



Framework.NET – C#

Programação III

Prof. Edson Mota, PhD, MSc, PMP

Framework.NET



- Consiste em uma **plataforma** de **desenvolvimento** e **cross-plataforma** utilizada para construir os mais diversos tipos de aplicações.
- .NET oferece suporte a **múltiplas linguagens**, editores e bibliotecas para construir aplicações web, mobile, games, entre outras.
- Algumas linguagens suportadas são – C#, VB.NET, Python, F#
- A principal IDE para programação neste framework é o **Visual Studio.NET**

Algumas definições



BIBLIOTECAS

Conjunto de recursos independentes e complementares que podem ser usados em um projeto.

Normalmente, compostas de um conjunto de arquivos compilados.

Geralmente, oferecem funcionalidades específicas, como processamento de imagem, criptografia, comunicação...

Podem ser adicionadas ou removidas de um projeto, conforme necessário.

FRAMEWORKS

Conjunto completo de ferramentas, bibliotecas e padrões para desenvolvimento de aplicativos.

Fornecem uma estrutura de trabalho para o desenvolvimento de novos projetos.

Oferecem funcionalidades mais abrangentes do que bibliotecas, como gerenciamento de banco de dados, autenticação, tratamento de erros, etc.

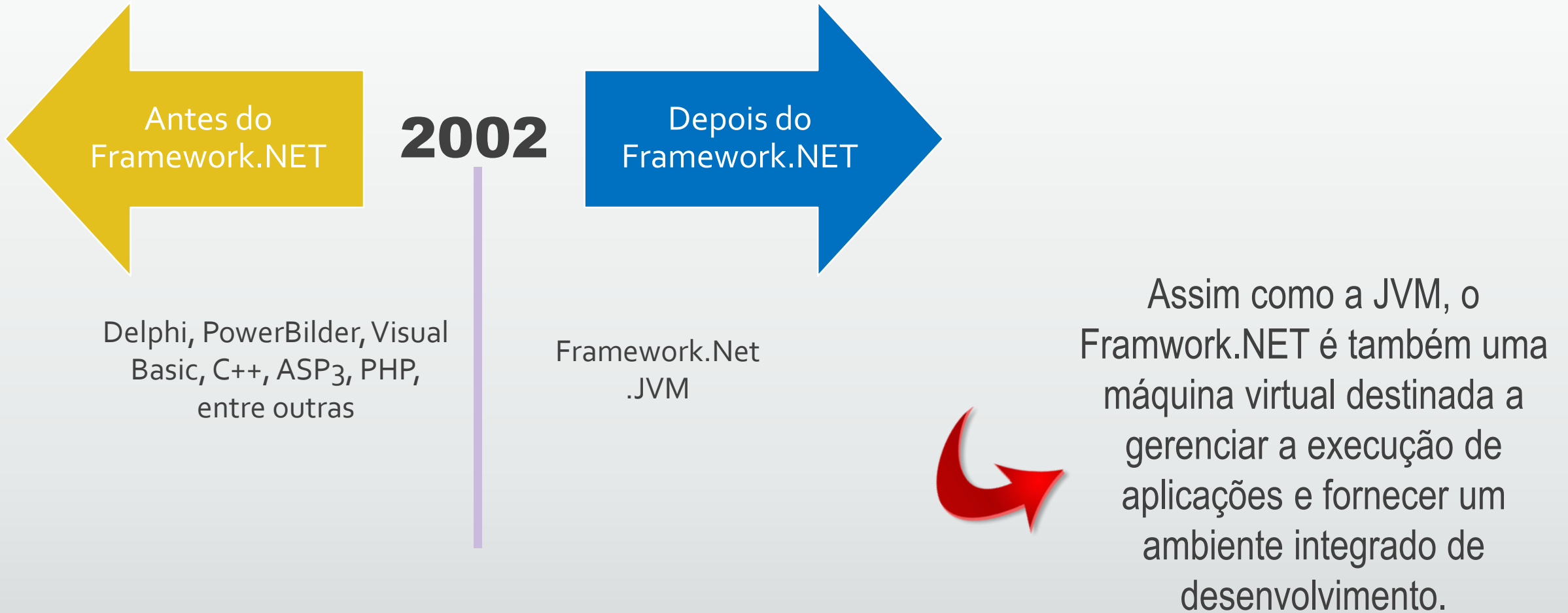
Requerem mais conhecimento e compreensão por parte do desenvolvedor para ser usado com eficácia.

Característica da Plataforma

- Suporte a diferentes Linguagens
 - Todas as linguagens .NET utilizam a mesma estrutura de execução
- Suporte a plataforma cruzada
 - Permite o desenvolvimento de aplicações para diferentes ambientes (web, mobile, etc)



Uma Mudança no Processo



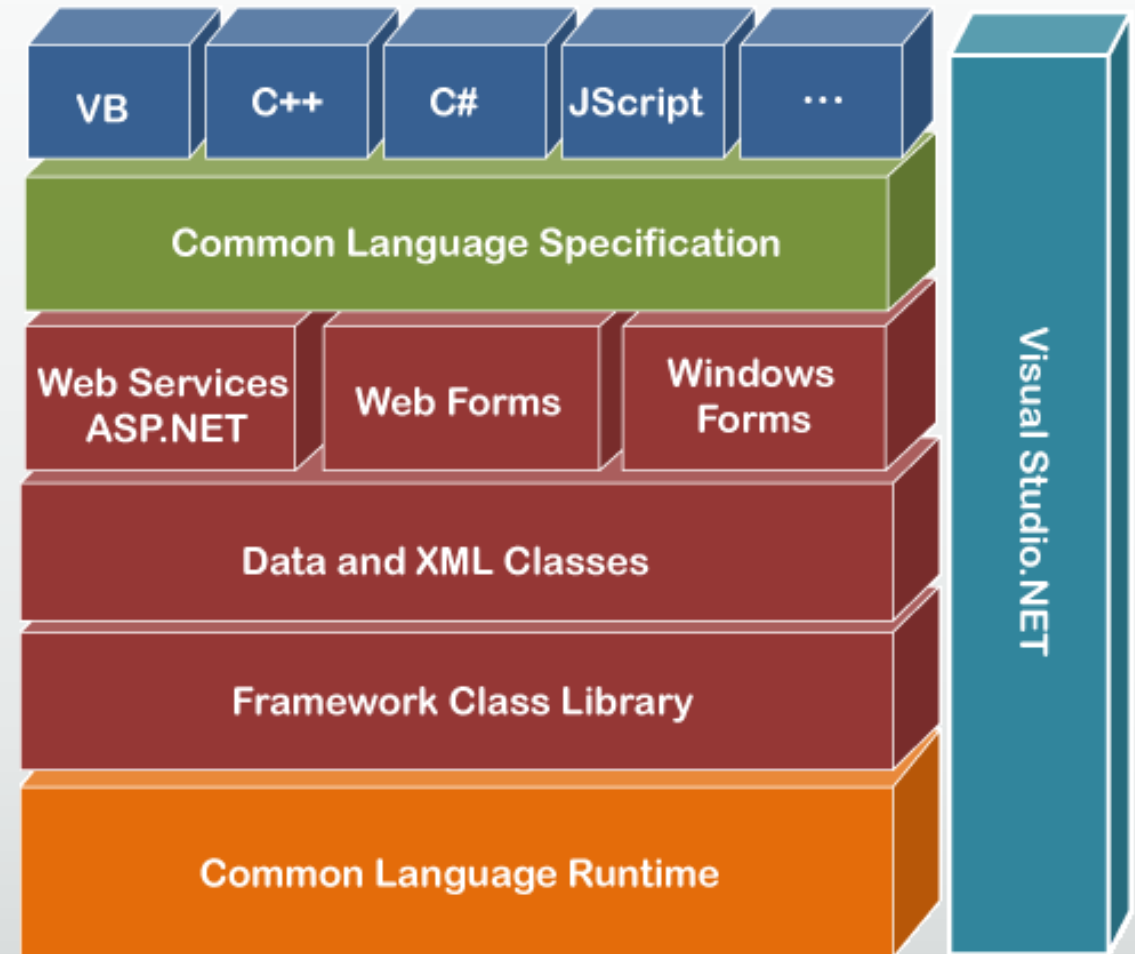
Framework.NET

- Surgiu em 2002 com a proposta de se tornar uma plataforma única para desenvolvimento e execução de aplicações.
- Desde então, o Framework vem crescendo, realizando importantes contribuições para o desenvolvimento de software.
- Atualmente, o framework se encontra na versão 4.8



Framework.NET - Arquitetura

- A arquitetura fundamental do Framework.NET **incorpora um conjunto de camadas e componentes** com o objetivo de garantir a unificação dos códigos e um formato eficiente e próximo a linguagem de máquina.
- Entre eles estão:
 - **Common Language Infraestrutura (CLI)**
 - **Common Language Specification (CLS)**
 - **Common Language Runtime (CLR)**
 - Bibliotecas de uso geral



Common Language Infrastructure (CLI)



- **Padrão aberto** que descreve códigos executáveis (*runtime*).
 - Esse padrão é adotado pelo **CLR** (implementação CLI/Microsoft).
 - A **especificação define** o **ambiente** que permite que **múltiplas linguagens** de alto nível sejam utilizadas em diferentes plataformas evitando problemas em função da arquitetura. (Mono ou CLR)
 - Entre as linguagens , estão: **C#**, VB.NET, Python, J#, entre outras.

Common Language Specification (CLS)



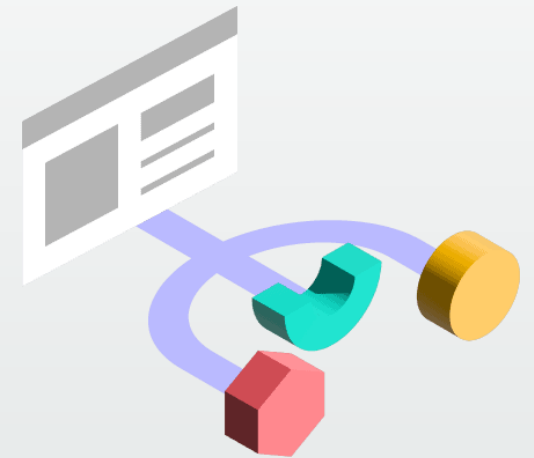
- É responsável por **especificar o conjunto de regras que precisam ser satisfeitas** por todas as **linguagens compiladas** por meio do Common Language Runtime (CLR).
- Essa característica ajuda no suporte a múltiplas linguagens.
- **Algumas regras são:**
 - Coerência na representações de textos (string);
 - Representações de enumerations;
 - Definição de membros estáticos;
 - Entre outras...
- O conjunto de regras definidas no CLS **funciona como um guia** com o objetivo de **permitir a comunicação** com os **módulos** de gerenciamento **independentemente** da **linguagem** adotada.



