

# ► Testes unitários

Como construir testes unitários para aplicações dotnet core utilizando o xUnit.

# ✕Unit.net

xUnit.net is a free, open source, community-focused unit testing tool for the .NET Framework. Written by the original inventor of NUnit v2, xUnit.net is the latest technology for unit testing C#, F#, VB.NET and other .NET languages. xUnit.net works with ReSharper, CodeRush, TestDriven.NET and Xamarin. It is part of the .NET Foundation, and operates under their code of conduct. It is licensed under Apache 2 (an OSI approved license).

<https://xunit.net/>

# Classe a ser testada

```
public class PedidoService : IPedidoService
{
    private readonly IPedidosSqlAdapter pedidosSqlAdapter;
    private readonly IAtendenteSqlAdapter atendenteSqlAdapter;

    public PedidoService(IPedidosSqlAdapter pedidosSqlAdapter,
        IAtendenteSqlAdapter atendenteSqlAdapter)
    { ... }

    ...
}
```

# Classe a ser testada

...

```
public async Task<Pedido> CriarPedidoAsync(Guid atendenteId, int numeroMesa)
{
    var atendente = await atendenteSqlAdapter.GetAsync(atendenteId);

    if (atendente == null)
        throw new PedidosCoreException(PedidosCoreError.AtendenteNaoEncontrado);

    var pedido = new Pedido() { Atendente = atendente, Mesa = numeroMesa };
    pedido = await this.pedidosSqlAdapter.InsertAsync(pedido);

    return pedido;
}
```

<https://arquitetura.oleconsignado.com.br/testes-unitarios-com-xunit/>

# Construindo o teste

```
public class PedidoServiceTest
{
    private readonly IPedidoService pedidoService;

    private readonly Mock<IPedidosSqlAdapter> pedidoSqlAdapter;
    private readonly Mock<IAtendenteSqlAdapter> atendenteSqlAdapter;

    public PedidoServiceTest()
    {
        ...
    }
}
```

<https://arquitetura.oleconsignado.com.br/testes-unitarios-com-xunit/>

# Construindo o teste

```
public class PedidoServiceTest
{
    private readonly IPedidoService pedidoService;

    private readonly Mock<IPedidosSqlAdapter> pedidoSqlAdapter;
    private readonly Mock<IAtendenteSqlAdapter> atendenteSqlAdapter;

    public PedidoServiceTest()
    {
        ...
    }
}
```

# Construindo o teste

```
public class PedidoServiceTest
{
    private readonly IPedidoService pedidoService;

    private readonly Mock<IPedidosSqlAdapter> pedidoSqlAdapter;
    private readonly Mock<IAtendenteSqlAdapter> atendenteSqlAdapter;

    public PedidoServiceTest()
    {
        pedidoSqlAdapter = new Mock<IPedidosSqlAdapter>();

        pedidoSqlAdapter
            .Setup(p => p.InsertAsync(It.IsAny<Pedido>()))
            .ReturnsAsync((Pedido p) => p);
        pedidoSqlAdapter
            .Setup(p => p.UpdateAsync(It.IsAny<Pedido>()))
            .ReturnsAsync((Pedido p) => p);
    }
}
```

<https://arquitetura.oleconsignado.com.br/testes-unitarios-com-xunit/>

# Construindo o teste

```
public PedidoServiceTest()
{
    pedidoSqlAdapter = new Mock<IPedidosSqlAdapter>();

    pedidoSqlAdapter
        .Setup(p => p.InsertAsync(It.IsAny<Pedido>()))
        .ReturnsAsync((Pedido p) => p);
    pedidoSqlAdapter
        .Setup(p => p.UpdateAsync(It.IsAny<Pedido>()))
        .ReturnsAsync((Pedido p) => p);

    atendenteSqlAdapter = new Mock<IAtendenteSqlAdapter>();

    atendenteSqlAdapter
        .Setup(p => p.GetAsync(It.IsAny<Guid>()))
        .ReturnsAsync(new Atendente() { Nome = "...", Cpf = "..." });
}
```

