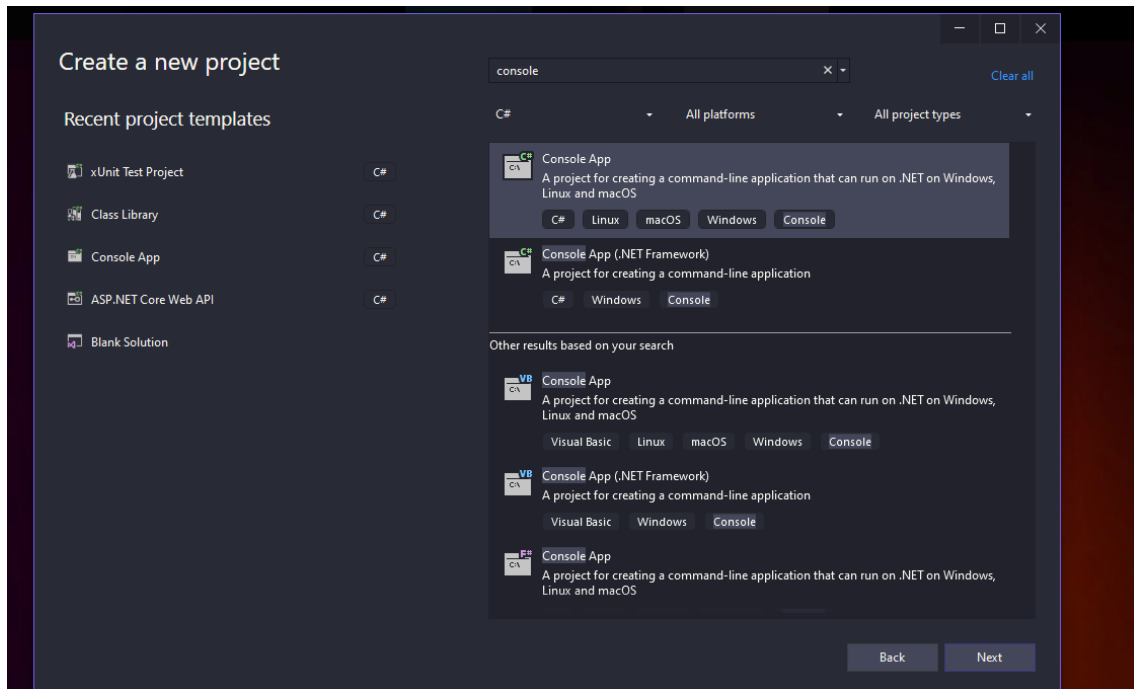
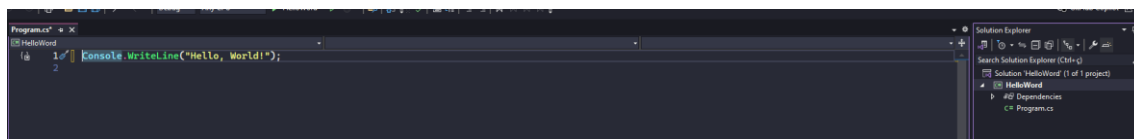


