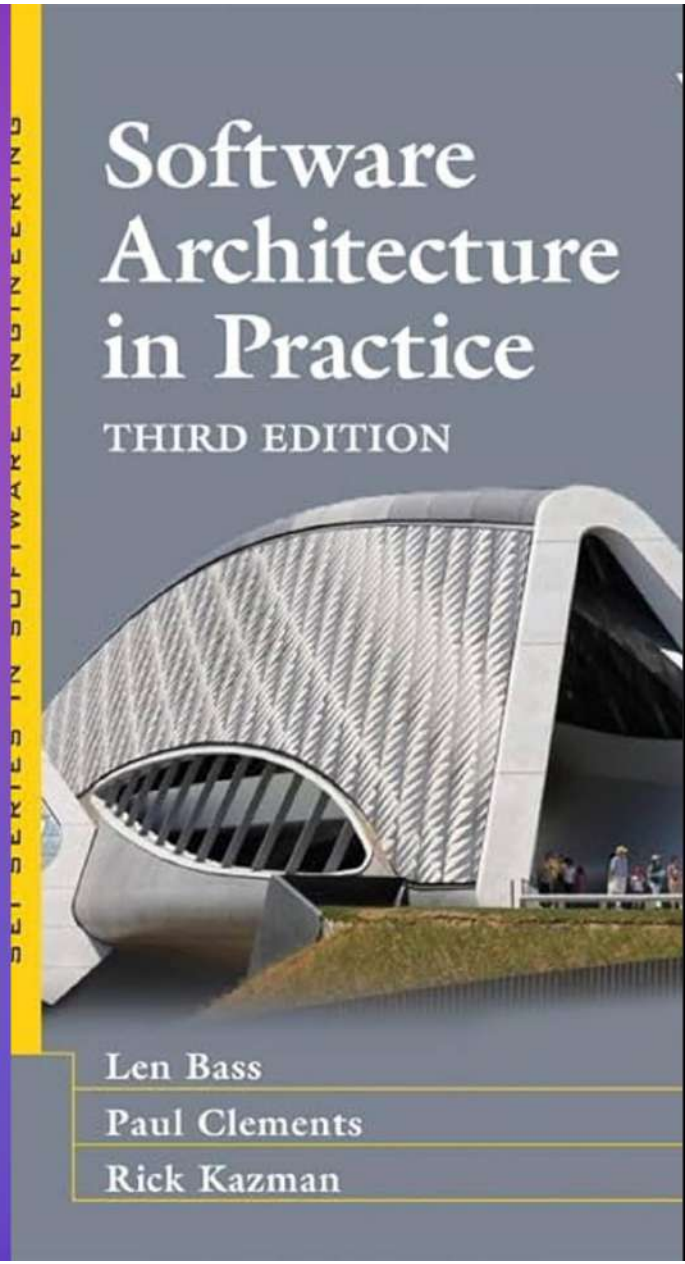
Arquitetura de Software: Princípios e Práticas Essenciais

Felipe Moreira, Frederico Garcez, Joseppe Fellini, Mauro Sérgio.



Índice

Introdução à Arquitetura de Software

Princípios Fundamentais

A modularidade divide o sistema em partes gerenciáveis.

Objetivos da Arquitetura

Dividir um sistema em módulos para melhor gerenciamento.

Impacto no Desenvolvimento

Melhorar a eficiência e a segurança dos sistemas.

Fundamentos da Arquitetura de Software

Modularidade

Importância do design modular em sistemas de software.

Divisão de um sistema em módulos gerenciáveis.

Escalabilidade

Capacidade de um sistema lidar com crescimento.

Aumento da carga de trabalho ou demanda de usuários.

Manutenibilidade

Facilidade de manter e atualizar o software.

O código deve ser claro e de fácil modificação.

Segurança

Medidas para proteger o sistema contra ameaças.

Garantir a segurança das aplicações de software.

Desempenho

Eficiência e velocidade do sistema.

Importância da performance em sistemas de software.

Modularidade: Práticas e Benefícios

01

Separação de Responsabilidades

Divisão do sistema em componentes independentes.

Facilita a manutenção e a escalabilidade.

02

Encapsulamento

Protege dados e funcionalidades de acesso externo.

Promove a integridade do sistema.

03

Padrões de Projeto

Uso de padrões como MVC e Singleton.

Melhora a manutenibilidade e reduz a complexidade.

04

Bibliotecas Reutilizáveis

Criação de bibliotecas independentes.

Permite reaproveitamento em diferentes partes do sistema.

05

Interfaces Bem Definidas

Comunicações entre módulos por meio de APIs.

Facilita a integração e a interoperabilidade.

Escalabilidade: Estratégias e Exemplos

Escalabilidade Horizontal

Distribuir carga entre servidores.

Escalabilidade Vertical

Atualiza a infraestrutura de um servidor, aumentando sua robustez.

Uso de Cache

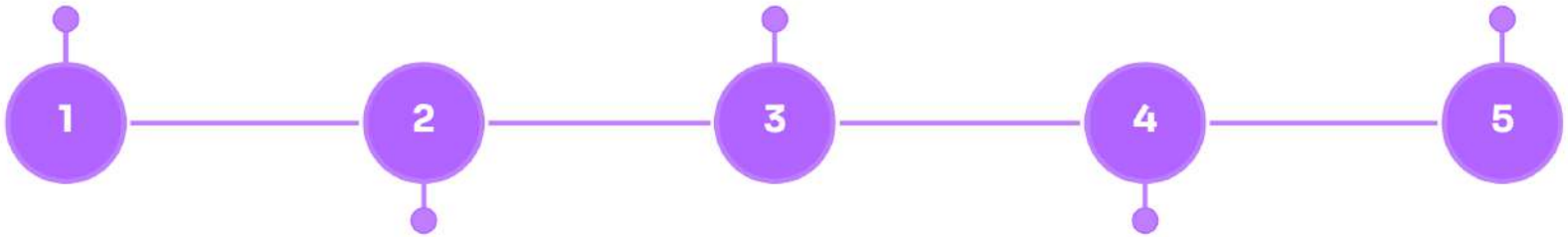
Reduzir carga sobre bancos de dados.