Common Intermediate Language (CIL)



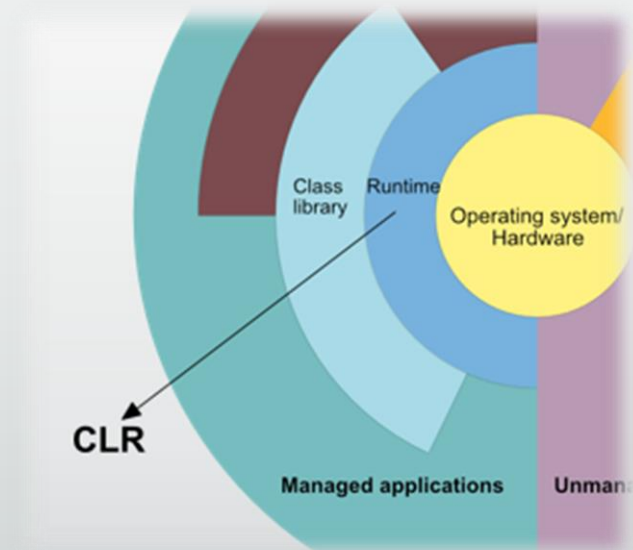
- Um CLR ajuda a converter um código-fonte em **ByteCode** que é conhecido como:
 - CIL (*Common Intermediate Language*)
 - MSIL (*Microsoft Intermediate Language*)
- Depois de converter em um **ByteCode**, o CLR usa um **compilador JIT** em tempo de execução que ajuda a converter um código CIL ou MSIL em um **código nativo**.



Common Language Runtime (CLR)



- Consiste na **máquina virtual** e o componente principal do Framework .NET
- É responsável por gerenciar a execução dos programas.
- CLR é um **ambiente de tempo** de execução para Native Code em .NET
- Essa **conversão é alcançada** pelo CIL (Common Intermediate Language)
- O CLR realiza diferentes tarefas para gerenciar a execução das aplicações .NET, **entre elas**:
 - ✓ Gerenciamento de Memória Automática;
 - ✓ Coletor de Lixo (Garbage Collector);
 - ✓ Segurança de acesso ao código;
 - ✓ Verificação de código;
 - ✓ JIT – Just in Compiler (Compilador de tempo de execução).



Bibliotecas Compartilhadas

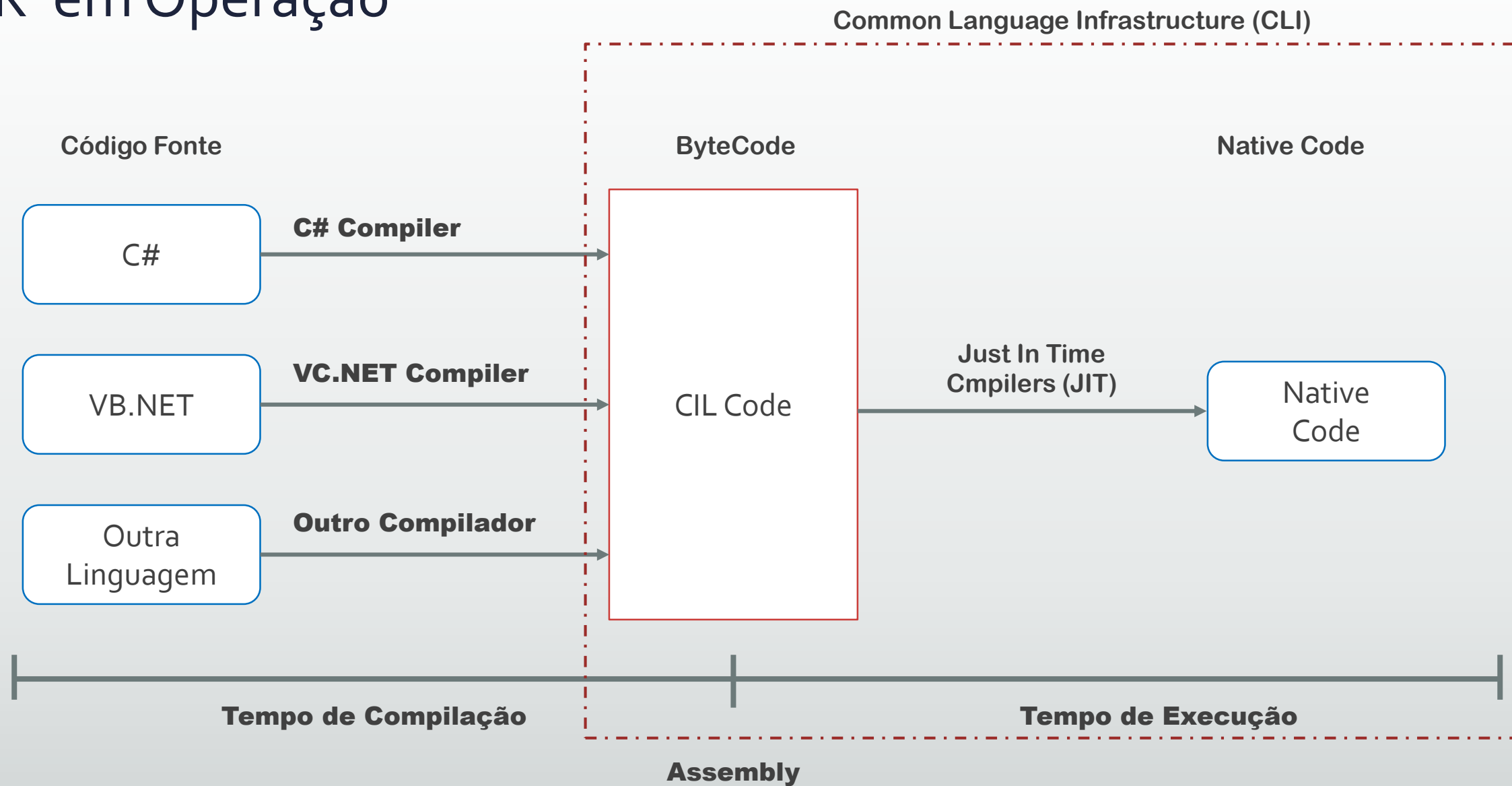
- **Basic Class Library (BCL)**

- Tipos intrínsecos (string, datetime, ect)
- Além de outras BaseClass (Ex: FileStream)
- Alinha-se às especificações ECMA (European Computer Manufacturers Association)
 - Padrões internacionais que definem a arquitetura, a linguagem e as bibliotecas comuns do .NET Framework.

- **Framework Class Library (FCL)**

- Suporte para entrada/saída, gráficos, segurança, acesso a dados, rede, criptografia, threads, entre outros.
- System.Web, System.Data, System.Collection, System.xml, etc.

CLR em Operação

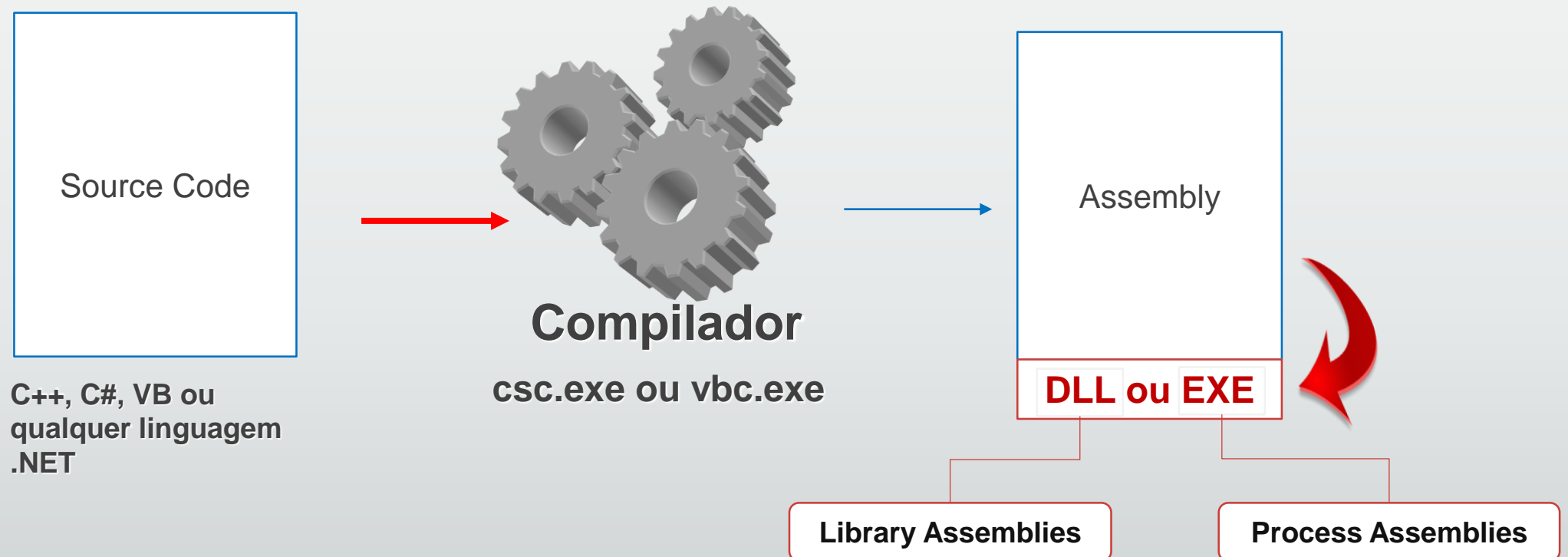


Assembly

- O **compilador**...
 - Converte o código da aplicação utilizando o *padrão Common Intermediate Language* (CIL)
 - Adiciona **metadados** requeridos
 - Dados como: Nomes, Namespaces, Assinaturas, Atributos, etc
 - Então, têm-se um **Assembly**
 - O **Assembly** consiste em uma **pequena unidade de código** que contém a **lógica** e tudo que é necessário para executar a aplicação.
 - Consiste na **unidade fundamental para implantação**, versionamento, reuso, segurança.
 - Trata-se de um ou mais arquivos (***.dll** ou ***.exe**) e representam uma coleção de tipos e recursos que trabalham juntos para compor a lógica da aplicação.

