Idade Adulta
84 anos = Velhice
```

Programa

Estrutura Switch no C#

- **Exemplo 3 – Neste 3º exemplo vemos a utilização no *switch* de comparações recorrendo aos sinais de "menor", "maior", "menor ou igual", "maior ou igual", etc o que o aproxima bastante do que tradicionalmente se faz com o *if***

Nota: Tem de se compilar com pelo menos o C# 9 para este exemplo funcionar

```
using System;
namespace TesteComStrings {
   class Program    {
      static void Main(string[] args)       {
         var valor = 3;
         switch (valor)
         {
            case < 0:
               Console.WriteLine($"Valor {valor} é menor que 0!");
               break;
            case 0:
               Console.WriteLine($"Valor {valor} é igual a 0!");
               break;
```

Programação Web – 2023/2024 – LEI D+CE+PL – ISEC/PC @JB

isec Engenharia DEIS - Instituto Superior de Engenharia de Coimbra

Estrutura Switch no C#

```
            case > 0 and <= 10:
                Console.WriteLine($"Valor {valor} é maior que 0 e menor ou igual a 10!");
                break;
            default:
                Console.WriteLine($"Valor {valor} é maior que 10!");
                break;
        }
    }
  }
}
```

Programação Web – 2023/2024 – LEI D+CE+PL – ISEC/IPC@JB

isec Engenharia DEIS - Instituto Superior de Engenharia de Coimbra

Estrutura Switch no C#

- Execute o código e interprete-o
  - A saída deste código onde se utiliza comparações no *switch* para enquadrar um número num intervalo de valores é a seguinte:

2023/2024 – LEI DAE+PL – ISEC/IPC @JB

Programa

isec Engenharia DEIS - Instituto Superior de Engenharia de Coimbra

Instrução Switch no C#

- ***INSTRUÇÕES SWITCH USANDO EXPRESSÕES SWITCH***
  - Instruções *Switch* <u>não é o mesmo</u> do que Expressões *Switch*
  - Caracterizam-se por a *variável aparecer antes da palavra chave switch*
  - Permite uma maior compactação
  - Utiliza-se um operador *Lambda*

Programação Web – 2023/2024 – LEI D+CG PL – ISEC/IPC @ ISEC

isec Engenharia DEIS - Instituto Superior de Engenharia de Coimbra

Instrução Switch no C#

- Alterações na sintaxe desta construção:
  - O posicionamento da variável antes da palavra-chave *switch* ajuda a distinguir visualmente a expressão *switch* da instrução *switch*
  - Os corpos são expressões, não instruções
  - Os elementos ***case*** e **:** são substituídos por **=>** (mais conciso e intuitivo)
  - O caso ***default*** é substituído por um **_**

Programação Web – 2023/2024 – LEI+CE+PL – ISEC/PC @JB

- No exemplo seguinte mostra-se a utilização de instruções ***switch*** em expressões ***switch***

isec Engenharia DEIS - Instituto Superior de Engenharia de Coimbra

Instrução Switch no C#

- **Exemplo 4 – Neste utilizam-se instruções *switch* em expressões switch**
  Nota: Tem de se compilar com pelo menos o C# 9 para este exemplo funcionar

```
using System;
namespace TesteComStrings
{
    public enum Curso
    {
        Informatica,
        Civil,
        Mecanica,
        Electrotecnia
    }
    class Program
    {
        public static string Seleccionar(Curso curso) =>
```

Programação Web – 2023/2024 – LEI D+CE+PL – ISEC/IPC @JP

isec Engenharia DEIS - Instituto Superior de Engenharia de Coimbra

Instrução Switch no C#

```
class Program  {
    public static string Seleccionar(Curso curso) =>
    curso switch
        {
            Curso.Informatica => "Programação de Aplicações Web!",
            Curso.Civil => "Obras de Stª Engrácia!",
            Curso.Mecanica => "Carros? Isso é o quê?",
            Curso.Electrotecnia => "Não lhes peçam para arranjar um televisor!",
            _ => throw new ArgumentException(message: "Opção de Curso desconhecida!", paramName: nameof(curso))
        };
        static void Main(string[] args) {
            Console.WriteLine(Seleccionar(Curso.Informatica));
            Console.WriteLine(Seleccionar(Curso.Civil));
            Console.WriteLine(Seleccionar(Curso.Mecanica));
            Console.WriteLine(Seleccionar(Curso.Electrotecnia));
        }
  }
}
```

ProgramacaoWeb – 2023/2024 – LEI,D+CE+PL – ISEC/IPC @JB

isec Engenharia DEIS - Instituto Superior de Engenharia de Coimbra

Instrução Switch no C#

- Execute o código e interprete-o
  - A saída deste código onde se utiliza instruções *switch* dentro de expressões *switch* é a seguinte:

Prog 23/2024 – LEI DI+E+PL – ISEC/IPC @JB

Microsoft Visual Studio Debug Console

```
Programação e Manutenção de Sistemas Informáticos!
Obras de Stª Engrácia!
Carros? Isso é o quê?
Não lhes peçam para arranjar um televisor!
```

Instrução Switch no C#

- **O exemplo 3 reescrito com recurso a instruções *switch* e expressões *switch* ficaria como mostrado na listagem seguinte**
  Nota: Tem de se compilar com pelo menos o C# 9 para este exemplo funcionar

```
using System;
namespace TesteComStrings {
        class Program {
            static void Main(string[] args) {
            var valor = 3;
            var mensagem = valor switch {
            < 0 => $"Valor {valor} é menor que 0!",
            0 => $"Valor {valor} é igual a 0!",
            > 0 and <= 10 => $"Valor {valor} é maior que 0 e
            menor ou igual a 10!",
            _ => $"Valor {valor} é maior que 10!"
            };
            Console.WriteLine(mensagem);
            }
        }
}
```

Programação Web – 2023/2024 – LEI D+CE+PL – ISEC/IPC @JB

isec Engenharia DEIS - Instituto Superior de Engenharia de Coimbra

Tipos e Membros no C#

- Sintaxe Simplificada em Tipos Anuláveis

  - A declaração formal de um tipo anulável é:

    Nullable<T> variável = null;

  - É recomendável usar a forma curta alternativa, mais legível e simples:

    T? variável = null;

Programação Web – 2023/2024 – LEI+LD+CE – PL – ISEC/IPC @JS

isec Engenharia DEIS - Instituto Superior de Engenharia de Coimbra

Programação Web

Aulas Teóricas – Capítulo 1 – 1.3

1º Semestre - 2023/2024

Departamento de Engenharia Informática e de Sistemas
Instituto Superior de Engenharia de Coimbra/Instituto Politécnico de Coimbra

Programação Web

C# – Conceitos Avançados - 3ª Parte

Departamento de Engenharia Informática e de Sistemas
Instituto Superior de Engenharia de Coimbra/Instituto Politécnico de Coimbra

Collections
Visão Geral

Colecções em C#

- Existem dois tipos disponíveis:
  - ***Non-generic collections*** - Tipicamente desenvolvidas para trabalhar com tipos *System.Object* e portanto, são *containers* fracamente tipados:
    - *`ArrayList`*, *`BitArray`*, *`Hashtable`*, *`Queue`*, *`SortedList`*, `Stack` `HybridDictionary`, `ListDictionary`, `StringCollection`, `BitVector32`
    - Namespaces:
      `System.Collections`,`System.Collections.Specialized`
      `System.Collections.ObjectModel`,
      `System.Collections.Concurrent`

Colecções em C#

- Existem dois tipos disponíveis (cont.):
  - ***Generic collections*** – Mais seguras que as anteriores, sendo necessário especificar o “tipo” do tipo.
    O indicador de qualquer item genérico é o "type parameter" especificado com os <>, como por exemplo, `List<T>`
    - Namespace: `System.Collections.Generic`
- Todas as classes *collections* implementam a interface *IEnumerable*, portanto os valores da colecção podem ser acedidos usando o ***foreach***

isec Engenharia DEIS - Instituto Superior de Engenharia de Coimbra

Colecções em C# - Genéricas

- *Dictionary<TKey,Tvalue>*
- *LinkedList<T>*
- *List<T>*
- *Queue<T>*
- *SortedDictionary<TKey,Tvalue>*
- *SortedSet<T>*
- *Stack<T>*

Collections: Inicialização

- Inicialização com valores específicos:

```
var stringList = new List<string>
{
      "Blabla",
      "BliBli",
};
```

```
var temp = new List<string>();
temp.Add("Blabla");
temp.Add("Blibli");
var stringList = temp;
```

Collections: Inicialização

```
List<int> listaGenerica = new List<int>();
```

```
List<int> listaGenerica = new List<int> { 4,2,5,6,2,1,4,5};
```

Collections: Inicialização

- Inicialização para tipos que permitem vários parâmetros no método:

```
var numberDictionary = new Dictionary<int, string>
{
      { 1, "One" },
      { 2, "Two" },
};
```

```
var temp = new Dictionary<int, string>();
temp.Add(1, "One");
temp.Add(2, "Two");
var numberDictionarynumberDictionary = temp;
```


isec Engenharia DEIS - Instituto Superior de Engenharia de Coimbra

Collections: Inicialização

```
var dic = new Dictionary<string, int>
{
      ["key1"] = 20,
      ["key2"] = 80
};
```

A partir do C#6

```
var dic = new Dictionary<string, int>();
dict["key1"] = 20;
dict["key2"] = 80
```

Collections: Inicialização

```
public class ClassIndice
  {
      public int this[int index] {
          set {
              Console.WriteLine("{0} Atribuido ao indice {1}",
               value, index);
          }
  }
```

```
static void Main(string[] args) {
      var exemplo = new ClassIndice
      {
          [0] = 12,
          [1] = 15
      };
  }
```

```
12 Atribuido ao indice 0
15 Atribuido ao indice 1
```


isec Engenharia DEIS - Instituto Superior de Engenharia de Coimbra

Collections: Inicialização

```
class Aluno  {
   public int BI { get; set;}
   public string Nome {
                    get; set;}
}
```

```
Aluno aluno= new Aluno {
        BI = 10,
       Nome = "Maria"
};
Console.WriteLine($"Aluno:{aluno.BI}-{aluno.Nome}");
```

```
List<Aluno> listaGenAlunos = new List<Aluno> {
    new Aluno { BI=102212232, Nome="Maria" },
    new Aluno { BI=21222334, Nome="Jose" },
    new Aluno { BI=102324532, Nome="Nuno" },
};
```

Inicialização de class com *Collection Initializers*

- Para que uma classe suporte a inicialização do estilo de uma collection, esta deve implementar a interface **`IEnumerable`** e ter, pelo menos, um **método `Add`**

```
var col = new MinhaColecao
            {
                  "Programacao",
                  { "PW", 5 },
                  "Web"
             };

foreach (var mCol in col)
   Console.WriteLine("{0}", mCol);
```


isec Engenharia DEIS - Instituto Superior de Engenharia de Coimbra

Inicialização de class com *Collection Initializers*

```
class MinhaColecao : IEnumerable {
        private IList list = new ArrayList();
        public void Add(string item) {
            list.Add(item);
        }
        public void Add(string item, int quant) {
            for (int i = 0; i < quant; i++)
                list.Add(item);
        }
        public IEnumerator GetEnumerator() {
            return list.GetEnumerator();
        } }
```

```
var col = new MinhaColecao
            {
                "Programacao",
                { "PW", 5 },
                "Web"
            };

foreach (var mCol in col)
   Console.WriteLine("{0}", mCol);
```

```
Programacao
PW
PW
PW
PW
PW
Web
```


isec Engenharia DEIS - Instituto Superior de Engenharia de Coimbra

Collection Initializers

- ***PropList*** é uma propriedade do tipo collection

```
class MinhaClass
{
    public IList<string> PropList { get; set; }
}
```

```
MinhaClass col = new MinhaClass
 {
     PropList = new List<string> { "PW", "C#" }
 };

 foreach (var mCol in col.PropList)
     Console.WriteLine("{0}", mCol);
```

isec Engenharia DEIS - Instituto Superior de Engenharia de Coimbra

Inicialização de class com *Collection Initializers*

```
var col = new MinhaColecao
   {
       "Programacao",
        "Web"
   };
foreach (var mCol in col)
   Console.WriteLine("{0}", mCol);
```

```
class MinhaColecao : IEnumerable {
        private IList list = new ArrayList();
        public void Add(string item) {
            list.Add(item);
        }
        public void Add(string item, int quant) {
            for (int i = 0; i < quant; i++)
                list.Add(item);
        }
        public IEnumerator GetEnumerator() {
            return list.GetEnumerator();
        } }
```

```
var col = new MinhaColecao {
          list=new ArrayList {
             "Programacao",
             "Web"
          }
   };
foreach (var mCol in col.list)
   Console.WriteLine("{0}", mCol);
```

```
class MinhaColecao  {
  public IList list = new ArrayList();
}
```


isec Engenharia DEIS - Instituto Superior de Engenharia de Coimbra

Collection Initializers

```
class MinhaClass
{
    public IList<string> PropList { get; private set; }
}
```

```
MinhaClass col = new MinhaClass
 {
     PropList = new List<string> { "PW", "C#" }
 };

 foreach (var mCol in col.PropList)
     Console.WriteLine("{0}", mCol);
```


isec Engenharia DEIS - Instituto Superior de Engenharia de Coimbra

Collection Initializers

```
class MinhaClass
{
    public IList<string> PropList { get; private set; }
}
```

```
MinhaClass col = new MinhaClass
 {
     PropList = new List<string> { "PW", "C#" }
 };
```


isec Engenharia DEIS - Instituto Superior de Engenharia de Coimbra

Collection Initializers

```
class MinhaClass
{
    public IList<string> PropList { get; private set; }
}
```

```
MinhaClass col = new MinhaClass
```

```
PropList = new List<string> { "PW", "C#" }
```

IList<string> MinhaClass.PropList { get; private set; }

The property or indexer 'MinhaClass.PropList' cannot be used in this context because the set accessor is inaccessible

```
each (v
Console.WriteLine("{0}", mCol);
```

```
    Console.WriteLine("{0}", mCol);
```

Collection Initializers

```
class MinhaClass
{
    public IList<string> PropList { get; private set; }
}
```

```
MinhaClass col = new MinhaClass
 {
      PropList  =  { "PW", "C#" }
 };
```

isec Engenharia DEIS - Instituto Superior de Engenharia de Coimbra

Collection Initializers

```
class MinhaClass
{
    public IList<string> PropList { get; private set; }
}
```

```
MinhaClass col = new MinhaClass
 {
      PropList  =  { "PW", "C#" }
 };
```

```
Excepção não processada: System.NullReferenceException: A referência de objecto não foi
definida como uma instância de um objecto.
```


isec Engenharia DEIS - Instituto Superior de Engenharia de Coimbra

Collection Initializers

```
class MinhaClass
{
    public IList<string> PropList { get; private set; }
}
public MinhaClass()
{
    PropList = new IList<string>();
}
```

```
MinhaClass col = new MinhaClass
 {
      PropList  =  { "PW", "C#" }
 };
```


isec Engenharia DEIS - Instituto Superior de Engenharia de Coimbra

Alguns Metodos úteis da List<T>

- Add(T)
- AddRange(IEnumerable<T>)
- BinarySearch(T)
- Clear()
- Contains(T)
- CopyTo(T[])
- Equals(Object)
- Exists(Predicate<T>)
- Find(Predicate<T>)
- FindAll(Predicate<T>)
- FindIndex(Predicate<T>)
- FindLast(Predicate<T>)
- FindLastIndex(Predicate<T>)
- ForEach(Action<T>)
- GetEnumerator()
- IndexOf(T)
- Insert(Int32, T)
- LastIndexOf(T)
- Remove(T)
- RemoveAll(Predicate<T>)
- RemoveAt(Int32)
- Reverse()
- Reverse(Int32, Int32)
- Sort()
- Sort(Comparison<T>)
- ToArray()
- ToString()
- ...

Atenção!

```
var a = new List<int>();
var b = a;
a.Add(5);

Console.WriteLine(a.Count);
Console.WriteLine(b.Count);
```

Qual será o resultado?

```
1
1
```


isec Engenharia DEIS - Instituto Superior de Engenharia de Coimbra

Recordar…

```
var a = new List<int>();
var b = a;
a.Add(5);
b.Add(6);
b.Add(2);
Console.WriteLine(a.Count);
Console.WriteLine(b.Count);
```

Qual será o resultado?

3
3

isec Engenharia DEIS - Instituto Superior de Engenharia de Coimbra

Mais info…

- **Collections (C#)**

  https://docs.microsoft.com/en-us/dotnet/csharp/programming-guide/concepts/collections

- **C# Quickstart: Collections**

  https://docs.microsoft.com/en-us/dotnet/csharp/quick-starts/arrays-and-collections

- **Collection<T> Class**

  https://docs.microsoft.com/en-us/dotnet/api/system.collections.objectmodel.collection-1?view=netframework-4.7.2

- **Collections and Data Structures**

  https://docs.microsoft.com/en-us/dotnet/standard/collections/

- **Commonly Used Collection Types**

  https://docs.microsoft.com/en-us/dotnet/standard/collections/commonly-used-collection-types

Delegates
Visão Geral

O que é um Delegate em C# ?

Delegates em C# - Introdução

- Tipos que permitem fazer referência a métodos (eventos/*callbacks*)
- Usado para passar métodos como argumentos a outros métodos
- Um *delegate* é um tipo *referência*
- Pode ser usado para encapsular um método com nome ou anónimo
- Permite **flexibilidade** e **extensibilidade**

Delegates em C#

- **Declaração** de um *delegate* é semelhante à assinatura de um método
- Tem um retorno e qualquer número de parâmetros, de qualquer tipo
- Recorrer à *keyword* ***delegate***

```
public delegate void Exemplo(int valor);
```

- Todos os *delegates* derivam da class `System.Delegate`

isec Engenharia DEIS - Instituto Superior de Engenharia de Coimbra

Delegates em C#

- Quando se atribui um método a um *delegate*, o método deve ter o mesmo tipo de retorno bem como o mesmo tipo de parâmetros;

```
public static void NomeMetodo(string msg) {
    Console.WriteLine(msg);
}
```

- Usar o Delegate:
  - Declarar
  - Instanciar
  - Invocar

```
public delegate void Declarar(string msg);
Declarar xpto=NomeMetodo;
xpto("mensagem");
```


isec Engenharia DEIS - Instituto Superior de Engenharia de Coimbra

Delegates em C#

- *Na prática pode-se considerar um delegate como algo similiar ao conceito de ponteiros para funções usados nas linguagens C e C++*
- Desse modo, um *delegate* em C#, é semelhante a um ponteiro de função *(a vantagem é que é um ponteiro seguro)*.
  - Assim, a utilização de *delegate* permite encapsular a referência a um método dentro de um objeto de delegação.

isec Engenharia DEIS - Instituto Superior de Engenharia de Coimbra

Delegates em C# - Métodos Anóninos

- Métodos Anónimos

  - A ideia por trás dos métodos anónimos é permitir escrever métodos *inline* no código sem declarar um método formal com nome

Delegates em C# - Exemplo Simples

1. Especificação dos métodos

```
...
static public string ConverteMaiusculas(string txt)
{
    return "Maiuscula = “ + txt.ToUpper();
}

static public string ConverteMinusculas(string txt)
{
    return "Minuscula = " + txt.ToLower();
}
...
```

Delegates em C# - Exemplo Simples

2. Declaração, instanciação e invocação do *Delegate*

```
delegate string MetodoDel(string texto);

static void Main(string[] args)
    {
        string txt = "Programacao Web";

        MetodoDel del = ConverteMaiusculas;
        string msg = del(txt);
        Console.WriteLine("{0}", msg);
        del = ConverteMinusculas;
        Console.WriteLine("{0}", del(txt));
    }
```

```
culas(string txt)

per();

culas(string txt)

ower();
```

```
MAIUSCULA = PROGRAMACAO WEB
Minuscula = programacao web
```


isec Engenharia DEIS - Instituto Superior de Engenharia de Coimbra

Delegates em C# - Exemplo Simples

2. Declaração, instanciação e invocação do *Delegate*

```
delegate string MetodoDel(string texto);

static void Main(string[] args)
    {
        string txt = "Programacao

        MetodoDel del = ConverteMa
        string msg = del(txt);
        Console.WriteLine("{0}", m
        del = ConverteMinusculas;
        Console.WriteLine("{0}", d
    }
```

```
las(string txt)

r();

string txt)

);
PROGRAMACAO WEB
Minuscula = programacao web
```

Forma reduzida para invocar o delegate

Poderia ser:
del.Invoke(txt)

isec Engenharia DEIS - Instituto Superior de Engenharia de Coimbra

Delegates em C# - Exemplo Simples

- Tipo delegate como parâmetro

```
static void DoOperacao(MetodoDel del,string txt)
{
    var ret = del(txt);
    Console.WriteLine(string.Format(" > delegate retornou {0}", ret));
}
```

```
static public string ConverteMaiusculas(string txt)
{
    return "Maiuscula = "+txt.ToUpper();
}
static public string ConverteMinusculas(string txt)
{
    return "Minuscula = " + txt.ToLower();
}
```


isec Engenharia DEIS - Instituto Superior de Engenharia de Coimbra

Delegates em C# - Exemplo Simples

- Tipo delegate como parâmetro

```
static void DoOperacao(MetodoDel del,string txt)
{
    var ret = del(txt);
    Console.WriteLine(string.Format(" > delegate retornou {0}", ret));
}
```

```
static public st
{
    return "Maiu
}
static public st
{
    return "Minu
}
```

```
delegate string MetodoDel(string texto);
static void Main(string[] args)
{
    string txt = "Programacao Web";
    DoOperacao(ConverteMaiusculas, "Programacao Web");
    DoOperacao(ConverteMinusculas, "Programacao Web");
}
```

isec Engenharia DEIS - Instituto Superior de Engenharia de Coimbra

Delegates em C# - Exemplo Simples

- Tipo delegate como parâmetro

```
static void DoOperacao(MetodoDel del,string txt)
{
    var ret = del(txt);
                                                                    );
}
```

```
> delegate retornou MAIUSCULA = PROGRAMACAO WEB
> delegate retornou Minuscula = programacao web
```

```
static
{
    return "Maiu
}
static public st
{
    return "Minu
}
```

```
static void Main(string[] args)
{
    string txt = "Programacao Web";
    DoOperacao(ConverteMaiusculas, "Programacao Web");
    DoOperacao(ConverteMinusculas, "Programacao Web");
}
```


isec Engenharia DEIS - Instituto Superior de Engenharia de Coimbra

Delegates em C# - Exemplo Simples

- Instanciar um delegate com um método anónimo

```
delegate string MetodoDel(string texto);
 static void Main(string[] args)
{
    string txt = "Programacao Web";
     MetodoDel del2 = delegate (string texto)
     {
         return texto + "|" + texto;
     };
     Console.WriteLine("{0}", del2(txt));
    }
```

Delegates em C# - Exemplo Simples

- Instanciar um delegate com um método anónimo

```
delegate string MetodoDel(string texto);
static void Main(string[] args)

Programacao Web|Programacao Web

                                                    to)
    {
        return texto + "|" + texto;
    };
    Console.WriteLine("{0}", del2(txt));
}
```


isec Engenharia DEIS - Instituto Superior de Engenharia de Coimbra

Delegates Genéricos

- C# inclui tipos de **bluit-in delegates**!
- Existem 3 tipos de delegate pré-definidos:
  - ***Func***

```
Func<int, int, int> soma = Soma;
```

  - ***Action***

```
Action<int> imp = ImprimeConsola;
```

  - ***Predicate***

```
Predicate<string> estaMai = EstaMaiuscula;
```


isec Engenharia DEIS - Instituto Superior de Engenharia de Coimbra

Delegates Genéricos

- Estes Delegates "**bluit-in**" **existem para facilitar a vida ao programador**!
- Poderiam sempre ser implementados a partir de um *Delegate* Genérico
- Na prática correspondem às situações mais vulgares, mais comuns, de utilização de Delegates e então em vez de o programador estar a "criar" um a partir de um Genérico, a ideia é usar um destes que já está, digamos, que "feito"

Delegate Func

- ***Func***
  - É um **tipo genérico de delegate**
  - Tem **zero** ou **mais parâmetros de entrada** e **um parâmetro de saída**
  - O último parâmetro é considerado o parâmetro de saída
  - Um delegate do tipo `Func` pode incluir 0 a 16 parâmetros de entrada de tipos diferentes
  - Deve incluir um parâmetro ***out*** para resultado

Delegate Func

```
public delegate int Operacao(int i, int j);
class Program
{
    static int Soma(int x, int y)
    {
        return x + y;
    }
    static void Main(string[] args)
    {
        Operacao soma = Soma;
        int resultado = soma(10, 10);
        Console.WriteLine(resultado);
    }
}
```

```
class Program
{
    static int Soma(int x, int y)
    {
        return x + y;
    }
    static void Main(string[] args)
    {
        Func<int, int, int> soma = Soma;
        int resultado = soma(10, 10);
        Console.WriteLine(resultado);
    }
}
```

isec Engenharia DEIS - Instituto Superior de Engenharia de Coimbra

Delegate Func

- ***Func*** com método **Anónimo**

```
Func<int> geraAleatorio = delegate()
    {
        Random rnd = new Random();
        return rnd.Next(1, 100);
    };
```

Delegate Action

- ***Action***
  - Do mesmo género que o delegate `Func`, mas o `Action` não devolve resultado
  - Em termos conceptuais, pode pensar nos "*retornos*" `void` de funções (atenção é um Delegate não uma função)

```
Action<int> imp = ImprimeConsola;
```

```
Action<int> imp =  new Action<int>(ImprimeConsola);
```

isec Engenharia DEIS - Instituto Superior de Engenharia de Coimbra

Delegate Action

```
public delegate void Imprime(int val);

static void ImprimeConsola(int i)
{
    Console.WriteLine(i);
}

static void Main(string[] args)
{
    Imprime imp = ImprimeConsola;
    imp(10);
    imp(20);
    imp(30);

}
```

```
static void ImprimeConsola(int i)
{
    Console.WriteLine(i);
}

static void Main(string[] args)
{
    Action<int> imp = ImprimeConsola;
    imp(10);
    imp(20);
    imp(30);

}
```

isec Engenharia DEIS - Instituto Superior de Engenharia de Coimbra

Delegate Action – Método Anónimo

```
static void ImprimeConsola(int i)
{
    Console.WriteLine(i);
}

static void Main(string[] args)
{
    Action<int> imp = ImprimeConsola;
    imp(10);
    imp(20);
    imp(30);

}
```

```
static void Main(string[] args)
{
    Action<int> imp = delegate (int i)
    {
        Console.WriteLine(i);
    };
    imp(10);
    imp(20);
    imp(30);
}
```

Delegate Predicate

- ***Predicate***

  - Representa um método que <u>contém um conjunto de critérios</u> e verifica se os parâmetros passados cumprem esse critério ou não -> caso “cumpram” retorna “TRUE”; caso contrário retorna “FALSE”

  - Deve ter um parâmetro de entrada e <u>retorna um booleano</u> – verdadeiro ou falso

Delegate Predicate

```
static bool EstaMaiuscula(string str)
{
    return str.Equals(str.ToUpper());
}

static void Main(string[] args)
{
    Predicate<string> estaMai = EstaMaiuscula;

    bool result = estaMai("programacao");
    Console.WriteLine(result);
    result = estaMai("WEB");
    Console.WriteLine(result);
}
```

```
false
true
```


isec Engenharia DEIS - Instituto Superior de Engenharia de Coimbra

Delegate Predicate – Método Anónimo

```
static void Main(string[] args)
{
    Predicate<string> estaMai = delegate (string s)
                        {
                                return s.Equals(s.ToUpper());
                        };

    bool result = estaMai("programacao");
    Console.WriteLine(result);
    result = estaMai("WEB");
    Console.WriteLine(result);
}
```

```
false
true
```


isec Engenharia DEIS - Instituto Superior de Engenharia de Coimbra

Func vs *Action* vs *Predicate*

- Para referênciar um método que tem apenas um parâmetro e retorna **void**, deve-se usar antes o ***delegate genérico*** `Action<T>`

- `Predicate<T>` é também uma forma de ***Func*** mas retorna sempre um **bool**
  - Forma de especificar um determinado critério;
  - Comporta-se do mesmo modo que **`Func<T, bool>`**

Delegates Encadeados

- *Delegates* ***Multicast*** permitem <u>guardar a referência para mais de um método</u>
- Para adicionar uma referência, pode-se usar o operador +=
- Os métodos são executados pela ordem que foram adicionados
- Faz sentido em métodos void

Delegates Encadeados: Exemplo

- Considere as classes:

```
public class Bebe
{

    public string Nome { get; set; }
    public Bebe(string n)
    {
        Nome = n;
    }

    public static Bebe Nasce(string nome)
    {
        return new Bebe(nome);
    }

    public void GravaInfo()  {}
}
```

```
public class BebeExames
 {
     public void Peso(Bebe bebe)
     {
         Console.WriteLine("Pesar Bébé!");
     }
     public void Audicao(Bebe bebe)
     {
         Console.WriteLine("Efetuar exame de audiçao");
     }
     public void FrequenciaCardiaca(Bebe bebe)
     {
         Console.WriteLine("Obter frequência cardiaca");
     }
 }
```

Delegates Encadeados: Exemplo

```
public class ProcessoDoBebe
{
    public void Nascimento(string nome)
    {
        var bebe = Bebe.Nasce(nome);
        var exames = new BebeExames();
        exames.Peso(bebe);
        exames.FrequenciaCardiaca(bebe);
        exames.Audicao(bebe);
        bebe.GravaInfo();
     }
}
```

```
ProcessoDoBebe processo = new ProcessoDoBebe();
processo.Nascimento("Nuno");
```

Pesar Bébé
Obter frenquencia cardíaca
Efetuar exame de audicao

Imaginemos que é efetuado o release, e queremos adicionar novo exame?

Delegates Encadeados: Exemplo

```
public class ProcessoDoBebe
{
    public void Nascimento(string nome)
    {
        var bebe = Bebe.Nasce(nome);
        var exames = new BebeExames();
        exames.Peso(bebe);
        exames.FrequenciaCardiaca(bebe);
        exames.Audicao(bebe);
        bebe.GravaInfo();
     }

public delegate void BebeExamesDel(Bebe bebe);
public void Nascimento(string nome, BebeExamesDel exame)
{
    var bebe = Bebe.Nasce(nome);
    exame(bebe);
    bebe.GravaInfo();
}
}
```

```
var processo = new ProcessoDoBebe();
var bebeExames = new BebeExames();
ProcessoDoBebe.BebeExamesDel exames = bebeExames.Peso;
exames += bebeExames.Audicao;
exames += bebeExames.FrequenciaCardiaca;
exames += bebeExames.Peso;

processo.Nascimento("Nuno", exames);
```

Pesar Bébé
Obter frenquencia cardíaca
Efetuar exame de audicao
Pesar Bébé

isec Engenharia DEIS - Instituto Superior de Engenharia de Coimbra

Delegates Encadeados: Exemplo

E adicionar novo exame inexistente na classe BebeExames?

```
static  void Main()
 {
     var processo = new ProcessoDoBebe();
     var bebeExames = new BebeExames();
     ProcessoDoBebe.BebeExamesDel exames = bebeExames.Peso;
     exames += bebeExames.Audicao;
     exames += bebeExames.FrequenciaCardiaca;
     exames += bebeExames.Peso;
     exames += bebeExames.Ictericia;

     processo.Nascimento("Nuno", exames);
}
```

```
static void Ictericia(Bebe bebe)
{
    Console.WriteLine("Verificar Ictericia!");
}
```


isec Engenharia DEIS - Instituto Superior de Engenharia de Coimbra

Delegates Encadeados: Exemplo

```
public class ProcessoDoBebe
{
public delegate void BebeExamesDel(Bebe bebe);
public void Nascimento(string nome, Action<Bebe> exame)
{
    var bebe = Bebe.Nasce(nome);
    exame(bebe);
    bebe.GravaInfo();
}
}
```

```
var processo = new ProcessoDoBebe();
var bebeExames = new BebeExames();
ProcessoDoBebe.BebeExamesDel exames = bebeExames.Peso;
Action<Bebe> exames = bebeExames.Peso;
exames += bebeExames.Audicao;
exames += bebeExames.FrequenciaCardiaca;
exames += bebeExames.Peso;

processo.Nascimento("Nuno", exames);
```

Recorrendo ao Action

Pesar Bébé
Obter frenquencia cardíaca
Efetuar exame de audicao
Pesar Bébé

isec Engenharia DEIS - Instituto Superior de Engenharia de Coimbra

Delegates em C# - Exemplo com Lambda

- Instanciar um delegate com uma **expressão lambda**

```
delegate string MetodoDel(string texto);
 static void Main(string[] args)
 {
     string txt = "Programacao Web";

     MetodoDel del3 = texto=>texto.ToUpper();

      Console.WriteLine("{0}", del3(txt));
    }
```

```
PROGRAMACAO WEB
```


isec Engenharia DEIS - Instituto Superior de Engenharia de Coimbra

Lambda
O que é uma Expressão Lambda ?

Operador Lambda =>

- ***OPERADOR LAMBDA* =>**
  - Este operador pode ser usado de duas maneiras no C#:
    1. Como *operador lambda* numa *expressão lambda*: separa as variáveis de entrada do lado esquerdo do corpo lambda do lado direito
    2. Numa definição de corpo da expressão, em que separa um nome de membro da implementação do membro.

- O operador => tem a mesma precedência que o operador de atribuição = e é associativo-à-direita

isec Engenharia DEIS - Instituto Superior de Engenharia de Coimbra

O que é uma Expressão Lambda?

- As expressões *lambda* são expressões embutidas semelhantes aos métodos anónimos, mas mais flexíveis;

Permitem escrever menos código e atingir o mesmo objetivo

- Método anónimo
  - Sem nome
  - Sem modificador de acesso (como público ou privado)
  - Não retorna valor
- Usado como um atalho para inicialização de um *delegate*

isec Engenharia DEIS - Instituto Superior de Engenharia de Coimbra

Expressão Lambda

- Sintaxe:

```
argumentos => expressão
```

- Exemplo:

```
int Quadrado(int numero)
{
    return numero * numero;
}
```

```
int Quadrado(int numero)=> numero * numero;
```

isec Engenharia DEIS - Instituto Superior de Engenharia de Coimbra

Lambda

- Com argumentos

```
x => ...
```

- Sem Argumentos

```
() => ...
```

- Com vários argumentos

```
(x,y) => ...
```

Lambda: Exemplo

```
public delegate int OperacaoComInteiro(int input);

static void Main(string[] args)
{

    OperacaoComInteiro dobro = x => x * 2;
    OperacaoComInteiro quadrado = x => x*x;

    Console.WriteLine("  ->{0}", dobro(5));
    Console.WriteLine("  ->{0}", quadrado(5));
}
```

```
->10
->25
```


isec Engenharia DEIS - Instituto Superior de Engenharia de Coimbra

Lambda: Exemplo com um Argumento

```
public delegate int DobroInteiro(int input);
DobroInteiro dobro = x => x * 2;
```



```
public delegate int DobroInteiro(int input);
DobroInteiro dobro = delegate (int x) {
    return x * 2;
};
```


isec Engenharia DEIS - Instituto Superior de Engenharia de Coimbra

Lambda: Exemplo vários argumentos

```
public delegate int MultiplicaInteiros(int n1, int n2);

MultiplicaInteiros mult = (x,y) => x*y;

Console.WriteLine("->{0}", mult(2,7));
```

Lambda: Exemplo sem argumentos

```
public delegate string EscreveMensagem();

EscreveMensagem msg = () => "Programacao";
EscreveMensagem msg1 = () => "Web";
EscreveMensagem msg2 = () => "C#";

Console.WriteLine("  {0} | {1} | {2} ", msg(), msg1(),msg2());
```

```
Programacao | Web | C#
```


isec Engenharia DEIS - Instituto Superior de Engenharia de Coimbra

Várias instruções numa Exp. Lambda

```
public delegate void MultiplicaInteiros(int n1, int n2);
MultiplicaInteiros mult = (x, y) =>
{
    int result = x * y;
    Console.WriteLine("Resultado Multiplicação = {0}", result);
};
mult(5, 1);
mult(5, 2);
mult(5, 3);
mult(5, 4);
mult(5, 5);
```

```
Resultado Multiplicação = 5
Resultado Multiplicação = 10
Resultado Multiplicação = 15
Resultado Multiplicação = 20
Resultado Multiplicação = 25
```

isec Engenharia DEIS - Instituto Superior de Engenharia de Coimbra

Outros formas de utilização do Lambda

- Passar uma expressão lambda como parâmetro de um método

```
List<int> lista1 = new List<int> { 1, 2, 6, 5, 4, 3, 2, 7, 8, 5, 24, 42 };
List<int> lista2 = new List<int> { 12,33,44,2,5,67,86,4,3,22,56,77 };

List<int> listaResult=lista1.FindAll(x => x >= 24);
```

```
List<int> listaResult=lista1.FindAll()
List<int> listaResult2
(int x in list
Console.WriteLine(
.WriteLine("-----");
```

List<int> List<int>.FindAll(Predicate<int> match)
Retrieves all the elements that match the conditions defined by the specified predicate.
match: *The Predicate<in T> delegate that defines the conditions of the elements to search for.*

isec Engenharia DEIS - Instituto Superior de Engenharia de Coimbra

Outros formas de utilização do Lambda

- Passar uma expressão lambda como parâmetro de um método

```
List<int> lista1 = new List<int> { 1, 2, 6, 5, 4, 3, 2, 7, 8, 5, 24, 42 };
List<int> lista2 = new List<int> { 12,33,44,2,5,67,86,4,3,22,56,77 };

List<int> listaResult=lista1.FindAll(x => x >= 24);

List<int> listaResult2 = lista2.FindAll(x => x >= 24
```

Lambda atua como um *Predicate* que garante que apenas os numeros maiores que 24 são retornados

isec Engenharia DEIS - Instituto Superior de Engenharia de Coimbra

Expressão Lambda Simples - Exemplo

```
static void Main(string[] args)
{
    Func<int, int> dobro= i => i * 2;
    Console.WriteLine(dobro(5));
}
```

10

```
public delegate int CalculaDobro(int n);
static void Main(string[] args)
{
    CalculaDobro dobro = i => i * 2;

    Console.WriteLine(dobro(5));
}
```

isec Engenharia DEIS - Instituto Superior de Engenharia de Coimbra

Expressão Lambda Simples - Exemplo

```
static void Main(string[] args)
{
    Func<int, int> dobro= i => i * 2;
    Func<int, int, int> soma = (i, j) => i + j;

    Console.WriteLine(dobro(5));
    Console.WriteLine(Dobro(5));

    Console.WriteLine(soma(11, 20));
    Console.WriteLine(Soma(11, 20));
}
```

Func

É um tipo genérico de delegate

Existem 3 tipos:

- *Func*
- *Action*
- *Predicate*

isec Engenharia DEIS - Instituto Superior de Engenharia de Coimbra

Expressáo Lambda Simples - Exemplo

```
static void Main(string[] args)
{
    Func<int, int> dobro= i => i * 2;
    Func<int, int, int> soma = (i, j) => i + j;

    Console.WriteLine(dobro(5));
    Console.WriteLine(Dobro(5));

    Console.WriteLine(soma(11, 20));
    Console.WriteLine(Soma(11, 20));
}
```

```
10
10
31
31
```

```
static int Dobro(int i)
{
    return i * 2;
}
static int Soma(int i, int j)
{
    return i + j;
}
```

isec Engenharia DEIS - Instituto Superior de Engenharia de Coimbra

Expressão Lambda Simples - Exemplo

```
static void Main(string[] args)
{
    Func<string, string> convMaiuscula = str => str.ToUpper();

    string[] palavras = { "programacao", "web", "disciplina“ };

    IEnumerable<string> selPal = palavras.Select(convMaiuscula);

    foreach (string palavra in  selPal)
        Console.WriteLine(palavra);
}
```

PROGRAMACAO
WEB
DISCIPLINA

DEIS - Instituto Superior de Engenharia de Coimbra
isec Engenharia

Func vs Action vs Predicate com Lambda

```
Predicate<string> predicado = s => s.StartsWith("a");
Func<string, bool> func = s => s.StartsWith("a");

string str = "programacao web";
Console.WriteLine(predicado(str));
Console.WriteLine(predicado("a"+str));
```

```
false
true
```


isec Engenharia DEIS - Instituto Superior de Engenharia de Coimbra

Expressão Lambda

- Uso do delegado `Func<T,TResult>` com métodos anónimos

```
Func<string, string> converte = delegate (string s)
{
     return s.ToUpper();
};

string str = "programacao web";
Console.WriteLine(converte(str));
```

14
30

isec Engenharia DEIS - Instituto Superior de Engenharia de Coimbra

Expressão Lambda

```
Func<int, string> dobroesomar10 = i => {
    var dobro = 2 * i;
    var soma = 10 + dobro;
    return $">{soma}<";
};

Console.WriteLine("{0}", dobroesomar10(2));
Console.WriteLine("{0}", dobroesomar10(10));
```

```
14
30
```


isec Engenharia DEIS - Instituto Superior de Engenharia de Coimbra

Lambda: Exemplo

- Criação de um array de Alunos

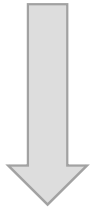
```
Aluno[] alunos = {
        new Aluno() { Numero = 1, Nome= "Jose Antunes", Idade= 18 },
        new Aluno() { Numero = 2, Nome = "Filipa Moreira",  Idade = 21 },
        new Aluno() { Numero = 3, Nome = "Cristina Fria",  Idade = 25 },
        new Aluno() { Numero = 4, Nome = "Dinis Campos" , Idade = 20 },
        new Aluno() { Numero = 5, Nome = "Soaraia Goncalves" , Idade = 31 },
        new Aluno() { Numero = 6, Nome = "Lena Pinheiro",  Idade = 17 },
        new Aluno() { Numero = 7, Nome = "
 };
```

```
public class Aluno
{
    public int Numero { get; set; }
    public string Nome { get; set; }
    public int Idade { get; set; }
}
```

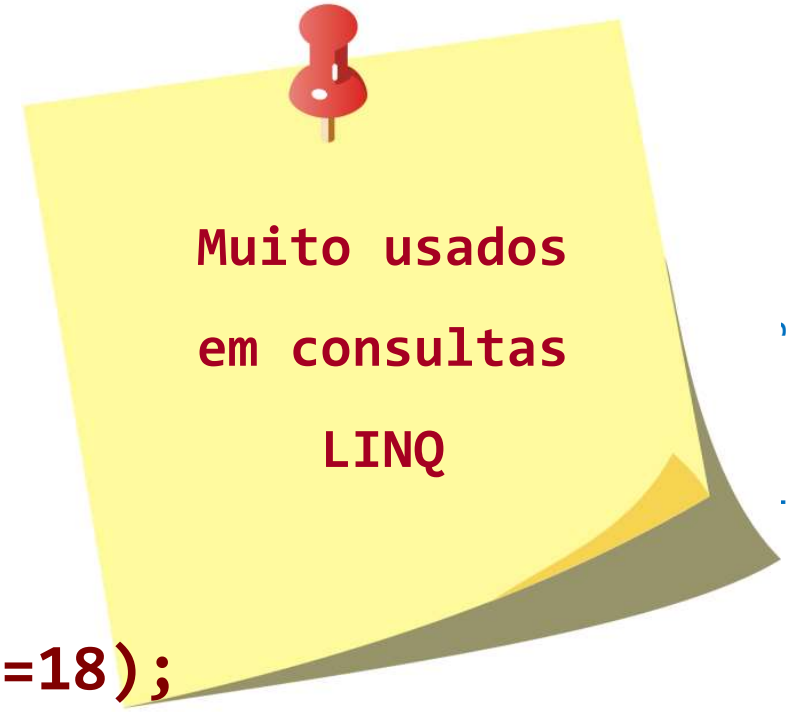
isec Engenharia DEIS - Instituto Superior de Engenharia de Coimbra

Lambda: Exemplo

```
static bool MaioresIdade(Aluno aluno)
{
    return aluno.Idade>=18;
}
```



```
alunos.FindAll(aluno => aluno.Idade>=18);
alunos.FindAll(a => a.Idade>=18);
```



Lambda: Exercício

- Considerando o array de alunos, especifique uma expressão lambda que permita obter a lista de alunos com a letra F no nome:

```
Aluno[] alunos = {
        new Aluno() { Numero = 1, Nome= "Jose Antunes", Idade= 18 },
        new Aluno() { Numero = 2, Nome = "Filipa Moreira",  Idade = 21 },
        new Aluno() { Numero = 3, Nome = "Cristina Fria",  Idade = 25 },
        new Aluno() { Numero = 4, Nome = "Dinis Campos" , Idade = 20 },
        new Aluno() { Numero = 5, Nome = "Soaraia Goncalves" , Idade = 31 },
        new Aluno() { Numero = 6, Nome = "Lena Pinheiro",  Idade = 17 },
        new Aluno() { Numero = 7, Nome = "Filipe Cruz",Idade = 19  },
 };
```

Lambda: Exemplo

- Considerando o array de alunos, especifique uma expressão lambda que permita obter a **lista de alunos com a letra F** no nome:

```
var alunosF = alunos.Where(???);
```

```
var alunosF = alunos.Where(
                (e) => {
                return (e.Nome.Contains('F'));
                } );
```

Lambda: Exemplo

- E se apenas a primeira letra começar por F?

