INE5646 – Programação para Web

Unidade III – Seção II

Aplicações Orientadas a Serviços

Prof. Frank Siqueira – Turma A

Prof. Leandro Komosinski – Turma B

Conteúdo

- Serviços Web
- SOAP
  - WSDL
  - UDDI
  - APIs
- REST
  - Métodos
  - JSON
  - APIs
- Informações Adicionais

Aplicações Orientadas a Serviços

- Servidores de aplicação podem hospedar:
  - Páginas Web e outros conteúdos estáticos
  - Páginas Web e outros conteúdos gerados dinamicamente
  - Aplicações Web completas
  - **Serviços!!!**

Um serviço provê uma funcionalidade que pode ser utilizada para construção de sistemas, geração de páginas dinâmicas, integração de sistemas, etc.

Aplicações Orientadas a Serviços

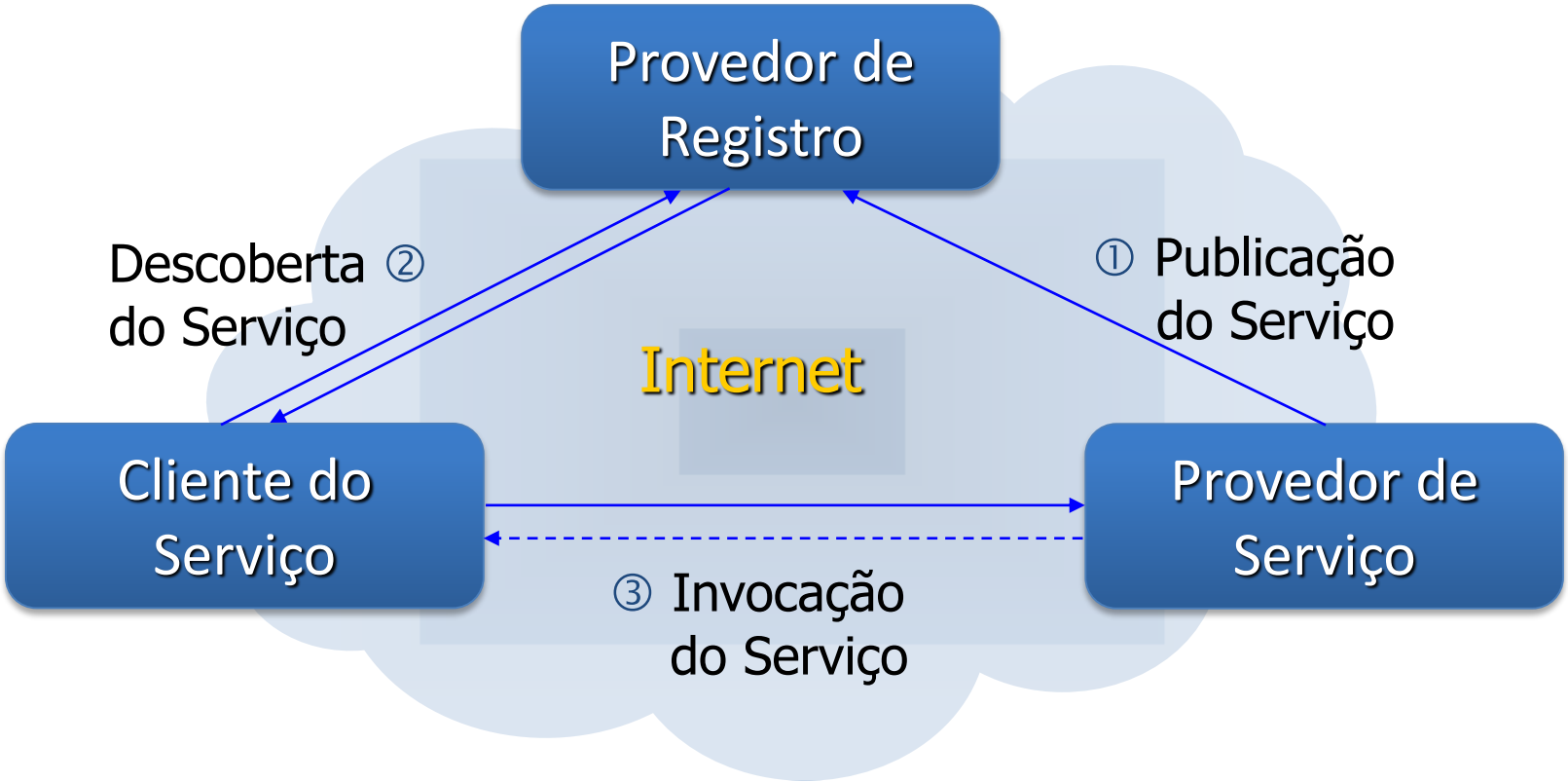
- Vantagens da Orientação a Serviços
  - O serviço encapsula a complexidade necessária para sua execução, permitindo a sua invocação através de uma interface bem definida
  - Serviços podem ser facilmente reutilizados em diversos sistemas computacionais
  - Serviços podem ser acessados remotamente por parceiros de negócio
  - Permitem a integração de sistemas heterogêneos, pois independem de linguagem e de plataforma

