Diferentes arquiteturas para diferentes problemas

Professor Jonathan M. Samara

O que é a Arquitetura Monolítica?

Modelo tradicional e amplamente adotado no desenvolvimento de software.

Todos os componentes (interface de usuário, lógica de negócios, acesso a dados) são combinados em uma única aplicação.

Simplicidade inicial e rapidez no desenvolvimento são prioridades, mas a integração total pode criar desafios com o crescimento do sistema.

Características da Arquitetura Monolítica

- **Unidade Integrada:**
  - Sistema como uma única unidade funcional.
  - Todos os componentes interligados dentro de um código-base.
- **Interdependência:**
  - Modificações em uma parte do código podem impactar outras partes.
  - Recompilação e redistribuição de toda a aplicação podem ser necessárias.

Vantagens da Arquitetura Monolítica

- **Simplicidade Inicial:**
  - Implementação e gerenciamento mais simples no início.
  - Menos sobrecarga em termos de configuração e orquestração.
- **Facilidade de Desenvolvimento:**
  - Desenvolvimento em um único ambiente facilita a colaboração e comunicação entre equipes.
- **Desempenho:**
  - Chamadas internas rápidas, sem sobrecarga de rede.

Desvantagens da Arquitetura Monolítica

- **Dificuldade de Escalabilidade:**
  - Toda a aplicação precisa ser escalada como um todo, mesmo que apenas uma parte esteja sobrecarregada.
- **Complexidade e Acoplamento:**
  - Código pode se tornar complexo e difícil de manter à medida que cresce.
  - Alto acoplamento pode dificultar a implementação de novas funcionalidades e melhorias.
- **Resistência a Mudanças:**
  - Alterações em uma parte podem ter efeitos colaterais em outras partes.
  - Atualizações e manutenção podem ser mais arriscadas e lentas.
- **Implantação e Atualização:**
  - Alterações requerem a implantação de toda a aplicação, podendo resultar em tempo de inatividade.

Exemplos e Uso

- **Aplicações Legadas:**
  - Aplicações e sistemas empresariais legados, como ERPs, ainda utilizam a arquitetura monolítica.
  - Simplicidade inicial e ferramentas maduras suportam esse estilo de arquitetura.
- **Migração para Arquiteturas Modulares:**
  - Tendência de migração para arquiteturas modulares, como microservices, para melhor escalabilidade e manutenção.

Reflexões

- **Adequação da Arquitetura Monolítica:**
  - Ainda válida para projetos menores ou quando a simplicidade e o tempo de desenvolvimento inicial são prioridades.
  - Para aplicações que precisam crescer rapidamente, considerar alternativas mais modulares pode ser benéfico.
- **Reflexão:**
  - Avaliar os desafios a longo prazo e a adequação da arquitetura monolítica em relação às necessidades de evolução do sistema.

Introdução à Arquitetura Multicamadas

Definição:

- Arquitetura que divide a aplicação em camadas distintas com responsabilidades específicas.
- Ideal para sistemas complexos e em larga escala.

Estrutura:

- Camada de Apresentação
- Camada de Negócios
- Camada de Dados

Introdução à Arquitetura Multicamadas

Camada de Apresentação (Presentation Layer):

- Interage com o usuário.
- Implementada com HTML, CSS, JavaScript, frameworks como Angular, React, ou Vue.js.

Camada de Negócios (Business Layer):

- Contém regras de negócio e lógica.
- Utiliza linguagens como Java, C#, Python, e frameworks como Spring, .NET, Django.

Camada de Dados (Data Layer):

- Armazena e recupera dados.
- Interage com bancos de dados relacionais (MySQL, PostgreSQL) ou NoSQL (MongoDB);

Vantagens da Arquitetura Multicamadas

Separação de Responsabilidades:

- Facilita desenvolvimento e manutenção.
- Permite modificações e substituições independentes.

Facilidade de Manutenção:

- Problemas confinados a uma camada.
- Implementação de novas funcionalidades simplificada.

Reutilização de Código:

- Camadas podem ser reutilizadas em diferentes aplicações.