Framework .NET

- Em resumo, quando o programa .NET é compilado, ele gera um metadado com **Common Intermediate Language** que é armazenado em um arquivo chamado **Assembly**.



ASP.NET Core

ASP.NET Core

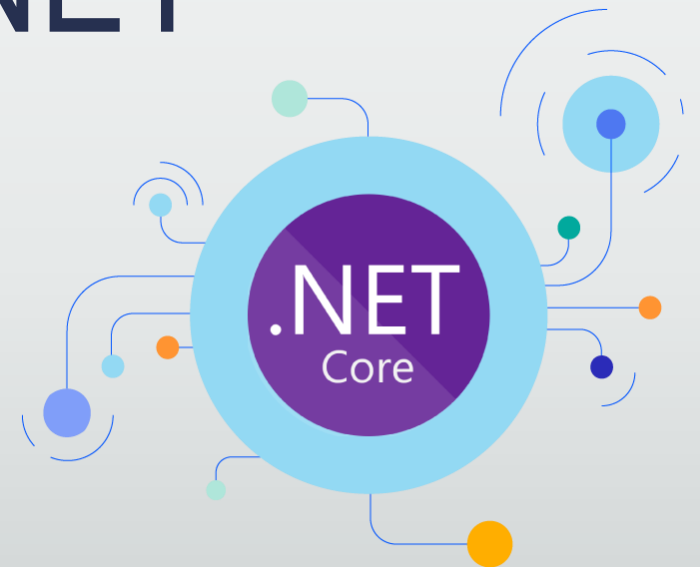


- Consiste em uma infraestrutura de desenvolvimento web modular e **multi-plataforma** desenvolvida e mantida pela Microsoft.
- Embora o ASP.Net Core e o Framework .NET compartilhem algumas tecnologias centrais, como a linguagem, bibliotecas, entre outros,
- **O ASP.NET Core é uma plataforma de desenvolvimento web independente** que pode ser executada em vários sistemas operacionais, incluindo Windows, Linux e macOS.

ASP.NET Core: <https://docs.microsoft.com/en-us/aspnet/core/>

.NET Framework: <https://docs.microsoft.com/en-us/dotnet/framework/>

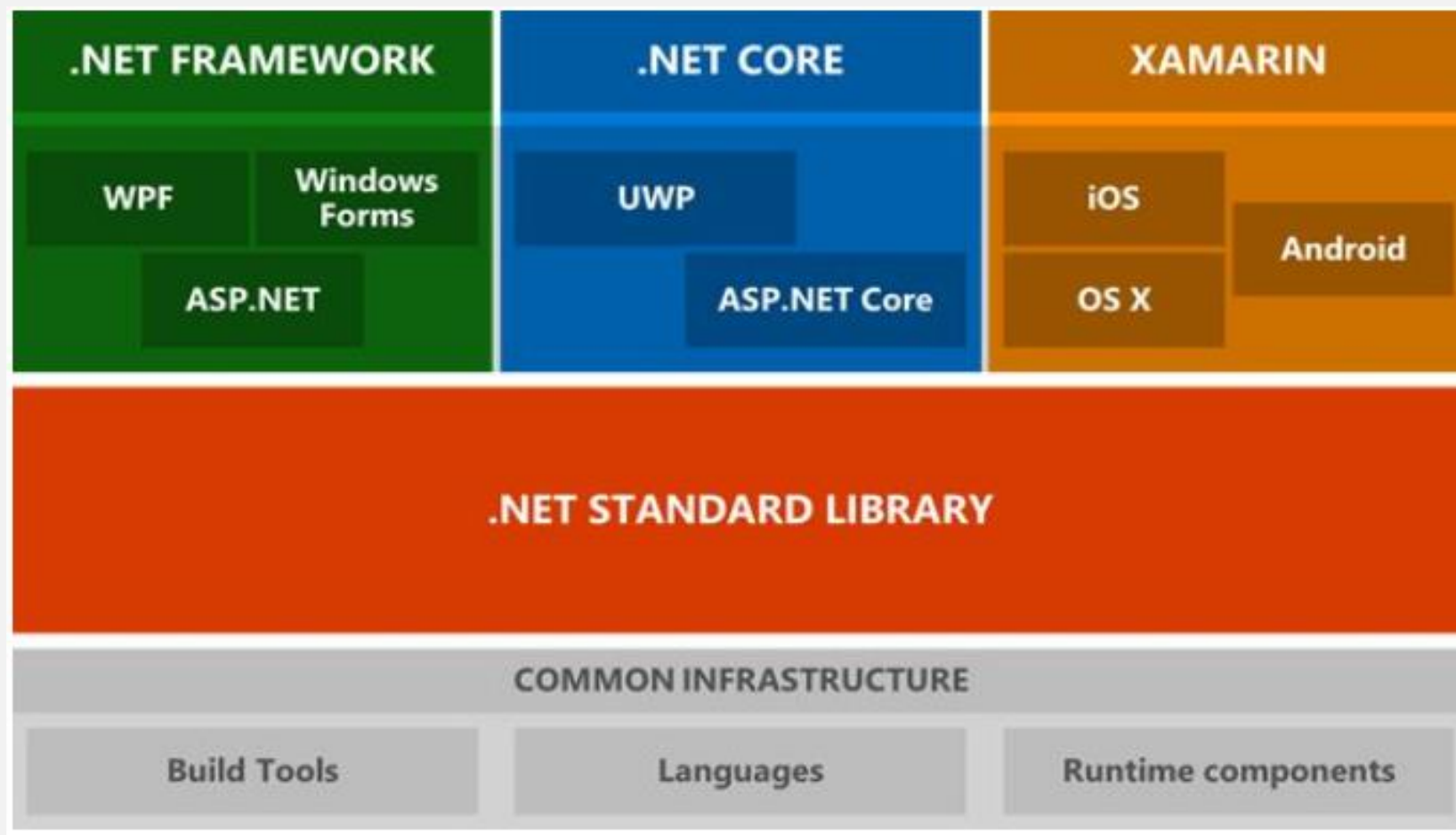
ASP.NET Core é um **Redesenho** do ASP.NET





- O projeto se iniciou como uma **extensão** do framework .NET;
- Essa arquitetura era suportada por diferentes ambientes de desenvolvimento;
 - O .NET Framework para **Windows**;
 - O .NET Core para Windows, **Linux** e **Mac**;
 - O **Xamarin** para desenvolvimento mobile **iOs**, **Android** e **UWP**.

.NET < 4.5

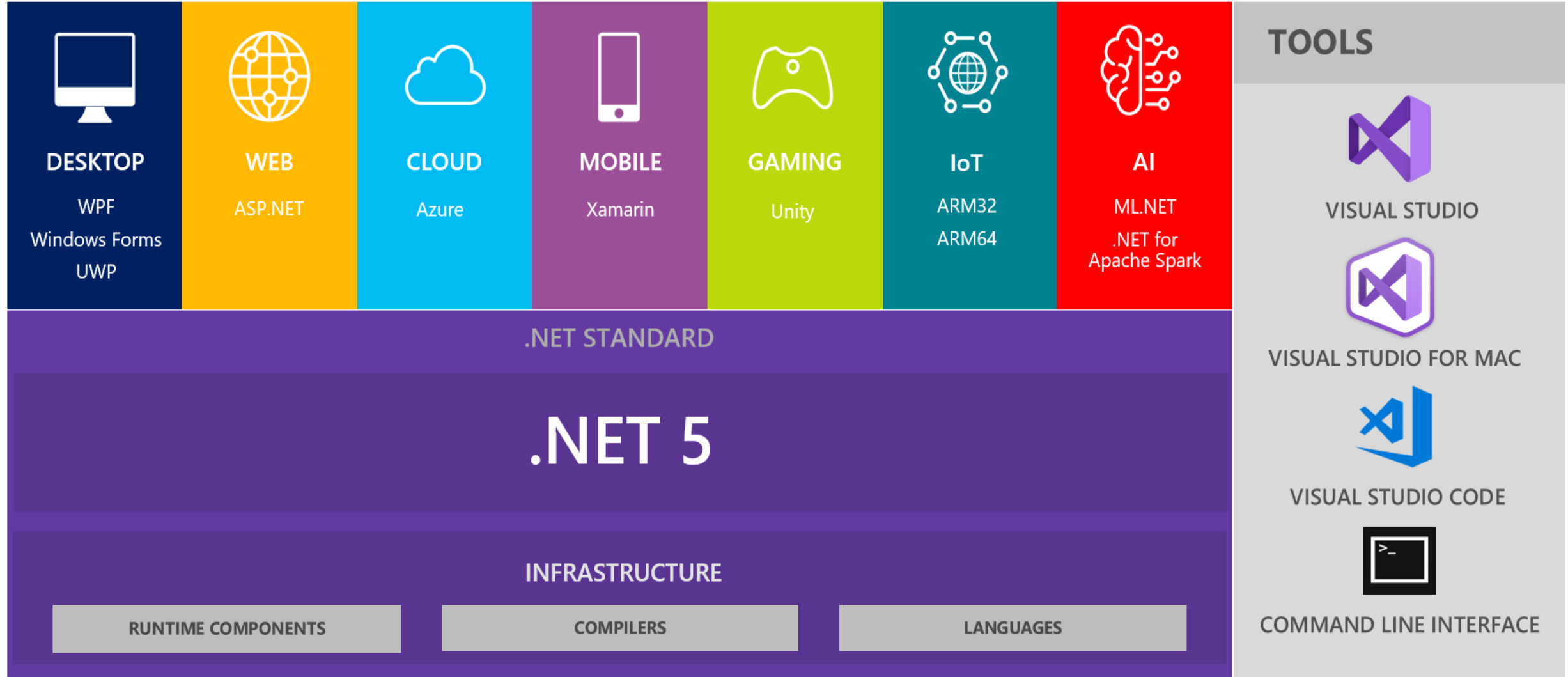


- Até novembro de 2020, essa arquitetura do Framework .NET foi amplamente conhecida e divulgada.
- Nesse cenário, eram requeridas diferentes plataformas para operar os diferentes cenários.
 - Mono, .NET Framework, .NETCore, Xamarin
- Essa estrutura motivou a criação de um conjunto comum de bibliotecas.

