Webservices

LEANDRO MENDES FERREIRA

Webservices - Agenda

- CONCEITOS
- RPC
- HTTP
- SOAP
- RESTFUL

CONCEITOS

Conceitos Básicos

- XML (eXtensible Markup Language)
  - Liguagem de marcação para descrição de documentos de forma auto contida.
- JSON (JavaScript Object Notation)
  - Liguagem de marcação para descrição de documentos de forma auto contida. Baseada em Java Script

XML x JSON

```
{
    "glossary": {
        "title": "example glossary",
                "GlossDiv": {
            "title": "S",
                        "GlossList": {
                "GlossEntry": {
                    "ID": "SGML",
                                            "SortAs": "SGML",
                                            "GlossTerm": "Standard Generalized Markup Language",
                                            "Acronym": "SGML",
                                            "Abbrev": "ISO 8879:1986",
                                            "GlossDef": {
                        "para": "A meta-markup language, used to create markup languages such as DocBook.",
                                                    "GlossSeeAlso": ["GML", "XML"]
                    },
                                            "GlossSee": "markup"
                }
            }
        }
    }
}
```

The same text expressed as XML:

```
<!DOCTYPE glossary PUBLIC "-//OASIS//DTD DocBook V3.1//EN">
 <glossary><title>example glossary</title>
  <GlossDiv><title>S</title>
   <GlossList>
    <GlossEntry ID="SGML" SortAs="SGML">
     <GlossTerm>Standard Generalized Markup Language</GlossTerm>
     <Acronym>SGML</Acronym>
     <Abbrev>ISO 8879:1986</Abbrev>
     <GlossDef>
      <para>A meta-markup language, used to create markup
languages such as DocBook.</para>
      <GlossSeeAlso OtherTerm="GML">
      <GlossSeeAlso OtherTerm="XML">
     </GlossDef>
     <GlossSee OtherTerm="markup">
    </GlossEntry>
   </GlossList>
  </GlossDiv>
 </glossary>
```

Conceitos Básicos

- API – (Application Programming Interface)
  - Interface de comunicação de programas
- HTTP – (**Hypertext Transfer Protocol)**
  - Protocolo de comunicação e transferencia de arquivos da internet
- URI – (*Uniform Resource Identifier)*
  - Unidade de indentificação de algum recurso
- URL – (Uniform Resource Locator)
  - Indentificador de Endereço (Local – Host)

O que é WebServices? By W3C

- *"***Web service*** é uma solução utilizada na integração de sistemas e na comunicação entre aplicações diferentes. Com esta tecnologia é possível que novas aplicações possam interagir com aquelas que já existem e que sistemas desenvolvidos em plataformas diferentes sejam compatíveis. Os *Web services* são componentes que permitem às aplicações enviar e receber dados. Cada aplicação pode ter a sua própria "linguagem", que é traduzida para uma linguagem universal, um formato intermediário como *XML*', ***Json, CSV, etc*.”**

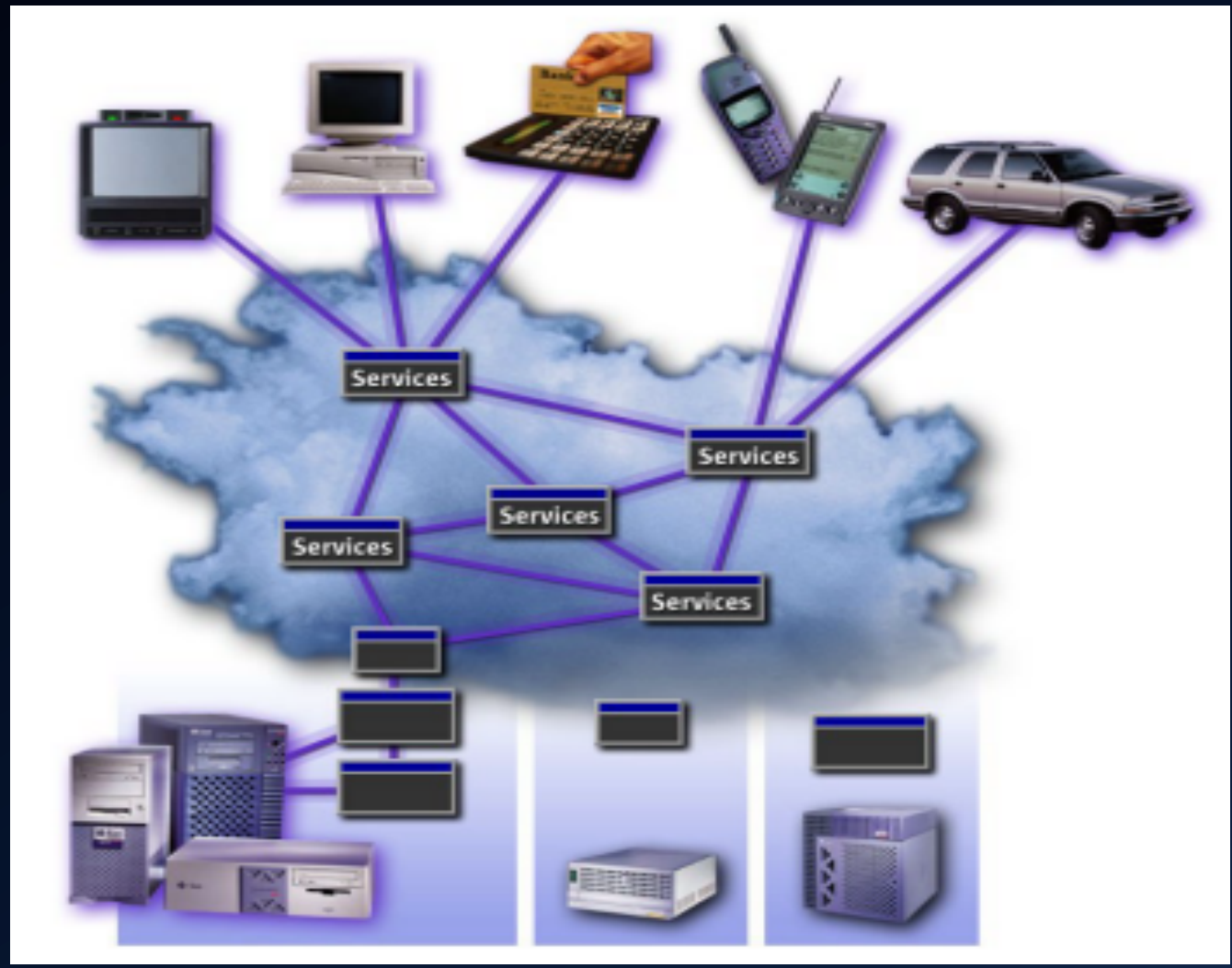
https://pt.wikipedia.org/wiki/Web_service

