Pós Tech Arquitetura de Sistemas .Net

Testes Unitários com xUnit

Prof. Daniele Garcia

/ O que são testes unitários?

Definição:

- Testes unitários são pequenos testes automatizados que verificam o comportamento de **unidades isoladas** de código, como funções, métodos ou classes.
- São escritos por desenvolvedores para garantir que cada parte do código funcione conforme esperado.

/ O que são testes unitários?

Características principais:

- **Isolados:** Não dependem de outras partes do sistema ou de serviços externos (como banco de dados ou APIs).
- **Reprodutíveis:** Sempre produzem os mesmos resultados quando executados sob as mesmas condições.
- **Rápidos:** Podem ser executados frequentemente, inclusive durante o desenvolvimento.

/ Benefícios dos testes unitários

- **Confiança no código:** permitem identificar e corrigir problemas cedo, antes que o código seja integrado a outras partes do sistema.
- **Prevenção de erros em produção**: reduzem a probabilidade de bugs, principalmente em funcionalidades críticas.
- **Suporte à refatoração:** oferecem segurança para modificar o código existente, garantindo que o comportamento esperado seja mantido.
- **Documentação viva:** servem como exemplos claros do uso esperado do código.
- **Facilitam a colaboração:** outros desenvolvedores podem entender e testar o sistema mais facilmente.

/ Testes no ciclo de desenvolvimento

- Testes em Metodologias Ágeis
  - Importância dos testes unitários em práticas como Test-Driven Development (TDD), onde os testes são escritos antes do código.
  - Sustentação da entrega contínua (Continuous Delivery) e do desenvolvimento incremental.
- Integração em CI/CD
  - Automação de testes em pipelines, garantindo qualidade antes de integrar ou lançar novas versões do sistema.
  - Contribuição para o DevOps, permitindo entregas rápidas e seguras.

/ Outros tipos de testes

- Testes de Integração
  - Verificam a interação entre diferentes partes do sistema (ex.: comunicação entre módulos ou sistemas externos).
  - Exemplo: Testar se um serviço consome corretamente uma API externa.
- Testes End-to-End (E2E):
  - Simulam o uso real do sistema, testando fluxos completos do usuário.
  - Exemplo: Testar se um usuário consegue realizar um login e acessar uma página específica.

/ O que testar?

- Lógica de negócios
  - Testar regras importantes, como cálculos financeiros, regras tributárias, ou validações críticas que influenciam o funcionamento do sistema.
- Regras de validação
  - Certificar-se de que os dados de entrada atendem aos critérios esperados.
- Fluxos críticos
  - Identificar pontos sensíveis que, se falharem, podem comprometer o funcionamento do sistema