Bem-Vindo .NET 5.0

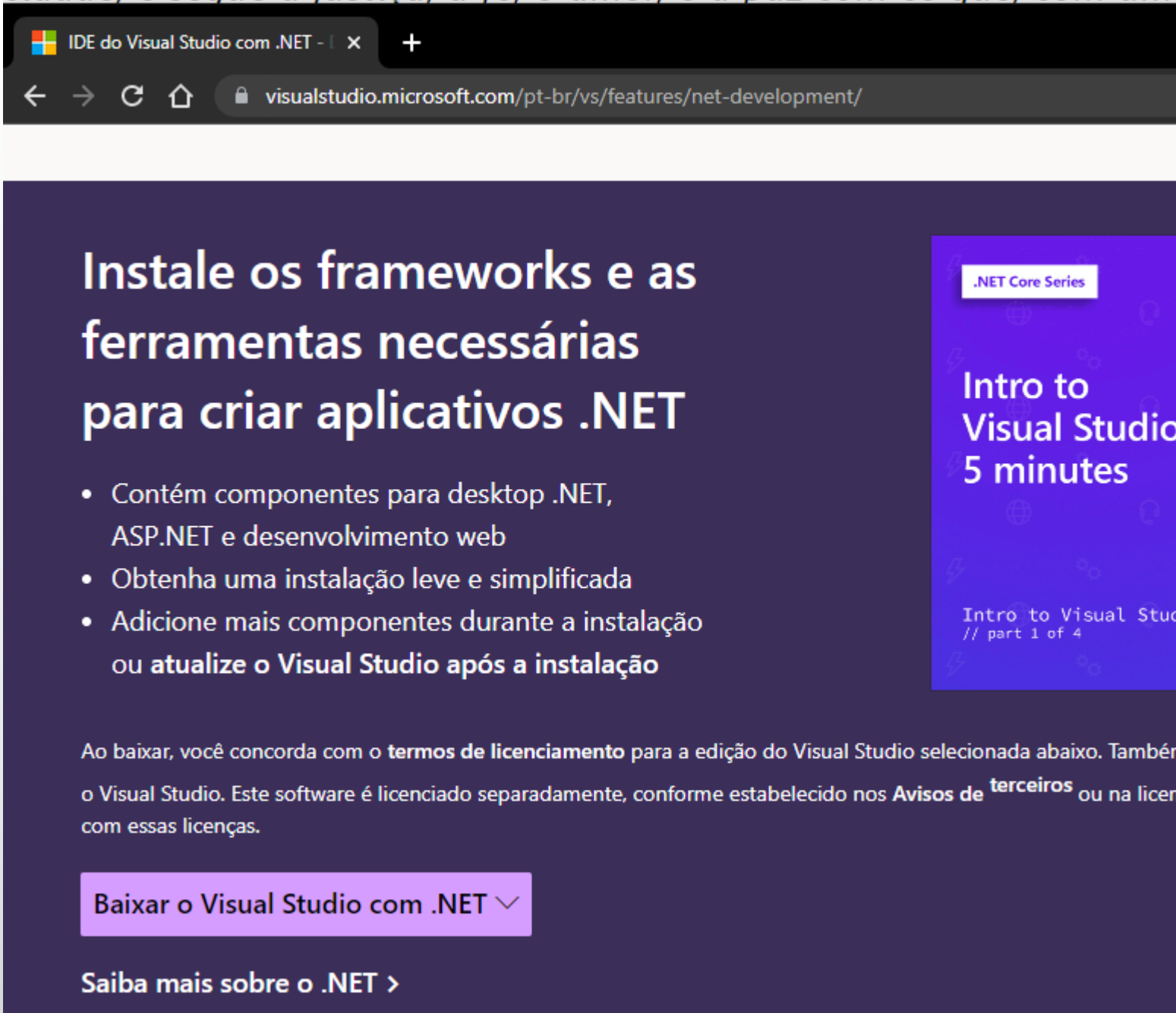
- A versão 5.0 do **.NET** propõe uma **plataforma unificada**, capaz de fornecer suporte aos mais variados tipos de implementação.
- Nessa versão, foi proposta a criação de uma **Basic Class Library (BCL) compartilhada**, utilizando diferentes VMs de tempo de execução como:
 - **MonoVM** (código aberto e multiplataforma), **CoreCLR** (Microsoft).
- A abordagem se apoia na ideia de um **produto único** no qual é possível escolher apenas os aspectos da plataforma relevantes para o seu projeto.

.NET – A unified platform



Visual Studio .NET

- Faça o download do Visual Studio NET
- Vamos utilizar a versão Educacional
- Ela é completa para fins acadêmicos



The screenshot shows a web browser window with the address bar displaying 'visualstudio.microsoft.com/pt-br/vs/features/net-development/'. The page has a dark blue background with white text. The main heading is 'Instale os frameworks e as ferramentas necessárias para criar aplicativos .NET'. Below this, there is a bulleted list of features. To the right, there is a purple sidebar with the text '.NET Core Series' and 'Intro to Visual Studio 5 minutes'. At the bottom, there is a large blue button with the text 'Baixar o Visual Studio com .NET' and a dropdown arrow. Below the button, there is a link 'Saiba mais sobre o .NET >'. A paragraph of text is also present, mentioning terms of service and licensing.

IDE do Visual Studio com .NET - X

visualstudio.microsoft.com/pt-br/vs/features/net-development/

Instale os frameworks e as ferramentas necessárias para criar aplicativos .NET

- Contém componentes para desktop .NET, ASP.NET e desenvolvimento web
- Obtenha uma instalação leve e simplificada
- Adicione mais componentes durante a instalação ou **atualize o Visual Studio após a instalação**

Ao baixar, você concorda com o **termos de licenciamento** para a edição do Visual Studio selecionada abaixo. Também o Visual Studio. Este software é licenciado separadamente, conforme estabelecido nos **Avisos de terceiros** ou na licença com essas licenças.

Baixar o Visual Studio com .NET ▼

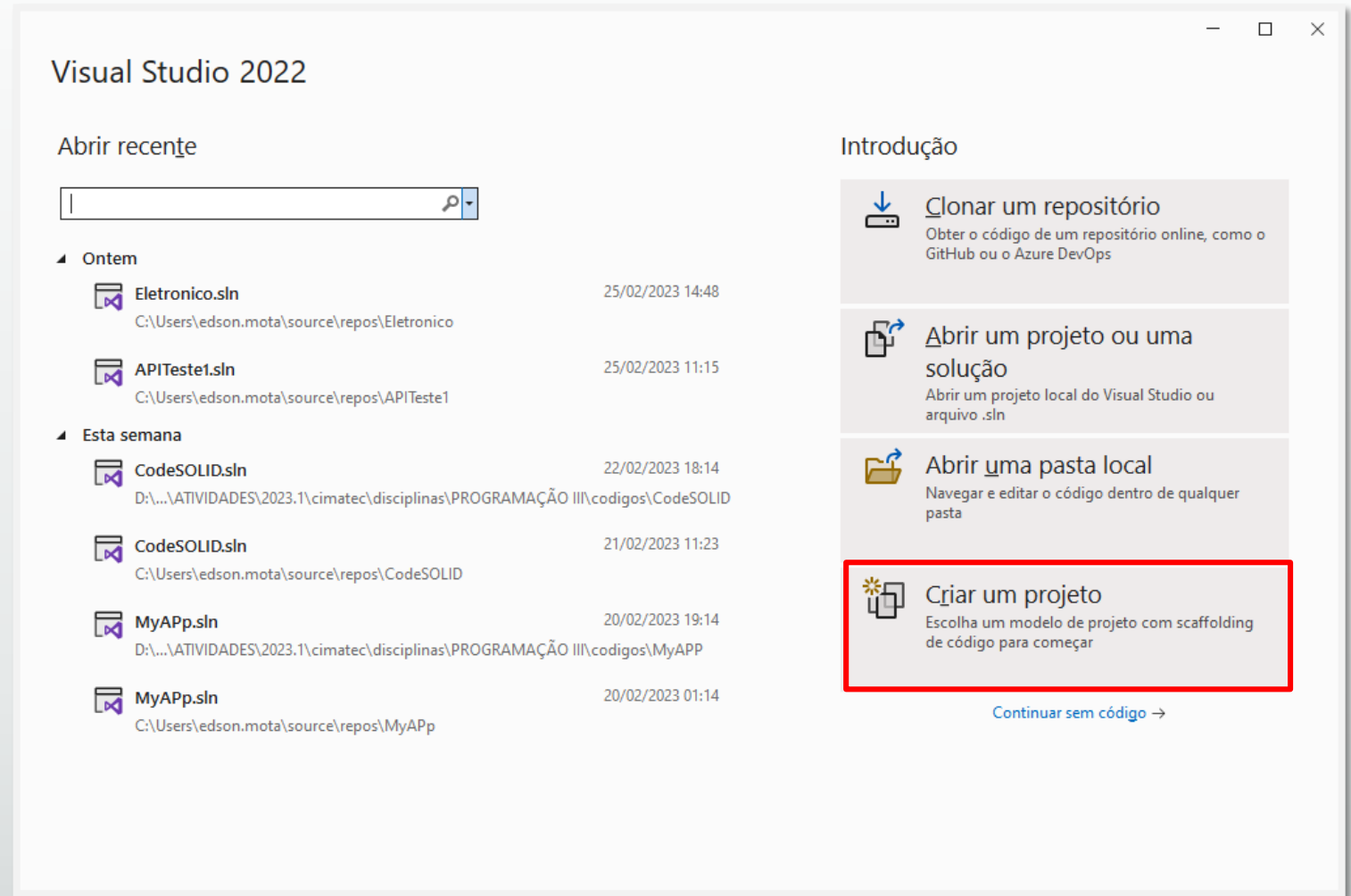
Saiba mais sobre o .NET >

.NET Core Series

Intro to Visual Studio 5 minutes


Intro to Visual Studio // part 1 of 4

Abrindo o Visual Studio



Criar um novo projeto

Modelos de projeto recentes

-  Aplicativo do Console C#
-  API Web do ASP.NET Core C#

Pesquisar modelos (Alt+S)

Todos os idiomas

Todas as plataformas

Todos os tipos de projeto



Aplicativo do Console

Um projeto para criar um aplicativo de linha de comando que pode ser executado no .NET no Windows, Linux e macOS

C#

Linux

macOS

Windows

Console



Aplicativo Web ASP.NET Core

Um modelo de projeto para criar um aplicativo ASP.NET Core com conteúdo de exemplo ASP.NET Core Razor Pages

C#

Linux

macOS

Windows

Nuvem

Serviço

Web



Aplicativo Blazor Server

Um modelo de projeto para criar um aplicativo Blazor Server que é executado do lado do servidor em um aplicativo ASP.NET Core e manipula as interações com o usuário em uma conexão SignalR. Esse modelo pode ser usado para aplicativos Web com UIs (interfaces do usuário) completas e dinâmicas.