# Módulo 01

## #01 criando o primeiro projeto



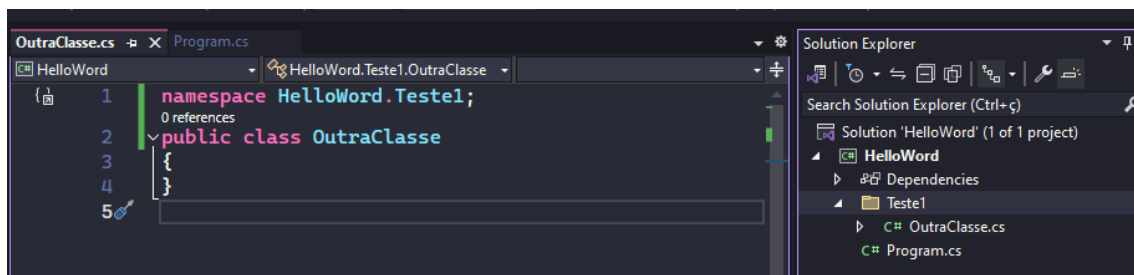
E da next e create

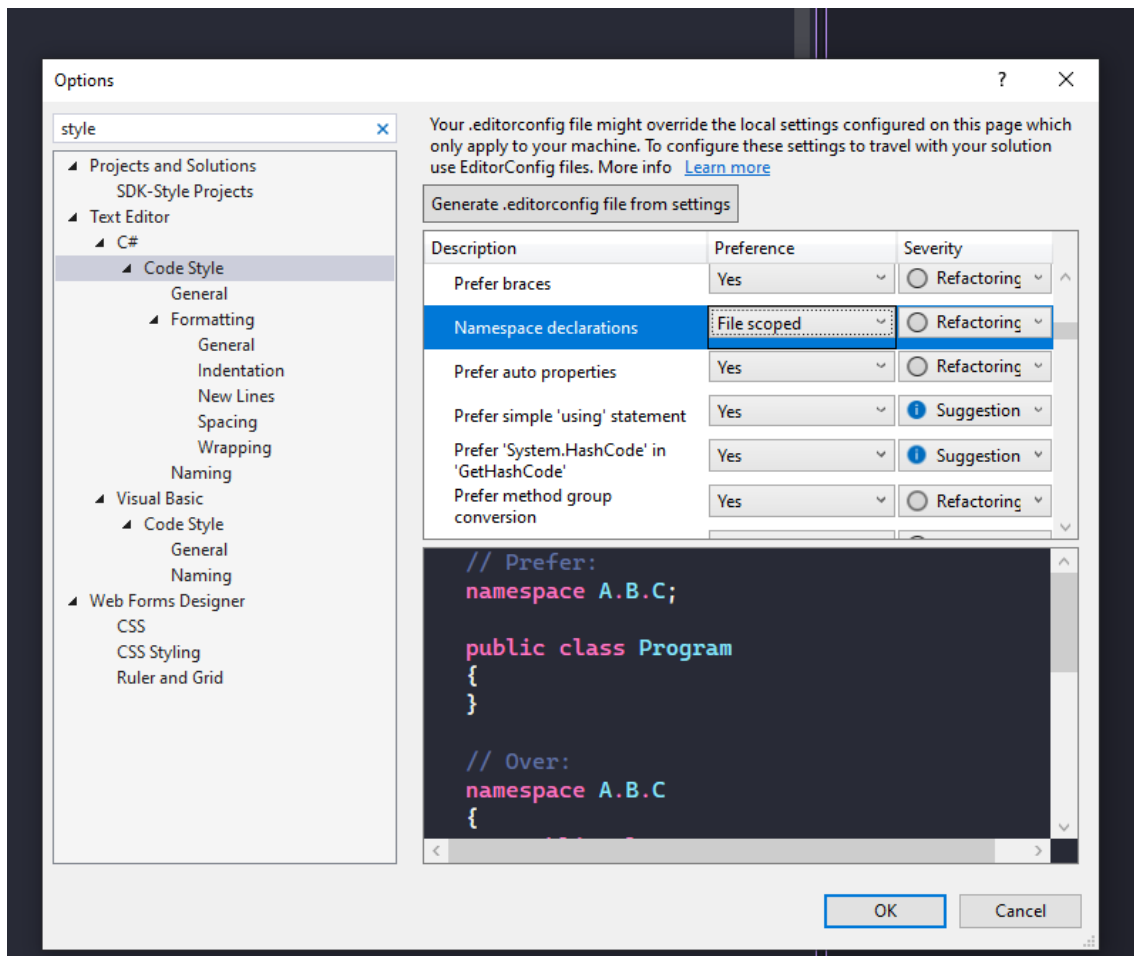


Projeto é onde implementa o código, solutions é um container onde agrupa projetos relacionados.