<https://arquitetura.oleconsignado.com.br/testes-unitarios-com-xunit/>




# Construindo o teste

```
...  
private readonly IPedidoService pedidoService;  
  
public PedidoServiceTest()  
{  
    ...  
    atendenteSqlAdapter = new Mock<IAtendenteSqlAdapter>();  
  
    atendenteSqlAdapter  
        .Setup(p => p.GetAsync(It.IsAny<Guid>()))  
        .ReturnsAsync(new Atendente() { Nome = "...", Cpf = "..."});
```

```
pedidoService = new PedidoService(  
    pedidoSqlAdapter.Object,  
    atendenteSqlAdapter.Object);
```

# Construindo o teste

```
...  
private readonly IPedidoService pedidoService;  
  
public PedidoServiceTest()  
{  
    ...  
    atendenteSqlAdapter = new Mock<IAtendenteSqlAdapter>();  
  
    atendenteSqlAdapter  
        .Setup(p => p.GetAsync(It.IsAny<Guid>()))  
        .ReturnsAsync(new Atendente() { Nome = "...", Cpf = "..."});  
  
    pedidoService = new PedidoService(  
        pedidoSqlAdapter.Object,  
        atendenteSqlAdapter.Object);  
}
```



Dependências do  
PedidoService

<https://arquitetura.oleconsignado.com.br/testes-unitarios-com-xunit/>

# Construindo o teste

*Caminho feliz (caso de sucesso)*

```
[Fact]
[Trait("CriarPedidoAsync", "Sucesso")]
public async Task CriarPedidoAsync_Sucesso()
{
    var pedido = await pedidoService
                  .CriarPedidoAsync(Guid.NewGuid(), 1);

    Assert.NotNull(pedido);
}
```

<https://arquitetura.oleconsignado.com.br/testes-unitarios-com-xunit/>

# Construindo o teste

*Caminho alternativo (caso de erro)*

```
[Fact]
[Trait("CriarPedidoAsync", "Erro")]
public async Task CriarPedidoAsync_AtendenteInvalido()
{
    atendenteSqlAdapter
        .Setup(a => a.GetAsync(It.IsAny<Guid>()))
        .ReturnsAsync((Atendente)null);

    var ex = await Assert.ThrowsAsync<PedidosCoreException>(
        () => pedidoService.CriarPedidoAsync(Guid.NewGuid(), 1));

    Assert.Contains(PedidosCoreError.AtendenteNaoEncontrado, ex.Errors);
}
```

<https://arquitetura.oleconsignado.com.br/testes-unitarios-com-xunit/>

# Problema

- ▶ O resultado de um teste não pode influenciar em outro teste, por conta disso, recursos não devem ser compartilhados.