C#

Linux

macOS

Windows

Blazor

Nuvem

Web



API Web do ASP.NET Core

Um modelo de projeto para criar um aplicativo ASP.NET Core com um Controlador de exemplo para um serviço HTTP RESTful. Esse modelo também pode ser usado para Controladores e Exibições do ASP.NET Core MVC.

C#

Linux

macOS

Windows

Nuvem

Serviço

Web

WebAPI



Biblioteca de Classes

Voltar

Próximo

Configurar seu novo projeto

Aplicativo do Console

C#

Linux

macOS

Windows

Console

Nome do projeto

DotNetExemplo1

Local

C:\Users\edson.mota\source\repos

Nome da solução ⓘ

DotNetExemplo1

☐ Colocar a solução e o projeto no mesmo diretório

Voltar

Próximo



Informações adicionais

Aplicativo do Console

C#

Linux

macOS

Windows

Console

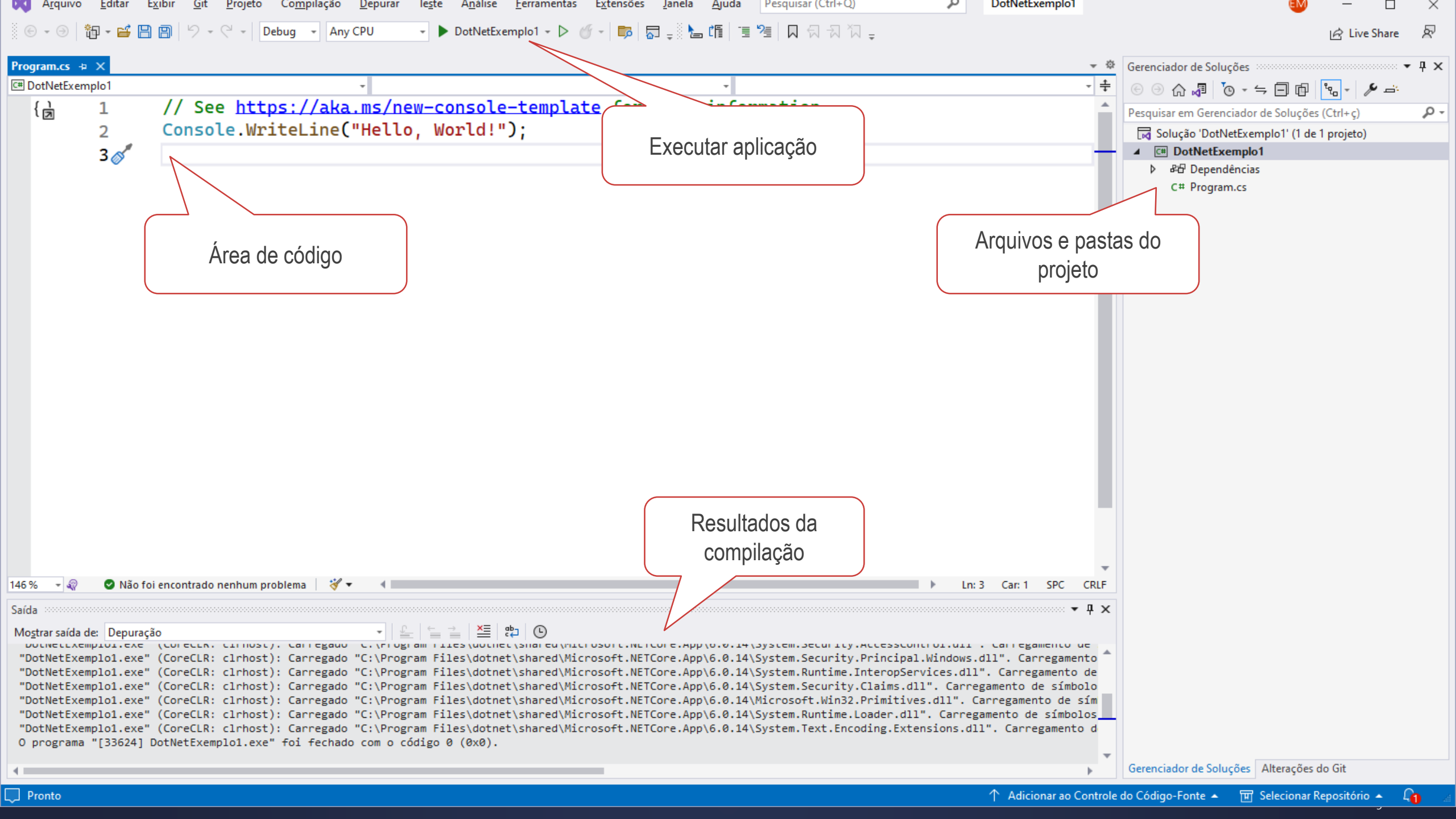
Estrutura ⓘ

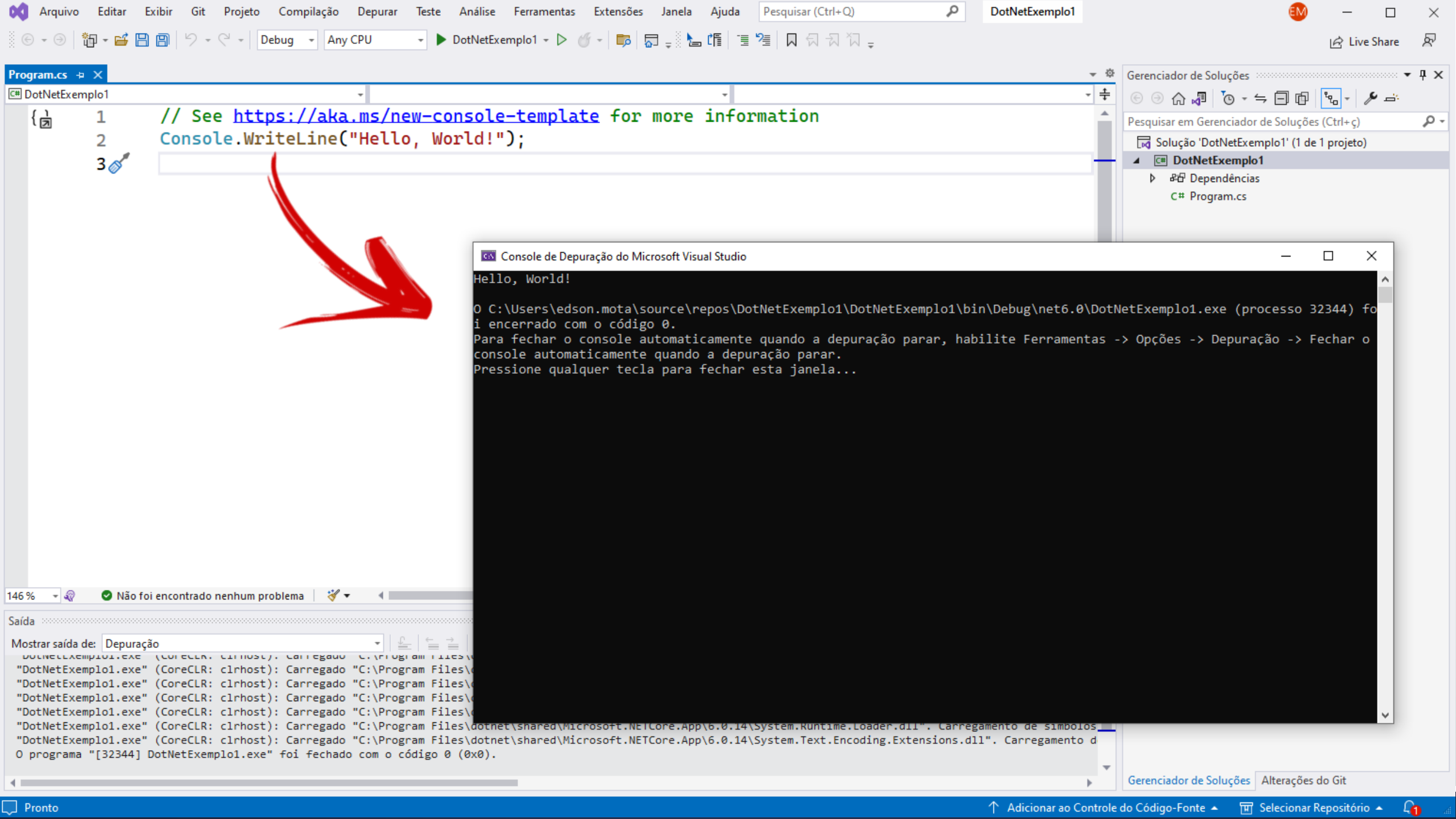
.NET 6.0 (Suporte de Longo Prazo)

