Conceitos e Tecnologias para Dispositivos Conectados (C115)

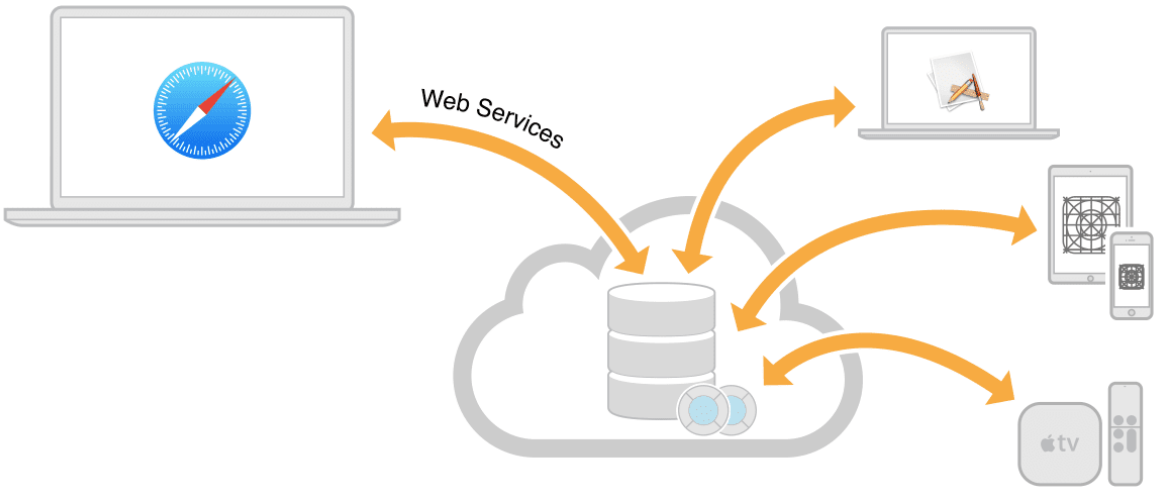
Prof. Samuel Baraldi Mafra

Inatel
Instituto Nacional de Telecomunicações

Agenda

- Web services;
  - SOAP;
  - REST.
- Virtualização;

Web services



Arquitetura SOA

- Service-Oriented Architecture (SOA), pode ser traduzido como arquitetura orientada a serviços;
- Estilo de arquitetura de software cujo princípio fundamental prega que as funcionalidades implementadas pelas aplicações devem ser disponibilizadas na forma de serviços.
- Web services;
- Computação distribuída;
- Comunicação baseada em request/reply.

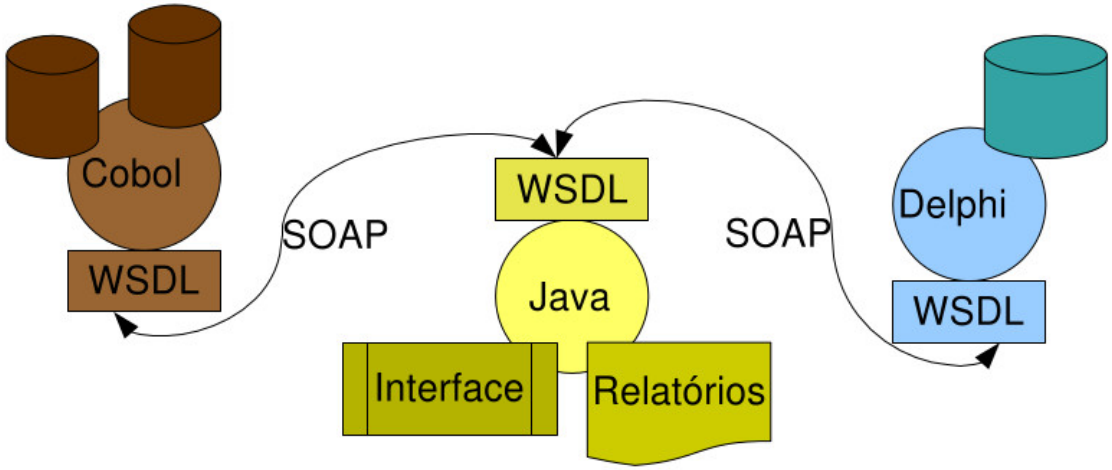
O que são web services?