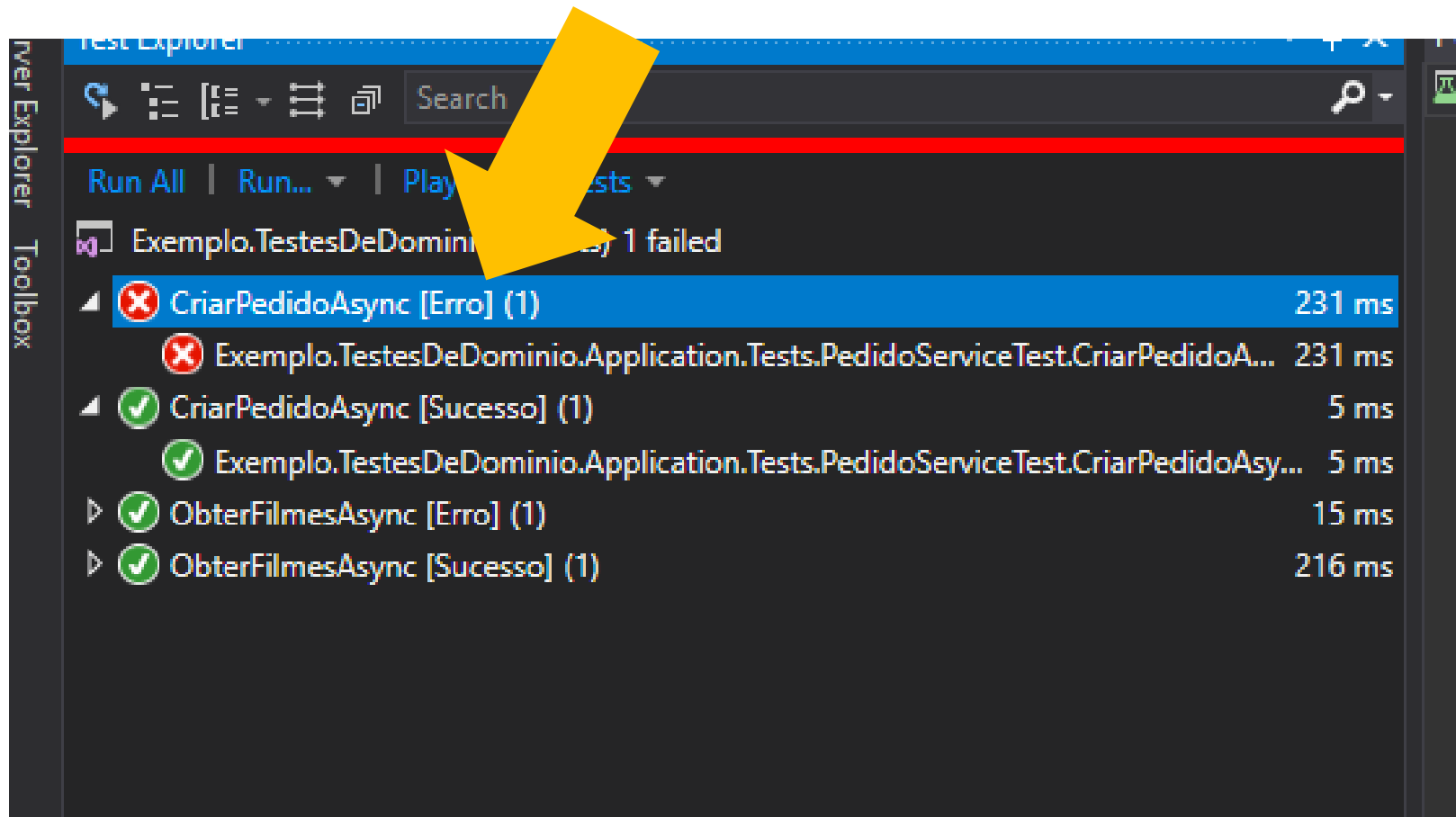
# Solução

- ▶ O xUnit tem como recurso **recrutar** a classe de teste a cada teste executado.
- ▶ Ou seja, dado uma classe X, que possui os testes Xy e Xz, ao executar o teste Xy, o construtor da classe X será executado, e ao executar o teste Xz, o construtor da classe X também será executado. Dessa forma, garantindo que os recursos serão recriados a cada iteração, e não haverá compartilhamento de recursos.

*Bruno Roldão*

<https://arquitetura.oleconsignado.com.br/testes-unitarios-com-xunit/>

# Para que servem os Traits?



# Theory

```
[Theory]
[InlineData(1)]
[InlineData(2)]
[InlineData(3)]
[Trait("CriarPedidoAsync", "Sucesso")]
public async Task CriarPedidoAsync_Sucesso(int numeroMesa)
{
    var pedido = await pedidoService
        .CriarPedidoAsync(Guid.NewGuid(), numeroMesa);

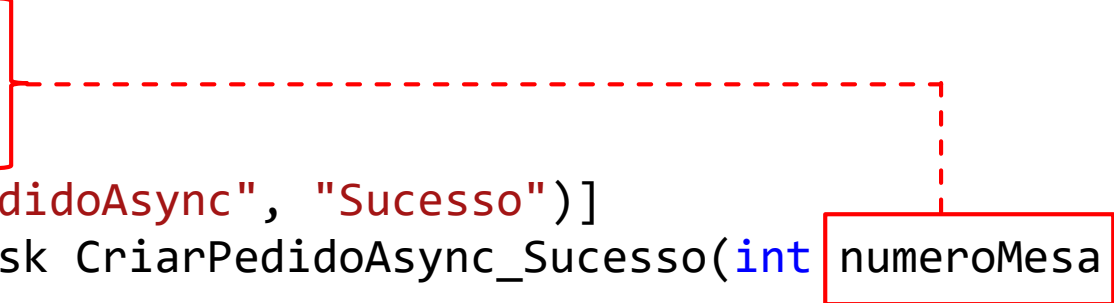
    Assert.NotNull(pedido);
}
```