Escalabilidade:

- Escala independente para cada camada.
- Ex.: Escalabilidade horizontal da camada de apresentação.

Desafios e Exemplos da Arquitetura Multicamadas

Desafios:

- Complexidade Inicial: Planejamento detalhado necessário.
- Desempenho: Latência e sobrecarga de comunicação entre camadas.
- Gerenciamento de Estado: Desafios na manutenção do estado entre camadas.

Exemplos:

- **Aplicações Web Corporativas:** Separação de interface, lógica de negócios e dados.
- **Sistemas Bancários:** Operações seguras e eficientes.
- **Aplicações ERP:** Módulos interagem com banco de dados central.

Conclusão:

- Arquitetura multicamadas é robusta para sistemas complexos, oferecendo boas práticas de organização e escalabilidade.

O que são Aplicações Distribuídas?

- Sistemas de software com componentes em diferentes dispositivos de rede.
- Comunicação entre componentes para alcançar um objetivo comum.
- Diferente da arquitetura monolítica, onde tudo está integrado em uma única aplicação.

Características das Aplicações Distribuídas

Distribuição de Componentes

- Lógica de negócios, camada de dados, e interface do usuário em diferentes servidores ou localizações geográficas.
- Comunicação através de redes com protocolos como HTTP, RPC, ou mensagens assíncronas.

Escalabilidade Horizontal

- Escalabilidade independente dos componentes.
- Adição de mais servidores ou instâncias sem replicar todo o sistema.

Características das Aplicações Distribuídas

Resiliência e Disponibilidade

- Maior resiliência com componentes distribuídos.
- Redundância para garantir alta disponibilidade mesmo com falhas.

Flexibilidade e Modularidade

- Desenvolvimento, implantação, e atualização independentes.
- Equipes podem trabalhar em partes diferentes simultaneamente.

Vantagens das Aplicações Distribuídas

Desempenho Melhorado

- Distribuição da carga de trabalho entre múltiplos servidores.
- Gerenciamento eficiente de grandes volumes de dados e tráfego.

Manutenção Facilitada

- Manutenção e atualizações em partes específicas sem interromper toda a aplicação.

Aproveitamento de Recursos

- Utilização de recursos em diferentes locais e data centers.
- Melhoria na latência e otimização do uso de recursos.

Vantagens das Aplicações Distribuídas

Complexidade de Implementação

- Comunicação entre componentes e gestão de estados distribuídos.
- Desafios na implementação e tratamento de falhas de rede.

Latência e Desempenho

- Latência adicional devido à comunicação via rede.
- Impacto no desempenho em sistemas de alta demanda.

Segurança

- Vulnerabilidades a ataques, como intercepção de dados ou DDoS.
- Necessidade de segurança em todas as camadas de comunicação.

O que é Arquitetura Cliente-Servidor?

- Modelo fundamental na computação.
- Dividido em duas entidades principais: cliente e servidor.
- Permite distribuição de tarefas e comunicação eficiente.

Estrutura da Arquitetura Cliente-Servidor

Cliente

- Inicia a comunicação solicitando serviços ou recursos.
- Pode ser um navegador web, aplicativo móvel, ou desktop.
- Interage com o usuário final e exibe resultados.

Servidor

- Responde às solicitações do cliente.
- Fornece serviços como acesso a dados, processamento, e cálculos.
- Geralmente máquinas poderosas ou clusters de servidores.

Funcionamento da Arquitetura Cliente-Servidor

1. **Solicitação (Request)**
   - Cliente envia uma solicitação ao servidor.
   - Exemplo: navegador web solicita uma página HTML.
2. **Processamento**
   - Servidor processa a solicitação, realiza ações necessárias.
   - Pode incluir consultas a banco de dados ou execução de scripts.
3. **Resposta (Response)**
   - Servidor envia a resposta de volta ao cliente.
   - Exemplo: servidor web envia o conteúdo HTML da página.
4. **Exibição e Interação**
   - Cliente exibe a resposta ao usuário final.
   - Permite novas interações como clicar em links ou enviar formulários.