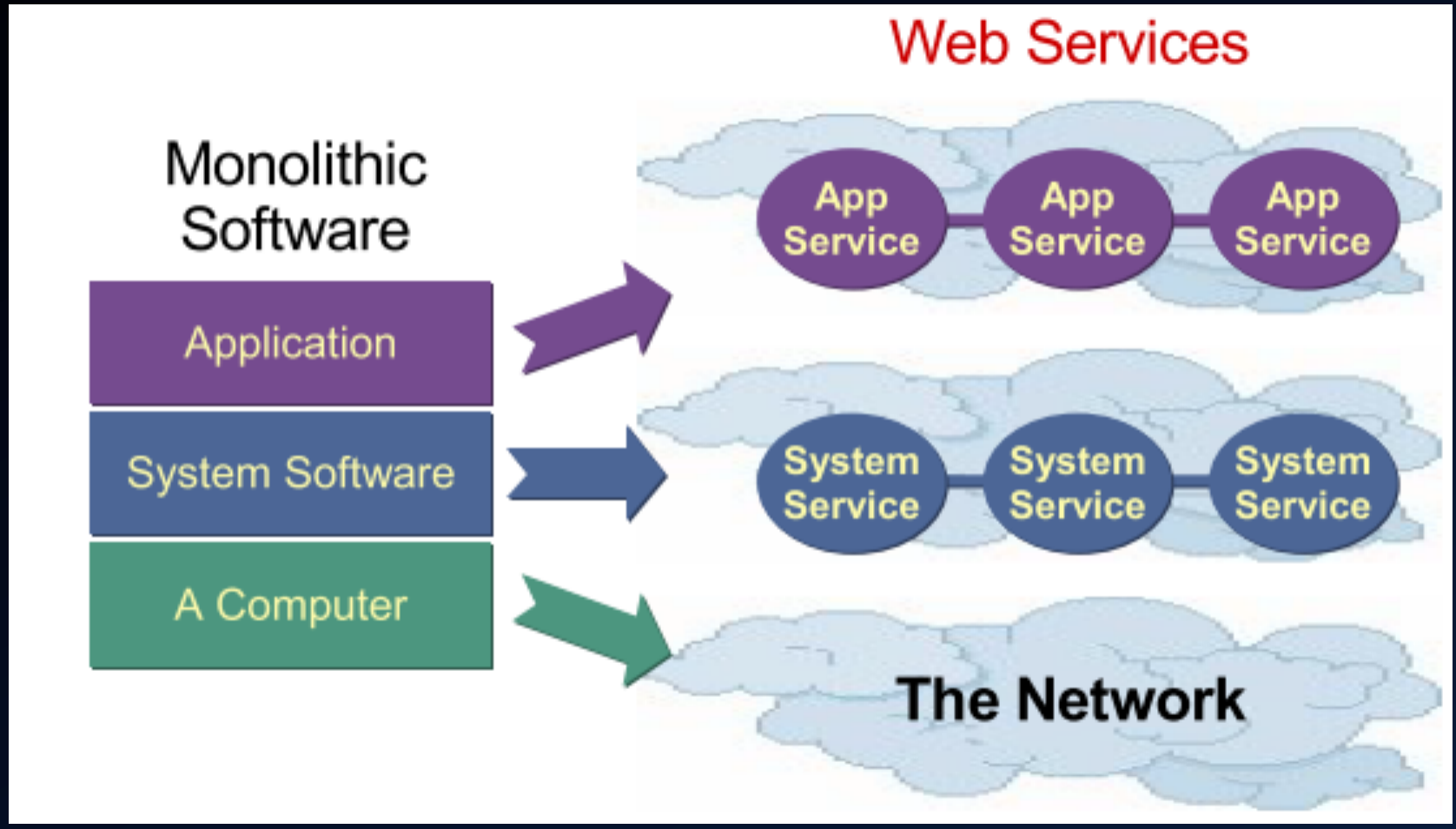
Introdução

- O que é WebServices ?
  - É uma tecnologia de invocação de serviços
  - Identificado através de uma URI
  - É composto de interfaces e implementações capazes de receber e enviar informações através de mensagens XML, JSON, TEXT, etc
  - Suporta interação direta com outros softwares
  - Comunicação via protocolo da Internet

Por que WebServices?