Aplicações Orientadas a Serviços

- Arquitetura Orientada a Serviços (SOA)
  - Composta pelos seguintes elementos:
    - Provedor do serviço (*service provider*)
    - Provedor de registro (*registry provider*)
    - Cliente do serviço (*service requestor*)
  - Interação entre os elementos:
    - Publicação de serviços: provedor de serviço informa ao provedor de registro que está fornecendo um serviço
    - Descoberta de serviços: cliente pergunta ao provedor de registro quem fornece um determinado serviço
    - Invocação de serviços: cliente solicita ao provedor de serviço que execute o serviço fornecido

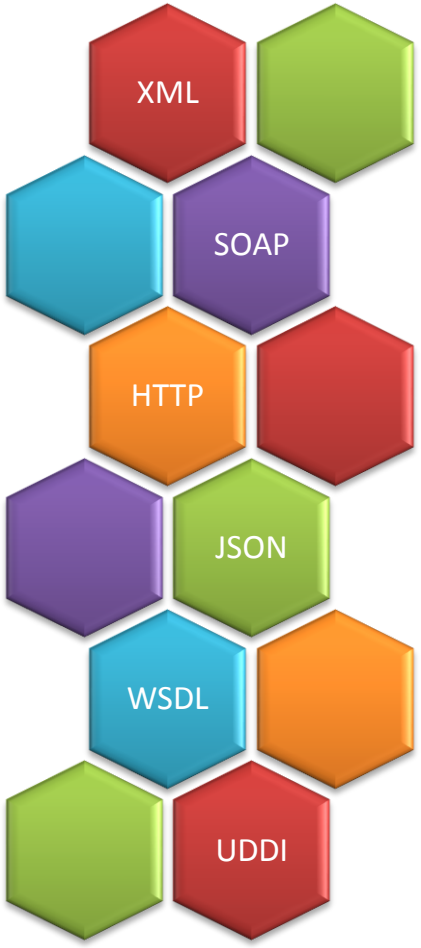
Aplicações Orientadas a Serviços

- Elementos da Arquitetura SOA



Serviços Web

A tecnologia de Serviços Web consiste em uma solução baseada em padrões abertos utilizados na Web que permite a criação de aplicações baseadas na Arquitetura Orientada a Serviços (SOA)



Serviços Web

- Definições

"Serviços Web são aplicações modulares autodescritas e autocontidas, que podem ser conectadas e acopladas a outros web services." [IBM]

"Serviços Web são componentes de software com baixo fator de acoplamento, utilizados por meio de padrões de tecnologia Internet. Um Serviço Web representa uma função de negócio ou um serviço que pode ser acessado por uma outra aplicação." [Gartner]

Serviços Web

- Principais Características
  - Fornecem serviços a clientes dispersos na Web
  - Podem ser facilmente localizados na rede
  - Interfaces são bem definidas e auto-descritas
  - Empregam padrões da Internet
    - Formato de dados
    - Protocolo de comunicação
  - Consiste em uma tecnologia aberta, independente de linguagem e de plataforma

Serviços Web

- Utilização
  - Disponibilização de serviços na Web (para uso próprio, restrito ou aberto)
  - Construção de aplicações Web a partir de serviços pré-existentes (próprios ou de terceiros)
  - Integração de sistemas: sistemas legados, sistemas heterogêneos, bases de dados, etc.
  - Implementação de lógica de negócio no servidor de aplicação

    …

Serviços Web

- Vantagens sobre outras tecnologias
  - Adota padrões abertos de ampla aceitação no mercado
  - Emprega protocolos que não são bloqueados pelos *firewalls* da rede
  - Infraestrutura para desenvolvimento e execução já está disponível e é conhecida pelos programadores
  - Aplicações podem ser desenvolvidas facilmente usando diversas linguagens de programação

