

# WEB - Services

---

Slides baseados no material do prof. FELIPE CUNHA

# Service Oriented Architecture – SOA

---

SOA é uma arquitetura que representa funcionalidades do software como serviços

Neste modelo de arquitetura os principais requisitos viram serviços e são acessados por outros serviços

Modularização e aumento da coesão dos componentes da aplicação

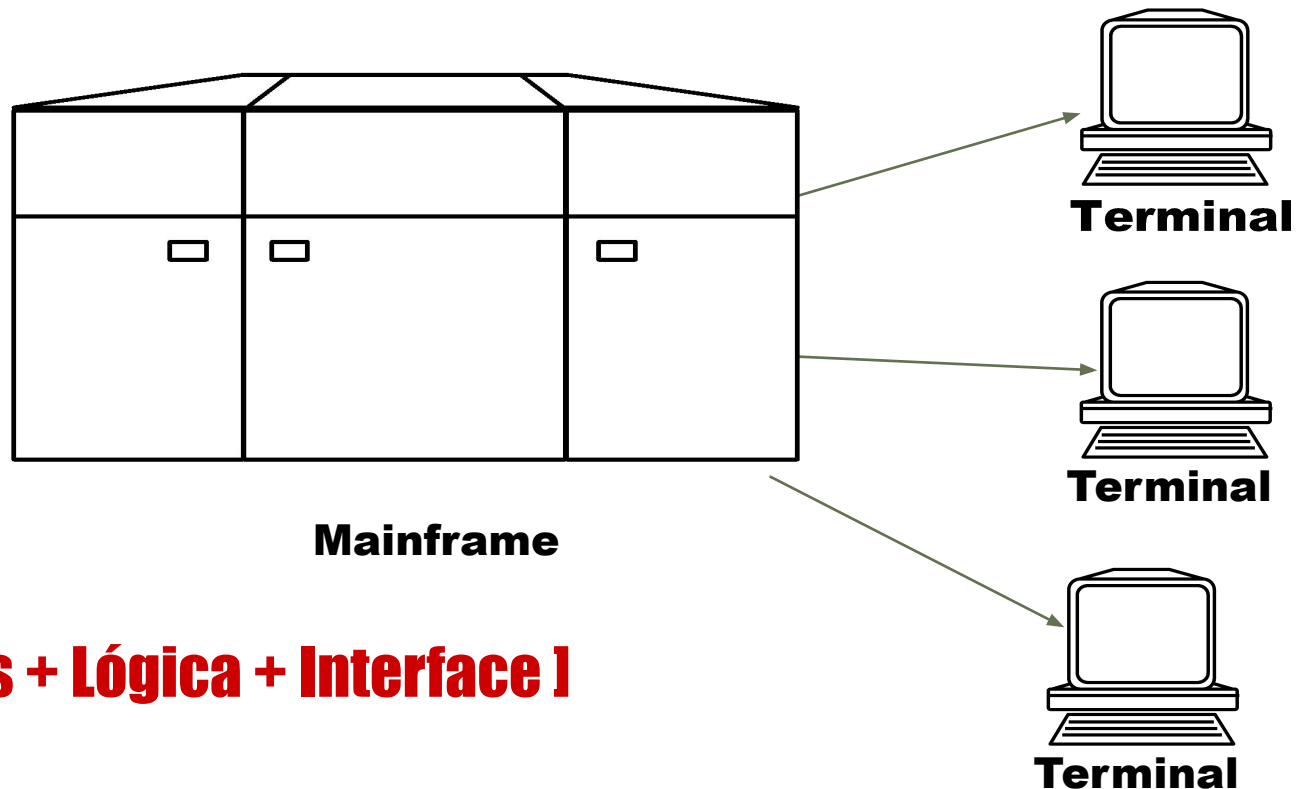
Interoperabilidade é muito importante

- Padronização
- Fraco acoplamento

# Histórico das Arquiteturas

---

## Mainframes

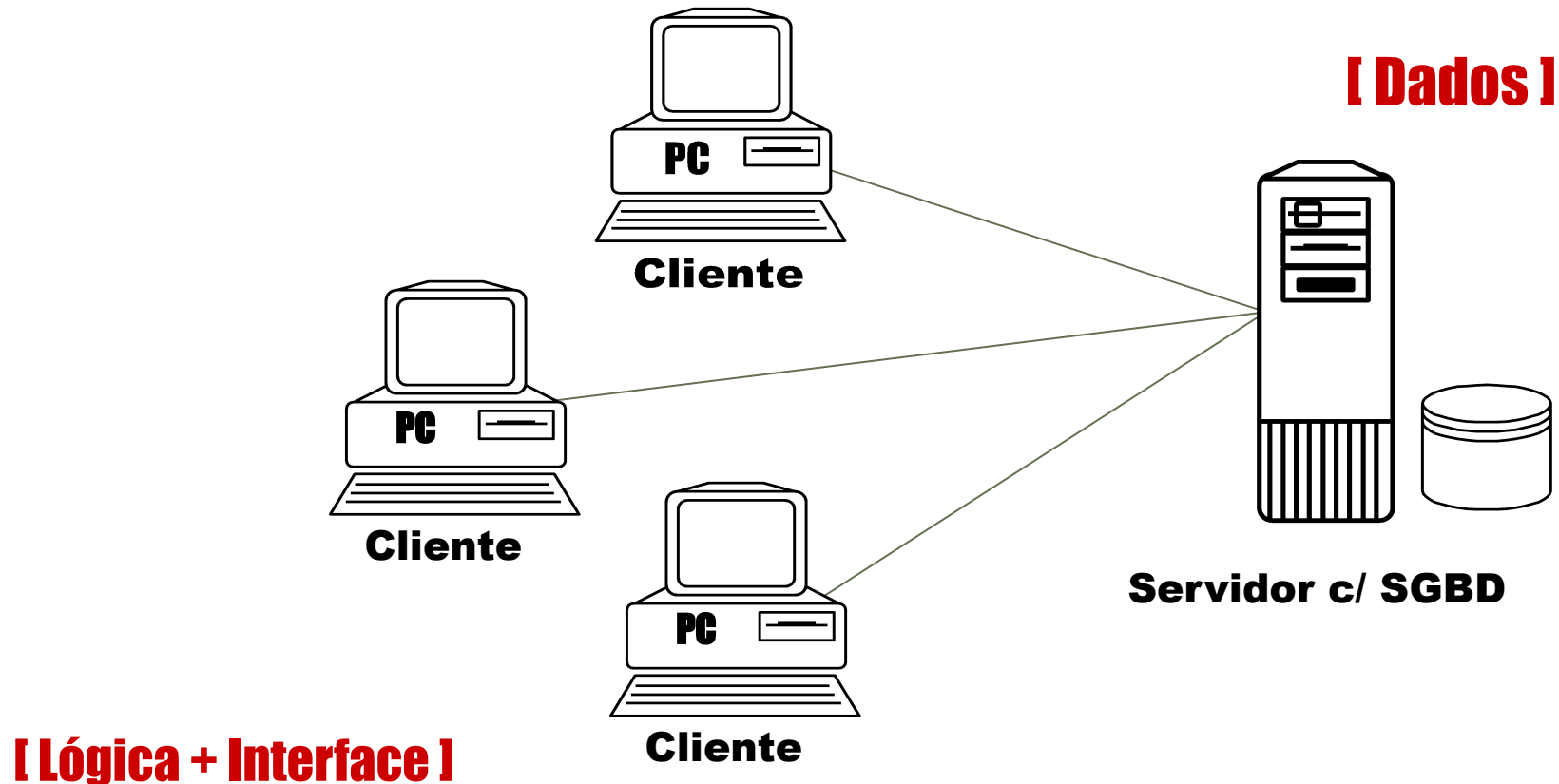


**[ Dados + Lógica + Interface ]**

# Histórico das Arquiteturas

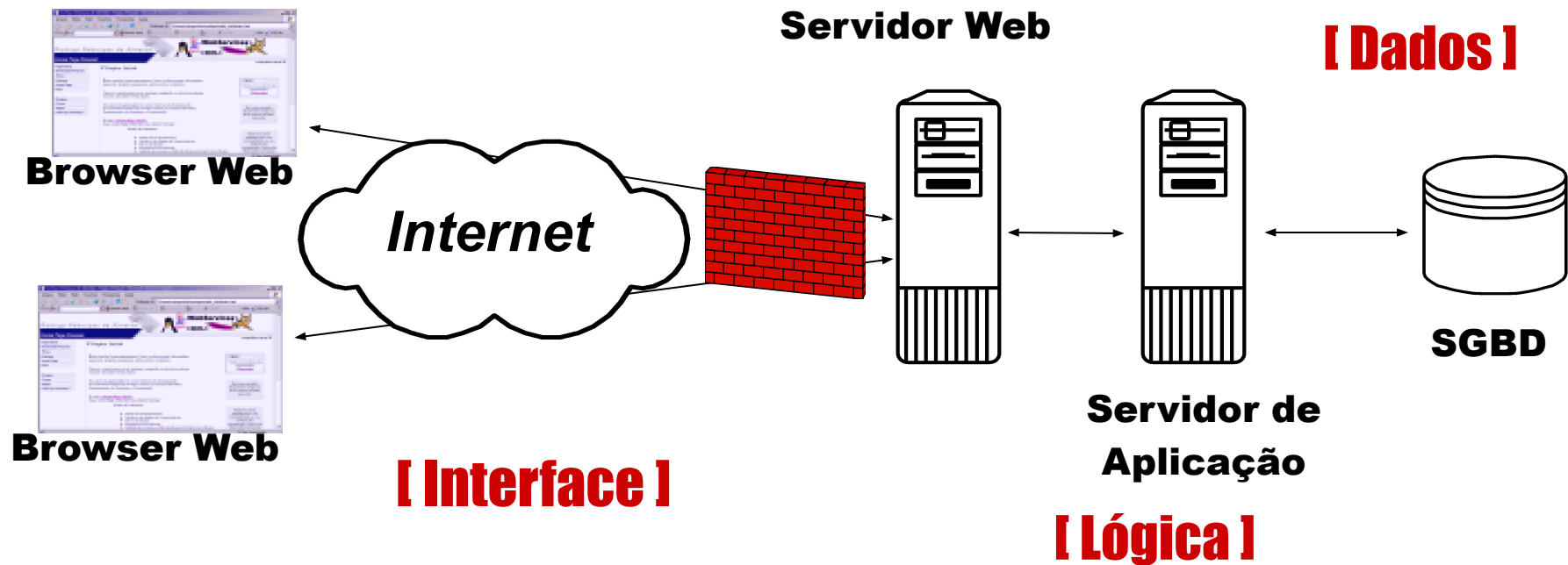
---

## Arquitetura Cliente-Servidor



# Histórico das Arquiteturas

## Arquitetura em N-Camadas



# Arquitetura Orientada a Serviço

---

Modelo arquitetural para construção de aplicativos que promove:

- Um baixo acoplamento entre os componentes que podem ser reusados e trabalham juntos como uma arquitetura distribuída



# SOA – Principais Conceitos

---

**Serviços:** fornecem as funcionalidades do negócio

**Interfaces auto descritivas:** independente de plataforma, separada da implementação contendo a descrição das operações do serviço

**Comunicação síncrona e assíncrona:** troca de mensagens realizadas por SOA deve suportar chamadas síncronas e assíncronas

**Baixo acoplamento:** descrição dos serviços pelo uso de interfaces e protocolos independentes de linguagem e plataforma

