

# Dell IT Academy

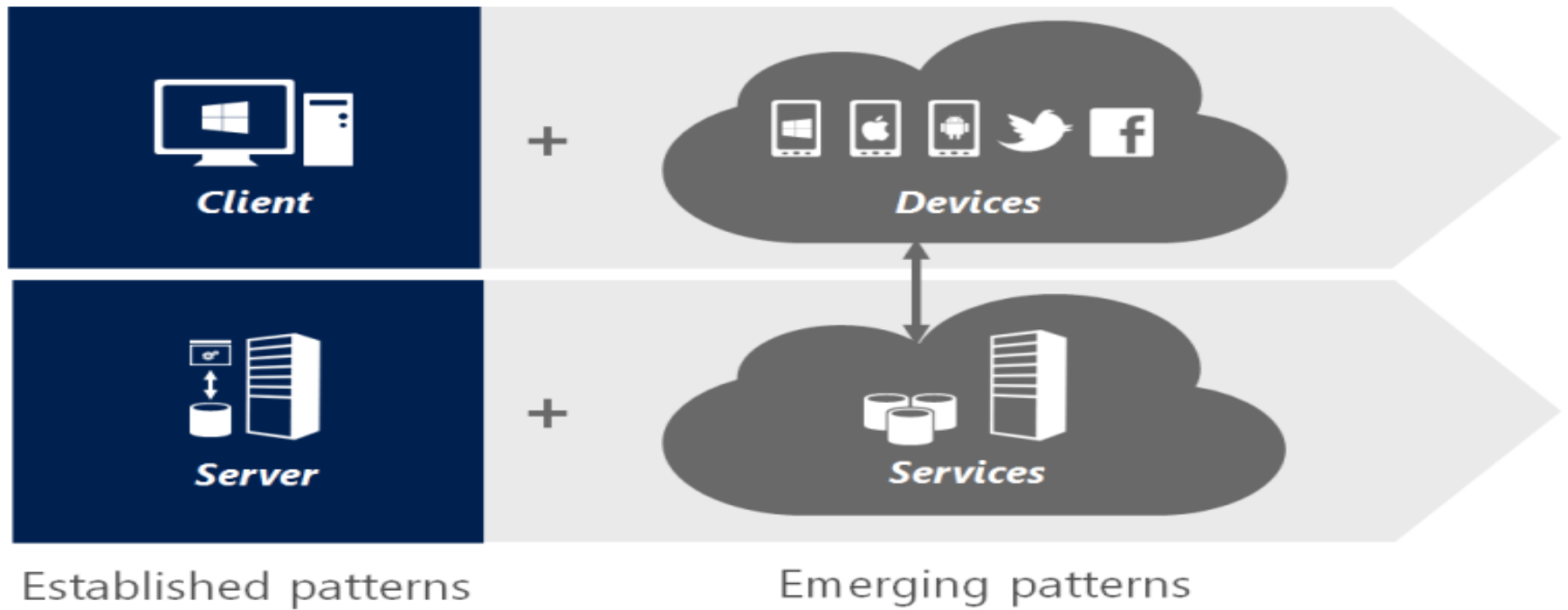


# ARQUITETURA ORIENTADA A SERVIÇOS

---

# SOA

## Application Patterns Evolution



# SOA

- Arquitetura Orientada a Serviço consiste em três elementos principais:
  - Serviços : representam funcionalidades de negócio independentes e podem ser implementados por diversas tecnologias em diversas plataformas;
  - Infraestrutura: permite combinar os serviços de maneira flexível;
  - Políticas e processos: para lidar com sistemas distribuídos heterogêneos.
- Ideias trabalhadas desde 1996

