Resenha do capítulo 7

A arquitetura de software é um campo essencial na engenharia de sistemas, abordando como organizar as unidades de um sistema, como pacotes, módulos ou serviços, para atingir objetivos específicos, como em sistemas de informações ou inteligência artificial. Essas decisões de arquitetura são críticas, pois são difíceis de reverter e impactam a estrutura e evolução do sistema ao longo do tempo. A escolha de banco de dados, linguagens de programação e outros componentes afeta diretamente o desenvolvimento e a manutenção do sistema.

Entre os padrões arquiteturais mais comuns estão a Arquitetura em Camadas, o padrão Model-View-Controller (MVC) e a arquitetura de Microsserviços. Esses modelos buscam otimizar a organização, a manutenção e a evolução de sistemas complexos, criando um sistema modular e mais flexível. No entanto, como demonstra o Debate Tanenbaum-Torvalds, a escolha de uma arquitetura pode ter consequências duradouras e, em muitos casos, irreversíveis. A divergência entre a arquitetura monolítica e microkernel, defendida respectivamente por Tanenbaum e Torvalds, ilustra como diferentes escolhas arquiteturais podem gerar resultados opostos, influenciando a complexidade e a manutenção dos sistemas operacionais ao longo dos anos.

A Arquitetura em Camadas é uma abordagem clássica, onde as funcionalidades são organizadas em camadas hierárquicas, com cada camada interagindo apenas com a camada imediatamente inferior. Esse modelo ajuda a modularizar o sistema, facilitando a manutenção e evolução, pois mudanças em uma camada não afetam diretamente as demais. A abordagem é amplamente aplicada em sistemas de rede, onde a separação de responsabilidades é essencial para a eficiência.

A Arquitetura em Três Camadas, por sua vez, é uma evolução da arquitetura em camadas, muito utilizada em sistemas corporativos. Ela divide o sistema em três camadas distintas: a Camada de Apresentação, que lida com a interação do usuário; a Camada de Lógica de Negócios, responsável pelo processamento das regras do sistema; e a Camada de Banco de Dados, que armazena e gerencia os dados. Essa arquitetura distribui as camadas entre servidores, o que facilita a escalabilidade do sistema, mas exige maior poder computacional do cliente.

Nesse contexto, a arquitetura MVC ganha destaque, especialmente em sistemas com interfaces gráficas. Criada para o desenvolvimento de aplicações em Smalltalk nos anos 70, ela divide o sistema em três componentes principais: o Modelo (Model), responsável pela lógica de negócios e dados; a Visão (View), que apresenta os dados ao usuário; e o Controlador (Controller), que gerencia as interações entre o Modelo e a Visão. O padrão MVC promove a separação de responsabilidades, o que facilita a manutenção, a escalabilidade e a reutilização de componentes.

Um exemplo claro de aplicação do padrão MVC são as Single Page Applications (SPAs), que imitam a experiência de aplicativos desktop, realizando comunicação assíncrona com o servidor para garantir uma experiência fluida ao usuário. Essas aplicações frequentemente utilizam o padrão MVC, onde a Visão e o Controlador são gerenciados no lado do cliente, enquanto o Modelo é mantido no servidor.

Um exemplo prático, em minha experiência desenvolvendo, especialmente em projetos de trabalho interdisciplinar, a arquitetura MVC é fundamental. Muitas vezes, no ambiente de desenvolvimento necessitamos reescrever códigos que foram implementados ignorando essa arquitetura, o que é um grande desafio e é extremamente complexo e trabalhoso. A separação entre os componentes facilita a compreensão do código e a implementação de novas funcionalidades, além de melhorar a organização das tarefas entre os membros da equipe. Em um contexto acadêmico, essa arquitetura permite que diferentes pessoas se concentrem em diferentes aspectos do sistema, tornando o desenvolvimento mais eficiente.

Os Microsserviços representam uma solução para problemas típicos encontrados em sistemas monolíticos. Em sistemas monolíticos, a interação entre os módulos ocorre no mesmo espaço de memória, o que pode causar falhas em cascata quando um módulo é alterado. A arquitetura de microsserviços propõe a divisão de um sistema em serviços independentes, o que facilita o desenvolvimento e a escalabilidade. Cada microsserviço pode ser escalado individualmente e pode ser desenvolvido com tecnologias diferentes, conforme a necessidade de cada módulo.

Além disso, os microsserviços oferecem maior resiliência a falhas, pois a falha de um microsserviço não afeta o sistema inteiro. Contudo, essa arquitetura também traz desafios, como a necessidade de gerenciar a comunicação entre serviços independentes e lidar com transações distribuídas. A gestão de dados também exige que cada microsserviço tenha seu próprio banco de dados, evitando gargalos e facilitando a evolução do sistema.

A Arquitetura baseada em Filas de Mensagens permite que sistemas se comuniquem de forma assíncrona e desacoplada, o que reduz a dependência entre os sistemas e melhora a eficiência geral. Por exemplo, um sistema de telecomunicações pode enviar uma mensagem para o sistema de engenharia ativar um serviço, sem a necessidade de esperar pela resposta imediata.

Já as arquiteturas Publish/Subscribe são baseadas em eventos, onde o publicador gera eventos que são distribuídos aos assinantes, de forma assíncrona e sem necessidade de interação direta. Um exemplo prático seria a integração dos sistemas de uma companhia aérea, onde a venda de uma passagem gera eventos que são recebidos por outros sistemas, como marketing e contabilidade, permitindo um alto grau de flexibilidade e desacoplamento. Outros padrões arquiteturais incluem Pipes e Filtros, onde dados são processados por filtros e passados por pipes, permitindo maior flexibilidade e execução paralela. Cliente/Servidor, que

organiza os sistemas em um modelo onde os clientes solicitam serviços ao servidor. Exemplos disso incluem sistemas de impressão e bancos de dados. Peer-to-Peer (P2P), onde cada módulo pode atuar tanto como cliente quanto como servidor, como no protocolo BitTorrent para compartilhamento de arquivos.

Um exemplo de anti-padrão é o Big Ball of Mud, no qual a falta de uma arquitetura bem definida resulta em sistemas desorganizados, difíceis de manter e evoluir. Em um sistema bancário, por exemplo, a ausência de uma arquitetura estruturada levou a um crescimento descontrolado e à complexidade, dificultando a manutenção, mesmo com práticas como documentação e revisões de código.

Em conclusão, a escolha de uma arquitetura de software é uma decisão crítica que influencia diretamente o desempenho, a escalabilidade, a manutenção e a evolução de um sistema. A análise dos diferentes padrões arquiteturais, como a Arquitetura em Camadas, MVC e Microsserviços, revela que cada abordagem possui vantagens e desafios específicos, dependendo das necessidades do sistema. Enquanto a Arquitetura em Três Camadas e o padrão MVC são mais adequados para sistemas com interfaces gráficas ou corporativos, os Microsserviços oferecem flexibilidade e resiliência em sistemas complexos e distribuídos. Por fim, é essencial considerar a complexidade do sistema e o contexto em que será aplicado para escolher a arquitetura mais apropriada, evitando, ao mesmo tempo, cair em anti-padrões como o Big Ball of Mud, que pode comprometer a eficiência e a sustentabilidade do projeto a longo prazo.