```
var alunosF = alunos.Where(???);
```

```
var alunosF = alunos.Where(
                (e) => {
                return (e.Nome.StartsWith("F"));
                } );
```


isec Engenharia DEIS - Instituto Superior de Engenharia de Coimbra

Lambda: Exemplo

- E apresentar os nomes dos alunos que tenham a vogal a na primeira palavra do nome?

```
var alunosF = alunos.Where(???);
```

```
var alunosF = alunos.Where(
      (e) => {
 return (e.Nome.Split(' ').First().Contains("a"));
      } );
```


isec Engenharia DEIS - Instituto Superior de Engenharia de Coimbra

C# - Extension Methods

Extension Methods

- ***Extension Methods***
  - **Permitem <u>adicionar métodos a uma *class* existente</u> sem alterar o seu código fonte ou criar uma nova *class* que derive dela**
    - *Útil principalmente em situações em que não é possível alterar o código fonte para um dado tipo*
  - Podem ser especificados para tipos do sistema, tipos definidos por terceiros, tipos próprios
  - Deve-se adicionar um método estático a uma classe estática

isec Engenharia DEIS - Instituto Superior de Engenharia de Coimbra

Extension Methods

```
static void Main(string[] args)
{
    string texto = "PROGRAMACAO WEB";
    PWString.AddPW(texto);
}

public static class PWString:string
{
    public string AddPW(string x)
    {
        return x + "+PW";
    }
}
```

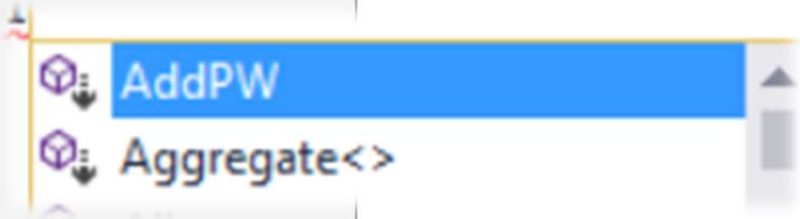

isec Engenharia DEIS - Instituto Superior de Engenharia de Coimbra

Extension Methods

```
public static class PWStringExtensions
{
    public static string AddPW(this String x)
    {
        return x + " +PW";
    }
}
class Program
{
    static void Main(string[] args)
    {
        string texto = "PROGRAMACAO WEB";
        var x = texto.AddPW();
        Console.WriteLine(x);
    }
}
```

Extension Methods

```
public static class PWExtensions
{
    public static string AddPW(this String x)
    {
        return x + " +PW";
    }
}
class Program
{
    static void Main(string[] args)
    {
        string texto = "PROGRAMACAO WEB";
        var x = texto.AddPW();
        Console.WriteLine(x);
    }
}
```



isec Engenharia DEIS - Instituto Superior de Engenharia de Coimbra

Tipo Nullable
Operador Null-Coalescing

Tipo *Nullable*

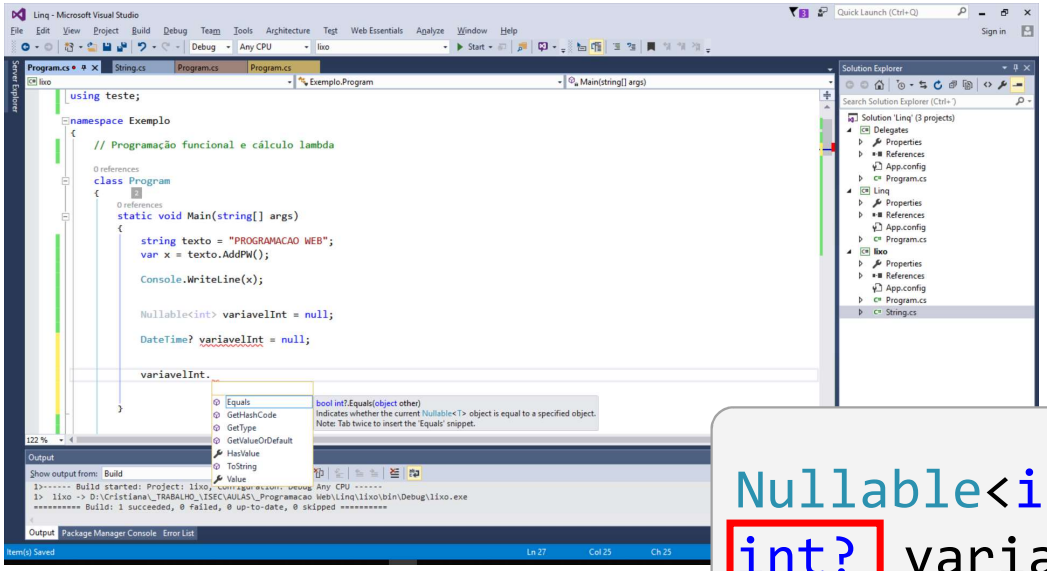
- Tipo ***Nullable – Revisitando estes tipos***
  - Em C# tipos que sejam do tipo valor não podem ter o valor null

Cannot convert null to 'int' because it is a non-nullable value type

```
int variavelInt = null;

bool variavelBool = null;

DateTime variavelDateTime = null;
```


isec Engenharia DEIS - Instituto Superior de Engenharia de Coimbra

Tipo *Nullable*



GetValueOrDefault

Value

HasValue

```
Nullable<int> variavelInt = null;
int? variavelInt = null;

DateTime? variavelDate = null;

variavelDate.
```


isec Engenharia DEIS - Instituto Superior de Engenharia de Coimbra

Tipo *Nullable*

```
int? varInt = null;
int varInt2;

Console.WriteLine(varInt.GetValueOrDefault());
Console.WriteLine(varInt.HasValue);
Console.WriteLine(varInt.Value);
```

C:\WINDOWS\system32\cmd.exe

```
0
False

Excepção não processada: System.InvalidOperationException: O objecto que
 pode ser nulo tem de ter um valor.
   em System.ThrowHelper.ThrowInvalidOperationException(ExceptionResourc
e resource)
   em System.Nullable`1.get_Value()
   em Exemplo.Program.Main(String[] args) em D:\Cristiana\_TRABALHO_\ISE
C\AULAS\_Programacao Web\Linq\lixo\Program.cs:line 20
Press any key to continue . . .
```

isec Engenharia DEIS - Instituto Superior de Engenharia de Coimbra

Tipo *Nullable*

```
int? varInt = null;
int varInt2;

varInt2=varInt;

varInt2=varInt.getValueOrDefault();

varInt = varInt2;

varInt2=0;
varInt = varInt2;
```

Operador *Null-Coalescing*

- Permite especificar um valor por omissão se o operando à esquerda for nulo

```
DateTime? data = null;
DateTime data1;
string str = null;
data1 = data ?? DateTime.Now;
Console.WriteLine("string = " + (str ?? " Sem valor!"));
```


isec Engenharia DEIS - Instituto Superior de Engenharia de Coimbra

Operador *Null-Coalescing*

```
int? varInt=null;
int varInt2;

varInt2 = varInt ?? 10;

int? varInt3 = varInt2;

Console.WriteLine(varInt2+" "+varInt3);
```

Operador *Null-Coalescing*

```
int? varInt = null;
int? varInt2=null;

int varInt3 = varInt2 ?? varInt ?? 10;
```

```
IEnumerable<Aluno> minhaLista = GetListaAlunos();
var al = minhaLista.SingleOrDefault(x => x.Id == 2) ??
                                        new Aluno { Id = 2 };
Console.WriteLine(al);
```

isec Engenharia DEIS - Instituto Superior de Engenharia de Coimbra

Programação Web

Aulas Teóricas – Capítulo 1 – 1.4

1º Semestre - 2023/2024

Departamento de Engenharia Informática e de Sistemas
Instituto Superior de Engenharia de Coimbra/Instituto Politécnico de Coimbra

Programação Web

C# – Conceitos Avançados - 4ª Parte

Departamento de Engenharia Informática e de Sistemas
Instituto Superior de Engenharia de Coimbra/Instituto Politécnico de Coimbra

LINQ
Language Integrated Query

LINQ - Language Integrated Query

- Introduzida com o .Net 3.5 e VS 2008
- Permite efetuar pesquisas em C# ou VB a conjuntos de dados:
  - Objectos em memória – Collections (*LINQ to Objects*)
  - Databases (***LINQ to Entities***) – **Entity Framework**
  - XML (*LINQ to XML*)
  - ADO.NET DataSets (*LINQ to DataSet*)
  - … (com implementação do `IQueryable, IQueryable<T>`)

Programação Web – 2023/2024 – LEID+CE+PL – ISEC/IPC @JB

LINQ *Project*

C# 3.0

VB 9.0

Others…

.NET Language Integrated Query

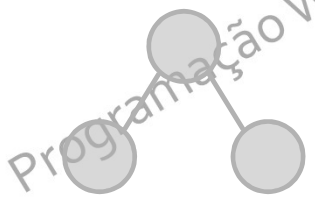
LINQ to Objects

LINQ to DataSets

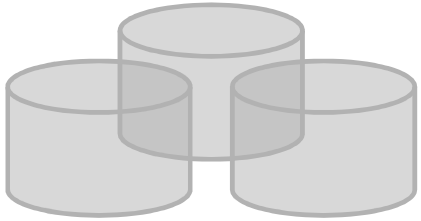
LINQ to SQL

LINQ to Entities

LINQ to XML



Objects



Relational

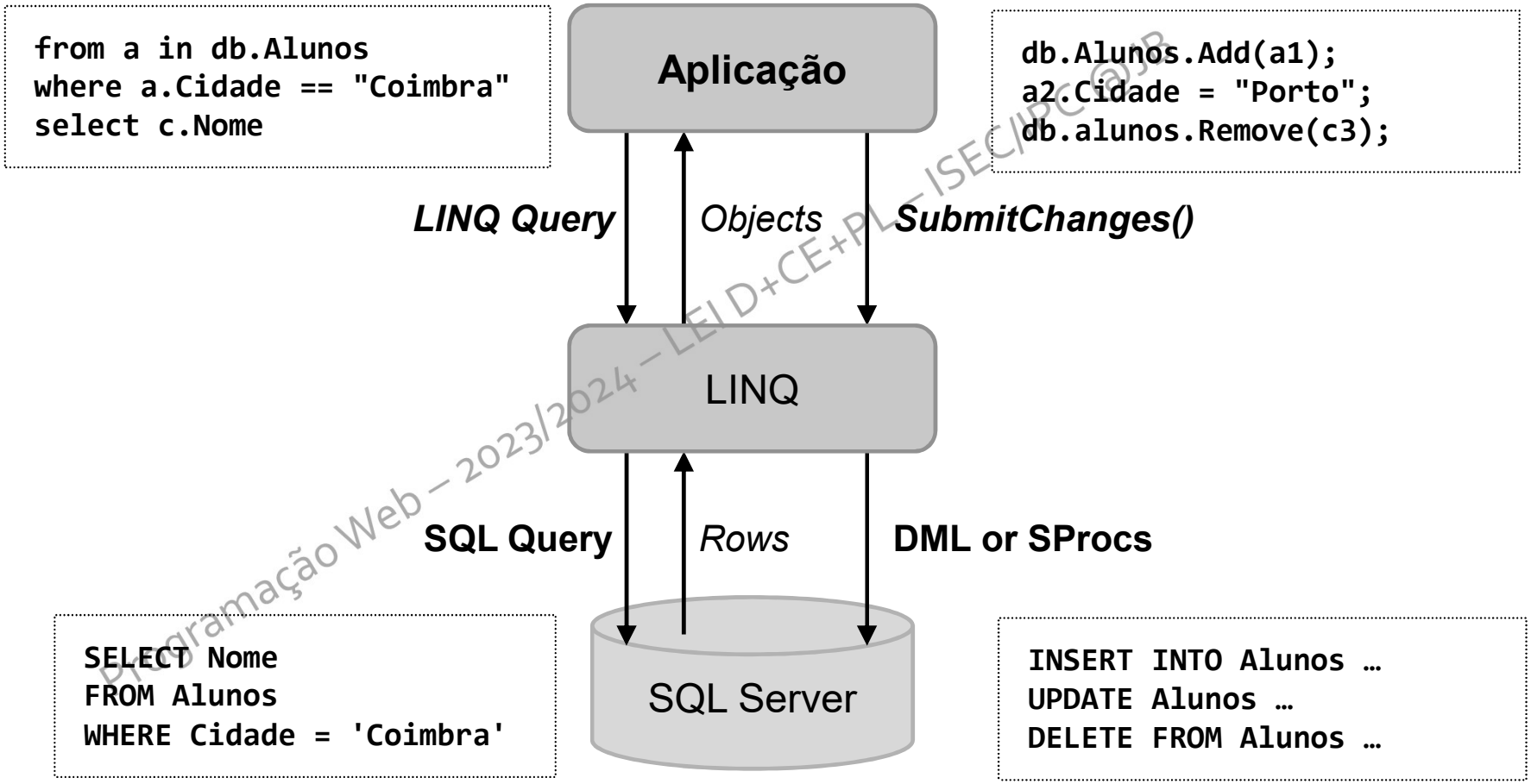
```
<book>
    <title/>
    <author/>
    <year/>
    <price/>
</book>
```

XML

Programação Web – 2023/2024 – LEID+CEPH – ISEC/IPC @JB

isec Engenharia DEIS - Instituto Superior de Engenharia de Coimbra

Arquitectura - *LINQ to Entities*



Programação Web – 2023/2024 – LEI D+CE+PL – ISEC/PC@JR

isec Engenharia DEIS - Instituto Superior de Engenharia de Coimbra

LINQ

- *Queries* LINQ são *strongly typed*
- Permite a simplificação substantial de código
- Namespace
  - `System.Linq`
- LINQ API inclui duas classes principais **`Enumerable`** & **`Queryable`**.
- *LINQ Operators apresentam um fluent interface*

Programação Web – 2023/2024 – LEI D+CE+PL – ISEC/IPC @JB

isec Engenharia DEIS - Instituto Superior de Engenharia de Coimbra

LINQ

- Vantagens
  - Permite identificação de erros em tempo de compilação
  - Reutilização de código
  - Suporta filtros, ordenação, agrupamentos com menos esforço de implementação

Programação Web – 2023/2024 – LEI D+CS+PL – ISEC/IPC @JB

isec Engenharia DEIS - Instituto Superior de Engenharia de Coimbra

LINQ

- Existem 2 tipos de sintaxe *LINQ*:
  - ***a) Query Operators | Query Syntax***
    - Semelhante à linguagem *SQL* (*Structured Query Language*)
    - Inicia com a palavra chave **`from`** para especificar de onde os dados provem e termina com a clausula **`select`** ou **`group`**
    - Introduz uma variável que pode ser utilizada para filtrar, agrupar

```
var alunosExcelentes = from a in db.Alunos
                       where a.Media >= 18
                       orderby a.Nome
                       select a.Nome
```

Programação Web – 2023/2024 – EI D+CE+RL – ISEC/IPC @JB

isec Engenharia DEIS - Instituto Superior de Engenharia de Coimbra

LINQ

- Existem 2 tipos de sintaxe *LINQ* (cont.):
  - ***b) Extension Methods – mais "modernos"***
    - Usa uma *extensão de métodos* e *expressões Lambda*
    - Recorre a extensão de métodos das classes estáticas **Enumerable** ou **Queryable**.
    - **<u>Inclui operadores que não estão disponíveis na *Query Syntax*</u>**

```
var alunosExcelentes = db.Alunos.Where(a => a.Media >= 18)
                                .OrderBy(a => a.Nome)
                                .Select(a => a.Nome);
```

Programação Web – 2022/2024 - LEI DAE+PL – ESEC/IPC @JB

LINQ

- ***Preferencialmente devemos utilizer os Extension Methods – mais "modernos"***
  - Instruções muito mais simples
  - Instruções muito mais compactas
  - **<u>Inclui operadores que não estão disponíveis na *Query Syntax*</u>**
    - *A MS tem somente introduzido novos "operadores" nos métodos de extensão e não tem "actualizado" a Query Syntax*
    - …

Programação Web – 2023/2024 – LEI D+CE+PL – ISEC/IPC @ JB

isec Engenharia DEIS - Instituto Superior de Engenharia de Coimbra

Com *LINQ* vs Sem *LINQ*

```
IList<string>  PWList = new List<string>() {
        "C#",
        "Entity Framework",
        "Aprender C#",
        "ASP.NET Core com C#" ,
        "Linq",
        "Lambda" ,
        "Identity" ,
};
```

```
var PWListCSharp = new List<string>();
foreach (string x in PWList)
{
    if (x.Contains("C#"))
        PWListCSharp.Add(x);
}
```

Sem LINQ

```
var listaCShap = from s in PWList
                 where s.Contains("C#")
                 select s;
```

Query Syntax

```
var listaCShap = PWList.Where(l=>l.Contains("C#"));
```

Extension Methods

Com LINQ

isec Engenharia DEIS - Instituto Superior de Engenharia de Coimbra

LINQ

- Seleccionar todas as keywords que incluem C# ?

```
IList<string>  PWList = new List<string>() {
        "C#",
        "Entity Framework",
        "Aprender C#",
        "ASP.NET Core com C#" ,
        "Linq",
        "Lambda" ,
        "Identity" ,
};
... ??????
foreach (string x in listaCShap)
    Console.WriteLine(x);
```

Extension Methods vs Query Operators

```
IList<string>  PWList = new List<string>() {
        "C#",
        "Entity Framework",
        "Aprender C#",
        "ASP.NET Core com C#" ,
        "Linq"
```

```
var listaCShap = from s in PWList
                 where s.Contains("C#")
                 select s;
```

LINQ Query Syntax

```
foreach (string x in listaCShap)
    Console.WriteLine(x);
```

Extension Methods vs Query Operators

```
IList<string> PWList = new List<string>() {
        "C#",
        "Entity Framework",
        "Aprender C#",
        "ASP.NET Core com C#" ,
        "Linq",
```

```
var listaCShapEMS =  PWList.
                     Where(x=>x.Contains("C#"));
```

LINQ Extension Methods

```
foreach (string x in listaCShap)
    Console.WriteLine(x);
```

isec Engenharia DEIS - Instituto Superior de Engenharia de Coimbra

LINQ Extension Methods

```
var queryAlunos = alunos.
```

- Contains<>
- CopyTo
- Count<>
- DefaultIfEmpty<>
- Distinct<>
- ElementAt<>
- ElementAtOrDefault<>
- Equals
- Except<>

Programação Web – 2023/2024 – LEI D... – ...PC @JB

- *Using System.Linq*
- Recorrem a expressões *Lambda*

isec Engenharia DEIS - Instituto Superior de Engenharia de Coimbra

LINQ Extension Methods

▲ 1 of 4 ▼ (extension) IQueryable<Aluno> IQueryable<Aluno>.Where<Aluno>(**System.Linq.Expressions.Expression<Func<Aluno, bool>> predicate**)
Filters a sequence of values based on a predicate.
predicate: *A function to test each element for a condition.*

Predicate

Expressão Lambda

```
var alunosExcelentes = db.Alunos.Where(a => a.Media >= 18);
```

ramação Web – 2023/2024 – LL – ISEC/IPC @JB

isec Engenharia DEIS - Instituto Superior de Engenharia de Coimbra

Operadores LINQ

Restriction	Where
Projection	Select, SelectMany
Ordering	OrderBy, ThenBy
Grouping	GroupBy
Joins	Join, GroupJoin
Quantifiers	Any, All
Partitioning	Take, Skip, TakeWhile, SkipWhile
Sets	Distinct, Union, Intersect, Except
Elements	First, Last, Single, ElementAt
Aggregation	Count, Sum, Min, Max, Average
Conversion	ToArray, ToList, ToDictionary
Casting	OfType<T>, Cast<T>

...

Programação Web – 2023/2024 – LEI & LCE+PL – ISEC/IPC @JB

isec Engenharia DEIS - Instituto Superior de Engenharia de Coimbra

LINQ x busca *case-insensitive*

```
IList<string>  PWList = new List<string>() {
        “c#",
        "Entity Framework",
        "Aprender C#",
        "ASP.NET Core com C#" ,
        "Linq",
        "Lambda" ,
        "Identity" ,
};
```

No exemplo anterior, a pesquisa de caracteres através do LINQ não tinha em conta se esses caracteres eram maiúsculos ou minúsculos e por isso só eram retornados os caracteres que corresponderiam exactamwente ao padrão indicado, ou seja neste exemplo o “C”!
Neste exemplo, utiliza-se também o **StringComparison.OrdinalIgnoreCase** e todas as ocorrências do caracter “C” serão obtidas quer sejam maiúsculas quer minúsculas!

```
var listaCSharp = from s in PWList
                  where s.Contains("C#“, StringComparison.OrdinalIgnoreCase )
                  select s;
```

```
var listaCSharp = PWList.Where(l=>l.Contains("C#“,StringComparison.OrdinalIgnoreCase));
```

DEIS - Instituto Superior de Engenharia de Coimbra

LINQ x Expressões Regulares

```
string acentuados = @"[^a-zA-Z]";

Regex regex = new Regex(acentuados);

List<string> nomes = new List<string>() { "Cláudio", "Flávia", "Inácio", "Miguel", "Renato", "Ana", "Aurélio", "Francisco",
"Jorge", "Patrícia" };
var resultado = from n in nomes  where regex.IsMatch(n) select n;

Console.WriteLine("Nomes que possuem caracteres acentuados:");
        foreach (var item in resultado)
        {
                Console.WriteLine(item);
        }
```

A utilização de Expressões regulares em conjunto com o LINQ permite a selecção de padrões de caracteres que estejam de acordo com a expressão regular definida para padrão de comparação!

Programaçã

isec Engenharia DEIS - Instituto Superior de Engenharia de Coimbra

LINQ
Restrição Where

LINQ: Restrição

- ***Where()***
  - Permite filtrar uma sequência de valores tendo por base um predicado

C/IPC @JB

```
[NotNull] ([NotNull] this IEnumerable<string> source, [NotNull] Func<string,bool>
predicate):IEnumerable<string>
  Filters a sequence of values based on a predicate.
  predicate: A function to test each element for a condition.
[NotNull] ([NotNull] this IEnumerable<string> source, [NotNull] Func<string,int,bool>
predicate):IEnumerable<string>
```

```
var comLinq=PWList.Where()
```

```
string => {}
(string, int) => {}
delegate (string) {}
delegate (string, int) {}
Create method Predicate(string)
Create method Predicate(string, int)
new Func<string,bool>()
new Func<string,int,bool>()
null
PWListCSharp                    List<string>
PWList                          IList<string>
```

```
(extension) IEnumerable<Aluno> IEnumerable<Aluno>.Where<Aluno>(Func<Aluno, bool> predicate) (+ 1 overload)
Filters a sequence of values based on a predicate.

Exceptions:
  ArgumentNullException
```


isec Engenharia DEIS - Instituto Superior de Engenharia de Coimbra

LINQ: Exemplo

```
public class Aluno
{
    public int Numero { get; set; }
    public string Nome { get; set; }
    public int Idade { get; set; }
}
```

Considere esta *class* e o *array de strings* e obtenha o nome dos alunos com 18<=idade<=23 usando *QS* e *EM*

DAWE+PL – ISEC/MIEC @JB

```
Aluno[] alunos = {
        new Aluno() { Numero = 1, Nome= "Jose Antunes", Idade= 18 },
        new Aluno() { Numero = 2, Nome = "Filipa Moreira",  Idade = 21 },
        new Aluno() { Numero = 3, Nome = "Cristina Frias",  Idade = 25 },
        new Aluno() { Numero = 4, Nome = "Dinis Campos" , Idade = 20 },
        new Aluno() { Numero = 5, Nome = "Soraia Goncalves" , Idade = 31 },
        new Aluno() { Numero = 6, Nome = "Lena Pinheiro",  Idade = 17 },
        new Aluno() { Numero = 7, Nome = "Filipe Cruz",Idade = 19  },
 };
```

LINQ Query Operators

- Seleccionar todos os alunos com idade entre 18 e 23 com QS

```
Aluno[] alunos = {
        new Aluno() { Numero = 1, Nome= "Jose Antunes", Idade= 18 },
        new Aluno() { Numero = 2, Nome = "Filipa Moreira",  Idade = 21 },
        new Aluno() { Numero = 3, Nome = "Cristina Fria",  Idade = 25 },
        new Aluno() { Numero = 4, Nome = "Dinis Campos" , Idade = 20 },
        new Aluno() { Numero = 5, Nome = "Soaraia Goncalves" , Idade = 31 },
 };
```

```
var alunosIdade18e23 = from a in alunos
                       where a.Idade > 18 &&
                             a.Idade < 23
                       select a;
```

DEIS - Instituto Superior de Engenharia de Coimbra

LINQ Query Operators

- Seleccionar todos os alunos com idade entre 18 e 23 com QS

```
var alunosIdade18e23 = from a in alunos
                       where a.Idade > 18 && a.Idade < 23
                       select a;
```

Será possível especificar a restrição num bloco à parte?

Programação Web – ISEC/IPC @ JP

LINQ Query Operators

- Seleccionar todos os alunos com idade entre 18 e 23 com QS

```
bool idadeEntre18e23(Aluno a) {
            return a.Idade > 18 && a.Idade < 23;
}
```

```
var alunosIdade18e23 = from a in alunos
                       where idadeEntre18e23(a)
                       select a;
```

DEIS - Instituto Superior de Engenharia de Coimbra

LINQ Query Operators

- Seleccionar todos os alunos com idade entre 18 e 23?

```
var alunosIdade18e23 = from a in alunos
                       where a.Idade > 18 && a.Idade < 23
                       select a;
```

```
Func<Aluno, bool> idadeEntre18e23 = delegate (Aluno s) {
                    return s.Idade > 18 && s.Idade< 23;
};

var alunosIdade18e23 = from s in alunos
                       where idadeEntre18e23(s)
                       select s;
```

isec Engenharia DEIS - Instituto Superior de Engenharia de Coimbra

LINQ Extension Methods

- Seleccionar todos os alunos com idade entre 18 e 23 com EM?

```
Aluno[] alunos = {
        new Aluno() { Numero = 1, Nome= "Jose Antunes", Idade= 18 },
        new Aluno() { Numero = 2, Nome = "Filipa Moreira",  Idade = 21 },
        new Aluno() { Numero = 3, Nome = "Cristina Fria",  Idade = 25 },
        new Aluno() { Numero = 4, Nome = "Dinis Campos" , Idade = 20 },
        new Aluno() { Numero = 5, Nome = "Soaraia Goncalves" , I
        new Aluno() { Numero = 6, Nome = "Lena Pinheiro",  Idade
        new Aluno() { Numero = 7, Nome = "Filipe Cruz",Idade = 1
```

```
var alunosIdade18e23 = alunos
                .Where(a=>a.Idade>18 && a.Idade<23);
```

Extension Methods Syntax

Jose Antunes
Lena Pinheiro
Filipe Cruz

DEIS - Instituto Superior de Engenharia de Coimbra
isec Engenharia

LINQ Extension Methods

- Seleccionar os alunos com média superior ou igual a 18?

```
...
List<Aluno> alunosExcelentes = new List<Aluno>();
foreach (var aluno in alunos)
{
    if (aluno.Media >= 18)
        alunosExcelentes.Add(aluno);
}
```

Sem LINQ

```
var alunosExcelentes = alunos.Where(a => a.Media >= 18);
```

LINQ EM

SEC/IPC @ B
ogramaç

LINQ - *Where*

```
int[] numeros = { 5, 4, 1, 3, 9, 8, 6, 7, 2, 0 };

var resultado = from n in numbers
                where n < 5
                select n;
```

```
string[] digits = { "zero", "one", "two", "three", "four",
                    "five", "six", "seven", "eight", "nine" };

var shortDigits = digits.Where((digit, index) =>
                               digit.Length < index));
```

```
five
six
seven
eight
nine
```

https://code.msdn.microsoft.com/101-LINQ-Samples-3fb9811b

isec Engenharia DEIS - Instituto Superior de Engenharia de Coimbra

Exercício:

- Seleccionar os alunos cujo nome é “Filipa Moreira” usando LINQ EM

```
Aluno[] alunos = {
        new Aluno() { Numero = 1, Nome= "Jose Antunes", Idade= 18 },
        new Aluno() { Numero = 2, Nome = "Filipa Moreira",  Idade = 21 },
        new Aluno() { Numero = 3, Nome = "Cristina Fria",  Idade = 25 },
        new Aluno() { Numero = 4, Nome = "Dinis Campos" , Idade = 20 },
        new Aluno() { Numero = 5, Nome = "Soaraia Goncalves" , Idade = 31 },
        new Aluno() { Numero = 6, Nome = "Lena Pinheiro",  Idade = 17 },
        new Aluno() { Numero = 7, Nome = "Filipe Cruz",Idade = 19  },
 };
```

```
var filipa = alunos.Where(a => a.Nome == "Filipa Moreira");
```

- Seleccionar os alunos com número 5 usando LINQ EM

```
Aluno aluno5 = alunos.Where(a => a.Numero==5);
```

isec Engenharia DEIS - Instituto Superior de Engenharia de Coimbra

Exercício:

- Obter os números pares da lista

```
IList<int> listInts= new List<int>() {
                    12,15,4,23,17,18,13
              };
```

```
var numPares = listInts.Where(n => n % 2 == 0);
```

UPC @JB
2023/2024
Prog

Exemplo: Operador *OfType*

- Obter apenas os números inteiros

```
object[] numeros =  {
      12,15.6,4.2,23,17.1,18,13
};
```

```
var apenasInteiros = numeros.OfType<int>();
```

OfType - retorna os elementos que podem ser convertidos com segurança no tipo indicado

ISEC/IPC @JB
Prog Web – 2023/2

LINQ
Ordenação: OrderBy, ThenBy

isec Engenharia DEIS - Instituto Superior de Engenharia de Coimbra

Ordenação - OrderBy

- *OrderBy()*
  - Permite ordenar uma collection por um determinado valor
- *OrderByDescending()*
  - Permite ordenar de forma descendente

23/2024 – LEI D+CE+PL – ISEC/IPC @JB

```
var alunosOrdenados =alunos
                        .OrderBy(a => a.Nome);
```

▲ 1 of 2 ▼ (extension) IOrderedEnumerable<Aluno> IEnumerable<Aluno>.OrderBy<Aluno, TKey>(Func<Aluno, TKey> keySelector)
Sorts the elements of a sequence in ascending order according to a key.
keySelector: *A function to extract a key from an element.*

Ordenação - OrderBy

```
var alunosExcelentes = from a in Alunos
                       where a.Media >= 18
                       orderby a.Nome
                       select a;
```

```
var alunosExcelentes = db.Alunos.Where(a => a.Media >= 18)
                                .OrderBy(a => a.Nome);
```

```
var alunosExcelentes = from a in Alunos
                       where a.Naluno == 18
                       orderby a.Nome, a.Media, a.DataNascimento
                      descending
                       select a;
```

isec Engenharia DEIS - Instituto Superior de Engenharia de Coimbra

LINQ Extension Methods - Ordenação

- *ThenBy() / ThenByDescending()*
  - Permite ordenação de outros campos

```
var alunosExcelentes = alunos.Where(a => a.Media >= 18)
                             .OrderBy(a => a.Nome).ThenBy(a => a.Media);
```

```
var alunosExcelentes = alunos.Where(a => a.Media >= 18)
                                 .OrderByDescending(a => a.Nome)
                                 .ThenByDescending(a => a.Media);
```

Programação Web – 2023/2024 – LEI DWCE+PL – ISEC/IPC @JB

isec Engenharia DEIS - Instituto Superior de Engenharia de Coimbra

LINQ
Projecção: Select, SelectMany

LINQ: Projeção

- *Select()*
  - Permite aplicar uma transformação a todos os elementos em qualquer estrutura de dados que implemente a *IEnumerable*

LEI D+CE DL – ISEC/IPC @JB

```
var alunos = from a in Alunos
             where a.NAluno == 1
             orderby a.Nome, a.DataNascimento descending
             select new {
                     Nome = a.Nome,
                     DataNasc = a.DataNascimento
               };
```

LINQ Extension Methods - Projeção

- Seleccionar <u>apenas o nome</u> dos alunos com média igual ou superior a 18. Devem ser ordenados pelo nome.

```
var alunosExcelentes =  alunos.Where(a => a.Media >= 18)
                               .OrderBy(a => a.Nome)
                               .Select(a => a.Nome);
```

```
var alunosExcelentes = db.Alunos.Where(a => a.Media >= 18)
```

(local variable) IQueryable<string> alunosExcelentes

```
                               .Select(a=>a.Nome);
```

ISEC - IPC @19

Web –

isec Engenharia DEIS - Instituto Superior de Engenharia de Coimbra

Projeção

- Algum problema aqui ?

```
IList<Aluno> alunosExcelentes =  alunos.Where(a => a.Media >= 18)
                                        .OrderBy(a => a.Nome)
                                        .Select(a => a.Nome);
```

```
IList<Aluno> alunosExcelentes =  alunos.Where(a => a.Media >= 18)
                                        .OrderBy(a => a.Nome)
                                        .Select(a => a.Nome);
```

Web – 2023/2024 – IPC @JB

isec Engenharia DEIS - Instituto Superior de Engenharia de Coimbra

Projeção

- Algum problema aqui ?

