Guia Prático de Arquitetura de Projeto C# + Angular

Este documento descreve um fluxo de trabalho recomendado, baseado em boas práticas de Clean Architecture, DDD pragmático e desenvolvimento guiado por casos de uso.

1. Etapa Inicial – Modelagem

Antes de qualquer implementação, é fundamental realizar a modelagem do domínio:

- MER – Modelo Entidade-Relacionamento
- DER – Diagrama Entidade-Relacionamento

2. Criação da Solution

Crie a solution e os projetos desde o início, mesmo que estejam vazios. Isso garante organização e evita retrabalho.

Estrutura recomendada:

- Domain – Regras de negócio e contratos
- Application – Services e casos de uso
- Infrastructure – EF Core, Repositories, integrações
- API – Controllers e endpoints HTTP

3. Configurações Iniciais

Nesta fase, configure apenas o essencial:

- Injeção de Dependência (DI)
- Swagger
- CORS e logging
- Entity Framework Core apenas de forma preparatória (pacotes, projeto Infrastructure e DbContext base)

4. Desenvolvimento Outside-in (Começando pelo Controller)

O desenvolvimento começa pelo Controller, pois ele representa o caso de uso real exposto pela API. Neste momento, o Controller depende apenas da interface da Service.

5. Camada Application – Services

Crie as interfaces e implementações de Service. A Service orquestra o fluxo do caso de uso e depende apenas do Domain.

6. Camada Domain

O Domain contém o núcleo do sistema e não depende de nenhuma outra camada.

- Entidades
- Value Objects
- Enums
- Interfaces de Repository (contratos)

7. Camada Infrastructure

Aqui são implementadas as interfaces definidas no Domain, utilizando EF Core ou outras tecnologias de persistência.

8. DbContext e Banco de Dados

Por fim, configure completamente o DbContext, as entidades, migrations e a connection string do banco de dados.

9. Ordem Resumida do Processo

- MER / DER
- Criar solution e projetos
- Configurações iniciais
- Controller
- Interface da Service
- Domain
- Service (implementação)
- Repository concreta
- DbContext, EF Core e migrations

Este fluxo equilibra boas práticas de arquitetura com pragmatismo, permitindo evolução segura e entrega contínua.