1. Plataforma neutra
2. São acessíveis de maneira padrão
3. Possuem interoperabilidade
4. São relativamente fáceis/simples
5. Simplifica integração nas empresas



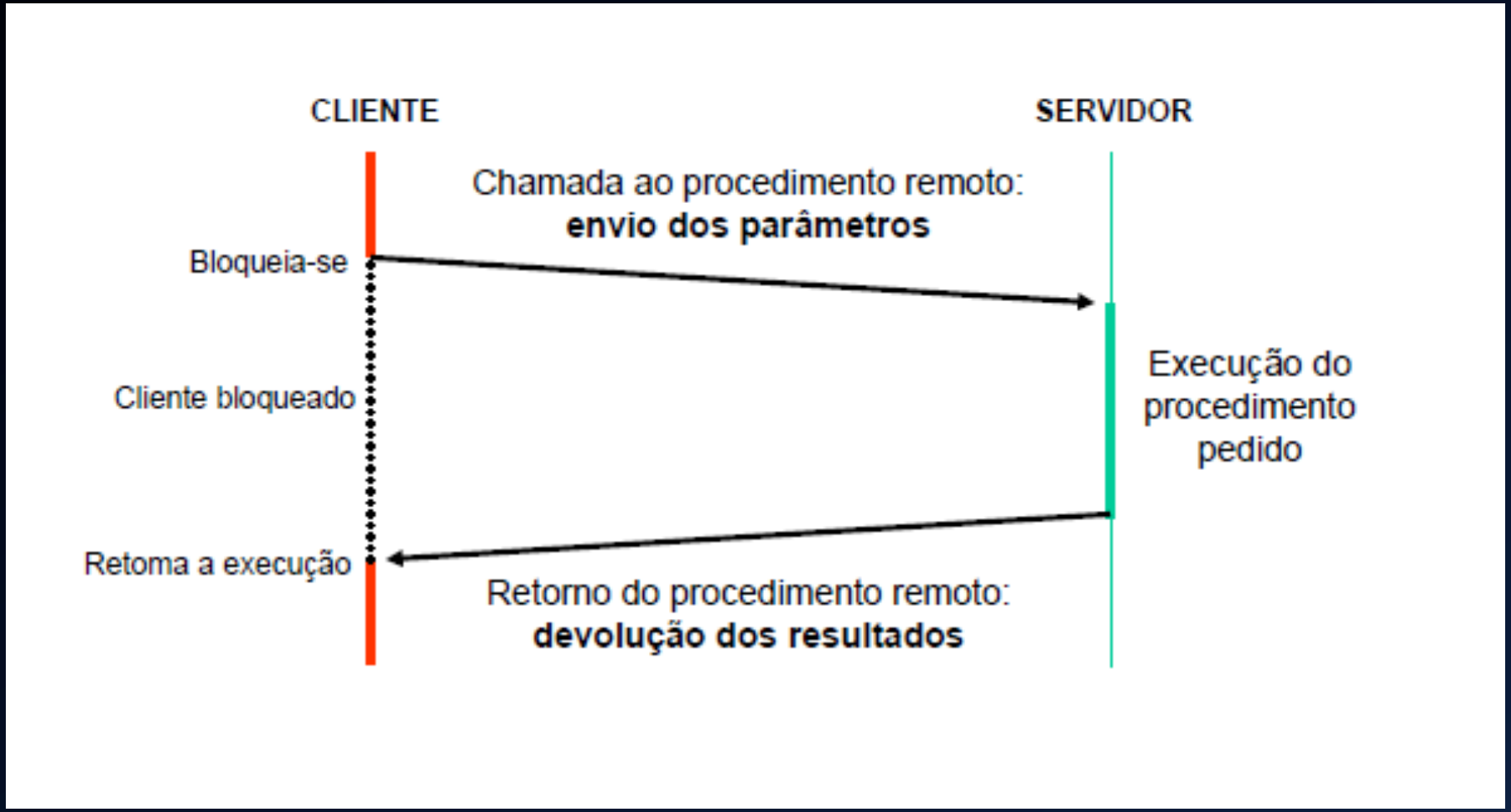

Web Services
Monolithic Software
Application
System Software
A Computer
App Service
App Service
App Service
System Service
System Service
System Service
The Network