```
Var alunosExcelentes =  alunos.Where(a => a.Media >= 18)
                                .Select(a => a.Nome)
                                .OrderBy(a => a.Nome);
```

IPC @JB
Web – 2023/2024 –

```
Var alunosExcelentes =  alunos.Where(a => a.Media >= 18)
                                .Select(a => a.Nome)
                                .OrderBy(a => a.Nome);
```

isec Engenharia DEIS - Instituto Superior de Engenharia de Coimbra

LINQ Extension Methods - Projeção

```
var alunosExcelentes = db.Alunos.Where(a => a.Media >= 18)
                                .OrderBy(a => a.Nome)
                                .Select(a => a.Nome);
```

```
foreach (var aluno in alunosExcelentes)
    Console.WriteLine(aluno.Nome);
```

```
foreach (var aluno in alunosExcelentes)
    Console.WriteLine(aluno);
```

Programação Web – 2023/2024 – LE

Exercício 2

- Extension Methods

- Obter os quadrados dos números impares

```
IList<int> listInts= new List<int>() {
                    12,15,4,23,17,18,13
              };
```

```
var quadradoImpares = listInts.Select(x => x * x)
                              .Where(y => y % 2 != 0);
```

```
var quadradoImpares = listInts.Where(y => y % 2 != 0)
                              .Select(x => x * x);
```

Resultado é igual?
Fazem o mesmo?

DEIS - Instituto Superior de Engenharia de Coimbra

Exercício 2

- Obter os quadrados dos números impares

```
IList<int> listInts= new List<int>() {
                    12,15,4,23,17,18,13
              };
```

```
var quadradoImpares = listInts.Select(x => x * x)
                              .Where(y => y % 2 != 0);
```

```
var quadradoImpares = listInts.Where(y => y % 2 != 0)
                              .Select(x => x * x);
```

DEIS - Instituto Superior de Engenharia de Coimbra

LINQ Extension Methods - Projeção

```
var alunos = alunos.Where(a => a.Media >= 18)
                   .OrderBy(a => a.Nome)
                   .ThenByDescending(a => a.Media)
                   .Select(a => new {
                                     NomeAluno = a.Nome,
                                     NomeCurso=a.Curso.Nome
                                    } );
```

isec Engenharia DEIS - Instituto Superior de Engenharia de Coimbra

LINQ Extension Methods

```
var tes = alunos.Where(a => a.Media >= 15)
                .OrderBy(a => a.Nome)
                .ThenByDescending(a => a.Media)
                .Select(a => new
                  {
                      Nome= a.Nome,
                      Disciplinas= a.Disciplinas
                  }
            );
```

```
(local variable) IEnumerable<'a> tes

Anonymous Types:
  'a is new { string Nome, List<Disciplina> Disciplinas }
```


isec Engenharia DEIS - Instituto Superior de Engenharia de Coimbra

LINQ Extension Methods

```
var disAlunos = alunos.Where(a => a.Media >= 15)
                      .OrderBy(a => a.Nome)
                      .ThenByDescending(a => a.Media)
                      .Select(a => a.Disciplinas);
```

(local variable) IEnumerable<List<Disciplina>> disAlunos

```
foreach (var a in disAlunos)
    foreach (var b in a)
        Console.Write(b.Nome + ",");
```

Programação Web – 2023/2024

isec Engenharia DEIS - Instituto Superior de Engenharia de Coimbra

Projecção - SelectMany

- *SelectMany()*
  - Permite efetuar o *select* de uma colecção de colecções e "passa" de um `IEnumerable<IEnumerable<T>>` para `IEnumerable<T>`

LEI D+CE-PL – ISEC/IPC @JB

```
var disAlunos = alunos.Where(a => a.Media >= 15)
                      .OrderBy(a => a.Nome)
                      .ThenByDescending(a => a.Media)
                      .SelectMany(a => a.Disciplinas);
```

(local variable) IEnumerable<Disciplina> disAlunos

```
foreach (var a in disAlunos)
        Console.Write(a.Nome + ",");
```


isec Engenharia DEIS - Instituto Superior de Engenharia de Coimbra

LINQ
Agrupar: GroupBy

Programação Web – 2023/2024 – JFLD+CE+PL – ISEC/IPC @JB

LINQ Query Syntax - Agrupar

- *GroupBy()*
  - Permite agrupar os dados em grupos diferentes

Com QS

```
var alunos = from a in alunos
             group a by a.Media
             into g
             select g;
```

```
foreach (var grupo in alunos){
    Console.Write("{0} - {1}", grupo.Key, grupo.Count());
    foreach (var aluno in grupo)
        Console.Write("\t{0}", aluno.Nome);
}
```

ISEC/IPC @JB

LINQ Extension Methods - Agrupar

- *GroupBy()*

Com EM

```
var grupos = alunos
                .GroupBy(a => a.Media);
foreach (var grupo in grupos)
{
    Console.Write(grupo.Key);
    foreach (var aluno in grupo)
        Console.Write("\t{0}", aluno.Nome);
}
```

Programação Web – 2023/... PL – ISEC/IPC @JB

LINQ
Joins: Join, GroupJoin

Programação Web – 2023/2024 – EID+CE+PL – ISEC/IPC @JB

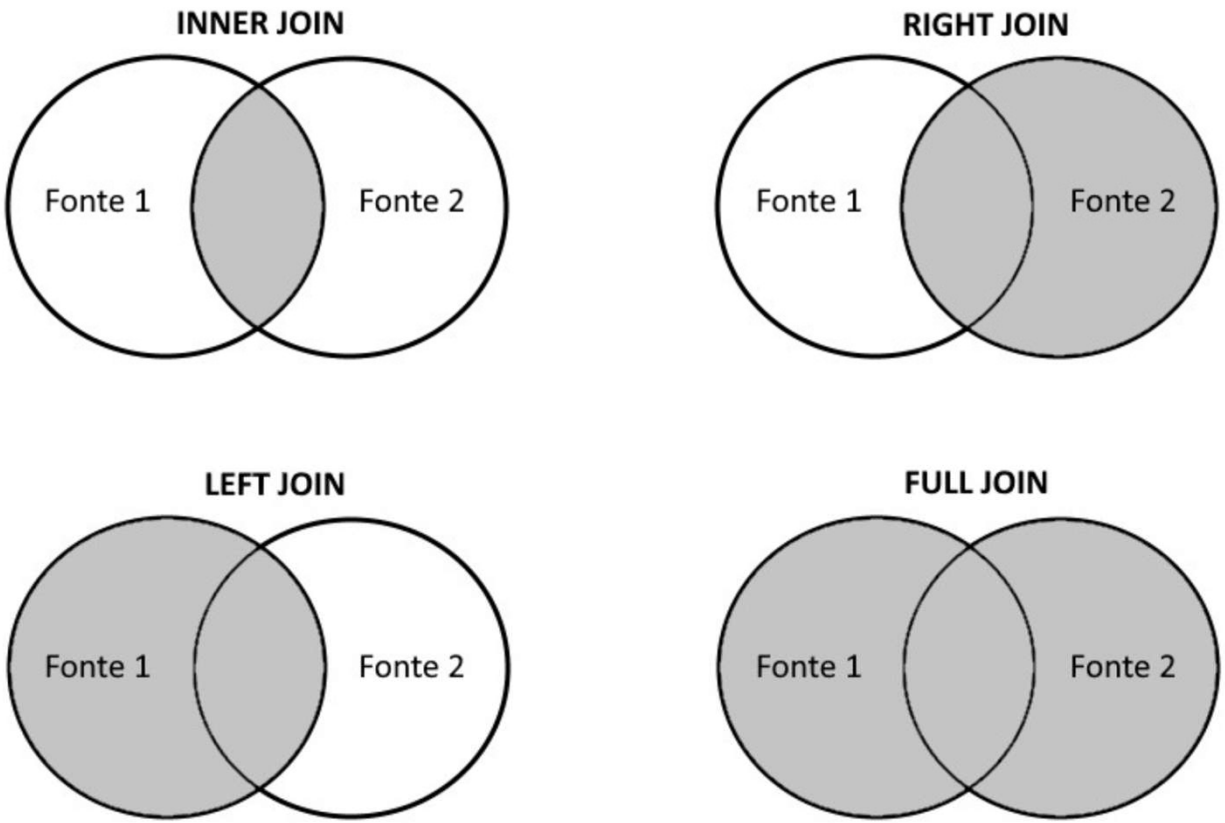
isec Engenharia DEIS - Instituto Superior de Engenharia de Coimbra

LINQ - Joins

- *Joins*
  - Utilizados para combinar diferentes listas ou tabelas com uma chave em comum.
- Existem vários tipos de Joins:
  - *Inner Join* - Equivalente ao Inner Join do SQL
  - *Group Join* - Não tem equivalência ao SQL
  - *Cross Join* - Igual ao Cross Join do SQL. Obtém-se o Produto Cartesiano

Programação Web – 2023/2024 – EI D+CE+PL – ISEC/IPC @JB

LINQ - Joins



Prc

isec Engenharia DEIS - Instituto Superior de Engenharia de Coimbra

LINQ: Inner Join

- **Exemplo**: Obter a lista comum entre as listas

```
var nomes1 = new List<string>() { "Jose", "Maria", "Filipa","Carlos" };
var nomes2 = new List<string>() { "Filipa", "Carlos", "Pedro","Joana","Nuno"};
```

```
var resultado = from n1 in nomes1
                join n2 in nomes2 on n1 equals n2
                select new { n1, n2 };
```

```
var  resultado = nomes1.Join(nomes2, n1 => n1, n2 => n2,
                                     (n1, n2) => new { n1, n2 });
```

isec Engenharia DEIS - Instituto Superior de Engenharia de Coimbra

LINQ: Inner Join

- **Exemplo:** Obter a lista de todos os nomes dos alunos e do nome do respetivo curso

```
public class Aluno
{
    public int Numero { get; set; }
    public string Nome { get; set; }
    public int Idade { get; set; }
    public int Media { get; set; }
    public Curso Curso { get; set; }
}
```

```
public class Curso
{
    public int CursoId { get; set; }
    public string Nome { get; set; }
}
```

CE+PL – ISEC/IPC @

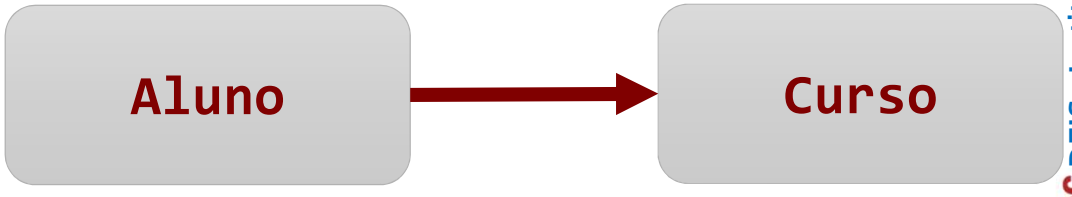
DEIS - Instituto Superior de Engenharia de Coimbra

LINQ: Inner Join

```
public class Aluno
{
    public int Numero { get; set; }
    public string Nome { get; set; }
    public int Idade { get; set; }
    public int Media { get; set; }
    public Curso Curso { get; set; }
}
```

```
public class Curso
{
    public int CursoId { get; set; }
    public string Nome { get; set; }
}
```

- Para ligação entre entidades, pode-se recorrer às propriedades específicas que as interligam



DEIS - Instituto Superior de Engenharia de Coimbra

LINQ: Inner Join

- **Exemplo:** Obter a lista de todos os nomes dos alunos e do nome do respetivo curso

ISEC/IPC @20

```
var alunos = from a in alunos
             select new { NomeAluno = a.Nome,
                          NomeCurso = a.Curso.Nome
                        };
```

```
var alunos = alunos.Select(a => new {
                          NomeAluno = a.Nome,
                          NomeCurso=a.Curso.Nome
      } );
```

LINQ Query Syntax: Group Join

- Total de alunos por curso

Curso	Alunos
Licenciatura em Engª Informática	4
Mestrado em Engenharia Informática	0

CE+PL – ISEC/IPC @JB
– 2023/2024

```
var alunos = from c in db.Cursos
             join a in db.Alunos on c.CursoId equals a.CursoId into g
             select new { NomeCurso = c.Nome, TotalAlunos = g.Count()};
```

Instituto Superior de Engenharia de Coimbra

LINQ Extension Methods: Group Join

- Total de alunos por curso

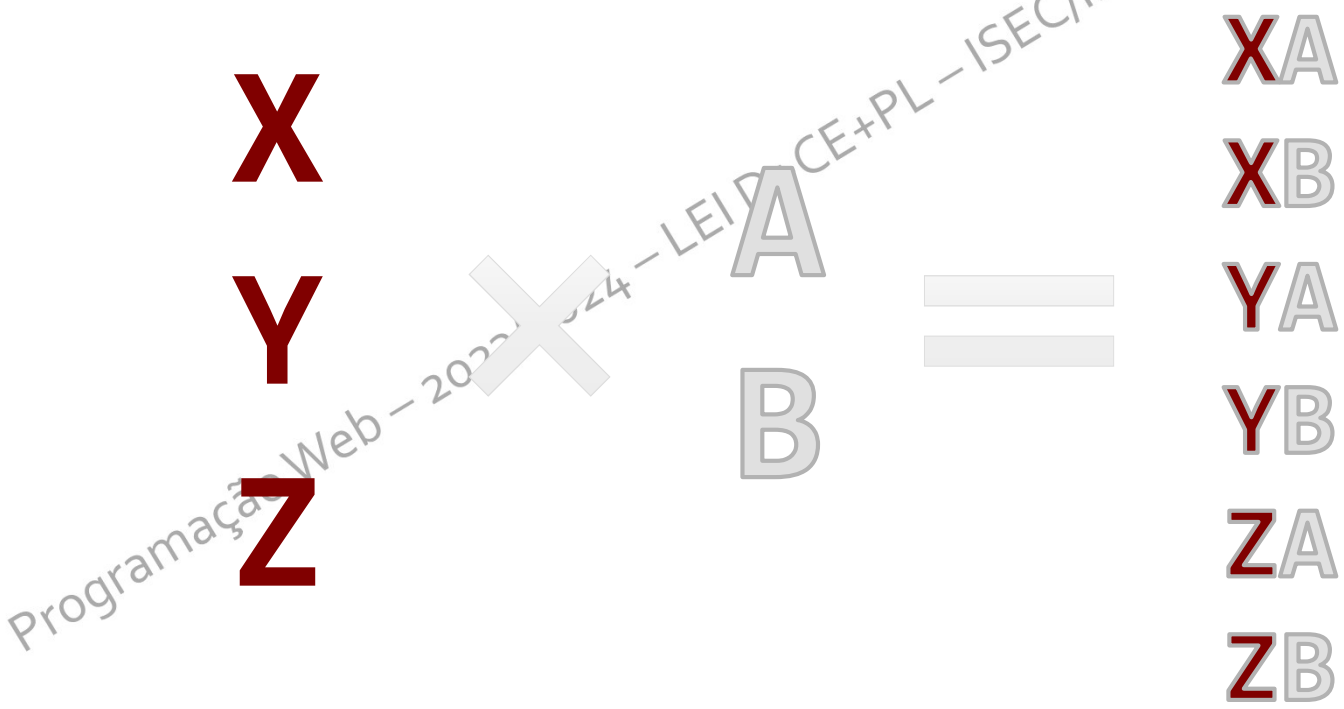
Curso	Alunos
Licenciatura em Engª Informática e de Sistemas	4
Mestrado em Engenharia Informática	0

```
var alunosPorCurso = cursos
            .GroupJoin(alunos, c => c.CursoId, a=>a.CursoId,
                        (curso, alunos) => new {
                            NomeCurso = curso.Nome,
                            TotalAlunos=alunos.Count()
                        });
```

DEIS - Instituto Superior de Engenharia de Coimbra

LINQ: Cross Join

- Exatamente igual ao *Cross Join* existente em SQL



Programação Web – 2023/2024 – LEI PCE+PL – ISEC/IPC @ 2

isec Engenharia DEIS - Instituto Superior de Engenharia de Coimbra

LINQ Query Syntax: Cross Join

- Lista de todos os cursos de todos os alunos:

ISEC/IPC @ UP

```
var alunosQ = from c in cursos
              from a in alunos
              select new {
                 NomeCurso = c.Nome,
                 NomeAluno = a.Nome
              };
```

Progra

isec Engenharia DEIS - Instituto Superior de Engenharia de Coimbra

LINQ Extension Methods: Cross Join

- Lista de todos os cursos de todos os alunos

```
var alunosEM = cursos.SelectMany(c => alunos,
                                 (curso, aluno) => new {
                                     NomeCurso = curso.Nome,
                                     NomeAluno = aluno.Nome
                                 });
```

PL – ISEC/IPC @ UP

Outros métodos

- *Any()*
  - Retorna um *boolean*
  - **Exemplo:** Existe algum aluno com média superior a 10?
- *All()*
  - Retorna um *boolean*
  - **Exemplo:** Todos os alunos tem média positiva?

```
var todosComMediaPositiva = alunos.All(a => a.Media > 10);
```

Programação Web – 2023/2024 – LEI D+CE+RL – ISEC/IPC @JB

isec Engenharia DEIS - Instituto Superior de Engenharia de Coimbra

Outros métodos - Particionamento

- Úteis para paginação
  - *Skip()* – Permite "saltar", *skip,* um determinado número de elementos e retorna os restantes
  - *Take()* – Retorna um número específico de elementos

```
var lAlunos = alunos.Skip(3).Take(2);
```

Program... 2023/2024 – LEI LD+CE TPL – ISEC/IPC @JB

isec Engenharia DEIS - Instituto Superior de Engenharia de Coimbra

Outros métodos - Particionamento

- Úteis para paginação
  - *TakeWhile* – Retorna elementos que satisfaçam uma determinada condição. Quando a condição já for verdadeira, termina.
  - *SkipWhile* – Permite fazer o skip de elementos em sequência baseados numa condição. Quando deixar de ser verdadeira, retorna os restantes elementos.

Programação Web – 2023/2024 – LEI D+CE+DL – ISEC/IPC @JB

Exercício:

```
IList<string> keywordPWList = new List<string>(){
    "C#",
    "Aprender C#",
    "ASP.NET MVC com C#",
    "Entity Framework",
    "Linq",
    "Lambda",
    "Identity",
    "C# in the End",
    };

var result = keywordPWList.TakeWhile(x => x.Contains("C#"));
var result = keywordPWList.SkipWhile(x => x.Contains("C#"));
```

Qual o output?

isec Engenharia DEIS - Instituto Superior de Engenharia de Coimbra

Outros métodos

- *First*
- *FirstOrDefault*
- *Last*
- *LastOrDefault*
- *Single*
- *SingleOrDefault*

Permitem retornar um único valor

Podem ser invocados com ou sem predicado.

Programação Web – 2023/2024 – LEI D+CE+M – ISEC/IPC @JB

isec Engenharia DEIS - Instituto Superior de Engenharia de Coimbra

LINQ – First() / Last()

- *First()* / *Last()*
  - Retorna o primeiro/último elemento da sequência ou o primeiro/último elemento correspondente ao predicado especificado
    - Se não existirem elementos na sequência, é gerada uma excepção `InvalidOperationException` com a mensagem *"Sequence contains no elements".*
    - Se não forem encontrados elementos que satisfaçam a condição do predicado é gerada uma `InvalidOperationException` com a mensagem "*Sequence contains no matching element*".

Programação Web – 2023/2024 – LEI D+CE+RL – ISEC/IPC @JB

isec Engenharia DEIS - Instituto Superior de Engenharia de Coimbra

LINQ – …OrDefault()

- *FirstOrDefault()* / *LastOrDefault()*
  - Retorna o primeiro/último elemento da sequência ou o primeiro/último elemento correspondente ao predicado especificado
    - Se não existirem elementos na sequência ou se não forem encontrados elementos que satisfaçam a condição do predicado é <u>retornado o valor por omissão</u> do tipo da sequência usando *default (T)*

Programação Web – 2023/2024 – LEI D+CE+RL – ISEC/IPC @JB

isec Engenharia DEIS - Instituto Superior de Engenharia de Coimbra

LINQ – Single()

- ***Single()***
  - Se a sequência contém exactamente um elemento, ou exactamente um elemento que satisfaça a condição especificada no predicado, então esse elemento é devolvido.
    - Se não existirem elementos na sequência, ou se não forem encontrados elementos que satisfaçam a condição do predicado, é gerada uma excepção `InvalidOperationException` com a mensagem *"Sequence contains no elements".*
    - Se existir na sequência mais do que um elemento, ou se existir mais do que um elemento que satisfaça a condição do predicado é gerada a excepcao `InvalidOperationException` com a mensagem "`Sequence contains more than one element`".

Programação Web – 2023/2024 – LEI D+CD, RL – ISEC/IPC @JB

LINQ – SingleOrDefault()

- *SingleOrDefault()*

  - Igual ao comportamento do Single(), no entanto se não existirem elementos ou se não existirem elementos que satisfaçam a condição do predicado, em vez de gerar excepção, é retornado o valor por omissão do tipo correspondente default(T).

Programação Web – 2023/2024 – LEID+CD+PL – ISEC/IPC @JB

isec Engenharia DEIS - Instituto Superior de Engenharia de Coimbra

LINQ – ElementAt, ElementAtOrDefault

- ***ElementAt()***
  - Retorna um elemento da sequência baseado na posição *index* especificada

- ***ElementAtOrDefault()***
  - Igual ao comportamento do Element(), no entanto se não existir elemento no indice especificado, retorna o valor por omissão

Programação Web – 2023/2024 – LEI D+CE+PL – ISEC/IPC @JB

isec Engenharia DEIS - Instituto Superior de Engenharia de Coimbra

Exercício:

- Qual o resultado das seguintes instruções?

```
Aluno a1= alunos.First();
var b1= alunos.Last();
var a2= alunos.First(a=> a.Nome == "Filipa Moreira");
var b2= alunos.Last(a=> a.Nome == "Filipa Moreira");
var a3 = alunos.First(a => a.Nome == "Filipa Que?");
var a4 = alunos.FirstOrDefault(a => a.Nome == "Filipa Nao Tem");
var b4 = alunos.LastOrDefault(a => a.Nome == "Filipa Nao Tem");
```

```
Aluno[] alunos = {
        new Aluno() { Numero = 1, Nome= "Jose Antunes", Idade= 18 },
        new Aluno() { Numero = 2, Nome = "Filipa Moreira",  Idade = 21 },
        new Aluno() { Numero = 3, Nome = "Cristina Fria",  Idade = 25 },
        new Aluno() { Numero = 4, Nome = "Dinis Campos" , Idade = 20 },
        new Aluno() { Numero = 5, Nome = "Soaraia Goncalves" , Idade = 31 },
        new Aluno() { Numero = 6, Nome = "Filipa Moreira",  Idade = 26 },
        new Aluno() { Numero = 7, Nome = "Lena Pinheiro",  Idade = 17 },
        new Aluno() { Numero = 8, Nome = "Filipe Cruz",Idade = 19  },
 };
```

isec Engenharia DEIS - Instituto Superior de Engenharia de Coimbra

Exercício:

- Qual o resultado das seguintes instruções?

```
var aluno = alunos.Single(a =>a.Nome==“António Soares");
var aluno = alunos.Single(a =>a.Nome=="Cristina Fria");
var aluno = alunos.SingleOrDefault(a => a.Nome == “Carlos Afonso");
```

```
Aluno[] alunos = {
        new Aluno() { Numero = 1, Nome= “Carlos Antunes", Idade= 18 },
        new Aluno() { Numero = 2, Nome = "Filipa Moreira",  Idade = 21 },
        new Aluno() { Numero = 3, Nome = "Cristina Fria",  Idade = 25 },
        new Aluno() { Numero = 4, Nome = "Dinis Campos" , Idade = 20 },
        new Aluno() { Numero = 5, Nome = "Soaraia Goncalves" , Idade = 31 }
,
        new Aluno() { Numero = 6, Nome = "Filipa Moreira",  Idade = 26 },
        new Aluno() { Numero = 7, Nome = "Lena Pinheiro",  Idade = 17 },
        new Aluno() { Numero = 8, Nome = "Filipe Cruz",Idade = 19  },
 };
```

LINQ
Agregação: Count, Sum, Min, Max, Average

Programação Web – 2023/2024 – LEI+CE+PL – ISEC/IPC @JB

isec Engenharia DEIS - Instituto Superior de Engenharia de Coimbra

Funções de Agregação

- ***Count()*** - Contagem

```
var num = alunos.Count();
```

```
var quant = alunos.Where(a=>a.Media > 18)
                  .Count();
```

```
var quant = alunos.Count(a=>a.Media > 18);
```

IPC @JB
LEI D+CL
mação We

isec Engenharia DEIS - Instituto Superior de Engenharia de Coimbra

Funções de Agregação

- ***Sum()*** - Somatório

```
var quant = alunos.Where(a=>a.Media)
                  .Sum();
```

```
var quant = alunos.Sum(a=>a.Media);
```

Programaç 023/2024 – DEIS/ISEC/IPC @JB

isec Engenharia DEIS - Instituto Superior de Engenharia de Coimbra

Funções de Agregação

- ***Min()*** / ***Max()***
  - Permite obter o valor mínimo/máximo de uma *list* ou *collection*

```
int[] num = {2,1,10,4,5,6,7,8,9};
int minimo = num.Min();
int maximo = num.Max();
```

```
var mediaMaior = alunos.Max(a => a.Media);
```

CE+PL – ISEC/IPC @JB

Funções de Agregação

- ***Average()***
  - Permite calcular a média

```
var quant = alunos.Select(a => a.Media).Average();
```

```
var quant = alunos.Average(a => a.Media);
```

Programação +PL – ISEC/IPC @JB 2024

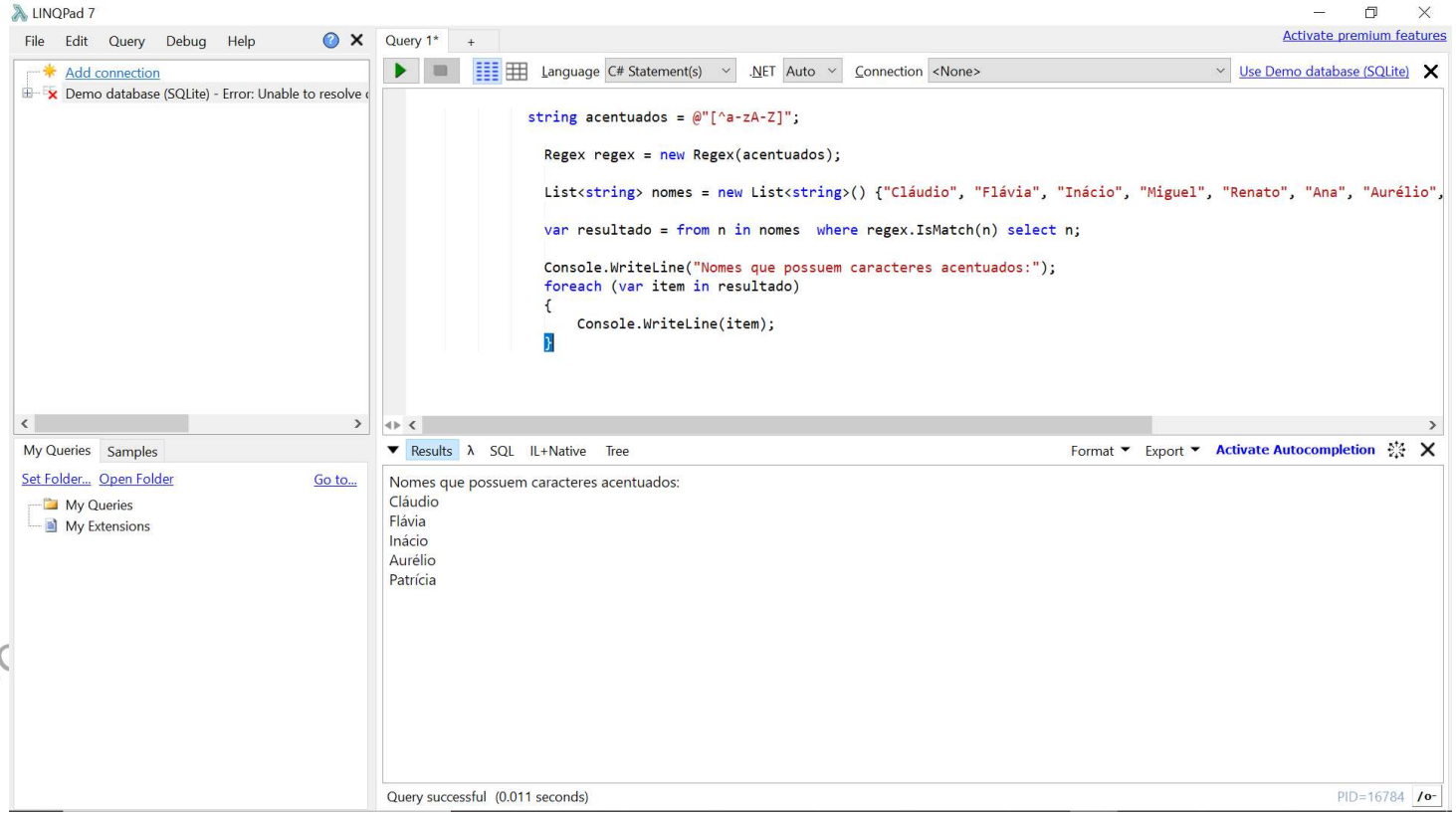
Métodos para Conjuntos

- ***Union()***
  - Remove elementos repetidos
- ***Intersect()***
- ***Concat()***
  - Não remove os elementos repetidos
- ***Except()***
  - Implica distintos, portanto, remove elementos repetidos
- ***Distinct()***

Programação Web – 2023/2024 – LEI D+CE+PL – ISEC/IPC @JB

isec Engenharia DEIS - Instituto Superior de Engenharia de Coimbra

Treinar LINQ



- Linqpad 7
  - Compatível com as últimas versões do C# e do .NET e Core 3.x
  - Gratuito disponível em https://www.linqpad.net/Download.aspx

Programaçãc

isec Engenharia DEIS - Instituto Superior de Engenharia de Coimbra

Exercício:

- Seleccionar os alunos cujo nomes começam por “F”

```
Aluno[] alunos = {
        new Aluno() { Numero = 1, Nome= "Jose Antunes", Idade= 18 },
        new Aluno() { Numero = 2, Nome = "Filipa Moreira",  Idade = 21 },
        new Aluno() { Numero = 3, Nome = "Cristina Fria",  Idade = 25 },
        new Aluno() { Numero = 4, Nome = "Dinis Campos" , Idade = 20 },
        new Aluno() { Numero = 5, Nome = "Soaraia Goncalves" , Idade = 31 },
        new Aluno() { Numero = 6, Nome = "Lena Pinheiro",  Idade = 17 },
        new Aluno() { Numero = 7, Nome = "Filipe Cruz",Idade = 19  },
 };
```

```
var alunosF = alunos.Where( (a) => {
                return (a.Nome.StartsWith("F"));
            }  );
```

```
var alunosF = alunos.Where( a =>
                a.Nome.StartsWith("F"));
```


isec Engenharia DEIS - Instituto Superior de Engenharia de Coimbra

Exercício:

- Seleccionar os alunos cujo ultimo nome começa por “F”

```
Aluno[] alunos = {
        new Aluno() { Numero = 1, Nome= "Jose Antunes", Idade= 18 },
        new Aluno() { Numero = 2, Nome = "Filipa Moreira",  Idade = 21 },
        new Aluno() { Numero = 3, Nome = "Cristina Fria",  Idade = 25 },
        new Aluno() { Numero = 4, Nome = "Dinis Campos" , Idade = 20 },
        new Aluno() { Numero = 5, Nome = "Soaraia Goncalves" , Idade = 31 },
        new Aluno() { Numero = 6, Nome = "Lena Pinheiro",  Idade = 17 },
        new Aluno() { Numero = 7, Nome = "Filipe Cruz",Idade = 19  },
 };
```

```
var alunosF = alunos.Where( a => a.Nome.Split(' ')
                                .Last()
                                .StartsWith("F"));
```

DEIS - Instituto Superior de Engenharia de Coimbra

Exercício:

- Obter o aluno mais novo da turma?

```
var maisNovo = alunos.OrderBy(x => x.Idade).First();
```

- E este ? Funciona?

```
var maisNovo = alunos.Select(x => x.Idade).Min();
```

ção Web – 2023/2024 – LEI D
ECC/IPC @JB

Exercício:

- Ordenar os alunos pela média descendentemente e depois pela idade ascentente

```
var ordenar = alunos.OrderByDescending(x => x.Media)
                    .ThenBy(x=>x.Idade);
```

Programação Web – 2 ... D+CE+PL – ISEC/IPC @ 2

isec Engenharia DEIS - Instituto Superior de Engenharia de Coimbra

Exercício:

- Obtenha, com uma simples instrução os alunos agrupados pela sua média.
- Apresente o resultado como se apresenta na figura

LEI D+CE – PL – ISEC/IPC @23

```
var gruposIdade = alunos.GroupBy(x => x.Media);

foreach (var item in gruposIdade)
{
    Console.WriteLine("Média = "+item.Key);
    foreach (var aluno in item)
        Console.WriteLine("   Nº "+aluno.Numero+": "+aluno.Nome);
}
```

