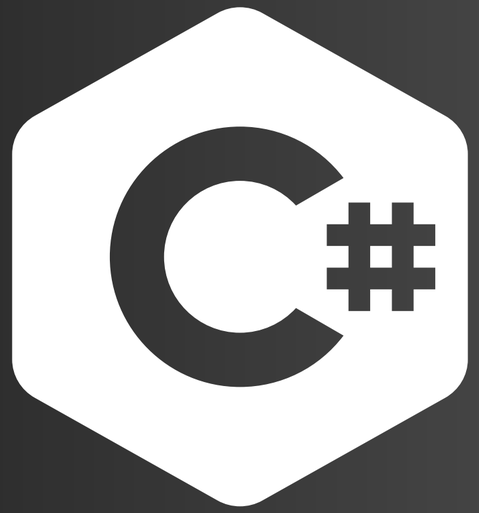


História

Prof. Wanderson Timóteo
www.wandersontimoteo.com.br

O que vamos aprender

- História do C# e do .NET Framework;
- Diferenças entre .NET Framework (Legado) e .NET;
- Compilador .NET e seu funcionamento;

Wanderson Timóteo

O que vamos aprender

- **História do C# e do .NET Framework;**
- ~~Diferenças entre .NET Framework (Legado) e .NET;~~
- ~~Compilador .NET e seu funcionamento;~~

História do C#

Na virada dos anos 1990 para os anos 2000, a Microsoft enfrentava um cenário desafiador no mercado de desenvolvimento de software, especialmente devido ao crescimento da popularidade do **Java**, desenvolvido pela Sun Microsystems. **Java** prometia um ambiente de desenvolvimento multiplataforma com o slogan **"Write Once, Run Anywhere" (*Escreva uma vez, execute em qualquer lugar*)**, o que atraiu muitos desenvolvedores que precisavam criar aplicações que funcionassem de forma consistente em diferentes sistemas operacionais. Nesta época a principal linguagem de programação da Microsoft era o Visual Basic.

Wanderson Timóteo

O Crescimento do Java e Seus Impactos

Java foi lançado em 1995 e rapidamente se tornou uma escolha popular para o desenvolvimento de aplicações empresariais e para a internet. Uma das grandes vantagens do Java era a **Java Virtual Machine (JVM)**, que permitia que o mesmo código Java fosse executado em várias plataformas, incluindo Windows, Linux, e macOS. Isso se tornou especialmente atraente para empresas que precisavam de soluções independentes de plataforma e que não queriam ficar presas a um único sistema operacional, como o Windows da Microsoft.

A Microsoft reconheceu que precisava de uma plataforma semelhante que fosse poderosa, eficiente e que atendesse às necessidades de desenvolvimento de software para Windows e, potencialmente, para outras plataformas, assim como o Java.

Microsoft vs Sun Microsystems

A competição entre a Microsoft e a Sun Microsystems sobre Java também levou a um conflito jurídico. Em 1997, a Microsoft licenciou a tecnologia Java da Sun para integrar ao Windows, mas fez modificações no Java que a Sun considerou incompatíveis com o padrão da linguagem. Isso levou a um processo em que a Sun acusou a Microsoft de violar o acordo de licenciamento. A disputa terminou em 2001, com a Microsoft concordando em pagar uma grande quantia em um acordo judicial e parar de usar a marca Java.

Esse embate legal evidenciou a necessidade da Microsoft de criar sua própria plataforma de desenvolvimento. Ao invés de depender de uma tecnologia controlada por outra empresa, a Microsoft optou por criar uma nova plataforma que fosse integrada ao Windows e que oferecesse uma experiência de desenvolvimento moderna, visando atrair os desenvolvedores que estavam migrando para Java.

Wanderson Timóteo

O Projeto "Cool" e o Desenvolvimento do C#

Em 1999, Anders Hejlsberg, um renomado engenheiro de software que anteriormente havia trabalhado no desenvolvimento da linguagem Delphi na Borland, foi contratado pela Microsoft. Ele foi designado para liderar o desenvolvimento de uma nova linguagem de programação que, inicialmente, recebeu o nome de **"Cool" (*C-like Object-Oriented Language*)**. Esse projeto visava criar uma linguagem moderna, orientada a objetos, que fosse mais robusta e eficiente que o **Visual Basic** e outras linguagens que a Microsoft possuía à época.

Wanderson Timóteo

O "Cool" evoluiu e, posteriormente, foi renomeado para **C# (*pronunciado como "C-sharp"*)**, inspirando-se na notação musical onde o símbolo de sustenido (#) representa um tom acima. O nome também sugere uma evolução do C e do C++ em direção a uma linguagem mais moderna.

Desenvolvimento Paralelo do .NET Framework

Ao mesmo tempo em que **C#** estava sendo desenvolvido, a Microsoft também criou o **.NET Framework**, uma plataforma de desenvolvimento que serviria como a base para a execução de programas escritos em **C#** e outras linguagens compatíveis. O objetivo do .NET Framework era proporcionar uma plataforma de desenvolvimento robusta, capaz de gerenciar memória de forma eficiente, oferecer uma vasta biblioteca de classes reutilizáveis, e permitir a criação de aplicações de alta performance.

Wanderson Timóteo

O .NET Framework foi projetado para ser executado em uma **Common Language Runtime (CLR)**, que é um ambiente de execução que gerencia a execução de código .NET, incluindo o gerenciamento automático de memória com garbage collection, segurança, e outras tarefas. O CLR é semelhante ao que a JVM faz para Java, mas foi otimizado para integrar-se profundamente ao sistema operacional Windows.