Serviços Web

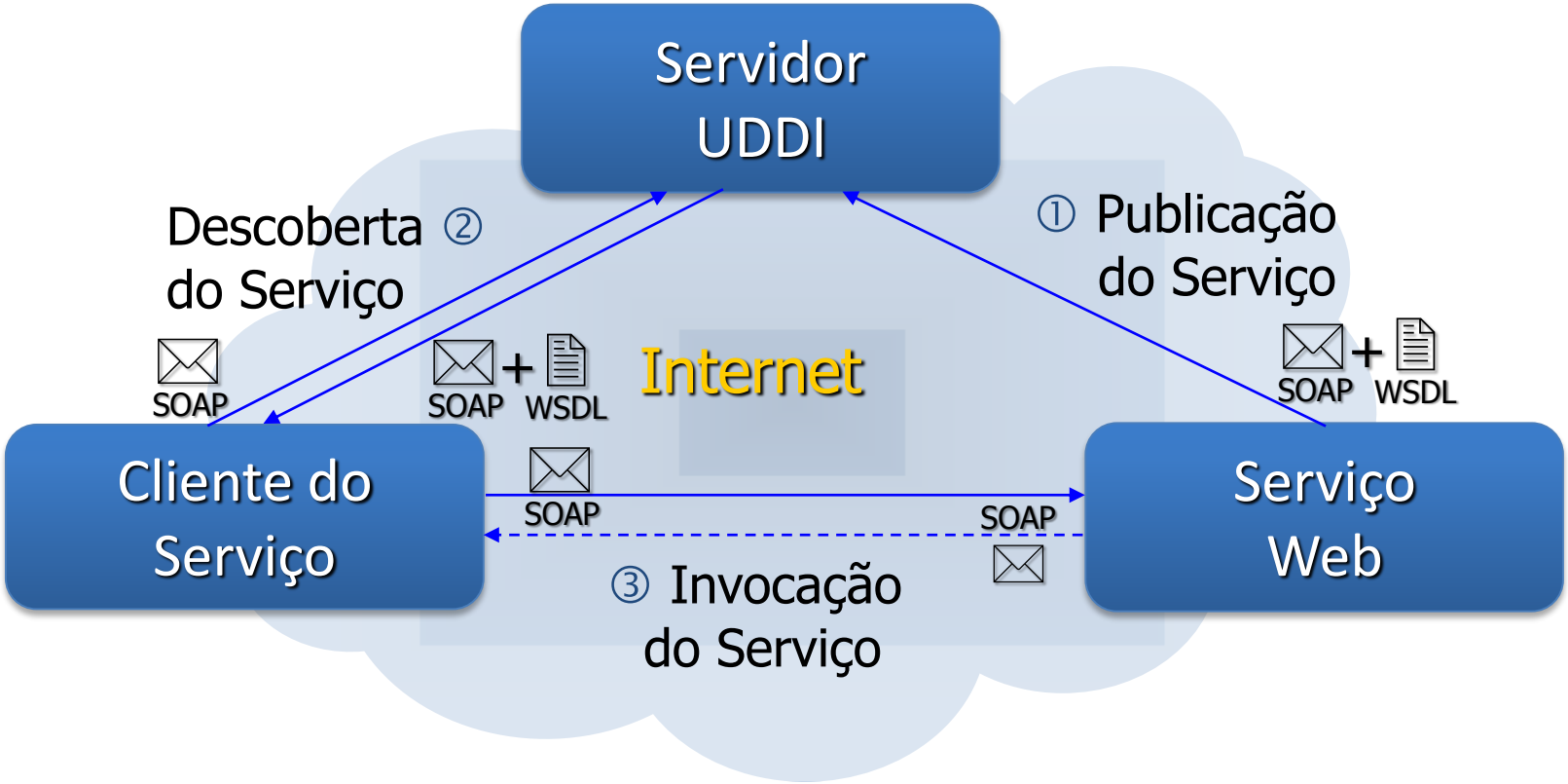
- Serviços Web podem ser baseados nos seguintes padrões/arquiteturas:
  - SOAP
    - Serviços Web “tradicionais”
    - Seguem padrões abertos definidos por W3C e OASIS
    - São mais “pesados” (desempenho pior)
    - Voltados principalmente à integração de sistemas
  - REST
    - Serviços Web “leves”
    - Voltados principalmente a aplicações Web

Serviços Web

- Os Serviços Web “tradicionais” são baseados nas seguintes tecnologias:
  - SOAP: protocolo utilizado na interação com os serviços Web
  - WSDL (*Web Services Description Language*): linguagem utilizada para descrever os serviços Web
  - UDDI (*Universal Description, Discovery and Integration*): permite localizar serviços na rede