☐ Não use instruções de nível superior ⓘ

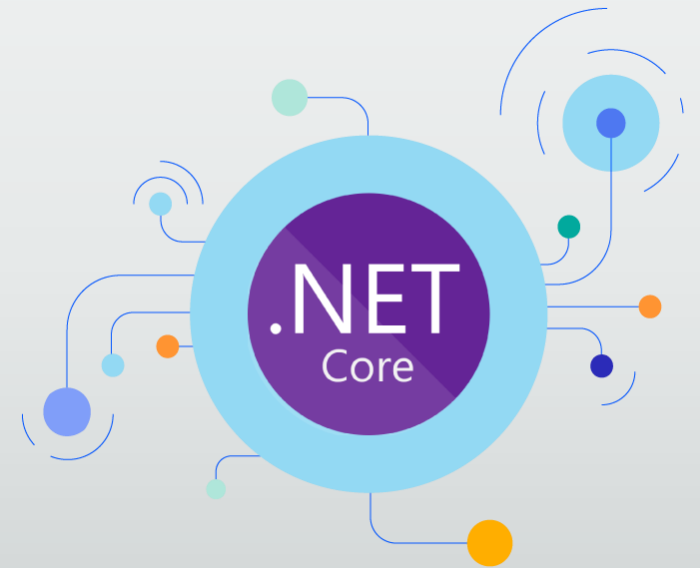
Voltar

Criar





Alguns recursos que podem ajudar



DEBUG

The screenshot shows the Visual Studio IDE with a C# project named 'DotNetExemplo1'. The code in 'Program.cs' is as follows:

```
1  
2 int a = 5;  
3 int b = 10;  
4 int result = 0;  
5  
6 result = a + b;  
7  
8 Console.WriteLine($"O Resultado é: {result}");  
9  
10  
11  
12  
13  
14  
15
```

The program is running, and the execution has stopped at line 6, 'result = a + b;'. A red dot on the left margin indicates the current execution point. A tooltip shows the variable 'result' with the value '15'. The status bar at the bottom indicates '≤ 1ms decorridos'.

Annotations and actions shown:

- BreakPoint:** Passa pelo código executando uma instrução por vez. (Indicated by a red dot on the left margin next to line 6).
- Verifica o resultado da variável em tempo de execução!** (Points to the variable 'result' in the tooltip).
- Salta para a próxima instrução** (Points to the 'Next Instruction' button in the top toolbar).

IMMEDIATE WINDOW

The screenshot displays the Visual Studio IDE with the **Depurar** (Debug) menu open. The **Janelas** (Windows) option is highlighted in the menu. In the **Imediata** (Immediate) window, the expression `result = a + b;` is entered and highlighted in red. The code editor on the left shows a C# program with the following code:

```
1  
2 int a = 5;  
3 int b = 10;  
4 int result = 0;  
5  
6 result = a + b;  
7  
8 Console.WriteLine(result);  
9  
10  
11  
12  
13  
14  
15  
16
```

The **Imediata** window is also highlighted in red. The **Depurar** menu is highlighted in blue. The **Imediata** window is highlighted in red.

Arquivo Editar Exibir Git Projeto Compilação Depurar Teste Análise Ferramentas Extensões Janela Ajuda Pesquisar (Ctrl+Q)

Processo: [15616] DotNetExemplo1.exe Eventos de Ciclo de Vida Thread: [27352] Thread Principal Registro de Ativação: Program.<Main>\$

Program.cs

```
1  
2 int a = 5;  
3 int b = 10;  
4 int result = 0;  
5  
6 result = a + b;  
7  
8 Console.WriteLine($"O Resultado é: {result}");  
9  
10  
11  
12
```

146 % Não foi encontrado nenhum problema Ln: 8

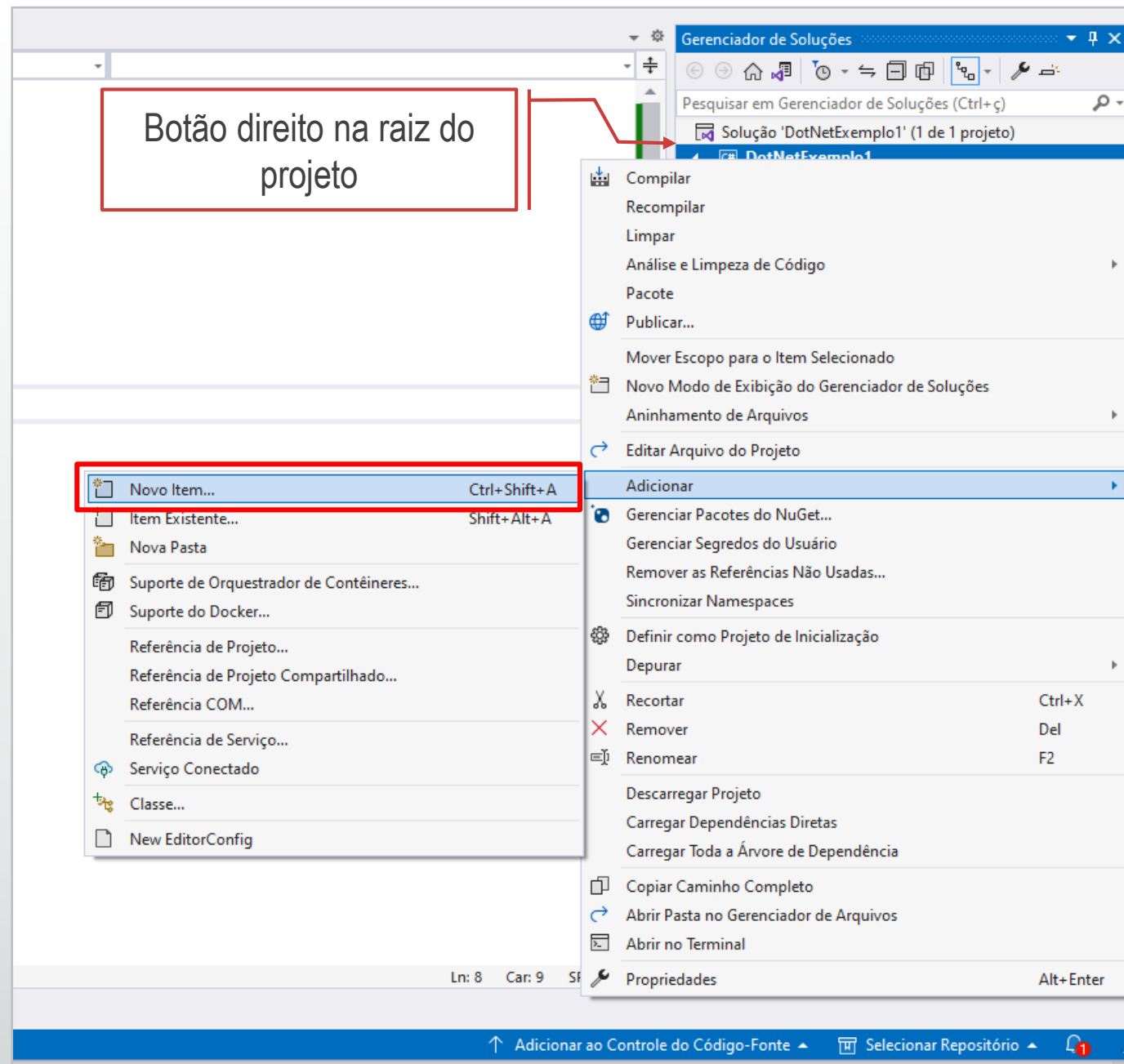
Janela Imediata
? result
15

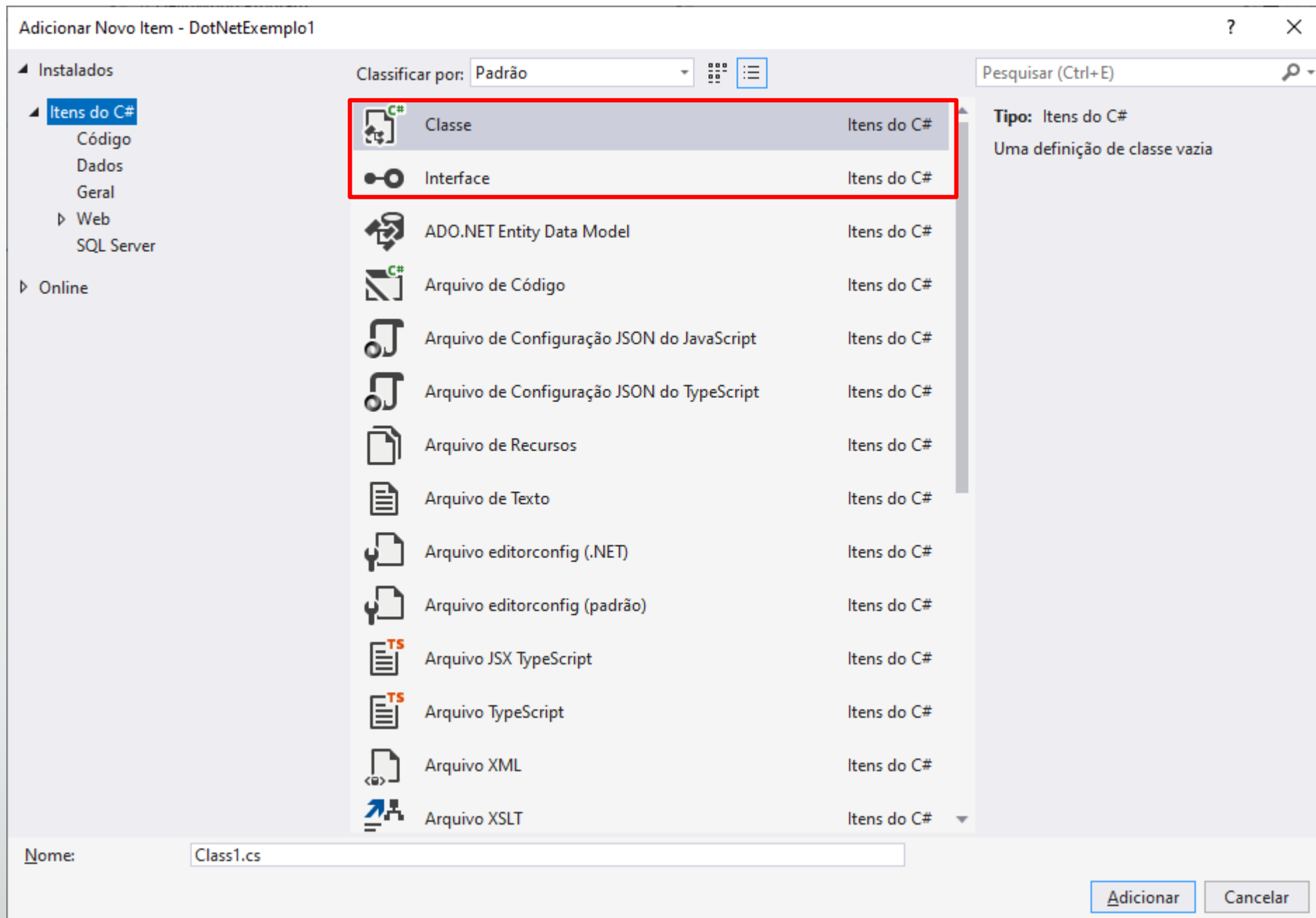
? Nome da variável
retorna o valor contido na mesma

Pilha de Chamadas Pontos de Interrupção Configurações de Exceção Janela Imediata Saída Automáticos Locais Inspeção 1

Classes e Interfaces

- A criação de classes e interfaces pode ser realizada diretamente na árvore do projeto.