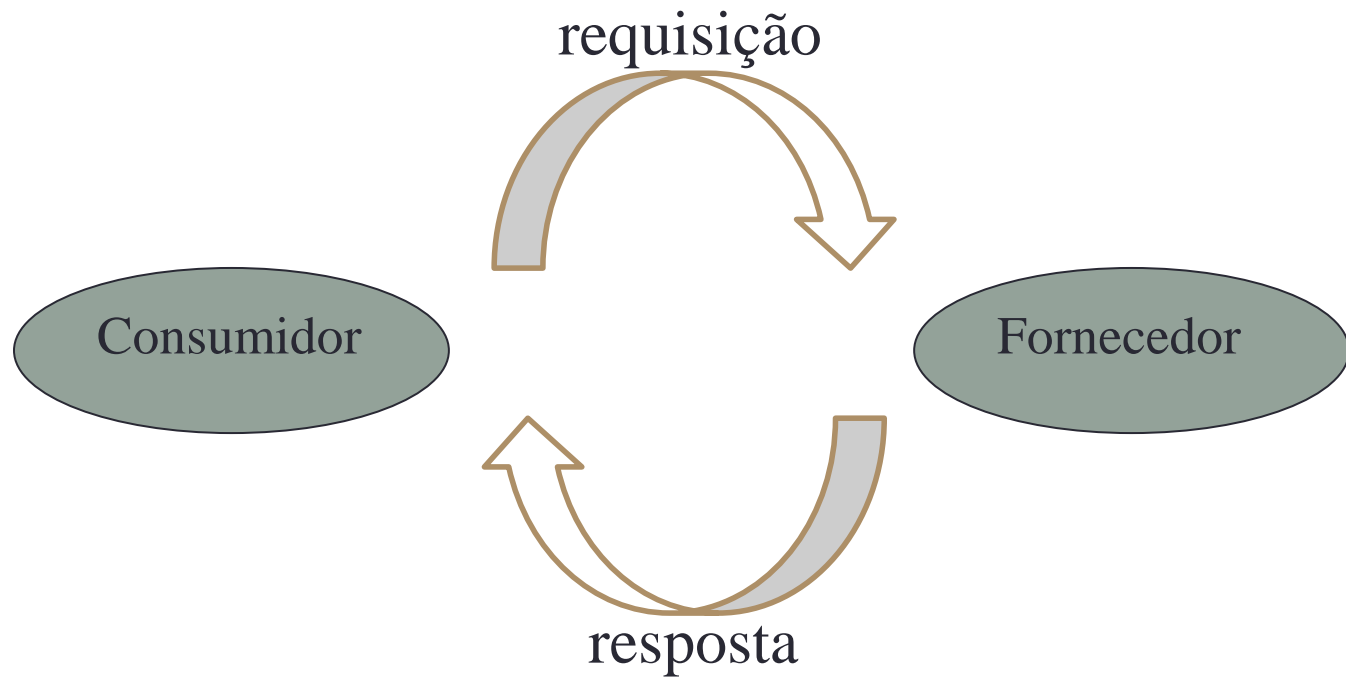
# SOA

- Um *serviço* é um pedaço de funcionalidade corporativa independente
  - A funcionalidade pode ser simples ou complexa
  - A idéia é que as interfaces externas dos sistemas devem ser projetadas de forma que as pessoas de negócio possam entendê-las
- A interface e o contrato de um serviço devem ser bem definidos

# SOA

- Um *fornecedor* é um sistema que implementa um serviço de tal forma que outros sistemas podem consumi-lo
- Um *consumidor* é um sistema que chama (usa) um serviço fornecido

# SOA



# SOA

- Um *barramento corporativo de serviços* (ESB – Enterprise Service Bus) é a infraestrutura que possibilita a interoperabilidade entre os sistemas distribuídos para serviços
  - Torna mais fácil distribuir processos corporativos através de múltiplos sistemas utilizando diferentes plataformas e tecnologias
  - Permite a chamada de serviços entre os sistemas heterogêneos
  - Permite a implementação de controles de fluxos de trabalho



# SOA

- Um ESB pode envolver as seguintes tarefas:
  - Prover conectividade
  - Transformar dados
  - Roteamento inteligente
  - Lidar com segurança
  - Lidar com confiabilidade
  - Gerenciamento de serviços
  - Monitoramento e logging