Vantagens da Arquitetura Cliente-Servidor

Centralização dos Recursos

- Facilita administração, atualização, e segurança.
- Dados mantidos de forma consistente para múltiplos clientes.

Escalabilidade

- Servidores dimensionáveis para atender a muitos clientes.
- Ideal para serviços de streaming e e-commerce.

Vantagens da Arquitetura Cliente-Servidor

Manutenção Facilitada

- Atualizações centralizadas no servidor.
- Sem necessidade de atualizar todos os clientes.

Segurança

- Políticas de segurança implementadas centralmente.

Desafios da Arquitetura Cliente-Servidor

Dependência de Rede

- Problemas de conectividade afetam o acesso aos serviços.
- Dependência da rede para funcionamento contínuo.

Sobrecarga do Servidor

- Alto número de solicitações pode sobrecarregar o servidor.
- Necessidade de planejamento cuidadoso para escalabilidade.

Complexidade de Implementação

- Mais complexo que sistemas monolíticos.
- Desafios em autenticação, autorização, e gerenciamento de estado.

Exemplos de Aplicações Cliente-Servidor

Navegadores Web e Servidores Web

- Navegadores solicitam páginas web a servidores.

Aplicações de E-mail

- Clientes de e-mail se conectam a servidores para enviar e receber mensagens.

Aplicações Bancárias

- Aplicativos bancários comunicam-se com servidores para transações e consultas.

Arquitetura Orientada a Serviços (SOA)

Abordagem de Design de Software

- Componentes são organizados como serviços independentes e interoperáveis.
- Comunicação através de interfaces e protocolos de rede.

Principais Conceitos da SOA

Serviços

- Unidade funcional autônoma para tarefas específicas.
- Exemplo: verificação de saldo, processamento de pagamento.

Interoperabilidade

- Comunicação entre serviços, independente das tecnologias subjacentes.

Contratos de Serviço

- Define o que o serviço faz, como acessá-lo, e quais dados ele manipula.

Acoplamento Fraco

- Mudanças em um serviço não afetam outros.

Reutilização

- Serviços reutilizáveis em diferentes contextos.

Principais Conceitos da SOA

Serviços

- Unidade funcional autônoma para tarefas específicas.
- Exemplo: verificação de saldo, processamento de pagamento.

Interoperabilidade

- Comunicação entre serviços, independente das tecnologias subjacentes.

Contratos de Serviço

- Define o que o serviço faz, como acessá-lo, e quais dados ele manipula.

Acoplamento Fraco

- Mudanças em um serviço não afetam outros.

Reutilização

- Serviços reutilizáveis em diferentes contextos.

Vantagens da SOA

Flexibilidade e Agilidade

- Adaptação rápida a mudanças nos requisitos.

Integração Simplificada

- Integração com sistemas legados ou de terceiros.

Escalabilidade

- Serviços escaláveis de forma independente.

Redução de Custos

- Reutilização de serviços existentes reduz custos.

Facilidade de Manutenção

- Modularidade facilita manutenção.

Desafios da SOA

Complexidade de Implementação

- Gestão de serviços, segurança e desempenho.

Sobrecarregamento de Comunicação

- Latência e confiabilidade na comunicação entre serviços.

Governança de Serviços

- Necessidade de padrões, políticas de segurança e monitoramento.

Segurança

- Proteção de dados e serviços distribuídos.

Exemplos de Uso da SOA

Sistemas Bancários

- Integra serviços como processamento de pagamentos e verificação de saldo.

Plataformas de E-commerce

- Conecta serviços de inventário, pedidos e pagamentos.

