

Teste de múltiplas entradas

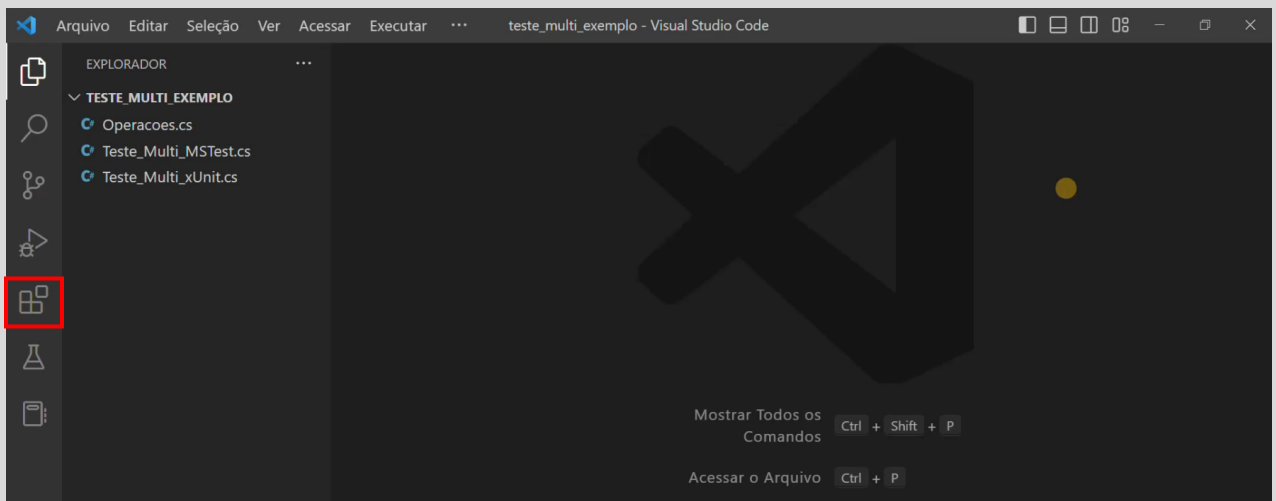
Introdução

Neste conteúdo, veremos como configurar o VS Code para projetos de teste com múltiplas entradas em C#, usando o xUnit e o MSTest.

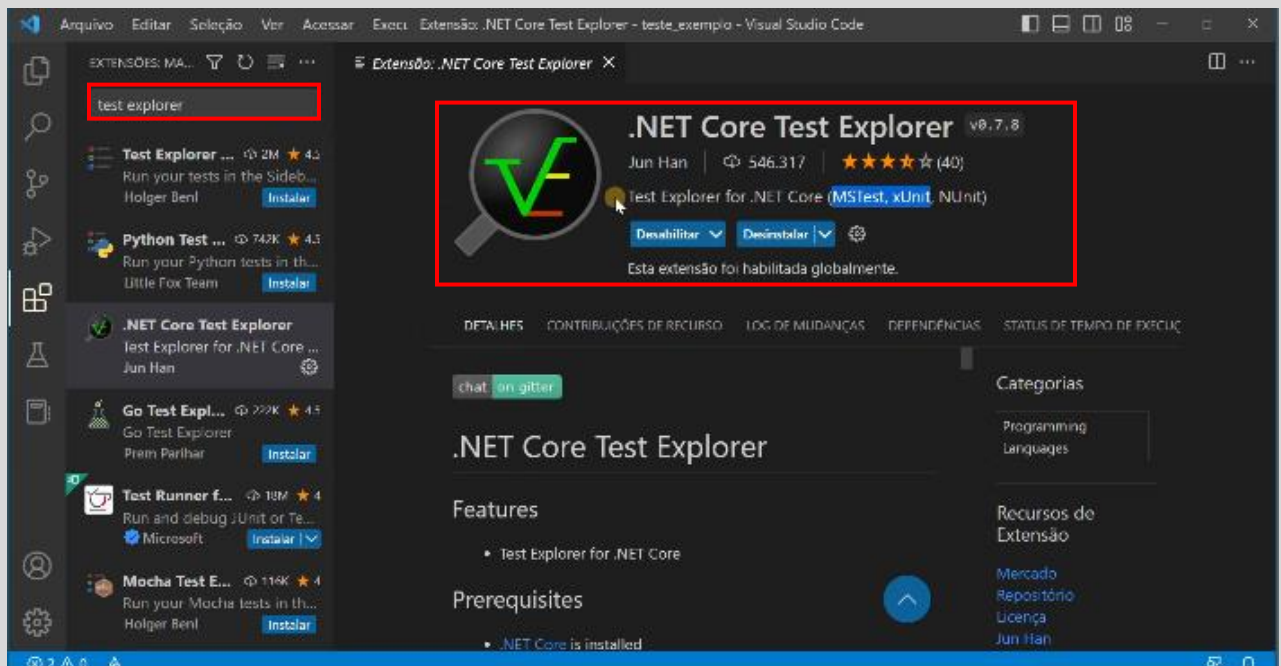
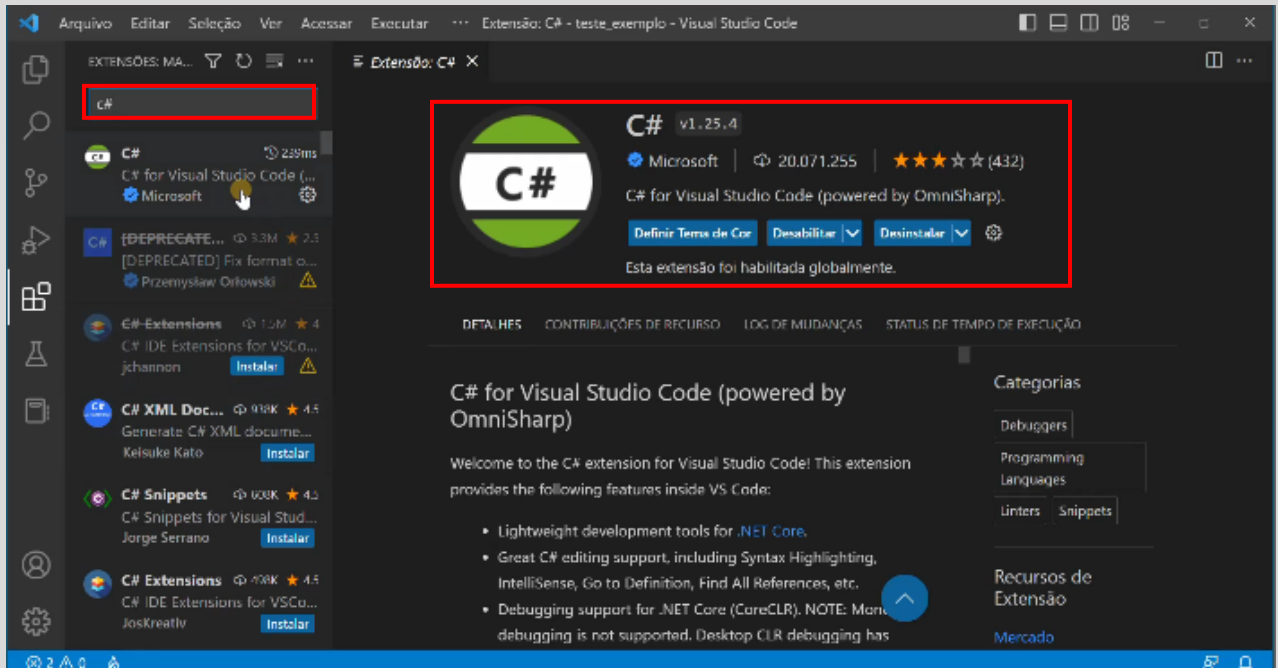
Para isso, você precisa ter instalado em sua máquina o VS Code com as extensões da linguagem C# e de teste.

Extensões

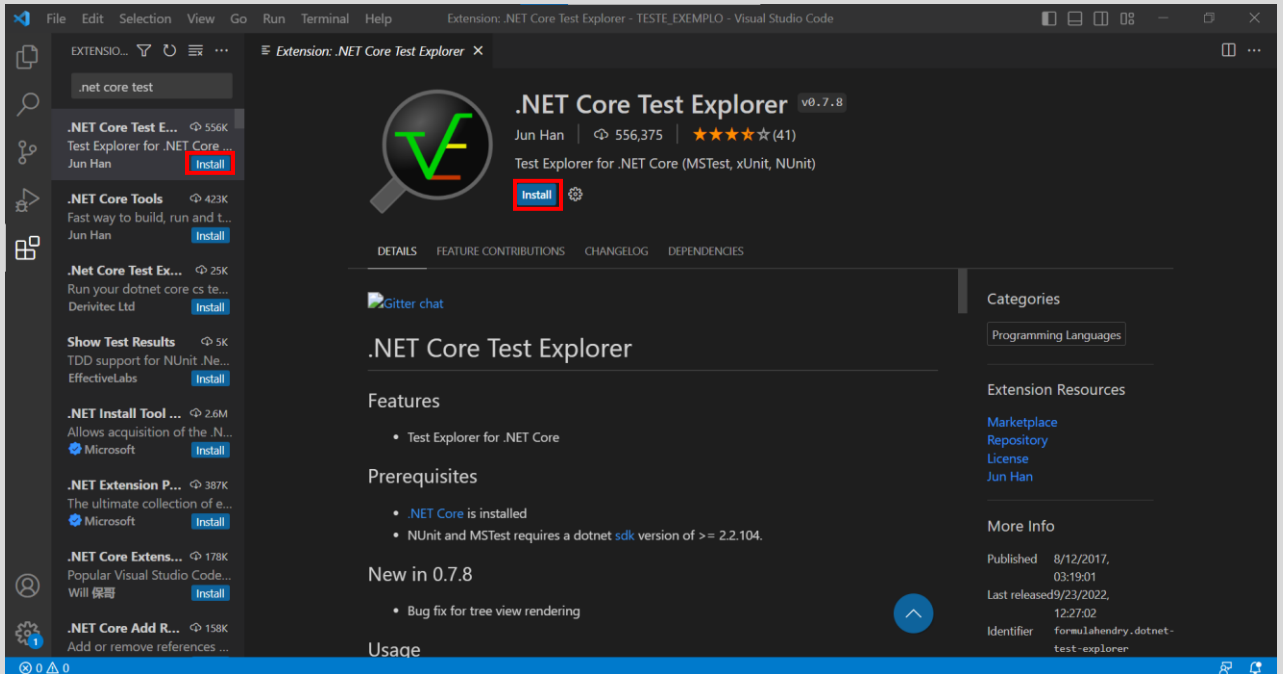
1. Clique em extensões (quinto ícone de cima para baixo no menu lateral esquerdo) para as instalar.



