ANOTAÇÕES SOBRE PROJETOS DE ARQUITETURAS

NICOLAS MATHEUS FERREIRA DE JESUS

1. Introdução

A arquitetura de software é o conjunto de decisões estruturais fundamentais que definem como um sistema será organizado, seus componentes principais e a forma como esses componentes interagem. Ela tem impacto direto na manutenção, escalabilidade, desempenho e segurança do sistema.
Entre as diversas arquiteturas existentes, destacam-se **Microkernel**, **Modular**, **Pipes and Filters**, **Arquitetura em Camadas**, **Cliente-Servidor**, **Microservices** e **Event-Driven**, cada uma com características e aplicações específicas.

2. Arquiteturas de Software

2.1. Arquitetura Microkernel

A arquitetura **Microkernel** (ou *Plug-in Architecture*) é composta por um **núcleo mínimo** responsável pelas funções básicas do sistema (como comunicação, gerenciamento de recursos e controle de ciclo de vida), enquanto as demais funcionalidades são implementadas como **módulos externos ou plug-ins**.

- **Componentes principais:**
    - **Microkernel (núcleo):** responsável pelas operações essenciais.
    - **Serviços internos:** fornecem funcionalidades básicas próximas ao sistema.
    - **Serviços externos (plug-ins):** implementam funcionalidades específicas.
- **Vantagens:**
    - Alta flexibilidade e extensibilidade.
    - Atualização e manutenção facilitadas sem impactar o núcleo.
    - Ideal para sistemas que precisam de customização (ex.: sistemas operacionais, IDEs).
- **Desvantagens:**
    - Comunicação entre módulos pode gerar overhead.

    - Complexidade na definição de interfaces entre o núcleo e os serviços.

- **Exemplo:** Sistemas operacionais como **Windows NT**, **MacOS X** e **Eclipse IDE**.

2.2. Arquitetura Modular

Na **arquitetura modular**, o sistema é dividido em **módulos independentes** que interagem entre si por meio de interfaces bem definidas. Cada módulo encapsula uma funcionalidade específica, promovendo **baixo acoplamento** e **alta coesão**.

- **Vantagens:**
    - Facilidade de manutenção e testes.
    - Possibilidade de reutilização de módulos em outros projetos.
    - Isolamento de falhas.
- **Desvantagens:**
    - Exige um bom planejamento de interfaces.
    - A comunicação entre módulos pode gerar dependências implícitas.
- **Exemplo:** Aplicações corporativas com módulos de vendas, estoque e financeiro.

2.3. Arquitetura Pipes and Filters

A arquitetura **Pipes and Filters** (ou *tubos e filtros*) é baseada em um conjunto de **filtros independentes** que processam dados em etapas, conectados por **pipes (canais)** que transportam os resultados de uma etapa para a seguinte.

- **Funcionamento:**
    - Cada filtro executa uma transformação sobre os dados recebidos.
    - Os pipes transmitem o fluxo de dados entre filtros.
- **Vantagens:**
    - Facilidade de composição e paralelização.
    - Reutilização de filtros.
    - Boa separação de responsabilidades.

- **Desvantagens:**
    - Dificuldade de lidar com dependências entre filtros.
    - Pode haver alto custo de comunicação entre etapas.
- **Exemplo:** Compiladores (análise léxica → sintática → semântica → geração de código), pipelines de dados e sistemas de processamento de imagem.

2.4. Arquitetura em Camadas (Layered Architecture)

A arquitetura em **camadas** organiza o sistema em **níveis hierárquicos**, onde cada camada oferece serviços à camada imediatamente superior e depende da inferior.

- **Camadas comuns:**
    - Apresentação
    - Aplicação
    - Negócio
    - Dados
- **Vantagens:**
    - Estrutura clara e de fácil manutenção.
    - Separação de responsabilidades.
    - Suporte a diferentes implementações de camadas.
- **Desvantagens:**
    - Possível queda de desempenho devido ao excesso de camadas.
    - Dificuldade em adaptar fluxos que não seguem a hierarquia.
- **Exemplo:** Sistemas web tradicionais (Front-end → API → Banco de Dados).

2.5. Arquitetura Cliente-Servidor

A arquitetura **Cliente-Servidor** divide o sistema em duas partes principais:

- **Cliente:** interface de interação com o usuário.

- **Servidor:** responsável pelo processamento e armazenamento dos dados.
- **Vantagens:**
    - Centralização de dados e segurança.
    - Escalabilidade horizontal (múltiplos clientes conectados).
- **Desvantagens:**
    - Dependência da disponibilidade do servidor.
    - Carga concentrada no servidor.
- **Exemplo:** Aplicações web, sistemas de e-mail, jogos online.

2.6. Arquitetura de Microservices

A arquitetura de **Microservices** (Microsserviços) é a evolução da modular, onde o sistema é dividido em **serviços pequenos e independentes**, cada um responsável por uma função de negócio específica e se comunicando por APIs.

- **Vantagens:**
    - Alta escalabilidade e resiliência.
    - Desenvolvimento e deploy independentes.
    - Melhor adaptação a times ágeis.
- **Desvantagens:**
    - Complexidade de orquestração e comunicação entre serviços.
    - Maior necessidade de monitoramento e infraestrutura.
- **Exemplo:** Netflix, Amazon e Spotify.

2.7. Arquitetura Orientada a Eventos (Event-Driven)

A arquitetura **Event-Driven** é centrada na **produção, detecção e consumo de eventos**. Os componentes reagem a eventos assíncronos, promovendo um alto grau de desacoplamento.

- **Componentes principais:**

    - **Produtores de eventos**
    - **Canal de eventos (bus/message broker)**
    - **Consumidores de eventos**
- **Vantagens:**
    - Alta escalabilidade e flexibilidade.
    - Processamento assíncrono eficiente.
    - Boa integração com sistemas distribuídos.
- **Desvantagens:**
    - Dificuldade de rastreamento e depuração.
    - Maior complexidade arquitetural.
- **Exemplo:** Sistemas de mensageria como Kafka, RabbitMQ, AWS Lambda.