RPC

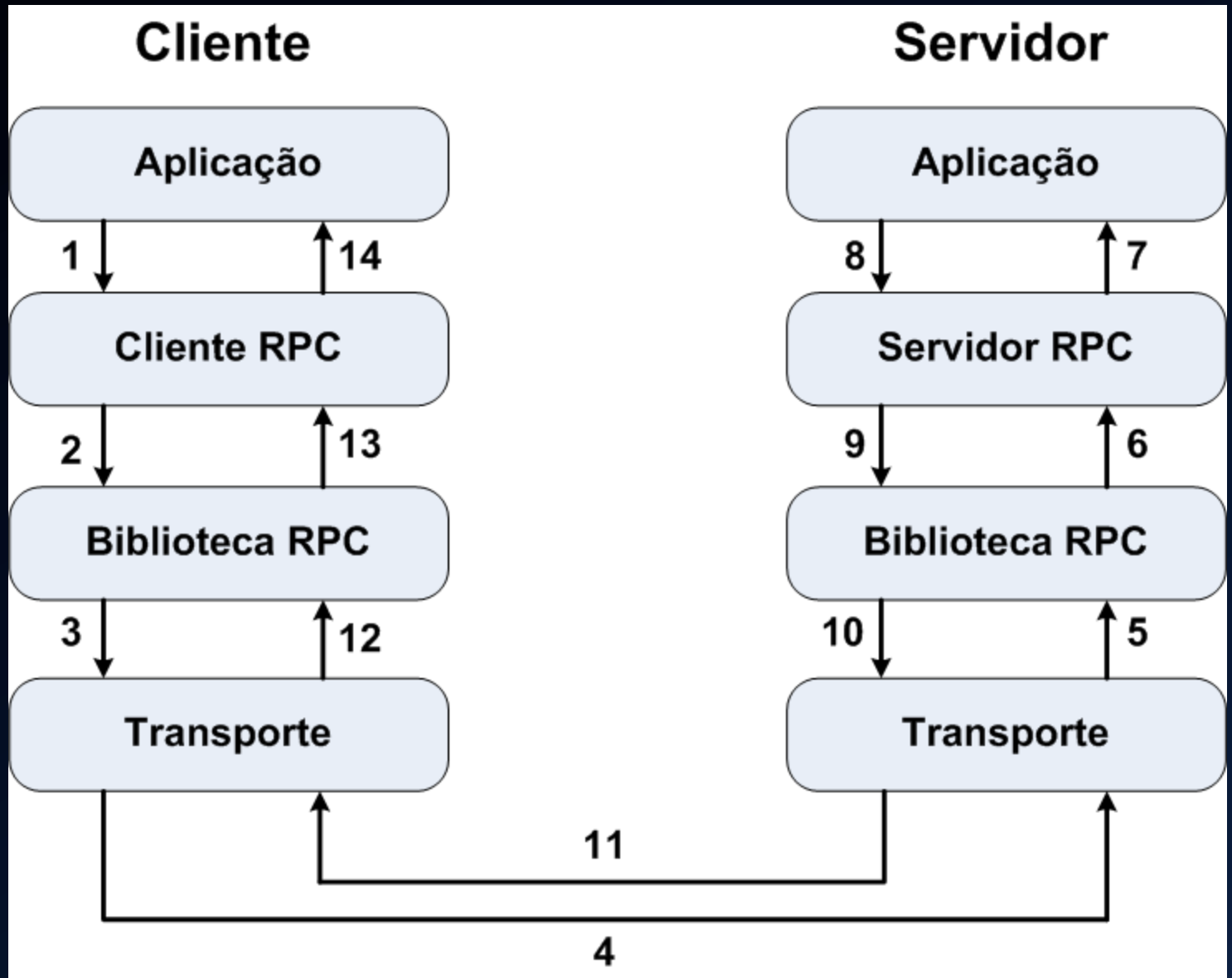
RPC – Remote Procedure Call

- Primeira conceito de processamento remoto
- *Independente de protocolo de comunicação*
  - *HTTP, TCP, UDP, SMTP*
- Base para muitos conceitos de webservice
  - CORBA, XML-RPC, SOAP
- Imprementado até sobre Sistemas Operacionais e Sistemas de Arquivos
  - NFS, Linux RPC, DCOM

RPC – Remote Procedure Call



RPC – Remote Procedure Call

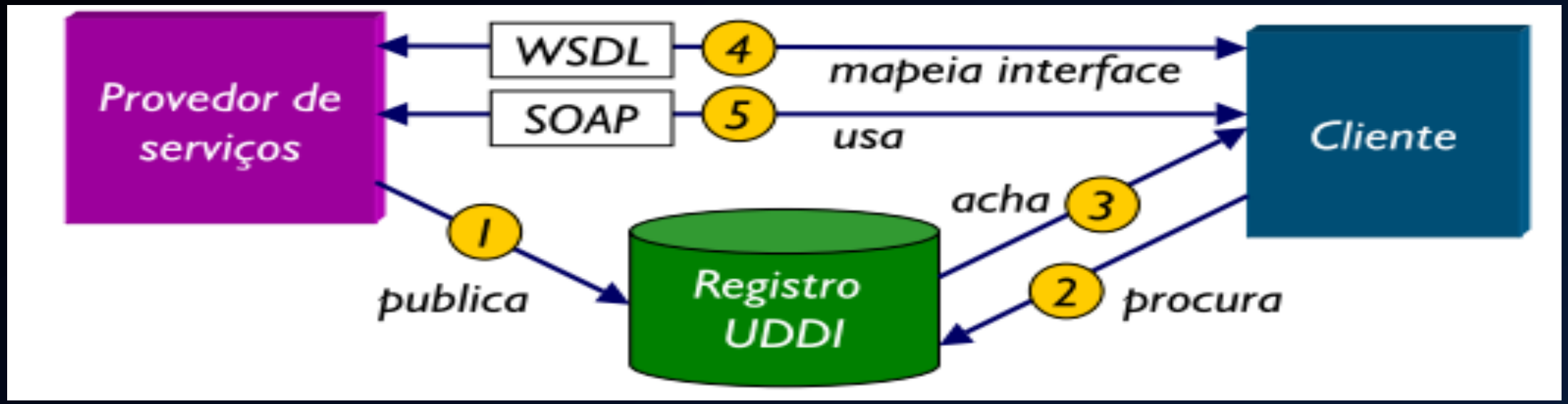


SOAP

Introdução - Computação Distribuída

- Arquitetura WebServices / SOAP
  1. Interação de programa para programa
  2. Tem a possibilidade de realizar uma integração entre componentes dinamicamente
  3. Possibilidade de agregação de serviços
  4. Mensagens baseadas em XML
  5. Baseada em padrões da indústria (fomentadas pelos grandes “players” de mercado)

