Resenha capítulos 6 e 7 Engenharia de Software Moderna

Os capítulos 6 e 7 do livro **"Engenharia de Software Moderna"** falam sobre duas coisas importantes no desenvolvimento de software: **Padrões de Projeto** e **Arquitetura de Software**. Esses conceitos ajudam a deixar os sistemas mais flexíveis, fáceis de manter e escaláveis.

O **Capítulo 6** explica o que são os **padrões de projeto**, que são soluções prontas para problemas que aparecem a todo momento quando se está criando software. Esses padrões foram inspirados nas ideias de Cristopher Alexander e adaptados para o mundo da programação pela "Gang of Four". São apresentados dez padrões que todo desenvolvedor precisa conhecer, como **Fábrica**, **Singleton**, **Proxy**, **Adaptador**, **Fachada**, **Decorador**, **Strategy**, **Observador**, **Template Method** e **Visitor**.

Cada padrão é mostrado com exemplos práticos: o **Strategy**, por exemplo, permite mudar o algoritmo de um processo sem alterar o código principal, enquanto o **Singleton** garante que só exista uma instância de uma classe, apesar de ser criticado por criar dependências complicadas. A ideia principal é que esses padrões ajudam a resolver problemas de design de forma mais eficiente, deixando o código mais fácil de entender e de manter.

Mas o capítulo também avisa: não dá para usar padrões de projeto para tudo. Em alguns casos, eles podem deixar o código mais complicado do que precisa ser. Então, é sempre importante saber quando **não** usar.

O **Capítulo 7** muda o foco para a **arquitetura de software**, que trata das decisões mais importantes no design de um sistema. Aqui, não se fala de classes e métodos individuais, mas de como os grandes blocos do sistema se organizam. O capítulo apresenta alguns padrões arquiteturais bem conhecidos, como a **Arquitetura em Camadas**, o **MVC** (Model-View-Controller) e os **Microsserviços**.

A **Arquitetura em Camadas** organiza o sistema em módulos hierárquicos, facilitando a troca de componentes e a manutenção. O **MVC** separa a lógica de apresentação, controle e dados, o que deixa o desenvolvimento

mais organizado e permite, por exemplo, trocar a interface gráfica sem mexer no resto do sistema.

Os **Microsserviços** são discutidos como uma solução moderna para sistemas grandes e monolíticos, onde cada serviço roda de forma independente, o que facilita muito na hora de escalar e atualizar o sistema sem mexer em tudo de uma vez. Só que, claro, isso também traz desafios, como a comunicação entre os serviços e a gestão de transações distribuídas.

O capítulo também fala sobre o que **não fazer**, como o famoso anti-padrão **Big Ball of Mud**, que acontece quando o sistema vira uma bagunça sem organização, difícil de manter e evoluir.

Esses dois capítulos se complementam bem. Enquanto os **padrões de projeto** ajudam a resolver problemas específicos de design no dia a dia, a **arquitetura** cuida da estrutura geral do sistema, garantindo que ele seja escalável e fácil de evoluir. Juntos, eles formam a base para criar software de qualidade, que consegue se adaptar e crescer sem perder a organização.