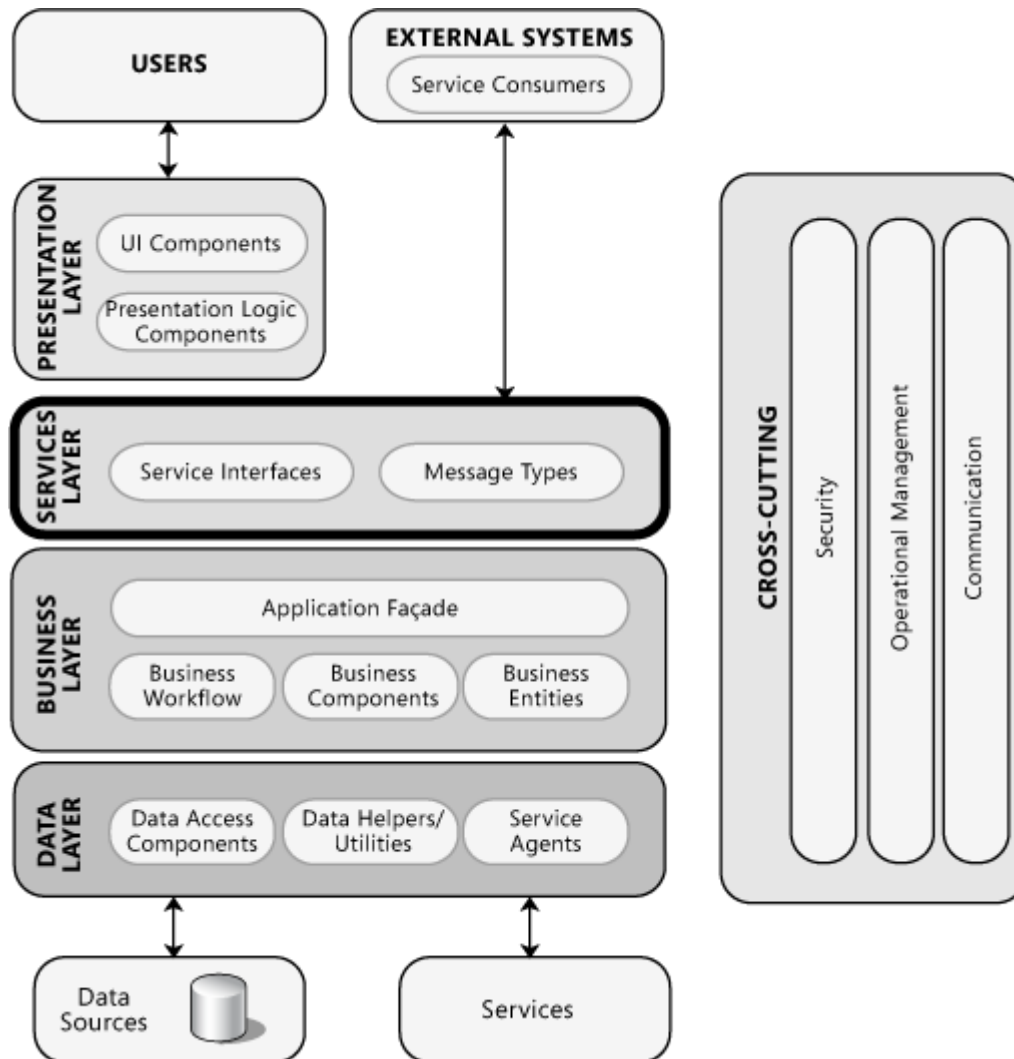
# SOA

- *Acoplamento fraco* é o conceito de redução de dependências do sistema
  - Como os processos corporativos estão distribuídos é importante minimizar os efeitos das modificações e falhas
  - Sistemas distribuídos fracamente acoplados são mais difíceis de desenvolver e manter
    - Flexibilidade, escalabilidade e tolerância a falhas são quesitos fundamentais!

# SOA

	Acoplamento forte	Acoplamento fraco
Conexões físicas	Ponto a ponto	Através de mediador
Estilo de comunicação	Síncrona	Assíncrona
Modelo de dados	Tipos comuns complexos	Tipos comuns simples
Sistema de tipagem	Forte	Fraco
Padrão de interação	Árvores de objetos complexos	Mensagens independentes
Controle da lógica de processos	Controle central	Controle distribuído
Ligação	Estática	Dinâmica
Plataforma	Dependências de plataformas fortes	Independência de plataforma
Transações	Commit em duas fases	Compensação
Deployment	Simultâneo	Em tempos diferentes
Controle de versões	Atualizações explícitas	Atualizações implícitas

# Camada de Serviços



# Lembrem-se!

- Muitas definições de SOA incluem o termo Web Service
- SOA é um conceito arquitetural
- Web Services são uma maneira possível para fornecer a infraestrutura de serviços com o uso de uma estratégia de implementação específica baseada em protocolos da Web

# WEB SERVICES

---

# Definição

- Ian Sommerville, Engenharia de Software, 8ed.:
  - “Um Web Service é um serviço, ou seja, um componente de software independente e fracamente acoplado que engloba funcionalidade discreta que pode ser distribuída e acessada por meio de uma aplicação, através de protocolos padrão.”

