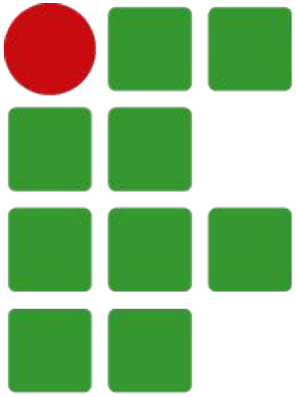

INSTITUTO FEDERAL
Paraná
Campus Paranaguá

Web Services

Dispositivos Móveis

Prof. Diego Stiehl

INSTITUTO FEDERAL

Web Service

- Solução para integração de sistemas
  - Comunicação entre diferentes aplicações
- Novas aplicações podem interagir com outras já existentes
  - Diferentes plataformas e linguagens
- Utiliza protocolo HTTP (ou HTTPS)
  - Troca-se informações usando algum formato
    - XML, CSV, JSON, texto puro, ...

INSTITUTO FEDERAL

Web Service

- São serviços na web
  - Arquitetura cliente-servidor
  - Recursos oferecidos via HTTP
  - Possuem uma URL
  - Como uma página “normal”
- Seu retorno (response) não é um HTML puro para ser renderizado no browser
  - Com uma requisição, você obterá:
    - **JSON**, XML, ...

INSTITUTO FEDERAL

Quando usar?

- Quando sua aplicação precisa acessar (ou gerar) dados fora de seu domínio, em outro servidor, localizado publicamente na web
  - Via protocolo HTTP

INSTITUTO FEDERAL

Exemplos de Web Services

- Facebook Graph API
  - Leitura e gravação de dados para Facebook
- Twitter REST APIs
  - Leitura e escrita de dados no Twitter
- Google Maps APIs Web Services
  - Fornece dados geográficos para apps de mapa
- Amazon Web Services
  - Coleção de serviços de computação em nuvem

INSTITUTO FEDERAL

SOAP / REST

- Atualmente, existem dois tipos de estratégias de web services largamente utilizadas
  - **SOAP**
    - **S**imple **O**bject **A**ccess **P**rotocol
    - É um protocolo
  - **REST** (nosso foco)
    - **RE**presentational **S**tate **T**ransfer
    - É um estilo de arquitetura

INSTITUTO FEDERAL

SOAP

- É um protocolo de comunicação
  - https://www.w3.org/TR/soap
  - Baseado em XML
- Mensagens bastante verbosas
  - Muita verificação e segurança
- Estrutura de dados bem definida
  - WSDL (Web Service Description Language)
- Preferida para estruturas de dados complexas
  - Garantia da consistência dos dados

INSTITUTO FEDERAL

Exemplo de Uso de SOAP

- Nota Fiscal Eletrônica Brasileira (NF-e)
  - Há uma estrutura de dados muito complexa
  - Informações sigilosas
  - Necessidade de integridade
  - Garantia de recepção
  - Validações
  - ...

INSTITUTO FEDERAL

SOAP

- Toda requisição / resposta com SOAP deve enviar um XML completo
  - O XML é validado com WSDL
    - Estrutura rígida e verbosa
- Aceita somente métodos GET e POST
- Serviços são registrados em UDDIs
  - Universal Description, Discovery and Integration

INSTITUTO FEDERAL

SOAP - Requisição

```
POST /InStock HTTP/1.1
Host: www.example.org
Content-Type: application/soap+xml; charset=utf-8
Content-Length: nnn

<?xml version="1.0"?>
<soap:Envelope
xmlns:soap="http://www.w3.org/2001/12/soap-envelope"
soap:encodingStyle="http://www.w3.org/2001/12/soap-encoding">

<soap:Body xmlns:m="http://www.example.org/stock">
 <m:GetStockPrice>
  <m:StockName>IBM</m:StockName>
 </m:GetStockPrice>
</soap:Body>

</soap:Envelope>
```