Além do CLR, o .NET Framework trouxe a **Base Class Library (BCL)**, uma coleção de classes e APIs que facilitam o desenvolvimento de aplicações para desktop, web, e outros tipos de serviços. Essas classes ofereciam funcionalidades essenciais, como manipulação de strings, acesso a arquivos, interação com redes e interfaces gráficas.

Lançamento e Competição com Java

Em 2000, o C# e o .NET Framework foram apresentados ao público durante a Professional Developers Conference (PDC) em Orlando, na Flórida, e lançados oficialmente em 2002. A Microsoft visava atrair desenvolvedores que já usavam o Windows, oferecendo integração profunda com o sistema operacional e uma experiência de desenvolvimento poderosa através do Visual Studio, um ambiente de desenvolvimento integrado (IDE) que se tornou a principal ferramenta para criar aplicações C# e .NET.

Resultados e Adaptação às Mudanças do Mercado

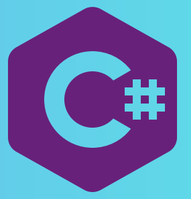
A estratégia da Microsoft funcionou, e o C# rapidamente se consolidou como uma das principais linguagens de programação do mercado, com o .NET Framework sendo amplamente utilizado para criar aplicações empresariais. A popularidade do C# também se expandiu para áreas como o desenvolvimento de jogos, especialmente com a adoção do C# pela Unity, um dos motores de jogo mais populares do mercado.

Wanderson Timóteo

Com o tempo, a Microsoft se adaptou às novas demandas do mercado. Em 2016, lançou o .NET Core, uma versão de código aberto e multiplataforma do .NET, que posteriormente foi unificada sob o nome .NET 5 e suas versões seguintes. Essa unificação permitiu que desenvolvedores C# pudessem criar aplicações que rodassem em diferentes sistemas operacionais, como Linux e macOS, além do Windows, aproximando a Microsoft da promessa inicial do Java de "escreva uma vez, execute em qualquer lugar".

Impacto no Mercado de Software

A criação do C# e do .NET Framework foi uma resposta estratégica da Microsoft ao crescimento do Java e da necessidade de uma plataforma moderna para o desenvolvimento de software corporativo e web. Essa decisão permitiu que a Microsoft competisse de igual para igual no mercado de desenvolvimento de software e oferecesse uma alternativa robusta para empresas e desenvolvedores.



Wanderson Timóteo

A combinação de uma linguagem poderosa como o C# e uma plataforma versátil como o .NET Framework permitiu que a Microsoft se mantivesse relevante e influente no mundo do desenvolvimento de software. A empresa conseguiu fidelizar desenvolvedores que preferiam o ecossistema Microsoft e, ao mesmo tempo, abrir novas possibilidades com a adoção de código aberto e suporte multiplataforma.

Em suma, a criação do C# e do .NET Framework foi um movimento fundamental para a Microsoft, permitindo que a empresa superasse o desafio imposto pelo crescimento do Java, adaptando-se às necessidades do mercado e ajudando a moldar o desenvolvimento de software nas décadas seguintes.

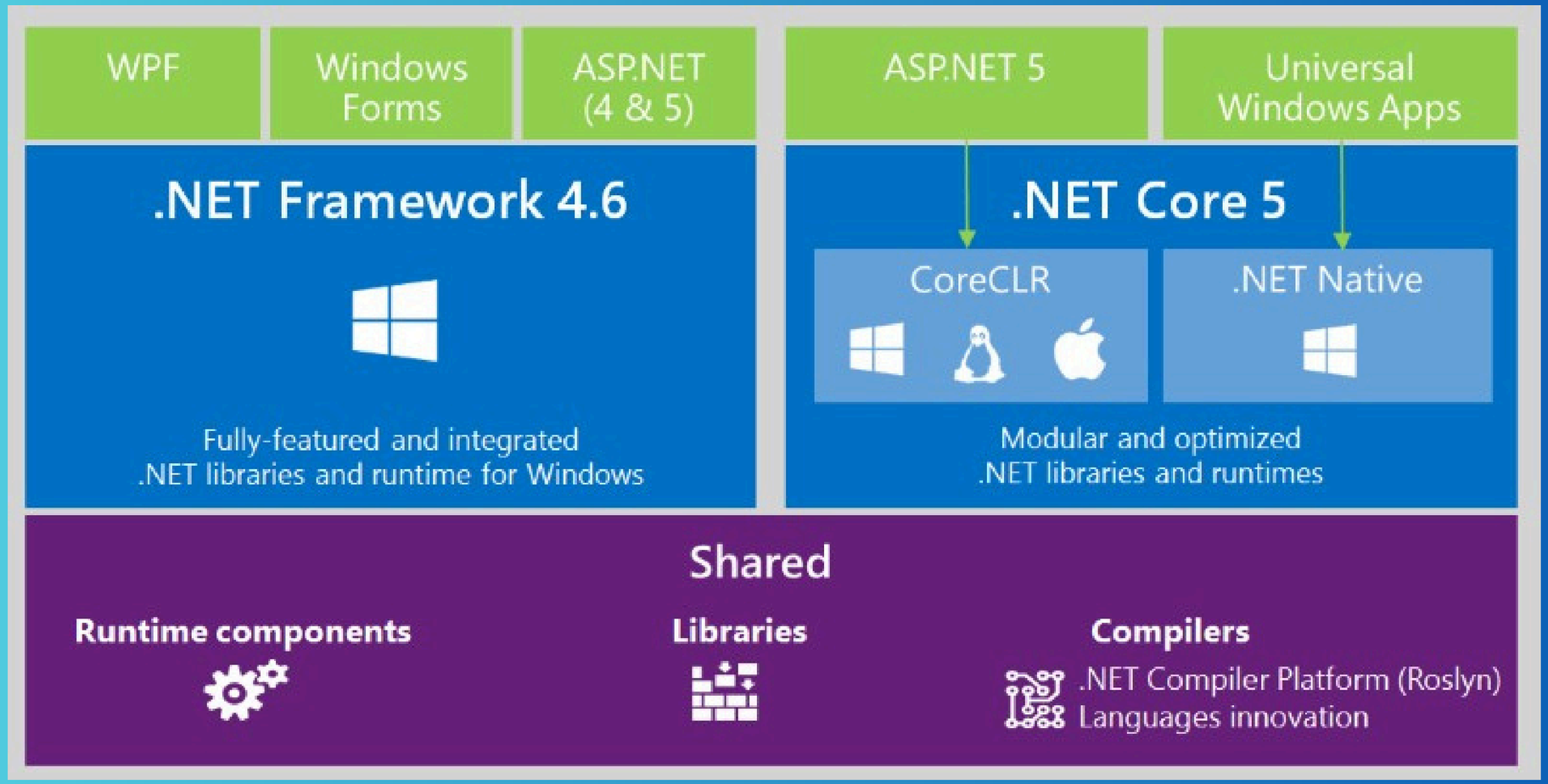
Para saber mais sobre a timeline do .NET e C# acesse: https://time.graphics/pt/line/291016