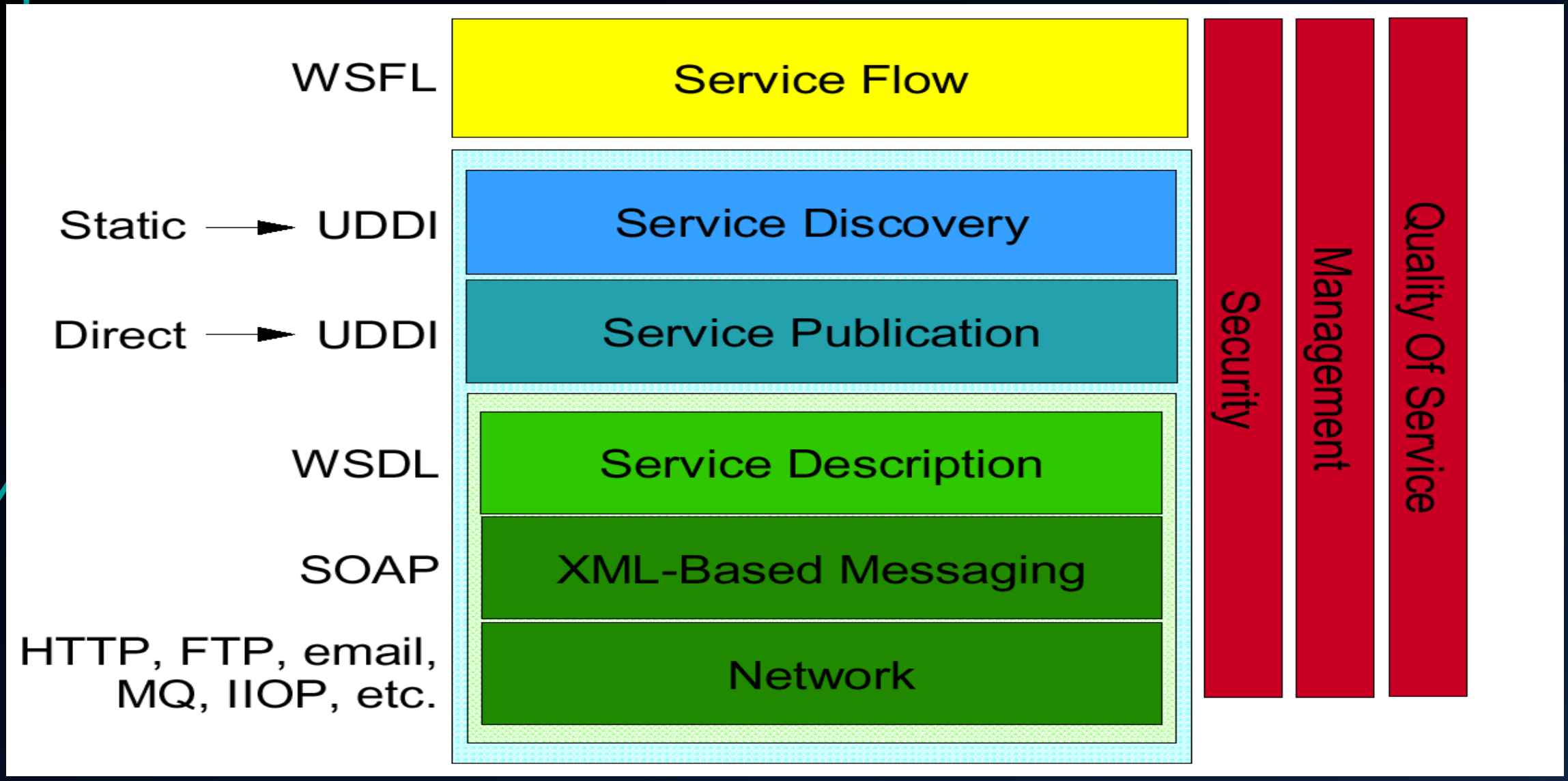
Arquitetura Simplificada de WebServices



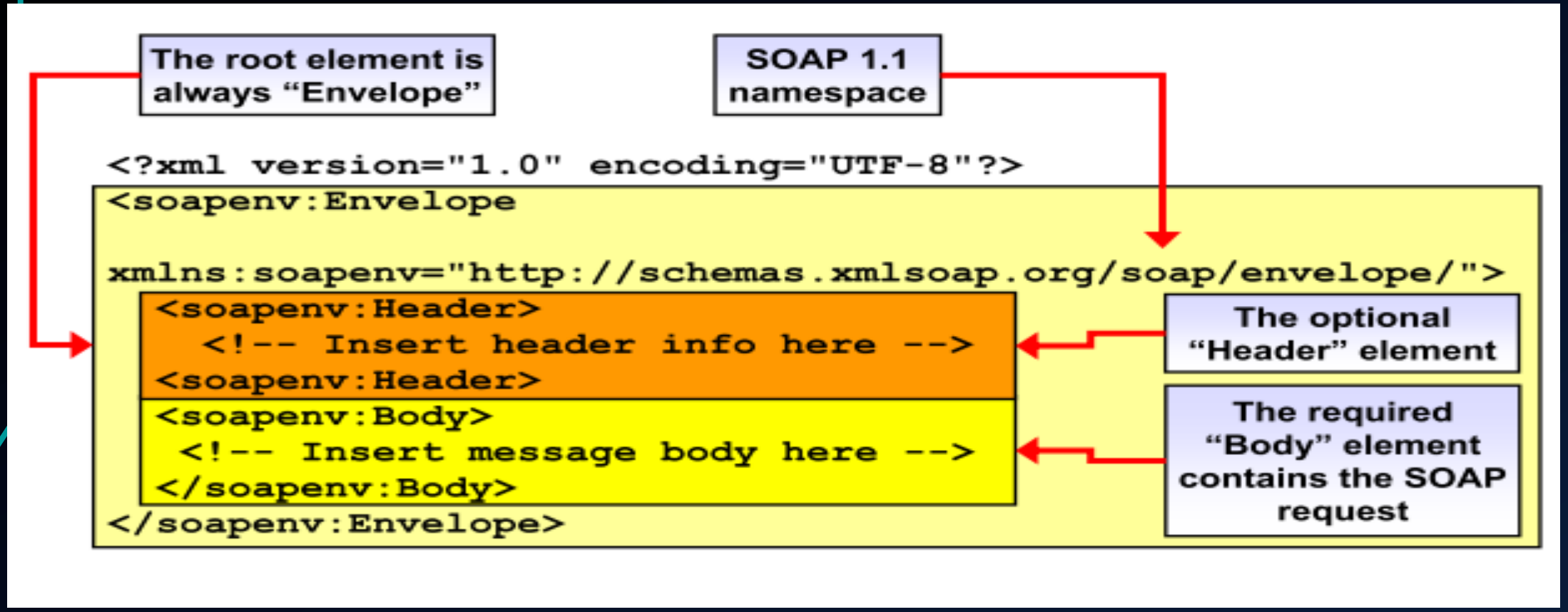
WebServices comparação com tecnologias Java RPC

	Java RMI	CORBA	RMI / IIOP	Web Services
Registro	RMI Registry	COS Naming	JNDI	UDDI
Descrição de Serviços	Java	OMG IDL	Java	WSDL
Transporte	Java RMI	IIOP	IIOP	SOAP