C#

C#

- Uma linguagem de programação criada por Anders Hejlsberg
- Desenvolvida pela **Microsoft**
- Apresentada juntamente com a **Plataforma .NET**
- Algo como C++ e Java (combinados)



Anders Hejlsberg

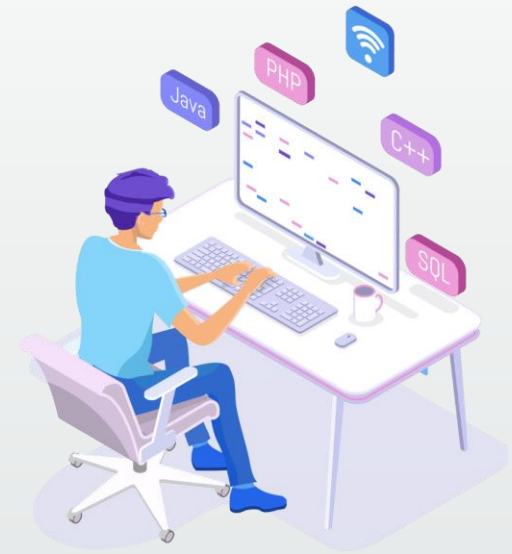
Características Gerais

- ✓ Orientada a objeto
- ✓ Possui alto nível de **abstração**
- ✓ Possui Coletor de Lixo
- ✓ Suporte à **tipagem dinâmica e estática**
 - **Dinâmica**: tipagem pode ocorrer em **tempo de execução**
 - **Estática**: definição dos tipos em **tempo de compilação**
- ✓ Vinculada ao Framework .NET (mas não exclusivamente)



Orientada a Componentes

- C# é a primeira linguagem “**orientada a componentes**” na família C/C++
- O que é um **componente**?
 - Um **módulo independente** de reutilização e implantação
 - **Mais granular** do que objetos
 - Inclui **várias classes**
 - Frequentemente **independente** da **linguagem**



Em geral, o **programador do componente** e o **usuário** **não se conhecem**, não trabalham para a mesma empresa e não falam o mesmo idioma.

Tipos de Dados e Objetos

- Um programa em **C#** é uma coleção de tipos
 - Classes, structs, enums, interfaces, delegate
- C# fornece ainda um conjunto de tipos pré-definidos
 - int, byte, char, string, object, etc;
- Naturalmente, **podemos criar nossos próprios tipos**

Tipos

▪ Value types

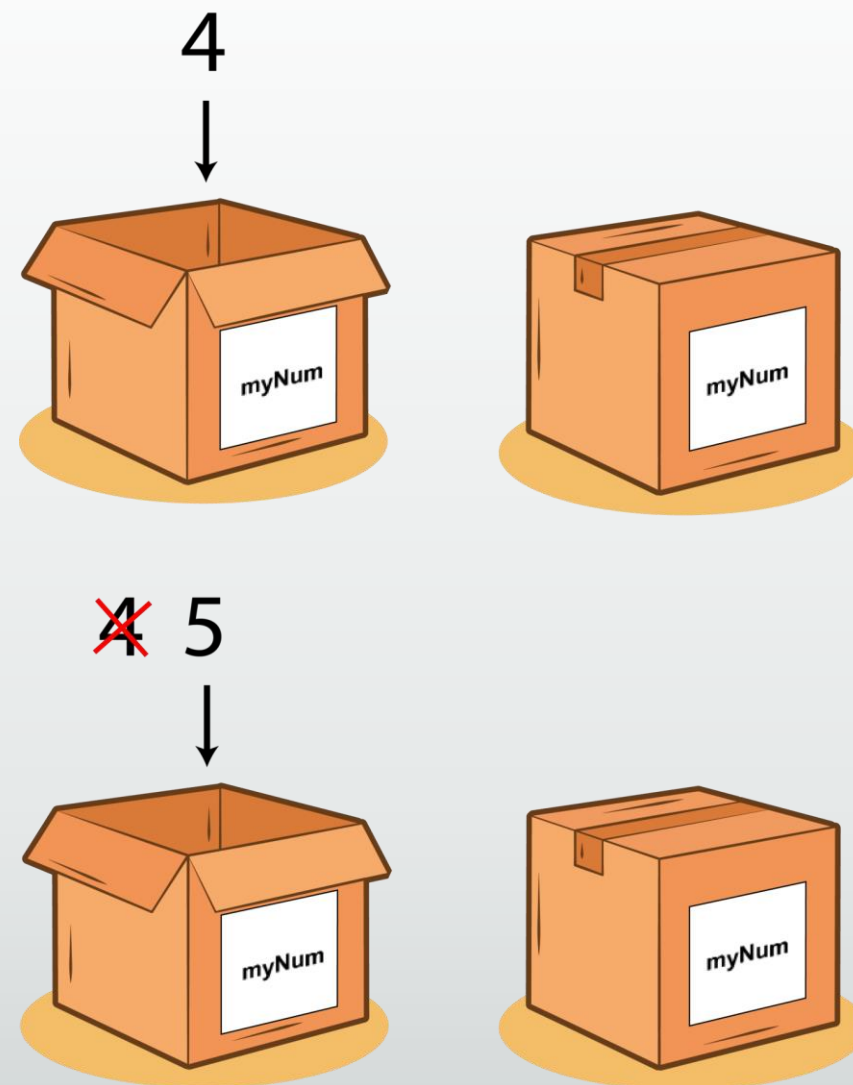
- Primitives
- Enums
- Structs

```
int i; float x;  
enum State { Off, On }  
struct Point {int x,y;}
```

▪ Reference types

- Root
- String
- Classes
- Interfaces
- Arrays
- Delegates

```
object  
string  
class Foo: Bar, IFoo {...}  
interface IFoo: IBar {...}  
string[] a = new string[10];  
delegate void Empty();
```



Tipos Definidos pelo Usuário

Enumerations	enum
Arrays	int[], string[]
Interface	interface
Reference type	class
Value type	struct
Function pointer	delegate

Enums em C# são fortemente tipados, o que significa que você não pode atribuir um int enum a um long, ou vice-versa, evitando alguns erros típicos de conversão.

Conversões de Tipos

▪ Conversões Implícitas

- Ocorrem automaticamente
- Sucesso é garantido

▪ Conversões Explícitas

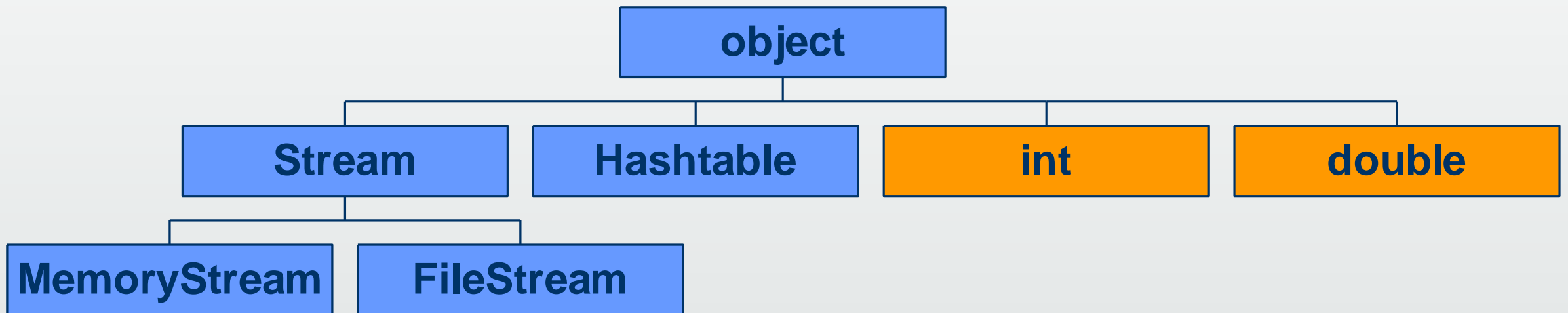
- Requerem um “cast”
- Podem não ser bem sucedidas

```
int x = 123456;
long y = x;           // implicit
short z = (short)x;   // explicit

double d = 1.2345678901234;
float f = (float)d;    // explicit
long l = (long)d;      // explicit
```

Tudo é um Objeto

- Todos os tipos herdam da superclasse object



Namespaces

- As classes são organizadas em **namespaces**

```
using System;

namespace HelloWorld
{
    0 referências
    public class Program
    {
        0 referências
        public static void Main(string[] args)
        {
            Console.WriteLine("Olá Mundo!");
        }
    }
}
```

Utilizamos a palavra chave **using** para referenciar um namespace


Console, por exemplo, pertence ao namespace "System"

Namespaces

- Em C#, um **namespace** é um mecanismo que **permite agrupar** classes, interfaces, enums e outros tipos relacionados em uma **unidade** lógica
 - Permitem a **melhor organização** do código
 - Contribuem para **segregar funções** similares facilitando a identificação e aumentando a **coesão**
 - Namespaces podem **abranger** diferentes **Assemblies**
 - Não há relação entre namespaces e estrutura de arquivo (ao contrário de Java)




```
namespace Pessoa
{
    2 referências
    public class Funcionario
    {
        int id;
        string? nome;
    }
}
```




```
using Pessoa;
```

```
0 referências
public class program
{
    0 referências
    public static void Main(string[] args)
    {
        Funcionario f = new Funcionario();
        Prestador p = new Prestador();
    }
}
```



```
namespace Pessoa
{
    2 referências
    public class Prestador
    {
        int id;
        string? nome;
    }
}
```



