FMX Academy

INFORMAÇÕES		
MÓDULO	**AULA**	**INSTRUTOR**
06 ASP.NET Core Web API – Avançado e Segurança Inicial	**02 - Aula de Revisão: ASP.NET Core Web API (Módulo 05)**	**Guilherme Paracatu**

1. Aula de Revisão: O Fluxo de Dados: Por onde o código passa?

Objetivo: Consolidar os pilares de APIs REST, EF Core 8 e Arquitetura

FERRAMENTAS: VS Code, Projeto ApiEstoque, PostgreSQL.

Muitas vezes a dificuldade surge por não saber onde cada parte do código se encaixa. Imagine a sua API como uma linha de produção:

1. **Controller (A Portaria):** Recebe o pedido HTTP. Não toma decisões difíceis, apenas repassa para o Service.

2. **Service (O Escritório/Cérebro):** Onde a mágica acontece. Ele recebe dados brutos e os transforma em informações úteis (DTOs). É aqui que validamos regras de negócio.

3. **Repository (O Almoxarifado):** É o único que mexe nas prateleiras do banco de dados (DbContext).

4. **Database (PostgreSQL):** Onde os dados estão guardados em tabelas minúsculas.

PostgreSQL: A "Armadilha" das Letras Maiúsculas

Se você recebeu erros como relação "**Produtos**" não existe ou coluna "**Id**" não existe, o culpado é o **Case Sensitivity.**

- **O Comportamento:** O PostgreSQL trata nomes sem aspas como minúsculos. O Entity Framework Core, por padrão, tenta buscar nomes em PascalCase (ex: Produtos, FabricanteId).

- **A Solução Definitiva (Fluent API):** Não confie nas convenções automáticas. Use o OnModelCreating no seu DbContext para dar o mapa exato ao EF Core.

Exemplo de Mapeamento Obrigatório:

```
modelBuilder.Entity<Produto>(entity => {
    entity.ToTable("produtos"); // Nome da tabela no banco
    entity.Property(p => p.Id).HasColumnName("id"); // Nome da coluna PK
    entity.Property(p => p.FabricanteId).HasColumnName("fabricanteid"); // Nome da coluna FK
});
```

2. Por que usamos DTOs e Services?

Na última prática, alguns sentiram dificuldade em separar as camadas. Lembre-se:

- **Entidade:** É o espelho do banco. Se ela mudar, o banco muda.
- **DTO (Data Transfer Object):** É o contrato com o cliente. Se o cliente quer ver o "Nome do Fabricante" em vez do "ID do Fabricante", o DTO resolve isso.
- **O papel do .Include():** Se você esquece o .Include(p => p.Fabricante) no Repository, o seu Service tentará acessar o nome do fabricante e receberá um erro de NullReferenceException.

Guia de Debug: "Meu código não funciona, e agora?"

Se o erro for...	Onde está o problema?	Como resolver?
42P01: relação "..." não existe	**DbContext / Fluent API**	Verifique se o nome no .ToTable() está igual ao banco.
42703: coluna "..." não existe	**DbContext / Fluent API**	Verifique se o nome no .HasColumnName() está minúsculo.
System.NullReferenceException	**Repository / LINQ**	Verifique se você esqueceu o .Include() para carregar os dados relacionados.
Unable to resolve service for...	**Program.cs**	Verifique se você esqueceu o builder.Services.AddScoped<I..., ...>().