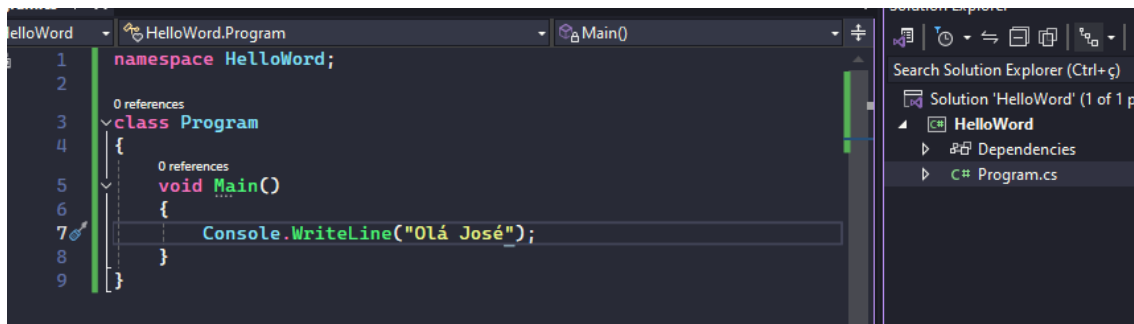
## #02 O que é namespace

É a forma de organizar as classes dentro do projeto





#03 O que são funções é algo que executa um bloco de código



#04 Utilizando outras classes, cria um classe carro com as função desligar e ligar



```

1 namespace HelloWord;
2
3 class Program
4 {
5     void Main()
6     {
7         Carro meuCarro = new Carro();
8         meuCarro.Ligar();
9         meuCarro.Desligar();
10    }
11 }

```

**#05 Modificadores de acesso**, public qualquer classe pode acessar, private somente a classe pode usar, protected somente quem instância pode usar, internal somente que tá dentro do mesmo projeto ou namespace

```

13
14 private void Teste1()
15 {
16     Console.WriteLine("Teste 1");
17 }
18
19 internal void Teste2()
20 {
21     Console.WriteLine("Teste 2");
22 }
23

```

**#06 classes públicas sem usings** entra onde vc instalou o visual em code 1033 e class e depois abra no notepad e salve

```

1 namespace $rootnamespace$
2 {
3     public class $safetitemrootname$
4     {
5     }
6 }

```

**#07 tipos numéricos**

**#08 tipos booleano** aceita apenas valores verdadeiro e falso

```

0 references
void Main()
{
    bool ativo = false;
    bool ativo2 = true;
}

```

**#09 tipos de texto**

```

5 static void Main()
6 {
7     char letra = 'a'; // apenas 1 caracteres
8     string texto = "hello"; // textos ou frases
9     char primeiraLetra = texto[0]; // pega o h do texto
10
11     string removeSpaco = "    dawd ".Trim(); // remove os espaço inicio e fim
12     bool começaCom = texto.StartsWith("h"); // verifica se começa com
13     bool terminaCom = texto.EndsWith("h"); // verifica se termina com
14
15     string troca = texto.Replace('e', 'u'); // troca o e por u
16
17     bool existe = texto.Contains("e"); // verifica se tem "e" dentro do texto
18     bool existe2 = texto.Equals("e"); // verifica se é igual o texto

```

#10 operações com texto, são concatenação com strings

#11 datas e horas

```

namespace HelloWorld;

0 references
class Program
{
    //      d: 6/15/2008
    //      D: Sunday, June 15, 2008
    //      f: Sunday, June 15, 2008 9:15 PM
    //      F: Sunday, June 15, 2008 9:15:07 PM
    //      g: 6/15/2008 9:15 PM
    //      G: 6/15/2008 9:15:07 PM
    //      m: June 15
    //      o: 2008-06-15T21:15:07.0000000
    //      R: Sun, 15 Jun 2008 21:15:07 GMT
    //      s: 2008-06-15T21:15:07
    //      t: 9:15 PM
    //      T: 9:15:07 PM
    //      u: 2008-06-15 21:15:07Z
    //      U: Monday, June 16, 2008 4:15:07 AM
    //      y: June, 2008
    //
    //      'h:mm:ss.ff t': 9:15:07.00 P
    //      'd MMM yyyy': 15 Jun 2008
    //      'HH:mm:ss.f': 21:15:07.0
    //      'dd MMM HH:mm:ss': 15 Jun 21:15:07
    //      '\Mon\t\h\ M': Month: 6
    //      'HH:mm:ss.ffffzzz': 21:15:07.0000-07:00

    0 references
    static void Main()
    {
        // apenas data
        DateOnly dia = new DateOnly(2024, 12, 1); // 01-janeiro-01

        string diaEmTexto = dia.ToString("dd/MM/yyyy", new CultureInfo("pt-BR")); // 01/12/2024

        DateTime dia1 = new DateTime(2024, 12, 1, 20, 00, 01); // 2024/12/01 20:00:01

        DateTime diaAgora = DateTime.Now; // pega data agora data e hora
        DateTime diaAgora2 = DateTime.Today; // pega data agora data
        DateTime diaAgora3 = DateTime.UtcNow; // pega data Mundial
        DateTime nvData = diaAgora.AddDays(1); // adiciona um dia
    }
}