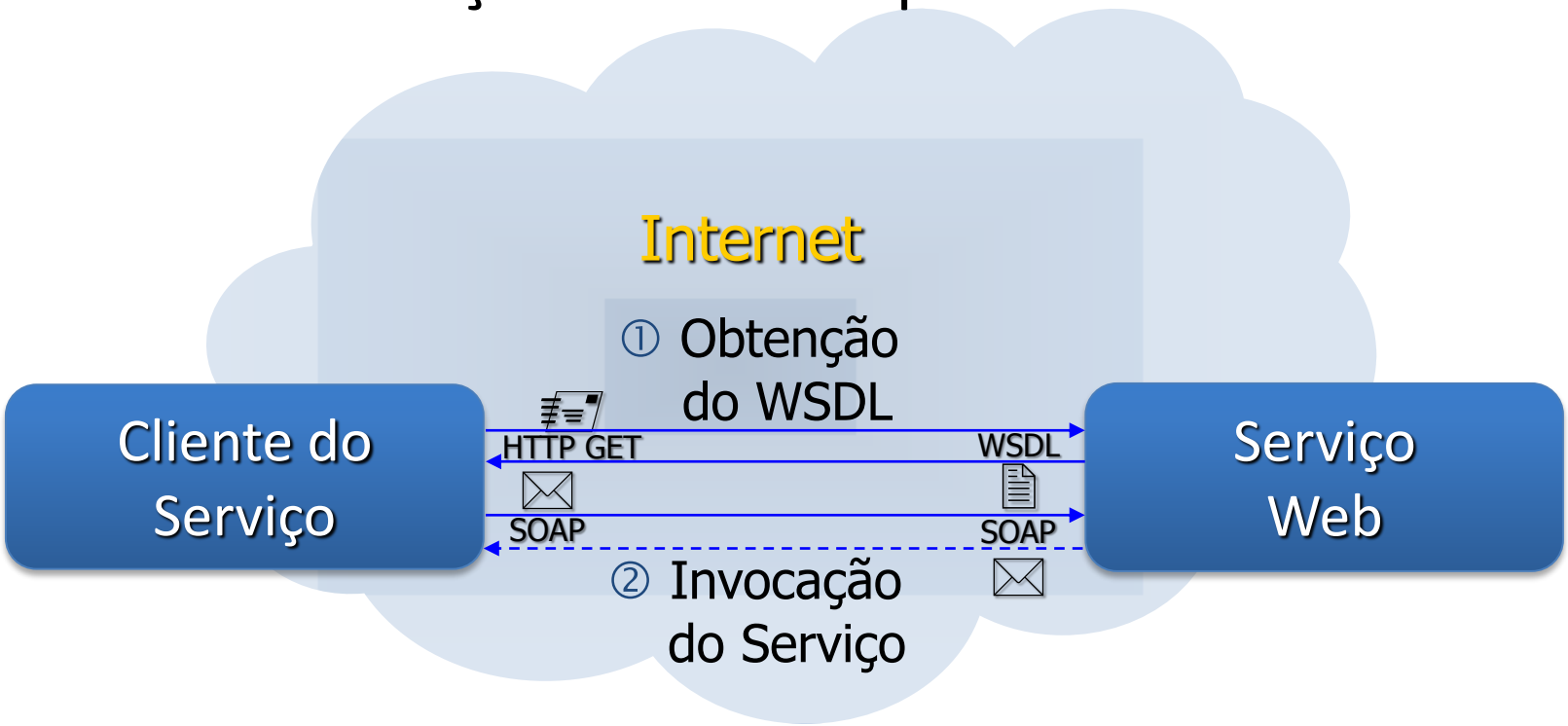
SOAP

- Serviços Web SOAP e a Arquitetura SOA



SOAP

- Interação Direta (sem UDDI)
  - WSDL obtido enviando uma requisição HTTP à URL do Serviço Web com o parâmetro ?wsdl



SOAP

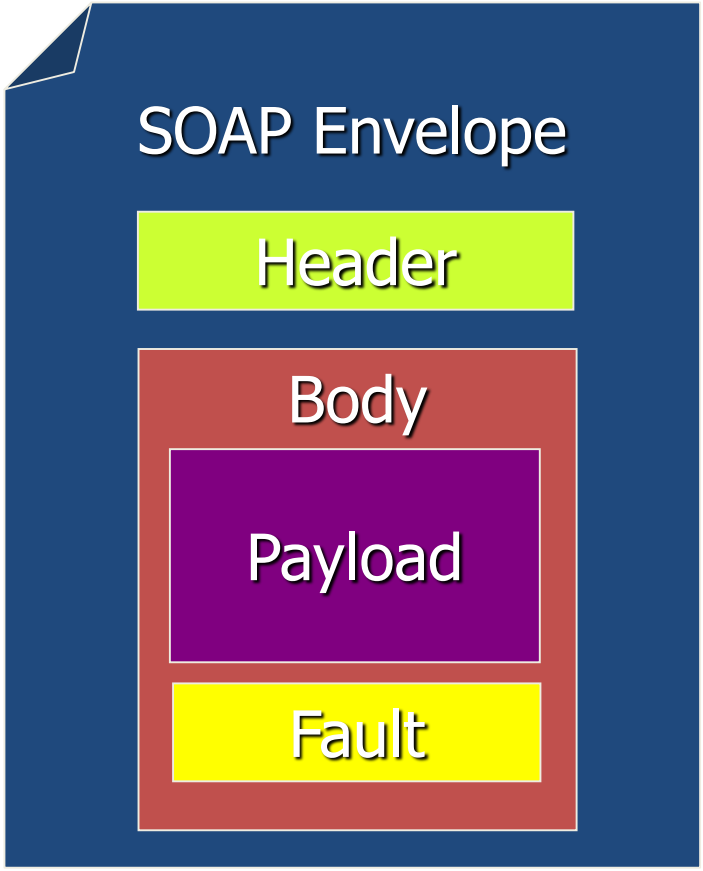
- Protocolo SOAP
  - Protocolo definido pelo W3C para comunicação entre serviços
  - Nome originou das iniciais de *Simple Object Access Protocol* (esse nome não é mais usado)
  - Define o formato das mensagens trocadas entre serviços Web
  - Independente de plataforma e de linguagem
  - Utiliza em geral HTTP[S] como protocolo de transporte (porta 80 → atravessa *firewalls*)

