Capitulo 7 resumo:

Introdução à Arquitetura de Software

Este capítulo apresenta a arquitetura de software, que envolve o design de alto nível e decisões essenciais em um sistema. São explorados padrões arquiteturais como Arquitetura em Camadas, MVC, Microsserviços, Arquiteturas Orientadas a Mensagens e Publish-Subscribe para organizar sistemas de software. Destaca-se a importância das decisões arquiteturais e a dificuldade de revertê-las após implementadas. A arquitetura de três camadas inclui interface do usuário, lógica de negócios e banco de dados, enquanto a de duas camadas combina interface e lógica de negócios.

Model-View-Controller (MVC)

O padrão MVC divide um sistema em três partes: View (interface gráfica), Controller (controle de eventos) e Model (dados e lógica de domínio). Essa separação facilita o desenvolvimento especializado, permite reutilização de Model por diferentes Views e melhora a testabilidade. Existem duas versões: a tradicional, originada no Smalltalk, e a versão Web, similar à arquitetura de três camadas, adotada em frameworks como Spring, Ruby on Rails e Django.

Arquitetura de Microsserviços

A arquitetura de microsserviços divide um sistema em módulos independentes que operam separadamente, reduzindo o impacto de mudanças. Esse modelo oferece escalabilidade, autonomia e suporte a diferentes tecnologias, além de permitir falhas parciais sem comprometer todo o sistema. É ideal para organizações grandes e descentralizadas e se adapta bem a plataformas de computação em nuvem.

Filas de Mensagens e Publish/Subscribe

Filas de mensagens desacoplam componentes no tempo e no espaço, permitindo maior autonomia no desenvolvimento e tornando a arquitetura mais robusta contra falhas. O padrão Publish/Subscribe também promove esse desacoplamento, mas com comunicação de um para muitos, sendo característico de arquiteturas orientadas a eventos.

Outros Padrões Arquiteturais e Considerações

Outros padrões incluem Pipes and Filters, Cliente/Servidor, Peer-to-Peer e o anti-padrão Big Ball of Mud, caracterizado pela ausência de uma arquitetura definida e excesso de dependências. A função das classes Controller é comparada entre o MVC tradicional e frameworks como Ruby on Rails. Embora os microsserviços ofereçam vantagens, apresentam desafios e sua adoção está relacionada à Lei de Conway. As filas de mensagens e o padrão Publish/Subscribe facilitam a distribuição direcionada de mensagens em sistemas distribuídos.