```
Média = 12
   Nº 1: Jose Antunes
   Nº 6: Lena Pinheiro
   Nº 7: Filipe Cruz
Média = 16
   Nº 2: Filipa Moreira
   Nº 2: Filipa Moreira
Média = 18
   Nº 3: Cristina Fria
Média = 17
   Nº 4: Dinis Campos
Média = 19
   Nº 5: Soaraia Goncalves
```

isec Engenharia DEIS - Instituto Superior de Engenharia de Coimbra

Programação Web

Aulas Teóricas – Capítulo 2 – 2.1

1º Semestre - 2023/2024

Departamento de Engenharia Informática e de Sistemas
Instituto Superior de Engenharia de Coimbra/Instituto Politécnico de Coimbra

Programação Web

Construção de Websites em ASP.NET Core

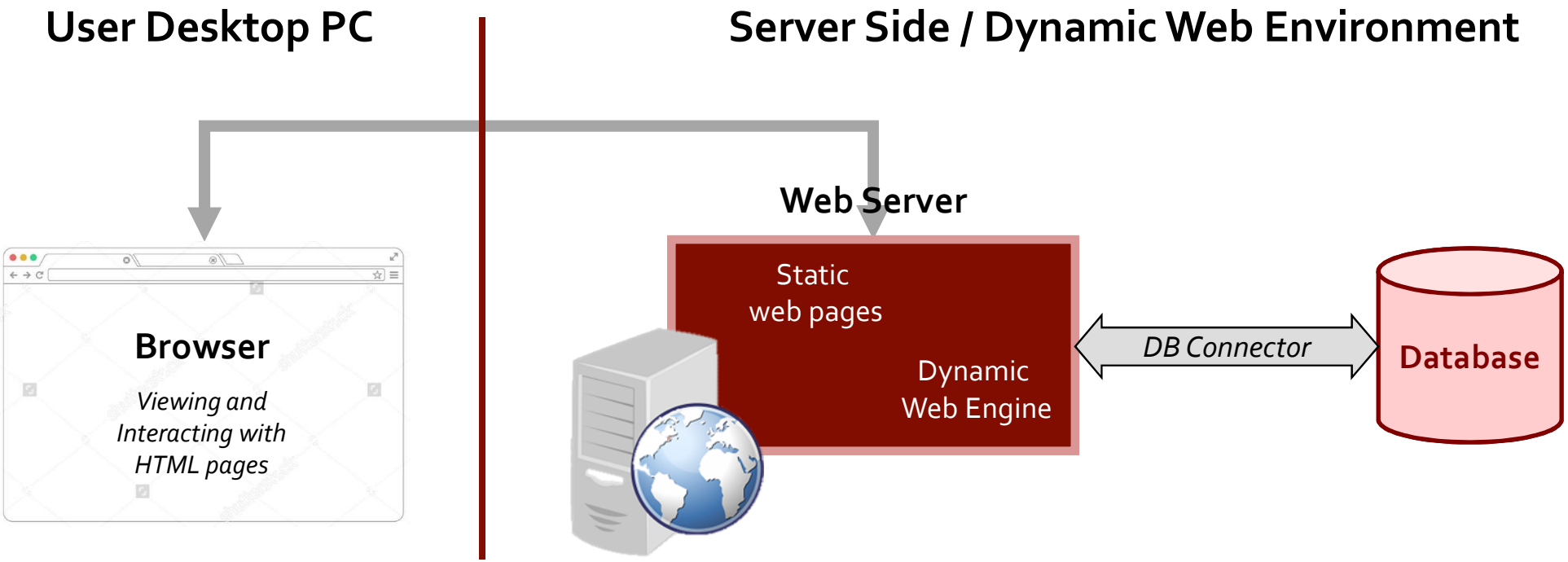
Departamento de Engenharia Informática e de Sistemas
Instituto Superior de Engenharia de Coimbra/Instituto Politécnico de Coimbra

Aplicações Web

- Conteúdo Estático
  - Não pode ser atualizado ou alterado sem efetuar alterações no código-fonte – praticamente já não existem

- Conteúdo Dinâmico
  - Conteúdos podem ser alterados ou atualizados sem necessidade de efetuar alterações no código-fonte
    - Existem Duas Perspectivas da noção de Dinamismo: alteração de <u>características</u> dos elementos HTML e alteração do <u>conteúdo</u> das páginas

Arquitectura Típica



Firefox, Chrome, Edge, etc
...

Microsoft IIS
Apache HTTPD, Tomcat

JDBC, ODBC, ADO.NET LINQ, ORMs

SQL Server, Oracle, MySQL, PostgreSQL

isec Engenharia DEIS - Instituto Superior de Engenharia de Coimbra

Aplicações Web

- Aplicação que pode ser acedida por utilizadores com recurso a um *web browser*
- Apenas necessita ser instalada no servidor web
  - Modelo Cliente – Servidor diferente do tradicional
- Manutenção fácil de providenciar
- Multiplataforma
- Acessível de qualquer parte, desde que haja ligação à Internet

Tecnologias e *Frameworks*

C# 9

LINQ
Language Integrated Query

ASP.NET Core

ASP.Net Core Entity Framework

ASP.Net Core Identity

Razor
(Razor View Engine)

E HTML?
E Java Script?
E CSS?

JSP

PHP

...

Ruby

Node.js

Blazor

isec Engenharia DEIS - Instituto Superior de Engenharia de Coimbra

ASP.NET Core

- Framework da Microsoft para desenvolvimento de aplicações Web.
- A actual versão do ASP.Net Core, permite desenvolver aplicativos da Web baseadas na Cloud usando MVC, Razor Pages e Blazor
  - *Server-side* + *Client Side*
  - *Blazor Server*
  - *Blazor WebAssembly (WAsm)*

ASP.NET Core

- Este framework está directamente relacionado com o .NET Framework
- Sucessor da tecnologia ASP e ASP.Net MVC
- Também é Baseado em eventos
- Apesar de se poder continuar a usar uma diversidade de linguagens de programação, C#, F#, VB.NET (parcialmente), no desenvolvimento em .Net tem sido usado preferencialmente o C#, nomeadamente a última versão 9.0

Evolução do ASP.NET

1996

Active Server Pages (Classic ASP)

2002

ASP.NET (Web Forms)

2008

ASP.NET MVC 1/2/3/4/5

2010

ASP.NET Web Pages

2012

ASP.NET Web API
ASP.NET SignalR

Desde 2015

ASP.NET ~~MVC 6~~
Core ~~MVC~~
Core

.Net

Razor | Blazor
(Browser+ Razor)
(Server | Wasm)

Evolução do ASP.NET

ASP.NET Core 1.0 → 1.0.4 → 1.1
(2016)

↓

ASP.NET Core 2.0 → 2.1 → 2.2
(2017)

↓

ASP.NET Core 3.0 → 3.1
(2018)

→

.NET 5
(2020)

↑

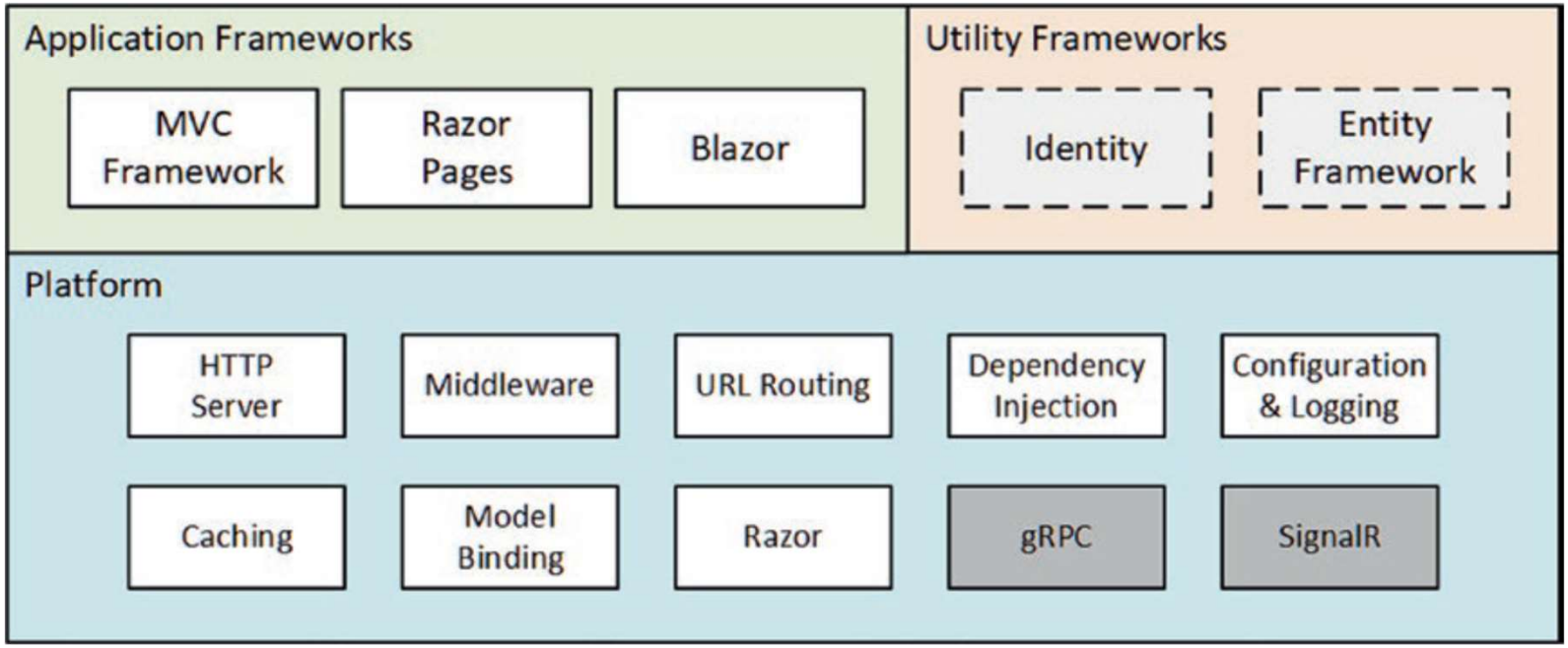
.NET 6
(2021)

↑

.NET 8
(2023)

DEIS - Instituto Superior de Engenharia de Coimbra

ASP.NET Core



Estrutura do ASP.Net Core [1]

isec Engenharia DEIS - Instituto Superior de Engenharia de Coimbra

ASP.NET Core

- O ASP.NET original dependia de um modelo de desenvolvimento designado por Web Pages, que significativamente o desenvolvimento de sites mas os projectos utilizados para o seu desenvolvimento não eram fáceis de gerir
- O MVC Framework, a versão anterior ao actual ASP.Net Core foi posteriormente introduzido e era utilizado em conjunto com as Web Pages com um modelo de desenvolvimento
- MVC significa Model-View-Controller, é um padrão de design através do qual as áreas de funcionalidade, também designadas por separação de interesses, são definidas de forma independente

ASP.NET Core

- As primeiras versões do framework MVC foram construídas sobre as bases do ASP.NET as quais tinham sido originalmente projetadas para o desenvolvimento das Web Pages
- Com a adopção do .NET Core, o ASP.NET tornou-se ASP.NET Core, e o framework MVC foi reconstruído tornando-se uma base aberta, extensível e multiplataforma
- O framework MVC continua sendo uma parte importante do ASP.NET Core mas no entanto a sua utilização já não é obrigatória e surgiu a abordagem de construção de sites designada por ***SPA's***, Aplicativos de Página Única.
  - Nesta abordagem SPA, os browsers fazem a solicitação de uma única página HTTP e em sequência o servidor envia um documento HTML, geralmente escrito num cliente JavaScript avançado por exemplo Angular ou React.
  -

isec Engenharia DEIS - Instituto Superior de Engenharia de Coimbra

ASP.NET Core

- A mudança para SPA's teve como resultado que a utilização do framework MVC já não é tão importante como originalmente era

  - No entanto no desenvolvimento utilizando o ASP.Net Core, continua-se a utilizar e é recomendável que assim seja feito, o MVC mas tão somente como um meio para se organizar minimamente as partes do site a desenvolver

- Desse modo o famework MVC continua a ser útil sendo então utilizado para dar suporte ao desenvolvimento de SPAs com recurso a serviços da web

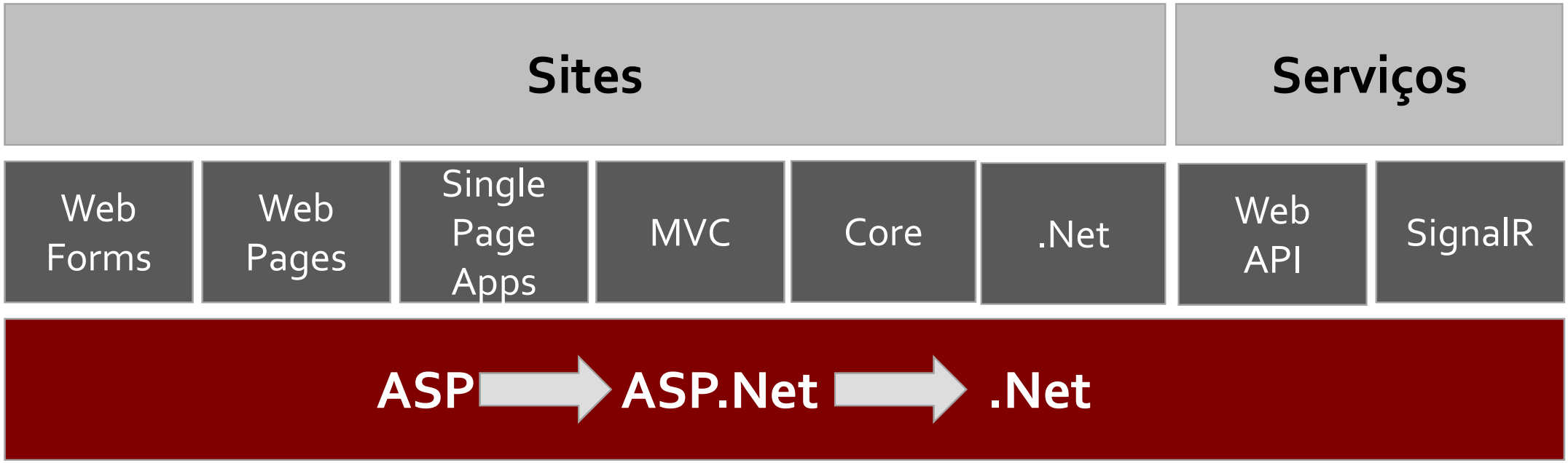
ASP.NET Core

Aulas Laboratoriais

ASP.NET Core 6

isec Engenharia DEIS - Instituto Superior de Engenharia de Coimbra

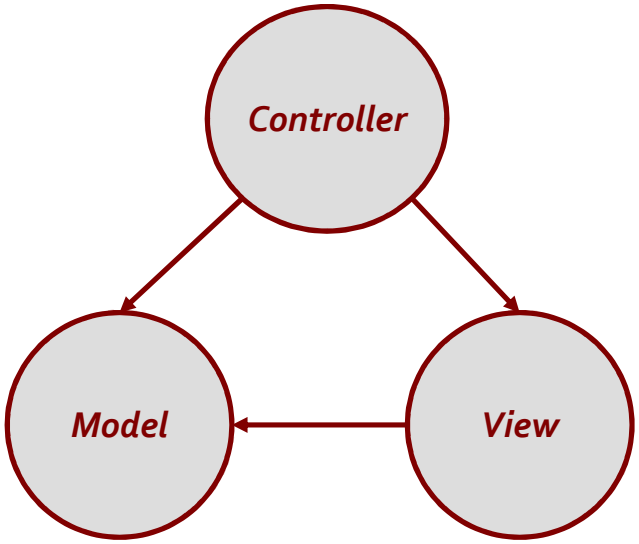
ASP.NET Core



isec Engenharia DEIS - Instituto Superior de Engenharia de Coimbra

ASP.NET Core

- Desenvolvimento no padrão MVC
- MVC
  - Padrão de arquitetura de software
  - Separação da aplicação em três **<u>componentes</u>**:
    - **Model** (Manipula Regras de Negócio, Lógica e Dados)
    - **Controller** (Interação com utilizador e Lógica de entrada)
    - **View** (Aqui é produzida a camada de Apresentação)



isec Engenharia DEIS - Instituto Superior de Engenharia de Coimbra

ASP.NET Core

- Com a estruturação no padrão MVC, consegue-se:
  - *Test Driven Development* e Reutilização de código
    - Em MVC o *controller* é uma classe separada, possibilitando automatização de testes
    - *Controllers* não estão limitados a uma view específica
    - Maior Desempenho e Escalabilidade
- Nas Views, através do Razor View Engine, são geradas as páginas HTML que depois são enviadas ao Browser

ASP.NET *Core*

- *Razor View Engine*
  - O Razor, é um motor que a plataforma usa para gerar o código HTML a ser enviado ao browser. Tem uma sintaxe própria e simples para criação dinâmica de páginas *(embedding server code (C#)*
  - As Views são como que **templates** que são usadas para produzir o HTML+JS+CSS que vai ser enviado aos browsers (o programador praticamente não escreve código HTML)
  - Para se produzir esse HTML, o **Controlador** de acordo com o solicitado pelo utilizador através do browser, obtém do **Model** os dados necessários, de acordo com as regras de negócio definidas no model, consultando por exemplo a Base de Dados associada ao site, de seguida seleciona a **View** correspondente a qual, a partir destes dados e do código definido na view, produz recorrendo ao Razor View Engine a págima HTML a ser enviada ao browser

ASP.NET *Core*

- *Blazor*
  - Actualmente há 2 variantes do Blazor:
    - O Blazor Server
    - O Blazor WebAssembly
  - *A designação Blazor resulta da junção das designações Browser + Razor*
  - O Blazor muda muito o paradigma atual da Programação Web, em particular o Blazor Web Assembly, Blazor WAsm
  - *A Microsoft lançou a versão RTM (release-to-manufacturing) do Blazor com o .NET Core 3, sendo uma nova estrutura de front-end que resolve "todos" os problemas mencionados anteriormente*

ASP.NET *Core*

- ***Blazor***
  - Com o ***Blazor***, pode-se usar *C#* e *.NET* como estrutura central para escrever o *front-end* das aplicações, e desse modo os programadores *Web* habituados a tecnologias *Microsoft*, como é o caso do *ASP.Net Core*, tem assim muitas vantagens na adopção do *Blazor* pois basicamente usarão para o desenvolvimento do *front-end* tecnologias que já dominam, tal como *Razor*, *HTML* e C# para definir a interface de utilizador

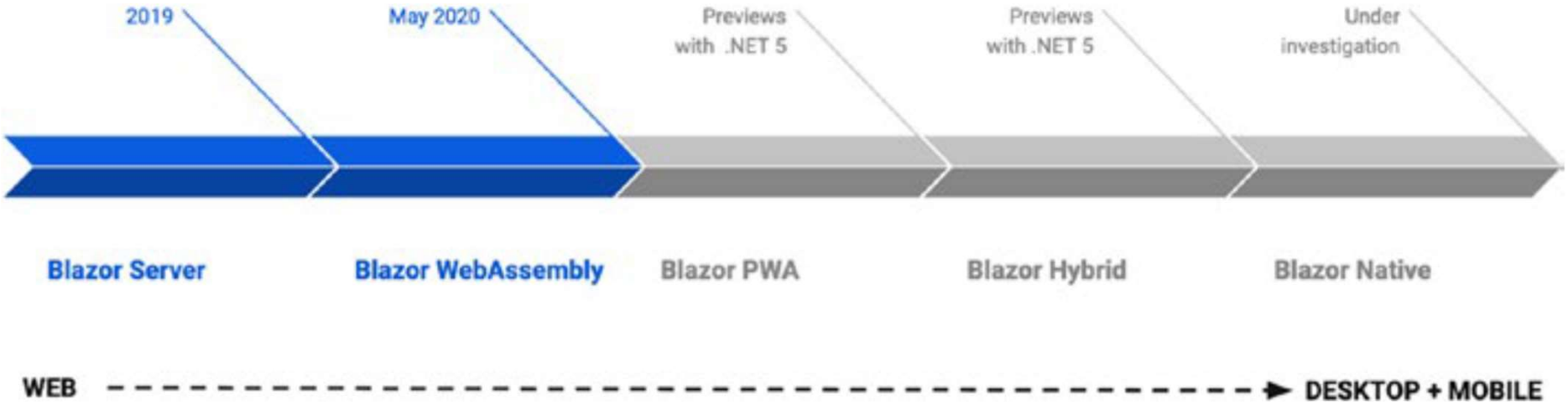
  - ***Blazor*** permite também que se execute o *front-end* diretamente no navegador, fornecendo todas as ferramentas necessárias para o desenvolvimento de aplicativos de página única

ASP.NET *Core*

- ***Blazor***
  - O ***Blazor*** foi criado em 2017 como um projeto pessoal de Steve Sanderson, que apresentou uma versão prévia do *Blazor* baseado em *DotNetAnywhere*, um *.NET* Intérprete de linguagem intermediária (IL)
  - Nesta altura o *Blazor* foi também adicionado ao repositório *ASP.Net GitHub* como um projeto experimental, mas a adesão da comunidade convenceu a *Microsoft* a mover o projeto para a equipe *ASP.Net*, substituindo *DotNetAnywhere* com *Mono*, que era o código aberto mais conhecido da plataforma baseada no *.NET Framework* (https://www.mono-project.com/).

ASP.NET *Core*

- ***Blazor***



Com o lançamento do ***.NET Core 3***, o ***Blazor*** tornou-se parte deste *framework* com um *road map* de desenvolvimento muito ambicioso como mostrado

ASP.NET *Core*

- Actualmente o *Blazor* só permite a criação de *front-end* de aplicativos web com, mas de acordo com o *road map*, será possível o desenvolvimento de aplicações para *desktops* e dispositivos móveis, passando por uma abordagem Progressiva da Aplicação Web (**PWA**) como uma etapa intermediária.

- ***Blazor Server*** é a versão que acompanha o ***.NET Core 3*** e permite pré-renderizar o ***HTML*** da aplicação e executar o código ***C#*** do lado do servidor e enviar as alterações da interface do utilizador para a página por meio do ***SignalR***

ASP.NET *Core*

- O ***Blazor WebAssembly*** que também já está disponível, executa o código C# diretamente no navegador, depois de compilado.
- O ***Blazor WebAssembly*** foi distribuído com o ***.NET Core 3.1.300*** e executa nesta versão ou numa posterior
- O ***Blazor WAsm***, traduz uma grande revolução no paradigma da Programação Web ao executar directamente no lado do cliente código que até aqui era somente executado do lado do servidor, como é o caso do ***C#***

ASP.NET *Core*

- O ***Blazor Hybrid*** será um renderizador ***.NET*** nativo para *Electron* e *WebView*, e permitirá o desenvolvimento de aplicações nativas funcionando *online* e *offline* (Elétron (electronjs.org) é um projeto de código aberto popular para a criação de aplicações de *desktop* de plataforma cruzada usando tecnologias da internet como por exemplo o *Visual Studio*. Esse código é baseado no Electron.
- ***Blazor Native*** terá o mesmo modelo de programação, mas sem renderização de *HTML*.
- Nota: No último capítulo o ***Blazor Server*** e ***Wasm*** serão abordados com mais detalhe

ASP.NET *Web API*

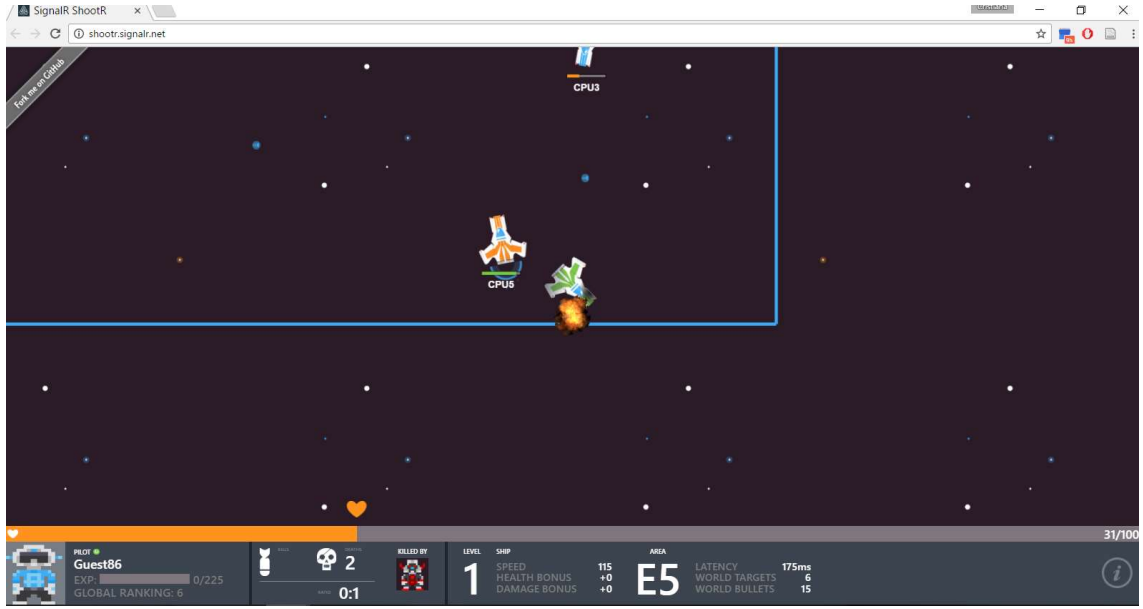
- Framework que permite desenvolver serviços HTTP
  - REST-based *webservices*
- ***REST*** significa *Representational State Transfer*. Em português, **Transferência de Estado Representacional**. Trata-se de uma abstração da arquitectura da Web. Resumidamente, o REST consiste em princípios/regras/constraints que, quando seguidas, permitem a criação de um projecto com interfaces bem definidas. Desta forma, permite, por exemplo, que aplicações comuniquem entre si (obtido de https://becode.com.br/o-que-e-api-rest-e-restful/, consulte)

ASP.NET *SignalR*

- É uma biblioteca integrada na framework
- Permite desenvolver funcionalidades "*real-time*" numa aplicação *ASP.NET Core*
  - Possibilita que o *server-side* faça, em tempo real, *push* aos clientes conectados
    - Ex: *Chat, Cotações da bolsa*
- Usa *WebSockets* quando disponível
- Actualmente é também usada para implementar o *Blazor WASM*

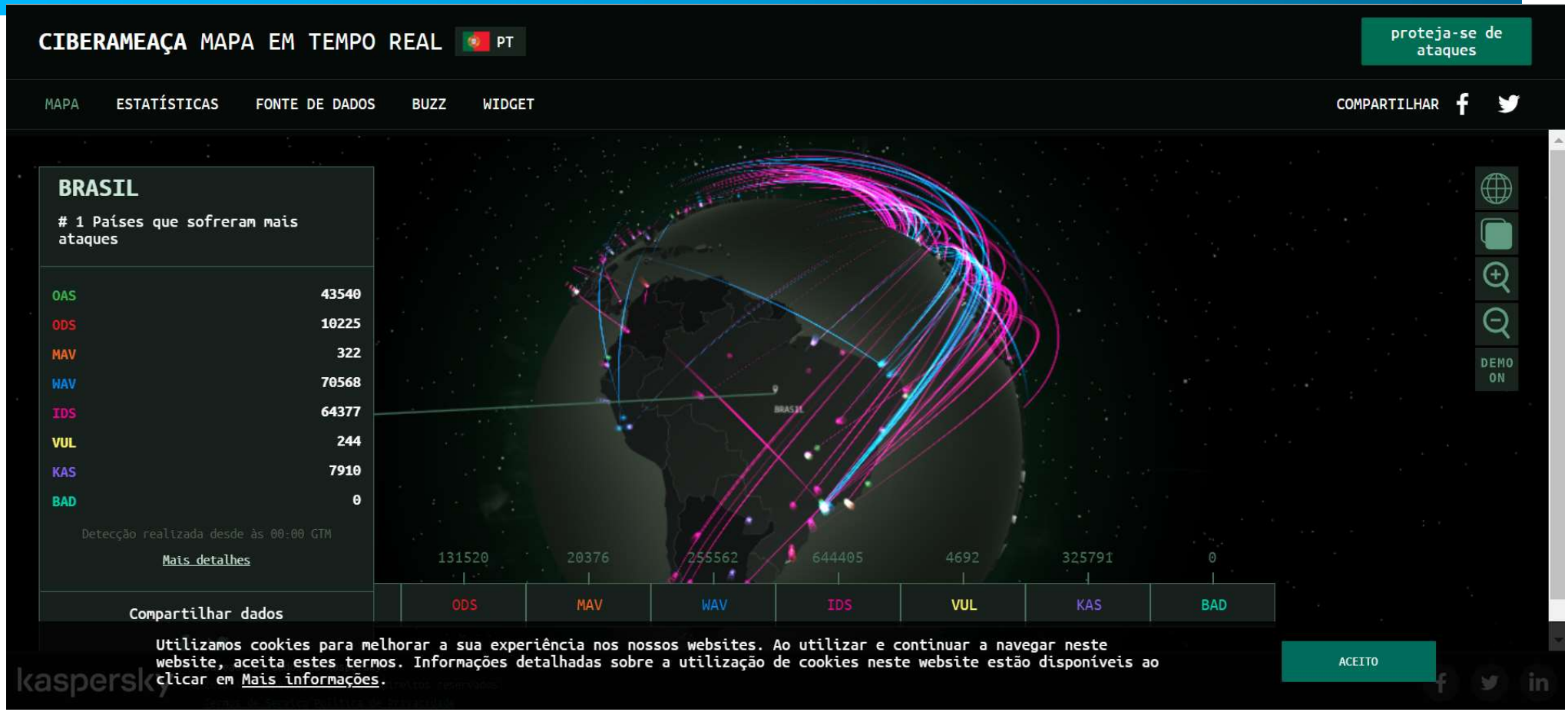
ASP.NET SignalR

- Permite desenvolver tipos de aplicações com alta frequência de atualizações a partir do servidor
  - Ex. Jogos em tempo real
  - Usado no Blazor



isec Engenharia DEIS - Instituto Superior de Engenharia de Coimbra

Exemplo: Site Dinâmico da Kaspersky



https://cybermap.kaspersky.com/pt

isec Engenharia DEIS - Instituto Superior de Engenharia de Coimbra

Programação Web

Aulas Teóricas – Capítulo 2 – 2.2

1º Semestre - 2023/2024

Departamento de Engenharia Informática e de Sistemas
Instituto Superior de Engenharia de Coimbra/Instituto Politécnico de Coimbra

Programação Web

ASP.NET Core

Departamento de Engenharia Informática e de Sistemas
Instituto Superior de Engenharia de Coimbra/Instituto Politécnico de Coimbra

ASP.NET

- Asp.NET Webforms
  - Utiliza a abordagem *Page controller*, no qual cada página tem o seu próprio controlo, i.e., ficheiro *code-behind* que processa o request
- ASP.MVC
  - Utiliza a abordagem *front controler*, no qual um controlo comum para todas as paginas processa o *request*
- …

Programação Web – 2023/2024 – LEI D+CE+PI – ISEC/IPC @JB

Model-View-Controller

- *MVC* é um *architectural pattern*
  - Desenvolvido nos anos 70 para aplicações *desktop*, inicialmente deignado como *Thing-Model-View-Editor*
  - Amplamente adotado como arquitetura para desenvolvimento de aplicações web
- Existem *frameworks* MVC para várias linguagens
  - Java (ex. Spring Framework)
  - Ruby (Ruby on Rails)
  - .NET (com introdução do ASP.NET MVC em 2007)
  - Várias *frameworks* para clientes Javascript (ex. Angular)

Programação Web – 2023/2024 – LEIDI-CE+PL – ISEC/IPC @JB

Do ASP.NET Webforms para MVC

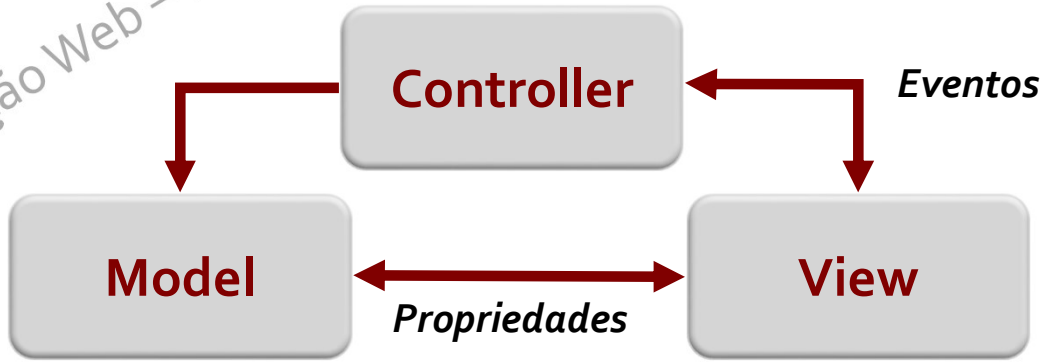
- Remoção do Code-behind para as vistas
- Remoção da *ViewState*
- Remoção do suporte ao *server-side control*
- Adição de *Model Binding*
- Adição de *Routing*
- Adição do *Razor View Engine (a partir do MVC 3)*

Programação Web (2023/2024) - LEI DTSE+PL – ISEC/IPC @ 23

isec Engenharia DEIS - Instituto Superior de Engenharia de Coimbra

Padrão MVC – *Model View Controller*

- No ASP.Net Core podem-se criar aplicações com a chamadas Razor Pages ou de acordo com o padrão MVC – iremos dar ênfase a esta 2ª abordagem
  - Na 2ª abordagem utiliza-se o padrão MVC para se fazer a separação do UI de uma aplicação em três aspetos

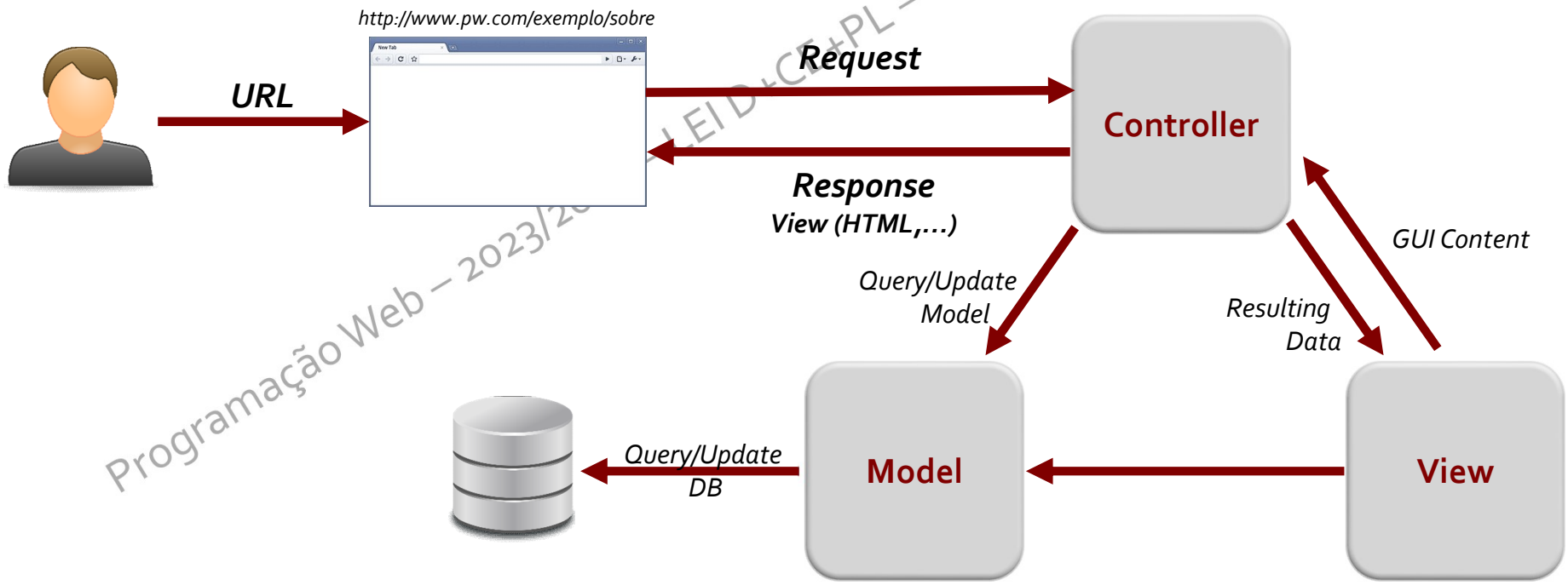


Programação Web – 2023/2024 – LEI+CE+RC – ISEC/DC @JB

isec Engenharia DEIS - Instituto Superior de Engenharia de Coimbra

ASP.NET Core MVC

- Fluxo de um pedido em ASP.NET MVC.

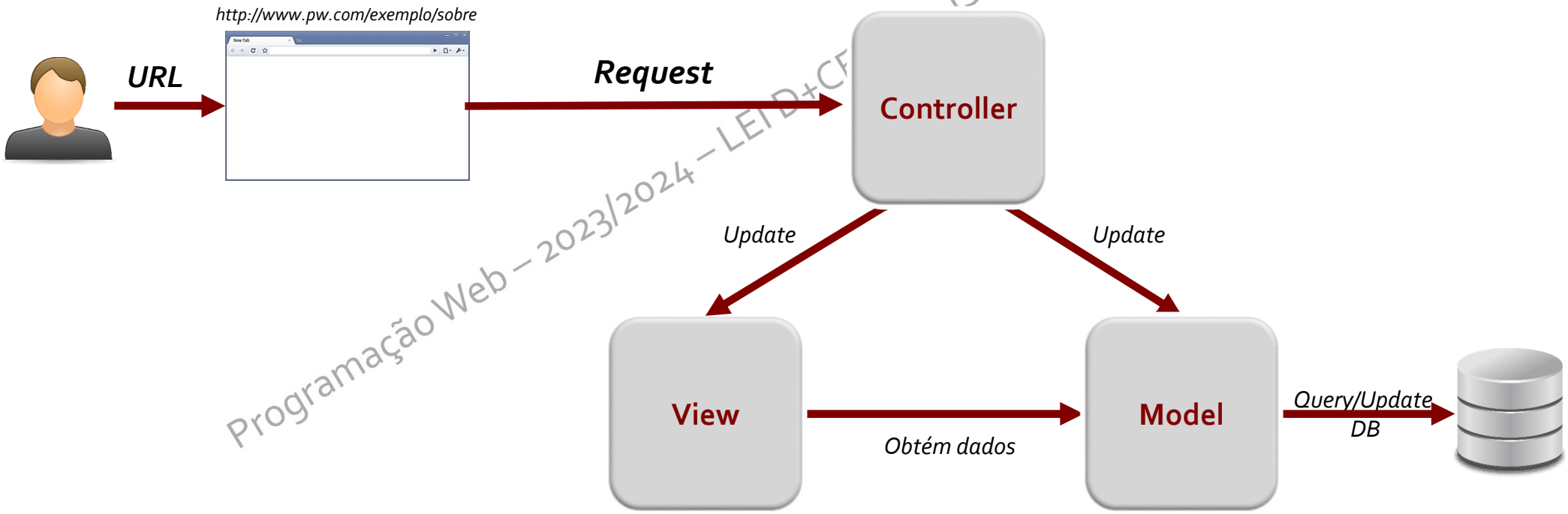


Programação Web – 2023/2024 – LEI D+CE+PL – ISEC/IPC @JB

DEIS - Instituto Superior de Engenharia de Coimbra

ASP.NET Core MVC

- Fluxo de um pedido em ASP.NET MVC.

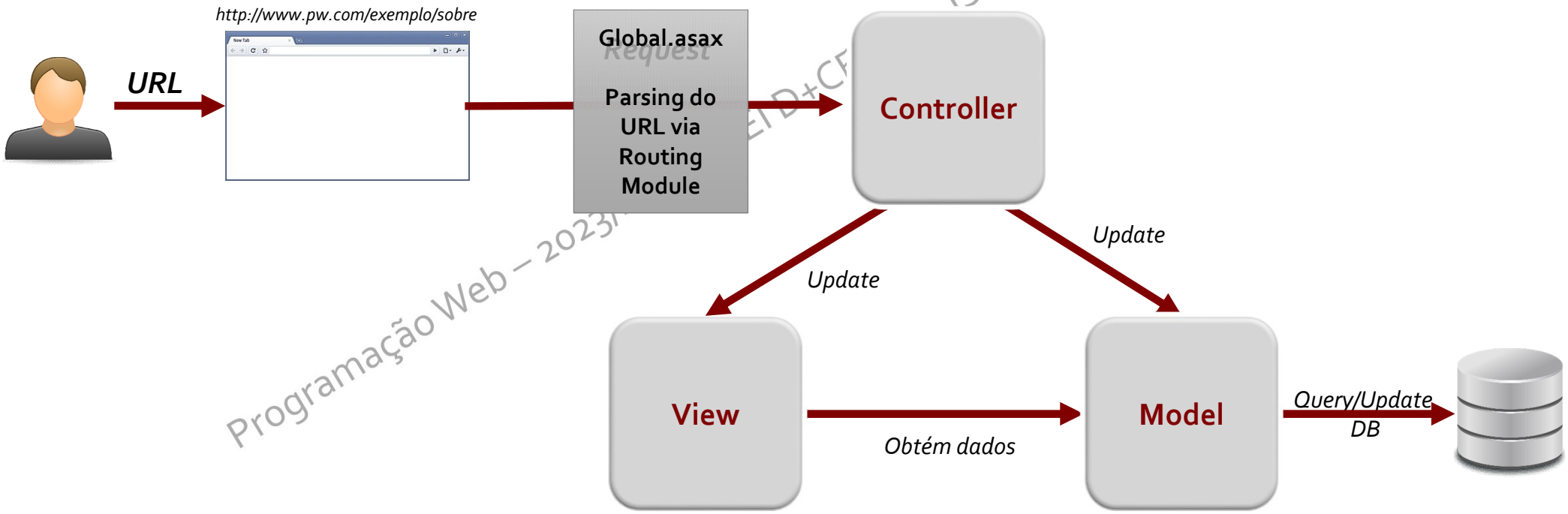


Programação Web – 2023/2024 – LEI D+CF ISEC/IPC @JB

isec Engenharia DEIS - Instituto Superior de Engenharia de Coimbra

ASP.NET Core MVC

- Fluxo de um pedido em ASP.NET MVC.



Programação Web – 2023/2024 ISEC/IPC @JB

isec Engenharia DEIS - Instituto Superior de Engenharia de Coimbra

Model-View-Controller

- Permite o desenvolvimento de aplicações mais flexíveis;
- Separação de responsabilidades entre:
  - a **vista**
  - o **modelo**
  - o **controlador**

Programação Web – 2023/2024 – LEI D+CL PL – ISEC/IPC @ 23

isec Engenharia DEIS - Instituto Superior de Engenharia de Coimbra

ASP.NET Core MVC - *Model*

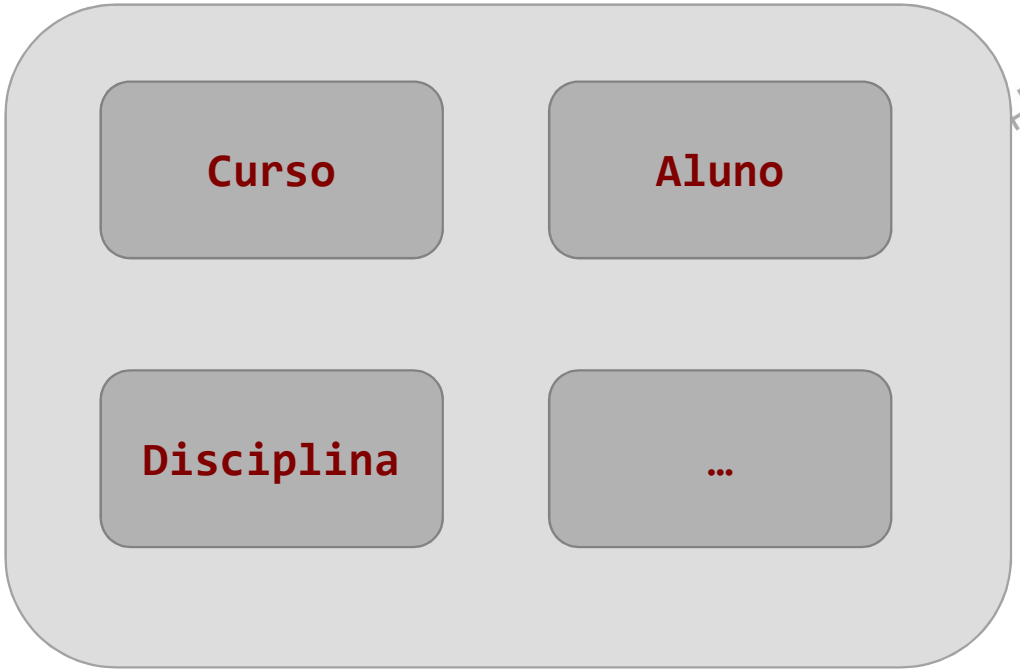
- Domínio dos dados e comportamento da aplicação, independente do UI
- Tipicamente representados por ***P**lain **O**ld **C**LR **O**bjects* (**POCOs**)
- O modelo de classes tem validação *built in* e, depende da *framework* Javascript do lado cliente utilizada (como por exemplo a knockout.js)

Programação Web – 2023/2024 – LEI D+CP+PL – ISEC/IPC @ ...

isec Engenharia DEIS - Instituto Superior de Engenharia de Coimbra

ASP.NET Core MVC - *Model*

- Model



FPL – ISEC/IPC @JB

```
namespace MVC1.Models
{
    public class Aluno
    {
        public int Id { get; set; }
        public string Nome { get; set; }
        public string Mail { get; set; }
    }
}
```

isec Engenharia DEIS - Instituto Superior de Engenharia de Coimbra

ASP.NET Core MVC – *View*

- Interface do utilizador para apresentação dos dados, isto é, representa o UI da aplicação;
- O HTML que se apresenta ao utilizador;

```
@{
    ViewBag.Title = “Aluno";
    Layout = "~/Views/Shared/_Layout.cshtml";
}