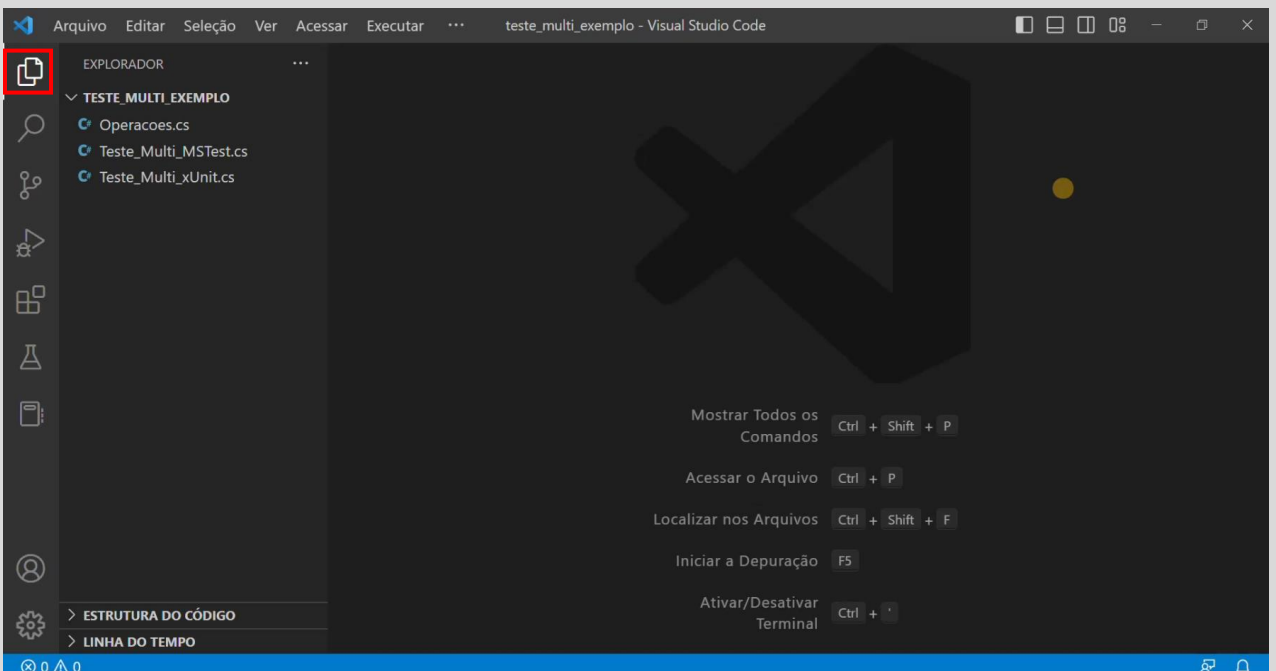
2. Na busca, digite parte do nome das extensões desejadas e dê **Enter**. As duas necessárias são **C# for Visual Studio Code** e **.NET Core Test Explorer**, as quais serão usadas tanto no xUnit quanto no MSTest.



3. Clique em **Install**.

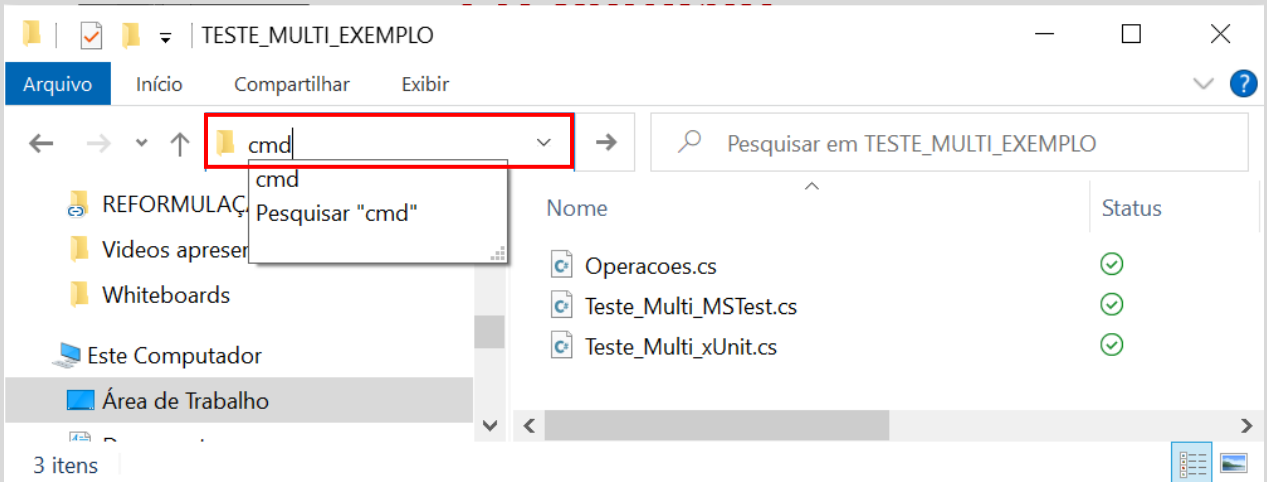


4. Clique no explorer (primeiro ícone de cima para baixo no menu lateral esquerdo) para voltar para a pasta do exemplo.

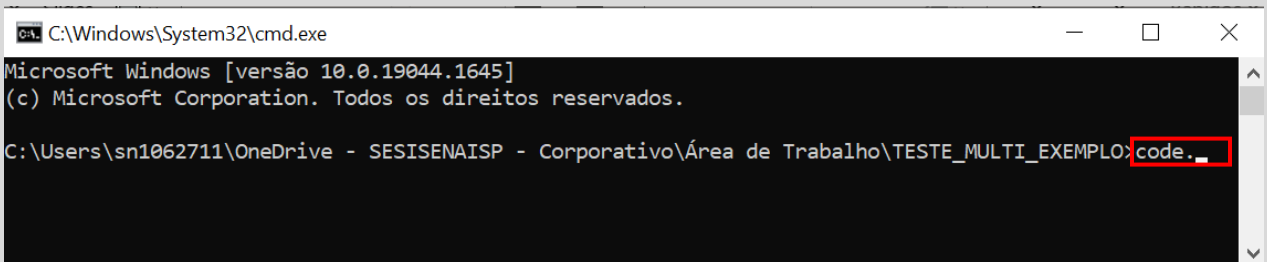


Preparação

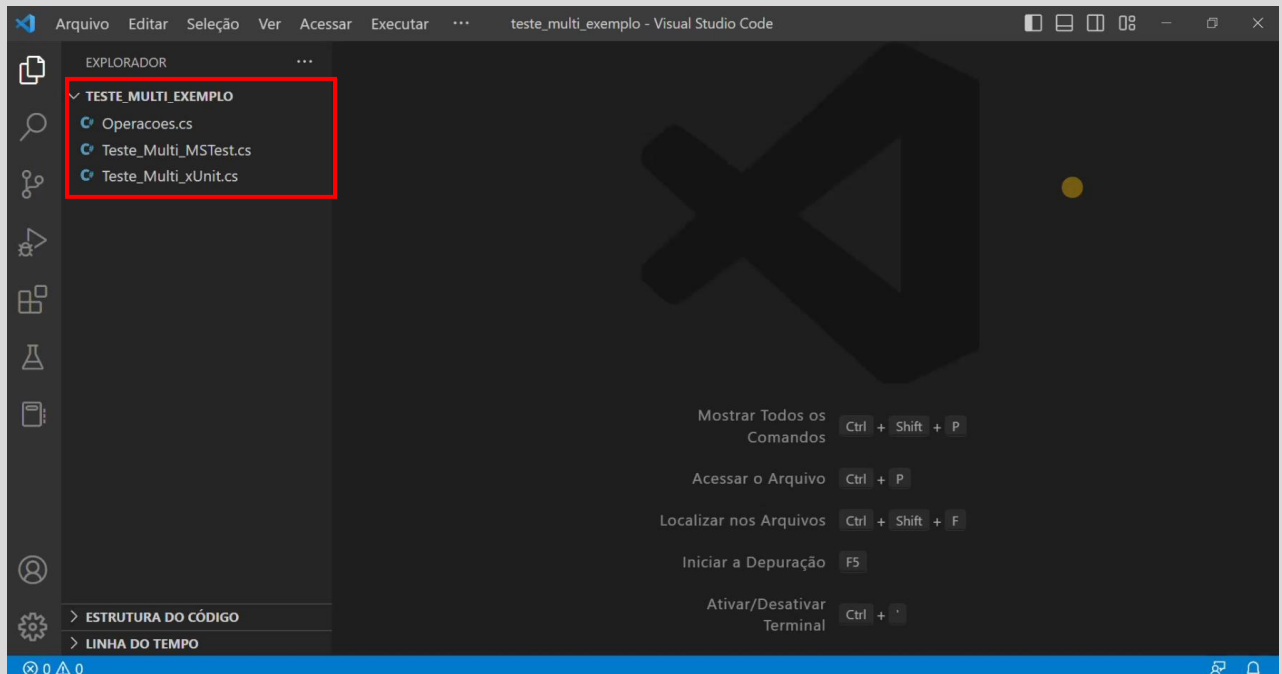
1. Com a pasta do exemplo aberta, digite “cmd” na barra de endereços e dê **Enter**.



2. No terminal, digite **code .** e dê **Enter** para abrir o VS Code.



3. O VS Code abrirá com a estrutura de pasta mapeada.

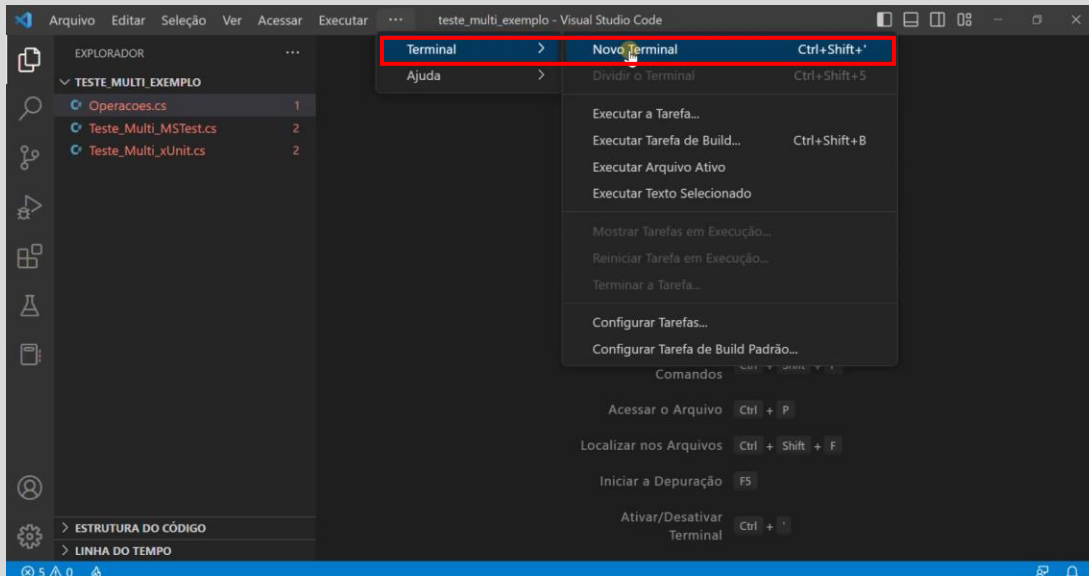


A classe `Operacoes.cs`, fornecida como arquivo de apoio, tem apenas dois métodos: um de soma e outro de multiplicação. O método **somar** precisa de dois parâmetros do tipo `double`: `primeiroNumero` e `segundoNumero`. O método **multiplicar** também precisa de dois parâmetros do tipo `double` (as mesmas variáveis do método `somar`).

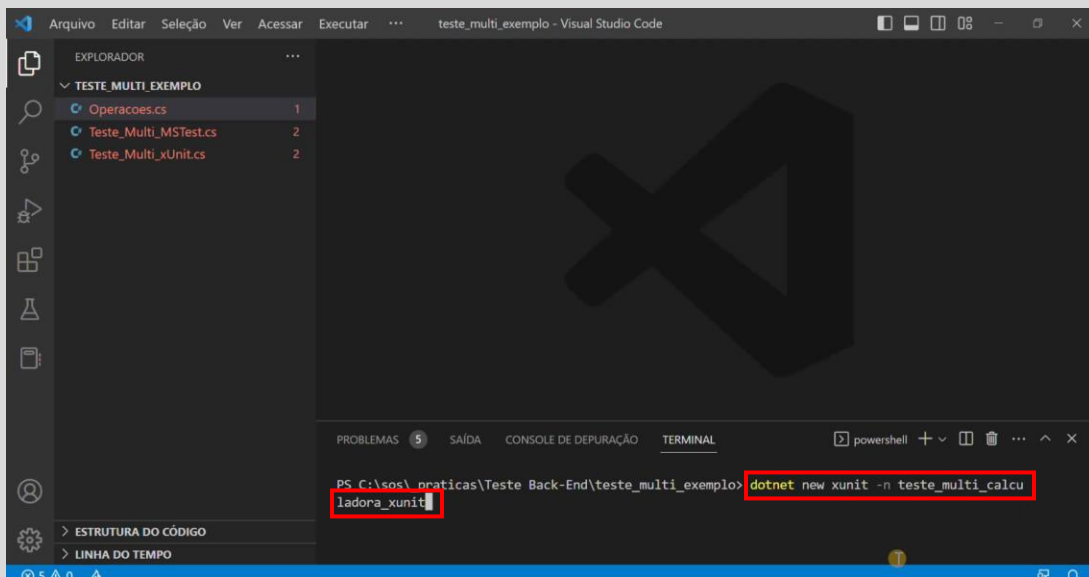
As outras duas classes são arquivos de apoio para os testes propriamente.