Wanderson Timóteo

O que vamos aprender

- ~~História do C# e do .NET Framework;~~
- **Diferenças entre .NET Framework (Legado) e .NET;**
- ~~Compilador .NET e seu funcionamento;~~

Diferenças entre .NET Framework (Legado) e .NET



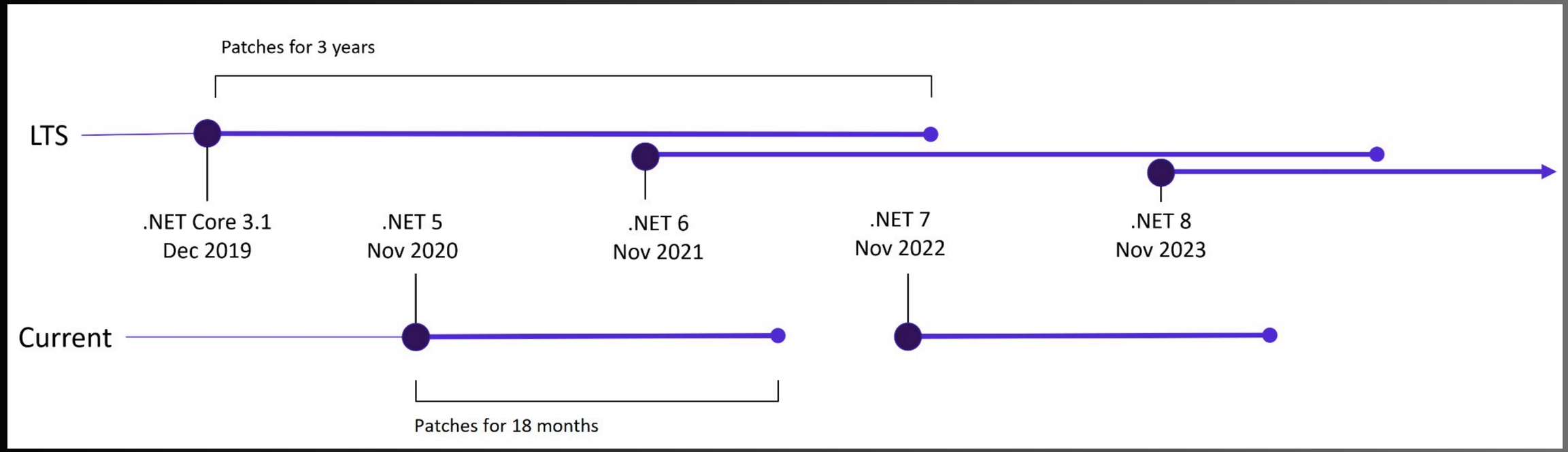
Versões .NET Framework

Resumo histórico de versões do .NET Framework[3][4]

Versão	Versão CLR	Data lançamento	Visual Studio	Incluso no		Substitui
				Windows	Windows Server	
1.0	1.0	2002-02-13	Visual Studio .NET[5]	XP[a]	—	—
1.1	1.1	2003-04-24	Visual Studio .NET 2003[5]	—	2003	1.0[6]
2.0	2.0	2005-11-07	Visual Studio 2005[7]	—	2003, 2003 R2,[8] 2008 SP2, 2008 R2 SP1	—
3.0	2.0	2006-11-06	Expression Blend[9][b]	Vista	2008 SP2, 2008 R2 SP1	2.0
3.5	2.0	2007-11-19	Visual Studio 2008[10]	7, 8[c], 8.1[c], 10[c]	2008 R2 SP1	2.0, 3.0
4.0	4	2010-04-12	Visual Studio 2010[11]	—	—	—
4.5	4	2012-08-15	Visual Studio 2012[12]	8	2012	4.0
4.5.1	4	2013-10-17	Visual Studio 2013[13]	8.1	2012 R2	4.0, 4.5
4.5.2	4	2014-05-05	—	—	—	4.0–4.5.1
4.6	4	2015-07-20	Visual Studio 2015[14]	10	—	4.0–4.5.2
4.6.1	4	2015-11-30[15]	Visual Studio 2015 Update 1	10 v1511	—	4.0–4.6
4.6.2	4	2016-08-02[16]		10 v1607	—	4.0–4.6.1
4.7	4	2017-04-05[17]	Visual Studio 2017	10 v1703	N/A	4.0-4.6.2
4.8	4	2019-04-18[18]	Visual Studio 2019	10 v1903	2019, 2016, 2012, 2012 R2, 2008 R2 SP1	4.0-4.7

Wanderson Timóteo

Versões .NET (dotNET)



Com a plataforma .NET podemos programar usando as linguagens C#, F#, Visual Basic .NET (VB.NET), C++/CLI e PowerShell.

