Curso ATDEV330 - Testes Unitários Eficazes

Duração: 14h

Introdução

Escrever código sólido, fácil de manter e que funciona é o objetivo de qualquer programador. Este curso conduz-o pela arte dos testes de software, onde aprende a construir código testável e várias técnicas para garantir qualidade. No fim, terá uma compreensão sólida e prática dos testes de software e das suas aplicações reais. Cada capítulo é acompanhado por um laboratório prático.

Público-alvo

Este curso destina-se a programadores experientes, muito familiarizados com C# e com experiência prática de .NET 8 a 10.

No final do curso

Após a conclusão deste curso, os formandos deverão ser capazes de:
• Escrever testes unitários em .NET com MSTest e xUnit.
• Utilizar dependency injection e frameworks de isolamento como o NSubstitute.
• Tornar testável o código legado.
• Implementar testes de integração e testes ponta a ponta com Playwright.
• Validar regras de arquitetura com testes de arquitetura.

Pré-requisitos

Bons conhecimentos de C# e experiência prática de .NET 8 a 10. Todos os exemplos e laboratórios usam a versão LTS mais recente do .NET e do Visual Studio.

Exames de certificação

(não existem exames)

Conteúdo programático em detalhe

Testes unitários em .NET
• O que qualquer programador quer: código de qualidade que funciona — e continua a funcionar
• Encontrar erros: não só no código. Ou como a NASA perdeu um orbitador de Marte de 125 milhões de dólares
• O que são testes unitários? E o que é um bom teste unitário?
• Compreender a diferença entre um teste unitário e um teste de integração
• Test-driven development — deve praticá-lo?
• Jogo de papéis: a Marge, o Homer e o Bart, ou o caminho feliz vs. o caminho negativo
• O triplo A dos testes unitários — e o que tem a cozinha a ver com isto?
• Given-When-Then
• A diferença entre um fake, um stub e um mock
Testes unitários com MSTest e xUnit
• MSTest — vem incorporado no Visual Studio

- Construir e executar testes unitários com MSTest
- Utilizar a janela Test Explorer
- Utilizar as test settings
- Live unit testing com o Visual Studio
- Boas práticas de testes unitários com xUnit
- Facts e theories
- Executar testes unitários em paralelo, ou não
- Convenções de nomes para testes
- Medir e melhorar a cobertura de código
- Comparação com o NUnit e o TUnit
- LAB: implementar uma biblioteca com testes

Dependency injection para testes

- O que é o acoplamento forte e como evitá-lo?
- Utilizar contentores de inversão de controlo (IoC)
- Injeção por construtor, por propriedade e por método
- Microsoft.Extensions.DependencyInjection como exemplo

Tornar testável o código legado

- Testar dependências — e a arte de escrever código testável
- O padrão Extract and Override
- Simular o tempo com o TimeProvider
- Substituir a configuração nos testes
- LAB: usar Extract and Override para testar código legado

Frameworks de isolamento: NSubstitute

- Compreender as frameworks de isolamento
- Construir stubs e mocks com NSubstitute
- Verificar argumentos e valores devolvidos
- Utilizar argument matchers
- Configurar valores devolvidos e exceções
- Verificar que as chamadas foram feitas
- Utilizar o Object Mother pattern
- LAB: usar o NSubstitute

Testes de integração

- Quando deve escrever testes de integração?
- Usar a WebApplicationFactory em testes de integração
- Simular o acesso a dados
- Testar com acesso real a dados
- Utilizar TestContainers
- Testar os seus controllers e views
- Usar o AngleSharp para verificar views
- Testar ApiControllers
- LAB: usar a WebApplicationFactory

Testes ponta a ponta com Playwright

- Testes ponta a ponta com o Microsoft Playwright
- Construir o seu primeiro teste em Playwright

- Gravar um teste em Playwright
- Registo de interações, capturas de ecrã e vídeos
- LAB: usar o Playwright

Testes de arquitetura

- Quando são úteis os testes de arquitetura?
- Escrever testes de arquitetura com o NetArchTest
- Verificar regras de camadas
- Validar a direção das dependências
- Dependências proibidas
- Convenções de nomes
- LAB: acrescentar testes de arquitetura