**Composição de serviços:** serviços individuais podem ser agrupados para formar um serviço mais elaborado

# SOA – Papéis e Funções

## Provedor

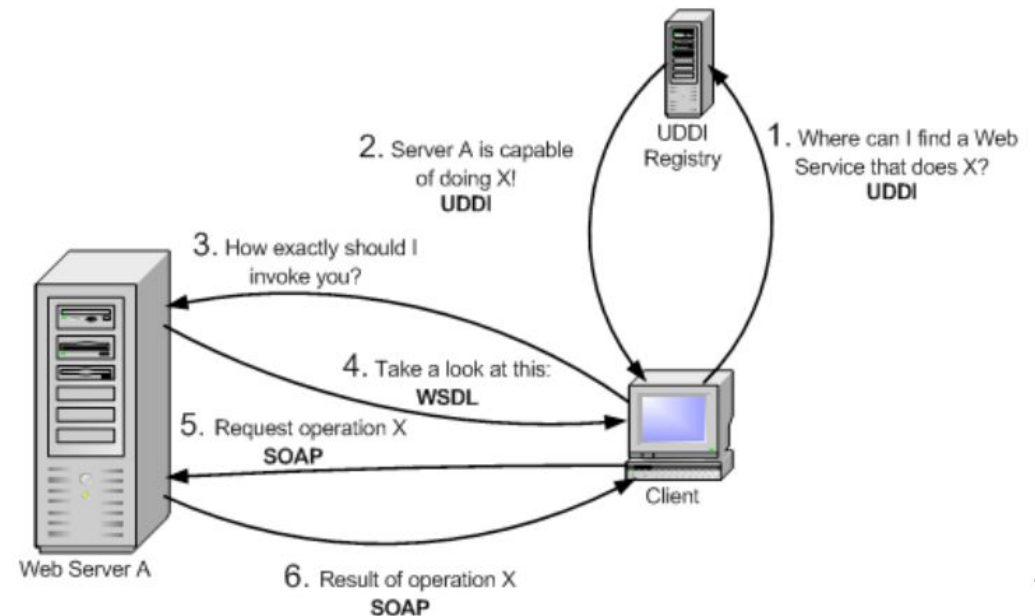
- descreve e publica seu serviço

## Registro de Serviços

- mantém informações sobre serviços e suas localizações

## Cliente

- localiza provedores de serviços através do registro de serviços



<sup>1</sup>globus.org, *Grid Computing*

# Computação Orientada a Serviço

---

SOC é um paradigma para a computação distribuída que define o modo com o qual aplicações são desenvolvidas, projetadas, disponibilizadas e “consumidas”

SOC é um paradigma computacional que utiliza serviços como unidades básicas:

- desenvolvimento rápido
- baixo custo
- fácil composição de aplicações distribuídas (até mesmo em ambientes heterogêneos)

# O Que é um Serviço ?

---

## Serviço Windows

- Servidor DHCP, Serviço de Terminal, Log de Eventos, ...

## Serviço de Software

- Serviços de *Middleware*
- Serviços Distribuídos
- RMI

## Serviço de Negócio

- Serviço de Mapas: *Google Maps*
- *Flickr*

# Serviço Web

---

Serviço Web (*Web Services*) é uma tecnologia de **chamada remota de objetos**

Fornece **infraestrutura para desenvolvimento** de aplicações distribuídas (Web ou não)

Permite a criação de pequenos módulos de código reutilizáveis e disponibilizados para construção de **aplicações “tipo LEGO”**

Utiliza protocolos **Web como meio de transporte** e comunicação

Alto grau de **abstração** em relação a **linguagens** de programação e **plataformas** de hardware / software

Base da integração de sistemas distribuídos modernos

# Serviço Web

---

Um Serviço Web é um ponto de acesso a funcionalidade que pode ser:

- LOCALIZADO dinamicamente
- Ter sua interface DESCOBERTA automaticamente, porque o serviço sabe se descrever
- Ser CHAMADO na Web

Não faz parte do conceito de Serviço Web a criação de interfaces gráficas para os usuários

Serviço Web é a tecnologia ideal para comunicação entre sistemas (aplicações Business to Business / B2B)

# XML foi a base de tudo

---

Oferece um formato ASCII para trocar qualquer tipo de informação estruturada

Usa o “estilo” HTML de *markup* com *tags*

- `< Pessoa nome=“João”>  
 < frutasFavoritas>  
 < fruta>Manga</fruta>  
 < fruta>Maçã</fruta>  
 < fruta>Uva</fruta>  
 </frutasFavoritas>  
</Pessoa>`

Um Serviço Web pode ser descrito e publicado em um arquivo XML de acordo com WSDL