INSTITUTO FEDERAL

SOAP - Resposta

```
HTTP/1.1 200 OK
Content-Type: application/soap+xml; charset=utf-8
Content-Length: nnn

<?xml version="1.0"?>
<soap:Envelope
xmlns:soap="http://www.w3.org/2001/12/soap-envelope"
soap:encodingStyle="http://www.w3.org/2001/12/soap-encoding">

<soap:Body xmlns:m="http://www.example.org/stock">
  <m:GetStockPriceResponse>
    <m:Price>34.5</m:Price>
  </m:GetStockPriceResponse>
</soap:Body>

</soap:Envelope>
```

INSTITUTO FEDERAL

SOAP - WSDL

```
<message name="getTermRequest">
 <part name="term" type="xs:string"/>
</message>

<message name="getTermResponse">
 <part name="value" type="xs:string"/>
</message>

<portType name="glossaryTerms">
 <operation name="getTerm">
  <input message="getTermRequest"/>
  <output message="getTermResponse"/>
 </operation>
</portType>
```

INSTITUTO FEDERAL

SOAP

- SOAP não é nosso foco, por ora
  - Mas lembre-se que ele tem seu motivo de uso

INSTITUTO FEDERAL

REST

- REST
  - É considerado um estilo de arquitetura
    - Forma de montar e enxergar uma aplicação web
- Se baseia fortemente no que o protocolo **HTTP** oferece nativamente
  - Tenta não reinventar a roda
- Comunicação **menos rígida**
  - Não há restrição no formato das mensagens
- Não obriga envio de conteúdo de requisição

INSTITUTO FEDERAL

REST

- Oferece bastante flexibilidade
- Dá **significado** aos recursos do HTTP
  - URLs, métodos, cabeçalhos, query strings, …
- As mensagens podem ser enviadas e recebidas em qualquer formato
  - Mais comum:
    - JSON
- Web Services ficam menores e mais leves

INSTITUTO FEDERAL

JSON

- O XML é um padrão muito utilizado
  - É bastante extenso e possui muitas qualidades
  - Mas seus documentos costumam ficar grandes
    - Não ideais para casos onde há muitas requisições
- O padrão **JSON** consegue apresentar as mesmas informações
  - Com menos burocracia que o XML
  - Com **menos código** que o XML
    - Ou seja: menos caracteres → menos bytes

INSTITUTO FEDERAL

JSON

- **J**ava**S**cript **O**bject **N**otation
  - Formato que nasceu no JavaScript
- Um documento JSON é pequeno
- Scripts JS podem facilmente “parsear” JSON
  - Criam objetos dentro da linguagem

INSTITUTO FEDERAL

REST

- No REST, os **recursos** são o ponto central
  - Tudo pode ter sentido semântico:
    - URL
    - Método HTTP de requisição
    - Cabeçalhos HTTP de requisição e resposta
    - Códigos de status
    - …
- A ideia é que não seja preciso “inventar” um novo protocolo somente para descrever os dados da sua aplicação

INSTITUTO FEDERAL

REST

- Não retém estado
  - Cada requisição é tratada em separado
- Exemplos de URLs com REST
  - GET → http://meusite.com/produtos/1234
    - Retorna um JSON contendo o produto 1234
  - DELETE → http://meusite.com/produtos/33
    - Apaga o produto 33 no servidor e retorna um JSON
  - GET → http://meusite.com/produtos
    - Retorna um JSON com todos os produtos

INSTITUTO FEDERAL

REST

- Desvantagem
  - A flexibilidade e pouca rigidez do REST geram uma de suas desvantagens
    - Algumas vezes é **difícil saber** exatamente o que enviar e o que se receberá de um web services
    - Podem haver problemas de interoperabilidade
  - Há necessidade de documentação paralela

INSTITUTO FEDERAL

REST - Exemplo

- Apenas usando URL, método HTTP e query strings, conseguimos oferecer um CRUD completo

URL	Método	Função
http://exemplo.com/users	GET	Retorna JSON com todos usuários
http://exemplo.com/users	POST	Recebe os dados de formulário para inserir novo usuário
http://exemplo.com/users/1	GET	Retorna JSON com dados do usuário 1
http://exemplo.com/users/1	PUT	Recebe os dados de formulário para editar o usuário 1
http://exemplo.com/users/1	DELETE	Remove o usuário 1