Wanderson Timóteo

O que vamos aprender

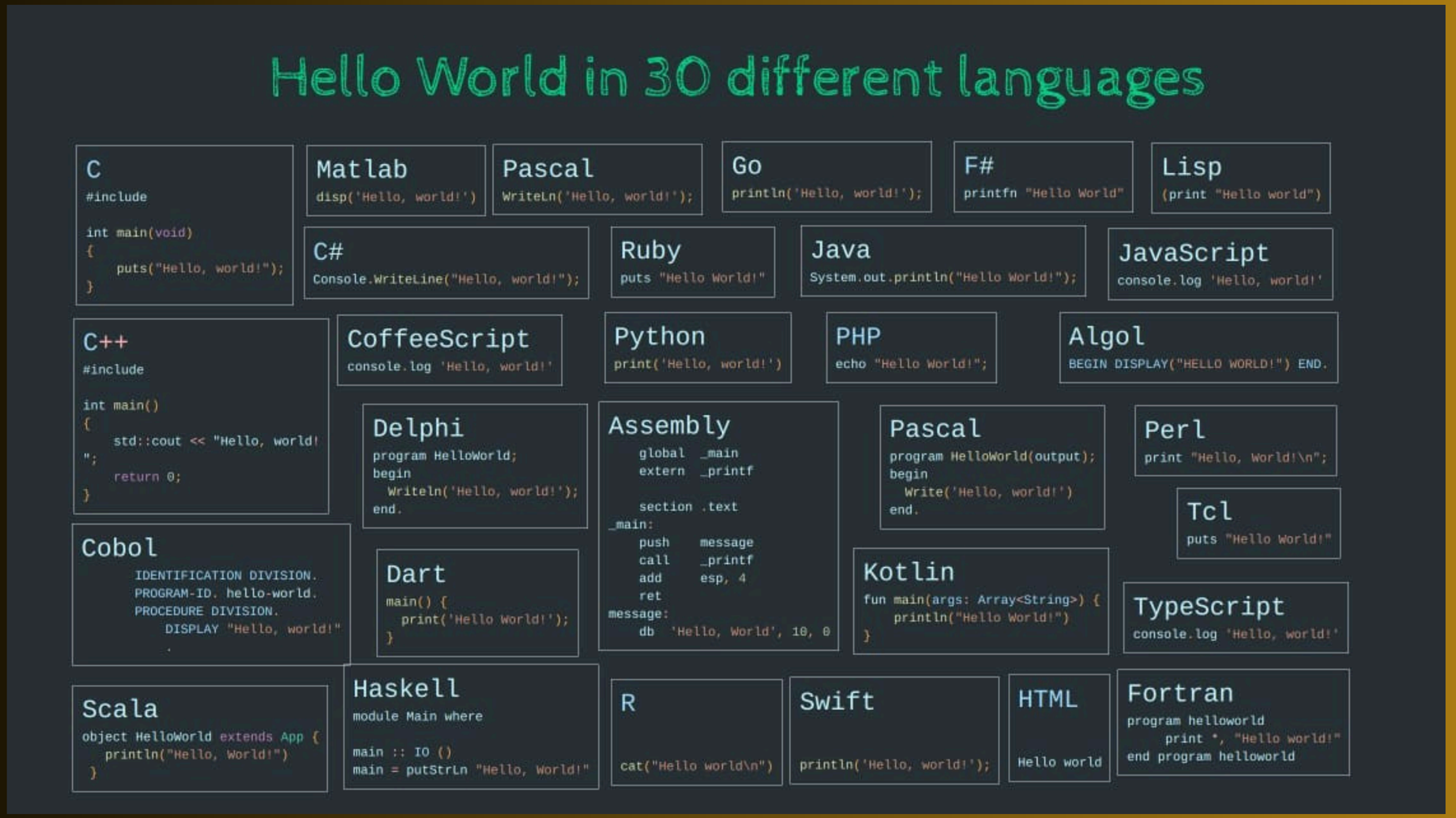
- ~~História do C# e do .NET Framework;~~
- ~~Diferenças entre .NET Framework (Legado) e .NET;~~
- **Compilador .NET e seu funcionamento;**

O que é um compilador?

Linguagem de alto nível: A linguagem que entendemos e escrevemos nosso código fonte.

Linguagem de baixo nível: A linguagem que o computador ou a máquina entende. Possui pouca abstração, sendo difícil de entender.

Linguagens de Programação



Wanderson Timóteo

Assembly e Machine Code

Assembly vs. machine code

Machine code bytes	Assembly language statements
	foo:
B8 22 11 00 FF	movl $0xFF001122, %eax
01 CA	addl %ecx, %edx
31 F6	xorl %esi, %esi
53	pushl %ebx
8B 5C 24 04	movl 4(%esp), %ebx
8D 34 48	leal (%eax,%ecx,2), %esi
39 C3	cmpl %eax, %ebx
72 EB	jnae foo
C3	retl

