



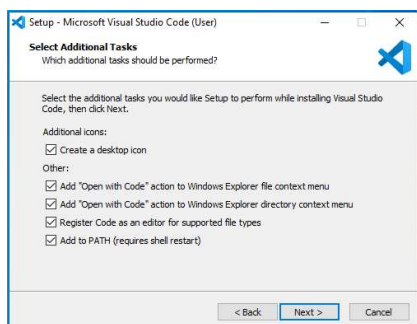
AULA 01 - INAUGURAL - DESENVOLVIMENTO DE SISTEMAS

- Apresentação da disciplina
- Competências, habilidades e bases tecnológicas da disciplina
- Formas de Avaliação
- Introdução e desenvolvimento do conteúdo

Download e Instalação do Visual Studio Code

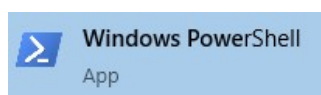
Faça o Download do Visual Studio Code acessando <https://code.visualstudio.com/download>

Instale deixando todas as opções selecionadas



Linhas de Comandos PowerShell

Para criação de projetos podemos utilizar a ferramenta de linha de comandos do Windows chamada de powershell para ao menos navegar



Verificando a versão do .NET Core instalado

```
PS C:\Users\luiz> dotnet --version
7.0.100
```

```
PS C:\Users\luiz> dotnet --list-sdks
5.0.408 [C:\Program Files\dotnet\sdk]
6.0.301 [C:\Program Files\dotnet\sdk]
7.0.100 [C:\Program Files\dotnet\sdk]
```

Se seu computador não exibir nenhuma versão ou não reconhecer o comando, instale o .Net através do link a seguir: <https://dotnet.microsoft.com/en-us/download/dotnet>. A versão recomendada para as aulas é a 7.0

Version	Release type	Support phase	Latest release	Latest release date	End of support
.NET 7.0 (latest)	Standard Term Support ⓘ	Active ⓘ	7.0.2	January 10, 2023	May 14, 2024
.NET 6.0	Long Term Support ⓘ	Active ⓘ	6.0.13	January 10, 2023	November 12, 2024



Vá em computador, clique com o direito do mouse e em propriedades, para verificar se seu Windows é 32 ou 64 bits e faça o download compatível com seu computador

OS	Installers	Binaries
Linux	Package manager instructions	ARM32 ARM64 RHEL 6 x64 x64 x64 Alpine
macOS	x64	x64
Windows	x64 x86	ARM32 x64 x86
All	dotnet-install scripts	

Comandos básicos

Mudar pasta → cd (change directory) - Ex:

```
PS C:\Users\luizfsgs> cd d:\
PS D:\> cd etec
PS D:\etec>
```

Listar arquivos e pastas → ls (list)

```
PS D:\etec> ls

Diretório: D:\etec

Mode                LastWriteTime         Length Name
----                -
-a----             07/02/2023   11:51             0 Arquivo Aula 01.txt
-a----             07/02/2023   11:51             0 Arquivo Exemplo.txt
```

Criar Pasta → mkdir (make directory)

```
PS D:\etec> mkdir PastaTeste

Diretório: D:\etec

Mode                LastWriteTime         Length Name
----                -
d-----             07/02/2023   11:52             PastaTeste
```

- Navegue até a pasta recém criada → `cd PastaTeste`
- Estando dentro da pasta em que deseja abrir o VS Code realize o comando → `code .`
- Estes caminhos se referem as pastas locais de exemplo, você poderá criar e listar os diretórios da maneira que achar melhor



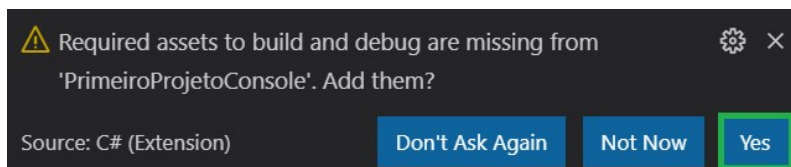
Criação de Projeto no Visual Studio Code

- Crie uma pasta onde costuma guardar os projetos do curso através dos comandos revisados até aqui.
Sugestão: criar uma pasta no endereço **d:/2DS/SEU_NOME/DS** (No laboratório)
- Certifique de que não é um caminho longo e que não existem caracteres especiais nas pastas.