Teste com xUnit

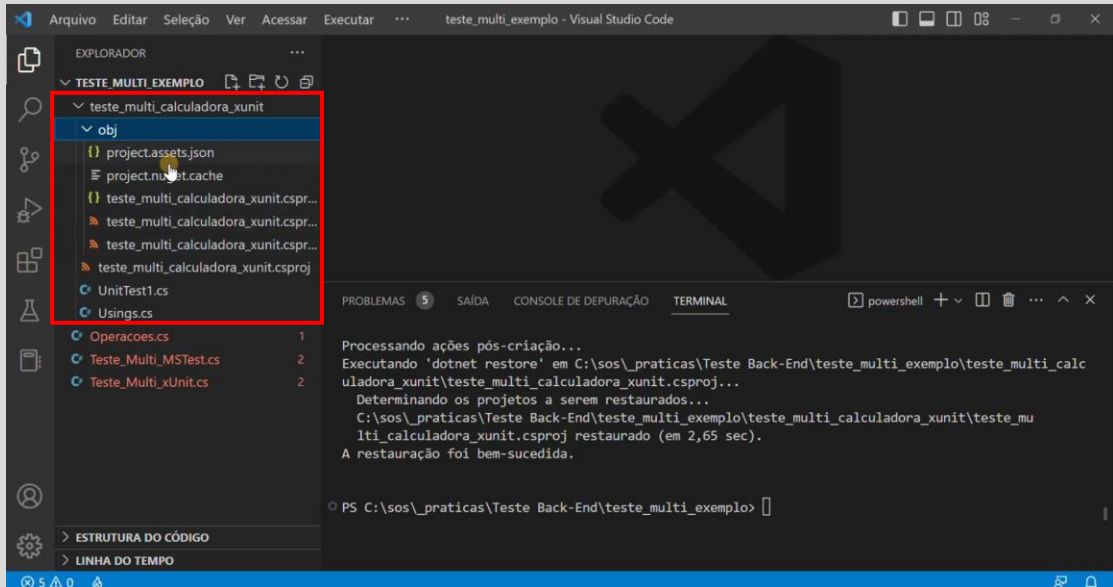
1. No menu superior do VS Code, selecione **Terminal/Novo Terminal**.



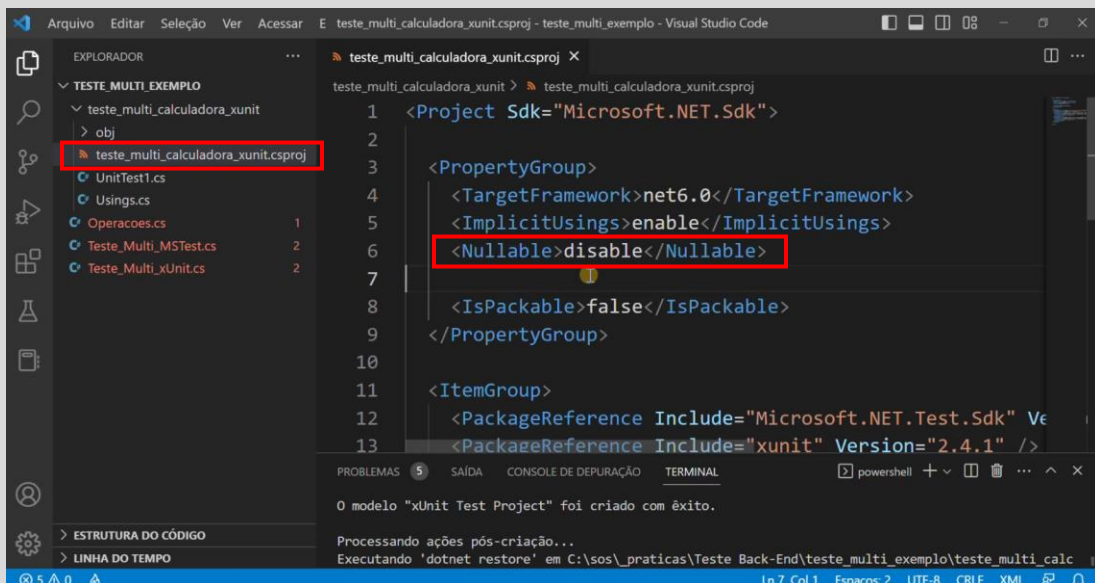
2. No terminal, digite **dotnet new xunit -n teste_multi_calculadora_xunit** e dê **Enter** para criar uma aplicação de teste.



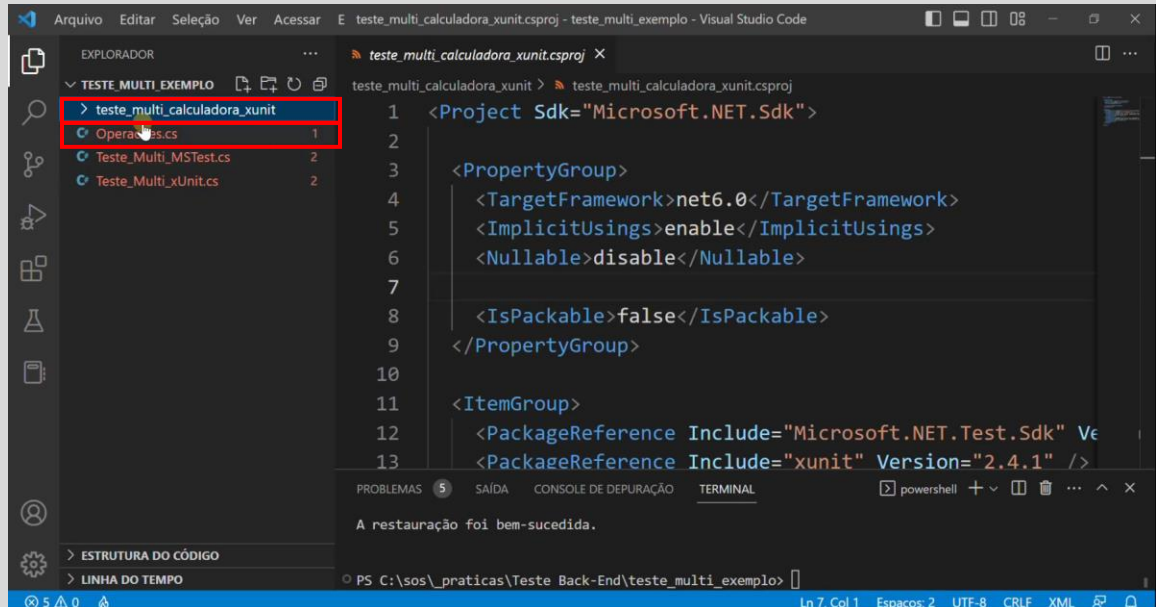
3. Observe que, após a execução bem-sucedida, o explorador de arquivos, na lateral esquerda, mostra a pasta teste_multi_calculadora_xunit criada com a estrutura principal pronta e as dependências necessárias.



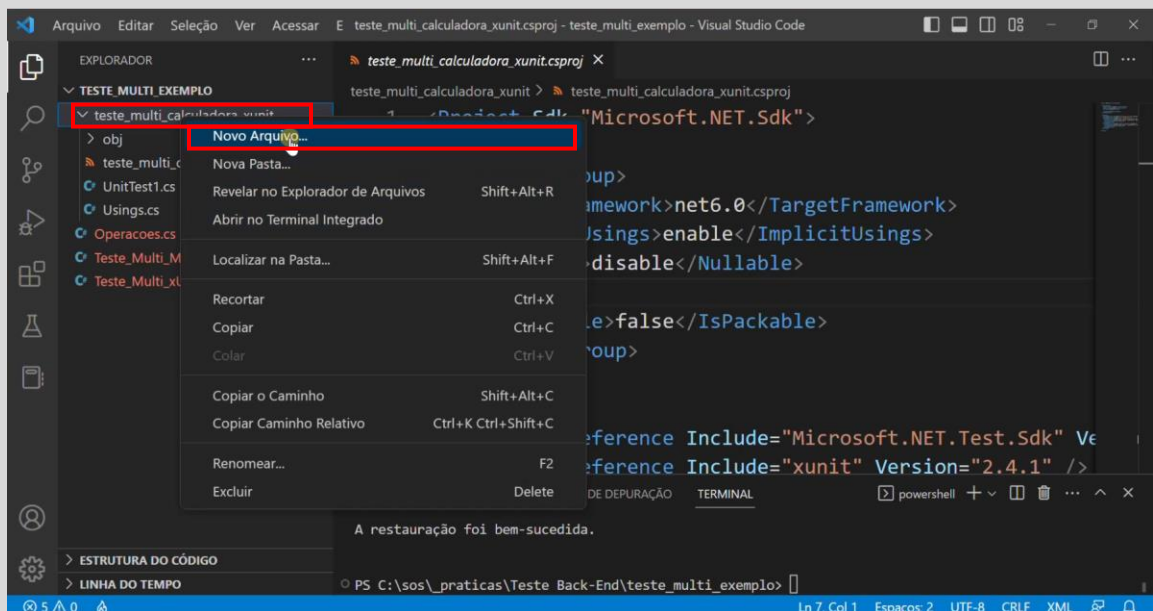
4. No arquivo teste_multi_calculadora_xunit.csproj, altere a linha `<Nullable>enable</Nullable>` para `<Nullable>disable</Nullable>`. Assim, todas as variáveis nulas não serão destacadas.



5. A próxima etapa é acrescentar a classe a ser testada à estrutura recém-criada. Note que a classe `Operacoes.cs` está fora da pasta `teste_multi_calculadora_xunit`.



6. Clique com o botão direito na pasta `teste_multi_calculadora_xunit` e selecione **Novo Arquivo**. Nomeie-o como `operacoes.cs`.



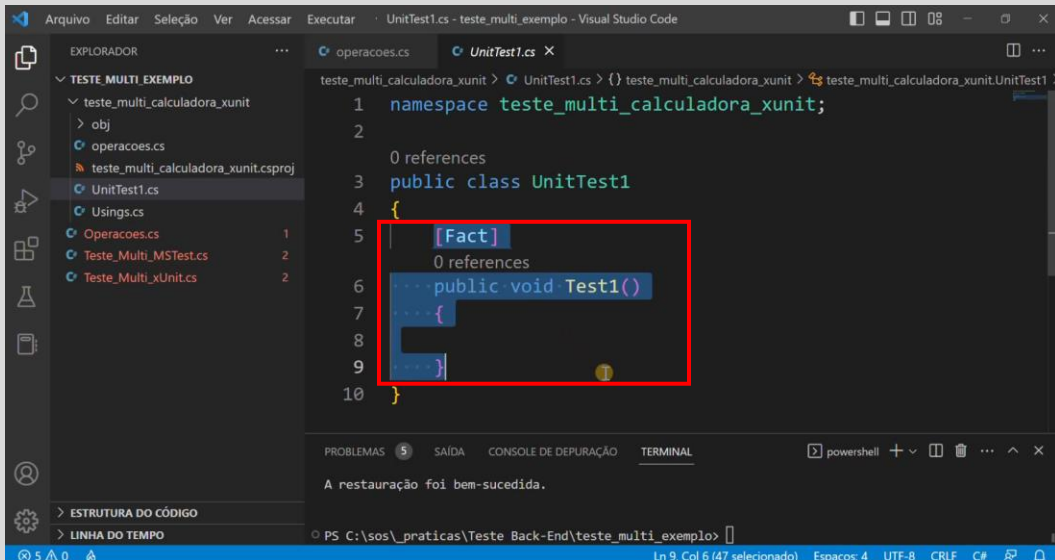
7. Selecione e copie todo o código de Operacoes.cs (arquivo fora da pasta) e cole em operacoes.cs (classe dentro da pasta teste_multi_calculadora_xunit). Salve as alterações.

```
6: segundoNumero)
7: {
8:     return (primeiroNumero + segundoNumero);
9: }
10:
11:
12: 0 references
13: public double Multiplicar(double primeiroNumero,
14:     double segundoNumero)
15: {
16:     return (primeiroNumero * segundoNumero);
17: }
```

