10

O que aprendemos?

Barbara presente!

Começamos a aula criando uma nova abstração `IDao` que subia um nível de abstração e garantia que todos os DAOs tivessem o mesmo comportamento. Contudo tivemos que relaxar a implementação dos métodos `Incluir()`, `Alterar()` e `Excluir()` de `CategoriaDaoComEfCore` porque o sistema não tinha casos de uso para isso. Barbara Liskov havia estudado o recurso de heranças em OO e reparou que hierarquias problemáticas eram aquelas onde os tipos descendentes não conseguiam ser substituídos plenamente por seus ancestrais. Robert Martin trouxe essa idéia para a letra L homenageando Barbara: surgia o Princípio da Substituição de Liskov (LSP). Robert aumentou a idéia para abranger qualquer abstração. Podemos então afirmar: **faça com que todas as implementações cumpram as promessas de suas abstrações**.

Você realmente precisa disso?

Um hábito bastante comum no desenvolvimento de software (fiquei pensando em escrever aqui *humanidade*, o que acha?) é antecipar problemas e com isso querer adiantar as soluções. Isso acaba enchendo o projeto de coisas que não serão usadas pelos usuários. **Não implemente funcionalidades que ainda não foram solicitadas**! Essa idéia foi cunhada no acrônimo Y.A.G.N.I., You Aint Gonna Need It.

Acima de qualquer suspeita?

Interfaces (e qualquer abstração!) também devem seguir os princípios de coesão e acoplamento. Senão correm o risco de propagar promessas que em algum momento não poderão ser cumpridas por seus descendentes e implementações. Sempre que possível busque separar as operações das interfaces em grupos menores. Essa idéia é o **Princípio da Segregação das Interfaces** (ISP), uma lembrança do tio Bob para não descuidarmos no design de nossas interfaces. Para quebrar a interface `IDao` usamos a idéia do padrão CQRS (Command and Query Responsibility Segregation) para definir duas interfaces `ICommand` e `IQuery`, uma para as operações de leitura e outra para as operações de escrita.