# Definição

- W3C:
  - “Um Web Service é um sistema de software cujo propósito é suportar de maneira interoperável a interação máquina-máquina sobre uma rede de comunicação. Ele possui uma interface descrita em um formato processável por máquinas. Outros sistemas interagem com ele de acordo com a interface através de mensagens, tipicamente sobre um protocolo padrão da internet via serialização em conjunto com outros padrões web relacionados.”



# Definição

- Alonso et al, Web Services – concepts, architectures and applications:
  - “A way to expose the functionality of an information system and make it available through standards Web technologies.”

# Web Services

- Características:
  - Objetos remotos
  - Residem em um servidor Web e têm um endereço URL
  - Trabalham sobre o modelo de requisição/resposta
  - Utilizam protocolos que facilitam a comunicação entre sistemas
    - Independente do sistema operacional e da linguagem de programação (web services interoperáveis)
  - São objetos para soluções fracamente acopladas

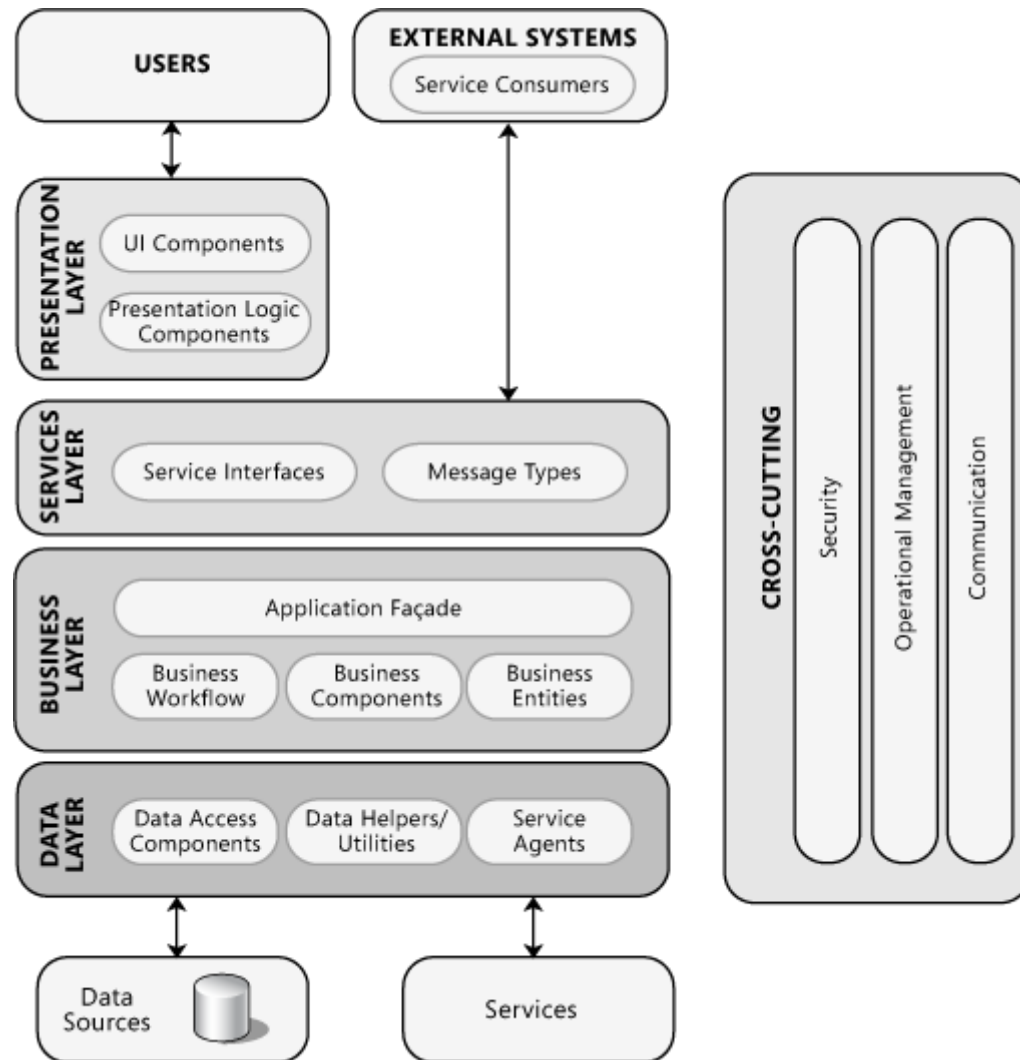
# ARQUITETURA WEB

---

# IMPORTANTE

- A discussão que segue não entra no mérito de arquiteturas de distribuição na nuvem
- Informações específicas de cada plataforma podem ser obtidas diretamente nos provedores de nuvem
  - <https://docs.microsoft.com/pt-br/azure/architecture/>
  - <https://aws.amazon.com/pt/architecture/>
  - <https://cloud.google.com/docs/tutorials>

# Arquitetura



# Web

- De acordo com o W3C:
- “A *World Wide Web* (WWW) é um espaço de informações no qual os itens de interesse, referidos como recursos, são identificados por identificadores globais chamados *Uniforme Resource Identifiers* (URI)”.

# Web

URI

`http://weather.example.com/oaxaca`

Identifies

Resource

*Oaxaca Weather Report*

Represents

Representation

**Metadata:**  
Content-type:  
`application/xhtml+xml`

**Data:**  
`<!DOCTYPE html PUBLIC "...  
 "http://www.w3.org/...  