<h2>Aluno</h2>
```

Programação Web 2023/2024 – LEI D+CE+PL – ISEC/IPC @ DEIS

ASP.NET Core MVC - *Controller*

- Traduz as ações do utilizador em operações apropriadas;
- **"Cérebro" da operação**, responsável por tratar o pedido HTTP
- *Controllers* (assim como os modelos e vistas) devem ser leves e deixar os outros componentes manter separação de responsabilidades/interesses.

Programação Web – 2023/2024 – LEI D+CL PL – ISEC/IPC @ ©

ASP.NET Core

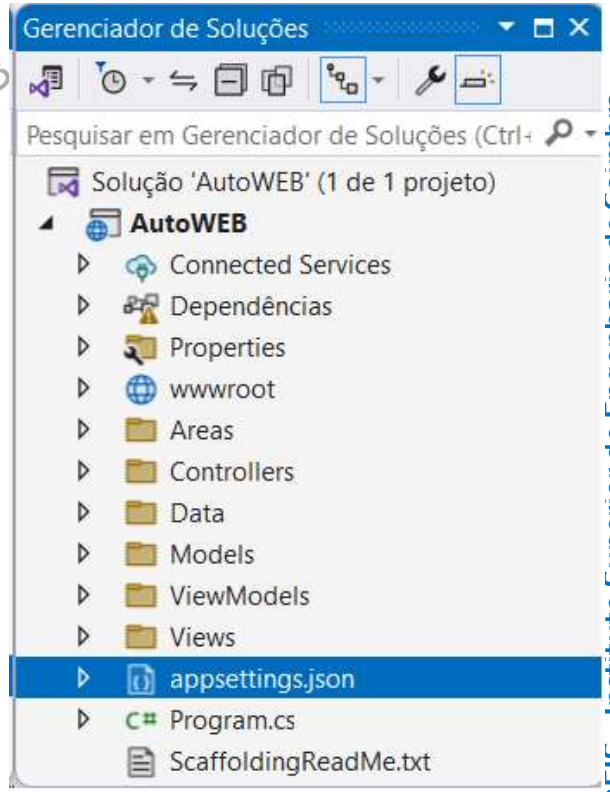
- Convenções de nomes
  - Simplificam a leitura de código entre varios elementos de uma equipa de desenvolvimento;
  - Evita que os programadores tenham de configurar e especificar elementos que podem ser inferidos por convençao:
- Exemplo:
  - Nome para a estrutura de directórios para as vistas (permite a omissão do caminho concreto quando se quer referenciar uma vista numa class controller)

Programação Web – 2023/2024 – LEIID+CE+PL – ISEC/IPC @JB

isec Engenharia DEIS - Instituto Superior de Engenharia de Coimbra

Estrutura ASP.N Core

- Estrutura de diretórios
  - *wwwroot* – Ficheiros CSS, imagens, entre outros elementos do site
  - *Data* – pode-se usar para as Migrations e o Contexto da Base de Dados
  - *Models – Classes de domínio referentes às entidades*
  - *Controllers* - Controladores
  - *Views* – Classes das views da aplicação (Razor)

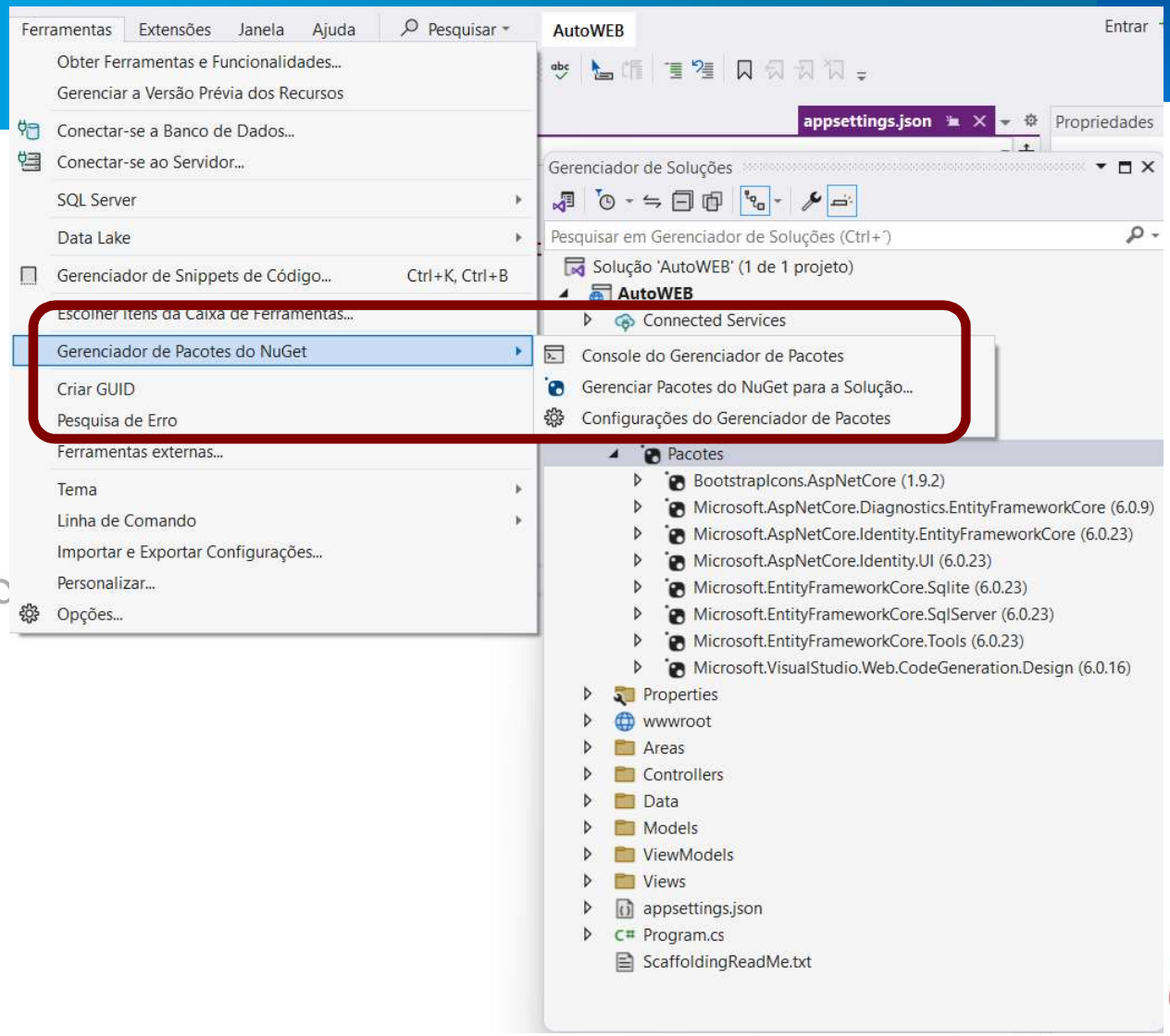


Programação Web – 2023/2024 – LEI+CE+PL – ISEC/IPC @JB

isec Engenharia DEIS - Instituto Superior de Engenharia de Coimbra

Pacotes

- Utilizado pelo *NuGet package manager*
  - Permite gerir as dependências da aplicação



Programação Web – 2023/20

isec Engenharia DEIS - Instituto Superior de Engenharia de Coimbra

Outros ficheiros de Configuração

- Program.cs
  - Utilizado na configuração e inicialização da aplicação
  - Nota: No .Net 6, só existe este ficheiro para as configurações

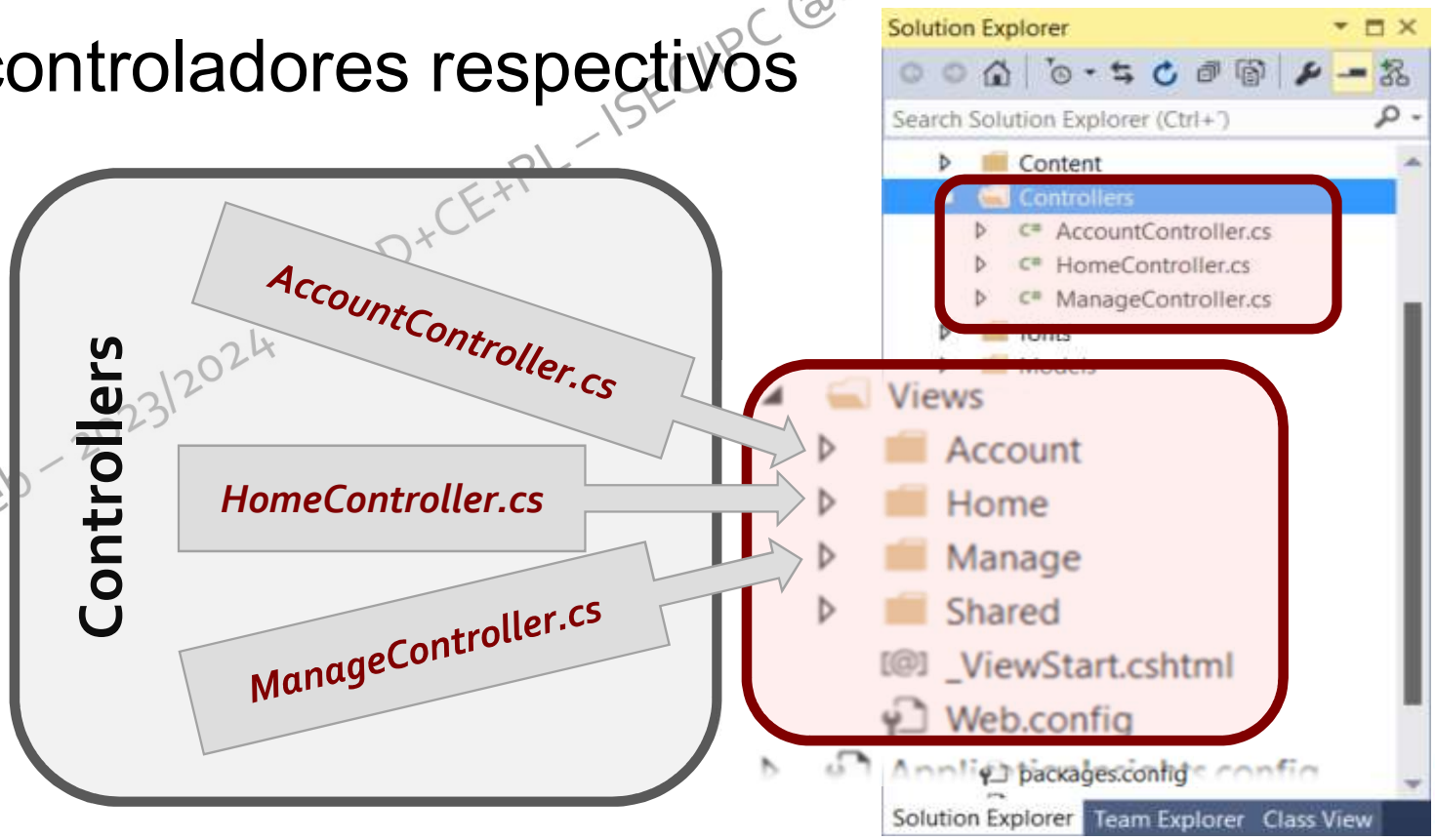
Programação Web – 2023/2024 – LEI D+CF+PL – ISEC/IPC @JB

isec Engenharia DEIS - Instituto Superior de Engenharia de Coimbra

Views

- Os ficheiros das Views são colocados em Pastas com o nome dos controladores respectivos

Convenções!

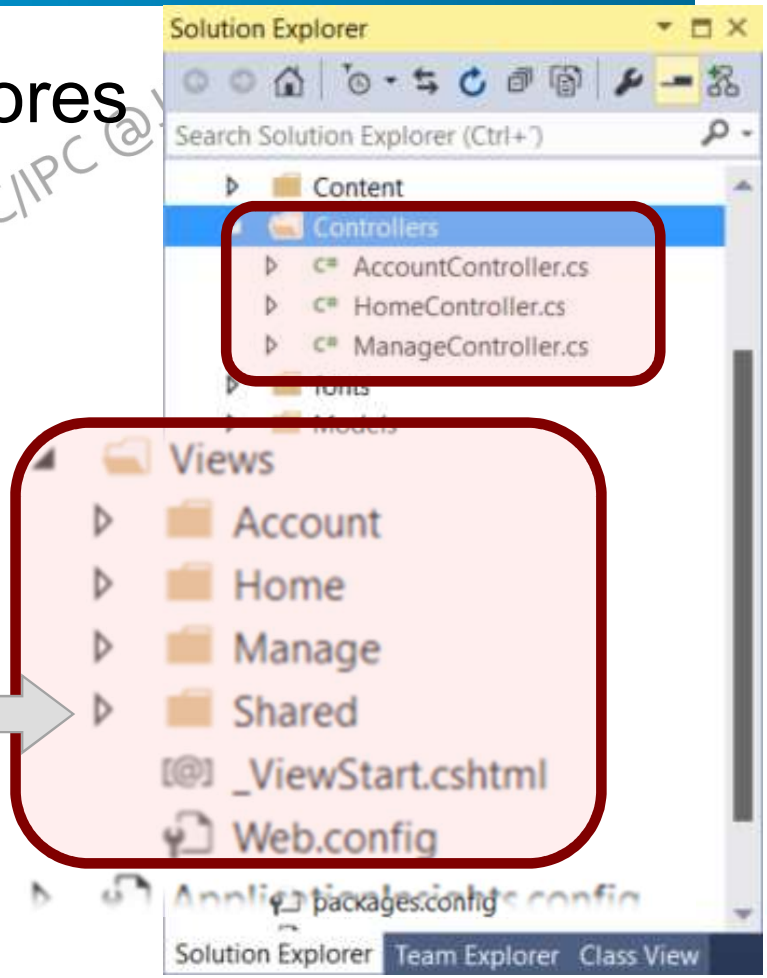


Programação Web – 2023/2024 D+CE+PL – ISEC/UPC @ 2

isec Engenharia DEIS - Instituto Superior de Engenharia de Coimbra

Views (2)

- Pastas com o nome dos controladores



Shared
Vista utilizada por todos os controladores

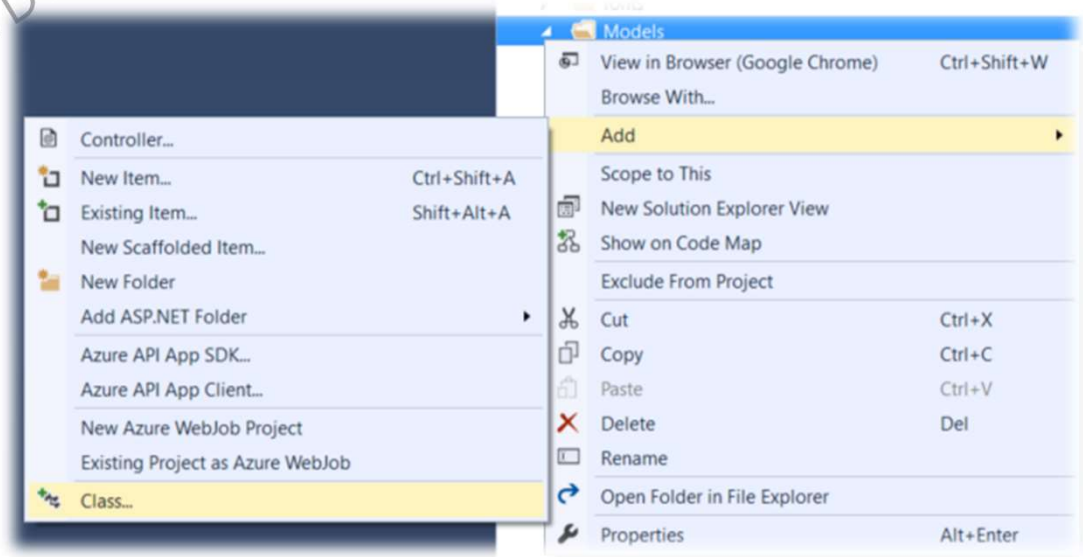
Pr – 2023/2024 – LEI D+CE+PL – ISEC/IPC @

isec Engenharia DEIS - Instituto Superior de Engenharia de Coimbra

Criação de um Model

- **Criação de Modelo**
  - Classe Aluno

```
namespace AutoWeb.Models
{
    public class Aluno
    {
        public int Id { get; set; }
        public string Nome { get; set; }
        public string Mail { get; set; }
    }
}
```

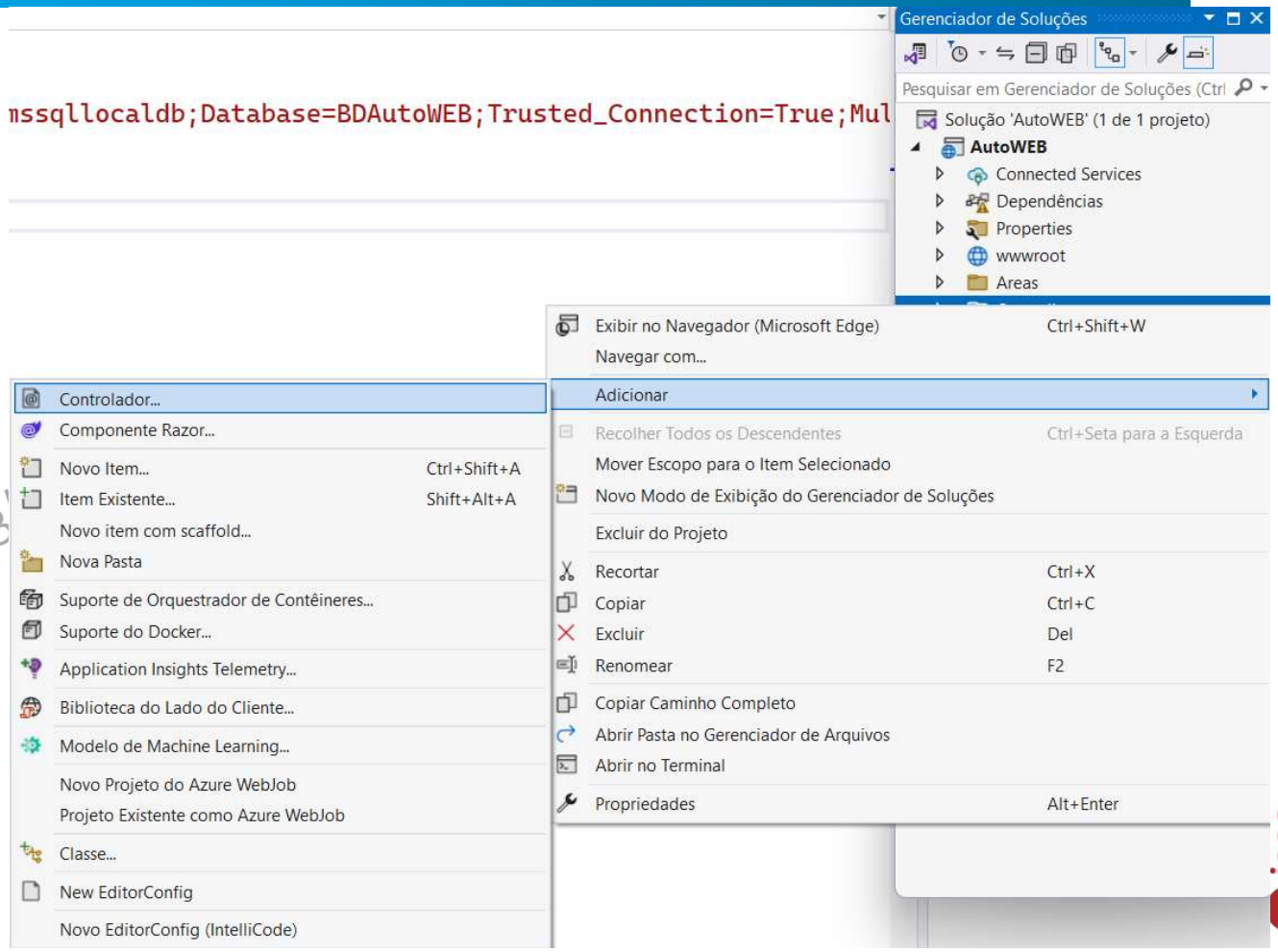


Programação Web – 2023/2024 – LEI D+CE+PL – ISEC/IPC @JB

isec Engenharia DEIS - Instituto Superior de Engenharia de Coimbra

Criação de um Controler

- Adicionar um controlador - 1

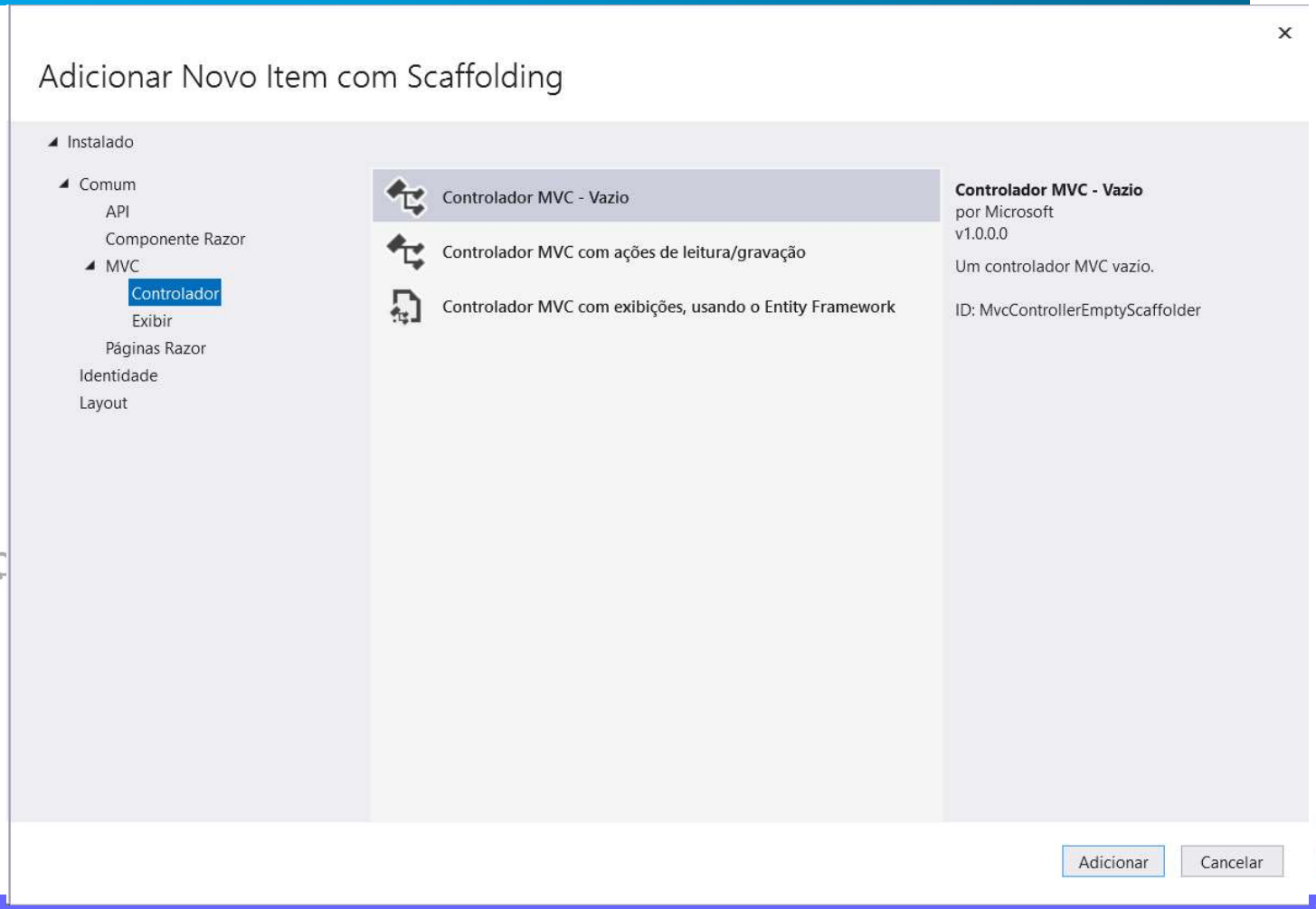


Programação Web – 2023

DEIS - Instituto Superior de Engenharia de Coimbra

Criação de um Controler

- Adicionar um controlador - 2

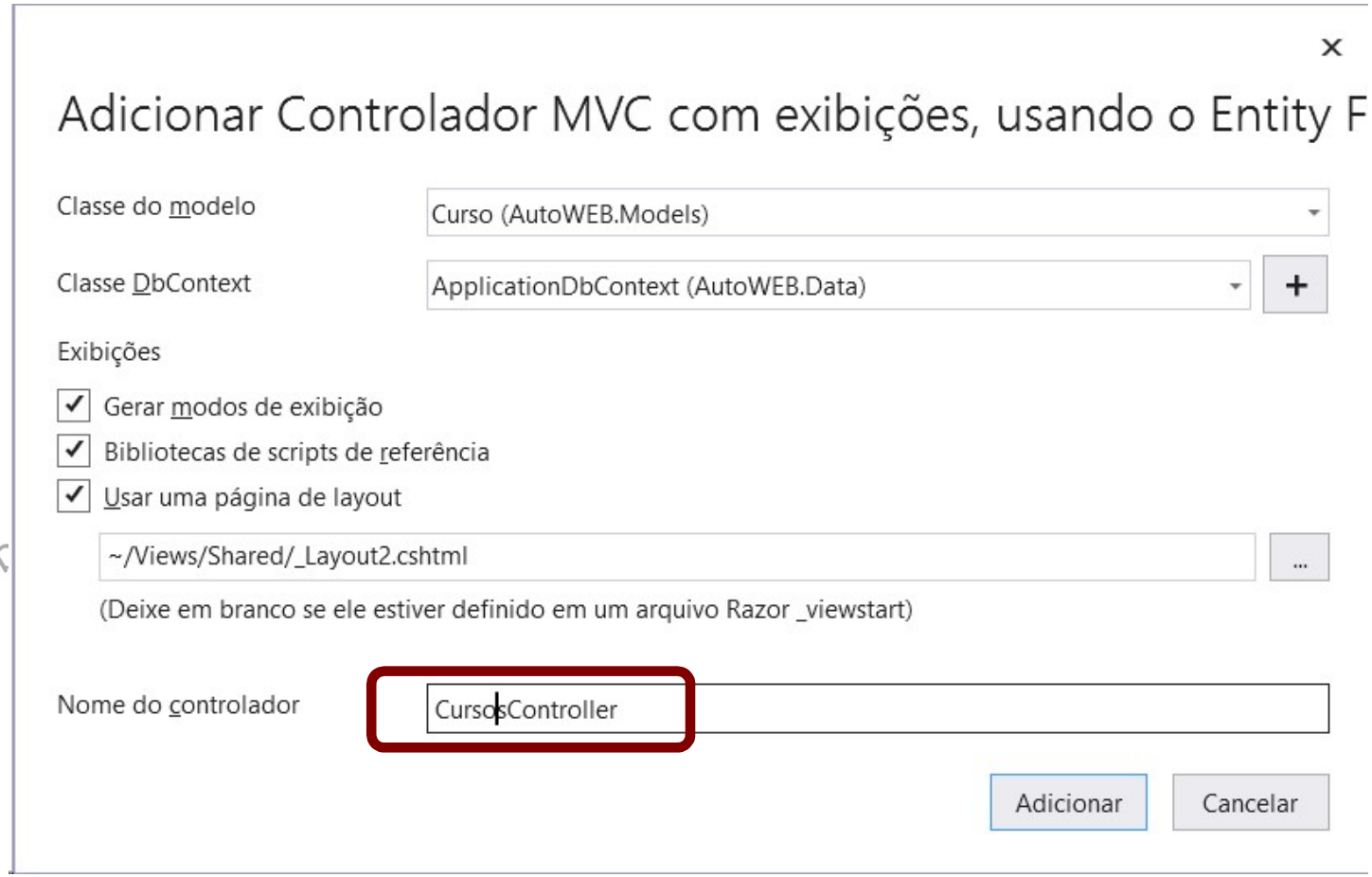


Programação Web

isec Engenharia DEIS - Instituto Superior de Engenharia de Coimbra

Criação de um Controler

- Adicionar um controlador - 3



Programação Web

isec Engenharia DEIS - Instituto Superior de Engenharia de Coimbra

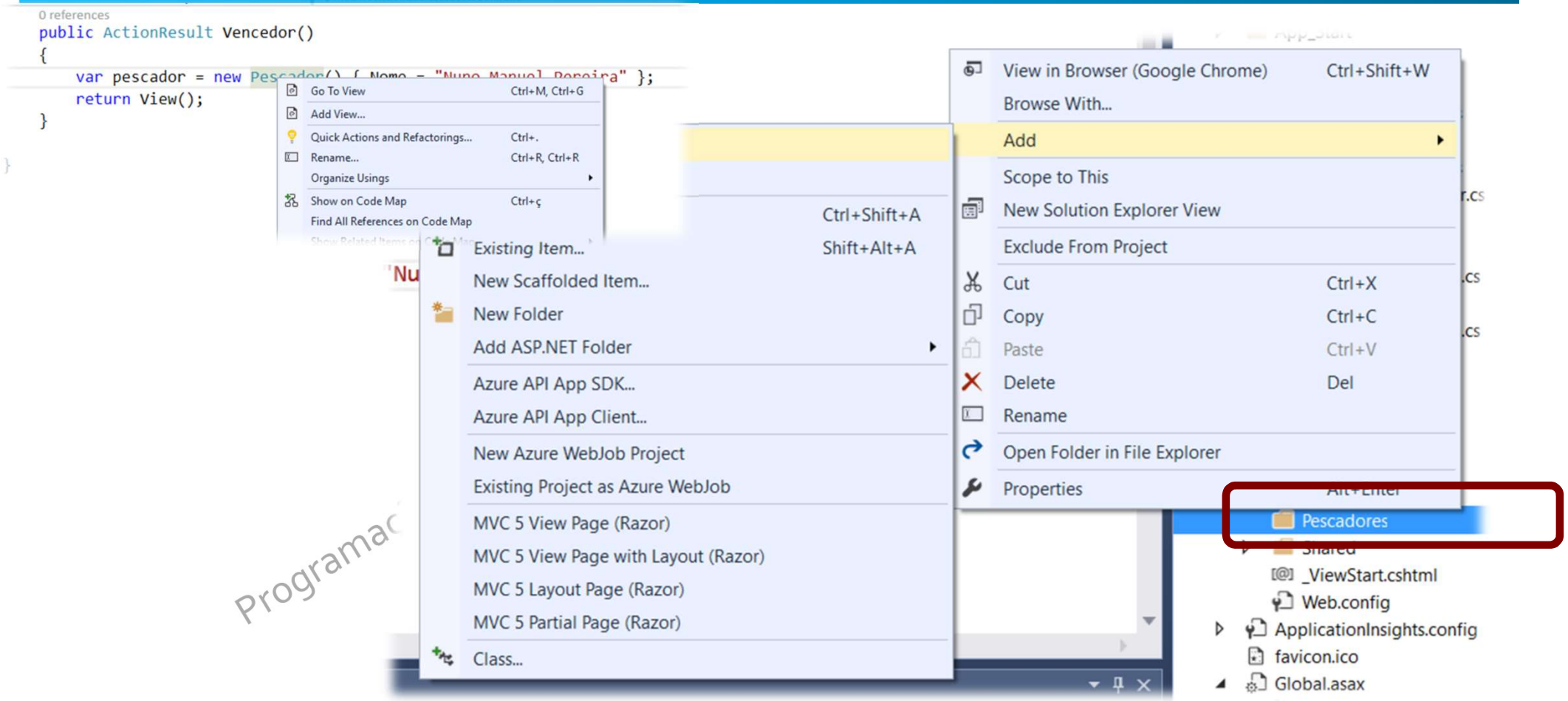
Criação do *Controller*

```
using AutoWEB.Models;
using AutoWEB.ViewModels;
using Microsoft.AspNetCore.Authorization;

namespace AutoWEB.Controllers
{
    1 referência
    public class CursosController : Controller
    {
        private readonly ApplicationDbContext _context;

        0 referências
        public CursosController(ApplicationDbContext context)
        {
            _context = context;
        }

        // GET: Cursos
        3 referências
        public async Task<IActionResult> Index(bool? disponivel)
        {
            ViewData["ListaDeCategorias"] = new SelectList(_context.Cate

            if (disponivel != null)
            {
                if (disponivel == true)
```

Programação Web – 2023/

isec Engenharia DEIS - Instituto Superior de Engenharia de Coimbra

Criação de uma View



isec Engenharia DEIS - Instituto Superior de Engenharia de Coimbra

Action Methods – Class Controller

- Todos os métodos públicos da classe *Controller* são designados como ***Action methods***

CE+PL – ISEC/IPC @ 23

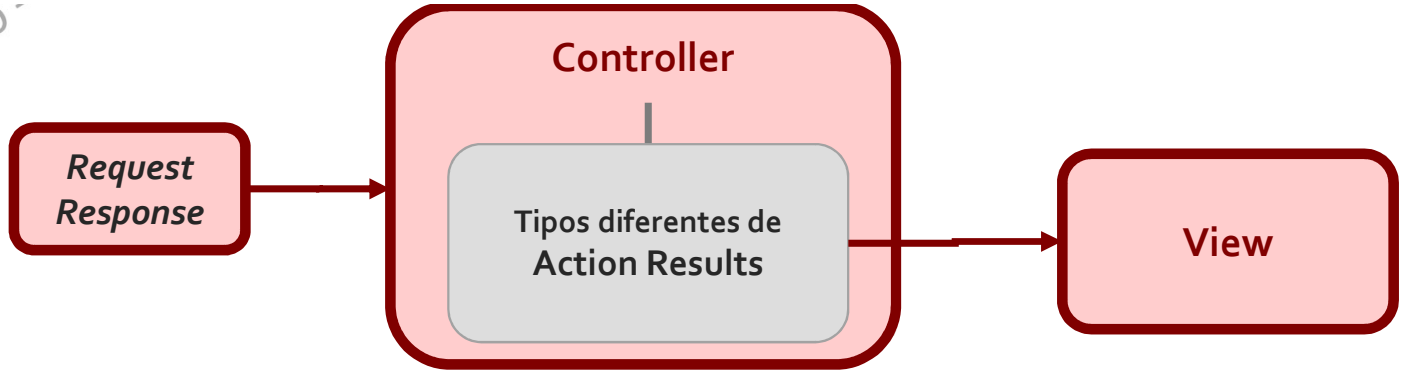
```
// POST: Cursos/Create
// To protect from overposting attacks, enable the specific properties you want to bind to.
// For more details, see http://go.microsoft.com/fwlink/?LinkId=317598.
[HttpPost]
[ValidateAntiForgeryToken]
[Authorize(Roles = "Admin")]
0 referências
public async Task<IActionResult> Create([Bind("Id,Nome,Descricao,DescricaoResumida,Requisitos,IdadeMinima,Disponivel,Preco,EmDestaque,CategoriaId")] Curso curso)
{
    ModelState.Remove(nameof(curso.categoria));

    if (ModelState.IsValid)
    {
        _context.Add(curso);
        await _context.SaveChangesAsync();
        return RedirectToAction(nameof(Index));
    }
    ViewData["ListaDeCategorias"] = new SelectList(_context.Categorias.OrderBy(c => c.Disponivel).ToList(), "Id", "Nome", curso.CategoriaId);

    return View(curso);
}
```

- Devem ser públicos
- Não podem ser *overloaded*
- *Não podem ser estáticos*

Action Methods - ActionResult

```
// POST: Cursos/Create
// To protect from overposting attacks, enable the specific properties you want to bind to.
// For more details, see http://go.microsoft.com/fwlink/?LinkId=317598.
[HttpPost]
[ValidateAntiForgeryToken]
[Authorize(Roles = "Admin")]
0 referências
public async Task<IActionResult> Create([Bind("Id,Nome,Descricao,DescricaoResumida,Requisitos,IdadeMinima,Disponivel,Preco,EmDestaque,CategoriaId")] Curso curso)
{
    ModelState.Remove(nameof(curso.categoria));

    if (ModelState.IsValid)
    {
        _context.Add(curso);
        await _context.SaveChangesAsync();
        return RedirectToAction(nameof(Index));
    }
    ViewData["ListaDeCategorias"] = new SelectList(_context.Categorias.OrderBy(c => c.Disponivel).ToList(), "Id", "Nome", curso.CategoriaId);

    return View(curso);
}
```

Retorna uma ***ActionResult usando o método View() definido na class Controller***



gramação Web -

isec Engenharia DEIS - Instituto Superior de Engenharia de Coimbra

Action Results

Action Result	Helper Method
ViewResult	View()
PartialViewResults	PartialView()
ContentResult	Content()
RedirectResult	Redirect()
RedirectToRouteResult	RedirectToAction()
JsonResult	Json()
FileResult	File()
HttpNotFountResult	HttpNotFound()
EmpyResult	

https://msdn.microsoft.com/en-us/library/system.web.mvc.actionresult(v=vs.118).aspx

GET vs POST vs PUT vs DELETE

- **GET**: Método genérico para qualquer pedido que **obtém** dados do servidor;

```
[HttpGet]
public ActionResult Edit(int id)
{

}
```

```
@Html.ActionLink("Editar", "Edit", new { id = item.Id })
```

ISEC/IPC @

GET vs POST vs PUT vs DELETE

- **POST**: Método genérico para qualquer pedido que **envia** dados ao servidor;

```
@using (Html.BeginForm("Edit", "Exemplo", FormMethod.Post))
```

```
 [HttpPost]
 public ActionResult Edit(Disciplina disciplina) {
...
}
```

2023/2024 – LPL – ISEC/IPC @ 19

Action Parameters – Parameter Binding

- As *actions* podem ter parâmetros de entrada (tipos de dados primitivos ou complexos), provenientes de:
  - *URL*
  - *Query String*
  - *Formulário de dados*
- Os parâmetros podem ser do tipo *Nullable*.

o Web – 2023/2024 – LEI D+CE+PL – ISEC/IPC @ 18

isec Engenharia DEIS - Instituto Superior de Engenharia de Coimbra

Programação Web

Aulas Teóricas – Capítulo 2 – 2.3

1º Semestre - 2023/2024

Departamento de Engenharia Informática e de Sistemas
Instituto Superior de Engenharia de Coimbra/Instituto Politécnico de Coimbra

Programação Web

Passagem de Dados, Views Razor Layouts

Departamento de Engenharia Informática e de Sistemas
Instituto Superior de Engenharia de Coimbra/Instituto Politécnico de Coimbra

Passagem de Dados

Controlador – Vista - Controlador

Programação Web – 2023/2024 – LEI D+DZ+PL – ISEC/IPC @JB

isec Engenharia DEIS - Instituto Superior de Engenharia de Coimbra

Passagem de Dados

- A passagem de dados do ***Controlador*** para a ***Vista***, e ao próximo *request*, pode ser efetuada de outras formas para além da passagem de dados através de um argumento da *View.*

```
public ActionResult Index()
        {
            var alunoTeste = new Aluno() { Nome = “Aluno de Teste" };
            return View(alunoTeste);
        }
```

Argumento na View

isec Engenharia DEIS - Instituto Superior de Engenharia de Coimbra

Passagem de Dados

Passagem de dados para views

- Dados mais complexos: ViewModel
- Dados simples
  - ***ViewData*** (ViewDataAttribute)
  - ***ViewBag***
  - ***TempData***

Passagem de Dados

Para Dados mais complexos (***ViewModel***)

- A abordagem mais robusta é especificar um tipo de modelo na view. Este modelo é comumente referido como um ***ViewModel***. É passada uma instância do tipo ***ViewModel*** para a ***view*** da ***action***

- Usar um ***ViewModel*** para passar dados para uma ***view*** permite que a ***view*** tire proveito da forte verificação de tipo. A tipagem forte (ou fortemente tipada) significa que cada variável e constante tem um tipo explicitamente definido (exemplo, *string*, *int, DateTime, etc*)
  - A validade dos tipos usados numa *view* é verificada em tempo de compilação.

Passagem de Dados

Para Dados mais simples: ViewData e ViewBag

- Pode-se usar esta abordagem para passar “pequenas quantidades” de dados para dentro e para fora de controladores e de vistas.

Passagem de dados

- A ter em conta que:
  - **ViewData**
  - **ViewBag**

Utilizados para comunicar entre o controlador e a vista correspondente, mas apenas para chamadas ao servidor.
*Torna-se **null** se ocorrer redireccionamento.*

Mecanismo para **manter estado** entre o controlador e a vista correspondente

DEIS - Instituto Superior de Engenharia de Coimbra

Passagem de dados – Outras formas

- A referenciação dos dados a passar pode ser feita por meio das propriedades ***ViewData*** ou ***ViewBag*** em controladores e views
- A propriedade ***ViewData*** é um dicionário de objetos fracamente digitados
- A propriedade ***ViewBag*** é um invólucro em torno de ***ViewData*** que fornece propriedades dinâmicas para a coleção ***ViewData*** subjacente
- ***ViewData*** e ***ViewBag*** são resolvidos dinamicamente em tempo de execução.
  - Nota: As pesquisas de chave não diferenciam maiúsculas de minúsculas para ViewData e ViewBag.

Passagem de dados – Outras formas

- ***Em Resumo:***
- ***ViewData*** *- Objeto do tipo dicionário*
  - Acessível usando *strings* como chaves
  - Necessita do *type casting* para tipos complexos
- **ViewBag** – *Objecto do tipo dinâmico*
  - Não necessita de *type casting* nem verificações de *null*
  - *Propriedades adicionadas em tempo de execução*

Passagem de dados – Outras formas

- ***TempData***
  - O ASP.NET Core expõe o ***Razor Pages TempData*** ou ***Controller TempData.*** Esta propriedade armazena dados até que sejam lidos noutra solicitação.

  - Assim, ***TempData*** é:
    - Útil para redirecionamento quando os dados são necessários para mais de uma única solicitação.
    - Implementado por provedores ***TempData*** usando *cookies* ou controlo do estado de sessão.

isec Engenharia DEIS - Instituto Superior de Engenharia de Coimbra

Passagem de dados – Outras formas

- ***Em Resumo:***

- ***TempData*** *- Objeto do tipo dicionário*

  - Permanece durante o tempo de um pedido HTTP

  - *Pode ser usado para manter dados entre redireccionamentos*

isec Engenharia DEIS - Instituto Superior de Engenharia de Coimbra

Passagem de dados: ViewData

Controller