SOAP

- Formato da Mensagem SOAP
  - A mensagem segue o formato XML
  - O *parsing* da mensagem pode ser feito usando diversas bibliotecas de linguagens
  - Os tipos de dados utilizados na mensagem podem ser definidos em um XSD (*XML Schema Definition*)
  - A mensagem é encapsulada em um “envelope”

SOAP

- Envelope SOAP



SOAP

- Funcionamento do Protocolo SOAP
  - Cliente cria um envelope SOAP especificando o nome da operação requisitada e os nomes e valores dos parâmetros da operação
  - Requisição é enviada pela rede ao provedor do serviço
  - Requisição é recebida e interpretada
  - A operação requisitada é executada
  - A resposta, se houver, é colocada em um envelope SOAP e enviada ao cliente

SOAP

- Exemplo de Requisição SOAP

```
<?xml version="1.0" encoding="UTF-8"?>
<S:Envelope xmlns:S="http://schemas.xmlsoap.org/soap/envelope/">
<S:Header/>
<S:Body>
   <ns1:getTemperaturaMinima xmlns:ns1="http://ufsc.br/previsao">
      <localidade>Florianópolis</localidade>
    </ns1:getTemperaturaMinima>
</S:Body>
</S:Envelope>
```

SOAP

- Exemplo de Resposta SOAP

```
<?xml version="1.0" encoding="UTF-8"?>
<S:Envelope xmlns:S="http://schemas.xmlsoap.org/soap/envelope/">
<S:Body>
   <ns1:getTemperaturaMinimaResponse
      xmlns:ns1="http://ufsc.br/previsao">
      <return>13.2</return>
   </ns1:getTemperaturaMinimaResponse>
</S:Body>
</S:Envelope>
```

SOAP

- Definição de Tipos em um XSD

```
<xs:schema xmlns:tns="http://ufsc.br/previsao"
                xmlns:xs="http://www.w3.org/2001/XMLSchema"
                version="1.0" targetNamespace="http://ufsc.br/previsao">
<xs:element name="getTemperaturaMinima"
                type="tns:getTemperaturaMinima" />
<xs:element name="getTemperaturaMinimaResponse"
                type="tns:getTemperaturaMinimaResponse" />
<xs:complexType name="getTemperaturaMinima">
  <xs:sequence>
    <xs:element name="localidade" type="xs:string" minOccurs="0"/>
  </xs:sequence>
</xs:complexType>
<xs:complexType name="getTemperaturaMinimaResponse">
  <xs:sequence>
    <xs:element name="return" type="xs:float" />
  </xs:sequence>
</xs:complexType>
...
</xs:schema>
```

WSDL

- WSDL (*Web Services Description Language*)
  - Padrão do W3C
  - Baseado no XML
  - Especifica a interface de um serviço Web
  - Através do WSDL de um Serviço Web é possível saber quais serviços estão disponíveis e como invocá-los remotamente
  - A especificação WSDL é independente da linguagem na qual o Serviço Web é implementado

WSDL

- Estrutura do WSDL

```
<?xml version="1.0" encoding="UTF-8"?>
<definitions name="PrevisaoDoTempo"
             targetNamespace="http://ufsc.br/previsao"
             xmlns:tns="http://ufsc.br/previsao"
             xmlns:xsd="http://www.w3.org/2001/XMLSchema"
             xmlns:soap="http://www.w3.org/2003/11/wsdl/soap12"
             xmlns="http://www.w3.org/2003/11/wsdl">
+ <types>
+ <message name="getTemperaturaMinima">
+ <message name="getTemperaturaMinimaResponse">
+ <message name="getTemperaturaMaxima">
+ <message name="getTemperaturaMaximaResponse">
+ <portType name="Tempo">
+ <binding name="TempoPortBinding" type="tns:Tempo">
+ <service name="TempoService">
</definitions>
```

WSDL

- Elementos do WSDL
  - <definitions>: elemento raiz
  - <types>: define os tipos de dados utilizados pelo serviço Web (pode referenciar um XSD)
  - <messages>: especifica as mensagens usadas na comunicação com o serviço Web
  - <portType>: define um conjunto de operações que são executadas por um serviço
  - <binding>: associa um protocolo ao serviço
  - <service>: especifica o endereço de rede no qual o serviço pode ser acessado