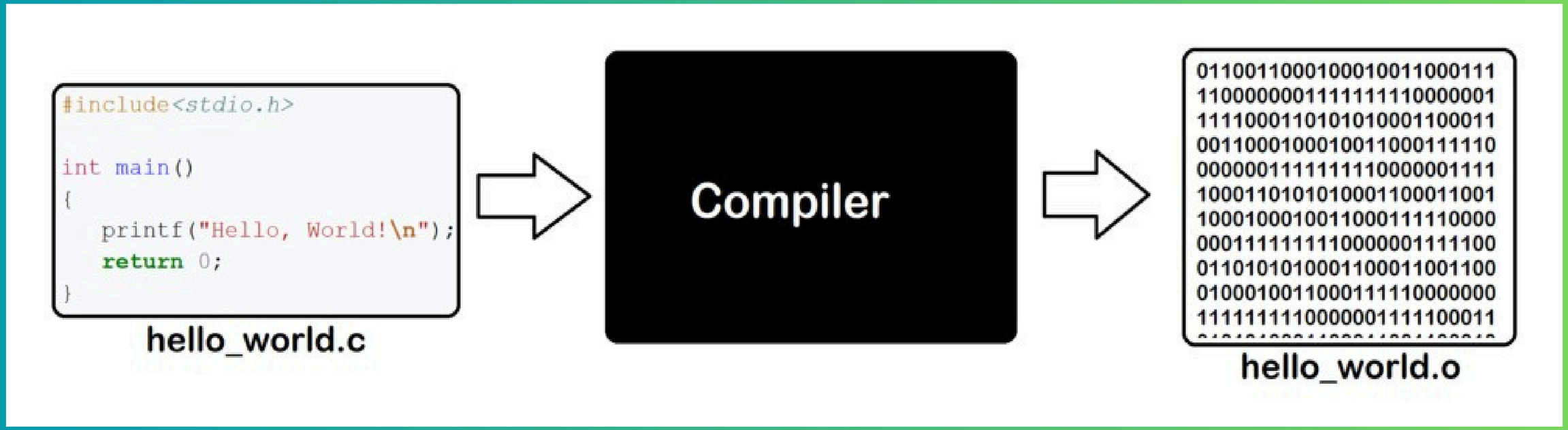
Instruction stream

```
B8 22 11 00 FF 01 CA 31 F6 53 8B 5C 24
04 8D 34 48 39 C3 72 EB C3
```

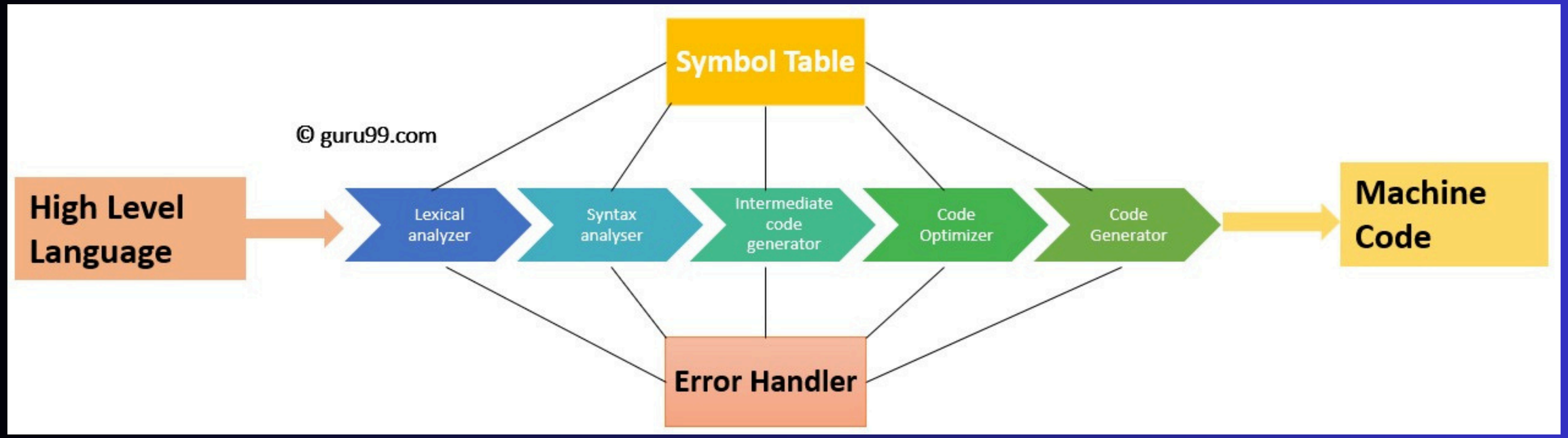
Wanderson Timóteo

O que vamos aprender

Compilador: É um programa que realiza a conversão de linguagem de alto nível para baixo nível.