Balanceamento de Carga

Distribuir solicitações para evitar sobrecarga.

Computação em Nuvem

Aproveitar serviços como AWS para escalabilidade.



Manutenibilidade: Boas Práticas

Manutenibilidade

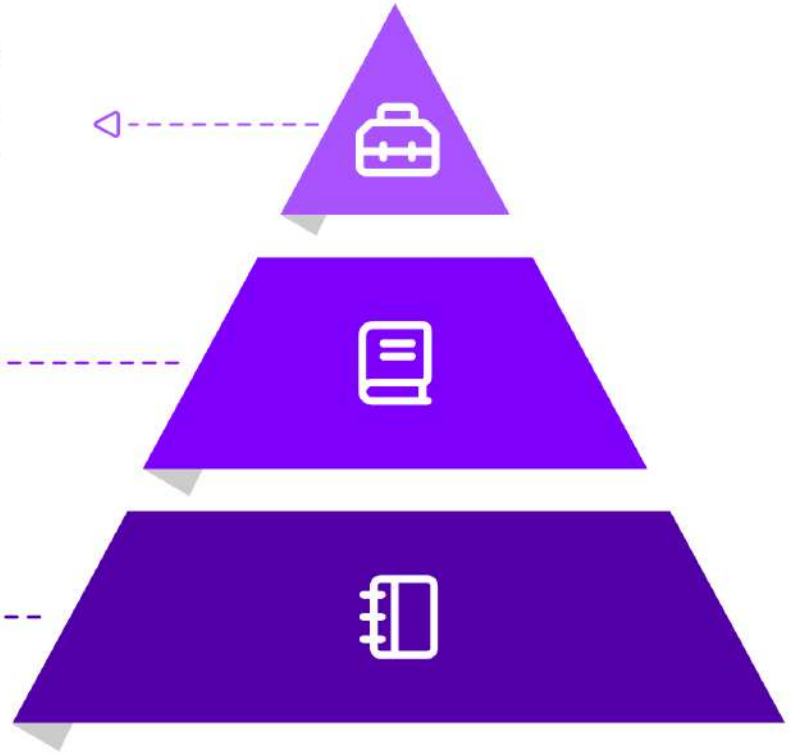
Garantida por boas práticas de desenvolvimento e documentação.

Código Limpo

Seguir boas práticas de programação.

Documentação Clara

Manter documentação sobre arquitetura e APIs.



Comparação: Arquitetura Monolítica vs Microserviços

Arquitetura Monolítica

- Simplicidade na implementação inicial.
- Facilidade de desenvolvimento.
- Menor complexidade em comparação com microserviços.
- Ideal para aplicações menores.
- Infraestrutura simplificada.

Microserviços

- Complexidade na comunicação entre serviços.
- Necessidade de gerenciamento eficiente.
- Alta escalabilidade.
- Maior exposição às inovações do mundo cloud.
- Baixo acoplamento.
- Maior sobrecarga de infraestrutura.

Estilos Arquiteturais: Visão Geral

Monolítica

Abordagem tradicional onde todos os componentes estão interconectados.

Desenvolvimento e implantação facilitados

Em Camadas

Organização do software em camadas.

Facilita a manutenção e escalabilidade.

Cliente-Servidor

Modelo que separa os papéis de cliente e servidor.

Permite computação distribuída.

Orientada a Serviços

Utiliza serviços como blocos fundamentais para comunicação em rede.

Facilita a interação entre serviços.

Microserviços

Estrutura aplicações como uma coleção de serviços desacoplados.

Permite desenvolvimento independente.

Interoperabilidade: Estratégias e Ferramentas

Uso de APIs

Permitem troca de dados padronizada e segura.

Integração Semântica

Ajudam os sistemas a entender o significado dos dados.

Padrões de Dados

Formatos garantem interpretação correta das informações.

Middleware

Conectam sistemas distintos sem grandes modificações.

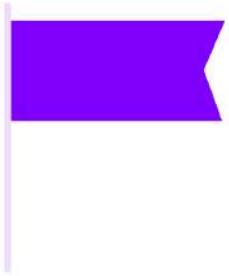
Resiliência: Garantindo Continuidade

Deploy Gradual

Blue-Green Deployment e Canary Releases testam novas versões sem comprometer a estabilidade do sistema.

Chaos Engineering

Ferramentas como Chaos Monkey simulam falhas para testar a resiliência do sistema em produção.



Backup e Recuperação

Estratégias de backup automático e recuperação rápida minimizam impactos de falhas críticas.

Portabilidade: Flexibilidade em Ambientes

Linguagens Multiplataforma

As linguagens multiplataforma permitem que o software funcione em diferentes sistemas operacionais sem reescrita de código.

Separação de Código

Manter o código fonte independente do sistema operacional.

APIs Padronizadas

Utilizar APIs que seguem padrões abertos, garantindo compatibilidade.

Testes em Ambientes

Realizar testes rigorosos em múltiplas plataformas.

Design Responsivo

Garantir que a interface se adapte a diferentes dispositivos.