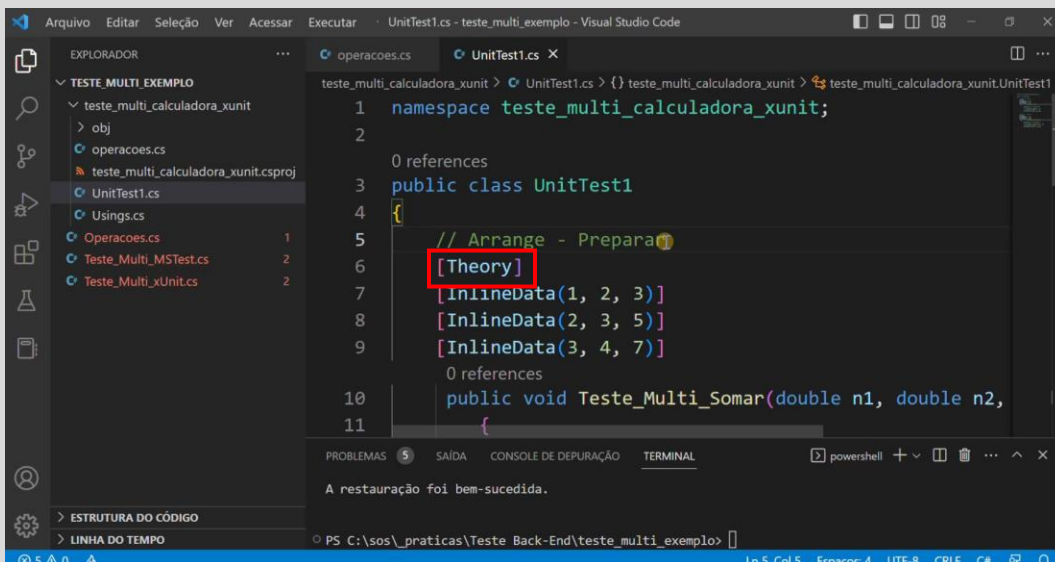
8. Em operações.cs, acrescente na primeira linha **namespace teste_multi_calculadora_xunit**. Lembre-se de abrir as chaves depois do namespace e de fechá-las no final, fazendo com que o namespace envolva todo o código. Salve as alterações.

```
1: namespace teste_multi_calculadora_xunit
2: {
3:     0 references
4:     public class Operacoes
5:     {
6:         0 references
7:         public double Somar(double primeiroNumero, double
8:             segundoNumero)
9:         {
10:             return (primeiroNumero + segundoNumero);
11:         }
12:         0 references
13:         public double Multiplicar(double primeiroNumero,
14:             double segundoNumero)
15:         {
16:             return (primeiroNumero * segundoNumero);
17:         }
18:     }
19: }
```

9. Selecione e copie todo o código de `Teste_Multi_xUnit.cs` (arquivo fora da pasta) e cole em `UnitTest1.cs` (classe dentro da pasta `teste_multi_calculadora_xunit`), substituindo o marcador **[Fact]** e o método **public void Test1()**, em destaque na imagem a seguir.



10. No xUnit, o marcador **[Theory]** do xUnit identifica um teste de múltiplas entradas.



11. Em um teste de múltiplas entradas, podemos notar a estrutura Arrange/Act/Assert. No bloco do Arrange estão as múltiplas entradas e o método **Teste_Multi_Somar()**, que precisa de três argumentos do tipo double (n1, n2 e o resultado esperado). Dentro de **Teste_Multi_Somar()** estão o Act e o Assert: no Act está a soma de dois números, e no assert está a comparação entre o resultado esperado e o resultado calculado.

```
// Arrange - Preparar
[Theory]
[InlineData(1, 2, 3)]
[InlineData(2, 3, 5)]
[InlineData(3, 4, 7)]
public void Teste_Multi_Somar(double n1, double n2, double res_esp)
{
    Operacoes o = new Operacoes();
    // Act - Agir
    var res_soma = o.Somar(n1, n2);
    // Assert - Verificar
    Assert.Equal(res_esp, res_soma);
}
```

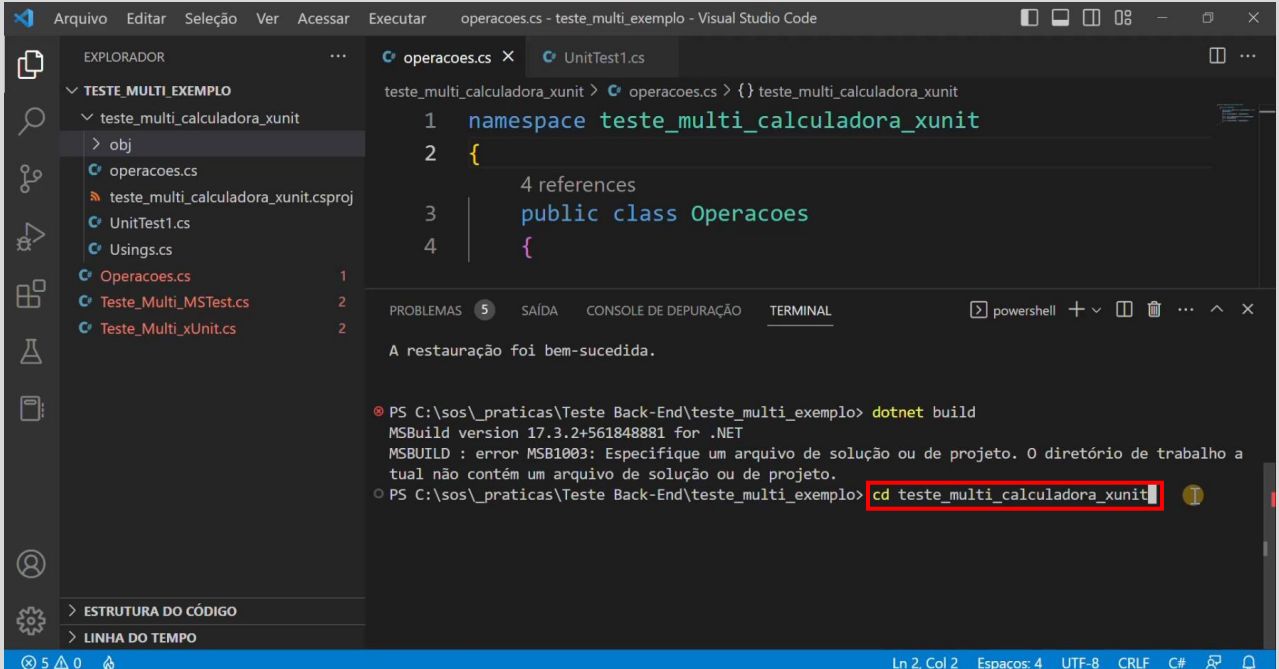
Assim como o marcador [Theory] marca um teste de múltiplas entradas, o marcador [InlineData] marca um conjunto de dados de entrada, ou seja, cada linha [InlineData] é um teste que será feito. No nosso exemplo, são três conjuntos de entrada de dados. Cada conjunto tem um primeiro número, um segundo número e o valor previsto. No primeiro teste, o método Teste_Multi_Somar() vai somar os valores 1 e 2, sendo 3 o valor previsto.

12. O método **Teste_Multi_Multiplicar()** tem a mesma estrutura do **Teste_Multi_Somar()**, ou seja, três argumentos do tipo double (n1, n2 e o resultado esperado). Dentro de **Teste_Multi_Multiplicar()** estão o Act e o Assert: no Act está a multiplicação dos dois números, e no assert está a comparação entre resultado esperado e o resultado calculado.

```
// Arrange - Preparar
[Theory]
[InlineData(1, 2, 2)]
[InlineData(2, 3, 6)]
[InlineData(3, 4, 12)]
public void Teste_Multi_Multiplicar(double n1, double n2, double res_esp)
{
    Operacoes o = new Operacoes();
    // Act - Agir
    var res_soma = o.Multiplicar(n1, n2);
    // Assert - Verificar
    Assert.Equal(res_esp, res_soma);
}
```

No nosso exemplo, são três conjuntos de entrada de dados. Cada um tem um primeiro número, um segundo número e o valor previsto. No primeiro teste, o método **Teste_Multi_Multiplicar()** vai multiplicar os valores 1 e 2, sendo 2 o valor previsto.

13. Agora, com os módulos completos, é hora de compilar o projeto. Antes disso, devemos entrar na pasta correta. Para isso, digite **cd teste_multi_calculadora_xunit** no terminal e dê **Enter**.



The screenshot shows the Visual Studio Code interface. On the left, the Explorer pane shows the project structure for 'TESTE_MULTI_EXEMPLO', with 'teste_multi_calculadora_xunit' selected. The main editor shows 'operacoes.cs' with the following code:

```
1 namespace teste_multi_calculadora_xunit
2 {
3     4 references
4     public class Operacoes
5     {
```

The bottom panel shows the TERMINAL with the following output:

```
A restauração foi bem-sucedida.

PS C:\sos_praticas\Teste Back-End\teste_multi_exemplo> dotnet build
MSBuild version 17.3.2+561848881 for .NET
MSBUILD : error MSB1003: Especifique um arquivo de solução ou de projeto. O diretório de trabalho a
tual não contém um arquivo de solução ou de projeto.
PS C:\sos_praticas\Teste Back-End\teste_multi_exemplo> cd teste_multi_calculadora_xunit
```

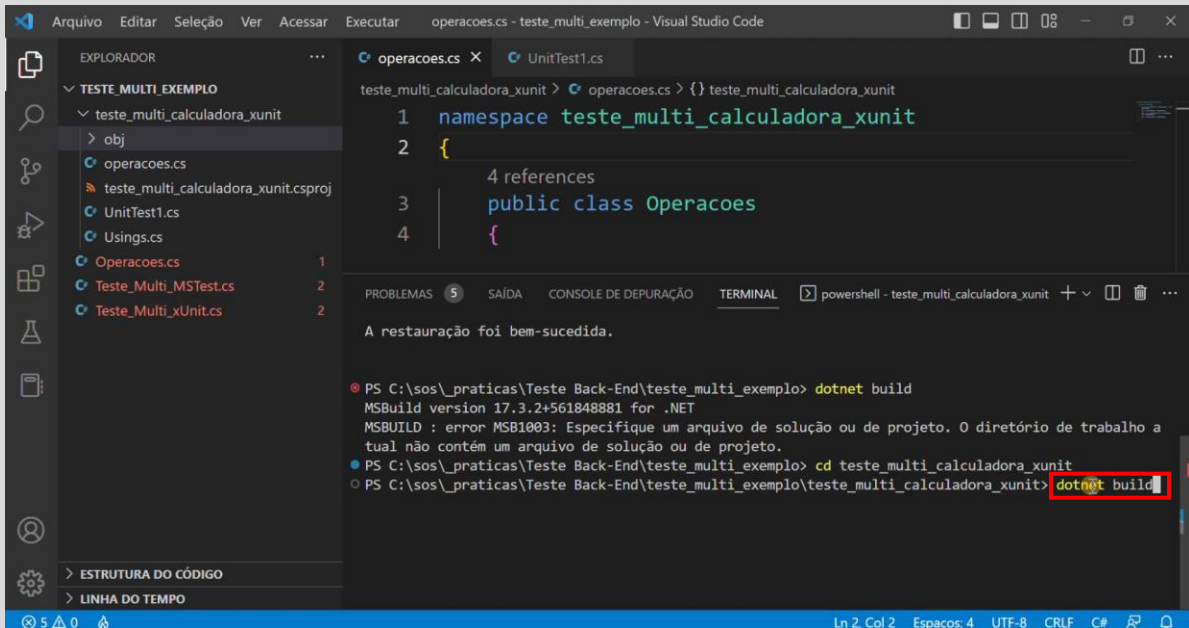
The command 'cd teste_multi_calculadora_xunit' is highlighted with a red box.

