Especialização em Desenvolvimento de Aplicações para Dispositivos Móveis e Cloud Computing

DM 112

Conceituação em Cloud Computing, Programação OO, Web Services e SOA

Prof. Roberto Ribeiro Rocha

Inatel
CAMINHOS QUE CONECTAM COM O FUTURO

Roteiro

1. Introdução
2. Evolução das Arquiteturas
3. Serviços
4. Ciclo de vida de Serviços
5. Web Services
6. Rest *Services*

Introdução

- Os sistemas atuais são determinados pelo arranjo de serviços
- Intenção: obter o máximo de reúso, mínimo de retrabalho
- Precisamos definir as interfaces dos serviços e suas capacidades
- Os detalhes de implementação deveriam ser parte interna do serviço
  $\rightarrow$ não faz parte dos conceitos de SOA[1]
- O uso de *web services* não significa que o desenvolvimento do projeto é uma solução com SOA
- É necessário garantir os princípios de *design*.

[1] Arquitetura Orientada a Serviços

Introdução

- Relembrando...
  - → Contrato padronizado
  - → Baixo Acoplamento
  - → Abstração
  - → Reusabilidade
  - → Autonomia
  - → Statelessness
  - → Discoverability
  - → Composability

Roteiro

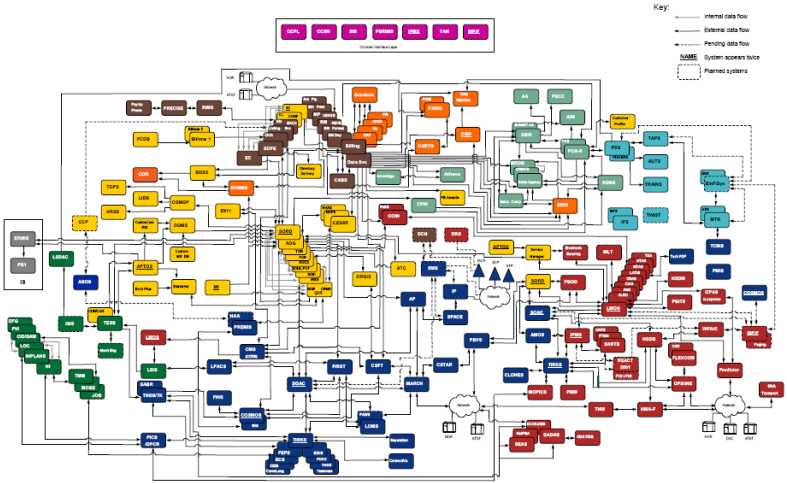
1. Introdução
2. Evolução das Arquiteturas
3. Serviços
4. Ciclo de vida de Serviços
5. Web Services
6. Rest *Services*

Evolução das Arquiteturas

Arquitetura:

- é uma visão de alto nível de um sistema
- se define os componentes e como eles estão conectados estruturados
- esta estrutura deve atender os requisitos adequadamente
- também define como o sistema interage com outros sistemas.
- garante a unidade do sistema, ou seja, a consistência entre as suas partes.