<html xmlns="http://www...  
<head>  
<title>5 Day Forecaste for  
Oaxaca</title>  
...  
</html>`

# HTTP

- Comunicar-se com servidores e aplicativos web se dá através do protocolo *Hypertext Transfer Protocol*
  - Protocolo de nível de aplicação
  - Protocolo textual
  - Protocolo baseado em mensagens de requisição/resposta no modelo cliente/servidor
  - Protocolo sem manutenção de estado
- Versões (em uso):
  - 1.1 (RFCs [7230](#), [7231](#), [7232](#), [7233](#), [7234](#), [7235](#))
  - 2 (RFC [7540](#))



# HTTP

- Um URL *Uniform Resource Locator* é utilizado para identificar elementos em um servidor web

- Ex.: `http://java.sun.com/index.html`

- Formato geral:

`"http:" "://" host [ ":" port ] [path [ "?" query ]]`

# HTTP

- Uma requisição HTTP consiste de:
  - Uma linha inicial
  - Um ou mais campos de cabeçalho
  - Uma linha em branco
  - Possivelmente um corpo da mensagem
- Uma resposta HTTP consiste de:
  - Uma linha de status com seu código (ver [RFC](#), [Wikipédia](#)) e mensagem associada
  - Um ou mais campos de cabeçalho
  - Uma linha em branco
  - Possivelmente um corpo da mensagem

# HTTP

- Alguns métodos (também chamados de verbos):

GET	Solicita um recurso ao servidor
POST	Fornece a entrada para um comando do lado do servidor e devolve o resultado
PUT	Envia um recurso ao servidor
DELETE	Exclui um recurso do servidor
TRACE	Rastreia a comunicação com o servidor

# HTTP

HTTP Method ↕	RFC ↕	Request Has Body ↕	Response Has Body ↕	Safe ↕	Idempotent ↕	Cacheable ↕
GET	RFC 7231 <a href="#">↗</a>	Optional	Yes	Yes	Yes	Yes
HEAD	RFC 7231 <a href="#">↗</a>	No	No	Yes	Yes	Yes
POST	RFC 7231 <a href="#">↗</a>	Yes	Yes	No	No	Yes
PUT	RFC 7231 <a href="#">↗</a>	Yes	Yes	No	Yes	No
DELETE	RFC 7231 <a href="#">↗</a>	No	Yes	No	Yes	No
CONNECT	RFC 7231 <a href="#">↗</a>	Yes	Yes	No	No	No
OPTIONS	RFC 7231 <a href="#">↗</a>	Optional	Yes	Yes	Yes	No
TRACE	RFC 7231 <a href="#">↗</a>	No	Yes	Yes	Yes	No
PATCH	RFC 5789 <a href="#">↗</a>	Yes	Yes	No	No	No