Importante

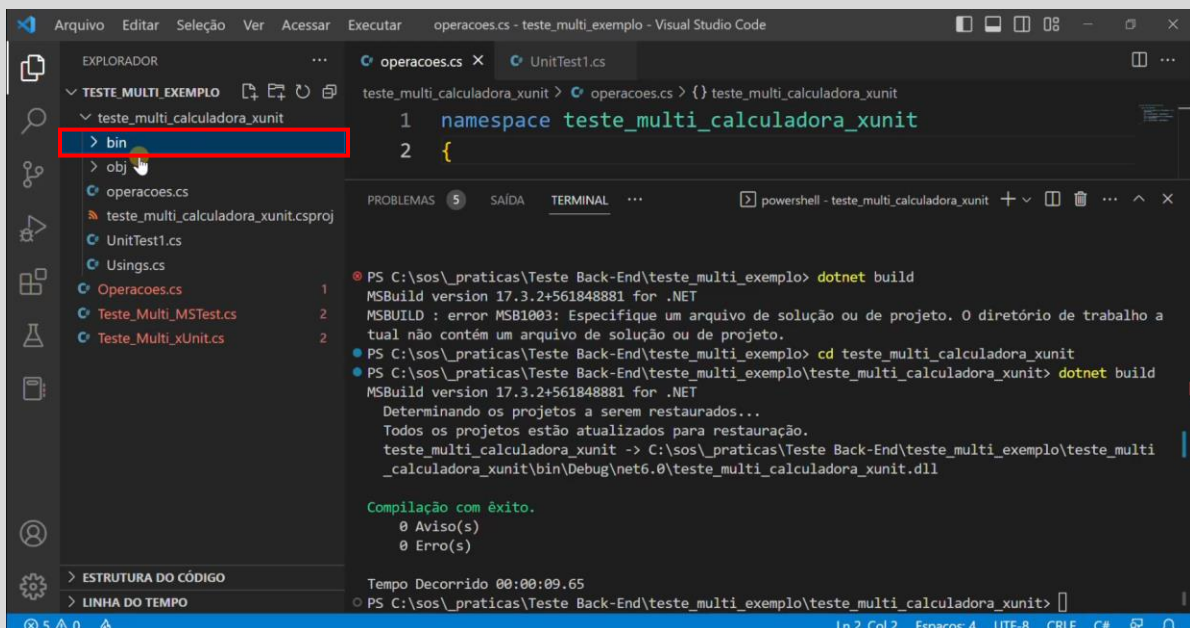
O nome da pasta do projeto deve ser o mesmo do namespace, e você deve estar dentro da pasta do projeto antes de seguir para o próximo passo.



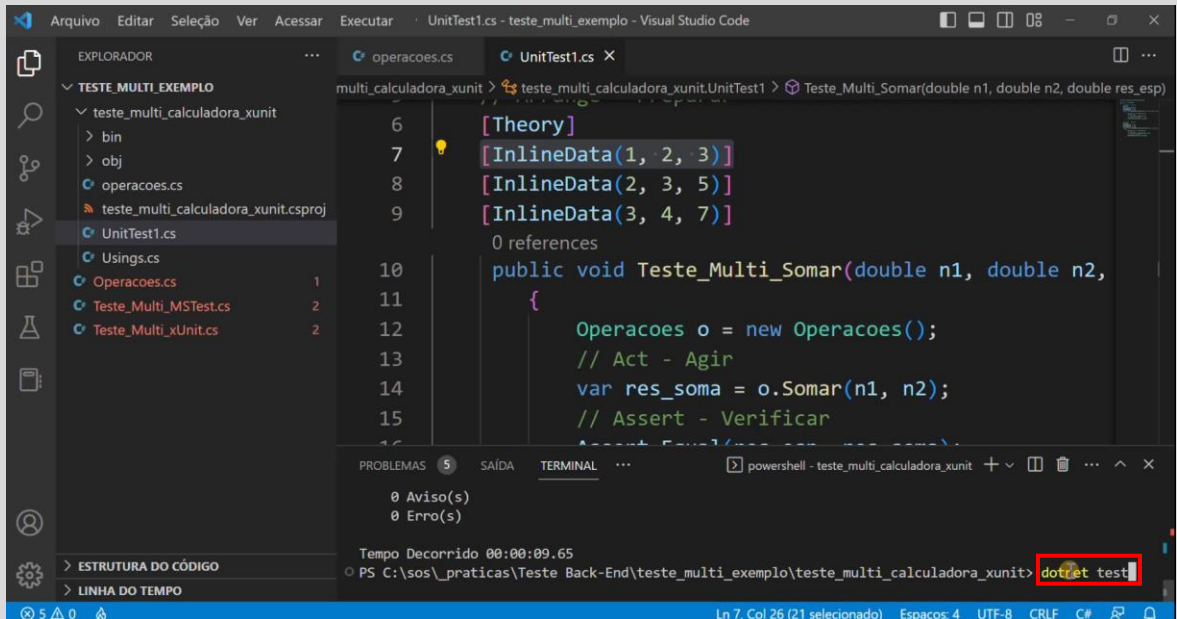
14. Dentro da pasta correta, digite no terminal **dotnet build** e dê **Enter**.



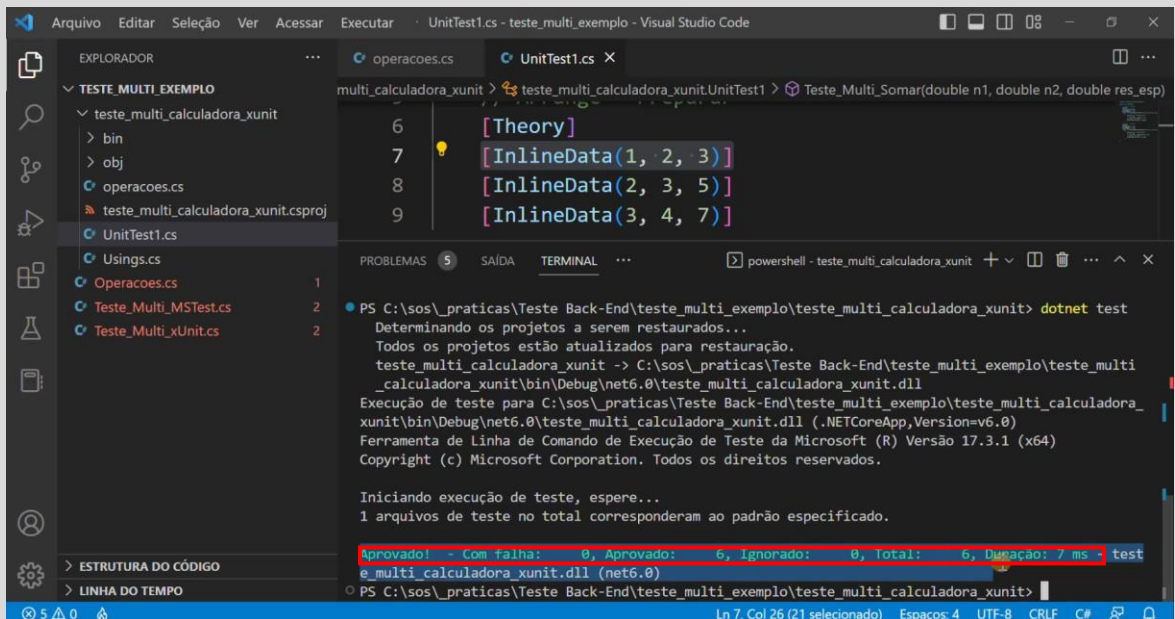
15. No explorador de arquivos, na lateral esquerda, aparecerá a pasta **bin** após a compilação bem-sucedida.



16. Para executar o teste propriamente dito, digite no terminal **dotnet test** e dê **Enter**.



17. Observe no terminal o resultado: 0 falha, 6 aprovados, 0 ignorado, total 2 e a duração do teste (7 ms).



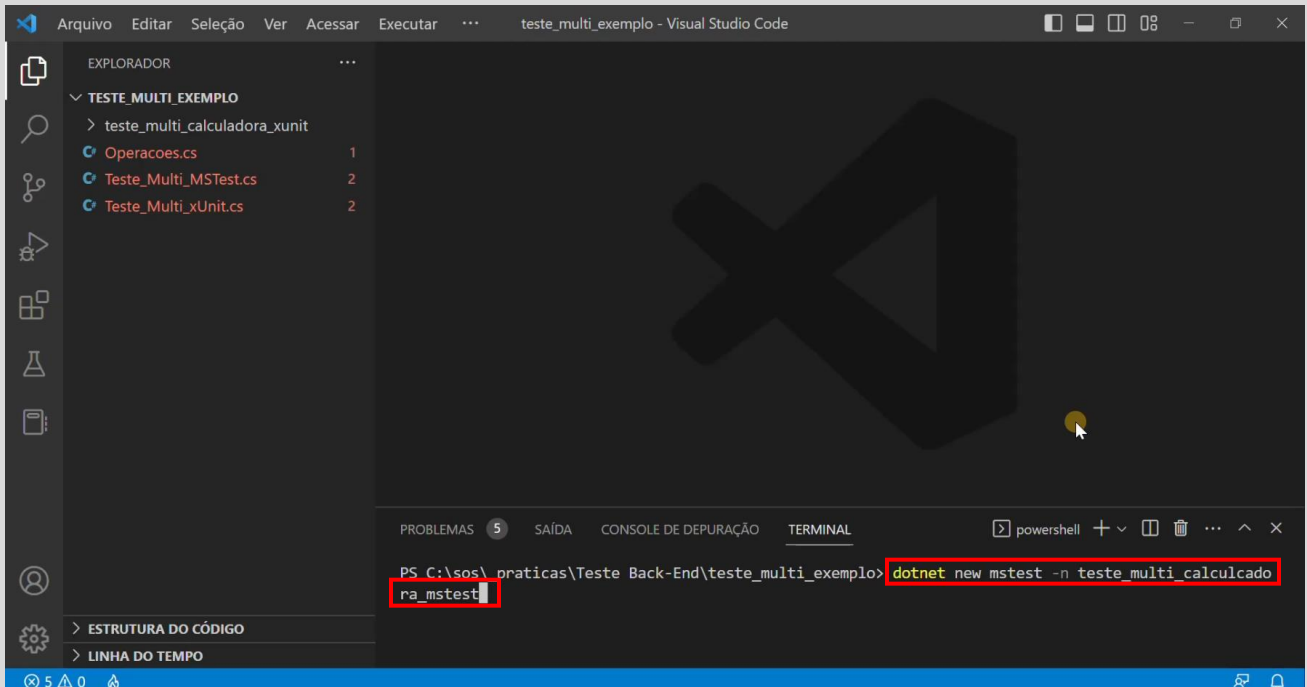
Dica!

Altere os valores de entrada, o resultado previsto ou, ainda, o operador aritmético dos métodos testados; salve as alterações e faça o teste novamente, simulando um erro. Analise os resultados.



Teste com MSTest

1. No terminal, digite **dotnet new mstest -n teste_multi_calculadora_mstest** e dê **Enter** para criar uma aplicação de teste.



