Fundamentos da Arquitetura de Software.

Discentes: Arthur Nucada Félix de Souza
Lucas Gabriel Nunes Alves
José Alves de Oliveira Neto
Sophia Fernandes Magalhães Almeida
Victor Martins Vieira

UFG

O que é Arquitetura de Software?

- **Definição:** Estrutura fundamental que organiza os componentes de um sistema, seus relacionamentos, propriedades e princípios que guiam seu design e evolução.
- **Importância:** Define como o software será construído, mantido e escalado ao longo do tempo.
- **Objetivo:** Criar sistemas que atendam aos requisitos funcionais e não-funcionais de forma eficiente e sustentável.

Porque é Fundamental?

- Proporciona uma visão clara do sistema como um todo;
- Facilita a comunicação entre as equipes de desenvolvimento, pois estabelece uma linguagem comum;
- Estabelece diretrizes para implementação e manutenção;
- Fornece critérios claros para tomada de decisões e resolução de conflitos técnicos;
- Antecipa potenciais problemas e permite a implementação de estratégias preventivas.

UFG

Fundamentos da Arquitetura de Software

Oito Fundamentos Essenciais

Modularidade:

- Divisão do sistema em componentes independentes e reutilizáveis;
- Permite desenvolvimento paralelo e facilita manutenção;
- Exemplos: Linux, Spring.

Escalabilidade:

- Capacidade do sistema crescer sem comprometer o desempenho;
- Tipos: horizontal (mais servidores) e vertical (mais recursos);
- Exemplos: Netflix, Amazon Web Services, Uber.

Manutenibilidade:

- Facilidade de corrigir falhas e implementar melhorias;
- Código limpo, documentação completa e testes automatizados;
- Reduz custos de operação e prolonga a vida útil do software.

UFG

Fundamentos da Arquitetura de Software

Oito Fundamentos Essenciais

Segurança:

- Protege dados sensíveis, garante conformidade com regulamentações e preserva a reputação da organização;
- Controle de acesso, criptografia de dados, tratamento de vulnerabilidades conhecidas, auditoria e monitoramento de atividades.

Desempenho:

- Capacidade do sistema de responder rapidamente e de forma eficiente às solicitações dos usuários;
- Alta disponibilidade, escalabilidade, uso eficiente de recursos;
- Melhora a experiência do usuário, reduz custos operacionais, sustenta o crescimento do sistema.

Interoperabilidade:

- Habilidade do sistema de interagir e funcionar corretamente com outros sistemas;
- Adoção de padrões abertos, integração com softwares de terceiros, Suporte a diferentes formatos;
- Amplia a vida útil e a utilidade do sistema, reduz custos de migração e manutenção.

UFG

Fundamentos da Arquitetura de Software

Oito Fundamentos Essenciais

Resiliência:

Capacidade de um sistema de se recuperar rapidamente de falhas, manter seu funcionamento e continuar operando mesmo diante de problemas.

- **Tolerância a falhas:** Se uma parte falhar (ex: banco de dados ficar fora), o sistema consegue continuar ou se recuperar automaticamente.
- **Detecção de falhas:** O sistema identifica rapidamente problemas internos (ex: erros de comunicação, travamentos).

Portabilidade:

Capacidade de um software de ser transferido e executado em diferentes ambientes, sistemas operacionais, ou plataformas com pouca ou nenhuma modificação.

- **Independência de plataforma**: Não depende fortemente de recursos específicos de um sistema (ex: funciona no Windows, Linux, Mac).
- **Configurações flexíveis:** Parâmetros como caminhos de arquivos, bancos de dados e redes são configuráveis, não fixos no código.

UFG

Estilos Arquiteturais

Arquitetura monolítica

Todos os componentes da aplicação são integrados em um único código-fonte e executados juntos, ou seja, todas as funcionalidades são parte do mesmo pacote, que é distribuído e executado em uma única aplicação, em uma única máquina.

Benefícios

- Facilidade de implementação;
- Facilidade de Deploy.

Aspectos negativos

- Dificuldade de escalar (não é possível replicar partes e funcionalidades específicas, e a escalabilidade ocorre de forma vertical, sendo mais onerosa);
- Possível dificuldade de manutenção (todo o código da aplicação é concentrado em um único pacote, havendo um alto acoplamento dos componentes)

UFG

Estilos Arquiteturais

Arquitetura em camadas

Nesse estilo arquitetural, o software é dividido em diferentes camadas, onde cada uma possui funções e responsabilidades específicas.

