CENTRO PAULA SOUZA
ETEC PROF. MARIA CRISTINA MEDEIROS
Técnico em Informática para a Internet Integrado Ao Ensino Médio

Enzo Caetano Peracio Rodrigues

CONCEITOS BÁSICOS DE SISTEMAS WEB

Ribeirão Pires
2025

Enzo Caetano Peracio Rodrigues

CONCEITOS BÁSICOS DE SISTEMAS WEB

Pesquisa Sobre Conceitos Básicos Apresentados na matéria SW-II apresentado ao Curso Técnico em Informática Para A Internet com Ensino Médio integrado ao tecnico da Etec Prof. Maria Cristina Medeiros, orientado pelo Prof. Anderson Vanin.

Ribeirão Pires 2025

“A internet não é uma rede de computadores, é uma rede de pessoas”

Conrado Adolpho

OBJETIVOS

Este estudo tem como objetivo analisar e comparar as abordagens de desenvolvimento de aplicações web baseadas em sistemas distribuídos, explorando as arquiteturas monolítica e de microsserviços. Busca-se compreender as vantagens e desafios de cada modelo, considerando aspectos como escalabilidade, desempenho, manutenção e complexidade. Além disso, a pesquisa visa identificar cenários ideais para a adoção de cada arquitetura, fornecendo diretrizes para a escolha adequada conforme as necessidades do projeto e infraestrutura disponível.

SUMÁRIO

1 Introdução

Nos últimos anos, o desenvolvimento de aplicações web tem evoluído significativamente, impulsionado pela crescente demanda por sistemas escaláveis, eficientes e de fácil manutenção. Nesse contexto, as arquiteturas de software desempenham um papel crucial na definição da estrutura e comportamento dessas aplicações. Duas abordagens amplamente discutidas são a arquitetura monolítica e a de microsserviços.

A arquitetura monolítica, tradicionalmente utilizada, caracteriza-se por consolidar todos os componentes e funcionalidades da aplicação em um único bloco coeso. Embora essa abordagem possa simplificar o desenvolvimento inicial, ela frequentemente enfrenta desafios relacionados à escalabilidade e à manutenção à medida que o sistema cresce em complexidade.

Por outro lado, a arquitetura de microsserviços propõe a divisão da aplicação em pequenos serviços independentes, cada um responsável por uma funcionalidade específica. Essa modularidade facilita a escalabilidade e a atualização de componentes individuais, mas também introduz desafios, especialmente no que tange à comunicação entre serviços e à gestão de sistemas distribuídos.

Diante disso, este estudo busca analisar e comparar essas duas abordagens arquiteturais no contexto de aplicações web e sistemas distribuídos. Serão exploradas as vantagens e desvantagens de cada modelo, considerando aspectos como desempenho, escalabilidade, manutenção e complexidade. Além disso, pretende-se identificar cenários nos quais uma arquitetura pode ser mais adequada que a outra, fornecendo diretrizes para a escolha informada conforme as necessidades específicas de projetos e infraestruturas.

1.1 O QUE SÃO APLICAÇÕES WEB?

Aplicações web são programas de software acessíveis e executados por meio de navegadores de internet, eliminando a necessidade de instalação local nos dispositivos dos usuários. Essas aplicações operam em um modelo cliente-servidor, onde o cliente é o navegador que interage com a interface do usuário, e o servidor processa as solicitações e gerencia os dados da aplicação. Desenvolvidas com tecnologias como HTML, CSS e Javascript, as aplicações web oferecem uma variedade de funcionalidades, desde serviços de e-mail e redes sociais até plataformas de comércio eletrônico e sistemas corporativos complexos. Sua principal característica é a acessibilidade, permitindo que usuários acessem os serviços de qualquer lugar com conexão à internet, sem a necessidade de instalações adicionais.

1.1.1 COMO FUNCIONA

As aplicações web operam em uma arquitetura cliente-servidor, onde o cliente é o navegador do usuário e o servidor hospeda a aplicação. Quando um usuário interage com a aplicação através do navegador, diversas etapas ocorrem para processar essa interação:

1. **Solicitação do Cliente**: O usuário insere uma URL ou interage com um elemento da página (como clicar em um botão), e o navegador envia uma requisição HTTP ou HTTPS ao servidor correspondente.
2. **Processamento no Servidor**: O servidor recebe a requisição e a processa. Dependendo da natureza da solicitação, o servidor pode consultar um banco de dados, executar cálculos ou realizar outras operações necessárias para gerar uma resposta adequada.
3. **Resposta do Servidor**: Após o processamento, o servidor envia uma resposta de volta ao navegador. Essa resposta geralmente inclui código HTML, CSS e JavaScript, que determinam a estrutura, o estilo e o comportamento da página exibida ao usuário.

1.1.2 DIFERENÇAS ENTRE WEB APP E SITE

Embora os termos "aplicação web" e "site" sejam frequentemente utilizados de forma intercambiável, eles possuem diferenças significativas em termos de funcionalidade, interação do usuário e propósito.