# Serviço Web - SOAP

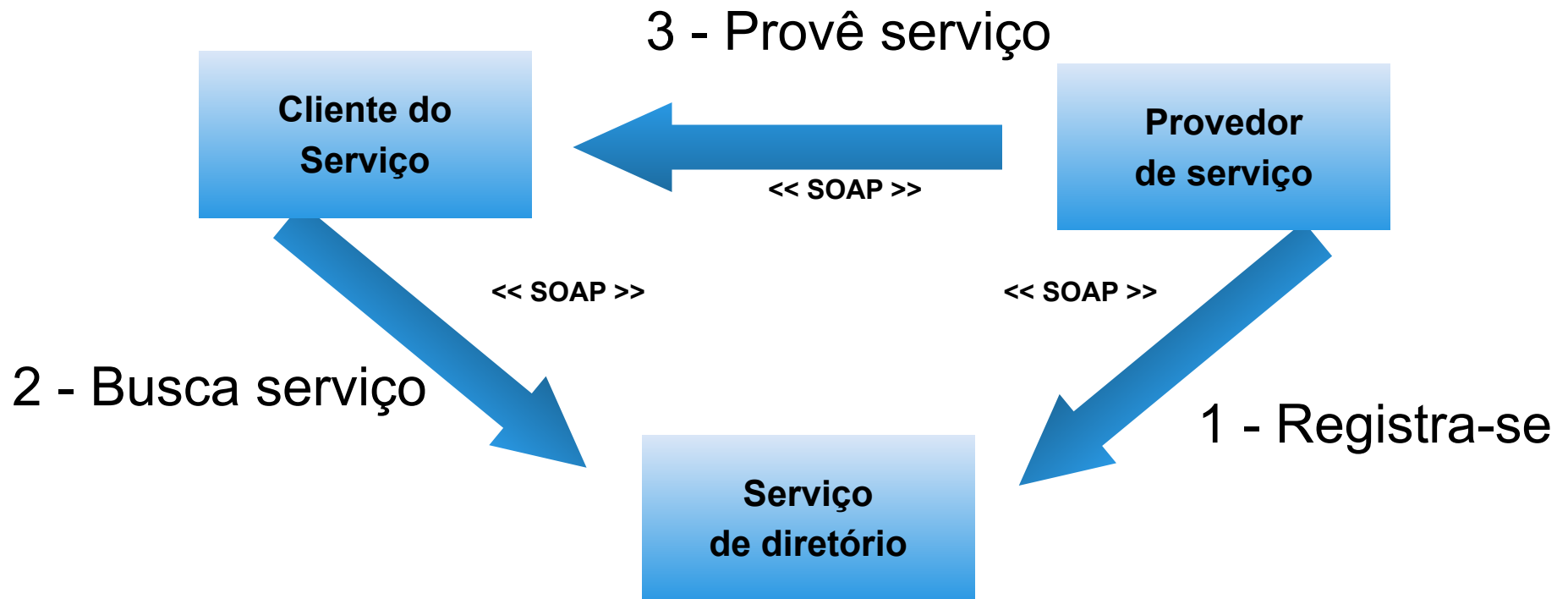
---

Web Services se fundamentavam basicamente em três tecnologias

- Simple Object Access Protocol (SOAP)
  - Um protocolo baseado em XML que permite que os clientes se comuniquem com os provedores de serviço
- Web Services Description Language (WSDL)
  - Linguagem para definição/descrição da interface de acesso ao serviço
- Universal Description, Discovery and Integration (UDDI)
  - Permite o registro dos Serviços Web possibilitando que outras aplicações os encontrem

# SOAP – Arquitetura

---



# Características do SOAP

---

Definido pelo consórcio W3C (principal organização de padronização da World Wide Web)

Protocolo baseado em XML para a troca de informações em um ambiente distribuído

Padrão de utilização com Serviços Web

Normalmente utiliza HTTP e SMTP como protocolo de transporte

É mais utilizado sobre HTTP pois consegue atravessar firewalls

# Estrutura da Mensagem SOAP

---

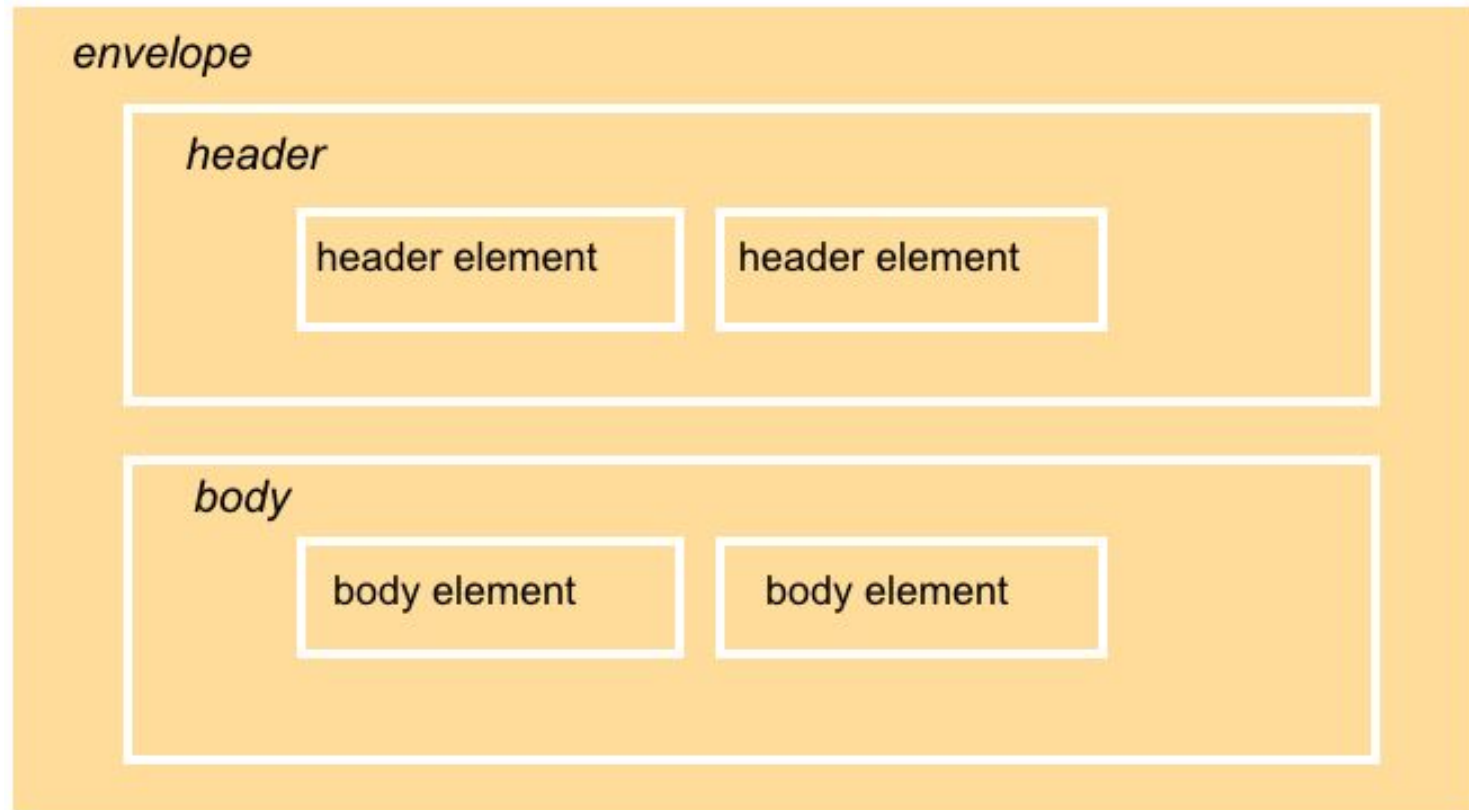
**Envelope:** toda mensagem SOAP deve contê-lo pois ele representa o elemento raiz do documento XML