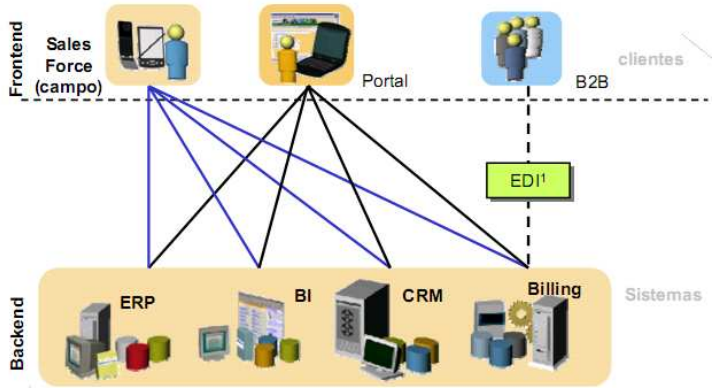
Evolução das Arquiteturas – Emaranhado



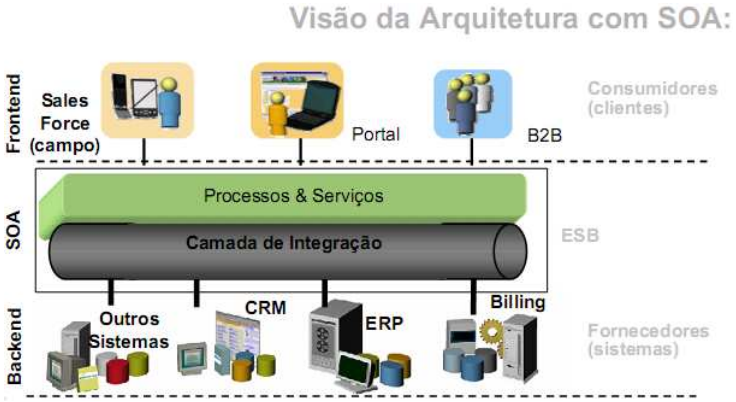
CAMINHOS QUE CONECTAM COM O FUTURO

Evolução das Arquiteturas – Tradicional

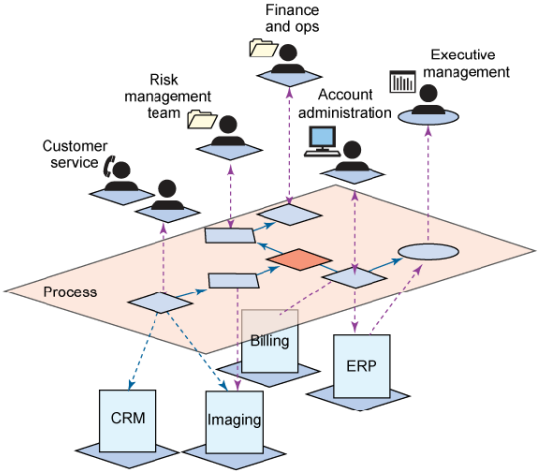
Visão da Arquitetura (tradicional):



Evolução das Arquiteturas – com SOA



Alinhamento da TI × Negócios



Resumindo SOA...

SOA é uma arquitetura voltada a negócios...
→ a qual se baseia na utilização de serviços ou módulos...
→ que podem ser reutilizados em várias áreas da organização...
→ sempre sintonizado com o processo de negócios e as necessidades da empresa.

Roteiro

1. Introdução
2. Evolução das Arquiteturas
3. Serviços
4. Ciclo de vida de Serviços
5. Web Services
6. Rest *Services*

Serviços

Definição: É uma função do sistema que pode ser facilmente vinculado a outros componentes de software.

Promove a interoperabilidade entre os sistemas.

O acesso a um serviço deve ser feito de maneira padronizada, permitindo uma maior liberdade nas tecnologias utilizadas.

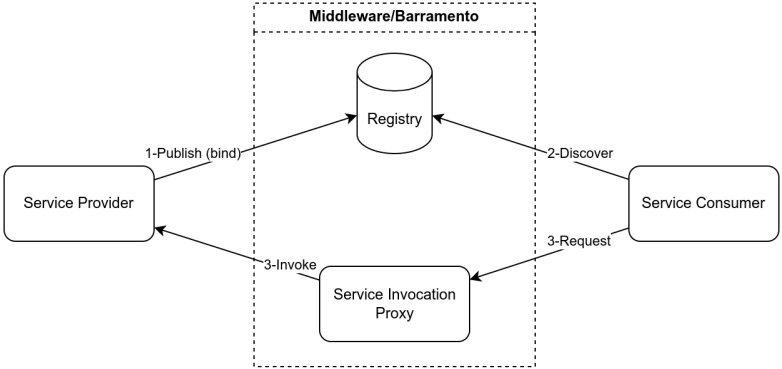
Segundo a W3C[1], Serviço é um sistema de software projetado para suportar a interoperabilidade entre máquinas sobre a rede.