[https://en.wikipedia.org/wiki/Hypertext\\_Transfer\\_Protocol](https://en.wikipedia.org/wiki/Hypertext_Transfer_Protocol)

# HTTP

- São dois os comandos mais utilizados para fornecer entrada de dados aos programas no lado servidor:
  - GET
  - POST

# HTTP

- GET:
  - Método mais simples
  - Quantidade de dados muito limitada
    - Limite implementado nos navegadores
  - Dados acrescentados à URL após um caractere “?”, no formato “campo=valor”, separados pelo caractere “&”
    - Recebe o nome de *query-string*
  - Ex.: `http://www.aqui.com/prog?id=1&ano=2008`

# HTTP

- Requisição:

GET /index.html HTTP/1.1

Host: www.example.com

# HTTP

- POST:
  - Método mais robusto
  - Quantidade de dados não é limitada como no GET
  - Dados (*query-string*) enviados no corpo da requisição do protocolo
  - Permite efeitos colaterais na execução no lado do servidor



# HTTP

- Requisição:

POST /index.html HTTP/1.0

Accept: text/html

If-modified-since: Sat, 29 Oct 1999 19:43:31 GMT

Content-Type: application/x-www-form-urlencoded

Content-Length: 41

Nome=NomePessoa&Idade=20&Curso=Computacao

# HTTP

- Resposta:

HTTP/1.1 200 OK

Date: Mon, 23 May 2005 22:38:34 GMT

Server: Apache/1.3.3.7 (Unix) (Red-Hat/Linux)

Last-Modified: Wed, 08 Jan 2003 23:11:55 GMT

Etag: "3f80f-1b6-3e1cb03b"

Accept-Ranges: bytes

Content-Length: 438

Connection: close

Content-Type: text/html; charset=UTF-8

# WEB SERVICES

---

SOAP+XML

# Arquitetura Básica

- Web Services (tipo SOAP+XML) provêm meios de objetos interagirem utilizando a Internet como meio de transmissão
- Baseado em diversos padrões:
  - *Extensible Markup Language* (XML)
  - *SOAP*
  - *Web Services Description Language* (WSDL)
  - *Hypertext Transfer Protocol* (HTTP)
  - *Etc*

# Arquitetura Básica

- Utilizam um modelo de chamada remota de procedimentos
- Provedores de serviços projetam e implementam serviços e os especificam em uma linguagem chamada WSDL
- Provedores de serviço publicam informações sobre esses serviços em um serviço de registro
- Os solicitantes de serviços, que desejam fazer uso de um serviço, buscam o registro para descobrir a especificação do serviço e para localizar o provedor do serviço
- O solicitante do serviço pode então vincular sua aplicação a um serviço específico e se comunicar com ele através de um protocolo como o SOAP
- Interoperável sobre diferentes protocolos de transporte

# Arquitetura Básica

Publicação e Descoberta: WSDL

Troca de Mensagens: SOAP

Formato Padrão de Dados: XML

Comunicação Universal: Internet

# Arquitetura Básica

- Exemplos:
  - <http://www.gxchart.com/service/webchart.asmx>
  - <http://fc.isima.fr/~lacomme/ORWebServices/index.php?idx=1>
  - <http://www.thomas-bayer.com/soap/csv-xml-converter-webservice.htm>

# Arquitetura Básica

- Comunicação:
  - Protocolo HTTP para envio e recebimento de dados (é um dos mais utilizados)
    - GET dados enviados via *query string* na URL
    - POST dados enviados no corpo da mensagem
    - Sem manutenção de estado
    - Recursos identificados por URL (“Uniform Resource Locator”)



# Arquitetura Básica

- Dados:
  - Informações serializadas em XML
- Troca de Mensagens:
  - Mensagens para objetos remotos via protocolo SOAP
  - Envelopes SOAP encapsulam dados XML
    - Nome do método
    - Parâmetros do método
    - Valores de retorno