```

#12 enums

```

3  class Program
4  {
5      2 references
6      enum NivelDeDificuldade
7      {
8          baixo = 0,
9          medio = 5,
10         alto = 10,
11     }
12
13     0 references
14     static void Main()
15     {
16         NivelDeDificuldade nivel = NivelDeDificuldade.baixo; // baixo
17         int nivelInteiro = (int)nivel; // 0
18     }
19 }

```

#13 var pode receber qualquer tipo para a variável, o compilador escolhe o tipo para você

```

0 references
static void Main()
{
    var teste = "bom dia"; // bom dia
}

```

#14 objects é do tipo obeto, usa para não definir um objeto

```

static void Main()
{
    object meutexto = "jose romario"; // jose romario
}

```

#15 null é um valor nullo

```

0 references
static void Main()
{
    int? idade = null; // aceita valores do tipo null e int
}

```

#16 arrays armazena coleção de valores

```

0 references
static void Main()
{
    int[] inteiro = new int[3]; // tem 3 tamanho o array [0,0,0]
    inteiro[1] = 20; // [0,20,0]

    int[] inteiro2 = [1,2,3]; // [1,2,3]

    int[,] duasDimencao = new int[2, 2]; // [0,0] [0,0]
    duasDimencao[0, 1] = 2; // [0,2] [0,0]
    duasDimencao[1, 0] = 4; // [0,2] [4,0]
}

```

#17 listas

```

{
    List<int> inteiro = new List<int>(); // usa o Count para contar quantos tem
    inteiro.Add(1); // 1 tamanho pode acessar pelo indice tbm inteiro[0] = 1
    // inteiro.Remove(1) remove o elemento e indice passa o elemento
    // inteiro.First() pega o primeiro elemento
    // inteiro.RemoveAt(0) remove pelo indice
    // inteiro.Clear() remove todos elementos
}

```

## #18 um pouco mais sobre listas

```

static void Main()
{
    List<object> obj = new List<object>();

    obj.Add("carro");
    obj.Add(1);
    obj.Add(false);

    List<string> list = new List<string>();

    list.Add("Hello");
    list.Add("World");

    string result = string.Join(", ", list); // 0 Join junta = hello, world
}

```

## #19 dicionário

```

0 references
static void Main()
{
    //chave valor
    Dictionary<int, string> dicionario = new Dictionary<int, string>();

    dicionario.Add(1, "maria");
    dicionario.Add(2, "pedro");
    dicionario.Add(3, "julia");

    string value = dicionario[3]; // julia
    bool value2 = dicionario.ContainsKey(2); // true
}

```

## #20 hashset

```

0 references
static void Main()
{
    //os valores tem q ser único
    HashSet<int> hashset = new HashSet<int>();

    hashset.Add(1);
    hashset.Add(2);
    hashset.Add(3);
    hashset.Add(3);
}

```

## #21 funções com parâmetros

```
namespace HelloWorld;
1 reference
public class Operacoes
{
    1 reference
    public void Adicionar(int v1, int v2)
    {
        var resultado = v1 + v2;
    }
}
```

Search Solution Explorer (Ctrl+ç)  
Solution 'HelloWord' (1 of 1 project)  
HelloWord  
Dependencies  
C# Operacoes.cs  
C# Program.cs

```
0 references
class Program
{
    0 references
    static void Main()
    {
        var matematica = new Operacoes();
        matematica.Adicionar(v1: 5, v2: 5); // 10
    }
}
```

## #22 funções devolvendo valores

```
1 reference
public class Operacoes
{
    1 reference
    public int Adicionar(int v1, int v2)
    {
        var resultado = v1 + v2;
        return resultado;
    }
}
```

```
0 references
static void Main()
{
    var matematica = new Operacoes();
    var resultado = matematica.Adicionar(v1: 5, v2: 5); // 10
    Console.WriteLine(resultado);
}
```

Ou pode fazer assim

```
public class Operacoes
{
    1 reference
    public int Adicionar(int v1, int v2) => v1 + v2;
}
```