WSDL

- WSDL: Definição de Tipos
  - Importa um XSD com a descrição dos tipos

```
<types>
<xsd:schema>
  <xsd:import namespace="http://ufsc.br/previsao"
    schemaLocation="http://ufsc.br/previsao/tempo.xsd" />
</xsd:schema>
</types>
```

WSDL

- WSDL: Definição de Mensagens

```
<message name="getTemperaturaMinima">
  <part name="parameters" element="tns:getTemperaturaMinima" />
</message>
<message name="getTemperaturaMinimaResponse">
  <part name="parameters"
        element="tns:getTemperaturaMinimaResponse"/>
</message>
<message name="getTemperaturaMaxima">
  <part name="parameters" element="tns:getTemperaturaMaxima" />
</message>
<message name="getTemperaturaMaximaResponse">
  <part name="parameters"
        element="tns:getTemperaturaMaximaResponse"/>
</message>
```

WSDL

- WSDL: Definição de Porta

```
<portType name="Tempo">
  <operation name="getTemperaturaMinima">
    <input message="tns:getTemperaturaMinima" />
    <output message="tns:getTemperaturaMinimaResponse" />
  </operation>
  <operation name="getTemperaturaMaxima">
    <input message="tns:getTemperaturaMaxima" />
    <output message="tns:getTemperaturaMaximaResponse" />
  </operation>
 </portType>
```

WSDL

- WSDL: *Binding* com o Protocolo SOAP

```
<binding name="TempoPortBinding" type="tns:Tempo">
  <soap:binding transport="http://schemas.xmlsoap.org/soap/http"
    style="document" />
  <operation name="getTemperaturaMinima">
    <soap:operation soapAction="" />
      <input> <soap:body /> </input>
      <output> <soap:body /> </output>
  </operation>
  <operation name="getTemperaturaMaxima">
    <soap:operation soapAction="" />
      <input> <soap:body /> </input>
      <output> <soap:body /> </output>
  </operation>
</binding>
```

WSDL

- WSDL: Definição de Serviço

```
<service name="TempoService">
   <documentation>Serviço de Previsão do Tempo</documentation>
  <port name="TempoPort" binding="tns:TempoPortBinding">
    <soap:address location="http://ufsc.br/previsao/TempoService" />
  </port>
</service>
```

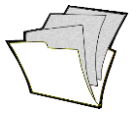
UDDI

- UDDI (*Universal Description, Discovery and Integration*)
  - Infra-estrutura para registro e localização de serviços Web
  - Padrão do OASIS que define um provedor de registros de Serviços Web
  - Criado por Ariba, IBM e Microsoft
  - Armazena as especificações WSDL dos provedores de serviços
  - Permite que os clientes encontrem os provedores dos serviços dos quais necessitam e descubram como solicitar tais serviços

UDDI

- Características do UDDI
  - Repositório centralizado e universal para registro de serviços
  - Interfaces registradas são descritas em WSDL
  - Registros são armazenados em XML
  - Recebe requisições de registro e descoberta utilizando o protocolo SOAP
  - Empresas podem ter seus servidores UDDI privados para registro de serviços internos