[1] World Wide Web Consortium

Características do Serviço

- Não existe herança ou dependência entre os serviços
- Separação clara entre fornecedor (*provider*) e consumidor (*consumer*) $\rightarrow$ um desafio para todo o processo de desenvolvimento.
- Processos de negócio podem utilizar diversos serviços para compor toda sua execução;
- A especificação do serviço não deve ser atrelada a uma tecnologia específica.

Invocação de um serviço



Serviços

Vantagens

- Baixo acoplamento entre aplicações
- Alta interoperabilidade entre plataformas tecnológicas
- Alta reutilização de regras de negócio
- Resposta mais rápida nos processos de negócio
- Facilidade em executar testes nos serviços

Serviços

Desvantagens

- Criar serviços que fogem do padrão de SOA
- Maior complexidade (sistema maior e mais crítico)
- Maior envolvimento entre gerente de negócios e área técnica
- Velocidade/*performance* (deploy, execução, debug, monitoramento...)
- Aumento de ”custos”.

Serviços

Exemplos

- Cotação de algum item.
- Previsão do tempo, notícias, consultas de CEP[2], etc.
- Status ou pagamento de um Pedido.
- Efetuar consulta de extrato ou de crédito ou um índice de risco.
- Gerenciamento de clientes.
- Verificar a disponibilidade de um produto em estoque.
- Outros.

[2] http://www.republicavirtual.com.br/cep/exemplos.php

Especirficações ligadas a SOA

- Hoje usamos P.O.O. (é um paradigma de escrita de software).
- WOA[3]:
  - → Os softwares são acessados via rede.
  - → Os recursos são artefatos que possam ser identificados e acessados por uma URI[4].

[3] Web Oriented Architeture
[4] Universal Resource Identifier - identificador (endereço) de um recurso

Especirficações ligadas a SOA

- Web Services (serviços disponibilizados na WEB)
  - → são componentes que permitem às aplicações enviar e receber dados em formato XML ou JSON[5];
  - → respondem requisições baseadas em *request/response* (ex.: Java Servlets, Node http server, ASP, Serverside JavaScript, etc.);
  - → são padronizados segundo UDDI[6].

[5] Java Script Object Notation
[6] Universal Description, Discovery and Integration (registro de web services)

Especirficações ligadas a SOA

- WSDL[7]: é um documento XML que descrever o serviço, especifica como acessá-lo e quais as operações ou métodos disponíveis.
- SOAP[8]: Protocolo padronizado para troca de informações estruturadas usando como base o XML. Possui um envelope que define a estrutura para descrever o conteúdo da mensagem e como processá-lo.

[7] Web Services Description Language
[8] Simple Object Access Protocol

Roteiro

1. Introdução
2. Evolução das Arquiteturas
3. Serviços
4. Ciclo de vida de Serviços
5. Web Services
6. Rest *Services*

Ciclo de vida de desenvolvimento de serviços

- Facilita obter os benefícios da orientação a serviços
- Os serviços devem atender os princípios de orientação a serviços
- Thomas Erl definiu um método de análise e projeto orientados a serviços
- A arquitetura é montada através de um ciclo de vida de desenvolvimento de serviços
- Ele foca nas atividades de análise e projeto, para identificação e especificação de serviços.

Fases do ciclo de vida de desenvolvimento SOA

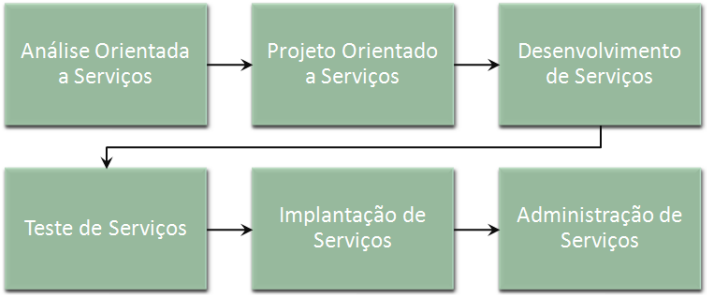


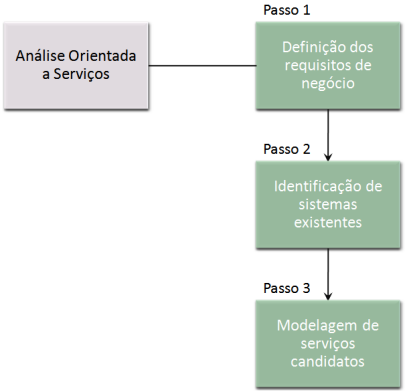
Figura: ERL, 2008.