WebServices- pilha de Componentes



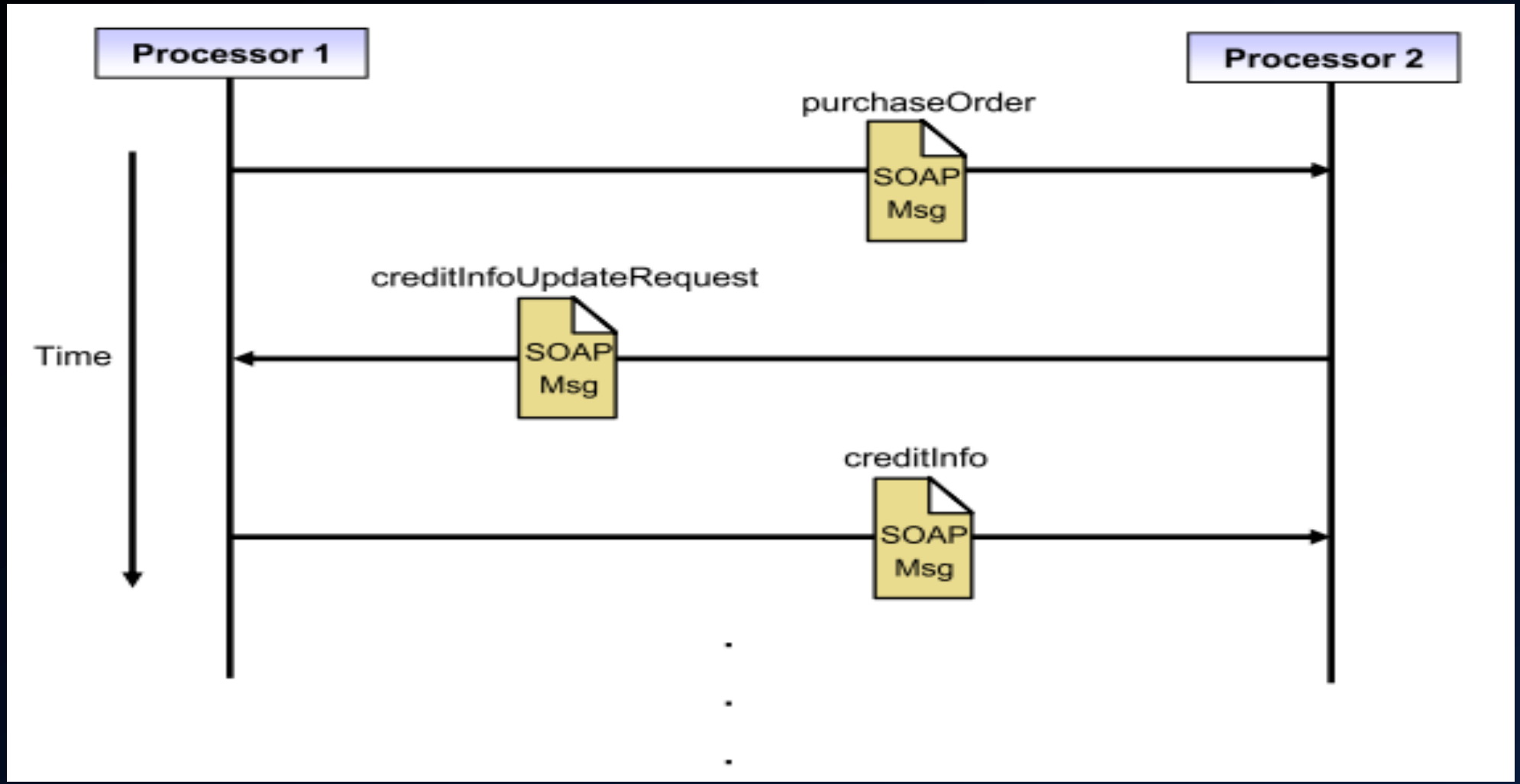
Estrutura da Mensagem SOAP



Exemplo de mensagem SOAP

```
<?xml version="1.0" encoding="UTF-8"?>
<soapenv:Envelope
 xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
 xmlns:q0="http://www.example.com/">
  <soapenv:Header>
    <q0:logMode
        SOAP-ENV:actor="http://somecompany.com/log"
        SOAP-ENV:mustUnderstand="1">
      Secure
    </q0:logMode>
  <soapenv:Header>
  <soapenv:Body>
    <q0:Register>
      <q0:FirstName>John</q0:FirstName>
      <q0:LastName>Doe</q0:LastName>
    </q0:Register>
  </soapenv:Body>
</soapenv:Envelope>
```

