CAPÍTULO 7 - ARQUITETURA

O sétimo capítulo, intitulado Arquitetura, inicia explicando o conceito de arquitetura de software, destacando que esta se refere a um "projeto em nível mais elevado". Ou seja, o foco não está na organização ou interfaces de classes individuais, mas sim em estruturas maiores. Além disso, o capítulo enfatiza que os componentes arquiteturais não são apenas maiores, mas também fundamentais para que o sistema cumpra seus propósitos.

Existe ainda uma segunda definição para arquitetura de software, que a compreende como um conjunto das decisões mais relevantes sobre o projeto de um sistema. Dessa forma, essa visão é mais abrangente do que a primeira, pois considera a arquitetura não apenas como um agrupamento de módulos, mas sim como um conjunto de escolhas estratégicas.

Os padrões arquiteturais estabelecem uma estrutura de alto nível para sistemas de software, definindo seus principais módulos e as relações entre eles. Alguns dos principais padrões são:

- **Arquitetura em camadas**: Um dos padrões mais utilizados, no qual as classes são organizadas em módulos maiores chamados de camadas, dispostas de maneira hierárquica.

- **Arquitetura em três camadas**: Comum em sistemas corporativos, divide-se em:
    - Interface com o usuário: responsável pela interação com o usuário.
    - Lógica de negócio: implementa as regras do sistema.
    - Banco de dados: armazena as informações manipuladas.

- **Arquitetura MVC (Model-View-Controller)**: Usada em sistemas orientados a objetos, separa as classes em:
    - Visão: cuida da apresentação da interface gráfica.
    - Controladoras: processam e interpretam eventos dos dispositivos de entrada.
    - Modelo: mantém os dados e regras de domínio.

- **Microsserviços**: Estrutura onde grupos de módulos funcionam como processos independentes, minimizando impactos de alterações. Entre suas vantagens estão a evolução modular e independente, possibilitando trabalho simultâneo entre equipes. Porém, exige maior domínio de protocolos de comunicação como HTTP/REST.

- **Arquiteturas orientadas a mensagens**: Modelo em que clientes e servidores se comunicam via um serviço intermediário que mantém uma fila de mensagens. Nesse sistema, os clientes inserem informações na fila e os servidores as processam posteriormente. Essa abordagem proporciona:
    - **Desacoplamento espacial**: clientes e servidores não precisam se conhecer.
    - **Desacoplamento temporal**: não há necessidade de disponibilidade simultânea para comunicação.

- **Arquiteturas publish/subscribe**: Nesse modelo, as mensagens são tratadas como eventos. Os componentes assumem o papel de publicadores (publishers) e assinantes (subscribers). Essa abordagem compartilha algumas características com filas de mensagens, mas apresenta diferenças:
    - No publish/subscribe, eventos notificam múltiplos assinantes, criando uma comunicação 1 para n. Já em filas de mensagens, cada item é consumido por um único servidor (1 para 1).
    - No publish/subscribe, os assinantes recebem notificações automaticamente quando ocorre um evento, enquanto em filas de mensagens os servidores precisam consultar manualmente a fila.

- **Outros padrões arquiteturais**:
    - **Pipes e filtros**: Modelo baseado no fluxo de dados, onde programas (filtros) processam informações e as repassam através de pipes, que atuam como buffers.
    - **Cliente/Servidor**: Estrutura amplamente usada para serviços de rede, com clientes solicitando e servidores respondendo às requisições.

- **Arquitetura peer-to-peer (P2P)**: Módulos que podem atuar tanto como clientes quanto como servidores simultaneamente, descentralizando o processamento e compartilhamento de recursos.

O capítulo se encerra abordando os **anti-padrões arquiteturais**, que representam abordagens inadequadas na organização de sistemas. Um exemplo comum é o **big ball of mud**, caracterizado por uma estrutura caótica onde qualquer módulo pode se comunicar diretamente com qualquer outro, resultando em um grande número de dependências desorganizadas.