Ciclo de vida de desenvolvimento SOA – fases

- 1^a Fase (Análise):
  - $\rightarrow$ determinar o escopo da arquitetura
  - $\rightarrow$ identificar os serviços preliminares (chamados de serviços **candidatos**).
  - $\rightarrow$ levantar a situação atual do ambiente
  - $\rightarrow$ determinar os serviços e a lógica a ser encapsulada por cada um deles
- Na 2^a fase (Projeto), determina-se como os serviços serão agrupados e construídos.

1ª Fase: Análise orientada a serviços

Possui 2 passos de coleta de informações 1 subprocesso de modelagem de serviços.



1ª Fase: Análise orientada a serviços

Passo 1: Definir os requisitos de negócio:

- Requisitos
- Fronteira da análise
- Partes envolvidas
- Partes afetadas da corporação

Passo 2: Identificar os sistemas existentes (legados).

Passo 3: Modelagem de serviços candidatos

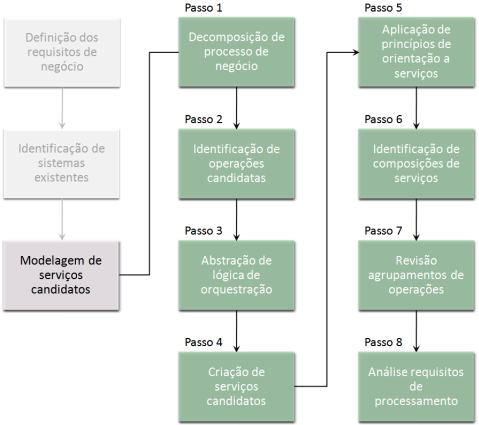
- conceitua os serviços e capacidades
- usa o termo **candidato** (não há nada concreto ainda).

Análise orientada a serviços

Prática 1 – Fazer a análise do serviço do pagamento de pedidos

Modelagem de serviços candidatos

Subprocesso que refina os conceitos, encontra e especifica os serviços.



Modelagem de serviços candidatos

- **Passo 1:** Fazer a decomposição (detalhamento) do processo de negócio.
- **Passos 2 e 3:** Identificar as operações candidatas dos serviços e fazer a abstração de lógica de orquestração.
- **Passo 4:** Criar os serviços candidatos e agrupá-los em contextos lógicos.
- **Passo 5:** Aplicar os princípios de orientação a serviços: reusabilidade, autonomia, statelessness, etc.

Modelagem de serviços candidatos

- **Passo 6:** Identificar a composição de serviços. Pode-se criar novos serviços ou separá-los para melhorar o reuso por outros que fazem a orquestração.
- **Passo 7:** Revisar os agrupamentos de serviços
- **Passo 8:** Revisar os requisitos de processamento dos serviços candidatos.

Para documentar os serviços, usa-se uma notação semelhante ao diagrama de classes da UML(Thomas Erl).

Outros diagramas também podem ser utilizados.

Modelagem de serviços candidatos

Prática 2 – Fazer modelagem de serviços candidatos para o pagamento de pedidos

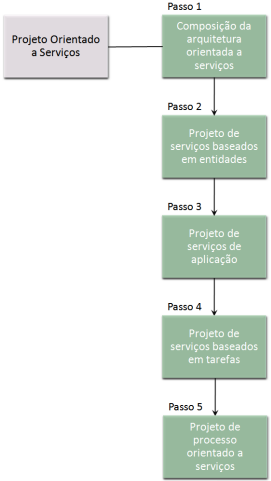
Projeto Orientado a Serviços

Principais objetivos:

- definir as interfaces dos serviços candidatos modelados na análise
- garantir a adequação aos princípios de orientação a serviços
- definir quais padrões serão suportados e utilizados na implementação dos serviços

Os resultados desta fase são as especificações dos serviços nas camadas de negócio, aplicação e orquestração.
Esta fase realiza definição de serviços físicos, arranjados em composições, que fornecerão suporte para a implementação do sistema.