Obtendo Dados do Usuário

- Usando o método **ReadLine()** para obter dados do usuário

```
namespace HelloWorld
{
    0 referências
    public class Program
    {
        0 referências
        public static void Main(string[] args)
        {
            Console.WriteLine("Qual o seu nome?");
            Console.ReadLine();
        }
    }
}
```

enum (enumeração)

enum é um tipo de dado que define um conjunto de **constantes nomeadas** (System.Enum)

2 referências

```
enum DiasDaSemana : byte
```

```
{
```

```
    Domingo = 1,
```

```
    Segunda = 2,
```

```
    Terca = 3,
```

```
    Quarta = 4,
```

```
    Quinta = 5,
```

```
    Sexta = 6,
```

```
    Sabado = 7
```

```
}
```

0 referências

```
public static void Main(string[] args)
```

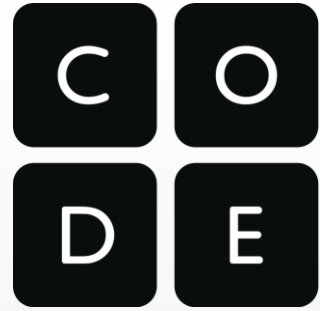
```
{
```

```
    DiasDaSemana hoje = DiasDaSemana.Terca;
```

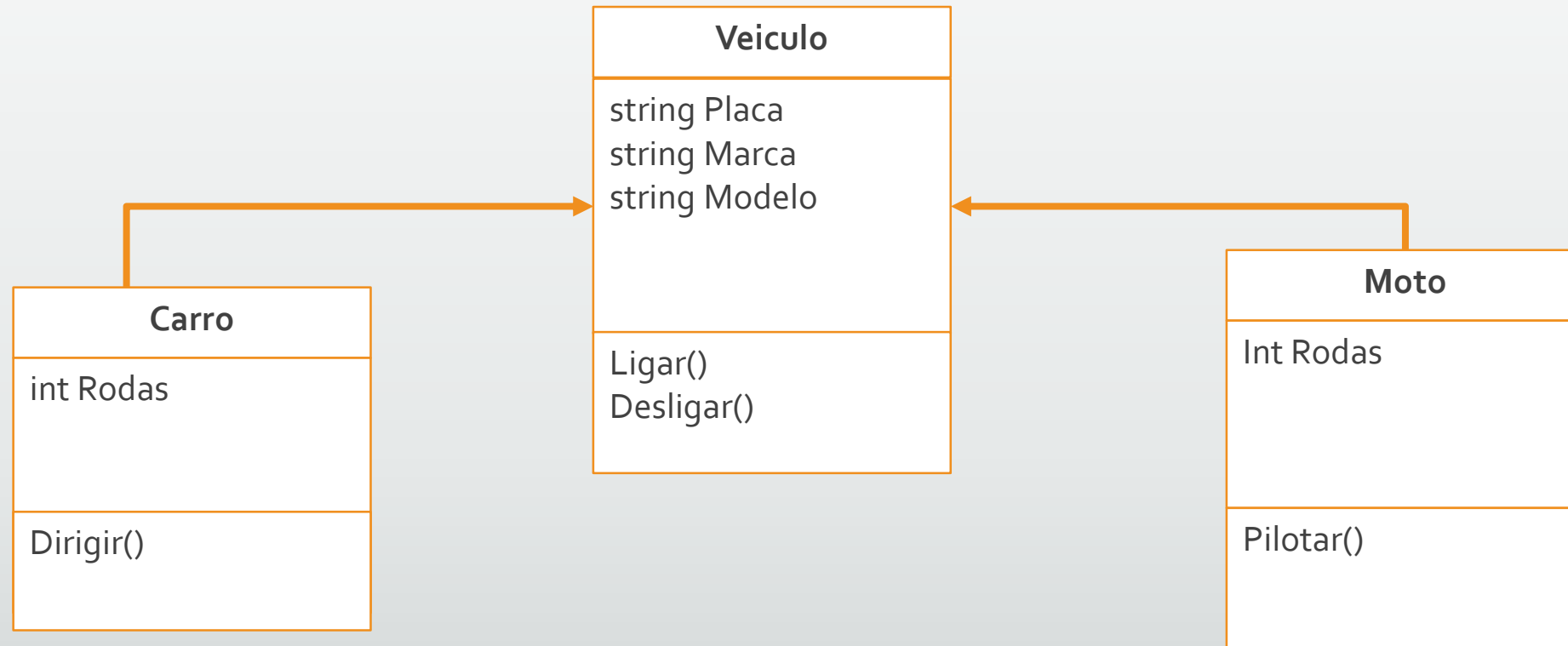
```
    Console.WriteLine($"Hoje é: {hoje}, o número do dia é: {(byte)hoje}");
```

```
}
```

Revisão



- Crie um novo projeto e programe as seguintes classes:



Classe Veículo

```
namespace Exercicio1
{
    internal class Veiculo
    {
        private string? placa;
        private string? modelo;
        private string? marca;

        public string? Placa { get => placa; set => placa = value; }
        public string? Modelo { get => modelo; set => modelo = value; }
        public string? Marca { get => marca; set => marca = value; }

        public void Ligar()
        {
            Console.WriteLine("Ligar");
        }

        public void Desligar()
        {
            Console.WriteLine("Desligar");
        }
    }
}
```

Classe "Carro"

```
namespace Exercicio1
{
    internal class Carro : Veiculo
    {
        private int qt_rodas;

        public int Qt_rodas { get => qt_rodas; set => qt_rodas = value; }

        public void Dirigir()
        {
            Console.WriteLine("Dirigir meu Carro");
        }
    }
}
```

Classe "Moto"

```
namespace Exercicio1
{
    internal class Moto : Veiculo
    {
        private int qt_rodas;

        public int Qt_rodas { get => qt_rodas; set => qt_rodas = value; }

        public void Pilotar()
        {
            Console.WriteLine("Dirigir minha Moto");
        }
    }
}
```

Vamos Utilizar?

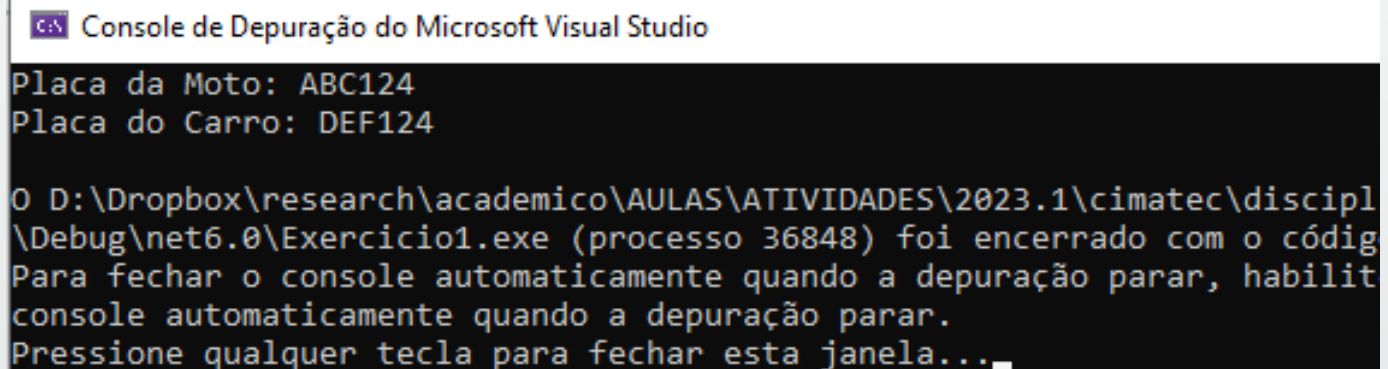
```
public class Program
{
    public static void Main(string[] args)
    {
        Moto m = new Moto();

        m.Placa = "ABC124";
        m.Marca = "Opala";
        m.Modelo = "XMTO";
        m.Qt_rodas = 2;

        Carro c = new Carro();

        c.Placa = "DEF124";
        c.Marca = "Monza";
        c.Modelo = "ABCD";
        c.Qt_rodas = 4;

        Console.WriteLine($"Placa da Moto: {m.Placa}");
        Console.WriteLine($"Placa do Carro: {c.Placa}");
    }
}
```



Console de Depuração do Microsoft Visual Studio

Placa da Moto: ABC124
Placa do Carro: DEF124

O D:\Dropbox\research\academico\AULAS\ATIVIDADES\2023.1\cimatec\discipl\Debug\net6.0\Exercicio1.exe (processo 36848) foi encerrado com o código 0.
Para fechar o console automaticamente quando a depuração parar, habilite o console automaticamente quando a depuração parar.
Pressione qualquer tecla para fechar esta janela..._

Ciclo de Vida dos Objetos

Ciclo de Vida de um Objeto

- Um objeto tem um determinado tempo de vida.
 - Quando utilizamos a palavra chave “**new**”, algumas coisas acontecem na memória.

```
Cliente c = new cliente (“nome”);
```



Carregamento

Alocação de
memória

Utilização

Descarte

Ciclo de Vida de um Objeto

1. Carregamento da classe

- Antes do objeto ser criado, a classe que modela este objeto precisa ser carregada
- Isso ocorre em tempo de execução

2. Alocação de memória

- Ao usar o **new**, o objeto é criado. A classe é inicializada e a memória é alocada para o novo objeto.
- Uma referência entre a classe e o novo objeto é então estabelecida.
- Na inicialização do objeto, o construtor é chamado apenas uma vez.

Ciclo de Vida de um Objeto

3. Tempo de vida

- O objeto então está disponível para utilização.
- O desenvolvedor utiliza os objetos criados para resolver um problema ou modelar uma funcionalidade.

4. Descarte

- Após sua contribuição no projeto, o objeto precisa ser descartado.
- Um dos pontos altos de VMs como Java ou C# é que você não precisa descartar estes objetos manualmente.
- O C# possui um coletor de lixo (***garbage collector***) que é responsável por descartar, automaticamente, todos os objetos não utilizados da memória.

Bons estudos!