# Theory

```
[Theory]
[InlineData(1)]
[InlineData(2)]
[InlineData(3)]
[Trait("CriarPedidoAsync", "Sucesso")]
public async Task CriarPedidoAsync_Sucesso(int numeroMesa)
{
    var pedido = await pedidoService
        .CriarPedidoAsync(Guid.NewGuid(), numeroMesa);

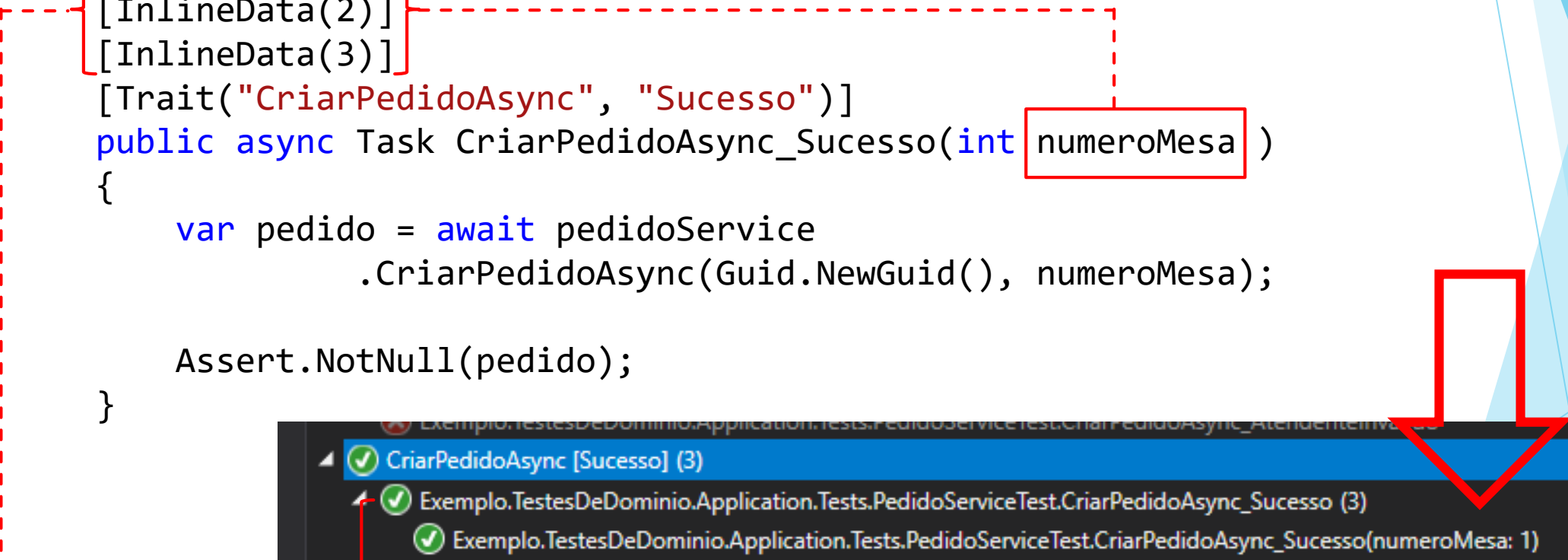
    Assert.NotNull(pedido);
}
```



# Theory

```
[Theory]
[InlineData(1)]
[InlineData(2)]
[InlineData(3)]
[Trait("CriarPedidoAsync", "Sucesso")]
public async Task CriarPedidoAsync_Sucesso(int numeroMesa )
{
    var pedido = await pedidoService
        .CriarPedidoAsync(Guid.NewGuid(), numeroMesa);

    Assert.NotNull(pedido);
}
```