Projeto Orientado a Serviços



Projeto Orientado a Serviços

Passos para o projeto de serviços:

- **Passo 1:** Definição das camadas de serviços e os padrões de especificação e implementação, bibliotecas, recursos específicos de uma linguagem.
- **Passo 2:** Definir os serviços de entidades, com operações de CRUD.
- **Passo 3:** Definir os serviços de utilidade (ou de aplicação), reaproveitáveis em várias partes do sistema.
- **Passo 4:** Definir os serviços de tarefa.
- **Passo 5:** O projeto pode ser abstraído através do mapeamento que indica onde cada serviço será inserido ou utilizado no processo de negócio.

Projeto Orientado a Serviços

Artefatos

- A documentação dos requisitos de negócio
- O modelo de processo de negócio e seu refinamento
- O modelo de casos de usos
- A modelagem orientada a serviços gera artefatos que irão cada vez mais se aproximar da implementação
- As entidades de serviços são definidas (suas operações e a estruturas das mensagens de entrada e saída).
- SoaML: linguagem para modelagem de serviços, baseada na UML (usa o mecanismo de extensão através de estereótipos)

Projeto Orientado a Serviços

Prática 3 – Fazer o projeto orientado a serviços para o pagamento de pedidos

Roteiro

1. Introdução
2. Evolução das Arquiteturas
3. Serviços
4. Ciclo de vida de Serviços
5. Web Services
6. Rest *Services*

Web Services – Definição

Web service é um sistema, disponibilizado na web, pronto para processar requisições de outros sistemas.

A comunicação é definida através de sua **interface**, que descreve:

- os métodos disponíveis,
- como invocá-los adequadamente,
- tipos de retorno e possíveis exceções.

Ele possui características declaradas e acessíveis para consulta em algum ponto da rede.

Web Services – Características

Algumas características sobre abordagens de sistemas distribuídos para aplicações, incluindo os web services.

	CORBA	Microsoft COM, DCOM	Java RMI	WebServices
Interoperável	😐	☹	☹	☺
Complexidade	☹	☺	😐	☺
Desempenho	😐	😐	😐	☹
Facilidade de Manutenção	☹	😐	😐	😐

Web Service - Cenário geral

Problemática: o grande desafio na integração de diferentes aplicações.
Fator principal: interoperabilidade em ambientes heterogêneos.
Consequência: necessidade de criação de módulos específicos (e complicados), para permitir a comunicação entre os sistemas.

Este cenário é propício para uma solução com web services.

- lógica de negócio disponibilizada de maneira padronizada, acessível na rede para aplicações cliente.

Web Service – Interface de serviço

A ”interface bem definida” pode ser feita usando padrões de mercado:

- JSON, XML, WSDL[9] (baseado em XML).

O WSDL especifica:

→ O que o serviço faz,
→ Como chamar suas operações,
→ Onde encontrar o serviço.

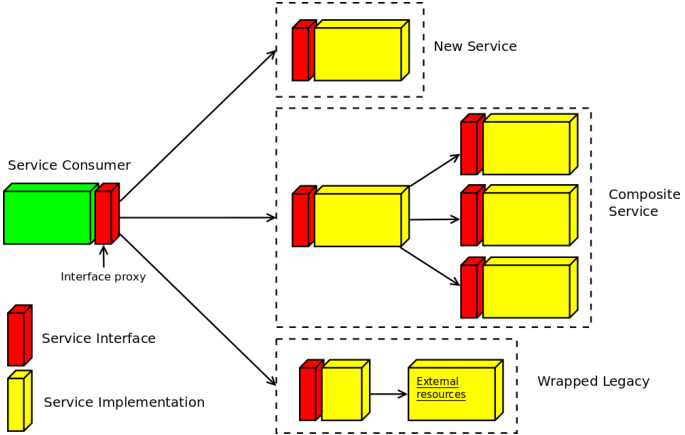
* O WSDL tem mesmo propósito que a IDL do CORBA