Retornar 2 valores

```

1 reference
2 public class Operacoes
3 {
4     1 reference
5     public (int, string) Adicionar(int v1, int v2)
6     {
7         var resultado = v1 + v2;
8         return (resultado, "teste");
9     }
10 }

```

```

0 references
static void Main()
{
    var matematica = new Operacoes();

    var resultado = matematica.Adicionar(v1: 5, v2: 5); // 10

    Console.WriteLine(resultado.Item1); // int valor 1
    Console.WriteLine(resultado.Item2); // string valor 2
}

```

Pode criar 2 variável também

```

0 references
static void Main()
{
    var matematica = new OperacoesMatematicas();

    (int resultado, string nome) = matematica.Adicionar(7, 3);

    Console.WriteLine(resultado);
    Console.WriteLine(nome);
}

```

#23 parâmetros opcionais somente os últimos valores pode ser opcional

```

7
8     return (resultado, "teste");
9 }
10
11 0 references
12 public void Teste(int valor1, int valo2 = 7) // transforma o valor 2 em opcional
13 {
14     Console.WriteLine(valor1 + valo2);
15 }
16

```

#24 criando classes com valores

```

Cor.cs  Carro.cs  Program.cs
C# HelloWorld  HelloWorld.Cor
1 namespace HelloWorld;
2 1 reference
3 public enum Cor
4 {
5     Vermelho,
6     Azul,
7     Amarelo
8 }

```



```

2
3 namespace HelloWorld;
4 2 references
5 public class Carro
6 {
7     4 references
8     public required string Modelo { get; set; } // o required obriga a preencher o valores
9     1 reference
10    public DateOnly LancadoEm { get; set; }
11    1 reference
12    public Cor Cor { get; set; }
13
14    1 reference
15    public Carro(string modelo) // um construtor da classe
16    {
17        Modelo = modelo;
18    }
19
20    1 reference
21    public void NomeDoModelo() => Console.WriteLine(Modelo);
22 }

```

```

4
5 0 references
6 class Program
7 {
8     0 references
9     static void Main()
10    {
11        var carro = new Carro("Teste") { Modelo = "dwd" };
12        carro.Modelo = "Palio"; // adicionar valores as propriedades
13        carro.Cor = Cor.Vermelho;
14        carro.LancadoEm = new DateOnly(2020, 01, 01);
15        carro.NomeDoModelo();
16    }
17 }

```

#25 o que é static usa o static assim não precisa instanciar a classe para poder usar a função

```

11 public void teste(int valor1, int valor2) // transforma o valor 2 em optional
12 {
13     Console.WriteLine(valor1 + valor2);
14 }
15
16 0 references
17 public static int Adicionar2(int valor1, int valor2) => valor1 + valor2;
18 }

```

E usa ela assim com o nome da classe e a função

```

7
8 0 references
9 static void Main()
10 {
11     var resultado = Operacoes.Adicionar2(1, 20);
12 }

```

```

1 namespace HelloWorld;
2 1 reference
3 public class Carro
4 {
5     2 references
6     public static string Modelo { get; set; } // é compartilhado com todos por ser static
7     0 references
8     public DateOnly LancadoEm { get; set; }
9     0 references

```

Pode fazer uma classe static também, porém uma classe static não pode ser instanciada

```

1 namespace HelloWorld;
2 1 reference
3 public static class Operacoes
4 {
5     0 references

```

#26 o que é e como funciona o debug clica no canto da linha e ficará um bolinha aí roda o programa aí ele para ali e aperta f10 para ir próxima linha e entrar numa função é f11

```
3 class Program
4 {
5     0 references:
6     static void Main()
7     {
8         int numero = 10;
9         var meuCarro = new Carro("Porsche")
10     }
```

## #27 if else e else if parte 1

```
int numero = 10;
double saldo = 100.50;
bool ativo = true;
string autor = "Welisson";
List<int> lista = new List<int> { 1, 7 };
Cores cor = Cores.Azul;

/*
 > Maior
 < Menor
 >= Maior ou igual
 <= Menor ou igual
 != diferente
 */

if (numero > 0)
{
    Console.WriteLine("Este numero é positivo");
}
```

```
if (numero > 0)
{
    Console.WriteLine("Este numero é POSITIVO");
}
else
{
    Console.WriteLine("Este numero é NEGATIVO");
}
```

```
*/

if (numero > 0)
{
    Console.WriteLine("Este numero é POSITIVO");
}
else if (numero == 0)
{
    Console.WriteLine("Este numero é NEUTRO");
}
else
{
    Console.WriteLine("Este numero é NEGATIVO");
}
```