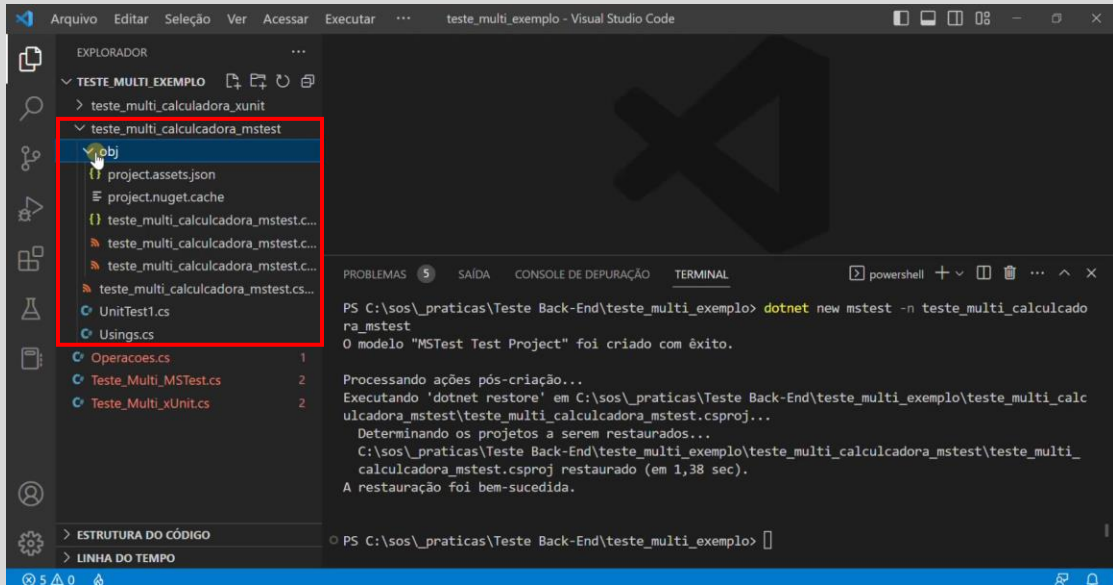
Importante

Houve um pequeno erro de digitação no vídeo: em vez de «calculadora» foi digitado «calculadora». No conteúdo escrito, usaremos a grafia correta e, portanto, diferente do vídeo.

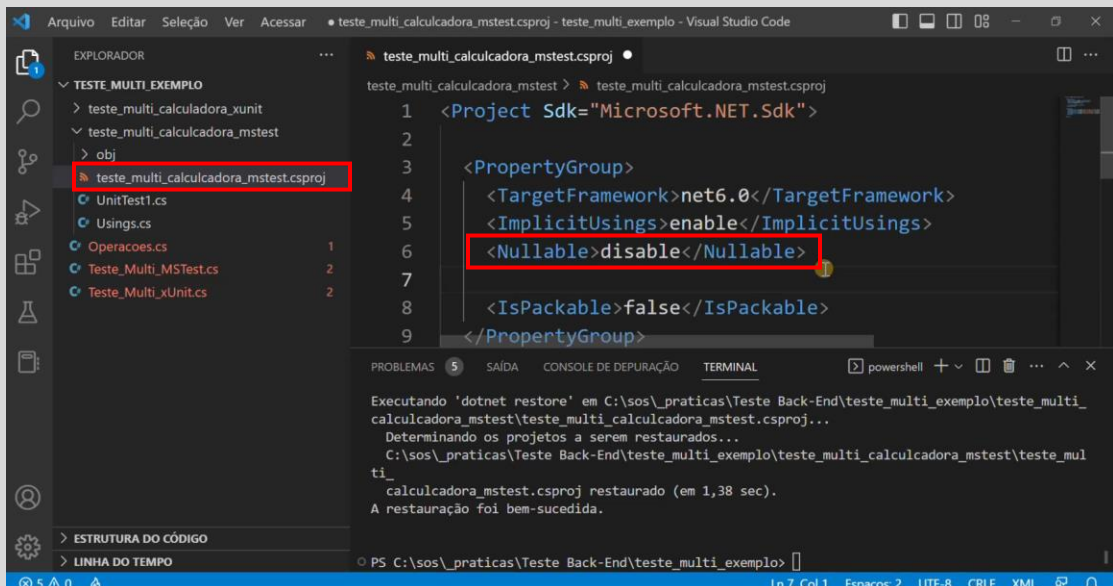


Certifique-se de usar o mesmo nome da pasta no namespace do seu projeto.

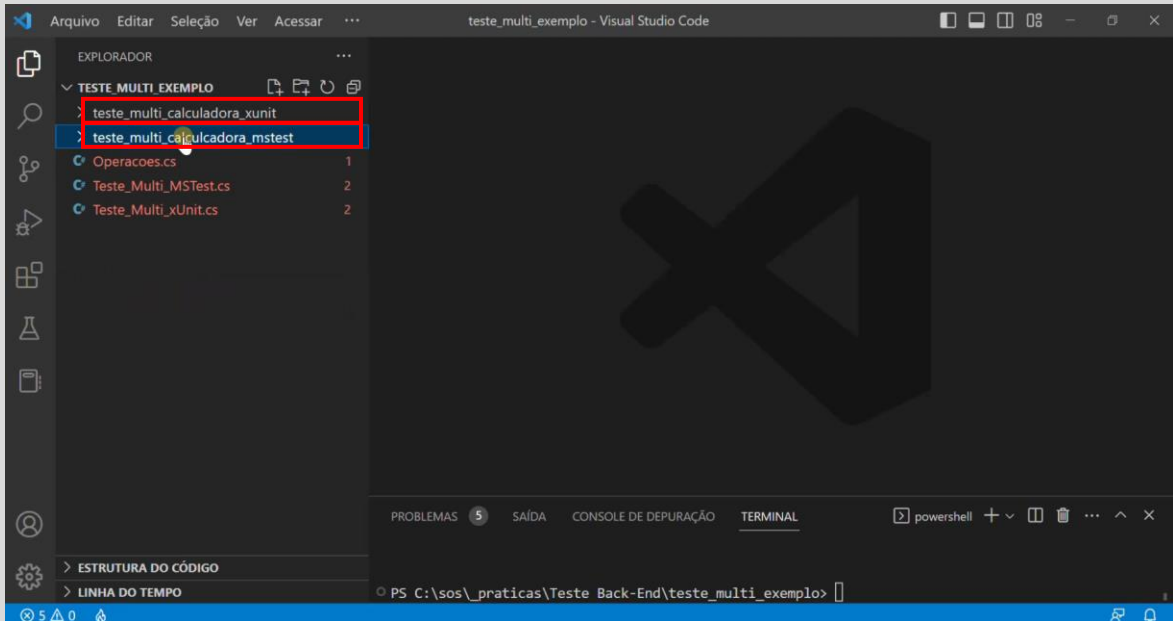
2. Observe que, após a execução bem-sucedida, o explorador de arquivos na lateral esquerda mostra a pasta **teste_multi_calculadora_mstest**, criada com a estrutura principal pronta e as dependências necessárias.



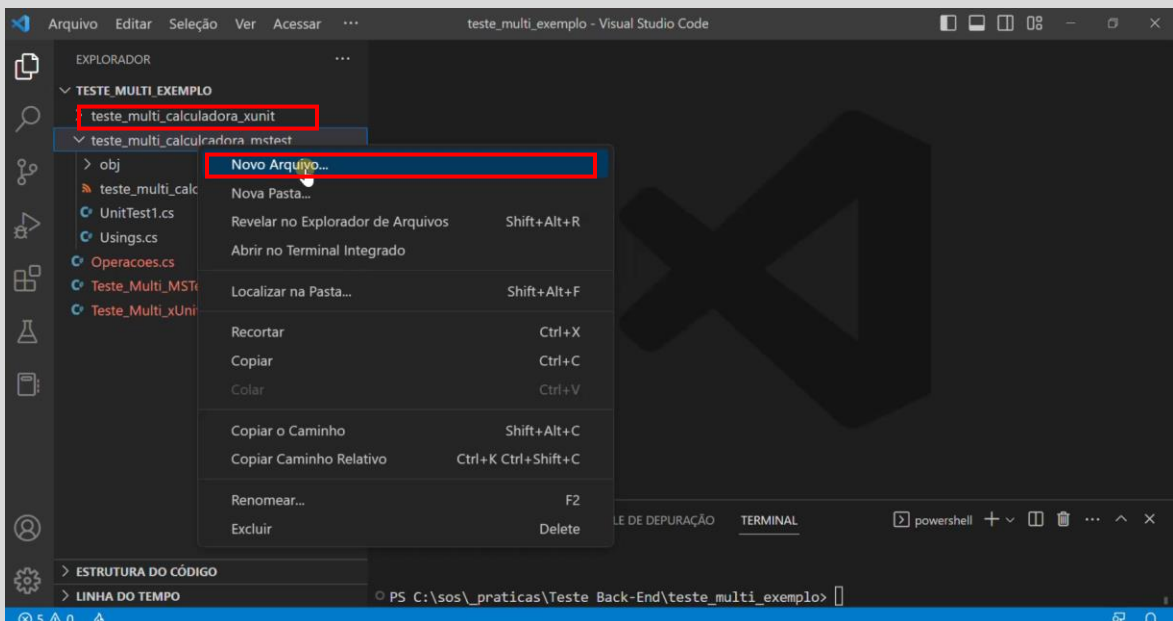
3. No arquivo teste_multi_calculadora_mstest.csproj, altere a linha `<Nullable>enable</Nullable>` para `<Nullable>disable</Nullable>`. Assim, todas as variáveis nulas não serão destacadas.



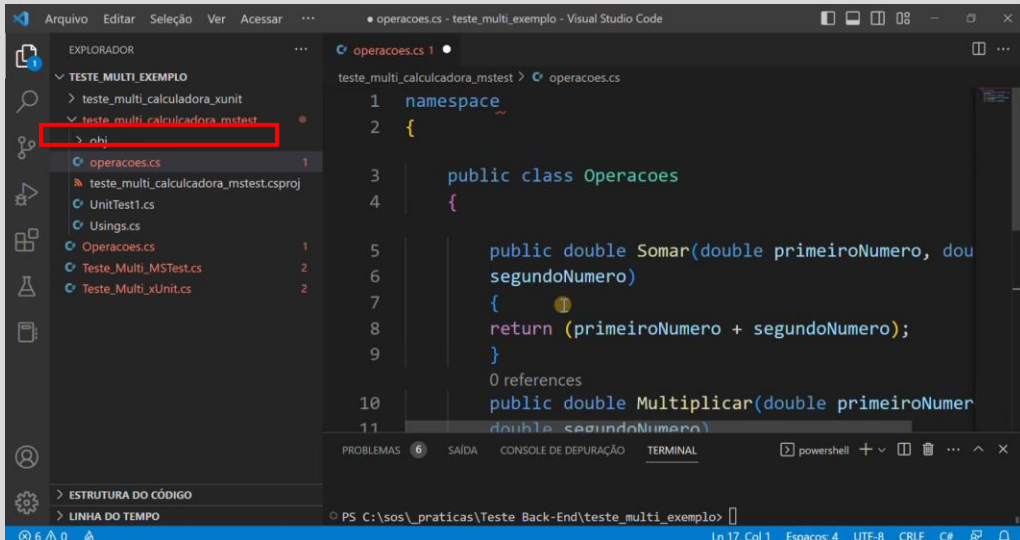
4. A próxima etapa é acrescentar a classe a ser testada à estrutura recém-criada. Note que a classe **Operacoes.cs** está fora da pasta **teste_multi_calculadora_mstest**.



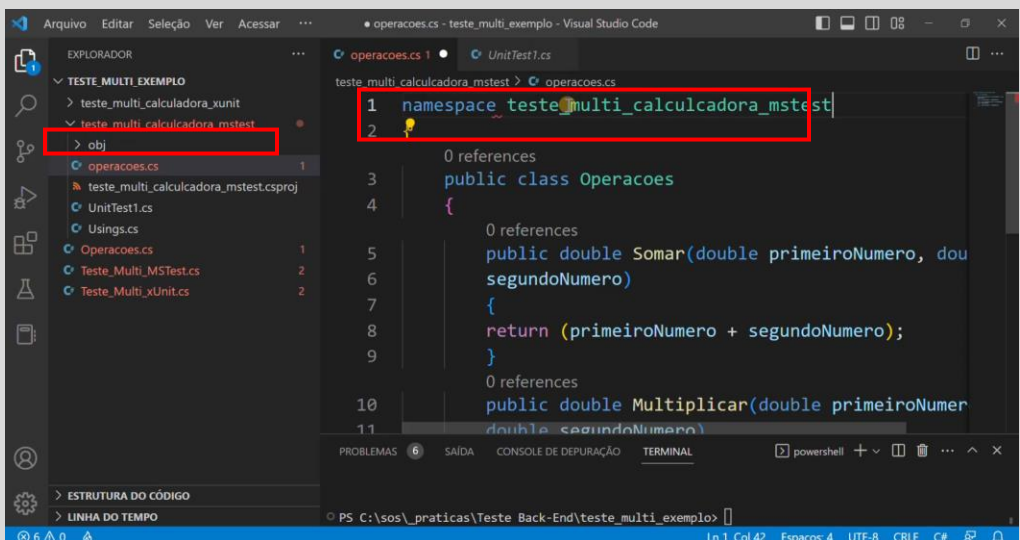
5. Clique com o botão direito na pasta **teste_multi_calculadora_mstest** e selecione **Novo Arquivo**. Nomeie-o como **operacoes.cs**.



