ARQUITETURA DE SOFTWARE

João Choma Neto
joao.choma@unicesumar.edu.br
https://github.com/JoaoChoma/arquitetura-software
Unicesumar – Maringá

2 BIMESTRE

- Arquitetura monolítica
- Arquitetura em camadas
- Arquitetura cliente-servidor
- Arquitetura baseada em serviços
- Arquitetura de microserviços

CLIENTE-SERVIDOR

CLIENTE-SERVIDOR

- A arquitetura cliente-servidor é um modelo de design de software em que duas partes interagem em uma rede:
  - o **cliente**, que faz solicitações por serviços ou recursos, e
  - o **servidor**, que processa essas solicitações e fornece respostas

CLIENTE

- **Definição**: Um cliente é uma entidade que solicita recursos ou serviços. Em um contexto web, um cliente geralmente é um navegador que solicita páginas da web.
- **Funções**:
  - Interface com o usuário: Fornece a interface através da qual os usuários interagem.
  - Solicitação de recursos: Envia pedidos ao servidor para acessar recursos como arquivos, dados, etc.
  - Recebimento e processamento de respostas: Recebe dados do servidor e os processa localmente.

SERVIDOR

- **Definição**: Um servidor é uma entidade que fornece recursos, dados ou serviços. Ele responde às solicitações dos clientes.
- **Funções**:
  - Gerenciamento de recursos: Mantém e gerencia recursos como bancos de dados, arquivos, etc.
  - Processamento de solicitações: Processa os pedidos recebidos dos clientes.
  - Envio de respostas: Envia as respostas para as solicitações dos clientes após o processamento.

TIPOS DE CLIENTES

- Navegadores web
- Aplicativos móveis
- Clientes de desktop
- Scripts e bots

TIPOS DE SERVIDORES

- Servidores web (Apache, Nginx)
- Servidores de aplicação (Tomcat, Node.js)
- Servidores de banco de dados (MySQL, MongoDB)
- Servidores de e-mail (Exchange, Postfix)

APLICAÇÕES WEB

- Os verbos HTTP, também conhecidos como métodos HTTP, são um componente crucial do protocolo HTTP (HyperText Transfer Protocol), usado para especificar a ação desejada em um recurso determinado.
- Cada método tem um propósito específico e é projetado para comunicar ao servidor o tipo de operação que o cliente deseja executar.

GET

- **Propósito**: O método GET é usado para solicitar dados de um recurso específico. Geralmente, é usado para recuperar a representação de um recurso sem causar qualquer efeito colateral (ou seja, não modifica o estado do recurso).
- GET é considerado um método seguro pois não altera o estado do recurso
- **Uso comum**: Carregar uma página web, solicitar uma imagem ou outros arquivos.

POST

- **Propósito**: POST é usado para enviar dados ao servidor para criar ou modificar um recurso. Por exemplo, quando você preenche um formulário em uma página web e o envia, seus dados são geralmente enviados ao servidor usando o método POST.
- POST não é seguro, pois modifica o estado do servidor (cria ou altera recursos).
- **Uso comum**: Enviar formulários, fazer upload de um arquivo, realizar uma operação que resulta em mudança de estado.

PUT

- **Propósito**: PUT é usado para atualizar um recurso existente ou criar um novo recurso.
- **Uso comum**: Atualizar um registro completo, como um perfil de usuário ou um post em um blog.

DELETE

- **Propósito**: DELETE é usado para remover um recurso especificado.
- **Uso comum**: Deletar um registro, como um usuário, uma postagem de blog, ou um arquivo.

emails

- O envio de e-mails na internet é primariamente gerenciado por um protocolo específico chamado SMTP (Simple Mail Transfer Protocol).
- Este protocolo é essencial para a operação dos sistemas de e-mail e desempenha um papel fundamental no processo de envio e encaminhamento de mensagens de e-mail entre remetentes e destinatários.

ARQUITETURA BASEADA EM SERVIÇOS

SERVIÇOS

- Arquitetura baseada em serviços, também conhecida como Arquitetura Orientada a Serviços ou SOA (Service-Oriented Architecture)

- Hoje reconhecida como um padrão de projeto que permite a criação de sistemas distribuídos onde os componentes são organizados como serviços

SERVIÇOS

- Estes serviços são projetados para serem reutilizáveis, modulares e capazes de operar de forma independente, possibilitando uma maior flexibilidade e escalabilidade nas aplicações de software

- Um "serviço" é uma unidade funcional de software projetada para realizar uma tarefa específica ou um conjunto de tarefas relacionadas

CARACTERÍSTICAS

- **AUTONOMIA**: Cada serviço é independente e responsável por uma função específica dentro do sistema
- **INTERFACE BEM DEFINIDA:** Serviços comunicam-se com o mundo externo através de interfaces bem definidas. Essas interfaces descrevem os métodos de operação que outros sistemas ou serviços podem invocar
- **REUSABILIDADE:** Os serviços são projetados para serem reutilizáveis em diferentes cenários.

CARACTERÍSTICAS

- **INTEROPERABILIDADE:** Os serviços são construídos para funcionar em diferentes ambientes de tecnologia, o que é facilitado pelo uso de padrões abertos de comunicação, como SOAP ou REST
- **ENCAPSULAMENTO:** Serviços encapsulam a lógica de negócios e os dados necessários para executar suas funções

SOAP (Simple Object Access Protocol)

- SOAP é um protocolo baseado em XML para troca de informações em redes de computadores.
- SOAP pode operar sobre qualquer protocolo de transporte como HTTP, SMTP, TCP, etc., embora HTTP seja o mais comum.

REST (Representational State Transfer)

- REST não é um protocolo, mas sim um conjunto de princípios arquiteturais. Utiliza os métodos HTTP padrões (GET, POST, PUT, DELETE, etc.) de maneira mais direta e eficiente.

- REST é um estilo arquitetural sem estado, significando que cada pedido HTTP deve conter todas as informações necessárias para compreendê-lo, sem exigir que o servidor memorize qualquer estado de sessão.

REST (Representational State Transfer)

- Embora o JSON seja o formato de dados mais popular usado em APIs REST devido à sua facilidade de uso com JavaScript e sua leveza em comparação ao XML, REST permite o uso de diferentes formatos como XML, HTML, YAML, etc.

REST

- **APIs de Mídias Sociais**
  - Twitter API
  - Facebook Graph API
- **APIs de Mapas e Localização**
  - Google Maps API
  - OpenStreetMap API

REST

- **APIs de Compartilhamento de Vídeos**
  - YouTube API:
- **APIs de Comércio Eletrônico**
  - Shopify API
  - Ebay API

REST

- **APIs de Armazenamento em Nuvem**
  - Dropbox API
  - Google Drive API
- **APIs de Meteorologia**
  - OpenWeatherMap API
  - Weather API

REST

- **APIs Financeiras**
  - Stripe API
  - PayPal API

FIREBASE

- Firebase, uma plataforma de desenvolvimento de aplicativos da Google, também oferece várias APIs REST que permitem aos desenvolvedores interagir com seus serviços de backend

ARQUITETURA DE SOFTWARE

João Choma Neto
joao.choma@unicesumar.edu.br
https://github.com/JoaoChoma/arquitetura-software
Unicesumar – Maringá