/ Características de um bom teste

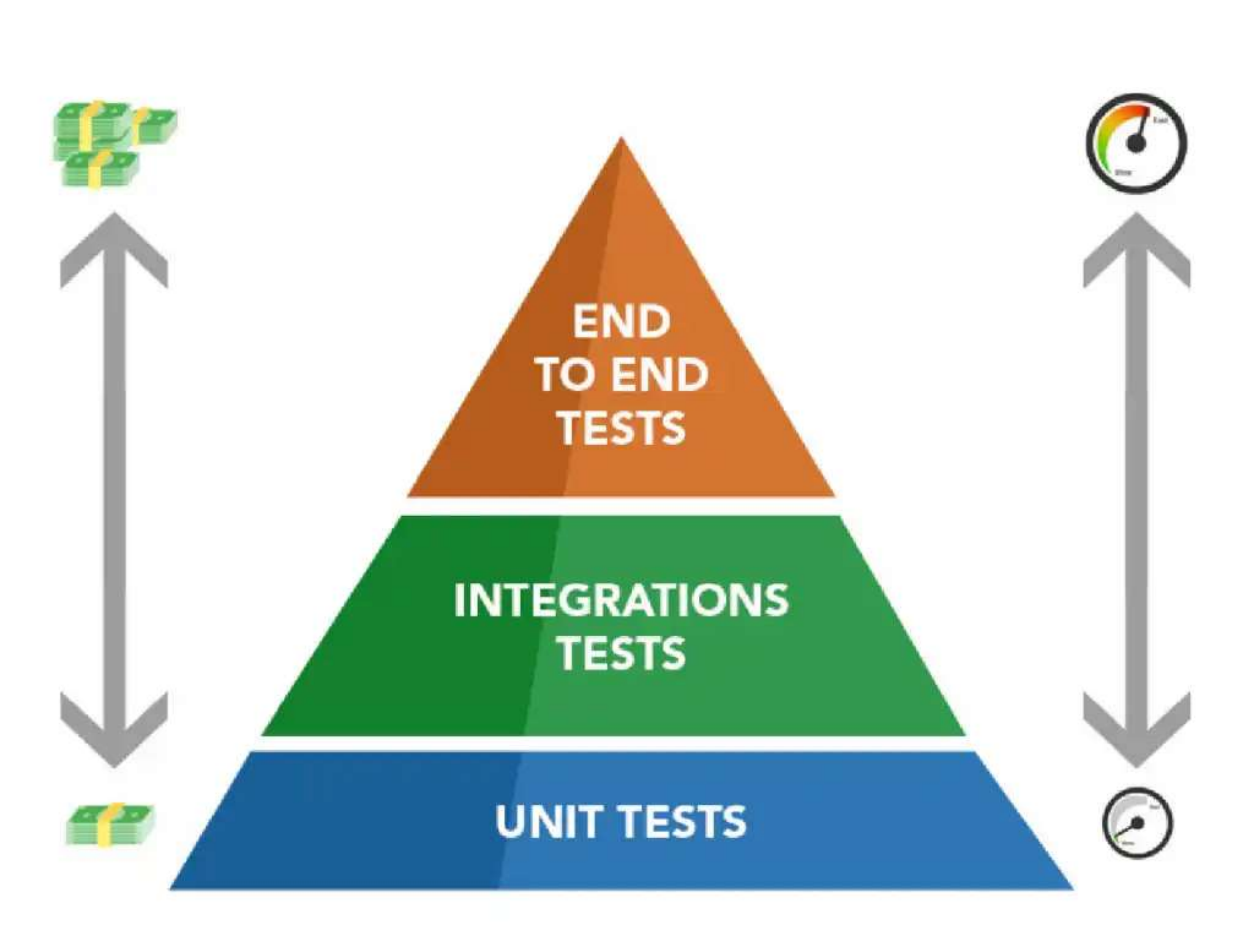
- Independência: deve ser isolado e não depender de outros testes ou de elementos externos (banco de dados, APIs, etc.).
- Clareza: o nome do teste deve ser descritivo e explicar o que está sendo testado e em qual cenário.
- Foco em uma única responsabilidade
- Reprodutibilidade: sempre produza o mesmo resultado para as mesmas condições de entrada.

/ Pirâmide de Testes

Baseada no conceito de garantir qualidade no desenvolvimento, a pirâmide de testes organiza os diferentes tipos de testes em camadas.

- **Base (Testes Unitários):** Muitos testes, rápidos, baratos, cobrindo unidades isoladas.
- **Meio (Testes de Integração):** Menos testes, validando a comunicação entre componentes.
- **Topo (Testes End-to-End):** Poucos testes, mais lentos e caros, cobrindo fluxos completos.

/ Pirâmide de Testes



/ Frameworks de Teste

- Por que usar frameworks?
  - **Automatização:** Simplificam a execução de testes de forma programática.
  - **Convenções e boas práticas:** Oferecem padrões que ajudam a organizar e estruturar os testes.
  - **Produtividade:** Recursos como asserções, parametrização e relatórios facilitam o trabalho.

/ XUnit

O que é o XUnit?