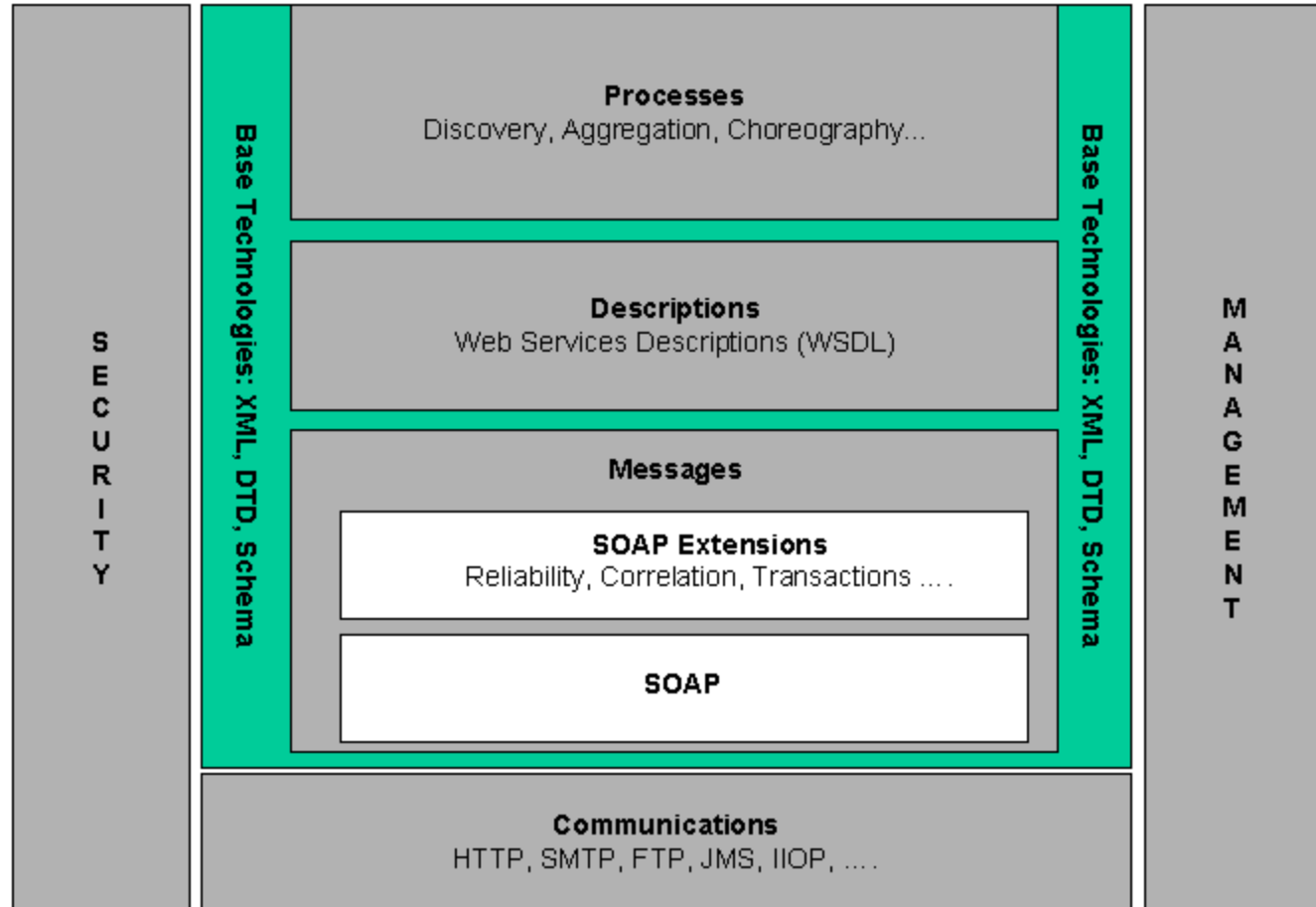
# Arquitetura Básica

- Descrição:
  - Arquivo WSDL descreve as mensagens e tipos de retorno do *web service*
  - É um documento XML