UDDI

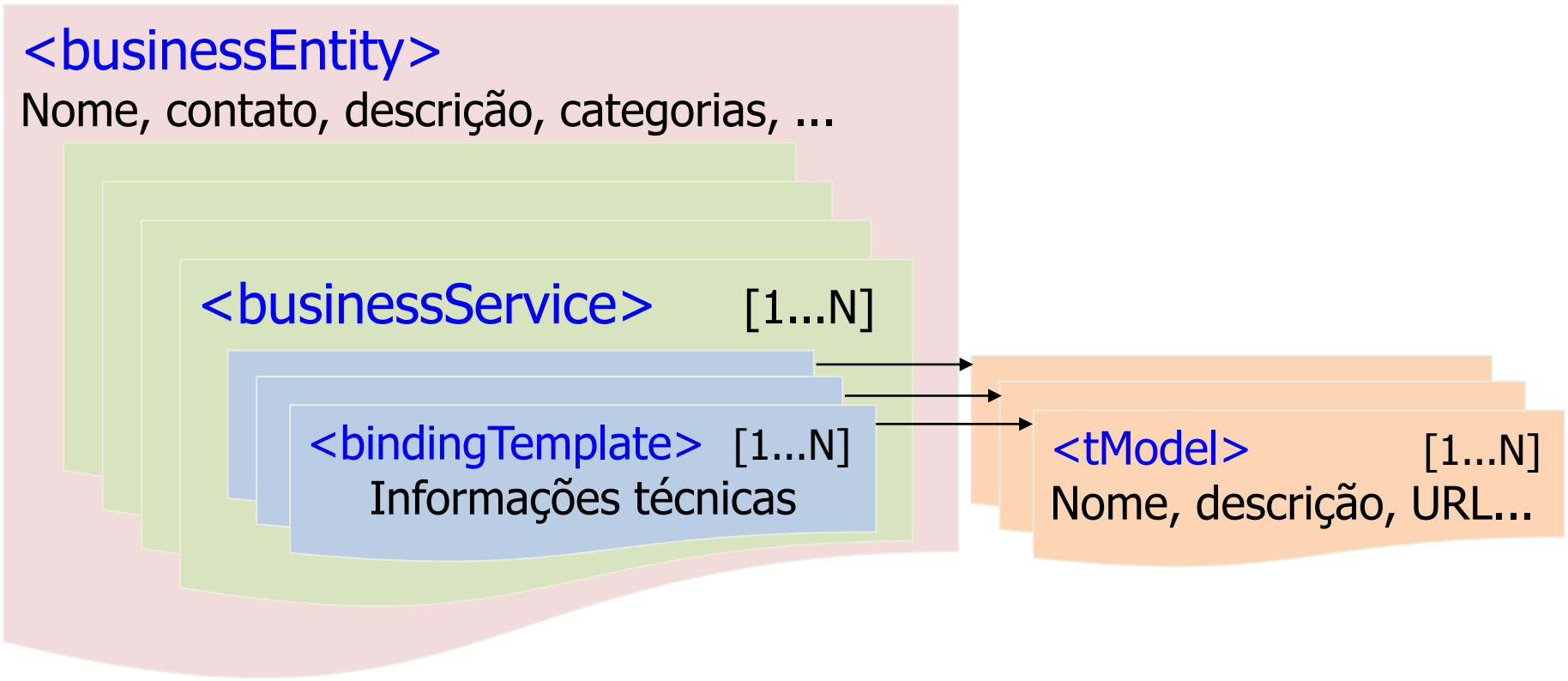
- Consulta de Registros no UDDI
  -  Páginas Brancas
    - Fornecem o endereço para contato do provedor do serviço
  -  Páginas Amarelas
    - Classificam os provedores em categorias de acordo com o seu ramo de negócio
  -  Páginas Verdes
    - Fornecem informações técnicas sobre os serviços executados pelos provedores

UDDI

- Elementos do Registro UDDI
  - <businessEntity>: fornece informações sobre uma família de serviços
  - <businessService>: provê informações sobre um determinado serviço
  - <bindingTemplate>: contém informações técnicas referentes a um serviço
  - <tModel>: fornece informações específicas relacionadas ao serviço

UDDI

- Registro UDDI



UDDI

- UDDI possui APIs para:

Localização de Serviços

- find_binding
- find_business
- find_service
- find_tModel
- get_bindingDetail
- get_businessDetail
- get_serviceDetail
- get_tModelDetail

Publicação de Serviços

- delete_binding
- delete_business
- delete_service
- delete_tModel
- save_binding
- save_business
- save_service
- save_tModel

APIs

- APIs para Desenvolvimento de Serviços Web baseados no padrão SOAP
  - Facilitam a implementação de serviços Web
  - Podem gerar automaticamente:
    - Proxies: processam as mensagens SOAP
    - WSDL do serviço web
    - Código para acesso ao UDDI
  - Exemplos:
    - Apache Axis
    - gSOAP
    - kSOAP
    - …

APIs

- JAX-WS (*Java API for XML Web Services*)
  - Fornece rotinas/classes para facilitar a manipulação de mensagens SOAP/XML, para criar/interpretar descrições de serviços em WSDL e para acesso/registro no UDDI
  - Engloba as seguintes APIs:
    - JAXP (*Java API for XML Processing*)
    - JAXB (*Java API for XML Binding*)
    - JAX-RPC (*Java API for XML RPC*)
    - SAAJ (*SOAP with Attachments API for Java*)
    - JAXR (*Java API for XML Registries*)

APIs

- Exemplo de uso do JAX-WS

```
@WebService()
public class Tempo {
  @WebMethod(name="getTemperaturaMinima")
  public float getTemperaturaMinima(
      (@WebParam="nomeLocalidade") String nomeLocalidade )
  { /* código do método */ }

  @WebMethod(name="getTemperaturaMaxima")
  public float getTemperaturaMaxima(
      (@WebParam="nomeLocalidade") String nomeLocalidade )
  { /* código do método */ }
}
```

REST

- REST (*REpresentational State Transfer*)
  - Alternativa mais leve para o SOAP
  - É um estilo arquitetural, e não um padrão especificado formalmente, como o SOAP
  - Objetivos: simplicidade e melhor desempenho
  - Recursos mantidos no servidor são associados a uma URL e acessados através de mensagens HTTP
  - Dados podem ser representados em XML ou em formatos mais leves e compactos, como o JSON