Exemplo de fluxo para troca de mensagens SOAP - DOCUMENTOS



Troca de mensagens SOAP - RPC

Request

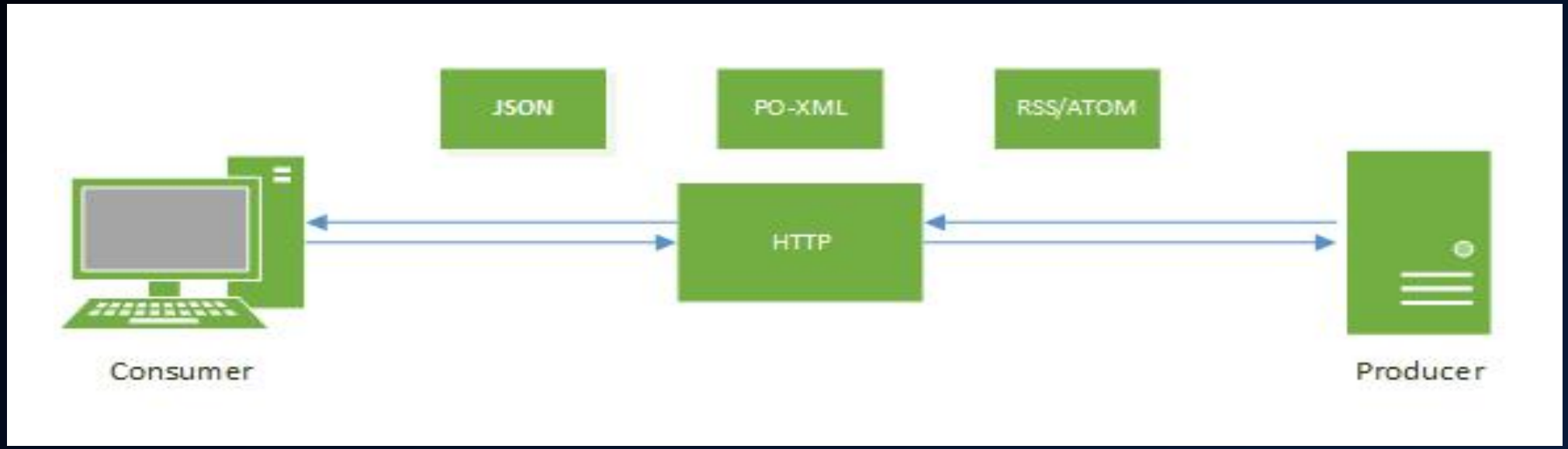
```
<?xml version="1.0" encoding="UTF-8"?>
<soapenv:Envelope
  xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  xmlns:q0="http://calculator.ibm.com">
  <soapenv:Body>
    <q0:addOp>
      <q0:input1 type="xsd:int">5</q0:input1>
      <q0:input2 type="xsd:int">3</q0:input2>
    </q0:addOp>
  </soapenv:Body>
</soapenv:Envelope>
```

Response

```
<?xml version="1.0" encoding="UTF-8"?>
<soapenv:Envelope
  xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  xmlns:q0="http://calculator.ibm.com">
  <soapenv:Body>
    <addOpResponse xmlns="http://calculator.ibm.com">
      <return>8</return>
    </addOpResponse>
  </soapenv:Body>
</soapenv:Envelope>
```

HTTP

Introdução



Protocolo HTTP

O protocolo HTTP (HyperText Transfer Protocol – Protocolo de Transferência de Hipertexto) data de 1996.

Este protocolo foi desenvolvido de maneira a ser o mais flexível possível para comportar diversas necessidades diferentes. Em linhas gerais, ele segue o seguinte formato de requisições:

```
<método> <URL> HTTP/<versão>
<Cabeçalhos - Sempre vários, um em cada linha>
<corpo da requisição>
```

Protocolo HTTP

```
GET /cervejaria/clientes HTTP/1.1
Host: localhost:8080
Accept: text/xml
```

Protocolo HTTP

A seguir o formato de resposta, composto pela versão HTTP, o código de status, a descrição do código de status e a resposta esperada:

```
HTTP/<versão> <código de status> <descrição do código>
<cabeçalhos>
<resposta>
```

Protocolo HTTP

```
HTTP/1.1 200 OK
Content-Type: text/xml
Content-Length: 245
<clientes>
 <cliente id="1">
  <nome>Alexandre</nome>
  <dataNascimento>2012-12-01</dataNascimento>
 </cliente>
 <cliente id="2">
  <nome>Paulo</nome>
  <dataNascimento>2012-11-01</dataNascimento>
 </cliente>
</clientes>
```

Métodos HTTP

A versão corrente do HTTP, 1.1, define oficialmente nove métodos – embora o protocolo seja extensível em relação a estes.

Cada método possui particularidades e aplicações de acordo com a necessidade.

Essas particularidades são definidas em termos do efeito provocado por sucessivas invocações (**idempotência**), **segurança** e **mecanismo de passagem de parâmetros**. Além disso, cada um possui suas próprias particularidades de uso.

Métodos HTTP

Hoje, estes nove são:

- GET
- POST
- PUT
- DELETE
- OPTIONS
- HEAD
- TRACE
- CONNECT
- PATCH

Métodos HTTP

Segurança dos métodos HTTP:

Quanto à segurança, os métodos são assim considerados se não provocarem quaisquer alterações nos dados contidos. Ainda considerando o exemplo das operações de bancos de dados, por exemplo, o método SELECT pode ser considerado seguro – INSERT, UPDATE e DELETE, não.

Em relação à estas duas características, a seguinte distribuição dos métodos HTTP e feita:

Métodos HTTP

	Idempotente	Seguro
GET	X	X
POST		
PUT	X	
DELETE	X	
OPTIONS	X	X
HEAD	X	X
PATCH	X	

Códigos de Status

Toda requisição enviada ao servidor retorna um código de status. Esses códigos são divididos em 5 famílias, sendo:

- 1xx – Informacionais
- 2xx – Códigos de sucesso
- 3xx – Códigos de redirecionamento
- 4xx – Erros causados pelo cliente
- 5xx – Erros causados pelo servidor

Restful

Restful

- Representational State Transfer (REST)
- Criado por Roy Fielding em 2000,como tese de doutorado (PHD) na UC Irvine
- Criado para ser completamente aderente ao padrão HTTP
- Criado para ser baseado em estados

Recursos

Serviços REST são baseados em ***resources*** (recursos), que são entidades bem definidas em sistemas, que possuem identificadores e endereços (URLs) próprios. Estas URLs devem descrever objetivamente o recurso.

Ex:

`www.cadastro.com.br/cliente`

Sendo, por exemplo, /**cliente** a representação do recurso REST de acesso a listagem clientes.

Interação por Métodos HTTP

Estas interações são padronizadas, de maneira que:

- GET – recupera dados identificados pela URL
- POST – cria um novo recurso
- PUT – atualiza um recurso
- DELETE – apaga um recurso

- É possível fazer uma relação direta destes 4 métodos principais com as operações de banco de dados CRUD (Create, Retrieve, Update, Delete).

Interação por Métodos HTTP

Representações distintas:

Através dos Media Types, é possível recuperar, utilizando a mesma URL, diferentes representações para o mesmo recurso.

Utilizando o cabeçalho Accept, o cliente pode fazer uma requisição à um serviço REST e receber um retorno no formato XML, JSON, JPEG, etc.