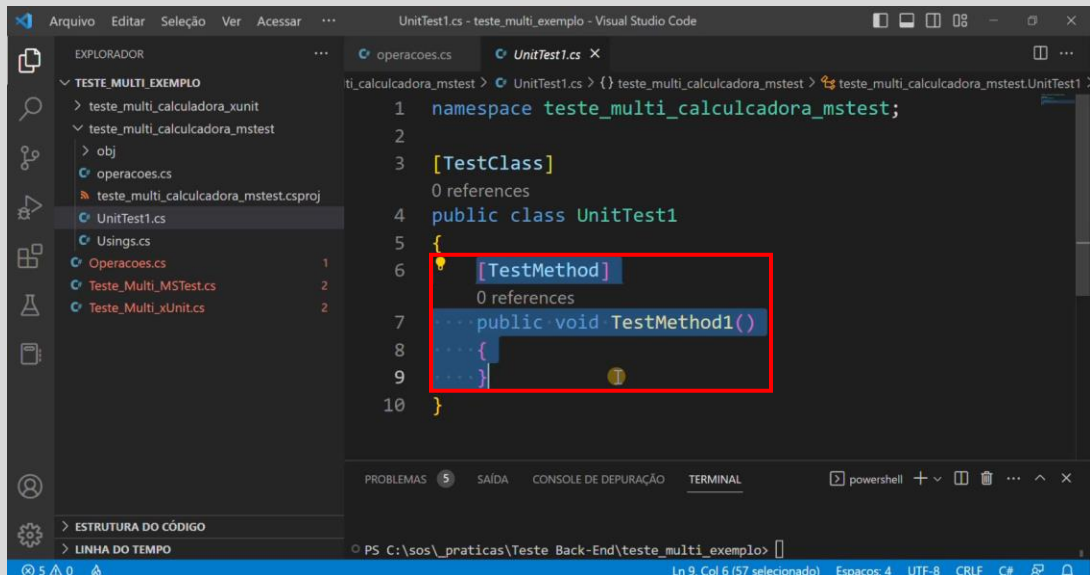
6. Selecione todo o código de **Operacoes.cs** (arquivo fora da pasta) e cole-o em **operacoes.cs** (classe dentro da pasta teste_multi_calculadora_mstest). Salve as alterações.



7. Em **operacoes.cs**, acrescente na primeira linha **namespace teste_multi_calculadora_mstest**. Lembre-se de abrir as chaves depois do namespace e de fechá-las no final, fazendo com que o namespace envolva todo o código. Salve as alterações.



8. Selecione todo o código de **Teste_Multi_MSTest.cs** (arquivo fora da pasta) e cole-o em **UnitTest1.cs** (classe dentro da pasta teste_multi_calculadora_mstest), substituindo o marcador **[TestMethod]** e o método **public void Test1()**, em destaque na imagem a seguir.

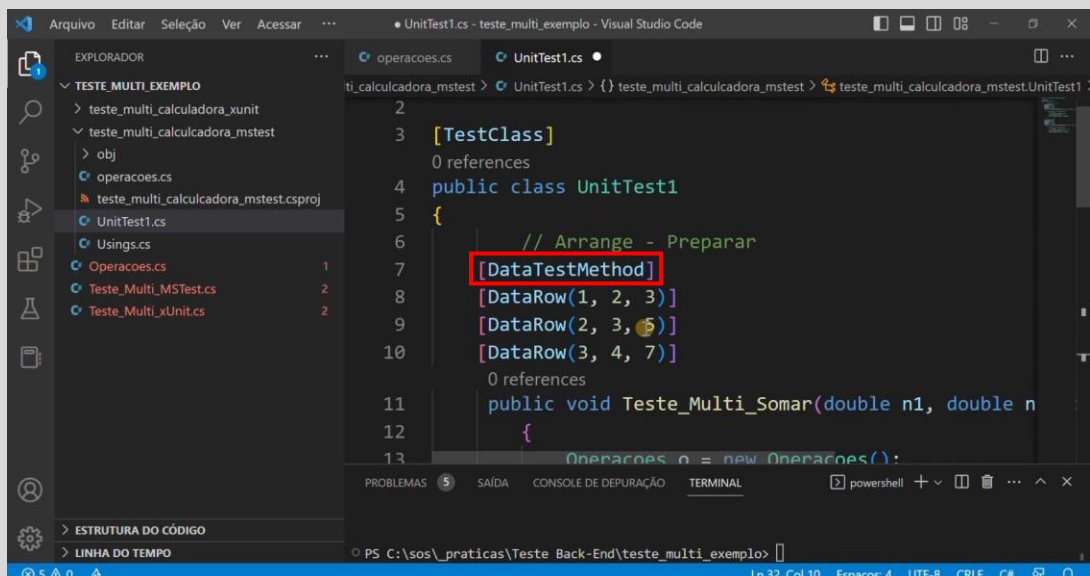


The screenshot shows the Visual Studio Code editor with the file explorer on the left and the code editor on the right. The file explorer shows the project structure with the file **Teste_Multi_MSTest.cs** selected. The code editor shows the **UnitTest1.cs** file with the following code:

```
1 namespace teste_multi_calculadora_mstest;
2
3 [TestClass]
4 0 references
5 public class UnitTest1
6 {
7     [TestMethod]
8     0 references
9     public void TestMethod1()
10 {
11 }
12 }
```

The **[TestMethod]** attribute and the **public void TestMethod1()** method signature are highlighted with a red box. The status bar at the bottom indicates the cursor is at line 9, column 6.

9. No MSTest, o marcador **[DataTestMethod]** do MSTest identifica um teste de múltiplas entradas.



The screenshot shows the Visual Studio Code editor with the file explorer on the left and the code editor on the right. The file explorer shows the project structure with the file **UnitTest1.cs** selected. The code editor shows the **UnitTest1.cs** file with the following code:

```
2
3 [TestClass]
4 0 references
5 public class UnitTest1
6 {
7     // Arrange - Preparar
8     [DataTestMethod]
9     [DataRow(1, 2, 3)]
10    [DataRow(2, 3, 5)]
11    [DataRow(3, 4, 7)]
12    0 references
13    public void Teste_Multi_Somar(double n1, double n2)
14    {
15        Operacoes o = new Operacoes();
16    }
17 }
```

The **[DataTestMethod]** attribute is highlighted with a red box. The status bar at the bottom indicates the cursor is at line 32, column 10.

10. Em um teste de múltiplas entradas, podemos notar a estrutura Arrange/Act/Assert. No bloco do Arrange estão as múltiplas entradas e o método **Teste_Multi_Somar()**, que precisa de três argumentos do tipo double (n1, n2 e o resultado esperado). Dentro de **Teste_Multi_Somar()** estão o Act e o Assert: no Act está a soma de dois números, e no assert está a comparação entre resultado esperado e o resultado calculado.

```
// Arrange - Preparar
[DataTestMethod]
[DataRow(1, 2, 3)]
[DataRow(2, 3, 5)]
[DataRow(3, 4, 7)]
public void Teste_Multi_Somar(double n1, double n2, double res_esp)
{
    Operacoes o = new Operacoes();
    // Act - Agir
    var res_soma = o.Somar(n1, n2);
    // Assert - Verificar
    Assert.AreEqual(res_esp, res_soma);
}
```

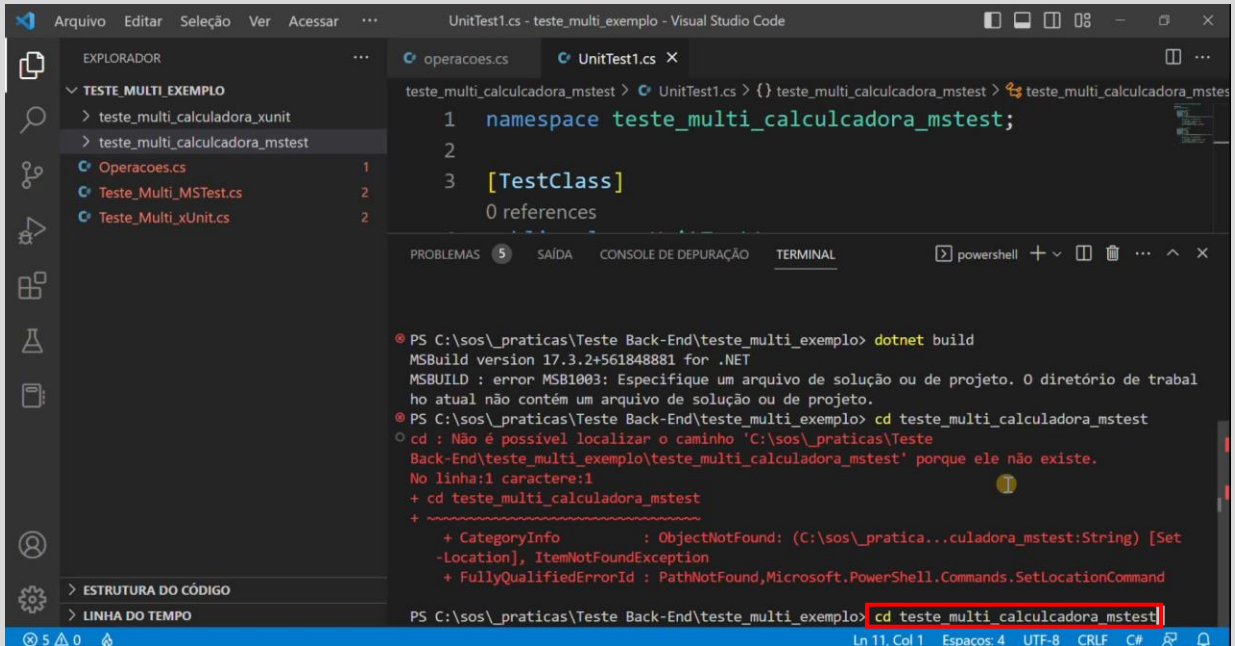
Assim como o marcador **[DataTestMethod]** marca um teste de múltiplas entradas, o marcador **[DataRow]** marca um conjunto de dados de entrada, ou seja, cada linha **[DataRow]** é um teste que será feito. No nosso exemplo, são três conjuntos de entrada de dados. Cada conjunto tem um primeiro número, um segundo número e o valor previsto. No primeiro teste, o método **Teste_Multi_Somar()** vai somar os valores 1 e 2, sendo 3 o valor previsto.

11. O método **Teste_Multi_Multiplicar()** tem a mesma estrutura do **Teste_Multi_Somar()**, ou seja, três argumentos do tipo double (n1, n2 e o resultado esperado). Dentro de **Teste_Multi_Multiplicar()** estão o Act e o Assert: no Act está a multiplicação dos dois números, e no assert está a comparação entre o resultado esperado e o resultado calculado.

```
// Arrange - Preparar
[DataTestMethod]
[DataRow(1, 2, 2)]
[DataRow(2, 3, 6)]
[DataRow(3, 4, 12)]
public void Teste_Multi_Multiplicar(double n1, double n2, double res_esp)
{
    Operacoes o = new Operacoes();
    // Act - Agir
    var res_soma = o.Multiplicar(n1, n2);
    // Assert - Verificar
    Assert.AreEqual(res_esp, res_soma);
}
```

No nosso exemplo, são três conjuntos de entrada de dados. Cada um deles tem um primeiro número, um segundo número e o valor previsto. No primeiro teste, o método **Teste_Multi_Multiplicar()** vai multiplicar os valores 1 e 2, sendo 2 o valor previsto.

12. Agora com os módulos completos, é hora de compilar o projeto. Antes disso, devemos entrar na pasta correta. Para isso, digite **cd teste_multi_calculadora_mstest** no terminal e dê **Enter**.