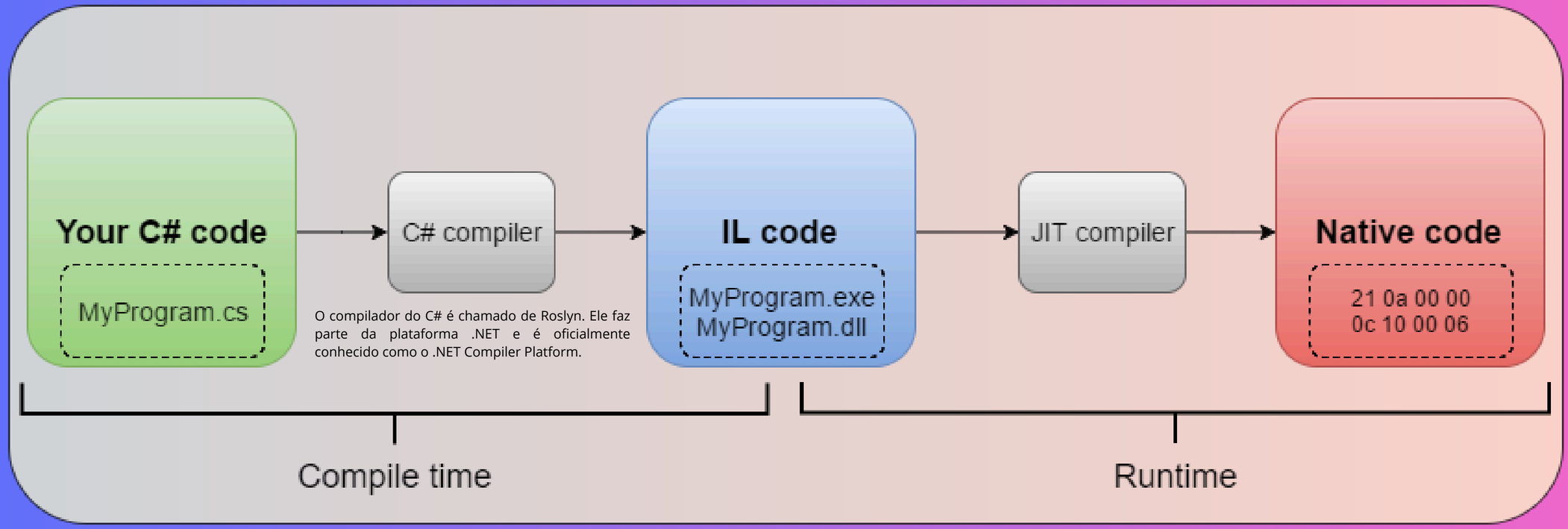
Fases de um compilador



https://www.guru99.com/compiler-design-phases-of-compiler.html

https://freecontent.manning.com/how-is-c-compiled

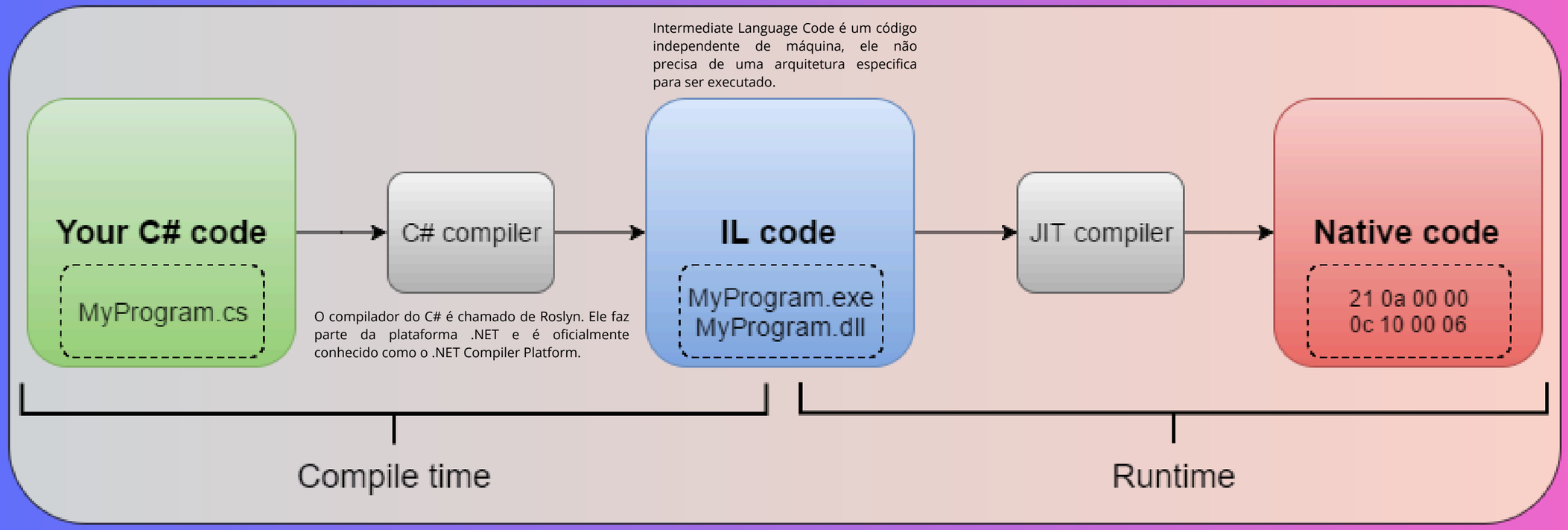
Compilador do .NET



Compilação Inicial para IL: Quando um programa em C# é compilado usando o compilador Roslyn, ele não é diretamente transformado em código nativo, ou seja, código que a CPU pode executar. Em vez disso, o código C# é transformado em **Intermediate Language (IL)**, que é uma forma de código intermediário que pode ser entendida pela **Common Language Runtime (CLR)**, o ambiente de execução do .NET.