Exemplo.TestesDeDominio.Application.Tests.PedidoServiceTest.CriarPedidoAsync_AtendimentoInv...	231 ms
✓ CriarPedidoAsync [Sucesso] (3)	232 ms
✓ Exemplo.TestesDeDominio.Application.Tests.PedidoServiceTest.CriarPedidoAsync_Sucesso (3)	232 ms
✓ Exemplo.TestesDeDominio.Application.Tests.PedidoServiceTest.CriarPedidoAsync_Sucesso(numeroMesa: 1)	1 ms
✓ Exemplo.TestesDeDominio.Application.Tests.PedidoServiceTest.CriarPedidoAsync_Sucesso(numeroMesa: 2)	230 ms
✓ Exemplo.TestesDeDominio.Application.Tests.PedidoServiceTest.CriarPedidoAsync_Sucesso(numeroMesa: 3)	1 ms

# Theory (origem dos dados)

- ▶ MemberData

<https://github.com/OleConsignado/otc-validations-br/blob/master/Source/Otc.Validations.Br.Tests/CpfTest.cs>

- ▶ ClassData

```
[Theory]  
[ClassData(typeof(CriarPedidoData))]  
[Trait("CriarPedidoAsync", "Sucesso")]  
public async Task CriarPedidoAsync_Sucesso(int numeroMesa)
```

# Interceptando métodos de Mocks

[Theory]

...

```
public async Task CriarPedidoAsync_Sucesso(int numeroMesa)
{
    pedidoSqlAdapter
        .Setup(p => p.InsertAsync(It.IsAny<Pedido>()))
        .Callback<Pedido>(p =>
        {
            Assert.Equal(expected: numeroMesa, actual: p.Mesa);
        })
        .ReturnsAsync(() => new Pedido() { Mesa = numeroMesa });

    var pedido = await pedidoService.CriarPedidoAsync(Guid.NewGuid(), numeroMesa);

    Assert.NotNull(pedido);
}
```

# Boas práticas

- ▶ Validar o resultado (por meio de Assert)

- ▶ `void Equal<T>(T expected, T actual);`
- ▶ `void Contains<T>(T expected, IEnumerable<T> collection);`
- ▶ `void True(bool condition); / void False(bool condition);`
- ▶ `void Null(object @object); / void NotNull(object @object);`
- ▶ `T Throws<T>(Func<object> testCode) where T : Exception;`
- ▶ `Task<T> ThrowsAsync<T>(Func<Task> testCode)`

Métodos mais  
comuns

- ▶ Evitar: `Assert.True(valorEsperado == valorProduzido);`

# Avaliando Cobertura de Testes

The screenshot displays the Visual Studio IDE with a C# code file and a code coverage results window.

