

O Retorno do .Net



Por Fábio Santos



Quem sou eu?

Olá, meu nome é Fábio.

Iniciei minha carreira na TI em 2011 como **suporte técnico**, onde atuei por aproximadamente 2 anos.

No desenvolvimento, comecei em um estágio codando em **PHP**.

Atuei como **DBA** por 2 anos, realizando o desenvolvimento de scripts em **PL/SQL** e **T-SQL**.

Migrei para o **.NET** há aproximadamente **7 anos** e há 2 anos atuo na Framework como **Arquiteto de Software .NET**.



O que é .Net

.Net é um framework baseado na linguagem de programação C#. C# por sua vez é uma linguagem de programação fortemente tipada.

Isso quer dizer que, diferente do PHP, que é uma linguagem fracamente tipada, no .Net trabalha-se com variáveis informando os tipos delas e o compilador te barrando de inserir valores de um tipo em uma variável de outro tipo.

Para trabalharmos com tipos diferentes do tipo atribuído para sua variável, devemos sempre realizar conversões para o tipo da variável.

Falando sobre tipos, temos os primitivos, que são os tipos base do .Net, eles são identificados por palavras chaves que na verdade são um apelido para os tipos predefinidos nas bibliotecas base do .Net.



Tipos de Primitivos:

Tipo	Descrição	Valor de Padrão
Byte	Inteiro de 8 bits sem sinal (0 a 255).	0
Short	Inteiro de 16 bits com sinal (-32768 a 32767).	0
Integer	Inteiro de 32 bits com sinal (-2147483648 a 2147483647).	0
Long	Inteiro de 64 bits com sinal (-9223372036854775808 a 9223372036854775807). 0	0
Single	Ponto flutuante 32 bits (System.Single).	0
Double	Ponto Flutuante 64 bits. (System.Double).	0
Decimal	Ponto Flutuante decimal de 128 bits (1.0×10^{-28} a 7.9×10^{-28}), 28 dígitos decimais de precisão.	0
Boolean	Pode ter os valores True e False. (System.Boolean).	False
Char	Representa um único caractere unicode de 16 bits.	
Date	Representa uma data ou hora. (System.DateTime).	# 01/01/0001 12:00:00AM #
String	Representa uma sequência de caracteres unicode.	nul.



Tipos por referência:

Outro tipo que temos no C# são os por referência, esses tipos possuem um ponteiro para outra alocação de memória que gerencia os dados.

São esses tipos:

- ⌚ String
- ⌚ Todos os vetores(arrays) , mesmo se seus elementos são tipos de valor
- ☔ Tipos Classes , como um Form
- ⌚ Tipos herdados de object
- ⌚ Interface
- ⌚ Delegate

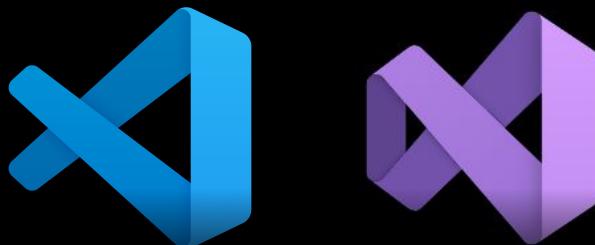
Memoria				
p 1	p 2	p4		
	p 3			
r1	r1.nome	r1.sexo	r1.dataNasc	

IDE: Interfaces de desenvolvimento

Interfaces de desenvolvimentos são ferramentas criadas para facilitar o trabalho do desenvolvedor.

Essas ferramentas oferecem uma gama de facilitadores, compilação da aplicação, integração com repositório Git, autocomplete e mostra os erros de código sem necessidade de compilação.

Para desenvolvimento .Net os mais comuns utilizados são o VS Code e o Visual Studio, que se encontra na versão 2022 e tem a versão Comunit que é gratuita para estudantes da linguagem c#.