```
// GET: Cursos/Create
[Authorize(Roles = "Admin")]
0 referências
public IActionResult Create()
{
    ViewData["ListaDeCategorias"] = new SelectList(_context.Categorias.OrderBy(c => c.Disponivel).ToList(), "Id", "Nome");

    return View();
}
```

View

```
<div class="form-group">
    <label asp-for="CategoriaId" class="control-label fw-bold"></label>
    <select asp-for="CategoriaId"
            asp-items="ViewBag.ListaDeCategorias"
            class="form-control">
    </select>
    <span asp-validation-for="CategoriaId" class="text-danger"></span>
</div>
```

Views

Views

- A view não deve executar qualquer lógica de negócio, ou interagir com a base de dados directamente.
- Deve trabalhar somente com os dados que lhe são fornecidos pelo controlador.
- A "*separation of concerns*" ajuda a manter um código limpo, testável e de mais facil manutenção.

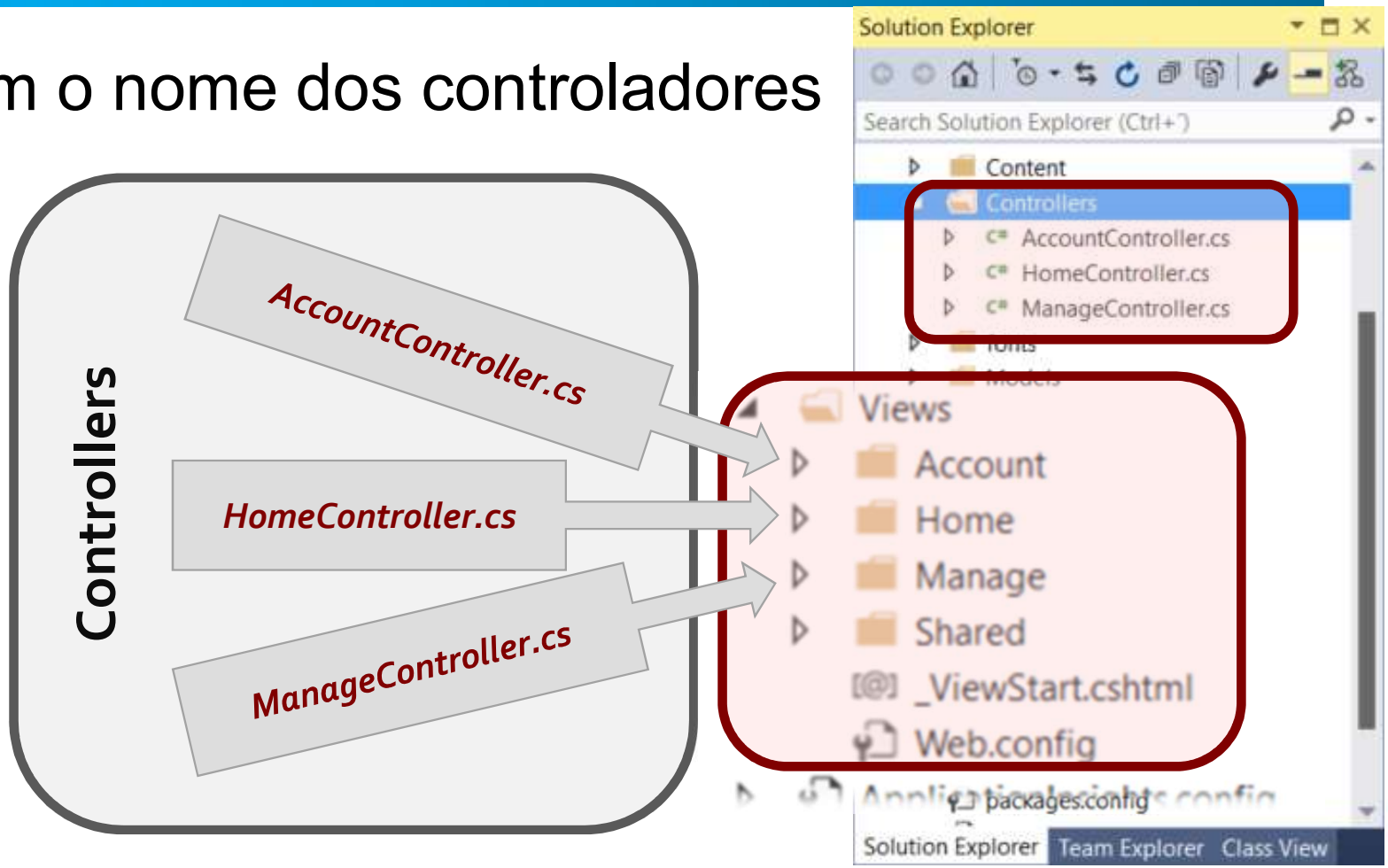
Views

- Simples
- Usam *ViewData*, *ViewModel*, *ViewBag*
- Bibliotecas JQuery incluídas
- Possibilidade de *Partial Views*
- Html helper class
- Tag Helpers class

Views - Associação com as *Actions*

- Pastas com o nome dos controladores

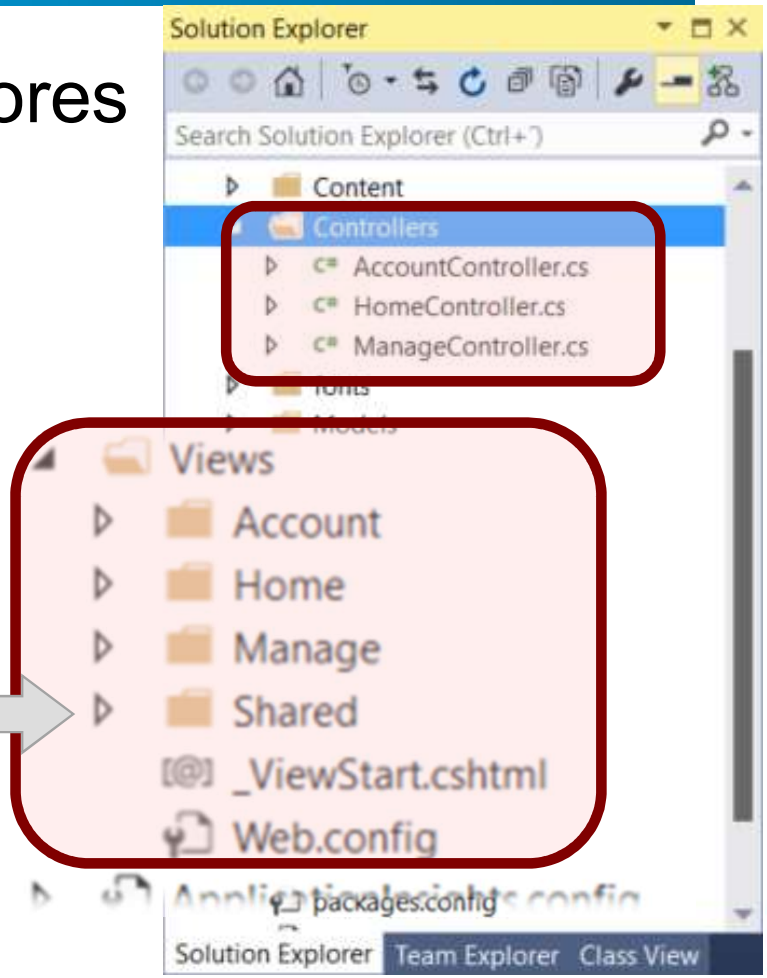
Convenções!



isec Engenharia DEIS - Instituto Superior de Engenharia de Coimbra

Views - Associação com as *Actions*

- Pastas com o nome dos controladores



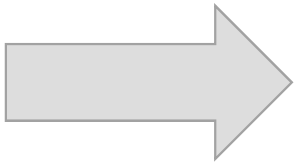
Shared

Vistas que pode ser utilizada por todos os controladores

isec Engenharia DEIS - Instituto Superior de Engenharia de Coimbra

Qual a *View* apresentada?

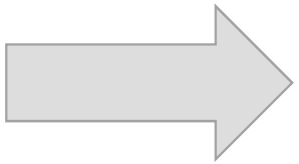
```
public ActionResult Index()
{
    return View();
}
```



Renderiza a vista

Index.cshtml

```
public ActionResult Index()
{
    return View("OutraView");
}
```



Renderiza a vista

OutraView.cshtml

View Index de Curso – Especificar modelo

```
@model IEnumerable<AutoWEB.Models.Curso>

@{
    ViewData["Title"] = "Cursos";
    Layout = "~/Views/Shared/_Layout2.cshtml";
}

<h1>
    @ViewData["Title"]
    @if (@User.IsInRole("Admin"))
    {
        <small><a class="btn btn-primary btn-sm" asp-action="Create">Adicionar</a></small>
    }
</h1>
```


isec Engenharia DEIS - Instituto Superior de Engenharia de Coimbra

View Index de UserRolesManager

```
@using AutoWEB.Models
@model IEnumerable<AutoWEB.ViewModels.UserRolesViewModel>
@{
    ViewData["Title"] = "Index";
    Layout = "~/Views/Shared/_Layout2.cshtml";
}
<h1>User Roles</h1>
```

Criação de View Model

View Models

- Nem todos os modelos de domínio mapeiam diretamente o que se pretende apresentar numa *View*
- Um View Model é um modelo que permite fornecer informação à *view sem ser de uma só classe*

isec Engenharia DEIS - Instituto Superior de Engenharia de Coimbra

Class View Model Agendamento

```
using AutoWEB.Models;
using System.ComponentModel.DataAnnotations;

namespace AutoWEB.ViewModels
{
    public class AgendamentoViewModel
    {
        [Display(Name = "Data de Início", Prompt = "yyyy-mm-dd")]
        [Required(ErrorMessage = "Indique a Data de Início!")]
        8 referências
        public DateTime? DataInicio { get; set; }

        [Display(Name = "Data de Fim", Prompt = "yyyy-mm-dd")]
        [Required(ErrorMessage = "Indique a Data de Fim!")]
        8 referências
        public DateTime? DataFim { get; set; }

        [Display(Name = "Tipo de Aula", Prompt = "Escolha o tipo de aula que pretende")]
        4 referências
        public int TipoDeAulaId { get; set; }
    }
}
```

View Agendamento - Pedido

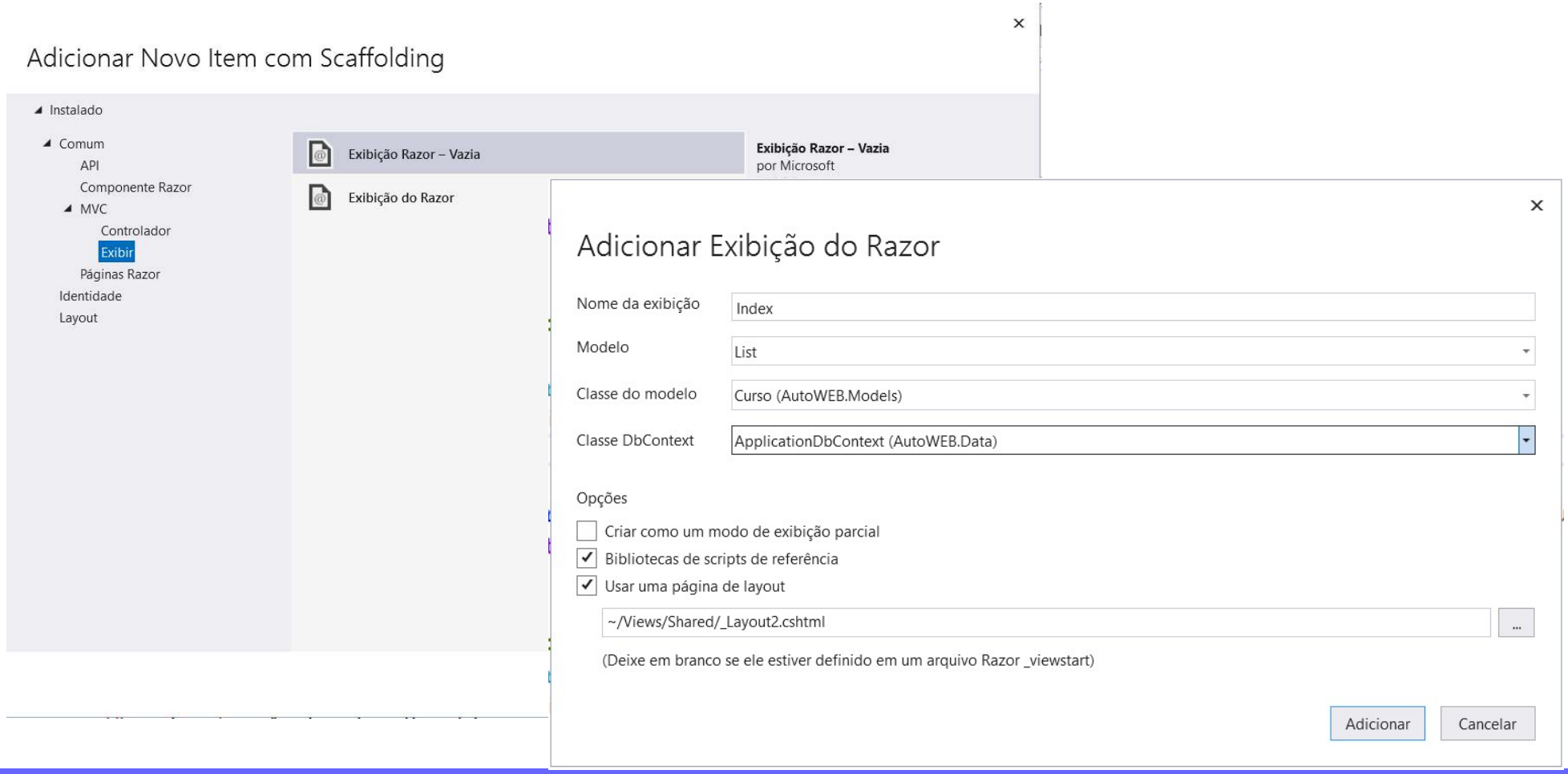
```
@model AutoWEB.ViewModels.AgendamentoViewModel
@{
    Layout = "~/Views/Shared/_Layout2.cshtml";
}
<h1>Agendamento - Pedido</h1>
<hr />
@if(User.Identity.IsAuthenticated){
<div class="row">
    <div class="col-md-4">
        <form asp-action="Calcular">
            <div asp-validation-summary="All" class="text-danger"></div>
            <div class="form-group">
                <label asp-for="TipoDeAulaId" class="control-label fw-bold"></label>
                <select asp-for="TipoDeAulaId" class="form-control" asp-items="ViewBag.TipoDeAulaId"></select>
            </div>

            <div class="form-group col-md-6">
                <label asp-for="DataInicio" class="control-label fw-bold"></label>
                <input asp-for="DataInicio" class="form-control" />
                <span asp-validation-for="DataInicio" class="text-danger"></span>
            </div>
            <div class="form-group col-md-6">
                <label asp-for="DataFim" class="control-label fw-bold"></label>
                <input asp-for="DataFim" class="form-control" />
            </div>
            <div class="col-md-12">
                <span asp-validation-for="DataFim" class="text-danger"></span>
            </div>
           <br/>
            <div class="form-group">
                <input type="submit" value="Calcular" class="btn btn-success" /> |
                    <a class="btn btn-primary" asp-action="OsMeusAgendamentos">Voltar à Listagem</a>
            </div>
            <br />
        </form>
    </div>
</div>
}
else
{
    <p class="alert alert-warning">
        Só utilizadores autenticados é que
        podem efetuar agendamentos. Por favor, autentique-se!
    </p>
}
```


isec Engenharia DEIS - Instituto Superior de Engenharia de Coimbra

Criação de Views

Templates

Criação de uma *View Template*



isec Engenharia DEIS - Instituto Superior de Engenharia de Coimbra

Razor View Engine

Razor

- Introduzido com o ASP.NET MVC 3
- Sintaxe simplificada
- Minimiza a quantidade de sintaxe e caracteres extra
- Codifica automaticamente qualquer *output* enviado através do @ de forma a prevenir ataques *cross site scripting*
- Razor suporta expressões e blocos de código
  - Razor View Engine

Razor – Carater @

Informa o Razor View Engine que os carateres seguintes são código, não HTML

```
@{
    Layout = "~/Views/Shared/_Layout.cshtml";
}
```


isec Engenharia DEIS - Instituto Superior de Engenharia de Coimbra

Caracter @

```
@{ var mensagem = "Programação Web"; }

<p>O valor da Mensagem é: @mensagem</p>

@{
    var boasVindas= "Bem vindo ao Razor!";
    var diaSemana= DateTime.Now.DayOfWeek;
    var mensagem = boasVindas + " Hoje é " + diaSemana;
}
```

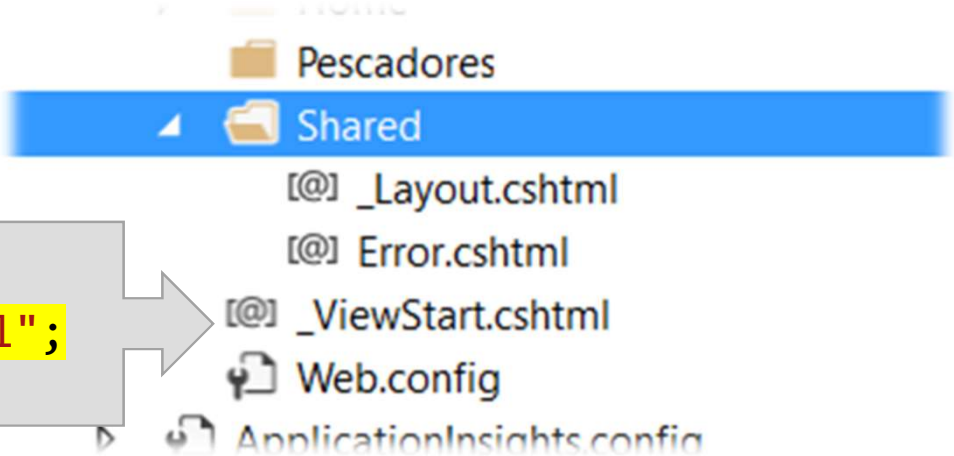
Razor – Algumas Regras

- Blocos de código estão delimitados com @{ ... }
- Expressões *Inline* (variáveis e funções) iniciam com @
- Declarações de código terminam com ;
- Variáveis são declaradas com a palavra chave *var*
- Strings devem estar entre aspas
- Código C# é *case sensitive*
- Ficheiros Razor tem extensão .cshtml

Razor

Informa o Razor View Engine que os carateres seguintes são código, não HTML

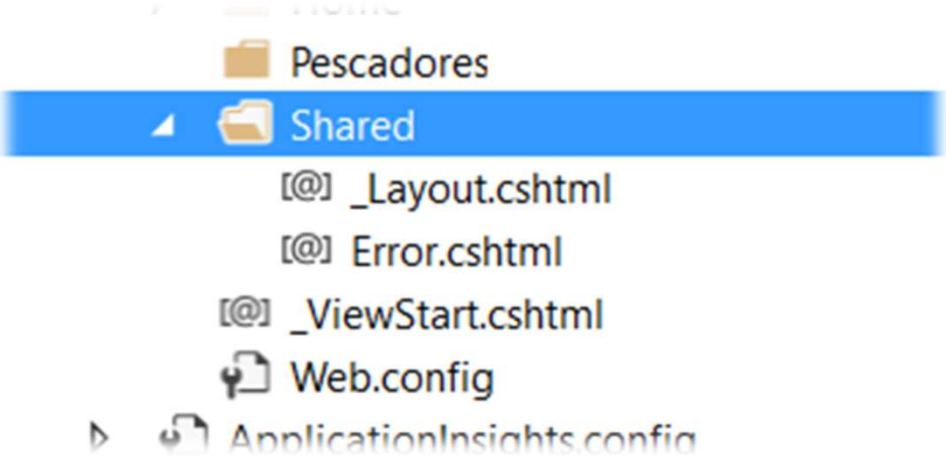
```
@{
    Layout = "~/Views/Shared/_Layout.cshtml";
}
```



Layouts

Layout com Razor

- _Layout.cshtml
  - Define o que se quer apresentar em todas as páginas da aplicação
  - Recorre a métodos herdados para especificar áreas de conteúdo
    - **RenderBody**
    - **RenderSection**



_Layout.cshtml

- A *view _Layout* deve ser especificada na pasta *Shared* existente na secção Views, permitindo assim definir o que se pretende que apareça em qualquer página da aplicação
- Tem o corpo do documento HTML, link de navegação, menus…

RenderBody

- RenderBody permite especificar onde será apresentado o conteúdo individual das *views*

- Obrigatório a sua chamada numa vista layout

_Layout.cshtml

```
<html>
<head>
    <meta charset="utf-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0"
    <title>@ViewBag.Title - My ASP.NET Application</title>
    @Styles.Render("~/Content/css")
    @Scripts.Render("~/bundles/modernizr")
</head>
<body>
    <div class="navbar navbar-inverse navbar-fixed-top">
        <div class="container">
            <div class="navbar-header">
                <button type="button" class="navbar-toggle" data-toggle="
                    <span class="icon-bar"></span>
                    <span class="icon-bar"></span>
                    <span class="icon-bar"></span>
```


isec Engenharia DEIS - Instituto Superior de Engenharia de Coimbra

_Layout.cshtml

```
                <ul class="nav navbar-nav">
                    <li>@Html.ActionLink("Home", "Index", "Home")
                    <li>@Html.ActionLink("About", "About", "Home"
                    <li>@Html.ActionLink("Contact", "Contact", "H
                </ul>
            </div>
        </div>
    </div>
    <div class="container body-content">
        @RenderBody()
        <hr />
        <footer>
            <p>&copy; @DateTime.Now.Year - My ASP.NET Application
        </footer>
    </div>
```


isec Engenharia DEIS - Instituto Superior de Engenharia de Coimbra

Aplicação do *layout* nas *Views*

```
@{
    Layout = "~/Views/Shared/_Layout.cshtml";
}
```


isec Engenharia DEIS - Instituto Superior de Engenharia de Coimbra

RenderSection

- Chamada opcional numa *Layout View*
- Podem existir uma ou mais *rendersection*
- Fornece uma *content view* como a view Index, permitindo aplicar conteúdo em outras secções da página
- A secção necessita de ter um nome e estar definida no ficheiro *content view*
- É possivel que a secção exista apenas em algumas *views*

isec Engenharia DEIS - Instituto Superior de Engenharia de Coimbra

Partial Views

- Permite especificar código Razor e Tag Helpers ou HTML Helpers num ficheiro para que depois possa ser reutilizado em várias vistas;

- Pode ser usado simplesmente para simplificar uma vista;

- Por convenção, deve-se especificar no prefixo do nome, o caracter _

isec Engenharia DEIS - Instituto Superior de Engenharia de Coimbra

Partial Views

- Para *renderizar* uma vista parcial, deve-se recorrer ao *html helper* Html.Partial
  - Deve ser passado o nome da vista parcial e o modelo que necessita

```
@foreach (var item in Model)
{
    @Html.Partial("_Rever",item)
}
```

Partial Views

- Normalmente, especifica-se uma vista parcial na pasta onde esta irá ser utilizada.

- Caso seja utilizada por várias vistas, deverá ser colocada na pasta *shared*

Tag Helpers

- **HTML Helpers**
  - Nas views e a partir do ASP.Net MVC utilizavam-se os **HTML Helpers** com o objetivo de facilitar a criação de views.
    - Os HTML Helpers podem ser invocados a partir da propriedade **Html** de uma view

Tag Helpers

- O processamento por parte do ***Razor*** da maioria dos ***HTML Helpers*** origina a renderização de tags ***HTML***

  - Por exemplo, a renderização pelo ***Razor*** do ***Html.TextBox*** resultaria em:

@Html.TextBox("nome")	**<input id="nome" name="nome" type="text" value="" />**

isec Engenharia DEIS - Instituto Superior de Engenharia de Coimbra

Tag Helpers

- ***Tag Helpers*** - são um novo recurso disponível no ASP.Net Core
  - Continua a ser possível a utilização de *Html.Helpers*

- Os ***Tag Helpers*** possibilitam a obtenção dos mesmos resultados escrevendo controles que interagem com Models, Views, etc
  - A sua sintaxe é mais próxima da do *HTML* (com atributos em seus elementos)

Tag Helpers

- A renderização pelo ***RAZOR*** de uma ***TextBox*** utilizando o ***Tag Helper*** <input … > resultaria em:

<input type="text" asp-for="nome" />	**<input id="nome" name="nome" type="text" value="" />**

isec Engenharia DEIS - Instituto Superior de Engenharia de Coimbra

Tag Helpers

- ***Tag Helpers***
  - A sintaxe dos ***Tag Helpers*** é semelhante à sintaxe do ***HTML***, elementos e atributos, mas é processado pelo ***Razor*** no servidor.
  - Os ***TagHelpers*** são uma alternativa aos ***Html Helpers***, mas disponibilizam algumas funcionalidades difíceis ou mesmo impossíveis de se obter com esses outros métodos
  - Cada ***Tag Helper*** tem um comportamento diferente e diferentes opções.

Tag Helpers

- ***Tag Helpers***

- A utilização dos ***Tag Helpers*** é disponibilizada quando se cria um projecto no *MS Visual Studio* mas também podem ser adicionados de *assembly/namespace* recorrendo-se à diretiva **@addTagHelper** nos ficheiros **cshtml** das views

@addTagHelper "*, Microsoft.AspNet.Mvc.TagHelpers"

isec Engenharia DEIS - Instituto Superior de Engenharia de Coimbra

Tag Helpers

- A utilização de ***Tag Helpers*** em formulários possibilita a criação de código de views mais limpo do que quando se utilizam os *Html Helpers*.
- Esta diferença pode ser percepcionada no exemplo abaixo onde para a criação de uma textbox se utilizam com o mesmo objectivo e base na propriedade ***Nome*** de um ***Model***, um ***Html Helper Html.EditorFor*** e um ***Tag Helper input***

Tag Helpers

- Os ***Tag Helper*** permitem adicionar atributos *HTML* mais facilmente à tag input os quais serão incluídos no *HTML* gerado
  - A inclusão da classe ***'form-control'*** é natural e de fácil entendimento se comparada com a sintaxe do *Html Helper* ***EditorFor***
  - O facto de o ***Intellisense*** do *VS* estar disponível para os *Tag Helpers* também facilita a sua utilização

Tag Helpers

- A geração do HTML para o elemento Form utilizando-se TagHelper é também mais simples e de mais fácil utilização como se pode ver no exemplo
  - Basta especificar o *Controller* e a *Action* que tratarão o formulário
  - Por default também é gerado automaticamente o *token anti-forgery* mas esta possibilidade pode ser desabilitada

Tag Helpers

- ***Tag Helpers*** habitualmente utilizados em formulários
  - **Input Tag Helper**
  - **Text Area Tag Helper**
  - **Validation Tag Helper**
  - **Label Tag Helper**
  - **Select Tag Helper**
- É possível aos programadores criarem os seus próprios **TagHelpers** e adicioná-los aos já existentes no ASP .NET Core

Programação Web

Aulas Teóricas – Capítulo 2 – 2.4

1º Semestre - 2023/2024

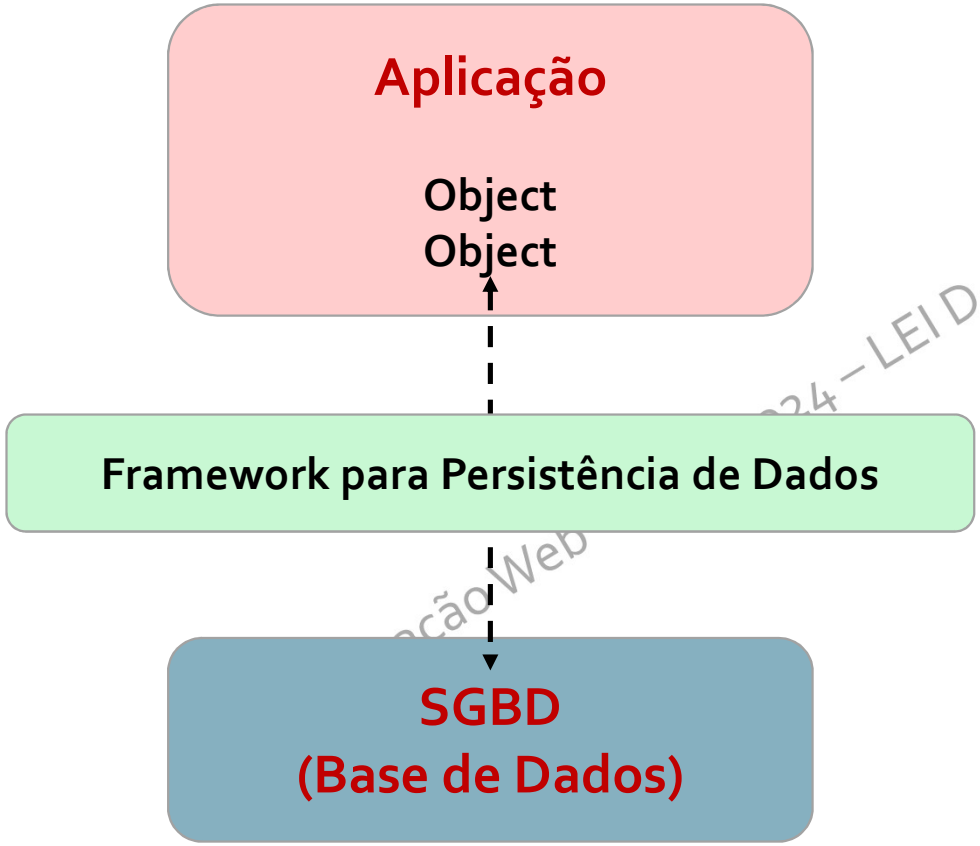
Departamento de Engenharia Informática e de Sistemas
Instituto Superior de Engenharia de Coimbra/Instituto Politécnico de Coimbra

Programação Web

Manipulação de Dados em ASP.NET Core

Departamento de Engenharia Informática e de Sistemas
Instituto Superior de Engenharia de Coimbra/Instituto Politécnico de Coimbra

Entity Framework Core (EF)



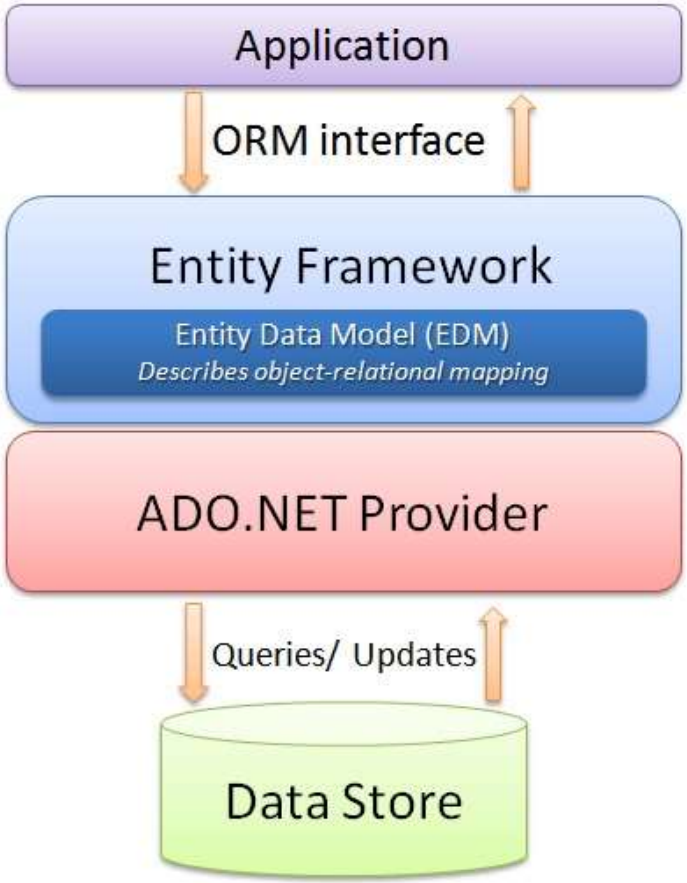
Entity Framework Core

ção Web 024 – LEI D+CE+PL – ISEC/IPC @JB

isec Engenharia DEIS - Instituto Superior de Engenharia de Coimbra

Entity Framework Core (EF)

- *Object Relational Mapper (ORM)*
- *Open Source*
- Aumentar a produtividade do programador, reduzindo tarefas redundantes para persistência de dados



Programação Web – 2023/2024 – LEI D+DE+PL – ISEC/IPC

isec Engenharia DEIS - Instituto Superior de Engenharia de Coimbra

Entity Framework Core (EF)

- **Entity Framework Core (EF Core)** é uma versão leve, extensível e multiplataforma do Entity Framework
- O EF Core introduz muitas melhorias e novos recursos quando comparado com as versões anteriores
- O EF Core mantém a experiência do programadores habituados a usar versões antigas do EF e a maioria das APIs de alto nível permanecem as mesmas

Programação Web – 2023/2024 – LEID+CE PL – ISEC/PC @ JB

isec Engenharia DEIS - Instituto Superior de Engenharia de Coimbra

Entity Framework Core (EF)

- Ao mesmo tempo, o ***EF Core*** é construído sobre um conjunto completamente novo de componentes principais. Isso significa que o EF Core não herda automaticamente todos os recursos de anteriores versões do EF
  - Alguns desses recursos serão exibidos em lançamentos futuros, outros recursos menos usados não serão implementados no ***EF Core***
- O novo núcleo permitiu adicionar alguns recursos ao ***EF Core*** que não serão implementados nas versões anteriores

Programação Web – 2023/2024 – LEI D+CE+PL – ISEC/DEIS – @JB

DEIS - Instituto Superior de Engenharia de Coimbra

Entity Framework Core

- Formas para especificar o modelo de dados

  1. ***Code First***
     - Parte-se das classes do domínio e a ***EF*** gera a Base de Dados
     - Iremos dar particular atenção à utilização de *Code First*
  2. ***Database First***
     - Parte-se da Base de Dados para obter as classes de domínio

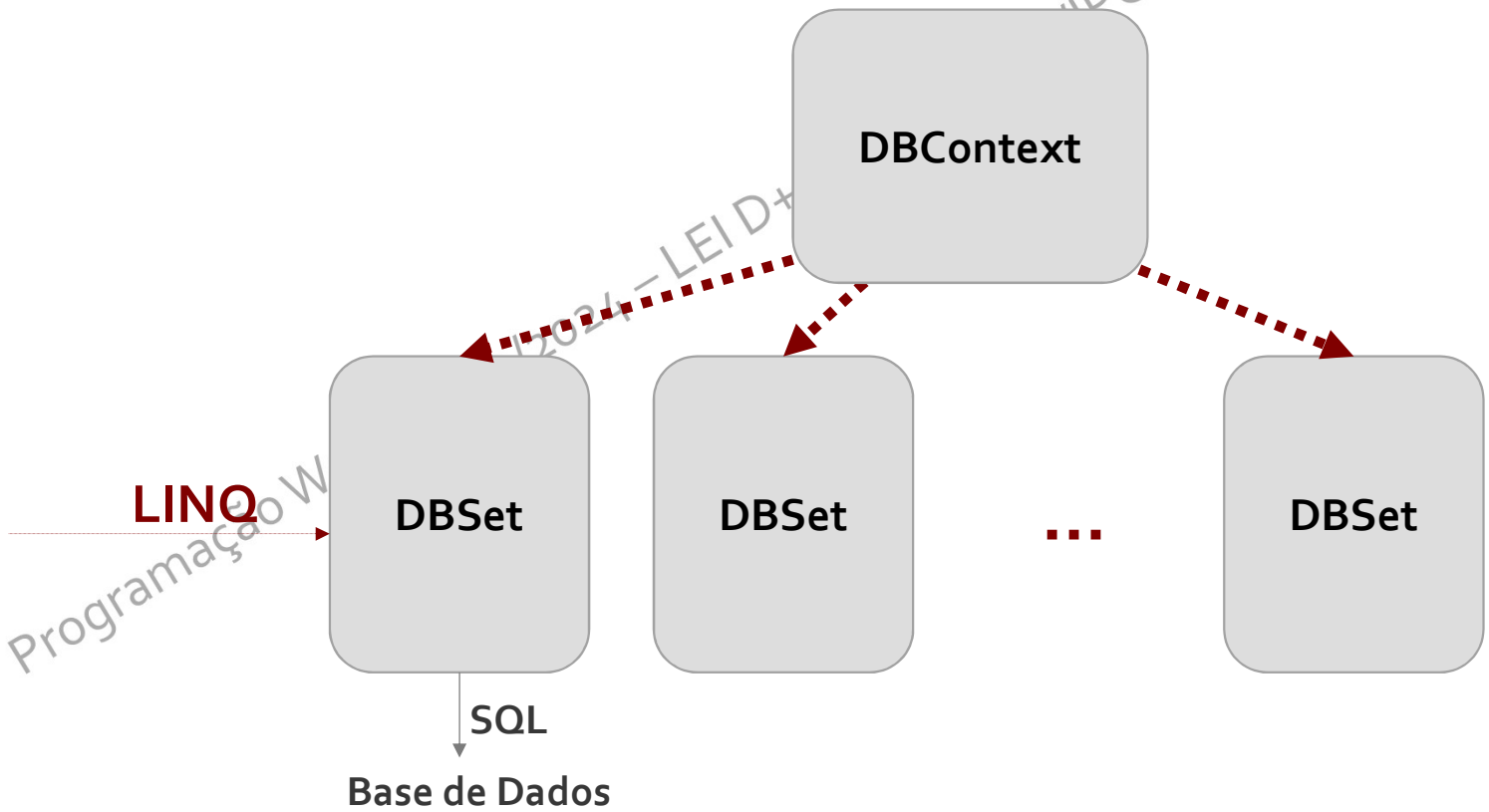
- Model First - Esta abordagem era possível com versões anteriores do ASP.Net, se bem que já na altura pouco suportada pela MS mas actualmente está descontinuada no Core

Programação Web – 2023/2024 – LEI D+CET PL – ISEC/IPC @ JS

isec Engenharia DEIS - Instituto Superior de Engenharia de Coimbra

Entity Framework

1. *Code First*



Programação Web (2023/2024 – LEI DPC @JB

isec Engenharia DEIS - Instituto Superior de Engenharia de Coimbra

Entity Framework – Code First

Etapas a seguir para a utilização da ***Entity Frameworke em CodeFirst***

1. Instalação da ***Entity Framework***
2. *Definição do* ***Modelo de Domínio***
3. Criação do ***Contexto*** (classe que herda da ***DbContext*** e que é uma abstração da base de dados usada no acesso aos dados
4. Registar no método ***Services*** de ***Program.cs*** o ***DBContext*** como um serviço
5. Especificação da ***Connection String*** no ficheiro ***appsettings.json***

Programação Web – 2022/2024 – LEI D+CE+PL – ISEC/IPC @ JB

Especificação das Entidades a Utilizar

2. *Definição do* ***Modelo de Domínio***

D+CE+PL - ISEC/IPC @JB

Na pasta *Models* crie as classes que representem as entidade de domínio
Por exemplo:

Criação do Modelo Curso