Site: Geralmente composto por páginas estáticas, um site tem como objetivo principal fornecer informações aos visitantes. A interação do usuário é limitada à navegação e leitura do conteúdo disponível. Exemplos incluem blogs, portfólios online e sites institucionais.

Aplicação Web: Trata-se de um software acessível via navegador oferecem funcionalidades interativas, permitindo ao usuário realizar tarefas específicas, como preencher formulários, gerenciar dados ou efetuar transações. Exemplos comuns são plataformas de e-commerce, sistemas de gerenciamento de conteúdo e serviços bancários online.

Site: Normalmente, não requer autenticação; o conteúdo é acessível a todos os visitantes de forma igual.

Aplicação Web: Frequentemente exige que o usuário se autentique, fornecendo acesso a dados personalizados e funcionalidades específicas, como em plataformas de e-mail ou redes sociais.

Site: Possui estrutura mais simples e é desenvolvido com foco na apresentação de informações, utilizando principalmente HTML, CSS e, ocasionalmente, Javascript para interatividade básica.

Aplicação Web: Envolve maior complexidade, integrando linguagens de programação no servidor, como PHP, Python ou Java, e frequentemente interage com bancos de dados para fornecer funcionalidades dinâmicas e interativas.

1.2 O QUE SÃO SISTEMAS DISTRIBUÍDOS?

Sistemas distribuídos são conjuntos de computadores autônomos que se comunicam e coordenam suas ações por meio de troca de mensagens através de uma rede, com o objetivo de alcançar um propósito comum. Nesses sistemas, os componentes interagem de maneira transparente, proporcionando aos usuários a impressão de estarem lidando com um único sistema coeso.

A principal característica dos sistemas distribuídos é a distribuição geográfica de seus componentes, permitindo que recursos e serviços sejam compartilhados e acessados de diferentes locais. Isso resulta em benefícios como escalabilidade, onde o sistema pode crescer conforme a demanda; tolerância a falhas, garantindo que a falha de um componente não comprometa todo o sistema; e flexibilidade, facilitando a adaptação a diferentes cargas de trabalho e requisitos.

Entretanto, projetar e gerenciar sistemas distribuídos apresenta desafios significativos, incluindo a necessidade de sincronização entre componentes, gerenciamento de concorrência, segurança na comunicação e manutenção da consistência dos dados. Abordar esses desafios de maneira eficaz é crucial para o sucesso e a eficiência de sistemas distribuídos em diversos contextos, como aplicações web, serviços financeiros e plataformas de comércio eletrônico.

1.2.1 ANATOMIA DE UM SISTEMA DISTRIBUÍDO

A anatomia de um sistema distribuído compreende diversos componentes e características que, em conjunto, permitem a operação eficiente e coordenada de múltiplos nós em uma rede. Esses elementos são essenciais para garantir a comunicação, a sincronização e a execução adequada das tarefas distribuídas. A seguir, destacam-se os principais aspectos:

Nós (Nodes): São as unidades de processamento que executam tarefas específicas dentro do sistema. Cada nó pode operar de forma autônoma, mas colabora com os demais para atingir objetivos comuns.

Rede de Comunicação: Infraestrutura que interconecta os nós, permitindo a troca de mensagens e dados. A eficiência e a confiabilidade da rede são cruciais para o desempenho do sistema distribuído.

Middleware: Camada de software que facilita a comunicação e a gestão de dados entre os nós, abstraindo a complexidade da rede e proporcionando uma interface uniforme para os desenvolvedores.

Transparência: Esconde dos usuários e desenvolvedores a complexidade inerente à distribuição dos componentes, oferecendo uma visão unificada do sistema.

Escalabilidade: Capacidade de expandir o sistema de forma eficiente, adicionando novos nós sem degradação significativa do desempenho.

Tolerância a Falhas: Habilidade de continuar operando corretamente mesmo quando alguns componentes falham, garantindo a disponibilidade e a confiabilidade do sistema.

Concorrência: Possibilidade de múltiplos processos ou threads serem executados simultaneamente, compartilhando recursos de forma coordenada.

1.3 O QUE É ARQUITETURA MONOLÍTICA?

A arquitetura monolítica é um estilo de desenvolvimento de software em que todos os componentes e funcionalidades de uma aplicação são integrados em uma única unidade coesa. Nesse modelo, a interface do usuário, a lógica de negócios e o acesso a dados estão interligados e operam como um único programa. Devido a essa estrutura unificada, qualquer alteração ou atualização no sistema requer a recompilação e a implantação de toda a aplicação. Embora a arquitetura monolítica possa simplificar o desenvolvimento inicial e a implantação de aplicações menores, ela pode apresentar desafios de escalabilidade e manutenção à medida que o sistema cresce em complexidade.

1.3.1 VANTAGENS E DESVANTAGENS

Simplicidade de Desenvolvimento e Implantação: Devido à sua estrutura unificada, o desenvolvimento e a implantação de aplicações monolíticas são mais diretos, facilitando o gerenciamento inicial do projeto.