Wanderson Timóteo

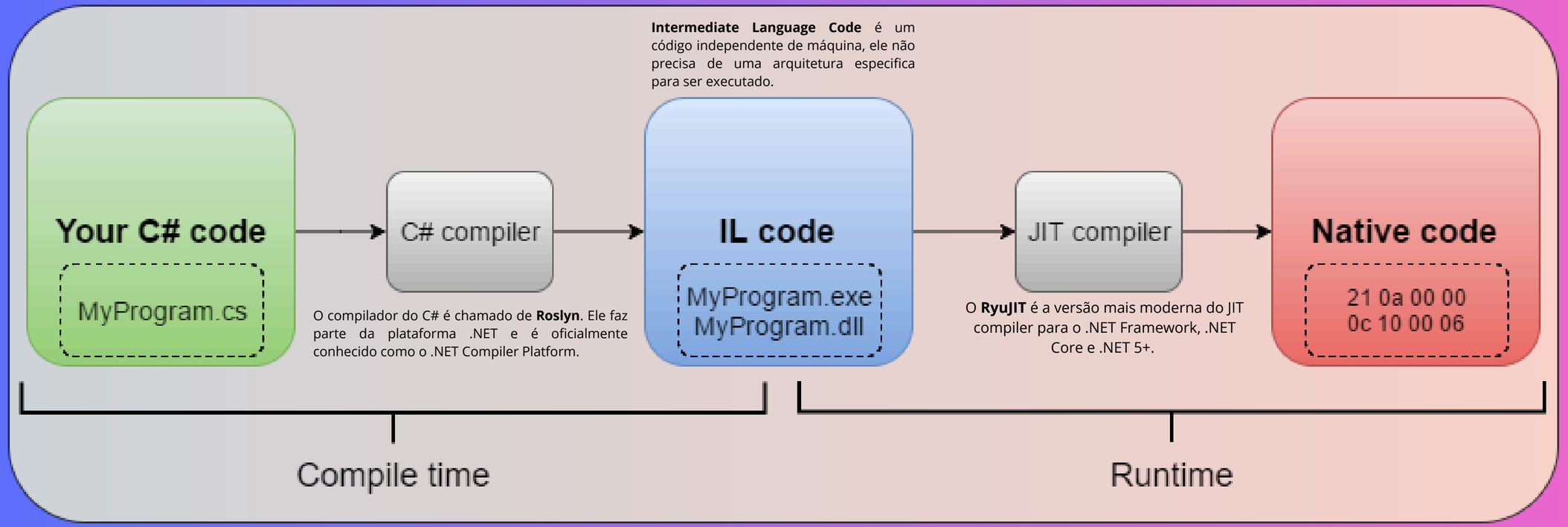
Compilador do .NET



Esse código IL é armazenado em um arquivo .exe ou .dll, que é a versão compilada do programa, mas ainda não é código específico para a CPU do sistema. O **compilador Just-In-Time (JIT)** é um componente do ambiente de tempo de execução que melhora o desempenho de aplicativos C# compilando os arquivos *.dll* e *.exe* para o código de máquina nativo no tempo de execução, usando a arquitetura do seu Sistema Operacional.

Wanderson Timóteo

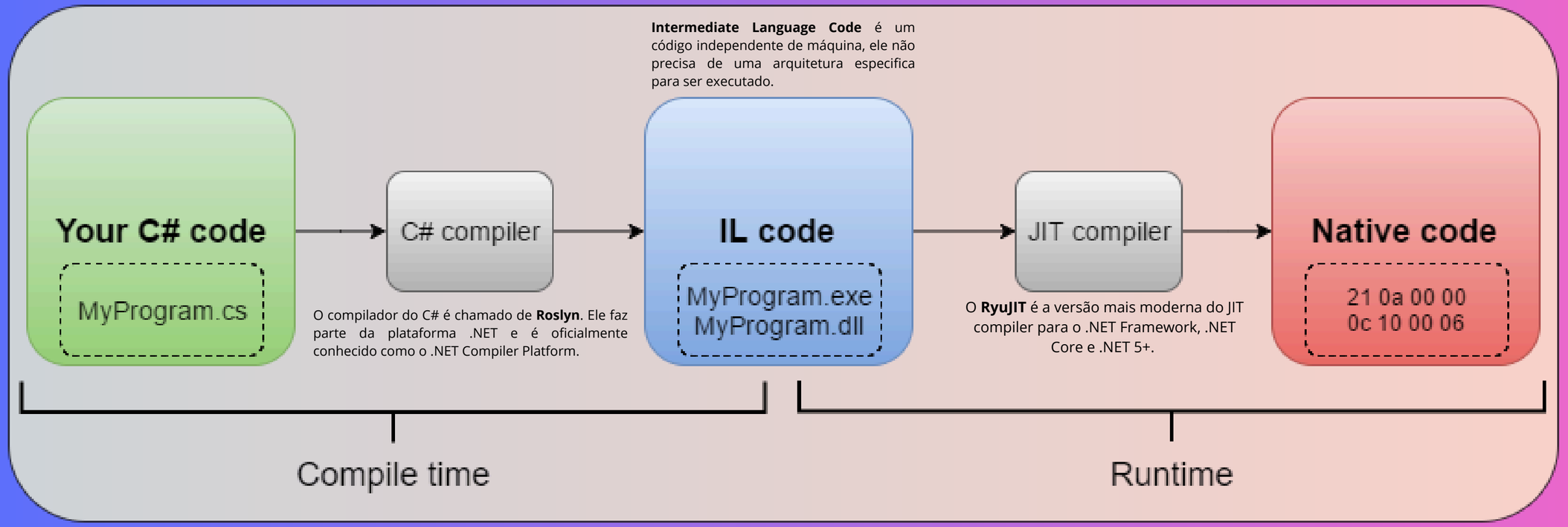
Compilador do .NET



Execução e Conversão para Código Nativo: Quando o programa é executado, o JIT compiler entra em ação. Ele é parte da **CLR (Common Language Runtime)** e é responsável por converter o código IL em código nativo, também chamado de código de máquina, que pode ser diretamente executado pela CPU do sistema. Esse processo acontece just-in-time, ou seja, no momento em que o método ou função é chamado pela primeira vez. Assim, o código é convertido conforme necessário, e não tudo de uma vez antes da execução.