```
using System.ComponentModel.DataAnnotations;
using System.ComponentModel.DataAnnotations.Schema;
namespace AutoWEB.Models
{
    26 referências
    public class Curso
    {
        15 referências
        public int Id { get; set; }
        [Display(Name = "Nome", Prompt = "Introduza o Nome do curso!")]
        [Required(ErrorMessage = "Indique o nome do curso!")]
        19 referências
        public string Nome { get; set; }
        [Display(Name = "Descrição", Prompt = "Introduza a Descrição do curso!")]
        [Required(ErrorMessage = "Descreva o curso!")]
        11 referências
        public string Descricao { get; set; }
        [Display(Name = "Descrição Resumida", Prompt = "Introduza a Descrição Resumida do curso!")]
        [Required(ErrorMessage = "Descreva resumidamente o curso!")]
        17 referências
        public string DescricaoResumida { get; set; }
        [Display(Name = "Requisitos", Prompt = "Introduza os Requisitos do curso!")]
        [Required(ErrorMessage = "Indique os requisitos do curso!")]
        10 referências
        public string Requisitos { get; set; }
        [Display(Name = "Idade mínima", Prompt = "Idade Mínima",
            Description = "Idade mínima para poder frequentar esta formação")]
        [Required(ErrorMessage = "Indique a Idade Mínima para este curso!")]
        [Range(14, 100, ErrorMessage = "Mínimo: 14 anos, Máximo: 100 anos")]
        10 referências
        public int IdadeMinima { get; set; }
        [Display(Name = "Curso activo?")]
        12 referências
        public bool Disponivel { get; set; }
        [Display(Name = "Curso em Destaque?")]
        11 referências
        public bool EmDestaque { get; set; }
        [Required(ErrorMessage = "Indique o Preço do curso!")]
        [Display(Name = "Preço", Prompt = "Introduza o Preço", Description = "Preço do curso")]
        [DisplayFormat(DataFormatString = "{0:C2}")]
        12 referências
        public decimal? Preco { get; set; }
        [Display(Name = "Categoria")]
        11 referências
        public int? CategoriaId { get; set; }
        10 referências
        public Categoria categoria { get; set; }
    }
}
```

Programac

Criação do Contexto

3. Definição do **Contexto** da Base de Dados

Na pasta ***Data*** crie o arquivo ***ApplicationDBContext***

IPC @ IB

```
using AutoWEB.Models;
using Microsoft.AspNetCore.Identity.EntityFrameworkCore;
using Microsoft.EntityFrameworkCore;

namespace AutoWEB.Data
{
    16 referências
    public class ApplicationDbContext : IdentityDbContext <ApplicationUser>
    {
        19 referências
        public DbSet<Curso> Cursos { get; set; }
        17 referências
        public DbSet<Categoria> Categorias { get; set; }
        12 referências
        public DbSet<Agendamento> Agendamentos { get; set; }
        18 referências
        public DbSet<TipoDeAula> TipoDeAula { get; set; }

        0 referências
        public ApplicationDbContext(DbContextOptions<ApplicationDbContext> options)
            : base(options)
        {
        }
    }
}
```

DBContext como Serviço – Program.cs

4. Registar em ***Program.cs*** o ***DBContext*** como um serviço

No ficheiro ***program.cs***, tem de se especificar o ***DBContext*** como um serviço

```
builder.Services.AddDbContext<ApplicationDbContext>(options =>
   options.UseSqlServer(connectionString));

...
```

Programação Web – 2022/2024 – LEI D+CE+PL – ISEC/IPC @JR

isec Engenharia DEIS - Instituto Superior de Engenharia de Coimbra

Especificação da *Connection String*

5. Especificação da ***Connection String*** no ficheiro ***appsettings.json***

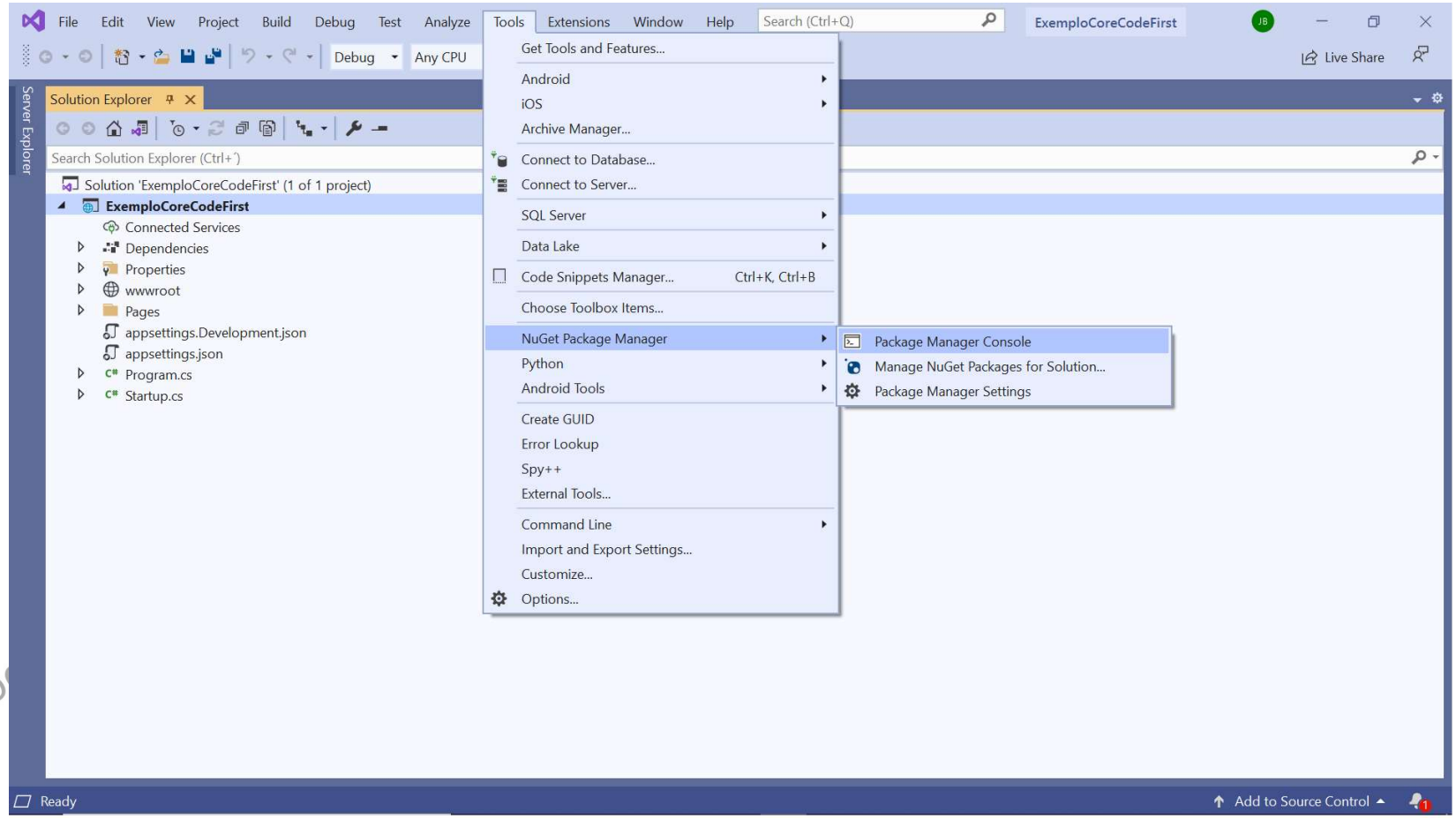
No ficheiro *appsettings.json*, define-se uma *string* de conexão com a base de dados **usando uma instância local da BD**

```
"ConnectionStrings": {
  "DefaultConnection":
"Server=(localdb)\\mssqllocaldb;Database=AutoWeb;Trusted_Connection=True;MultipleActiveResultSets=true"
 },
 ...
```

Programação Web – 2022/2024 – LEI D+CE+PL – ISEC/IPC @JS

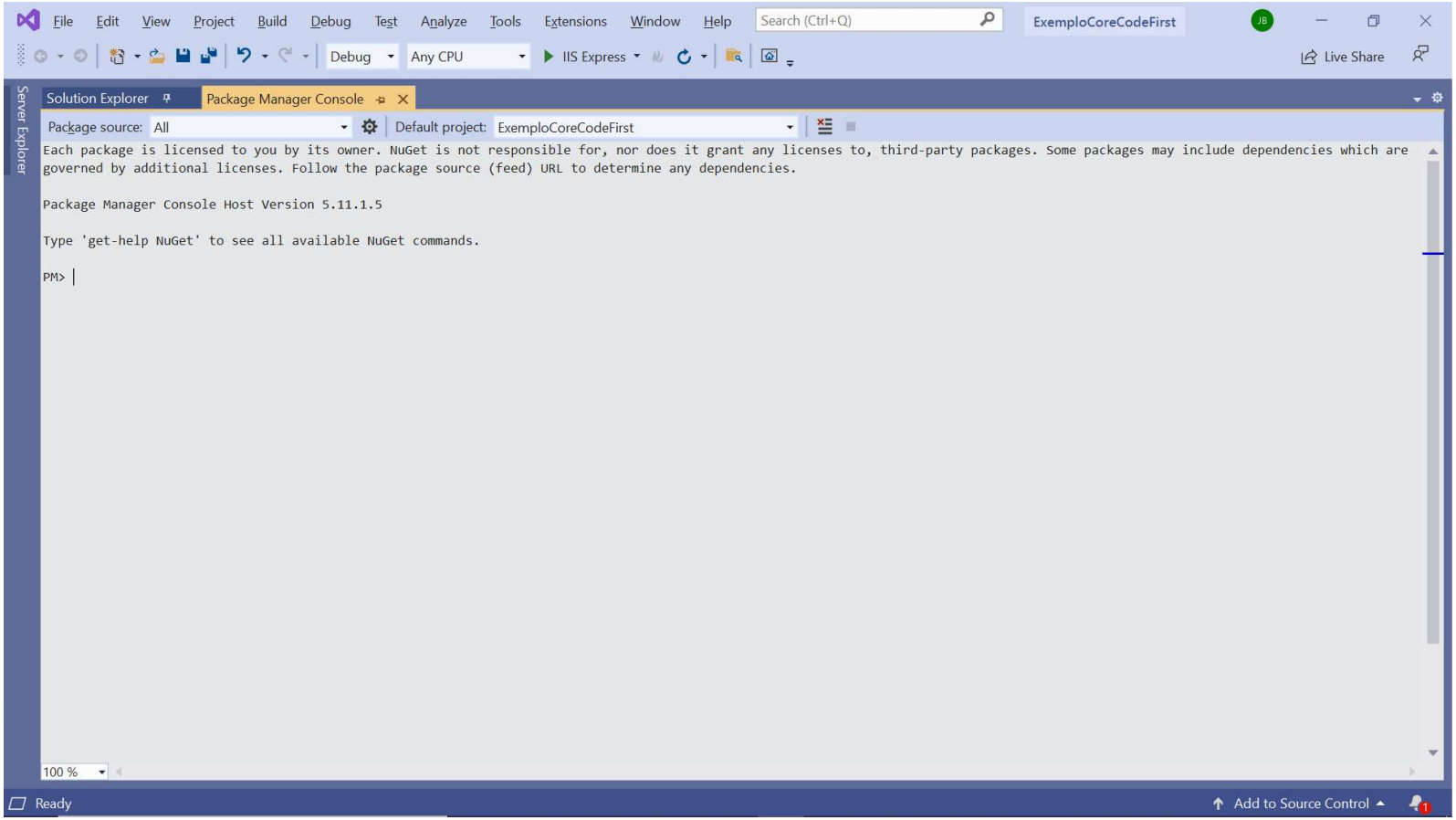
isec Engenharia DEIS - Instituto Superior de Engenharia de Coimbra

Ativação das *Migrations*



isec Engenharia DEIS - Instituto Superior de Engenharia de Coimbra

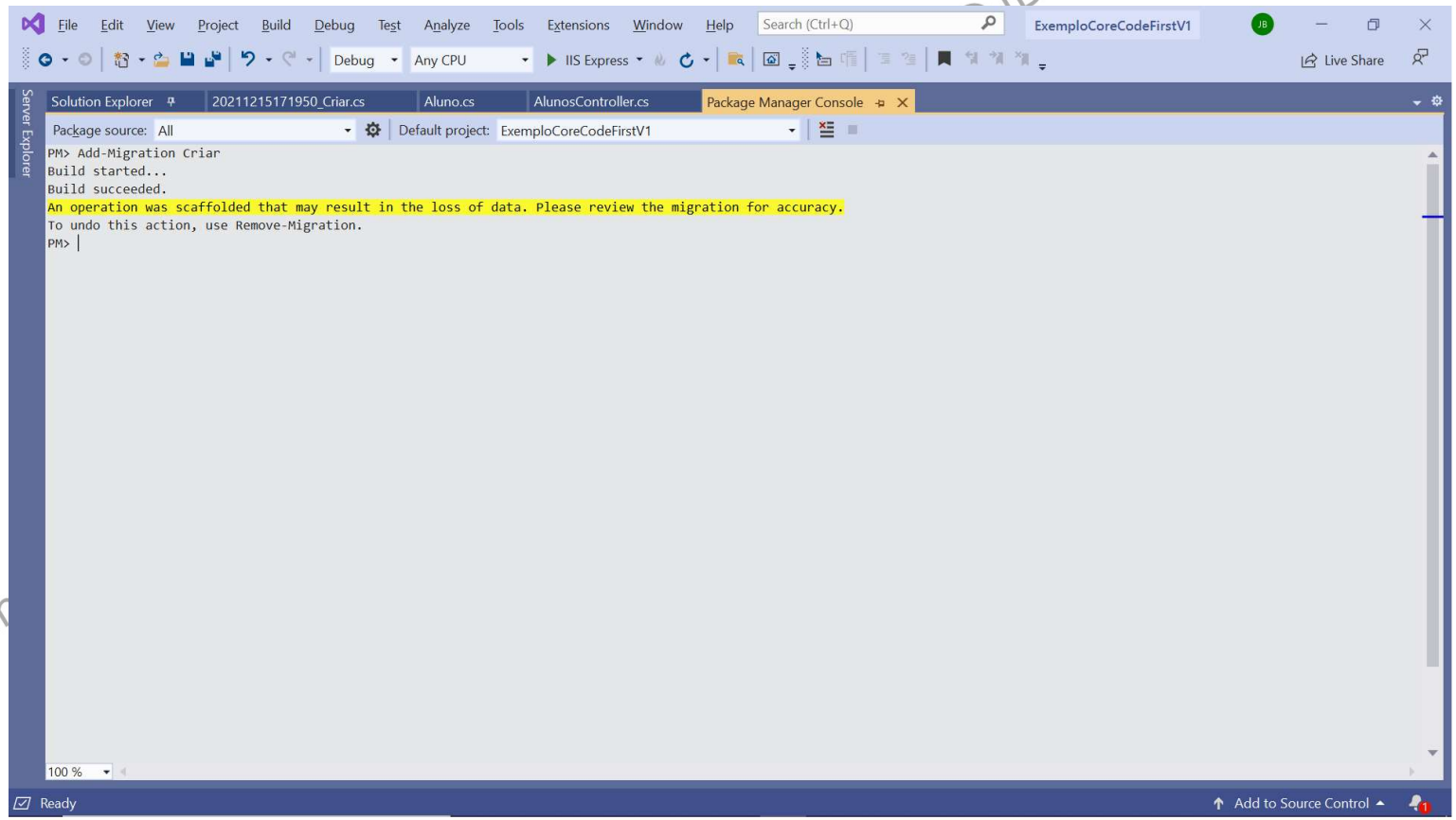
Package Manager Console



isec DEIS - Instituto Superior de Engenharia de Coimbra

Migrations

PM> Add-Migration Criar ou >dotnet ef migrations add Criar

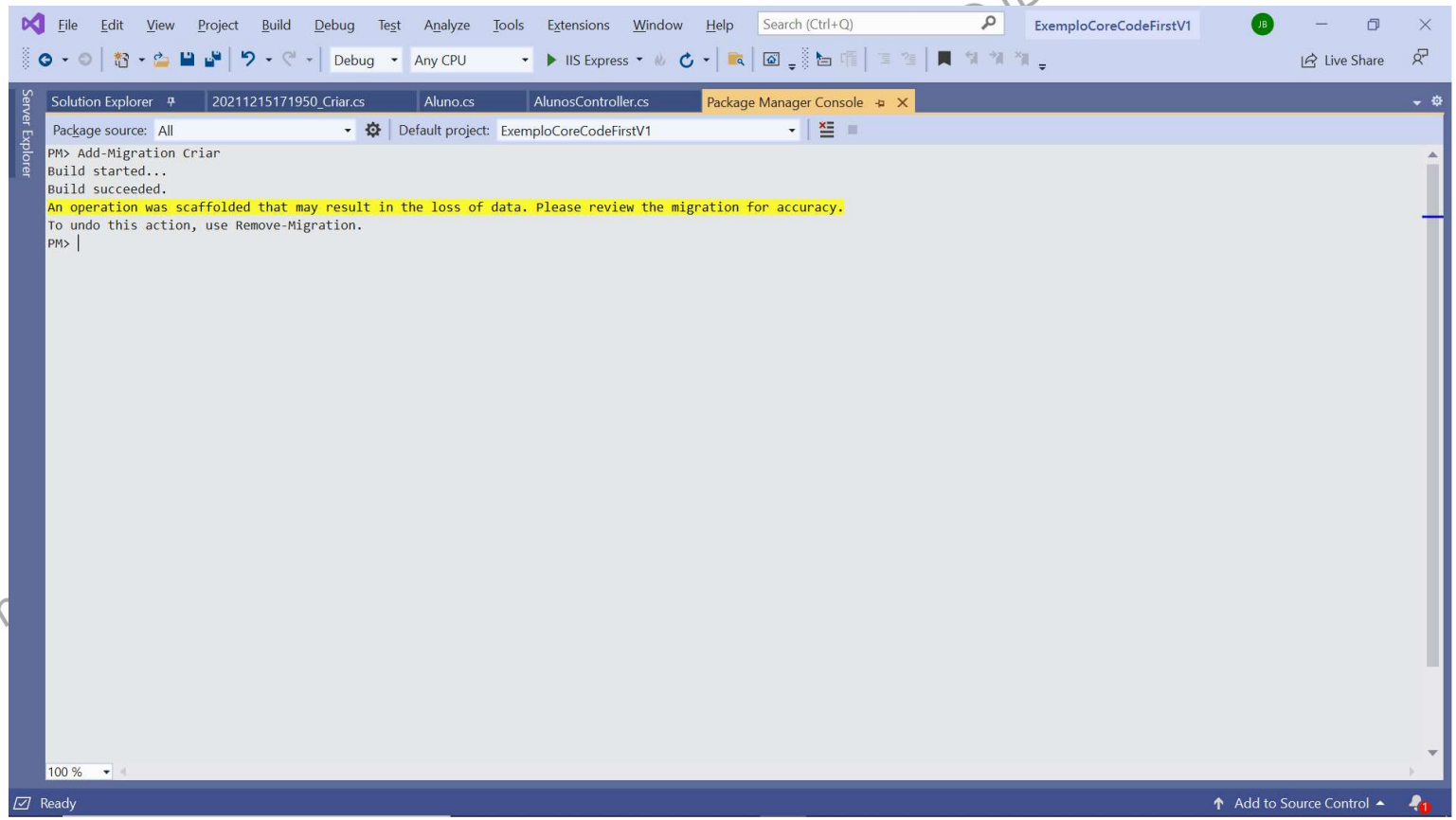


Progran

isec Engenharia DEIS - Instituto Superior de Engenharia de Coimbra

Migrations

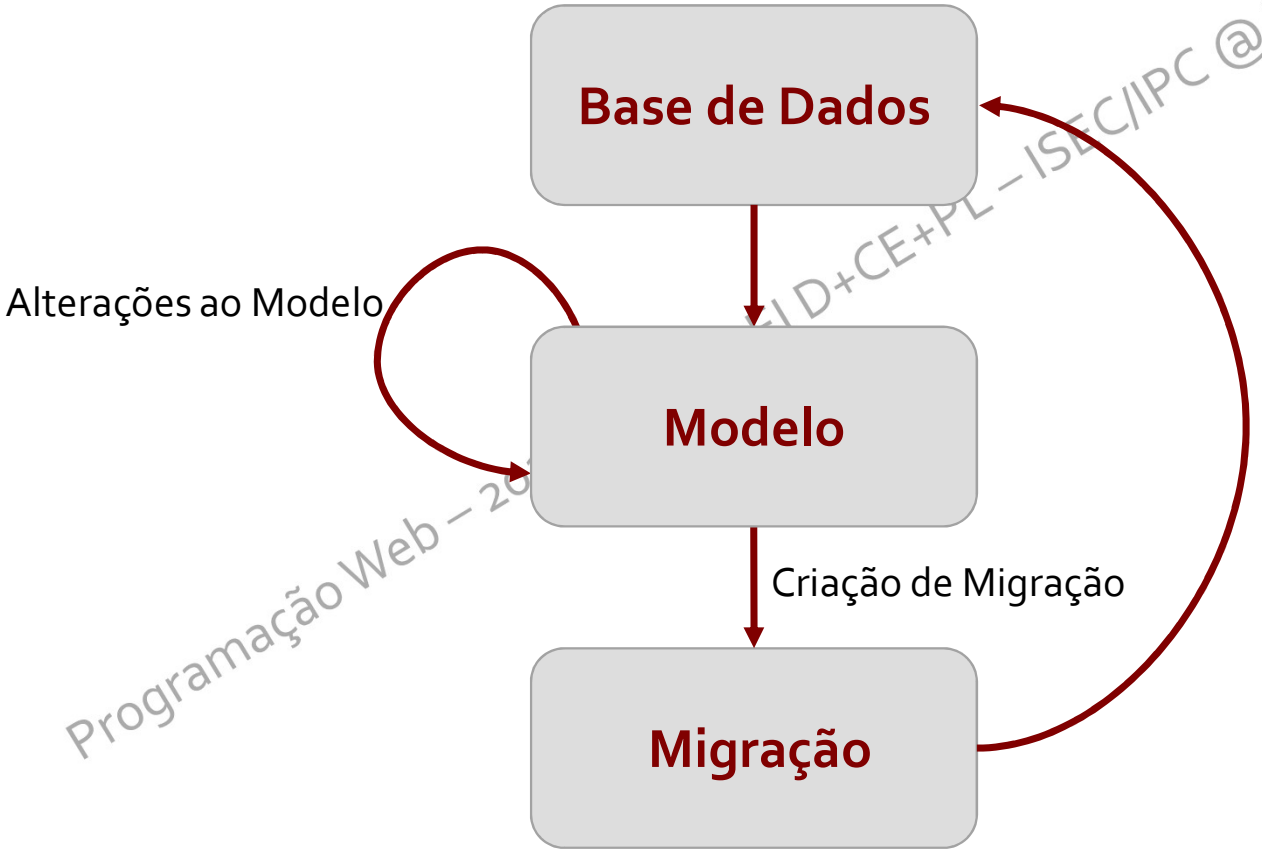
```
PM> update-database
```



Program

isec Engenharia DEIS - Instituto Superior de Engenharia de Coimbra

Processo Iterativo



Executa
Manter a Base de Dados atualizada

Permite obter o histórico da Base de Dados
Os scripts gerados na Migrations permitem fazer o Upgrade e o Downgrade das alterações

Programação Web – 20 ED+CE+PL – ISEC/IPC @JB

DEIS - Instituto Superior de Engenharia de Coimbra

Mas qual o objectivo das Migrações (Migrations) ?

Programação Web – 2022/2024 – LEI+LEI+PL – LEI+PÉ @JB

EF Core Migrations

A *EF Core API* constrói o modelo *EF Core* a partir das classes de domínio (entidades) e as *migrations EF Core* criarão ou atualizarão o esquema da base de dados com base no modelo *EF Core*

Sempre que as classes de domínio são alteradas, é necessário executar novamente a *migration* para manter o esquema da base de dados atualizado

Programação Web - 2022/2024 - LEI D+CB

EF Core Migrations

As *migrations EF Core* são um conjunto de comandos que se executam na consola do gestor de pacotes *NuGet* ou na interface de linha de comando dotnet

A tabela seguinte mostram-se os comandos de *migration* mais importantes no *EF Core*

Programação Web – 2022/2024 – LEI D+CE+PL – ISEC/IPC @JB

isec Engenharia DEIS - Instituto Superior de Engenharia de Coimbra

EF Core Migrations - Comandos

Comando PMC	Comando dotnet CLI	Utilização
add-migration <migration name>	Add <migration name>	Cria a migration adicionando uma migration snapshot.
Remove-migration	Remove	Remove a última migration snapshot.
Update-database	Update	Actualiza o esquema da base de dados baseada na útlima migration snapshot.
Script-migration	Script	Gera um script SQL usando todas as migration snapshots.

EF Core Migrations - Comandos

Adicionando uma Migration

Inicialmente, definiram-se as classes de domínio

Após esta definição, ainda não se tem a base de dados para armazenar os dados das classes de domínio.

O passo seguinte é então criar essa BD e para isso utiliza-se uma *migration*

Selecciona-se a opção do menu Ferramentas -> Gerenciador de pacotes NuGet -> Console do gerenciador de pacotes no Visual Studio

Com isto é aberto a Consola do NuGet e nesta consola executa-se o comando para adicionar uma migration

PM> add-migration MyFirstMigration

Caso se use a interface de linha de comando dotnet, executa-se o comando ***CLI***

> dotnet ef migrations add MyFirstMigration

Programação Web – 2023/2024 – LEID+CE+PL – ISEC/IPC @JB

EF Core Migrations - Comandos

Criar ou Actualizar a Base de Dados

Use um dos seguintes comandos para criar ou atualizar o esquema da base de dados

Package Manager Console

PM> Update-Database

CLI

> dotnet ef database update

O comando *Update* criará a base de dados com base no contexto e nas classes de domínio e no *snapshot* da *migration*, criado usando o comando *add-migration* ou *add*

- Na primeira migração, também será criada uma tabela chamada *__EFMigrationsHistory*, que armazenará o nome de todas as *migrations*, como e quando serão aplicadas à base de dados

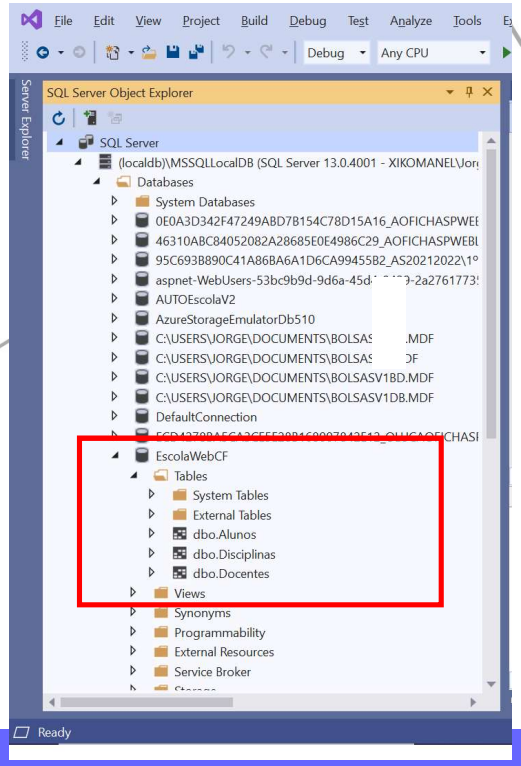
Programação Web – 2022/2024 – LEI D+CE+PL – ISEC/IPC @JB

isec Engenharia DEIS - Instituto Superior de Engenharia de Coimbra

EF Core Migrations - Comandos

Criar ou Actualizar a Base de Dados

Após a execução deste comandos em que ***MyFirstMigration*** é o nome de uma ***migration*** será criada no SQL Server a Base de Dados e as respectivas tabelas



Programação Web – ISEC/IPC @JB

isec Engenharia DEIS - Instituto Superior de Engenharia de Coimbra

Migrattions – Passos Principais

Usar Migrações Code First

2 Passos Principais:

1. **Adicionar as alterações**

PM> Add-Migration *darumnomequalqueraadicao*

Se falhar, fazer o Clean da Solution e a seguir o Rebuild

2. **Actualizar a Base de Dados**

PM> Update-Database

ProgramaçãoWeb – 2022/2024 – LEI D+CE+PL – ISEC/IPC @JB

isec Engenharia DEIS - Instituto Superior de Engenharia de Coimbra

Mapeamento entre tipo C# e tipo SQL

Tipo de Dados em C#	Tipo de Dados na coluna da DB	Tipo de dados da Coluna da chave Primária e Comprimento
int	int	int, Coluna Identity
string	nvarchar(Max)	nvarchar(128)
decimal	decimal(18,2)	decimal(18,2)
float	real	real
byte[]	varbinary(Max)	varbinary(128)
datetime	datetime	datetime
bool	bit	bit
byte	tinyint	tinyint
short	smallint	smallint
long	bigint	bigint
double	float	float
char	No mapping	No mapping
sbyte	No mapping (throws exception)	No mapping
object	No mapping	No mapping

Nota: Na ***class C#,*** a propriedade referente à ***Key*** deve ser definida como ***long***

isec Engenharia DEIS - Instituto Superior de Engenharia de Coimbra

DataAnnotations

- ***DataAnnotations***
  - Simples de especificar
  - Menos flexível
  - Útil para anotações simples como obrigatórios, tamanhos máximos…
  - *DataAnnotations* tem limitações de configuração

Programação Web – 2023/2024 – LEI+CE+PL – ISEC/IPC @JB

isec Engenharia DEIS - Instituto Superior de Engenharia de Coimbra

DataAnnotations: Key

- Chaves Primárias

```
using System.ComponentModel.DataAnnotations;

[Key]
private int NAluno { get; set; }

[Key]
[DatabaseGenerated(DatabaseGe
private string SIGLA { get; s
```

DatabaseGeneratedOption
- None
- Identity
- Computed

Programação Web – 2022/2024 – LEI D+CE+PL – ISEC/IPC @JB

isec Engenharia DEIS - Instituto Superior de Engenharia de Coimbra

DataAnnotations: Keys Compostas

- Chaves Primárias Compostas

```
public class Inscricoes
{
    [Key]
    [Column(Order=1)]
    public int DisciplinaId { get; set; }

    [Key,Column(Order=2)]
    public int AlunoId { get; set; }

    public DateTime DataInscricao { get; set; }
}
```

Programação Web – 2022/2024 – LEI D+CE+PL – ISEC/IPC @JB

isec Engenharia DEIS - Instituto Superior de Engenharia de Coimbra

DataAnnotations: NotMapped

```
[NotMapped]
public int Idade { get; set; }

public string PrimeiroNome
{
    get { return AlunoNome; }
}
public int IdadeGet
{
    get { return 0; }
}
```

Programação Web – 2022/2024 – LEID+CE+PL – ISEC/IPC @JB

DataAnnotations: Índices