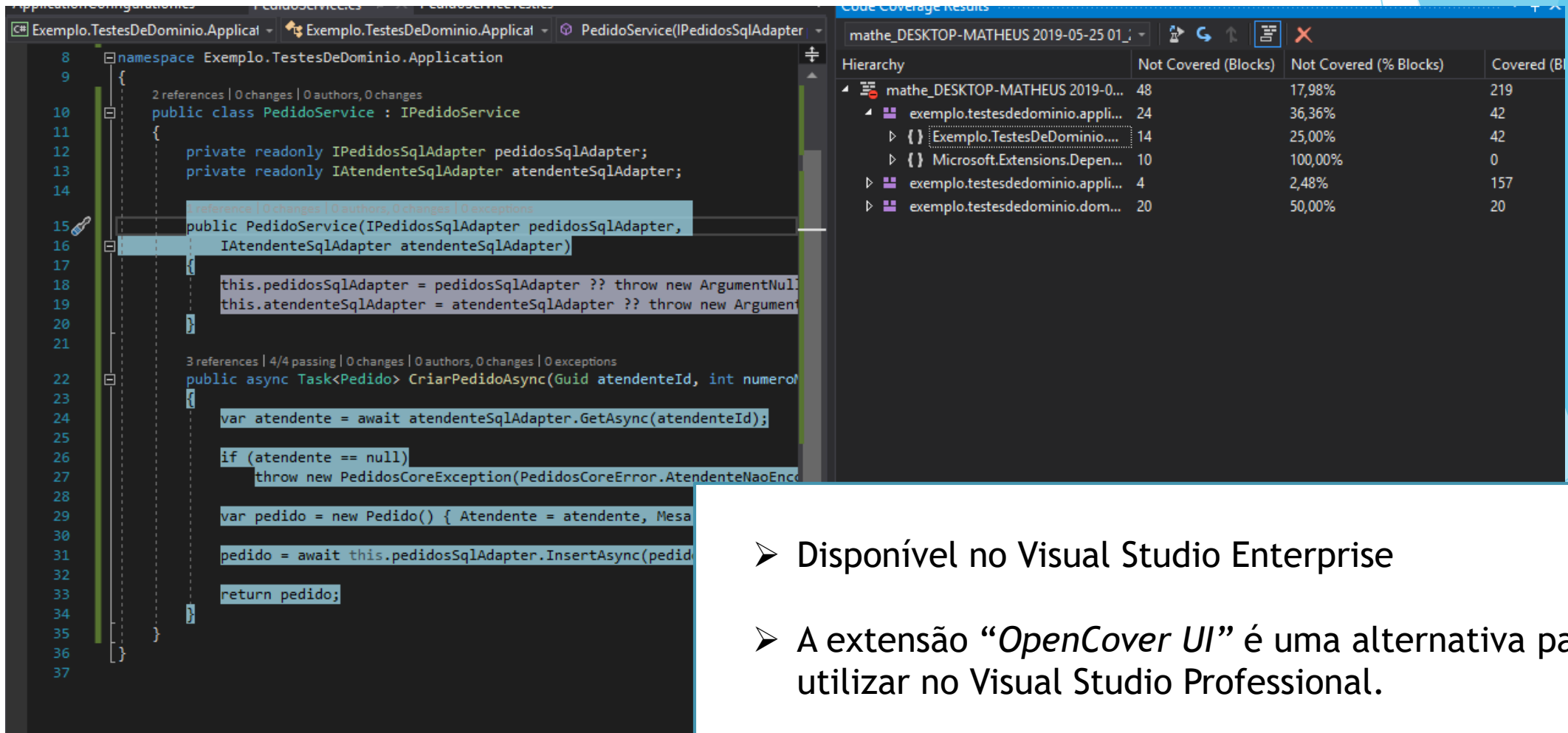
**Code File:** `Exemplo.TestesDeDominio.Application`

```
8 namespace Exemplo.TestesDeDominio.Application
9 {
10     2 references | 0 changes | 0 authors, 0 changes
11     public class PedidoService : IPedidoService
12     {
13         private readonly IPedidosSqlAdapter pedidosSqlAdapter;
14         private readonly IAtendenteSqlAdapter atendenteSqlAdapter;
15
16         reference | 0 changes | 0 authors, 0 changes | 0 exceptions
17         public PedidoService(IPedidosSqlAdapter pedidosSqlAdapter,
18                             IAtendenteSqlAdapter atendenteSqlAdapter)
19         {
20             this.pedidosSqlAdapter = pedidosSqlAdapter ?? throw new ArgumentNull
21             this.atendenteSqlAdapter = atendenteSqlAdapter ?? throw new Argument
22         }
23
24         3 references | 4/4 passing | 0 changes | 0 authors, 0 changes | 0 exceptions
25         public async Task<Pedido> CriarPedidoAsync(Guid atendenteId, int numeroM
26         {
27             var atendente = await atendenteSqlAdapter.GetAsync(atendenteId);
28
29             if (atendente == null)
30                 throw new PedidosCoreException(PedidosCoreError.AtendenteNaoEnco
31
32             var pedido = new Pedido() { Atendente = atendente, Mesa = numeroMesa
33
34             pedido = await this.pedidosSqlAdapter.InsertAsync(pedido);
35
36             return pedido;
37         }
38     }
39 }
```

**Code Coverage Results:**

Hierarchy	Not Covered (Blocks)	Not Covered (% Blocks)	Covered (B...
mathe_DESKTOP-MATHEUS 2019-05-25 01_...	48	17,98%	219
└─ exemplo.testesdedominio.appli...	24	36,36%	42
└─ {} Exemplo.TestesDeDominio...	14	25,00%	42
└─ {} Microsoft.Extensions.Depen...	10	100,00%	0
└─ exemplo.testesdedominio.appli...	4	2,48%	157
└─ exemplo.testesdedominio.dom...	20	50,00%	20

# Avaliando Cobertura de Testes



The screenshot displays the Visual Studio Enterprise interface with a C# code file and a code coverage results window.