REST

- Operações
  - Recursos são manipulados através de operações CRUD (Create, Read, Update & Delete), que são mapeadas para métodos do protocolo HTTP

Operação	Método HTTP	Descrição
Create	POST	Cria um recurso
	PUT	Modifica uma coleção de recursos
Read	GET	Obtém um recurso ou coleção
Update	PUT	Modifica ou cria um recurso
Delete	DELETE	Remove um recurso ou coleção

REST

- Uso de URLs em Serviços Web RESTful
  - Recursos são associados a URLs
  - Parâmetros podem ser passados na URL ou no corpo das mensagens PUT, POST
  - Exemplo – Serviço REST de uma loja virtual:

Método	URL: /loja/produtos	URL: /loja/produtos/{*id*}
GET	Retorna lista de produtos	Retorna dados do produto
PUT	Altera a lista de produtos	Cria/altera dados do produto
POST	Adiciona produto à lista	Geralmente não utilizado
DELETE	Remove a lista de produtos	Remove o produto especificado

REST

- REST x SOAP
  - Mais leve: usa diretamente os métodos do HTTP, evitando uma camada de protocolo adicional
  - Mais compacto: SOAP requer uso de XML, o que torna as mensagens desnecessariamente grandes
  - Mais eficiente: permite *caching* das respostas
  - Mais simples: é fácil de entender, desenvolver, testar e utilizar
  - Modelo de dados: manipula o estado de recursos/dados mantidos no servidor, enquanto SOAP trabalha com serviços/funcionalidades

JSON

- JSON (*JavaScript Object Notation*)
  - Formato de dados muito usado em aplicações REST
  - Existem *parsers* JSON em várias linguagens (não somente em *JavaScript*)
  - Comparação: JSON x XML
    - JSON é mais compacto: 30 a 40% menor que XML
    - JSON é mais leve: seu processamento (*parsing*) requer menos recursos computacionais
    - XML é mais completo: existem padrões para definição de esquemas de dados, espaços de nomes, criptografia, assinatura digital, especificação semântica, etc.

JSON

- JSON – Exemplo:

```
{ "loja": "Loja de Informática",
  "data": "01/04/2013",
  "catalogo": [
   { "id": "1",
     "nome": "CD-R",
     "preco": "0.99" },
   { "id": "2",
     "nome": "DVD-R",
     "preco": "1.49" },
    ... ]
}
```

APIs

- JAX-RS (*Java API for RESTful Services*)
  - Provê anotações para criação de serviços RESTful a partir de classes Java
  - Principais anotações:
    - @Path: caminho associado a uma classe/método
    - @GET, @POST, @PUT e @DELETE: associam um método HTTP a um método Java correspondente
    - @Consumes, @Produces: especificam os tipos de dados MIME aceitos/gerados pelo método/classe
    - @PathParam, @FormParam, @HeaderParam, etc.: associam o valor de um parâmetro do método Java a parte da URL, campo de formulário, cabeçalho, etc.

APIs

- Exemplo de uso do JAX-RS

```
@Path("/loja/")
public class Loja {
    @GET
    @Path("/produtos/{id}")
    @Produces("application/json")
    public Produto getProduto(@PathParam("id") String id) { ... }

    @POST
    @Path("/produtos")
    @Consumes("application/json")
    public javax.ws.rs.core.Response adicionaProduto(Produto produto) { ... }
    ...
}
```

APIs

- Vários sites da Web (redes sociais, lojas virtuais, sites de busca, etc.) fornecem APIs REST para acesso a seus serviços
  - Exemplo: REST API do Twitter
    - GET search.json?q=UFSC
      retorna *tweets* contendo a palavra "UFSC" (link)
    - POST direct_messages/new.json
      user_id=<*user-id*>&text=<*message*>
      envia mensagem para usuário (requer autenticação)

Informações Adicionais

- Especificações que adicionam recursos e funcionalidades aos Web Services
  - WS-Addressing: endereçamento e roteamento de mensagens na camada de aplicação
  - WS-BPEL/WS-CDL: orquestração/coreografia de processos de negócio
  - WS-Coordination/WS-Transaction: execução de transações distribuídas entre serviços
  - WS-ReliableMessaging: entrega confiável de mensagens SOAP a serviços Web
  - WS-Security: mecanismos para controle de acesso, integridade e confidencialidade
  - ...

Informações Adicionais

- Serviços Web Semânticos
  - Aperfeiçoam a descrição e descoberta de serviços usando tecnologias da Web Semântica
  - Propiciam a interação entre serviços sem intervenção humana
  - Dados e serviços são descritos usando ontologias, que representam um conjunto de conceitos dentro de um domínio de aplicação
  - Padrões relacionados:
    - OWL-S: *Ontology Web Language for Services*
    - WSMO: *Web Services Modeling Ontology*
    - SAWSDL: *Semantic Annotations for WSDL*