Abra a pasta recém-criada e crie outra chamada **HelloWorld**. Abra o VS Code escolha a opção → File → Open Folder, selecionando a pasta **HelloWorld**. Vá até o menu View → Terminal e realize os comandos a seguir:

- Digite o comando `dotnet new -h`. Ele exibe os tipos de projetos que podem ser criados. Criaremos do tipo console.
- Para criar um projeto utilize a linha de comando a seguir `dotnet new Console` (--framework netX.0) em caso de versão específica, sendo X o número da versão.

Ao clicar no arquivo Program.cs ou qualquer arquivo com a extensão “.cs”, o vs code solicita instalar e extensão C# para habilitar a depuração. Para essa pergunta sempre clique em **Yes**.

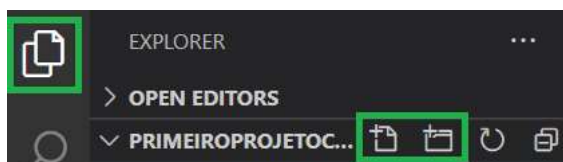


- Para compilar o projeto navegue até o menu View → Terminal e execute `dotnet build`
- O resultado esperado é sempre 0 erros
- Para rodar o projeto execute o comando a seguir no terminal `dotnet run`

Extensões importantes para o VS Code: É possível adicionar extensões ao VS Code. Vá até até até o menu View → Extensions e na caixa de busca, digite as extensões listada abaixo e clique no botão install

- C# Extensions – Autor: JosKreativ
- Material Icon Theme – Autor: Philipp Kief
- C# For Visual Studio Code – Autor: Microsoft (provavelmente já estará adicionada)

Podemos observar os arquivos abertos conforme a imagem abaixo e os ícones em que podemos criar arquivos e pastas, sendo possível criar arquivos e pastas clicando com o botão direito.



- O arquivo `Program.cs` é uma classe e é o ponto de partida para a execução do projeto.



Realize a programação abaixo. **Atenção:** A partir da versão 6.0 do .net o projeto passa a exibir apenas a linha do Hello, World, mas internamente existe esta estrutura presente no print abaixo.

```
using System; A

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

- Execute o comando *dotnet build* para compilar
- Execute o comando *dotnet run* para executar

Observações

- (A) Referência para classes do dotnet ou componentes para programação.
- (B) Namespace do arquivo: É um endereço lógico que representa em que hierarquia de pasta o arquivo está contido.
- (C) Declaração de classe Program, sendo a assinatura a descrição da classe e tudo que está entre as chaves sendo o corpo desta classe
- (D) Método principal da classe. Necessário para inicialização do programa. O comando Console.Write

Altere o programa conforme abaixo:

```
static void Main(string[] args)
{
    Console.WriteLine("Digite seu nome:");
    string nome = Console.ReadLine();

    Console.WriteLine($"Seu nome tem {nome.Length} caracteres.");
    Console.ReadKey();
}
```

Console.WriteLine("mensagem") → Exibe mensagem

Console.ReadLine() → Lê os caracteres digitados para uma variável

Console.ReadKey() → Aguarda uma digitação para encerrar a execução

\$ → Combinado com aspas duplas escreve uma mensagem reservando espaço para uma variável caso chaves sejam inseridas.

- Execute para testar



Acrescente a programação a seguir no corpo da classe após o método ReadKey() e execute novamente.

```
Console.WriteLine("Digite a data do seu nascimento: ");
DateTime dtNascimento = DateTime.Parse(Console.ReadLine());

int qtdDiasVividos = DateTime.Now.Subtract(dtNascimento).Days;
Console.WriteLine("Os dias vividos até hoje são: " + qtdDiasVividos);

Console.ReadKey();
```

Trabalhando com variáveis

Feche o projeto anterior através do menu File → Close Folder. Retorne ao *explorer* para e crie uma nova pasta chamada **Aula01Variaveis**. Abra a pasta criada no VS Code através do Menu File → Open Folder.

1. Realize a programação abaixo onde teremos aprendizado de concatenações de variáveis string, datetime e decimal, compilando e executando logo após para testar.

```
namespace Aula02Exemplos
{
    0 references
    class Program
    {
        0 references
        static void Main(string[] args)
        {
            Console.WriteLine("Digite seu nome: ");
            string nome = Console.ReadLine();

            string frase1 = $"Olá {nome}, hoje é {DateTime.Now}";
            Console.WriteLine(frase1);
            Console.WriteLine(" ");

            Console.WriteLine("Quanto custa um dólar em reais? ");
            decimal valorDolarReais = decimal.Parse(Console.ReadLine());
            string frase2 = string.Format("Hoje é {0:ddd}, o dólar está custando {1:c2} ", DateTime.Now, valorDolarReais);
            Console.WriteLine(frase2);