Benefícios

- Manutenção (a divisão facilita a identificação e correção de problemas);
- Escalabilidade (por ser modular, é possível escalar as camadas individualmente);
- Reutilização (o isolamento das camadas facilita a reutilização do código).

Aspectos negativos

- Utilização de recursos (a comunicação necessária entre as diferentes camadas demanda recursos de rede e processamento da infraestrutura);
- Tempo de desenvolvimento (a divisão do sistema em diferentes camadas demanda mais esforço e, consequentemente, tempo de desenvolvimento).

UFG

Estilos Arquiteturais

Cliente-Servidor

Nesse estilo arquitetural, o software é dividido em duas partes, de um lado, o cliente, que proporciona a interface pela qual o usuário realiza solicitações e, do outro, o servidor, que as atende, realizando o processamento e armazenamento dos dados.

Benefícios

- Centralização (a centralização do processamento e armazenamento no servidor facilita manutenções e atualizações);
- Escalabilidade (a arquitetura pode escalar para lidar com aumentos de requisições);
- Distribuição (a aplicação pode ser distribuída, melhorando o desempenho e resiliência).

Aspectos negativos

- Utilização de recursos e custo (é necessário existir hardware para ambas as partes);
- Dependência de rede (problemas na infraestrutura de rede podem afetar a aplicação).

UFG

Estilos Arquiteturais

Arquitetura Orientada a Serviços (SOA)

Arquitetura Orientada a Serviços (SOA): Baseada em serviços independentes que se comunicam entre si, facilitando a reutilização e integração de sistemas.

Benefícios

- Reutilização de Serviços (serviços podem ser usados em diferentes aplicações, evitando retrabalho e acelerando o desenvolvimento);
- Maior Flexibilidade e Agilidade (É fácil modificar, substituir ou atualizar serviços sem impactar todo o sistema);
- Facilidade de Integração (sistemas diferentes, mesmo com tecnologias distintas, podem se comunicar via serviços padronizados).

Aspectos negativos

- Complexidade Adicional (Implementar e gerenciar um ambiente SOA exige mais planejamento e conhecimento técnico).

UFG

Estilos Arquiteturais

Arquitetura de Microserviços

Divide o sistema em pequenos serviços independentes, permitindo escalabilidade e manutenção mais eficiente.

Benefícios

- Escalabilidade independente (Cada serviço pode ser escalado separadamente, otimizando o uso de recursos);
- Desenvolvimento e implantação ágeis (Equipes podem trabalhar de forma autônoma em diferentes serviços, acelerando entregas);
- Manutenção e evolução facilitadas (Atualizações e melhorias podem ser feitas em um serviço sem afetar o sistema todo).

Aspectos negativos

- Complexidade de gerenciamento (Com mais serviços existem mais pontos de falha, mais operações para monitorar e manter).

UFG

Estilos Arquiteturais

Arquitetura Event Driven

Baseada em eventos, onde componentes reagem a mudanças no sistema, garantindo flexibilidade e eficiência.

Benefícios

- Alta escalabilidade (Cada componente pode escalar independentemente conforme a demanda de eventos aumenta);
- Desacoplamento dos sistemas (Os produtores de eventos não precisam conhecer os consumidores. Isso facilita muito a manutenção e evolução do sistema);

Aspectos negativos

- Sobrecarga de infraestrutura (É necessário cuidar de brokers de mensagens (tipo Kafka, RabbitMQ, etc.), o que gera uma camada extra de operação e monitoramento);
- Debugging difícil (A origem e o caminho de um problema podem ser difíceis de identificar num ambiente altamente assíncrono).

UFG

Estilos Arquiteturais

Arquitetura Serverless

Utiliza serviços em nuvem para executar funções sob demanda, eliminando a necessidade de gerenciamento de servidores.

Benefícios

- Redução de custos operacionais (paga-se apenas pelo tempo de execução das funções, não por servidores ociosos);
- Escalabilidade automática (o provedor gerencia o escalonamento conforme a demanda);
- Foco no desenvolvimento (desenvolvedores podem concentrar-se no código, não na infraestrutura).

Aspectos negativos

- Dependência de provedores de nuvem (vendor lock-in pode dificultar migrações futuras);
- Limitações de configuração (restrições de tempo de execução, tamanho de pacotes, etc.);
- Cold start (primeira execução pode ter latência mais alta);
- Debugging complexo (ambiente difícil de replicar localmente).

UFG

Obrigado pela atenção! :)

UFG