**Header:** é um cabeçalho opcional que carrega informações adicionais como, por exemplo, se a mensagem deve ou não ser processada por um determinado nó intermediário (se utilizado, o Header deve ser o primeiro elemento do Envelope)

**Body:** este elemento é obrigatório e contém o *payload* ou a informação a ser transportada para o seu destino final

# Mensagem SOAP em um envelope

---



# Estrutura de Mensagem SOAP

---

```
<SOAP:Envelope xmlns:SOAP=  
    http://schemas.xmlsoap.org/soap/envelope/>
```

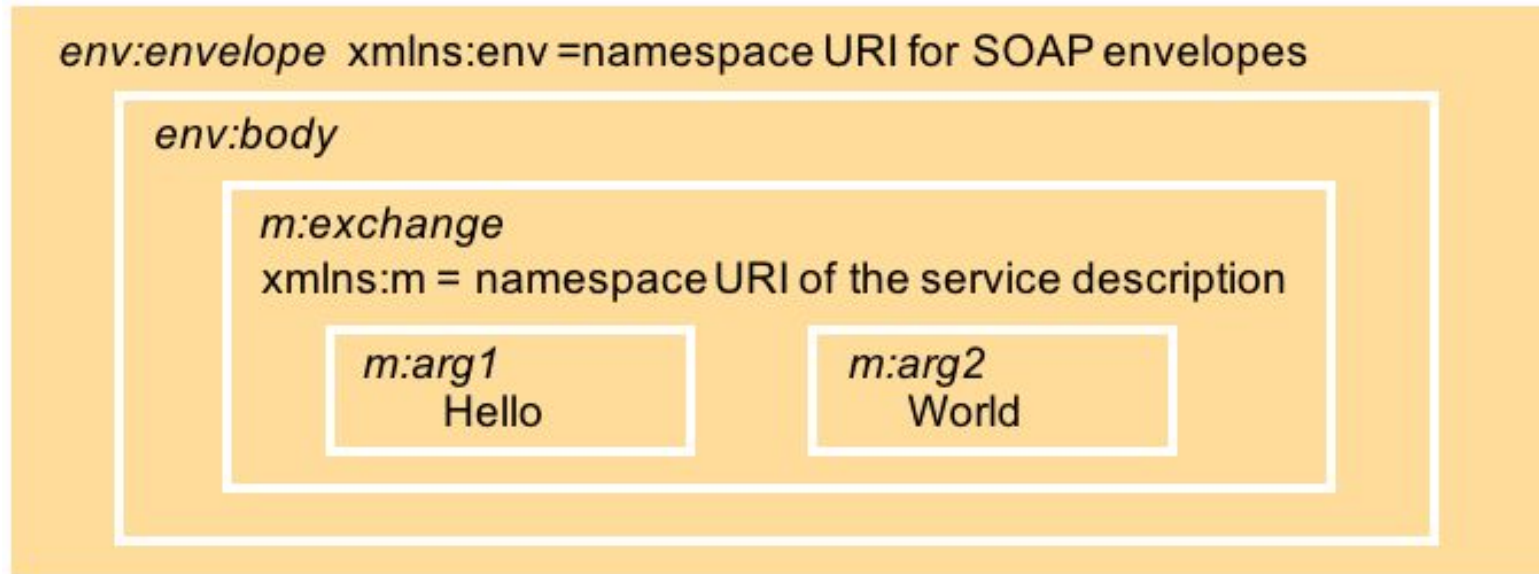
```
    <SOAP:Header>  
        <!conteudo do cabecalho >  
    </SOAP:Header>
```

```
    <SOAP:Body>  
        <!conteudo do corpo>  
    </SOAP:Body>
```

```
</SOAP:Envelope>
```