# Arquitetura Básica

- A arquitetura para disponibilizar web services pode ser bastante complexa
  - Grande conjunto de protocolos e camadas de serviços adicionais WS-\*
- Referência:
  - <http://www.w3.org/TR/ws-arch/>

# Arquitectura Básica



# Arquitectura Básica

Business Domain Specific extensions	Various	<b>Business Domain</b>
Distributed Management	WSDM, WS-Manageability	<b>Management</b>
Provisioning	WS-Provisioning	
Security	WS-Security	<b>Security</b>
Security Policy	WS-SecurityPolicy	
Secure Conversation	WS-SecureConversation	
Trusted Message	WS-Trust	
Federated Identity	WS-Federation	
Portal and Presentation	WSRP	<b>Portal and Presentation</b>
Asynchronous Services	ASAP	<b>Transactions and Business Process</b>
Transaction	WS-Transactions, WS-Coordination, WS-CAF	
Orchestration	BPEL4WS, WS-CDL	
Events and Notification	WS-Eventing, WS-Notification	<b>Messaging</b>
Multiple message Sessions	WS-Enumeration, WS-Transfer	
Routing/Addressing	WS-Addressing, WS-MessageDelivery	
Reliable Messaging	WS-ReliableMessaging, WS-Reliability	
Message Packaging	SOAP, MTOM	
Publication and Discovery	UDDI, WSIL	<b>Metadata</b>
Policy	WS-Policy, WS-PolicyAssertions	
Base Service and Message Description	WSDL	
Metadata Retrieval	WS-MetadataExchange	

# WEB SERVICES

---

REST

# REST

- Representational State Transfer é um estilo arquitetural
  - <https://www.ics.uci.edu/~fielding/pubs/dissertation/top.htm>
- Web services baseados em REST são usualmente chamados de RESTfull

# REST

- Características:
  - Serviços sem estado
  - Baseados no protocolo HTTP/HTTPS
  - Dados e funcionalidades são considerados recursos acessados via URIs
  - Infraestrutura mais leve que SOAP+XML



# REST

- Exemplos:
  - <http://predic8.com/rest-demo.htm>
  - <http://jsonplaceholder.typicode.com/>
  - <https://randomuser.me/>
  - <http://petstore.swagger.io/>

# REST

- Arquitetura baseada em quatro princípios:
  - Identificação dos recursos através de URIs – Uniform Resource Identifiers
  - Interface uniforme de acesso aos recursos
    - Operações de criação, leitura, alteração e remoção
    - Implementadas via HTTP
  - Mensagens autodescritivas
  - Iteração com manutenção de estado através de hiperlinks

# REST

- Questões para o desenvolvedor:
  - Definir quais são os “recursos” expostos
  - Definir o formato das URIs para os recursos
  - Decidir quais verbos do HTTP serão realmente utilizados
  - Estabelecer a real semântica da aplicação de cada verbo sobre um recurso

# REST

- Exemplo: URI de coleção
  - `http://exemplo/recursos/`
  - GET: lista as URIs e outros detalhes dos elementos da coleção
  - PUT: substitui a coleção por uma outra
  - PATCH: não é muito utilizado
  - POST: adiciona um novo elemento na coleção, retornando a URI para o novo elemento
  - DELETE: remove a coleção inteira

# REST

- Exemplo: URI de elemento
  - `http://exemplo/recursos/123`
  - GET: obtém a representação de um elemento específico da coleção
  - PUT: substitui um membro específico da coleção ou, se ele não existe, cria um novo
  - PATCH: atualiza (podendo ser só uma parte) do membro específico da coleção
  - POST: geralmente não utilizado; trata o elemento da coleção como uma própria coleção, adicionando um novo elemento nele
  - DELETE: remove o elemento da coleção

# REST

- Serialização de informação em vários formatos:
  - XML
  - JSON – JavaScript Object Notation
  - Texto
  - etc

# JSON

- JSON = JavaScript Object Notation
- Formato textual para serialização de dados
- Documentação: <http://json.org/>

```
{  
  "productName": "Computer Monitor",  
  "price": "229.00",  
  "specifications": {  
    "size": 22,  
    "type": "LCD",  
    "colors": ["black", "red", "white"]  
  }  
}
```