Curso ATDEV310 - Arquitetura Moderna em .NET

Duração: 35h

Introdução

A resposta à pergunta fundamental sobre a vida, o universo e a boa arquitetura é: depende. Esta formação prepara programadores e arquitetos .NET para tomarem decisões de compromisso fundamentadas entre estilos de arquitetura, modelos de deployment e práticas de governação. Dá as bases para estruturar projetos fáceis de manter, impor quality gates e equilibrar a produtividade da equipa com a sustentabilidade a longo prazo, sem cair em anti-padrões comuns como leaky abstractions ou o inferno das dependências.

Público-alvo

Esta formação destina-se a programadores .NET de nível intermédio e sénior que precisem de compreender mais a fundo a arquitetura com que trabalham:
• Programadores .NET séniores em transição para funções de arquitetura
• Arquitetos de solução que desenham aplicações .NET de dimensão empresarial
• Responsáveis técnicos encarregados da qualidade do código, da estratégia de testes e da governação da arquitetura

No final do curso

Após a conclusão deste curso, os formandos deverão ser capazes de:
• Avaliar estilos de arquitetura e tomar decisões de compromisso fundamentadas.
• Estruturar soluções .NET em N camadas, com Clean Architecture ou com Vertical Slice Architecture.
• Aplicar CQRS, messaging patterns e Result objects no desenho de aplicações.
• Escolher entre microserviços e monólito modular, e entre monorepo e polyrepo.
• Construir bibliotecas reutilizáveis e impor quality gates.
• Implementar monitorização, observabilidade e feature management.

Pré-requisitos

Base sólida de C#/.NET; não é necessária formação prévia em arquitetura.

Exames de certificação

(não existem exames)

Conteúdo programático em detalhe

Princípios de arquitetura
• O papel do arquiteto
• O que é boa arquitetura?
• Fatores determinantes da arquitetura
• Princípios de arquitetura: encapsulamento, separação de responsabilidades e outros
Arquitetura em N camadas
• O que é o N-Tier?
• Camadas físicas vs. camadas lógicas

• A pilha em camadas habitual: acesso a dados, negócio, apresentação
• Modelos anémicos
• Testabilidade, facilidade de manutenção, desempenho
• LAB: criar uma aplicação N-Tier simples

Clean Architecture

• Inversão de dependências
• Camadas de domínio, aplicação, infraestrutura e apresentação
• Modelação de domínio com entidades e value objects
• Serviços de domínio, de aplicação e de infraestrutura
• Eventos de domínio
• Testabilidade, facilidade de manutenção, desempenho
• LAB: migrar de N-Tier para Clean Architecture

Mapping

• Porquê e onde mapear
• Tipos de mapping
• Mapping manual vs. automático
• Pureza arquitetural e produtividade do programador
• Desempenho e facilidade de manutenção
• LAB: usar source generation para mapping

Estratégia de persistência

• Repository pattern
• Unit of Work pattern
• Utilização direta do DbContext
• Aggregates
• A armadilha do «repository genérico»
• Specification pattern
• A realidade dos testes unitários
• LAB: usar o DbContext em Clean Architecture

Vertical Slice Architecture

• O «imposto» das camadas
• A filosofia das vertical slices
• Gerir a lógica partilhada
• A pirâmide de testes vs. o diamante
• Facilidade de manutenção e desempenho
• LAB: migrar de Clean Architecture para vertical slices

Result objects e exceções

• Exceções vs. códigos de erro
• Exceções de domínio
• Padrões de controlo de fluxo — Result pattern
• Erros de domínio
• Exceções vs. results
• Abordagem híbrida
• LAB: explorar compromissos com o FluentResults

CQRS

- CQS
- CQRS ao nível do sistema
- CQRS ao nível da aplicação
- Commands, queries e handlers
- CQRS vs. CRUD
- Preocupações transversais
- LAB: acrescentar CQRS a Clean Architecture

Messaging patterns

- Mediator pattern
- Sistema de mensagens de uso geral
- Pedido/resposta e notificações
- Mediator: herói ou vilão?
- Bibliotecas de Mediator e as críticas que lhes são feitas
- Eventos de domínio
- Event aggregator
- LAB: implementar mensagens com o Wolverine

Microserviços

- Lidar com a evolução de uma aplicação
- Vantagens e desvantagens
- Comunicação: cliente-serviço e serviço-serviço
- Isolamento de dados
- LAB: construir um microserviço simples com Aspire

Monólito modular

- O «imposto» dos microserviços
- O monólito modular como meio-termo estratégico
- Comunicação e integração entre módulos
- Estratégias de isolamento de dados
- Riscos e armadilhas de implementação
- LAB: de microserviços a monólito modular

Monorepo vs polyrepo

- Refatoração atómica
- Gestão centralizada
- O mito da «build lenta»
- A experiência de depuração
- Propriedade das equipas
- Migração

Construir bibliotecas reutilizáveis

- Princípios de desenho de APIs
- Gestão de dependências
- Versionamento
- Configuração e extensibilidade
- Escrever código agnóstico ao contexto
- Evolução e governação
- LAB: construir uma biblioteca reutilizável para processamento de documentos

Quality gates

- Validação da build
- Estilo e formatação do código
- Análise estática de código
- Testes e limiares de cobertura
- Testes de arquitetura
- Auditorias de segurança e de dependências
- Revisões de código
- Orientações para geração de código por IA
- LAB: montar um projeto com quality gates

Monitorização e observabilidade

- Monitorização vs. observabilidade
- OpenTelemetry em .NET
- Logging estruturado, traces e métricas
- Amostragem
- Engenharia de fiabilidade
- Saúde e ciclo de vida da aplicação
- LAB: OpenTelemetry com Aspire

Feature management

- Feature management em .NET
- Trunk-based development com release flags
- Lançamento gradual
- Testes A/B
- Observabilidade
- Boas práticas e armadilhas
- LAB: implementar um kill switch de feature management