Aplicações Empresariais

- Integra módulos de diferentes fornecedores em sistemas ERP

Diferença entre SOA e Sistemas Distribuídos

- **SOA (Arquitetura Orientada a Serviços):**
  - **Definição:** SOA é uma abordagem de design de software em que o sistema é dividido em serviços independentes e interoperáveis. Esses serviços são unidades funcionais autônomas que se comunicam através de interfaces bem definidas.
  - **Objetivo:** Facilitar a reutilização e integração de serviços, promover a modularidade e permitir a interoperabilidade entre diferentes sistemas e plataformas.
- **Sistemas Distribuídos:**
  - **Definição:** Sistemas Distribuídos são sistemas de software onde componentes e serviços estão localizados em diferentes dispositivos ou redes e se comunicam entre si para alcançar um objetivo comum.
  - **Objetivo:** Melhorar a escalabilidade, resiliência e flexibilidade, permitindo que diferentes partes do sistema sejam distribuídas por diferentes locais físicos e ainda funcionem como um todo coeso.

Diferença entre SOA e Sistemas Distribuídos

Estrutura e Organização

- **SOA:**
  - **Componentes:** Envolve a criação de serviços que são expostos e consumidos através de interfaces padrão (por exemplo, SOAP, REST). Cada serviço é responsável por uma tarefa específica e pode ser reutilizado por diferentes aplicações.
  - **Enfoque:** Foca na reutilização de serviços, na interoperabilidade entre sistemas e na criação de um contrato bem definido para cada serviço.
- **Sistemas Distribuídos:**
  - **Componentes:** Compreendem uma rede de máquinas ou nós que colaboram para realizar um trabalho. A comunicação entre os componentes é feita através de protocolos de rede.
  - **Enfoque:** Foca na distribuição de componentes e dados, na escalabilidade horizontal (adicionar mais servidores) e na resiliência do sistema em caso de falhas.

Diferença entre SOA e Sistemas Distribuídos

Comunicação e Interoperabilidade

- **SOA:**
  - **Comunicação:** Serviços se comunicam através de protocolos e padrões de comunicação bem definidos. A comunicação é muitas vezes feita via mensagens XML, JSON, SOAP ou REST.
  - **Interoperabilidade:** SOA enfatiza a interoperabilidade entre diferentes sistemas e plataformas, utilizando padrões abertos e contratos de serviço para garantir que serviços escritos em diferentes tecnologias possam se comunicar.
- **Sistemas Distribuídos:**
  - **Comunicação:** A comunicação pode ocorrer através de diversos protocolos de rede, como HTTP, RPC, ou mensagens assíncronas. A comunicação pode ser mais diversificada e menos padronizada comparada à SOA.
  - **Interoperabilidade:** Embora a interoperabilidade possa ser uma consideração, o foco principal está na comunicação eficiente e na coordenação entre os componentes distribuídos.

Diferença entre SOA e Sistemas Distribuídos

Flexibilidade e Evolução

- **SOA:**
  - **Flexibilidade:** Permite a adição, modificação e substituição de serviços com impacto mínimo sobre o sistema como um todo, facilitando a evolução e manutenção.
  - **Evolução:** Facilita a evolução do sistema ao permitir que novos serviços sejam adicionados sem necessidade de reescrever ou substituir serviços existentes.
- **Sistemas Distribuídos:**
  - **Flexibilidade:** Oferece flexibilidade na escolha de tecnologias e plataformas para diferentes componentes, mas pode ser mais complexo gerenciar e atualizar componentes distribuídos.
  - **Evolução:** A evolução pode envolver a atualização de múltiplos componentes distribuídos, o que pode ser mais complexo em comparação com a atualização de serviços em uma arquitetura SOA.

Diferença entre SOA e Sistemas Distribuídos

Desafios e Considerações

- **SOA:**
  - **Desafios:** Complexidade na governança de serviços, sobrecarga de comunicação, e necessidade de uma boa gestão de contratos e interfaces.
  - **Considerações:** Requer um planejamento cuidadoso da arquitetura de serviços e uma abordagem rigorosa para garantir a interoperabilidade e a segurança.
- **Sistemas Distribuídos:**
  - **Desafios:** Complexidade na comunicação entre componentes, latência de rede, gerenciamento de estados distribuídos e tolerância a falhas.
  - **Considerações:** Exige estratégias eficazes para garantir a comunicação eficiente, a consistência dos dados e a resiliência do sistema.
  -