# Exemplo de uma requisição simples sem os cabeçalhos

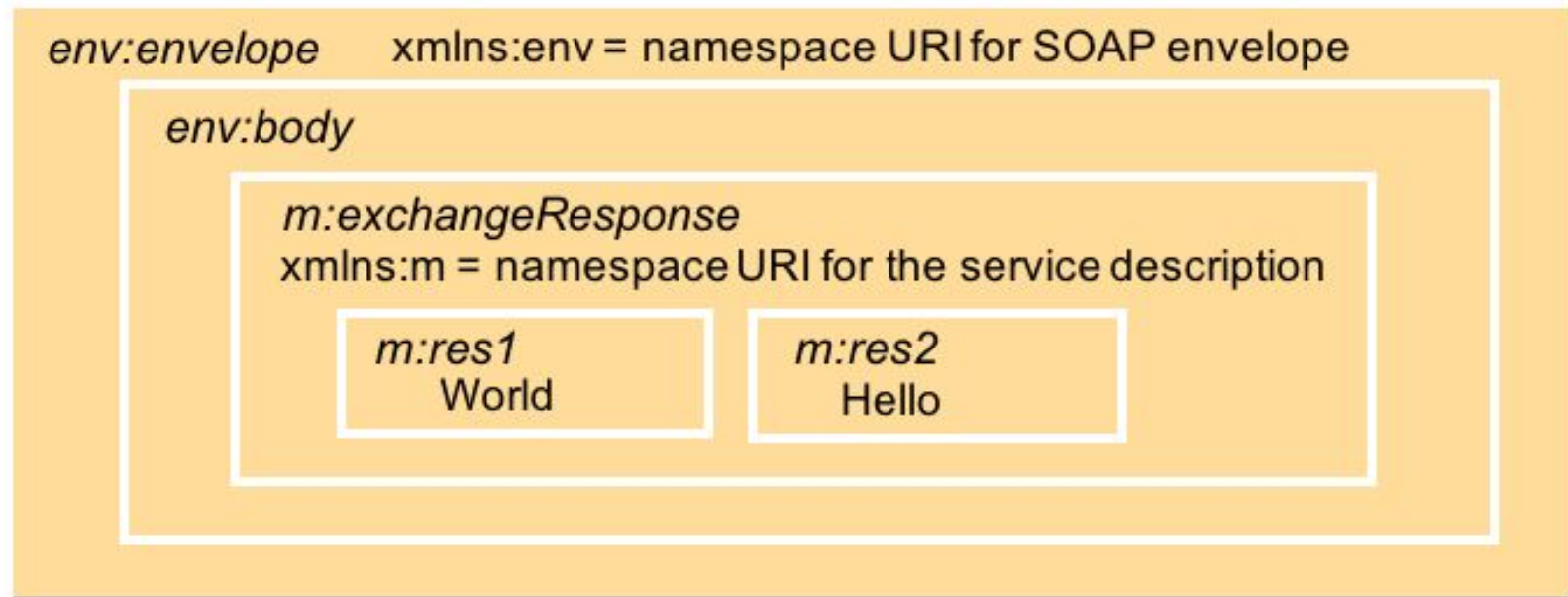
---



Cada elemento XML é representado por um bloco com seu nome seguido pelos seus argumentos e conteúdo

# Exemplo de uma resposta (reply) para uma requisição como a anterior

---



# Uso de uma requisição HTTP POST na comunicação cliente-servidor do SOAP

```
POST /examples/stringer ← endpoint address
Host: www.cdk4.net
Content-Type: application/soap+xml
Action: http://www.cdk4.net/examples/stringer#exchange ← action
```

HTTP  
header

```
<env:envelope xmlns:env= namespace URI for SOAP envelope
<env:header> </env:header>
<env:body> </env:body>
</env:Envelope>
```

Soap  
message

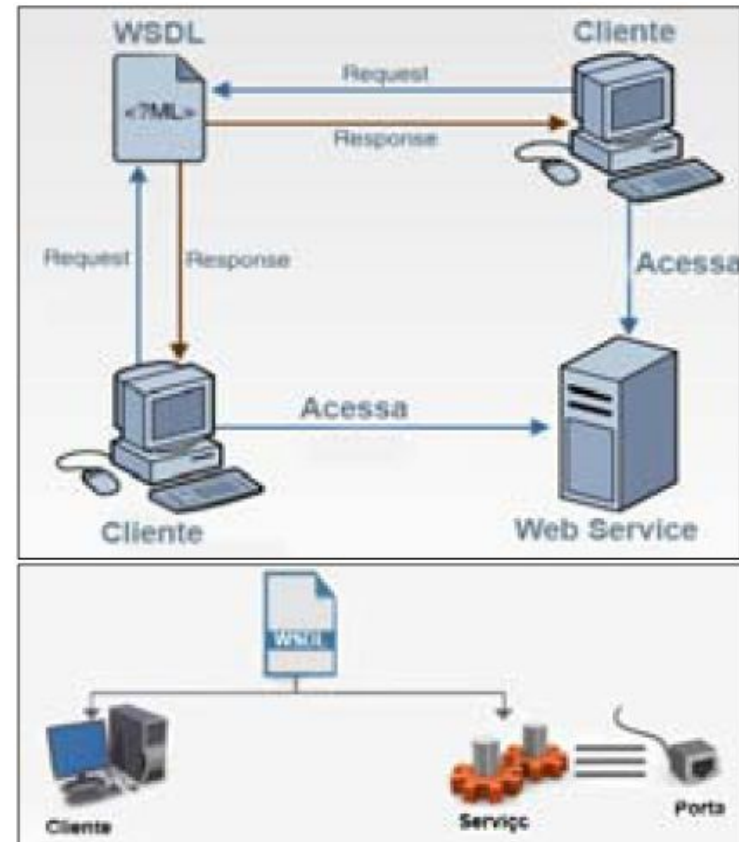
# Camada de Descrição – WSDL

WSDL é um documento XML que fornece informações sobre o Serviço Web de maneira independente de linguagem e plataforma

Clientes precisam saber como acessar um Serviço Web

- Qual a operação
- Quais os parâmetros
- Qual o endereço

WSDL define serviços por meio de portas em que cada porta está associada a um serviço específico



# Documento WSDL

---

WSDL  $\equiv$  Web Services Description Language

É uma linguagem XML que contém informação sobre a interface, a semântica, e outros detalhes de chamadas a um Serviço Web

Em resumo: Linguagem XML para descrever um Serviço Web

# Documento WSDL

---

Um documento WSDL define um XML Schema para descrever um Serviço Web

Cliente enviando uma mensagem a um Serviço Web:

- Obtém a descrição do serviço (WSDL)
- Constrói a mensagem, passando os parâmetros corretos baseados no documento
- Mensagem enviada para o endereço onde o serviço está localizado
- O Serviço Web quando recebe a mensagem, valida a mesma baseado no WSDL
- Executa o serviço e responde ao cliente