[9] Web Service Description Language

Web Service – Elementos

- **Provedor:** o servidor que executa a lógica do serviço
- **Consumidor:** o cliente que utiliza o serviço
- **Consumo do serviço:** ato do cliente fazer a requisição do serviço
- Os **recursos externos** podem encapsulados por um serviço:
  - → Exemplos: banco de dados, componentes Java, outros sistemas (legados), hardwares específicos, equipamentos (Indústria 4.0), etc.

Visão geral: aplicação cliente × web service



Comunicação com um web service

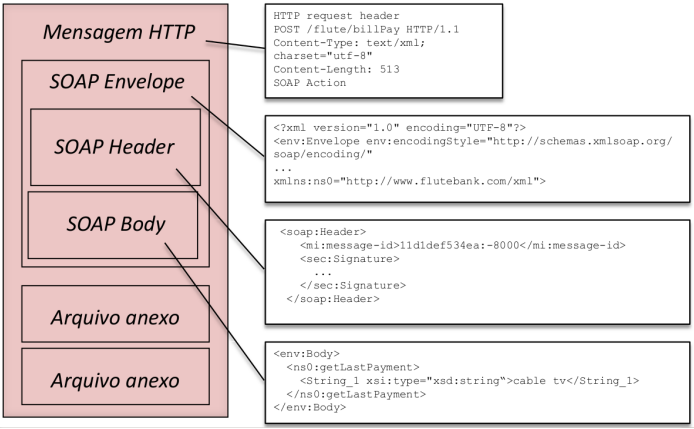
- A interação entre os sistemas através de web services nasceu da troca de mensagens XML
- Tipicamente sobre o protocolo HTTP (*firewall friendly*), evitando problemas de bloqueio de acesso.
- A padronização facilita a integração entre as aplicações.
  - → Torna a comunicação independente de SO, protocolos, linguagens de programação...
- Existem dois tipos comuns de web services:
  - → SOAP: baseado em XML;
  - → Rest: baseado em JSON (também pode usar XML).

SOAP Services

- O protocolo SOAP[10] estabelece as regras das trocas de mensagens
- **Envelope:** mensagem que segue uma gramática XML específica que padroniza o formato das suas estruturas
- Uma mensagem SOAP pode ser enviada tanto como *request* quanto *response*
- O protocolo não armazena estado (*stateless*) entre uma chamada e outra.

[10] Simple Object Access Protocol

Elementos de uma mensagem SOAP



Exemplo de um envelope simples

Código XML de um envelope:

```
<envelope>
    <header>
        <!-- informações de segurança, transação e roteamento -->
    </header>
    <body>
        <!-- conteúdo da mensagem -->
        <pedido>
            <codigo>150</codigo>
            <cliente>12345</cliente>
            <valor>200.00</valor>
        </pedido>
    </body>
</envelope>
```

SOAP Services

- A definição e validação da estrutura de um envelop SOAP é regida por um outro XML chamado de XSD[11]
  - → define os nós, elementos e atributos permitidos em um arquivo XML.
- Um outro ponto interessante de um web service é a capacidade de publicação e descoberta de um serviço. Isto é feito pelo UDDI[12]

[11] XML Schema Definition
[12] Universal Description, Discovery and Integration

SOAP Services – Abordagens

- Na abrodagem *top-down*:
  - $\rightarrow$ o WSDL é especificado primeiro para depois ocorrer a geração de código e implementações do serviço.
- Na abrodagem *botton-up*:
  - $\rightarrow$ o código fonte é criado primeiro, normalmente utilizando anotações no código, e o *framework* escolhido para a implementação gera o WSDL e publica o serviço disponibilizando para acesso pelos clientes.