The screenshot shows the Visual Studio Code interface. The Explorer pane on the left shows a project named 'TESTE_MULTI_EXEMPLO' with a subfolder 'teste_multi_calculadora_mstest' containing files 'Operacoes.cs', 'Teste_Multi_MSTest.cs', and 'Teste_Multi_xUnit.cs'. The main editor shows 'UnitTest1.cs' with the following code:

```
1 namespace teste_multi_calculadora_mstest;
2
3 [TestClass]
0 references
```

The TERMINAL pane at the bottom shows the following commands and output:

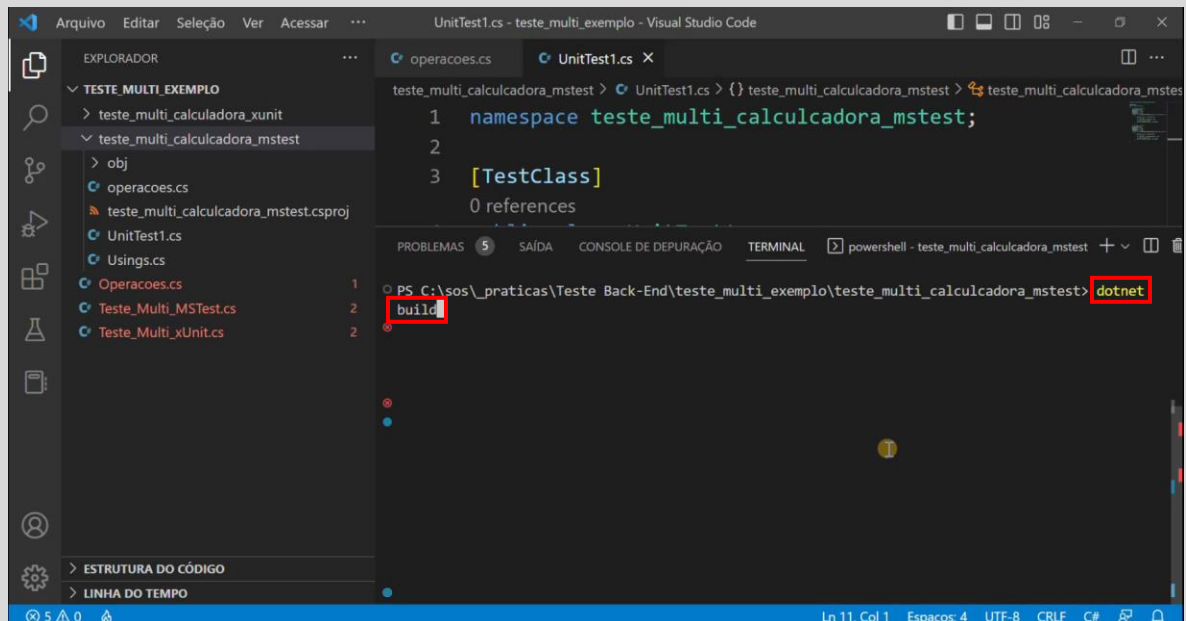
```
PS C:\sos\praticas\Teste Back-End\teste_multi_exemplo> dotnet build
MSBuild version 17.3.2+561848881 for .NET
MSBUILD : error MSB1003: Especifique um arquivo de solução ou de projeto. O diretório de trabalho atual não contém um arquivo de solução ou de projeto.
PS C:\sos\praticas\Teste Back-End\teste_multi_exemplo> cd teste_multi_calculadora_mstest
cd : Não é possível localizar o caminho 'C:\sos\praticas\Teste Back-End\teste_multi_exemplo\teste_multi_calculadora_mstest' porque ele não existe.
No linha:1 caractere:1
+ cd teste_multi_calculadora_mstest
+ ~~~~~
+ CategoryInfo          : ObjectNotFound: (C:\sos\pratica...culadora_mstest:String) [Set-Location], ItemNotFoundException
+ FullyQualifiedErrorId : PathNotFound,Microsoft.PowerShell.Commands.SetLocationCommand
PS C:\sos\praticas\Teste Back-End\teste_multi_exemplo> cd teste_multi_calculadora_mstest
```

Importante

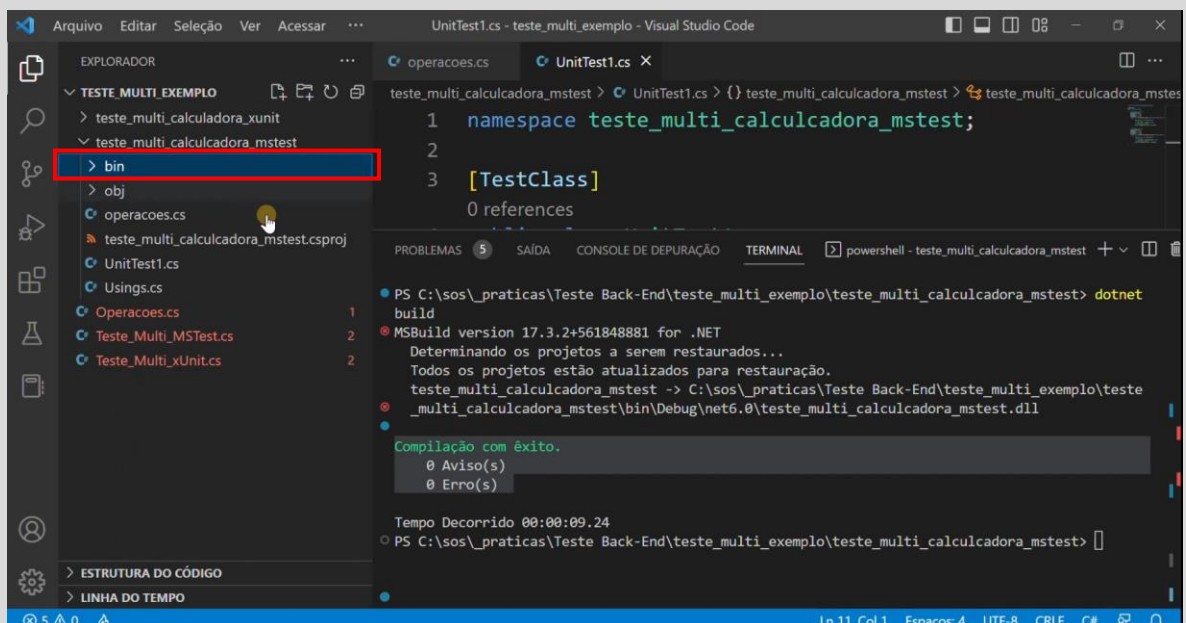
O nome da pasta do projeto deve ser o mesmo do namespace, e você deve estar dentro da pasta do projeto antes de seguir para o próximo passo.



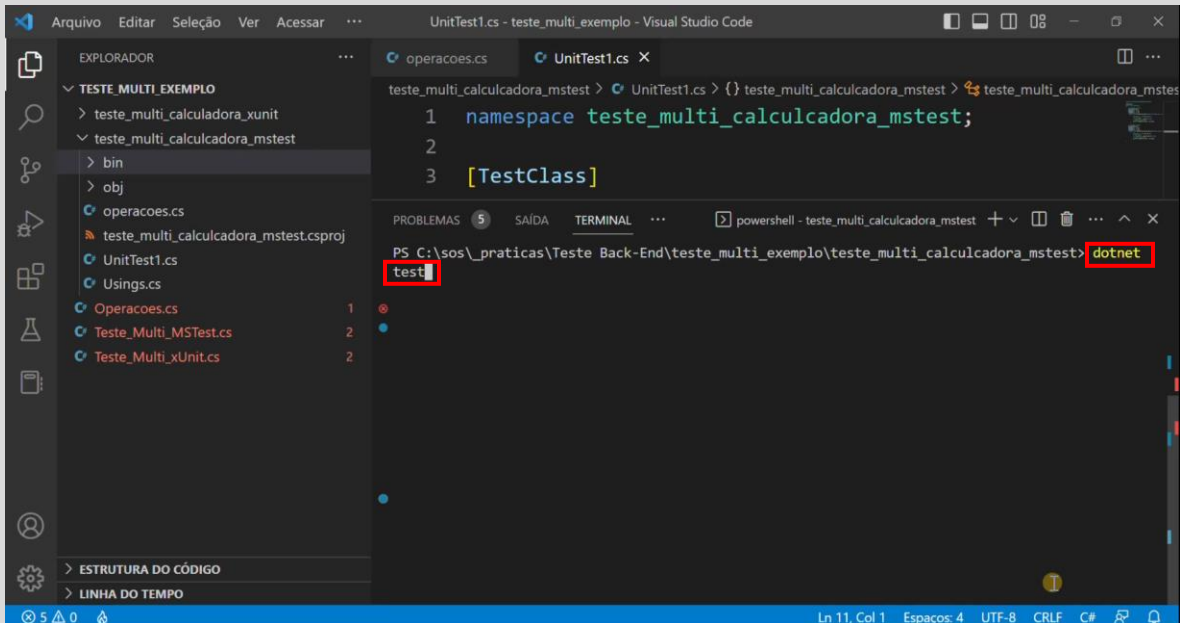
13. Agora dentro da pasta correta, digite **dotnet build** no terminal e dê **Enter**.



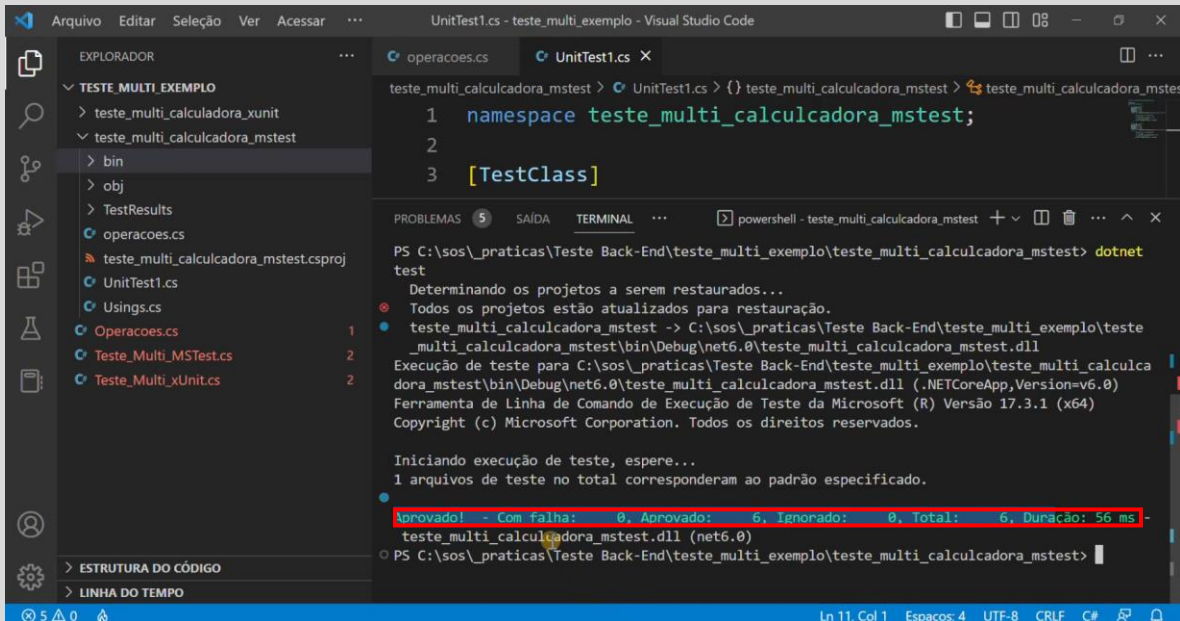
14. No explorador de arquivos, na lateral esquerda, aparecerá a pasta **bin** após a compilação bem-sucedida.



15. Para executar o teste propriamente dito, digite **dotnet test** no terminal e dê **Enter**.



16. Observe no terminal o resultado: 0 falhas, 6 aprovados, 0 ignorado, total 6 e a duração do teste (56 ms).



Dica!

Altere os valores de entrada, o resultado previsto ou, ainda, o operador aritmético dos métodos testados; salve as alterações e faça o teste novamente, simulando um erro. Analise os resultados.

