09 - Fundamentos

2025-12-12 10:12 am

Padrões podem ter subpadrões contidos em padrões de arquiteturas maiores.

Padrões Fundamentais

Grande Bola de Lama

- Antipadrão
- Sem organização e hierarquia
- Crescem sem ordem, às vezes começa assim como pequeno.
- Deve ser evitado a todo custo.
- Difícil de prever efeito de mudanças

Arquitetura Unitária

- Software e hardware eram uma coisa só no início
- Não havia rede de computadores para distribuir tarefas.
- Raros hoje em dia, exceto em sistemas embarcados e afins.

Cient/Servidor

- Separação em dois níveis: cliente e servidor, backend e front-end
- Tem muitas variações:
    - **Desktop + servidor de banco de dados**
        - Apresentação no desktop, computação intensiva no servidor
    - **Browser + Web Server**
        - Separação de responsabilidades também
    -

Três Níveis

- Popular nos anos 90.
- Database, Aplicação e Frontend
- Protocolos CORBA e DCOM facilitaram arquiteturas distribuidas (análogo a TCP/IP)
- Motivou requisito de serialização de objetos em Java, inexistente em C++, para facilitar transferência na rede. Hoje ninguém usa isso e é um fardo de retrocompatibilidade.

Arquiteturas Monolíticas Vs Distribuídas

- Dois grandes tipos.

- Código em um único deploy ou espalhado.

Monolíticos	Distribuídos
Em Camadas	Baseado em Serviços
Pipeline	Event-driven
Microkernel	Baseado em espaço
	Orientada a serviços
	Microsserviços

- Distribuídos costumam ser mais escaláveis e performáticos mas há desvantagens.
- **Falácia #1 - A Rede é Confiável**
  - Rede melhorou com o tempo mas ainda é um ponto de falha
  - Comunicação entre dois serviços perfeitamente saudáveis pode falhar, requer cuidados.
- **Falácia #2 - Latência é Zero**
  - Conexão pela rede é sempre mais lenta do que comunicação entre componentes locais.
  - Qual a latência de uma requisição ida/volta para um RESTful? Necessário saber.
  - Latência pode ir se somando a cada chamada, se tornando alta.
- **Falácia #3 - A Banda é Infinita**
  - Tamanho das requisições podem se somar até ser massivas.
  - *Stamp coupling* - obrigatoriedade pelo sistema de enviar muitos dados quando na verdade só se quer um pedacinho.
    - Stamp Coupling pode ser evitado selecionando campos, usando GraphQL para desacoplar dados ou ter endpoints específicos
- **Falácia #4 - A Rede é Segura**
  - Sistemas distribuídos requerem mais cuidados com segurança.
  - Cada endpoint é um risco, inclusive entre serviços.
- **Falácia #5 - A Topologia Nunca Muda**
  - Rede vai mudando com o tempo.
  - Upgrades no software de rede e ajustes podem desencadear problemas e requerer mudanças.
- **Falácia #6 - Só há um administrador**
  - Sempre há várias pessoas gerindo várias partes do projeto.
    - e.g. administrador de redes.
  - Requer coordenação que aplicações monolíticas não precisam.
- **Falácia #7 - Custo de Transporte é Zero**
  - Arquitetura distribuída custa mais que monolítica em geral.
  - Requer mais elementos de rede e hardware.
- **Falácia #8 - A Rede é Homogênea**

    - Hardwares de rede são diferentes e podem ter especificações diferentes dentro de si, com equipamento de marcas diferentes.

- Arquitetura distribuída também dificulta o logging e descobrir a causa de problemas.
    - Há ferramentas que consolidam diversas fontes, mas não resolvem tudo.
- Consistência de dados é mais difícil em sistemas distribuídos, mais difícil garantir que os mesmos dados estão presentes a todo momento em todas as partes do sistema (*eventual consistency*)
    - ACID vs BASE
- Manutenção de contratos entre cliente e serviço também é um ponto de dor em sistemas distribuídos. Também complexidade em organizar deprecação de versões.