Desempenho Eficiente: A comunicação interna entre os componentes é direta, resultando em menor latência e melhor desempenho geral da aplicação.

Facilidade de Testes e Depuração: Com todos os módulos integrados em uma única base de código, os processos de teste e depuração tornam-se mais simples, permitindo identificar e corrigir erros de forma mais eficiente.

Dificuldade de Escalabilidade: Escalar seletivamente partes específicas da aplicação é desafiador, pois qualquer alteração requer a implantação de todo o sistema, o que pode ser ineficiente em termos de recursos.

Manutenção Complexa: À medida que a aplicação cresce, a base de código pode se tornar extensa e intrincada, dificultando a implementação de novas funcionalidades e a correção de bugs sem impactar outras áreas do sistema.

Falta de Flexibilidade Tecnológica: A adoção de novas tecnologias ou frameworks é limitada, pois mudanças na estrutura ou na linguagem afetam todo o sistema, tornando o processo de atualização complexo e custoso

1.4 O QUE SÃO MICROSSERVIÇOS?

Os microsserviços constituem um estilo arquitetural de desenvolvimento de software em que uma aplicação é segmentada em um conjunto de serviços pequenos e independentes. Cada um desses serviços é projetado para executar uma única função ou capacidade de negócio e opera de forma autônoma, comunicando-se com os demais por meio de interfaces bem definidas, geralmente utilizando APIs leves.

Além disso, os microsserviços são frequentemente organizados em torno de capacidades de negócio específicas e podem ser gerenciados por equipes pequenas e multifuncionais. Essa abordagem promove a autonomia das equipes e possibilita a utilização de diferentes tecnologias ou linguagens de programação para cada serviço, conforme as necessidades específicas, sem comprometer o funcionamento geral da aplicação.

1.4.1 DIFERENÇAS ENTRE ARQUITETURA MONOLÍTICA E MICROSSERVIÇOS

Arquitetura Monolítica: Consiste em uma única base de código onde todos os componentes e funcionalidades da aplicação estão interligados e dependentes entre si. Isso significa que a interface do usuário, a lógica de negócios e o acesso a dados são desenvolvidos e implantados como uma única unidade.

Arquitetura de Microsserviços: A aplicação é dividida em pequenos serviços independentes, cada um responsável por uma funcionalidade específica do negócio. Esses serviços operam de forma autônoma e se comunicam entre si por meio de APIs bem definidas.

Arquitetura Monolítica: A escalabilidade é limitada, pois para aumentar a capacidade de uma funcionalidade específica, é necessário escalar toda a aplicação, o que pode ser ineficiente e custoso. Além disso, a manutenção se torna complexa devido ao alto acoplamento entre os componentes, onde uma alteração pode impactar diversas partes do sistema.

Arquitetura de Microsserviços: Oferece escalabilidade granular, permitindo que apenas os serviços que demandam mais recursos sejam escalados individualmente. A manutenção é facilitada pela independência dos serviços, reduzindo o risco de que mudanças em um serviço afetem os demais.

Arquitetura Monolítica: Embora seja mais simples no início, a complexidade cresce exponencialmente com o tempo, especialmente em projetos de grande porte, dificultando a implementação de novas funcionalidades e a correção de bugs.

Arquitetura de Microsserviços: Introduz uma complexidade adicional na gestão de sistemas distribuídos, exigindo mecanismos eficazes de comunicação entre serviços, monitoramento, segurança e gerenciamento de dados consistentes. A implementação requer um planejamento cuidadoso e uma cultura organizacional alinhada às práticas de DevOps.

2 Conclusão

Este estudo apresentou uma análise comparativa entre as arquiteturas monolítica e de microsserviços no contexto de aplicações web e sistemas distribuídos. Inicialmente, definiu-se o conceito de aplicações web, destacando sua estrutura e funcionamento. Em seguida, explorou-se a anatomia dos sistemas distribuídos, enfatizando seus componentes essenciais e características fundamentais.

Em suma, a decisão entre adotar uma arquitetura monolítica ou de microsserviços deve ser cuidadosamente avaliada, considerando as especificidades de cada projeto e os trade-offs associados a cada abordagem. A compreensão profunda das características, vantagens e desafios de cada modelo arquitetural é essencial para o sucesso e a eficiência das aplicações web em ambientes distribuídos.

REFERÊNCIAS

Microsoft, **Microservices architecture design**

Disponível em: https://learn.microsoft.com/enus/azure/architecture/microservices/

Acesso em: 20/02/2025

Amazon Web Services, **Microsserviços**

Disponível em: https://aws.amazon.com/pt/microservices/

Acesso em: 20/02/2025

Chandler , Harris. **Microsserviços versus arquitetura monolitica**

Disponível em: https://www.atlassian.com/br/microservices/microservices-architecture/microservices-vs-monolith

Acesso em: 20/02/2025

Kev , Zettler. **O que é um sistema distribuído?**

Disponível em: https://www.atlassian.com/br/microservices/microservices-architecture/distributed-architecture

Acesso em: 20/02/2025