Passagem de parâmetros

Um dos pontos mais complexos quando falamos de implementação REST é, sem dúvidas, a passagem correta de parâmetros para nossos serviços.

A seguir os tipos de passagem de parâmetros utilizados em serviços REST:

Path parameters – são parâmetros utilizados diretamente na URL.

Devem ser utilizados quando o parâmetro para o serviço é **obrigatório**.

Ex:

```
www.cadastro.com.br/cliente/1
```

Passagem de parâmetros

Query parameters – são parâmetros também utilizados diretamente na URL, mas são apresentados após um ponto de interrogação (?) e delimitados por um “e” comercial (&).

São melhores utilizados em situações onde os parâmetros são **opcionais**, ou seja, que apenas provocam alterações na exibição do recurso em caráter temporário (paginação por exemplo).

Ex:

```
www.cadastro.com.br/cliente?pagina=1
```

Verbos HTTP

Use o verbo correto para as operações de CRUD

(*Create*, *Read*, *Update* e *Delete*):

Verbo HTTP	Coleção de recursos. Ex: cliente
GET	Lê recuso(s). Exemplo: ***/cliente - Recupera uma lista de clientes.*** ***/cliente/33 - Recupera um cliente específico.***
POST	Cria recurso(s). Exemplo: ***/cliente - Cria um novo cliente.***
PUT	Atualiza todos os dados do(s) recurso(s). Exemplo: ***/cliente/33 - Atualiza todos os atributos do cliente número 33.***
PATCH	Atualiza atributos específicos(s) do recurso(s). Exemplo: ***/cliente/33 - Atualiza parcialmente dos dados do cliente número 33.***
DELETE	Remove recurso(s). Exemplo: ***/cliente/33 - Remove o cliente número 33.***

O uso correto do *status code* facilita muito a vida dos clientes (consumidores) de sua API, utilize-os da maneira correta e padronizada.

Verbo HTTP	Status Code	Descrição
SUCCESS	200, Ok	Sucesso. Utilizado na maioria dos casos, como um GET ou um PUT/PATH.
	201, Created	Sucesso. Utilizado quando um recurso é criado.
	204, No Content	Sucesso. Utilizado quando não há dados de retorno.
	206, Partial Content	Sucesso. O recurso retornado está incompleto, geralmente utilizado em casos onde haja paginação.
CLIENT ERROR	400, Bad Request	Recurso foi acessado, porém ocorreu algum erro na validação de alguma regra de negócio da aplicação.
	401, Unauthorized	Erro de autenticação.
	403, Forbidden	Autenticação bem sucedida, mas sem privilégios suficientes.
	404, Not Found	O recurso solicitado não existe. Aplica-se também à um ID inexistente, por exemplo. Para esse caso, é importante deixar explícito o motivo do status code 404. Basta colocar no corpo da mensagem (body) o motivo do erro.
SERVER ERROR	500, Internal Server Error	Algum erro inesperado na aplicação. Exemplo: um problema de infraestrutura.

Das seis maneiras de como enviar parâmetros para API REST, as mais mais utilizadas são: *Path Parameters*; *Query Parameters*; *Header Parameters* e *Body Parameters*.

	Utilização	
Tipo	Como	Quando
PATH Parameters	Diretamente na URI da chamada. Exemplo: ***/cliente/1***	Apenas um parâmetro e obrigatório para o serviço.
QUERY Paramaters	Diretamente na URI da chamada. Exemplo: ***/cliente?nome=joao&status=1***	Parâmetros opcionais ou listas de parâmetros. Exemplo: Imagine um que o serviço suporte refinamento do resultado apresentado. Caso não seja fornecido um nome, será apresentado todos os clientes cadastrados. O parâmetro nesse caso é opcional.
HEADER Parameters	Passado no cabeçalho do HTTP. Exemplo: ***Cache-control:no-cache***	Necessidade de fornecer de metadados ao servidor e atributos como: tokens, credenciais para autenticação, etc.
BODY Parameters	Definido na construção de mensagens HTTP. Os parâmetros são enviados no corpo das mensagens HTTP em estruturas de dados como JSON ou XML. ***Exemplo: {"nome":"joao","status":1}***	Em funcionalidades que utilizem os verbos PUT, PATH, POST, etc.

Bibliografia

RPC

- Chamadas de Procedimento Remotos
  - Instituto Superior Tecnico da Universidade de Lisboa - Departamento de Engenharia Informática
  - https://fenix.tecnico.ulisboa.pt/downloadFile/377957391601 1/3%20-%20RPC%202010.pdf

- RPC - Remote Procedure Call
  - Luís Fernando Fortes Garcia - Universidade Federal do Rio Grande do Sul
  - http://penta.ufrgs.br/rc952/trab1/rpc.html

- RPC: Remote Procedure Call Protocol Specification Version 2
  - Raj Srinivasan - Sun Microsystems, Inc.
  - https://tools.ietf.org/html/rfc1831

Bibliografia

SOAP

- Web Services SOAP em Java - 2ª Edição
  - Daniel Adorno Gomes
  - Novatec

- Programming Web Services with SOAP: Building Distributed Applications
  - James Snell, Doug Tidwell, Pavel Kulchenko
  - O'Reilly Media

- Building Web Services with Java: Making Sense of XML, SOAP, WSDL, and UDDI, Second Edition
  - Claudia Zentner , Et al.
- Sams

- Special Edition Using SOAP
  - John Paul Mueller
  - Que

Bibliografia

Restful

- REST - Construa API's inteligentes de maneira simples
  - Alexandre Saudate
  - Casa do Código
-
- Web Services RESTful
  - Ricardo R. Lecheta
  - Novatec

- RESTful Web Services Cookbook
  - Subbu Allamaraju
  - O'Reilly Media

- RESTful Web APIs: Services for a Changing World
  - Leonard Richardson
  - O'Reilly Media