Wanderson Timóteo

Compilador do .NET

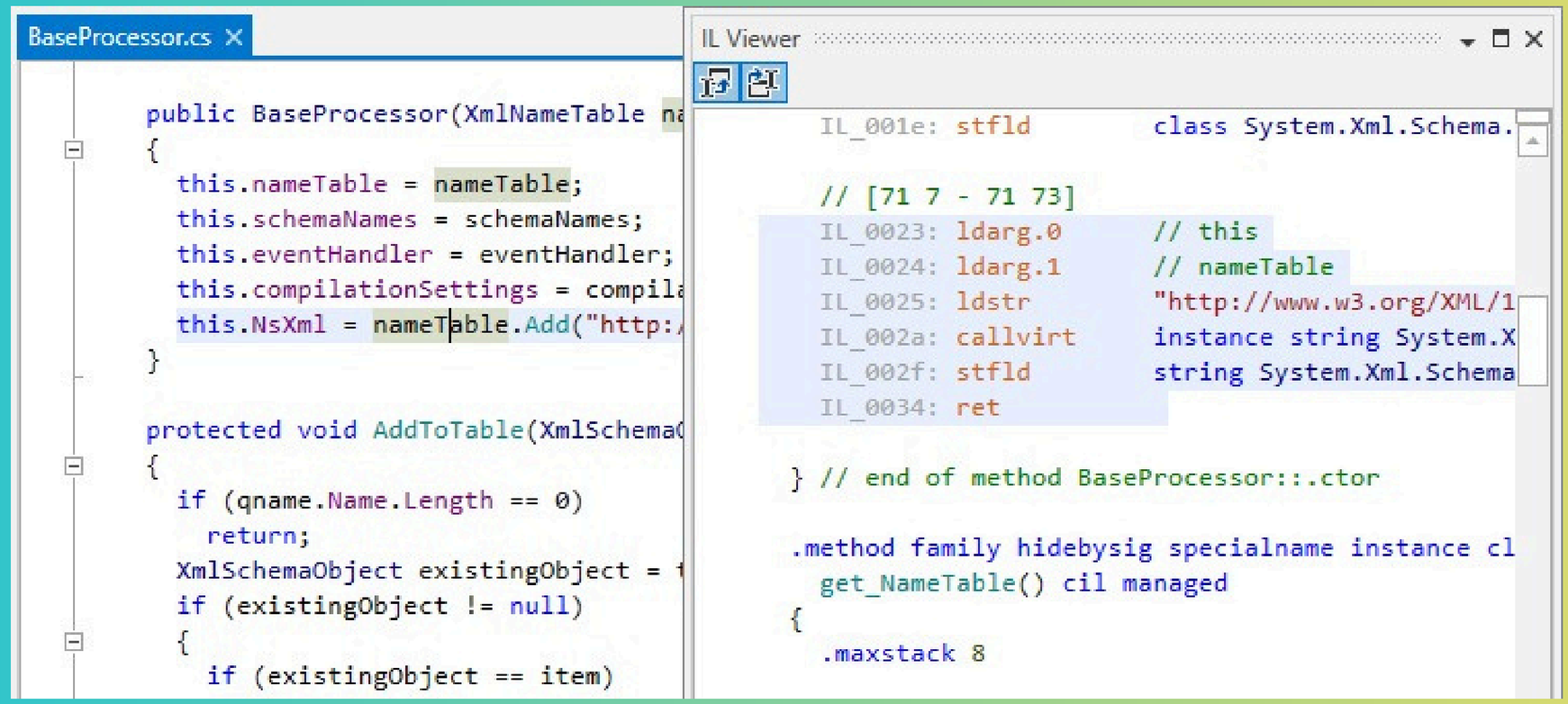


Código Nativo Cacheado: Após a primeira execução de um método, o JIT compiler armazena o código nativo resultante em cache na memória. Isso significa que, nas próximas vezes que o mesmo método for chamado, a CLR pode usar o código nativo já compilado, sem precisar passar pelo processo de compilação novamente, melhorando a performance.

Wanderson Timóteo

Compilador do .NET

Código-Fonte em C# | Intermediate Language (IL)



Quando um código em C# é compilado usando o compilador Roslyn, ele é transformado em Intermediate Language (IL)

Wanderson Timóteo

Compilador e Transpilador

Compilador: É um programa que realiza a conversão de linguagem de alto nível para baixo nível. Exemplo: C#, Java.

Transpilador: É a conversão de uma linguagem ou implementação para outra. A sua saída permanece em linguagem de alto nível. Exemplo: Typescript para Javascript

Nem toda linguagem é compilada!

- **Linguagem compilada**: São linguagens que o código-fonte é traduzido para o código de máquina. Exemplo: C# e Java.

- **Linguagem interpretada:** São linguagens que fazem a leitura e interpretação diretamente do código fonte. Exemplo: Javascript e PHP.

Compilador do .NET

Vídeo: Compilador (O Programa Essencial de Todos os Programadores) - Código Fonte TV. Acesse: https://www.youtube.com/watch?v=afUiVvDUIRA

Wanderson Timóteo

Referências:

- https://www.scalablepath.com/dot-net/microsoft-dot-net-framework
- https://www.quora.com/Why-was-C-developed-when-an-object-oriented-programming-language-like-C++-already-existed
- https://docs.microsoft.com/pt-br/dotnet/core/introduction
- https://time.graphics/pt/line/291016
- https://dotnet.microsoft.com/en-us/platform/support/policy/dotnet-core
- https://pt.wikipedia.org/wiki/.NET_Framework
- https://dotnet.microsoft.com/en-us/platform/support/policy/dotnet-core
- https://docs.microsoft.com/pt-br/dotnet/core/get-started
- https://www.guru99.com/compiler-design-phases-of-compiler.html
- https://freecontent.manning.com/how-is-c-compiled
- https://jpdeffo.medium.com/how-dotnet-core-compilation-work-for-absolute-beginners-fdba62b3167c
- https://www.geeksforgeeks.org/what-is-just-in-time-jit-compiler-in-dot-net/ https://www.quora.com/Is-assembly-just-another-name-for-machine-code https://hpc-wiki.info/hpc/Compiler https://freecontent.manning.com/how-is-c-compiled/
- https://codeasy.net/lesson/c_sharp_compilation_process