# WSDL – Especificação

---

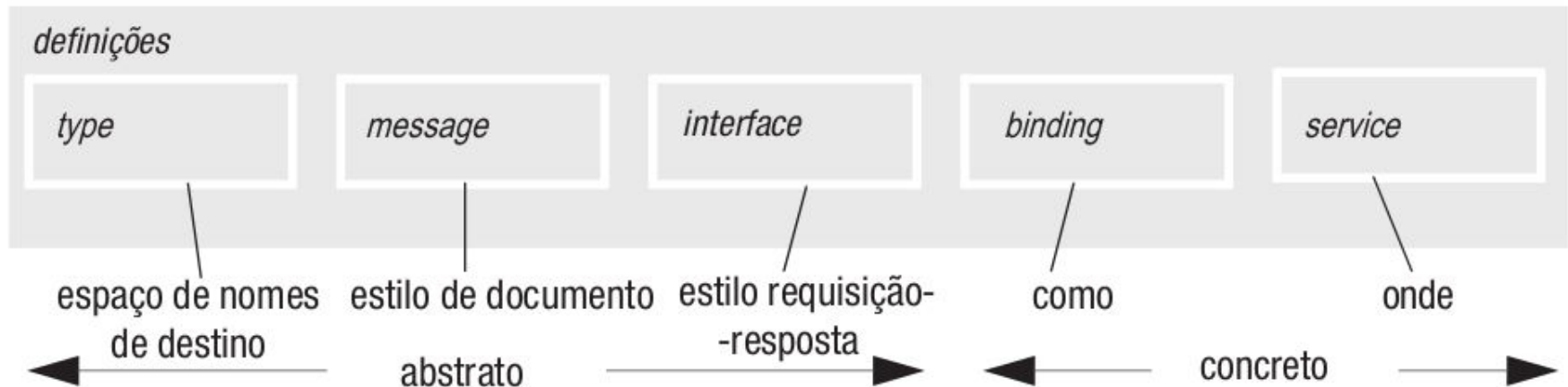
Um documento WSDL é formado por componentes

- Operações
- Tipo de Dados (*XML Schema*)
- Protocolos

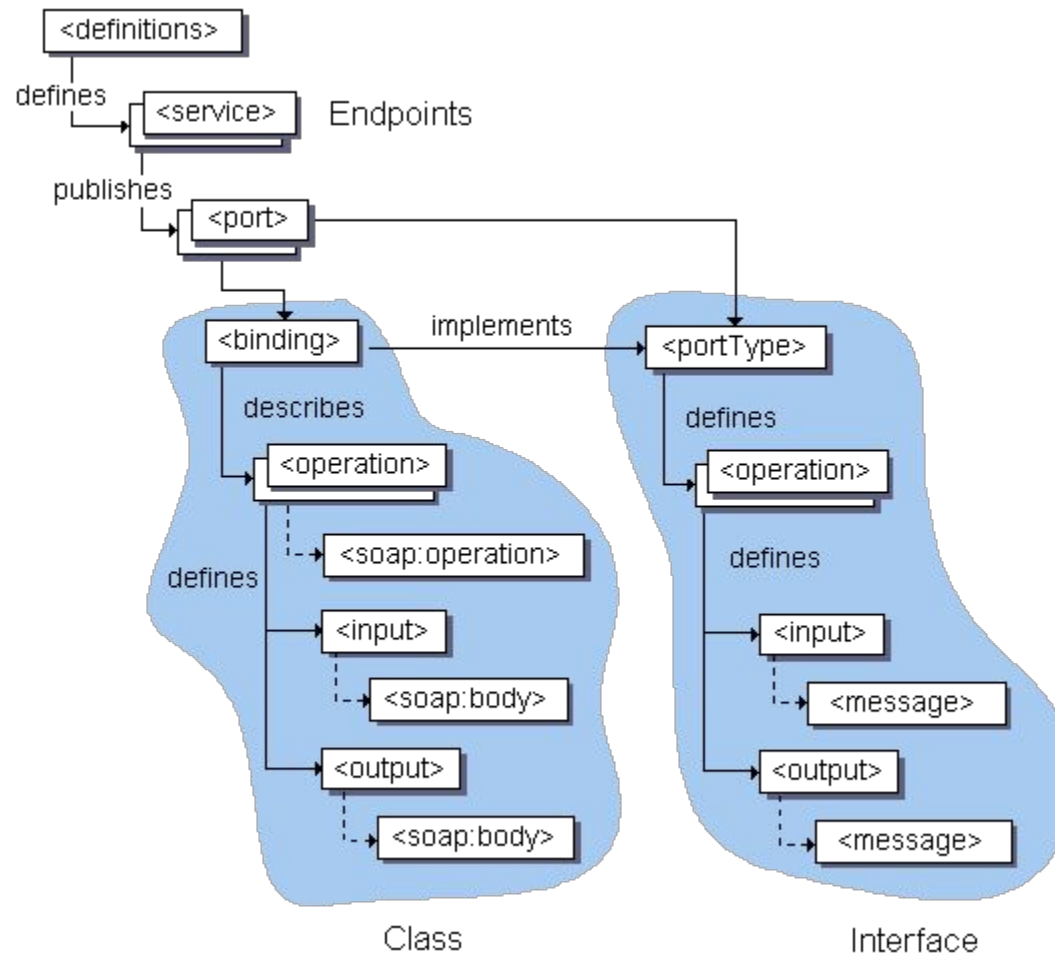


# WSDL – Especificação

- Um documento WSDL contém uma parte concreta e outra abstrata
  - o Parte abstrata é a interface
  - o Parte concreta define forma de acesso ao serviço
- Uma troca de mensagens é chamada de operação
- Operações (input/output) são agrupadas em interfaces
- Binding especifica detalhes concretos do transporte



# Definições do serviço e do binding SOAP



# Exemplo WSDL – Definição de Tipos

---

```
<?xml version='1.0' encoding='UTF-8'?>
<definitions
xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-
wssecurity-utility-1.0.xsd"
xmlns:wsp="http://www.w3.org/ns/ws-policy"
xmlns:wsp1_2="http://schemas.xmlsoap.org/ws/2004/09/policy"
xmlns:wsam="http://www.w3.org/2007/05/addressing/metadata"
xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
xmlns:tns="http://webservice.teste.my/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns="http://schemas.xmlsoap.org/wsdl/"
targetNamespace="http://webservice.teste.my/" name="TesteService">
<types>
<xsd:schema>
<xsd:import namespace="http://webservice.teste.my/"
schemaLocation="http://localhost:8080/TesteInicial/TesteService?xs
d=1" />
</xsd:schema>
</types> ...
```