            Console.WriteLine(" ");
            string cabecalho = string.Format("{0:ddd}, {0:dd} de {0:MMM} de {0:yyyy} - {0:HH:mm:ss}", DateTime.Now);
            Console.WriteLine(cabecalho.ToUpper());
        }
    }
}
```




2. Podemos criar **Métodos** para programar as operações em um bloco isolado e fazer a chamada desse bloco no método principal. O método se chamará *ConcatenarPalavras*, conforme abaixo

```
static void Main(string[] args)
{
    ConcatenarPalavras();
}

1 reference
public static void ConcatenarPalavras()
{
    Console.WriteLine("Digite seu nome: ");
    string nome = Console.ReadLine();

    string frase1 = $"Olá {nome}, hoje é {DateTime.Now}";
    Console.WriteLine(frase1);
    Console.WriteLine(" ");

    Console.WriteLine("Quanto custa um dólar em reais? ");
    decimal valorDolarReais = decimal.Parse(Console.ReadLine());
    string frase2 = string.Format("Hoje é {0:ddd}, o dólar está custando {1:c2} ", DateTime.Now, valorDolarReais);
    Console.WriteLine(frase2);

    Console.WriteLine(" ");
    string cabecalho = string.Format("{0:ddd}, {0:dd} de {0:MMMM} de {0:yyyy} - {0:HH:mm:ss}", DateTime.Now);
    Console.WriteLine(cabecalho.ToUpper());
}
```

3. Crie mais um método, imediatamente abaixo do fechamento do método anterior. Esse método poderá verificar se a data digitada é um final de semana ou não.

```
public static void VerificarAulaEtec()
{
    Console.WriteLine("Digite a data");
    DateTime data = DateTime.Parse(Console.ReadLine());

    if(data.DayOfWeek == DayOfWeek.Saturday || data.DayOfWeek == DayOfWeek.Sunday)
    {
        Console.WriteLine("Final de semana! Hoje não tem aula! Revisarei exercícios.");
    }
    else
    {
        Console.WriteLine("Dia da semana! Bora pra Etec!");
    }
}
```

4. Faça a chamada do método recém-criado no método principal e realize o teste.

```
static void Main(string[] args)
{
    ConcatenarPalavras();
    VerificarAulaEtec();
}
```



5. Crie o método para calcular média

```
public static void CalcularMedia()
{
    Console.WriteLine("Digite a primeira nota");
    decimal nota1 = decimal.Parse(Console.ReadLine());

    Console.WriteLine("Digite a segunda nota");
    decimal nota2 = decimal.Parse(Console.ReadLine());

    decimal media = (nota1 + nota2) / 2;
    Console.WriteLine($"A média é {media}");

    if(media >= 7)
    {
        Console.WriteLine("Aprovado");
    }
    else if(media < 7 && media >= 4)
    {
        Console.WriteLine("Recuperação");
    }
    else
    {
        Console.WriteLine("Reprovado");
    }
}
```

6. Como já sabemos que podemos chamar vários métodos no método principal, fazemos a chamada apenas deste método. Até o final da aula conseguiremos criar uma maneira de escolher qual método vamos usar. Execute para testar

```
static void Main(string[] args)
{
    CalcularMedia();
}
```

7. Crie o método para Calcular a Tabuada

```
public static void CalcularTabuada()
{
    Console.WriteLine("Digite a tabuada que deseja calcular");
    int tabuada = int.Parse(Console.ReadLine());
    int contador = 0;

    while(contador <= 10)
    {
        string mensagem = string.Format("{0} X {1} = {2}", tabuada, contador, tabuada * contador);
        Console.WriteLine(mensagem);
        contador++;
    }
}
```



8. Editaremos o método principal para poder escolher qual método vamos querer usar

```
static void Main(string[] args)
{
    Console.WriteLine("Observe o menu abaixo e digite o número referente a opção desejada: ");
    Console.WriteLine("1 - Concatenar Palavras");
    Console.WriteLine("2 - Verificar Dia da Semana");
    Console.WriteLine("3 - Calcular Média");
    Console.WriteLine("4 - Calcular Tabuada");

    int opcaoEscolhida = int.Parse(Console.ReadLine());

    switch (opcaoEscolhida)
    {
        case 1:
            ConcatenarPalavras();
            break;
        case 2:
            VerificarAulaEtec();
            break;
        case 3:
            CalcularMedia();
            break;
        case 4:
            CalcularTabuada();
            break;
        default:
            Console.WriteLine("Opção Inválida");
            break;
    }
}
```

- Execute o programa para testar as alterações.