Camadas Desacopladas SOLID NA PRÁTICA

Uma tendência na tecnologia que motivou a separação de camadas foi o fato de que em algumas aplicações, como era comum a época, tudo estava misturado na mesma camada: a visualização, as regras e a persistência de dados no banco. Isso tornava o código rígido para mudanças e testes, porque havia dependências internas que não podiam ser testadas de forma isolada. Conforme ensinado em sala, trata-se de uma forma de violar o Princípio de Responsabilidade Única, uma vez mistura as competências do banco de dados com o funcionamento das requisitções, no código.

Quando falamos em fazer uma arquitetura limpa, queremos dizer que o código deve ser organizado de forma que seja fácil fazer mudanças nele. Isso é feito seguindo princípios como o SOLID, que visam tornar o código independente de mudanças externas. Basicamente, isso significa que o software é dividido em diferentes partes, ou camadas, para que seja mais fácil adaptá-las e fazer manutenção. Uma solução para desenvolver melhor a aplicação, que também inclui testes, é o desacoplamento. Isso significa separar os casos de uso, como no exemplo do vídeo, para separar a camada do banco da camada com as regras de negócio. Poderia utilizar uma interface que contém as funcionalidades básicas para um servidor http. Então, a implementação se daria para cada caso de utilização do servidor, o que seria útil caso trocasse o provedor do servidor.

No caso do vídeo, começamos com uma rápida visão geral, comparando o CDD (com mau exemplo) com o QDD. Para tornar isso didático, criou-se uma abordagem inicial, que envolve a criação de um teste de integração. Ou seja, quando as camadas não estão bem definidas, usamos um teste de integração. O fundamental ao entendimento do QDD é que uma situação será dada, teremos as ocorrências do código (os eventos, as ações) e então o nosso resultado esperado.