Outras especificações de web services

- WS–I (”Interoperability”)
- WS–Security / WS–Policy
- WS–ReliableMessaging / WS–Addressing
- WS–Coordination / WS–Transaction
- WS–Management / WS–RemotePortlets

As especificações são submetidas para análise e aprovação da W3C.

Roteiro

1. Introdução
2. Evolução das Arquiteturas
3. Serviços
4. Ciclo de vida de Serviços
5. Web Services
6. Rest *Services*

Rest *Services*

- Rest[13] é uma alternativa ao SOAP (independente da plataforma)
- Rest define um conjunto de princípios arquiteturais para web services:
  - → Focado em recursos(capacidades): disponibilizados pelo serviço através de URI's (em forma de estrutura de diretórios).
  - → Utiliza as operações do HTTP para acessar os recursos.
- Ele permite transferência XML, JSON[14] ou ambos, porém o JSON é o mais utilizado (simplicidade e bibliotecas disponíveis).

[13] Rest: *Representational State Transfer*
[14] JavaScript Object Notation

Rest *Services* – Operações

Estabelece um mapeamento um-para-um entre as operações de CRUD (*create*, *read*, *update* e *delete*) e os métodos HTTP:

- **POST:** cria um recurso no servidor.
- **GET:** recupera(busca) um ou mais recursos.
- **PUT:** modifica o estado de um recurso ou cria um recurso.
- **DELETE:** remove ou exclui um recurso.

Rest × SOAP

- Rest e SOAP possuem diferenças técnicas e de *design*. A decisão entre eles depende dos requisitos de cada caso.
- O SOAP é utilizado em soluções que demandam validações complexas de dados ou funcionalidades avançadas.
- Rest é utilizado em casos que priorizem a simplicidade na comunicação e a redução do volume de informações trocadas.
- Curiosidade: os web services Rest também podem ser descritos usando a linguagem WADL[15] (também baseada em XML).

[15] *Web Application Description Language*

Rest – Recursos

- *Para acessar um recurso, um serviço Rest deveria possuir uma URI intuitiva, ser fácil de adivinhar, direta, previsível e fácil de entender*[16].
- A estrutura similar à de diretórios ajuda identificar o caminho dos recursos baseado na sua hierarquia. Exemplos:
  - → URI que indica uma coleção do recurso: `/aircrafts`
  - → URI que indica um elemento específico: `/aircrafts/N911NA` [17]
  - → URI que indica um elemento específico: `/aircrafts/model/747`

Dica: utilizar substantivos; e as coleções devem ser no plural.

[16] `www.ibm.com/developerworks/webservices/library/ws-restful/`
[17] `https://www.jetphotos.com/photo/10777645`

Operações HTTP × Rest

Endpoint / recurso	Método HTTP	Ação
/aircrafts/{id}	GET	Retorna um objeto com o id específico
/aircrafts	GET	Retorna uma lista de objetos
/aircrafts	POST	Insere um objeto
/aircrafts/{id}	PUT	Modifica um objeto
/aircrafts/{id}	DELETE	Remove um objeto
/aircrafts/models	GET	Retorna uma lista de objetos
/aircrafts/models/{id}	GET	Retorna um objeto com o id específico

Prática

Prática 4 – Implementar os serviços de pagamentos.

Dúvidas?

Inatel
CAMINHOS QUE CONECTAM COM O FUTURO

DM 112 – Cloud Computing, OO, Web Services e SOA

Roberto Ribeiro Rocha

- Mestre em Ciência e Tecnologia da Informação pela Unifei
- Experiência em desenvolvimento de sistemas corporativos e micro-serviços em Java
- Senior Developer na CI&T
- OCP Java 11 Certified Developer
- AWS Certified Cloud Practitioner

Email: rrocha.roberto@gmail.com

Celular: 35-9-9837-8496

Linkedin: linkedin.com/in/rrocharoberto

Inatel

CAMINHOS QUE CONECTAM COM O FUTURO