```
[Index]
public int DisciplinaId { get; set; }


[Index(IsUnique = true)]
public int DisciplinaId { get; set; }
```

Programação Web – 2022/2024 – LEI D+CE+PL ISEC/IPC @JB

isec Engenharia DEIS - Instituto Superior de Engenharia de Coimbra

ASP.Net Core Entity Framework Data Base First

Programação Web – 2022/2024 – LEI+CE+PL – ISEC/IPC @JB

Entity Framework – Data Base First

O ***Entity Framework Database First*** é uma abordagem alternativa no desenvolvimento de sites para a *Web* utilizando-se o ***ASP.Net Core Entity Framework***.

- Pode ser utilizada quer se disponha já de uma base de dados, por exemplo a utilizada num site que se quer actualizar ou quer se tenha desenvolvido especifica e previamente uma Base de Dados para um novo site
  - No entanto é comum que no desenvolvimento de raíz de um site, se utilize a abordagem *Code First*

Programação Web – 2022/2024 – EI D+CE+PL – ISEC/DEIS @ 2

isec Engenharia DEIS - Instituto Superior de Engenharia de Coimbra

Entity Framework – Data Base First

No caso de se dispor já de uma Base de Dados do um *site* em produção, a Base de Dados respectiva já deverá ter um número muito grande de tabelas as quais também já deverão conter um número muito significativo de dados inseridos

- Neste caso não faria sentido não se utilizar essa Base de Dados, por exemplo numa actualização desse *site*

Programação Web – 2022/2024 – LEI D – CE+PJ – ISEC/IPC @JB

isec Engenharia DEIS - Instituto Superior de Engenharia de Coimbra

Entity Framework – Data Base First

Nesta situação de se dispor já de uma Base de Dados, se se adoptasse a abordagem *Code First*, poderiam ocorrer alguns problemas:

- Ter-se-ia de se definir manualmente cada uma das classes C# correspondentes a cada uma das tabelas e as respectivas relações
- Não se poderia usar migrações *Code First* porque a base de dados existente provavelmente estará a ser gerida de uma maneira diferente.
  - Embora isto não fosse um problema, perder-se-ia uma das grandes vantagens de se usar o *Code First*

Programação Web – 2022/2024 – LEID+CT+PL – ISEC/IPC @JB

isec Engenharia DEIS - Instituto Superior de Engenharia de Coimbra

Entity Framework – Data Base First

Deste modo, nestas situações é preferível usar a abordagem ***Data Base First***:

- O que se faria criando manualmente o código das classes ***C#***, será feita pela ***EF*** através da inspecção da base de dados existente.

- O resultado dessa acção de análise da *BD* pela ***EF*** será então a criação das classes de dados ***C#***

Programação Web – 2022/2024 – LEI D+CE+PL – ISEC/IPC @ JP

isec Engenharia DEIS - Instituto Superior de Engenharia de Coimbra

Entity Framework – Data Base First

Para que o ***EF*** realize essa inspecção da *BD* e crie as respectivas classes ***C#***, basta executar o seguinte comando:

Scaffold-DbContext "MyConnectionStringHere“

E o Entity Framework fará o resto por nós

Programação Web – 2022/2024 – LEI-PL/CE+PL – ISEC/IPC @JB

isec Engenharia DEIS - Instituto Superior de Engenharia de Coimbra

Programação Web

Aulas Teóricas – Capítulo 2 – 2.5

1º Semestre - 2023/2024

Departamento de Engenharia Informática e de Sistemas
Instituto Superior de Engenharia de Coimbra/Instituto Politécnico de Coimbra

Programação Web

Segurança em Aplicações Web

Departamento de Engenharia Informática e de Sistemas
Instituto Superior de Engenharia de Coimbra/Instituto Politécnico de Coimbra

Conceitos Gerais

- Autenticação
  - Processo de verificação de identidade de um utilizador ou computador
    - Quem é? Como o provar?
    - Verifica se utilizador é realmente quem diz ser
    - Credenciais podem ser especificadas através de um código de acesso (*password*), um *token* externo, um smart card, …

Programação Web – 2023/2024 – LEI D+CE+PL – ISEC/IPC @JB

DEIS - Instituto Superior de Engenharia de Coimbra

Conceitos Gerais

- Autorização
  - Processo para determinar o que é que o utilizador pode aceder num determinado computador, rede, …
    - Quais as permissões?
    - Tem acesso a este conteúdo?
    - Tem permissões para ver o conteúdo desta página?

Programação Web – 2023/2024 – LEI D+CEPL – ISEC/IPC @JB

isec Engenharia DEIS - Instituto Superior de Engenharia de Coimbra

Evolução: Identidade em ASP.NET

- 2005: *ASP.NET Membership*
  - Armazenamento e gestão de dados da identidade
  - *SQL Server*
- 2011: *Simple Membership*
  - Sistema de *Membership* para modelo *ASP.NET Web Pages*
  - Não é possível usar com o *OWIN* (*Open Web Interface for .NET*)

Programação Web – 2023/2024 – LEI D+CE+PL – ISEC/IPC @JB

isec Engenharia DEIS - Instituto Superior de Engenharia de Coimbra

Evolução: Identidade em ASP.NET

- 2012: *Universal Providers Membership*
  - Passou a permitir armazenamento de dados adicionais
  - Suporte à associação de fornecedores externos
  - Permitiu armazenamento da informação *membership* no *Windows Azure SQL Database*
  - Permitiu o uso do *Membership* com a *EF*
- Após 2013: ***ASP.NET Identity***
- **Actualmente: *ASP.Net Core Identity*** *Framework*

Programação Web – 2023/2024 – EID+CE+PL – ISEC/IPC @JB

isec Engenharia DEIS - Instituto Superior de Engenharia de Coimbra

ASP.Net Core Identity Framework

ASP.NET Core Identity: Características

- **Autenticação** e **Autorização** para aplicações *ASP.NET Core*
  - Suporta *ASP.NET Core, MVC, Web API, Web Forms, SignalR, Web Pages*
- Permite efetuar a manipulação de utilizadores, *profiles*, processo de login/logout, criação de perfis, *claims*, etc…
  - Permite manter as contas do utilizador numa base de dados local ou num armazenamento de dados externo

Programação Web – 2023/2024 – LEI D+CE+PL – ISEC/IPC @ DEIS

isec Engenharia DEIS - Instituto Superior de Engenharia de Coimbra

ASP.NET Core Identity: Características

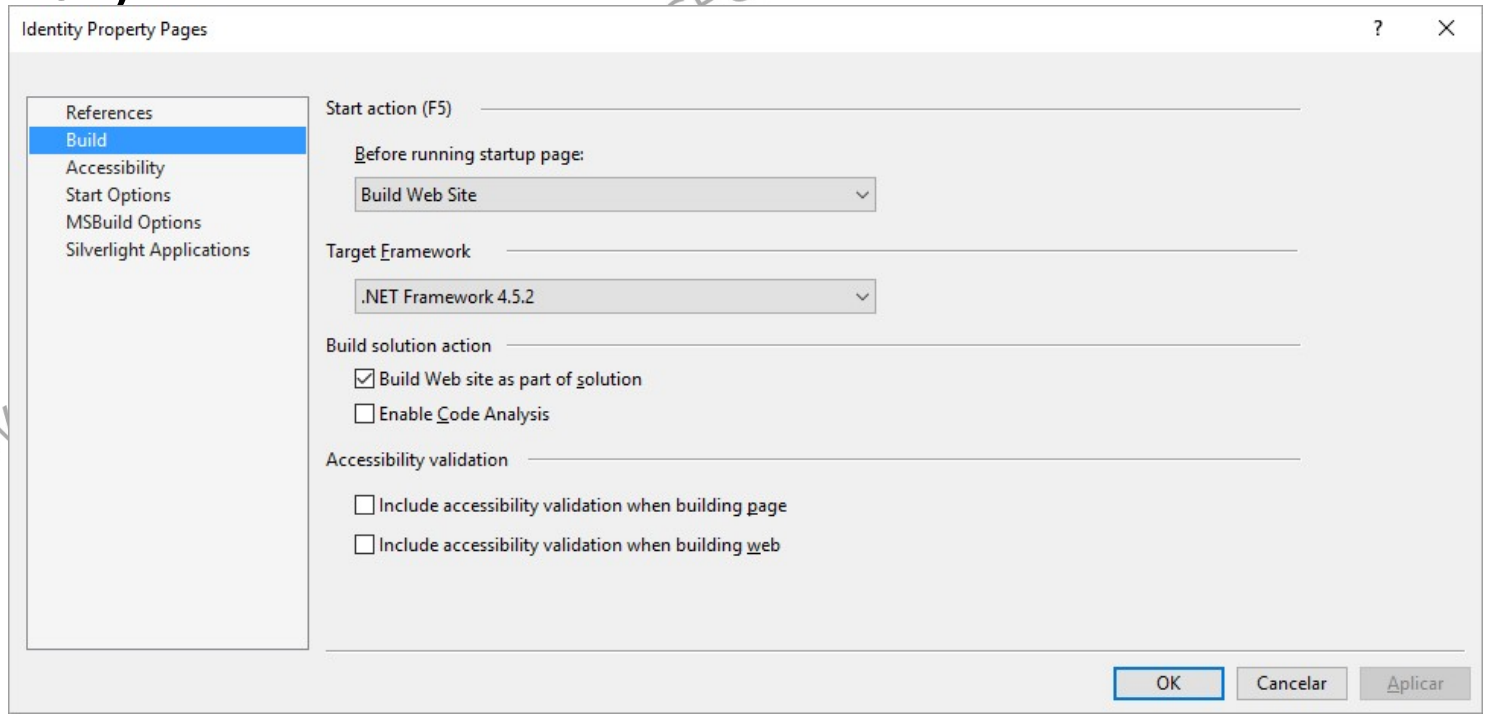
- Facilita escrita de testes unitários
- Permite login externo (através do ***OAuth 2.0***)
  - Suporta *Facebook*, *Google*, *Microsoft*, *Twitter*, *accounts*
- Suporta o *middleware* ***OWIN***
  - Pode executar num servidor que não o *IIS* da *Microsoft*
- Disponível através do gestor de bibliotecas ***NuGet***

Programação Web – 2023/2024 – LEI D+CT PL – ISEC/IPC @JB

isec Engenharia DEIS - Instituto Superior de Engenharia de Coimbra

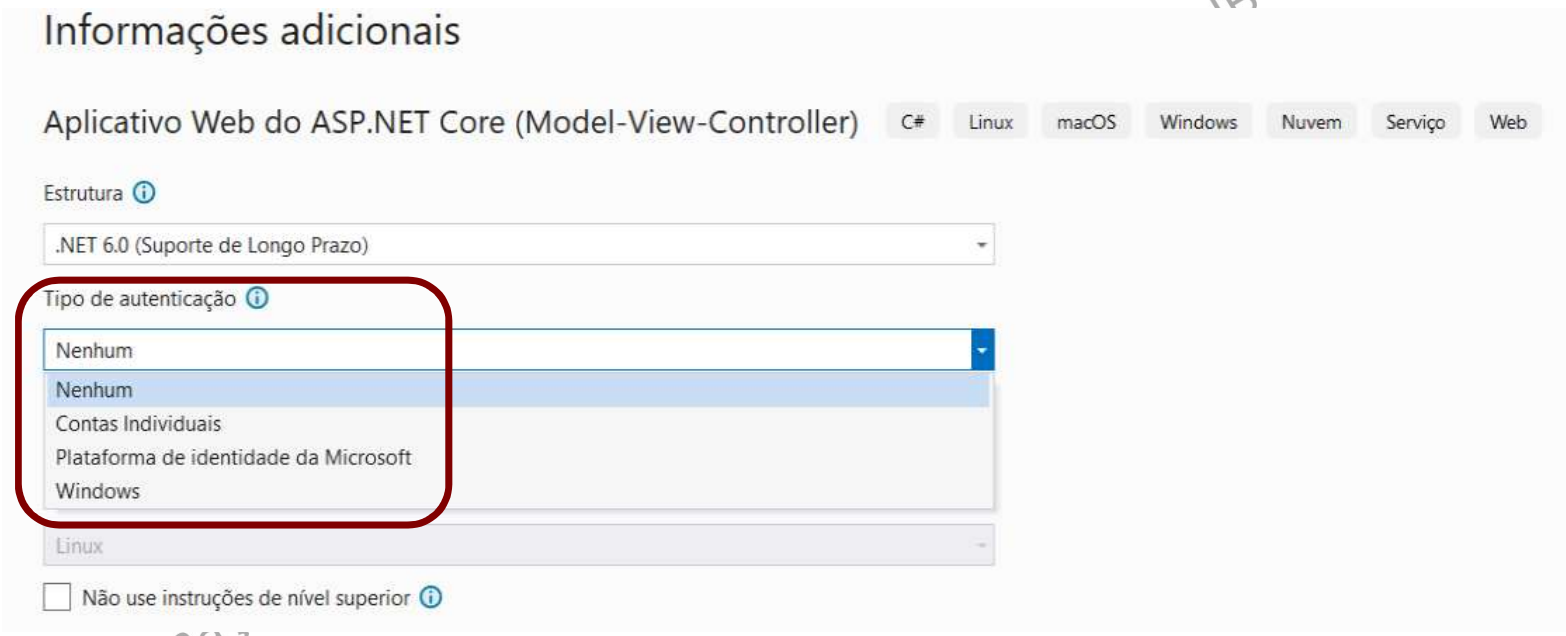
ASP.NET Core Identity

- O *ASP.NET Core Identity* recorre à framework 6 (ou versão superior)



Programação W
EC/IPC @

Formas de Autenticação



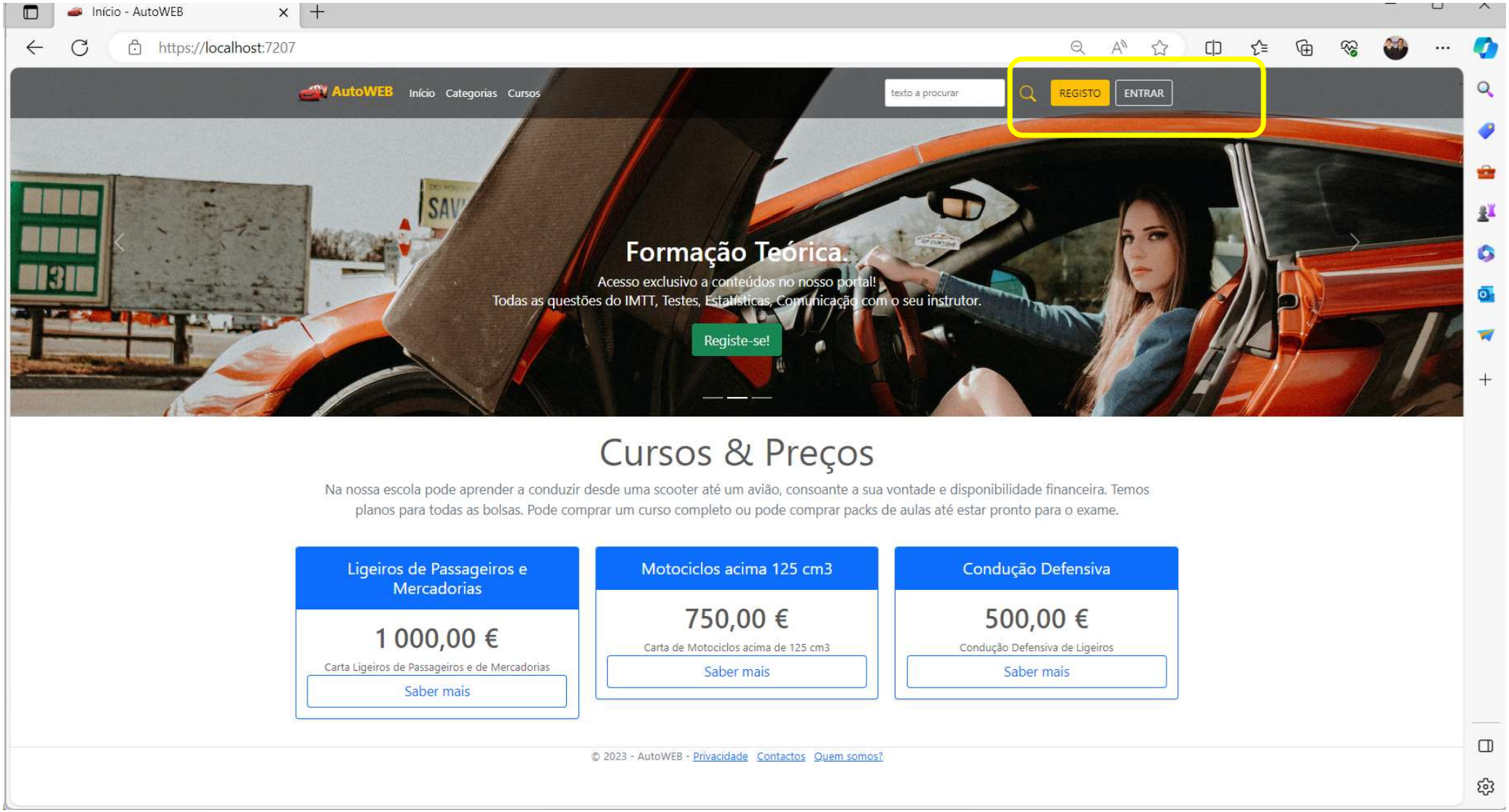
Programa

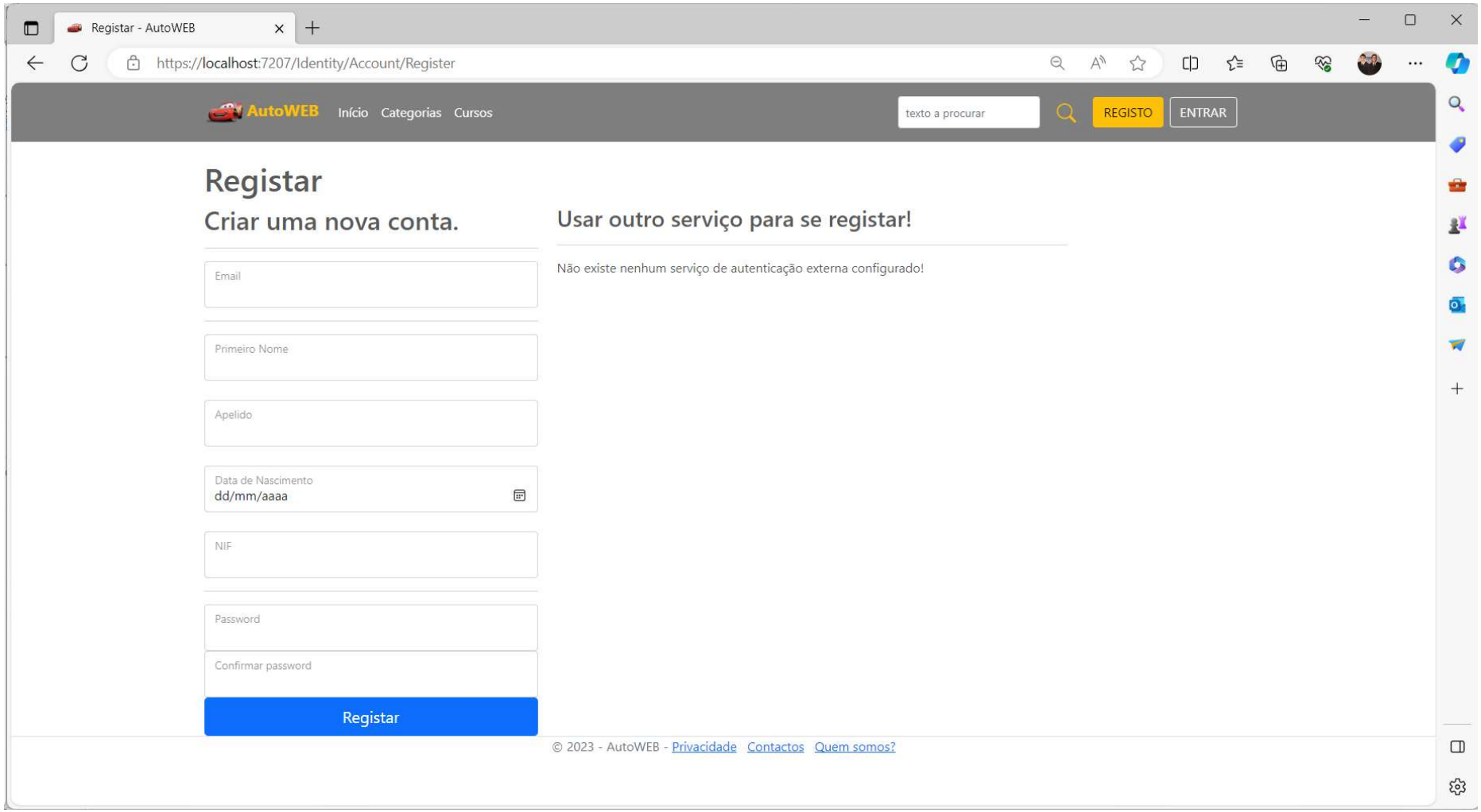
Formas de Autenticação

- **No Authentication**
  - Sem autenticação
- ***Individual User Accounts***
  - Tipicamente utilizado para *websites*
  - Permite a criação de utilizadores locais
  - Permite activação de logins com contas sociais como *Facebook, Google, Twitter*,…

Programação Web – 2023/2024 – LEI D+CE+PL – ISEC/IPC @JB

- Para gerir perfis, acessos diferenciados…devo criar uma tabela de utilizadores e perfis?


Início - AutoWEB
https://localhost:7207
AutoWEB
Início
Categorias
Cursos
texto a procurar
REGISTO
ENTRAR
Formação Teórica.
Acesso exclusivo a conteúdos no nosso portal!
Todas as questões do IMTT, Testes, Estatísticas, Comunicação com o seu instrutor.
Registe-se!
Cursos & Preços
Na nossa escola pode aprender a conduzir desde uma scooter até um avião, consoante a sua vontade e disponibilidade financeira. Temos planos para todas as bolsas. Pode comprar um curso completo ou pode comprar packs de aulas até estar pronto para o exame.
Ligeiros de Passageiros e Mercadorias
1 000,00 €
Carta Ligeiros de Passageiros e de Mercadorias
Saber mais
Motociclos acima 125 cm3
750,00 €
Carta de Motociclos acima de 125 cm3
Saber mais
Condução Defensiva
500,00 €
Condução Defensiva de Ligeiros
Saber mais
© 2023 - AutoWEB - Privacidade Contactos Quem somos?


Registar - AutoWEB
https://localhost:7207/Identity/Account/Register
AutoWEB
Início
Categorias
Cursos
texto a procurar
REGISTO
ENTRAR
Registar
Criar uma nova conta.
Email
Primeiro Nome
Apelido
Data de Nascimento
dd/mm/aaaa
NIF
Password
Confirmar password
Registar
Usar outro serviço para se registar!
Não existe nenhum serviço de autenticação externa configurado!
© 2023 - AutoWEB - Privacidade Contactos Quem somos?

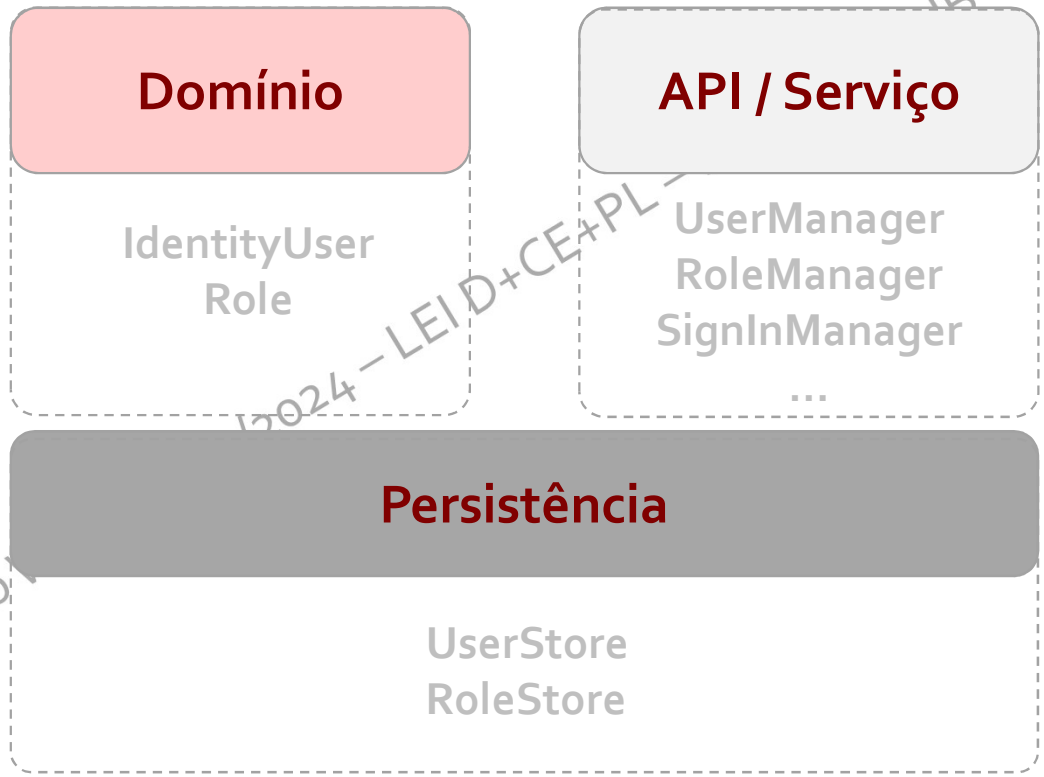
ASP.NET Core Identity com Razor Pages

```
// Licensed to the .NET Foundation under one or more agreements.
// The .NET Foundation licenses this file to you under the MIT license.
#nullable disable

using System;
using System.Collections.Generic;
using System.ComponentModel.DataAnnotations;
using System.Linq;
using System.Text;
using System.Text.Encodings.Web;
using System.Threading;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Authentication;
using Microsoft.AspNetCore.Authorization;
using Microsoft.AspNetCore.Identity;
using Microsoft.AspNetCore.Identity.UI.Services;
using Microsoft.AspNetCore.Mvc;
using Microsoft.AspNetCore.Mvc.RazorPages;
using Microsoft.AspNetCore.WebUtilities;
using Microsoft.Extensions.Logging;
using AutoWEB.Models;

namespace AutoWEB.Areas.Identity.Pages.Account
{
    8 referências
    public class RegisterModel : PageModel
    {
        private readonly SignInManager<ApplicationUser> _signInManager;
        private readonly UserManager<ApplicationUser> _userManager;
        private readonly IUserStore<ApplicationUser> _userStore;
        private readonly IUserEmailStore<ApplicationUser> _emailStore;
        private readonly ILogger<RegisterModel> _logger;
        private readonly IEmailSender _emailSender;

        0 referências
        public RegisterModel(
            UserManager<ApplicationUser> userManager,
            IUserStore<ApplicationUser> userStore,
            SignInManager<ApplicationUser> signInManager,
            ILogger<RegisterModel> logger,
            IEmailSender emailSender)
        {
            _userManager = userManager;
            _userStore = userStore;
            _emailStore = GetEmailStore();
            _signInManager = signInManager;
            _logger = logger;
            _emailSender = emailSender;
        }
```

Programa
_+PL – ISEC/IPC @JB

isec Engenharia DEIS - Instituto Superior de Engenharia de Coimbra

ASP.NET Core Identity: Arquitetura



Programação Web 2024 – LEI D+CE+PL – JB

isec Engenharia DEIS - Instituto Superior de Engenharia de Coimbra

ASP.NET *Identity*

UserManager, RoleManager

UserStore, RoleStore

SQL Server, MySql, Windows Azure Table Storage

Programaçã
PL – ISEC/IPC @JB

isec Engenharia DEIS - Instituto Superior de Engenharia de Coimbra

ASP.NET *Identity: Classes e Interfaces*

- As classes e interfaces mais importantes no *ASP.NET Identity* são as seguintes:
  - ***Iuser***
  - ***IdentityUser***
  - ***IdentityDbContext***
  - ***UserManager<Tuser>***
  - ***IAuthenticationManager***

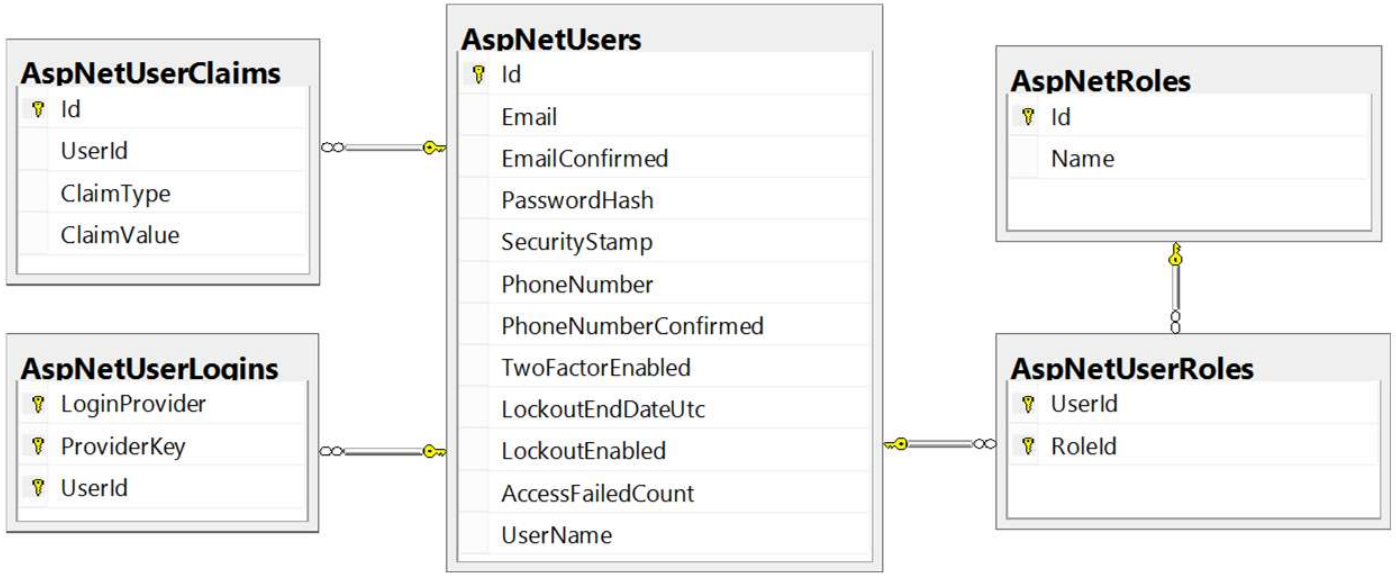
Programação Web – 2023/2024 – LEI D+CE+PL – ISEC/IPC @ DEIS

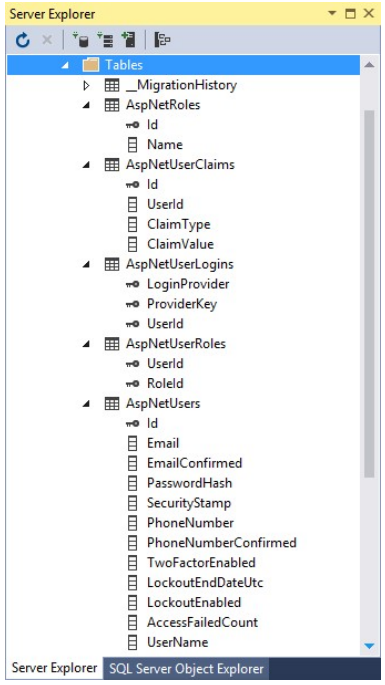
isec Engenharia DEIS - Instituto Superior de Engenharia de Coimbra

ASP.NET Core Identity: Modelo de Dados

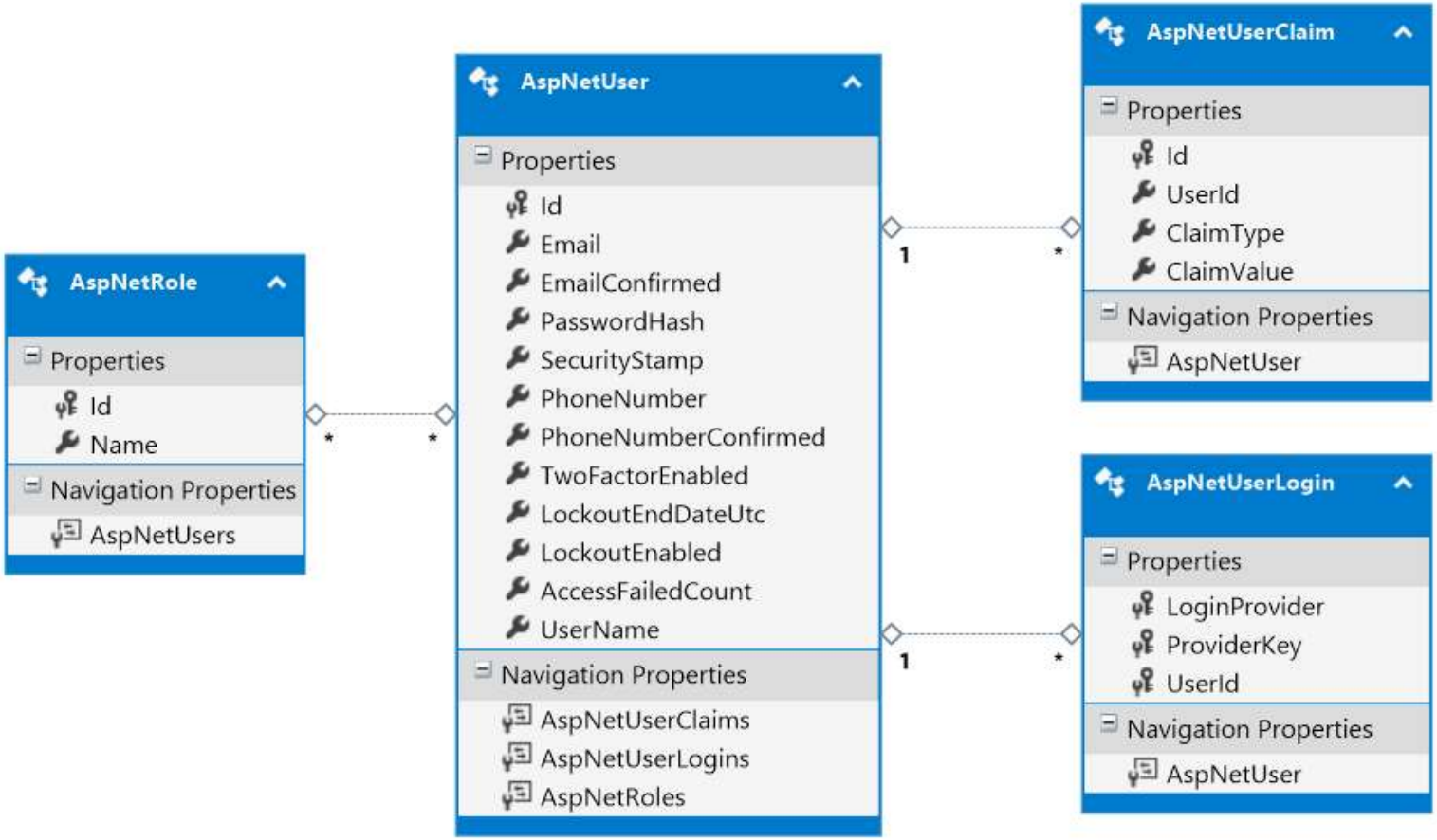
- Geralmente, os dados de identidade do ***ASP.NET Core*** (utilizadores, *passwords*, perfis) são armazenados numa base de dados relacional por meio da ***EF Code First***

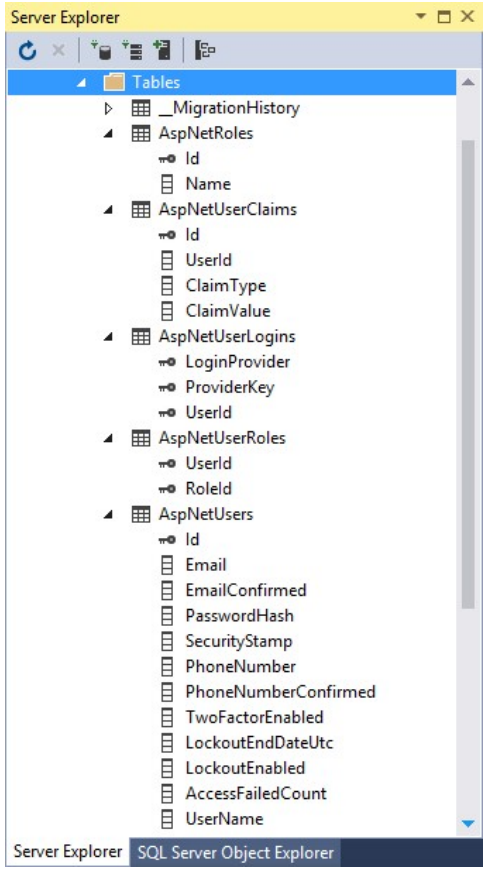
D+CE+D - ISEC/IPC @JP





ASP.NET Identity: Modelo de Dados





Estender IdentityUser - ApplicationUser

```
using Microsoft.AspNetCore.Identity;
using System.ComponentModel.DataAnnotations;

namespace AutoWEB.Models
{
    40 referências
    public class ApplicationUser : IdentityUser
    {
        [Display(Name = "Primeiro Nome")]
        9 referências
        public string PrimeiroNome { get; set; }

        [Display(Name = "Último Nome")]
        6 referências
        public string UltimoNome { get; set; }

        [Display(Name = "Data de Nascimento")]
        3 referências
        public DateTime DataNascimento { get; set; }
        3 referências
        public int NIF { get; set; }

        [Display(Name = "A minha fotografia")]
        3 referências
        public byte[]? Fotografia { get; set; }
        0 referências
        public ICollection<Agendamento> Agendamentos { get; set; }
    }
}
```

@JB

Pro

Identity Framework Core com Razor Pages

- Iremos implementar a Identity com o recurso a Razor Pages

Programação Web – 2023/2024 – LEI D+CE+PL – ISEC/IPC @JB

isec Engenharia DEIS - Instituto Superior de Engenharia de Coimbra

ASP.NET Identity: Configuração

- Formas de configurar a autenticação baseada no *ASP.NET Core Identity*
  - Recorrer ao *template* do *ASP.NET* Core a partir do *Visual Studio*
  - Ou… Instalar bibliotecas através do *Nuget*
  - Bibliotecas a instalar:
    - `Microsoft.AspNet.Identity.EntityFrameworkCore`

Programação Web – 2023/2024 – LEI D+CEM – ISEC/IPC @ DEIS

isec Engenharia DEIS - Instituto Superior de Engenharia de Coimbra

ASP.NET Core Identity: Autorize

```
[Authorize]

public ActionResult Index()
{
    return View();
}
```

Filtro

Será chamado pela Framework antes e depois da acção ser exectuada

O atributo *Autorize* irá verificar se o utilizador corrente está bloqueado ou não

Se não estiver logado, reencaminha para a página de login

ogramação Web – 2023/2024 – LEI D+CE+PL – ISEC/DEIS @JB

ASP.NET Identity: Autorize

- Aplicar no controlador

```
using System.Data.Entity;
using System.Linq;
using System.Net;
using System.Web.Mvc;
using CodeFirstApp.Models;

namespace CodeFirstApp.Controllers
{
    [Authorize]
    public class AlunosController : Controller
    {
        private EscolaContext db = new EscolaContext();

        // GET: Alunos
        public ActionResult Index()
        {
```

Programação Web – 2023/2024 – LEI D-CE-PL – ISEC/IPC @JB

isec Engenharia DEIS - Instituto Superior de Engenharia de Coimbra

ASP.NET Identity: Autorize + Role

- Aplicar no controlador

```
using System.Data.Entity;
using System.Linq;
using System.Net;
using System.Web.Mvc;
using CodeFirstApp.Models;

namespace CodeFirstApp.Controllers
{
    [Authorize(Roles="Admin, Gestor")]
    public class AlunosController : Controller
    {

        // GET: Alunos
        public ActionResult Index()
        {
```

Programação Web - 2023/2024 – LEI D-CE-PL – ISEC/IPC @JB

isec Engenharia DEIS - Instituto Superior de Engenharia de Coimbra

ASP.NET Identity: AllowAnonymous

```
using CodeFirstApp.Models;
using System;
using System.Linq;
using System.Web.Mvc;

namespace CodeFirstApp.Controllers
{
    [AllowAnonymous]
    public class HomeController : Controller
    {
        public ActionResult Index()
        {
            return View();
        }
```

Programação Web – 2023/2024 – LEI D+CE+PL – ISEC/IPC @JB

isec Engenharia DEIS - Instituto Superior de Engenharia de Coimbra

Restringir Ações a Utilizadores

```
[Authorize(Users="joao")]
public ActionResult Create()
{
   return View();
}
```

Programação Web – 2023/2024 – LEI DFCE+PL – ISEC/IPC @JB

isec Engenharia DEIS - Instituto Superior de Engenharia de Coimbra

- Como especificar utilizadores / perfis iniciais?

Perfis e Roles Iniciais

```
using Microsoft.AspNetCore.Identity;
using AutoWEB.Models;
using System;

namespace AutoWEB.Data
{
    4 referências
    public enum Roles
    {
        Admin,
        Formador,
        Cliente
    }
    1 referência
    public static class Inicializacao
    {
        1 referência
        public static async Task CriaDadosIniciais(UserManager<ApplicationUser> userManager, RoleManager<IdentityRole> roleManager)
        {
            //Adicionar defaut Roles
            await roleManager.CreateAsync(new IdentityRole(Roles.Admin.ToString()));
            await roleManager.CreateAsync(new IdentityRole(Roles.Formador.ToString()));
            await roleManager.CreateAsync(new IdentityRole(Roles.Cliente.ToString()));

            //Adicionar Default User - Admin
            var defaultUser = new ApplicationUser
            {
                UserName = "admin@localhost.com",
                Email = "admin@localhost.com",
                PrimeiroNome = "Administrador",
                UltimoNome = "Local",
                EmailConfirmed = true,
                PhoneNumberConfirmed = true
            };
            if (userManager.Users.All(u => u.Id != defaultUser.Id))
            {
                var user = await userManager.FindByEmailAsync(defaultUser.Email);
                if (user == null)
                {
                    await userManager.CreateAsync(defaultUser, "Is3C..00");
                    await userManager.AddToRoleAsync(defaultUser, Roles.Admin.ToString());
                }
            }
        }
    }
}
```


isec Engenharia DEIS - Instituto Superior de Engenharia de Coimbra

Restringir Ações

- Criar novas vistas quando é necessário restringir muitas partes de uma vista

```
public ActionResult Index()
{
    if (User.IsInRole("ManipulaAlunos"))
        return View(db.Alunos.ToList());
    return View("VistaApenasLeitura",db.Alunos.ToList());
}
```

Programação Web – 2023/2024 – LEI D+CE+PL – ISEC/IPC @JP

Restrições / Permissão

```
    // GET: Cursos/Edit/5
     [Authorize(Roles = "Admin")]
     public async Task<IActionResult> Edit(int? id)
     {
        if (id == null || _context.Cursos == null)
        {
           return NotFound();
        }
...
...
```

```
@if (@User.IsInRole("Admin"))
  {
     <small><a class="btn btn-primary btn-sm" asp-action="Create">Adicionar</a></small>
  }
```


isec Engenharia DEIS - Instituto Superior de Engenharia de Coimbra

Restrições / Permissão

- As restrições de acesso de funcionalidades a determinados utilizadores devem ser implementadas:
  - No inicio do Controller quando a restrição/permissão seja global a todo o Controller
  - Em determinadas Action dos Controllers quando a restrição/permissão se refira só a essa Action
  - Nas Views, restringido/permitindo o acesso a Links, Botões e etc …
  - Nos menús, nos ficheiros Layout, para restringir/permitir a visualização das opções do menú

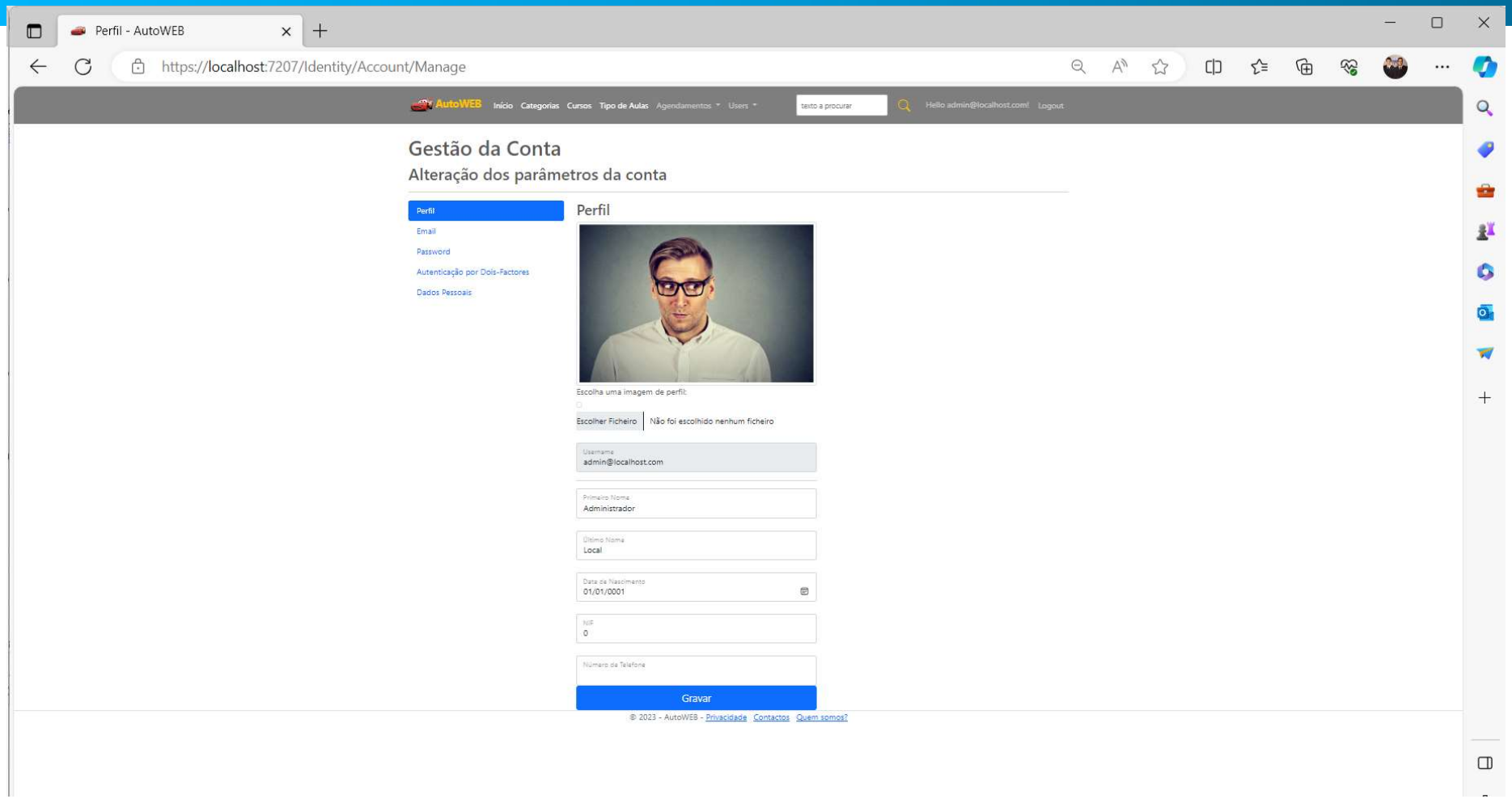
Programação Web – 2023/2024 – LEI D+CE, PL – ISEC/IPC @JB

Adicionar dados ao *Perfil*

- Adicionar propriedade PrimeiroNome

```
public class ApplicationUser : IdentityUser
    {
        public string PrimeiroNome { get; set;
}
        ...
```

- Criar Migração
- Atualizar a base de dados
- Alterar o Register.cshtml correspondente para aceitar a nova propriedade
- Alterar o Register.cshtml.cs para gravar esta propriedade
  - Razor Page: Register.cshtml + Register.cshtml.cs

Programação Web – 2023/2024 – LEI D+CE+PL – ISEC/IPC @JB

Adicionar dados ao *Perfil*