POO: Introdução à programação orientada à objetos

- 💡 É um conceito amplamente difundido no meio da programação e consiste em trabalhar com utilização de classes e objetos.
- 🌐 Uma classe nada mais é que a definição do mundo real.
- ☁️ Classes são compostas por propriedades, métodos de ação e construtor, qual o .Net cria como padrão, existindo a possibilidade de sobrecarga deste.
- 💡 Um objeto é a instanciação dessa classe na memória durante a execução da sua aplicação.



```
1  namespace Poc
2  {
3      1 referência
4      public class Carro
5      {
6          2 referências
7          public int Rodas { get; set; }
8          2 referências
9          public int CapacidadePortaMalas { get; set; }
10         2 referências
11         public int Portas { get; set; }
12         2 referências
13         public string PotenciaMotor { get; set; }
14         2 referências
15         public string TransparenciaVidros { get; set; }
16     }
17 }
```

```
1  using System;
2
3  namespace Poc
4  {
5      0 referências
6      internal class Program
7      {
8          0 referências
9          static void Main(string[] args)
10         {
11             var carro = new Carro();
12             carro.Rodas = 4;
13             carro.CapacidadePortaMalas = 310;
14             carro.Portas = 5;
15             carro.PotenciaMotor = "1.6";
16             carro.TransparenciaVidros = "100%";
17
18             var texto = $"Características do carro:";
19             texto += $"Quantidade de rodas: {carro.Rodas}{Environment.NewLine}";
20             texto += $"Capacidade do porta malas: {carro.CapacidadePortaMalas}{Environment.NewLine}";
21             texto += $"Quantidade de portas: {carro.Portas}{Environment.NewLine}";
22             texto += $"Potência do motor: {carro.PotenciaMotor}{Environment.NewLine}";
23             texto += $"Transparência dos vidros: {carro.TransparenciaVidros}{Environment.NewLine}";
24             Console.WriteLine(texto);
25         }
26     }
27 }
```

Console de Depuração do Microsoft Visual Studio

Características do carro:

Quantidade de rodas: 4

Capacidade do porta malas: 310

Quantidade de portas: 5

Potência do motor: 1.6

Transparência dos vidros: 100%



```
1  namespace Poc
2  {
3      2 referências
4      public class Veiculo
5      {
6          2 referências
7          public int Rodas { get; set; }
8          2 referências
9          public int CapacidadePortaMalas { get; set; }
10         2 referências
11         public int Portas { get; set; }
12         2 referências
13         public string PotenciaMotor { get; set; }
14         2 referências
15         public string TransparenciaVidros { get; set; }
16     }
17 }
```

```
1  namespace Poc
2  {
3      1 referência
4      public class Carro : Veiculo
5      {
6      }
6 }
```

```
1  namespace Poc
2  {
3      0 referências
4      public class Caminhao : Veiculo
5      {
6      }
6 }
```

```
5  internal class Program
6  {
7      0 referências
8      static void Main(string[] args)
9      {
10         var carro = new Carro();
11         carro.Rodas = 4;
12         carro.CapacidadePortaMalas = 310;
13         carro.Portas = 5;
14         carro.PotenciaMotor = "1.6";
15         carro.TransparenciaVidros = "100%";

16         var texto = $"Características do carro:{Environment.NewLine}";
17         texto += $"{Quantidade de rodas: {carro.Rodas}{Environment.NewLine}";
18         texto += $"{Capacidade do porta malas: {carro.CapacidadePortaMalas}{Environment.NewLine}";
19         texto += $"{Quantidade de portas: {carro.Portas}{Environment.NewLine}";
20         texto += $"{Potência do motor: {carro.PotenciaMotor}{Environment.NewLine}";
21         texto += $"{Transparéncia dos vidros: {carro.TransparenciaVidros}{Environment.NewLine}";
22         Console.WriteLine(texto);

23         var caminhao = new Caminhao();
24         caminhao.Rodas = 16;
25         caminhao.CapacidadePortaMalas = 1000;
26         caminhao.Portas = 2;
27         caminhao.PotenciaMotor = "5.0";
28         caminhao.TransparenciaVidros = "50%";

29         texto = $"Características do caminhão:{Environment.NewLine}";
30         texto += $"{Quantidade de rodas: {caminhao.Rodas}{Environment.NewLine}";
31         texto += $"{Capacidade do porta malas: {caminhao.CapacidadePortaMalas}{Environment.NewLine}";
32         texto += $"{Quantidade de portas: {caminhao.Portas}{Environment.NewLine}";
33         texto += $"{Potência do motor: {caminhao.PotenciaMotor}{Environment.NewLine}";
34         texto += $"{Transparéncia dos vidros: {caminhao.TransparenciaVidros}{Environment.NewLine}";
35         Console.WriteLine(texto);
36     }
37 }
```

Console de Depuração do Microsoft Visual Studio

Características do carro:

Quantidade de rodas: 4
Capacidade do porta malas: 310
Quantidade de portas: 5
Potência do motor: 1.6
Transparéncia dos vidros: 100%

Características do caminhão:

Quantidade de rodas: 16
Capacidade do porta malas: 1000
Quantidade de portas: 2
Potência do motor: 5.0
Transparéncia dos vidros: 50%

framework

framework
PADAWANS
EPISÓDIO VI

Boas práticas e padrões de desenvolvimento no .Net

No .Net são utilizados vários padrões, alguns deles são:

-  **Pascal Case** - É a prática de escrever palavras compostas ou frases de modo que cada palavra ou abreviatura comece com uma letra maiúscula.
-  **Camel Case** - É a prática de escrever palavras compostas ou frases de modo que cada palavra ou abreviatura no meio da frase comece com uma letra maiúscula.
-  **Encapsulamento** - É um princípio de design de código, geralmente ligado a programação orientada a objetos, que nos leva a esconder as funcionalidades e funcionamentos do nosso código dentro de pequenas unidades (normalmente métodos e funções). Isso possibilita que modificações no sistema possam ser feitas de maneira mais cirúrgicas, sem que uma funcionalidade esteja espalhada por diversas partes do sistema.



```
5  public class Veiculo
6  {
7      2 referências
8      public int Rodas { get; set; }
9      2 referências
10     public int CapacidadePortaMalas { get; set; }
11     2 referências
12     public int Portas { get; set; }
13     2 referências
14     public string PotenciaMotor { get; set; }
15     2 referências
16     public string TransparenciaVidros { get; set; }
17
18     8 referências
19     protected string _texto { get; set; }
20
21     2 referências
22     public Veiculo(int rodas, int capacidadePortaMalas, int portas, string potenciaMotor, string transparenciaVidros)
23     {
24         Rodas = rodas;
25         CapacidadePortaMalas = capacidadePortaMalas;
26         Portas = portas;
27         PotenciaMotor = potenciaMotor;
28         TransparenciaVidros = transparenciaVidros;
29     }
30
31     6 referências
32     public virtual string ImprimirCaracteristicas()
33     {
34         _texto += $"Quantidade de rodas: {Rodas}{Environment.NewLine}";
35         _texto += $"Capacidade do porta malas: {CapacidadePortaMalas}{Environment.NewLine}";
36         _texto += $"Quantidade de portas: {Portas}{Environment.NewLine}";
37         _texto += $"Potência do motor: {PotenciaMotor}{Environment.NewLine}";
38         _texto += $"Transparéncia dos vidros: {TransparenciaVidros}{Environment.NewLine}";
39         return _texto;
40     }
41 }
```

```
1  using System;
2
3  namespace Poc
4  {
5      2 referências
6      public class Caminhao : Veiculo
7      {
8          1 referência
9          public Caminhao(int rodas, int capacidadePortaMalas, int portas, string potenciaMotor, string transparenciaVidros)
10         : base(rodas, capacidadePortaMalas, portas, potenciaMotor, transparenciaVidros)
11     {
12     }
13
14     4 referências
15     public override string ImprimirCaracteristicas()
16     {
17         _texto = $"Características do caminhão:{Environment.NewLine}";
18         return base.ImprimirCaracteristicas();
19     }
20 }
```

```
1  using System;
2
3  namespace Poc
4  {
5      2 referências
6      public class Carro : Veiculo
7      {
8          1 referência
9          public Carro(int rodas, int capacidadePortaMalas, int portas, string potenciaMotor, string transparenciaVidros)
10         : base(rodas, capacidadePortaMalas, portas, potenciaMotor, transparenciaVidros)
11     {
12         _texto = $"Características do carro:{Environment.NewLine}";
13     }
14 }
```

```
1  using System;
2
3  namespace Poc
4  {
5      0 referências
6      internal class Program
7      {
8          0 referências
9          static void Main(string[] args)
10         {
11             var carro = new Carro(4, 310, 5, "1.6", "100%");
12             Console.WriteLine(carro.ImprimirCaracteristicas());
13
14             var caminhao = new Caminhao(16, 1000, 2, "5.0", "50%");
15             Console.WriteLine(caminhao.ImprimirCaracteristicas());
16         }
17     }
18 }
```

Console de Depuração do Microsoft Visual Studio

Características do carro:

Quantidade de rodas: 4
Capacidade do porta malas: 310
Quantidade de portas: 5
Potência do motor: 1.6
Transparência dos vidros: 100%

Características do caminhão:

Quantidade de rodas: 16
Capacidade do porta malas: 1000
Quantidade de portas: 2
Potência do motor: 5.0
Transparência dos vidros: 50%

Missão dada é missão cumprida!

- 💡 Como a minha tecnologia ou atuação contribui para o posicionamento da Frame de **Missão dada é missão cumprida?**
- 🌐 Hoje o .Net é um dos frameworks mais utilizados pelo mercado e com os clientes da Frame isso não é diferente! Temos desafios diários que são solucionados utilizando o .Net e estarmos com conhecimento nivelado faz toda a diferença para conseguirmos cumprir as missões dadas.
- ☁️ Nas equipes que atuei desde que entrei na Frame, sempre que possível houve distribuição de demandas e responsabilidades onde todos tiveram espaço para colocar seus conhecimentos em prática e aprender com os colegas em momentos de trocas de ideias e dificuldades nas demandas, sempre focando na solução dos problemas e aprendizagem.





framework
PADAWANS
EPISÓDIO VI