Para string usa o Equals

```
if (autor.Equals("Edilaine"))
{
    Console.WriteLine("ENTROUUU");
}
```

## #28 if else e else if parte 2

```
if (cor == Cores.Azul || numero >= 0)
{
    Console.WriteLine("ENTROUUU");
}
```

```
string mensagemDeErro = null;
if (mensagemDeErro is not null)
{
    Console.WriteLine("ENTROUUU");
}
```

```
if ((numero == 0 && saldo > 100.0) || ativo)
{
    Console.WriteLine("ENTROUUU");
}
```

## #29 condicional ternário

```
static void Main()
{
    int numero = 7;
    string autor = numero == 7 ? "welisson" : "willian";
    Console.WriteLine(author);
}
```

## #30 switch

#31

#32

#33

#34

#35

#36

#37

#38

#39

#40

#41

#42

#43

#44

#45

#46

#47

#48

#49

#50

#51

#52

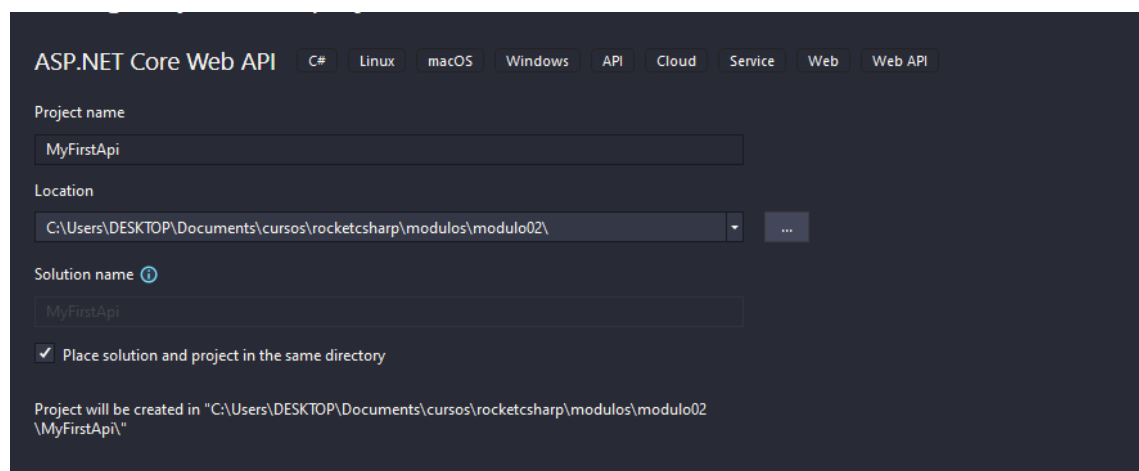
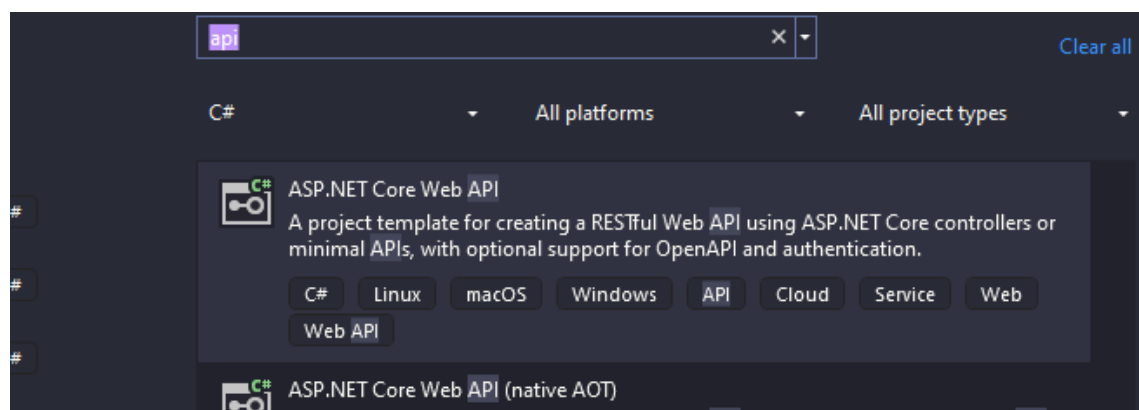
#53