# Exemplo WSDL – XML Schema

---

```
<?xml version='1.0' encoding='UTF-8'?>
<xs:schema xmlns:tns="http://webservice.teste.my/"
xmlns:xs="http://www.w3.org/2001/XMLSchema" version="1.0"
targetNamespace="http://webservice.teste.my/">

<xs:element name="TestOp" type="tns:TestOp" />
<xs:element name="TestOpResponse" type="tns:TestOpResponse" />

<xs:complexType name="TestOp">
  <xs:sequence>
    <xs:element name="param1" type="xs:string" minOccurs="0" />
  </xs:sequence>
</xs:complexType>
<xs:complexType name="TestOpResponse">
  <xs:sequence>
    <xs:element name="return" type="xs:string" minOccurs="0" />
  </xs:sequence>
</xs:complexType>
</xs:schema>
```

# Exemplo WSDL – Definição de Msgs

---

```
...  
<message name="TestOp">  
  <part name="parameters" element="tns:TestOp" />  
</message>  
<message name="TestOpResponse">  
  <part name="parameters" element="tns:TestOpResponse" />  
</message>  
  
<portType name="TestWebService">  
  <operation name="TestOp">  
    <input  
wsam:Action="http://webservice.teste.my/TestWebService/  
      TestOpRequest" message="tns:TestOp" />  
    <output wsam:Action="http://webservice.teste.my/TestWebService/  
      TestOpResponse" message="tns:TestOpResponse" />  
    </operation>  
  </portType>  
...
```

# Exemplo WSDL – Definição do Serviço

---

```
...
<binding name="TestWebServicePortBinding" type="tns:TestWebService">
  <soap:binding transport="http://schemas.xmlsoap.org/soap/http"
    style="document" />
  <operation name="TestOp">
    <soap:operation soapAction="" />
    <input> <soap:body use="literal" /> </input>
    <output> <soap:body use="literal" /> </output>
  </operation>
</binding>
<service name="TesteService">
  <port name="TestWebServicePort"
    binding="tns:TestWebServicePortBinding">
    <soap:address
location="http://localhost:8080/TesteInicial/TesteService" />
  </port>
</service>
</definitions>
```

# Camada de Busca – UDDI

---



*Framework independente de plataforma usado na comunicação entre provedores de serviço e consumidores*

*UDDI descreve como criar registro para armazenar as informações sobre Serviço Web*

*Funciona como uma lista telefônica de Serviços Web*

# Ferramentas para Desenvolvimento de Serviços Web

---

- WSDL podem ser gerados automaticamente a partir de classes Java ou C#.
- Proxies e skeletons podem ser gerados a partir do WSDL automaticamente.
  - P. Ex. wsimport (JDK) ou wsdl.exe (Microsoft).

# Exemplo: DataFlex Web Service for Country information

---

<http://webservices.oorsprong.org/websamples.countryinfo/CountryInfoService.wso>

# REST

---

## **RE**presentational **S**tate **T**ransfer

Estilo arquitetural (não é um protocolo)

Usa diretamente o protocolo HTTP

- Cada recurso web possui um endereço
- A web possui um número restrito de métodos usados para manipular os recursos

Stateless: cada requisição deve conter toda a informação necessária

Interface uniforme: todos os recursos são acessados com uma interface genérica baseada em HTTP

Componentes em camada: intermediários como proxies, caches, etc melhoram a performance e a segurança

Dados podem ser XML, JSON ou binário.

# REST

---

Tudo é um recurso: usuários, produtos, pedidos

- Representado em URLs: /users, /products/42

HTTP Methods

- GET: ler recursos
- POST: criar recurso
- PUT/PATCH: atualizar recurso
- DELETE: remover recurso

Servidor não mantém estado do cliente entre requisições

Exemplo:

- GET /users/1 -> { "id": 1, "name": "Matheus" }
- POST /users { "name": "Ana" } -> { "id": 2, "name": "Ana" }

# REST - Exemplo de Requisição

---

Requisição:

GET <https://api.exemplo.com/alunos/123>

Resposta:

```
{  
  "id": 123,  
  "nome": "Ana",  
  "curso": "Computação"  
}
```

# Estrutura de uma API RESTful

---

**Endpoints:** `/users`, `/products/{id}`

**Parâmetros de URL:**

- Path params: `/users/123`
- Query params: `/products?category=books`

**Request e Response:**

- Headers: autenticação, tipo de conteúdo (`Content-Type: application/json`)
- Body: dados enviados ou recebidos

**Status Codes HTTP:**

- `200 OK` → sucesso GET/PUT
- `201 Created` → recurso criado (POST)
- `204 No Content` → sucesso DELETE
- `400 Bad Request` → erro do cliente
- `401 Unauthorized` → falta autenticação
- `403 Forbidden` → sem permissão
- `404 Not Found` → recurso inexistente

# SOAP vs. REST

---

## ● SOAP:

- Baseado em XML
- Alta complexidade (WSDL, UDDI)
- Usado em bancos e governo
- Suporta uma variedade de protocolos (transporte, autenticação, criptografia, etc)

## ● REST:

- JSON ou XML
- Simples, direto sobre HTTP (métodos GET, PUT, POST, DELETE)
- Usado em web, mobile, APIs públicas

Muitos sites suportam ambos os padrões

# APIs REST no Mundo Real

---

- Microserviços conectados via APIs
- API Gateways centralizando chamadas
  - Roteamento de requisições
    - Exemplo: /users vai para o serviço de usuários, /orders para o serviço de pedidos
  - Autenticação e Autorização
  - Caching
- Documentação via OpenAPI
  - Especificação para descrever APIs RESTful de forma padronizada, legível por humanos e máquinas.
- Exemplos:
  - Google (Maps, Drive)
  - GitHub, Twitter