O XUnit é um framework open-source para criar e executar testes automatizados no .NET.

Ele é uma escolha comum devido à sua abordagem moderna e minimalista, que facilita a escrita de testes limpos e eficazes.

/ XUnit

- Características principais:
  - Atributos como [Fact] (para cenários únicos) e [Theory] (para testes parametrizados).
  - Compatível com ferramentas de CI/CD e com suporte para injeção de dependência.
- Vantagens do XUnit:
  - É leve, extensível e bem integrado ao ecossistema do .NET.

/ Frameworks para outras linguagens

- NET → xUnit, NUnit, MSTest
- Java → JUnit, TestNG
- JavaScript/TypeScript → Jest, Mocha
- Python → pytest, unittest
- C/C++ → Google Test (gTest), Catch2
- PHP → PHPUnit
- Dart/Flutter → test, Mockito

/ Estrutura básica de testes

1. Nomeação Clara e Significativa
   - Seguir o padrão: MétodoEmTeste_CenárioASerTestado_ResultadoEsperado

2. Três Partes Fundamentais (AAA - Arrange, Act, Assert)
   - Arrange (Preparação): Configura o ambiente e as variáveis necessárias.
   - Act (Ação): Executa o método a ser testado.
   - Assert (Verificação): Compara o resultado obtido com o esperado.

```
[Fact]
0 references
public void Add_TwoPositiveNumbers_ShouldReturnCorrectSum()
{
    // Arrange
    int number1 = 2;
    int number2 = 3;
    int expectedSum = number1 + number2;

    // Act
    var result = Calculator.Add(number1, number2);

    // Assert
    Assert.Equal(expectedSum, result);
}

[Theory]
[InlineData(2, 2, 4)]
[InlineData(2, 3, 5)]
[InlineData(3, 3, 6)]
0 references
public void Add_TwoNumbers_ShouldReturnCorrectSum(int number1, int number2, int expectedSum)
{
    // Act
    var result = Calculator.Add(number1, number2);

    // Assert
    Assert.Equal(expectedSum, result);
}
```

/ Configuração do ambiente

Exemplo prático...

/ ! FluentAssertions agora é paga

A partir de 2024, o FluentAssertions exige licença comercial para projetos empresariais.

Para projetos educacionais, open source ou pessoais, ainda é gratuito.

Alternativas recomendadas:

✅ Shouldly (fluente e gratuito)

✅ Asserts nativos do xUnit

⚠ Moq perdeu credibilidade

Em 2023, Moq adicionou um pacote externo de telemetria (SponsorLink), o que levantou preocupações sobre privacidade.

Comunidade reagiu negativamente, pois o pacote foi incluído sem consentimento explícito.

Como alternativa:

✅ NSubstitute: Conciso, fluente e sem polêmicas.

✅ Moq ainda pode ser usado, mas com consciência (verifique se a telemetria está desativada).

/ Automação em pipelines CI/CD

Por que integrar testes em pipelines?

- Garante que os testes sejam executados automaticamente a cada alteração no código.
- Identifica problemas antes da integração ou implantação.
- Melhora a qualidade e a confiança no sistema.

/ Links úteis

Melhores práticas de teste de unidade com .NET Core:

https://learn.microsoft.com/pt-br/dotnet/core/testing/unit-testing-best-practices

Teste de unidade do C# no .NET usando dotnet test e xUnit:

https://learn.microsoft.com/pt-br/dotnet/core/testing/unit-testing-with-dotnet-test

https://xunit.net/

https://documentation.help/Moq/

https://nsubstitute.github.io/help/getting-started/

Testar é como colocar o capacete antes de andar de moto: você pode até andar sem, mas um dia vai se arrepender de não ter usado.

/ Obrigada!

https://www.linkedin.com/in/danielegarciadecastroalves/

https://github.com/danielecastroalves/TestesUnitariosXUnit


PDS TECH
FIAP | alura | PM3