#54

#55

#56

## #25 Criando uma api



F5 roda o projeto

## #26 Properties

```
on.schemastore.org/launchsettings.json
{
  "$schema": "http://json.schemastore.org/launchsettings.json",
  "iisSettings": {
    "windowsAuthentication": false,
    "anonymousAuthentication": true,
    "iisExpress": {
      "applicationUrl": "http://localhost:40592",
      "sslPort": 44325
    }
  },
  "profiles": {
    "http": {
      "commandName": "Project",
      "dotnetRunMessages": true,
      "launchBrowser": true,
      "launchUrl": "swagger",
      "applicationUrl": "http://localhost:5149",
      "environmentVariables": {
        "ASPNETCORE_ENVIRONMENT": "Development"
      }
    },
    "https": {
      "commandName": "Project",
      "dotnetRunMessages": true,
      "launchBrowser": true,
      "launchUrl": "swagger",
      "applicationUrl": "https://localhost:7037;http://localhost:5149",
      "environmentVariables": {
        "ASPNETCORE_ENVIRONMENT": "Development"
      }
    },
    "IIS Express": {
      "commandName": "IISExpress",
      "launchBrowser": true,
      "launchUrl": "swagger",

```

## #27 O arquivo program.cs

```
var builder = WebApplication.CreateBuilder(args);

// Add services to the container.

builder.Services.AddControllers();
// Learn more about configuring Swagger/OpenAPI at https://aka.ms/aspnetcore/swashbuckle
builder.Services.AddEndpointsApiExplorer();
builder.Services.AddSwaggerGen();

var app = builder.Build();

// Configure the HTTP request pipeline.
if (app.Environment.IsDevelopment())
{
    app.UseSwagger();
    app.UseSwaggerUI();
}

app.UseHttpsRedirection();

app.UseAuthorization();

app.MapControllers();

app.Run();
```

## #28 função appsetting.json

```
1 1 {
2 2   "Logging": {
3 3     "LogLevel": {
4 4       "Default": "Information",
5 5       "Microsoft.AspNetCore": "Warning"
6 6     }
7 7   },
8 8   "AllowedHosts": "*"
9 9 }
10
```

## #29 Lendo appsettings.json

```
appsettings.Dvelopment.json  X appsettings.json
Schema: https://json.schemastore.org/appsettings.json
1 1 {
2 2   "props2": "romario"
3 3 }
4 4
```

```
1 https://json.schemastore.org/appsettings.json
2 {
3   "Logging": {
4     "LogLevel": {
5       "Default": "Information",
6       "Microsoft.AspNetCore": "Warning"
7     },
8     "AllowedHosts": "*",
9   },
10   "props1": "Giovana"
11 }
12
```

```
6 // Learn more about configuring Swagger/OpenAPI at https://aka.ms/aspnetcore/swashbuckle
7 builder.Services.AddEndpointsApiExplorer();
8 builder.Services.AddSwaggerGen();
9
10 var teste = builder.Configuration.GetSection("props1").Value; // pegar o valor
11
12 var app = builder.Build();
```

pegar valor dentro de um objeto, sempre da prioridade para o development caso tenha obj com o nomes igual

```
    "Logging": {
      "LogLevel": {
        "Default": "Information",
        "Microsoft.AspNetCore": "Warning"
      },
    },
    "AllowedHosts": "*"
  },
  "props1": "Giovana"
}
```

```
9
10 var teste = builder.Configuration.GetSection("Logging").GetSection("LogLevel").Value; // pegar o valor
11
```

Pegar o valor direto

```
> var test = builder.Configuration.GetValue<int>("MyClass:Number");
```

#30 oque é um controler

#31

#32

#33

#34

#35

#36

#37

#38

#39

#40

#41

**#42**

**#43**

**#44**

**#45**

**#46**

**#47**

**#48**

**#49**

**#50**

**#51**

**#52**

**#53**

**#54**

**#55**

**#56**