**Code File:** `Exemplo.TestesDeDominio.Application`

```
8 namespace Exemplo.TestesDeDominio.Application
9 {
10     2 references | 0 changes | 0 authors, 0 changes
11     public class PedidoService : IPedidoService
12     {
13         private readonly IPedidosSqlAdapter pedidosSqlAdapter;
14         private readonly IAtendenteSqlAdapter atendenteSqlAdapter;
15
16         reference | 0 changes | 0 authors, 0 changes | 0 exceptions
17         public PedidoService(IPedidosSqlAdapter pedidosSqlAdapter,
18                             IAtendenteSqlAdapter atendenteSqlAdapter)
19         {
20             this.pedidosSqlAdapter = pedidosSqlAdapter ?? throw new ArgumentNullException(nameof(pedidosSqlAdapter));
21             this.atendenteSqlAdapter = atendenteSqlAdapter ?? throw new ArgumentNullException(nameof(atendenteSqlAdapter));
22
23             3 references | 4/4 passing | 0 changes | 0 authors, 0 changes | 0 exceptions
24             public async Task<Pedido> CriarPedidoAsync(Guid atendenteId, int numero)
25             {
26                 var atendente = await atendenteSqlAdapter.GetAsync(atendenteId);
27
28                 if (atendente == null)
29                     throw new PedidosCoreException(PedidosCoreError.AtendenteNaoEncontrado);
30
31                 var pedido = new Pedido() { Atendente = atendente, Mesa = 1 };
32                 pedido = await this.pedidosSqlAdapter.InsertAsync(pedido);
33                 return pedido;
34             }
35     }
36 }
37
```

**Code Coverage Results:**

Hierarchy	Not Covered (Blocks)	Not Covered (% Blocks)	Covered (Blocks)
mathe_DESKTOP-MATHEUS 2019-05-25 01...	48	17,98%	219
└─ exemplo.testesdedominio.appli...	24	36,36%	42
└─ {} Exemplo.TestesDeDominio...	14	25,00%	42
└─ {} Microsoft.Extensions.Depen...	10	100,00%	0
└─ exemplo.testesdedominio.appli...	4	2,48%	157
└─ exemplo.testesdedominio.dom...	20	50,00%	20

- Disponível no Visual Studio Enterprise
- A extensão “*OpenCover UI*” é uma alternativa para se utilizar no Visual Studio Professional.

# Boas práticas

Respeitar o significado dos parâmetros

- ▶ Validar o resultado (por meio de Assert)

- ▶ `void Equal<T>(T expected, T actual);`
- ▶ `void Contains<T>(T expected, IEnumerable<T> collection);`
- ▶ `void True(bool condition);` / `void False(bool condition);`
- ▶ `void Null(object @object);` / `void NotNull(object @object);`
- ▶ `T Throws<T>(Func<object> testCode) where T : Exception;`
- ▶ `Task<T> ThrowsAsync<T>(Func<Task> testCode)`

Métodos mais comuns

- ▶ Evitar: `Assert.True(valorEsperado == valorProduzido);`



# Boas práticas

Respeitar o significado dos parâmetros

- ▶ Validar o resultado (por meio de Assert)

- ▶ `void Equal<T>(T expected, T actual);`
- ▶ `void Contains<T>(T expected, IEnumerable<T> collection);`
- ▶ `void True(bool condition);` / `void False(bool condition);`
- ▶ `void Null(object @object);` / `void NotNull(object @object);`
- ▶ `T Throws<T>(Func<object> testCode) where T : Exception;`
- ▶ `Task<T> ThrowsAsync<T>(Func<Task> testCode)`

Métodos mais comuns

- ▶ Evitar: `Assert.True(valorEsperado == valorProduzido);`