Os web services são um meio padronizado para comunicação entre os aplicativos cliente e servidor na internet.

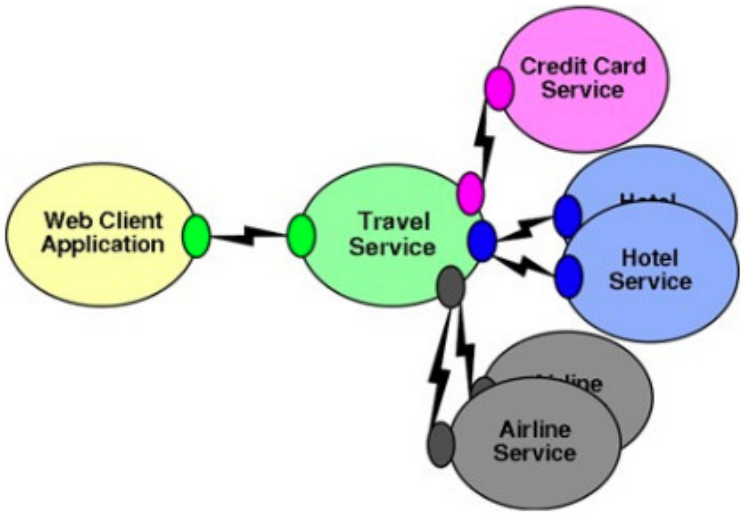
"Web Services são componentes de software com baixo fator de acoplamento, utilizados por meio de padrões de tecnologia Internet. Um Web Service representa uma função de negócio ou um serviço que pode ser acessado por uma outra aplicação." [Gartner]

Conexão entre aplicações com diferentes linguagens e plataformas.

Comunicação em sistemas legados.

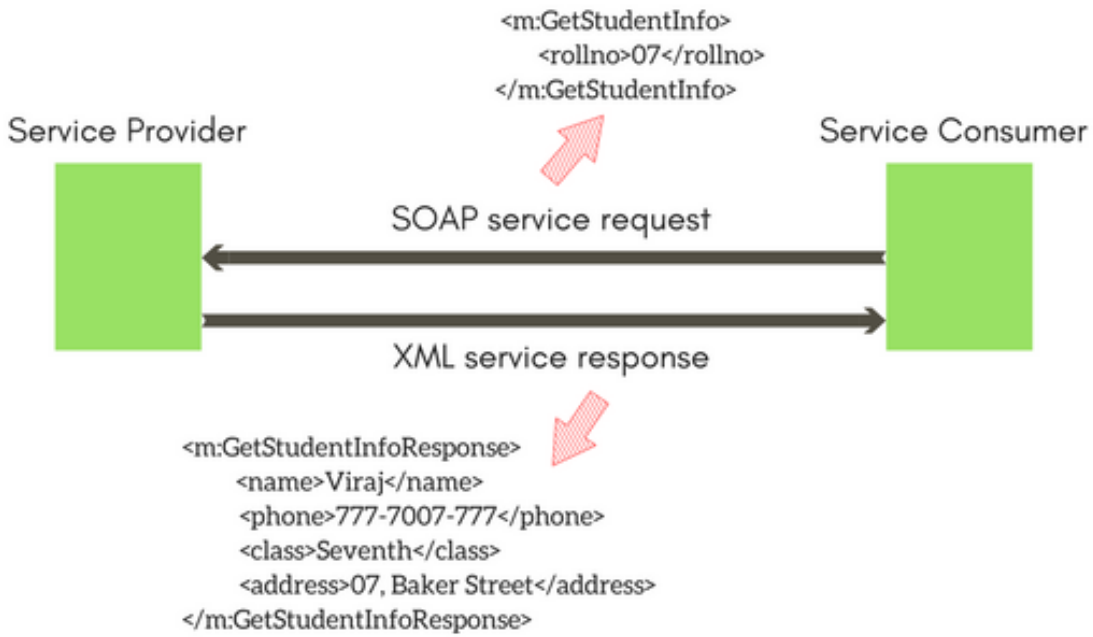


Exemplos de aplicação: Site de viagens.



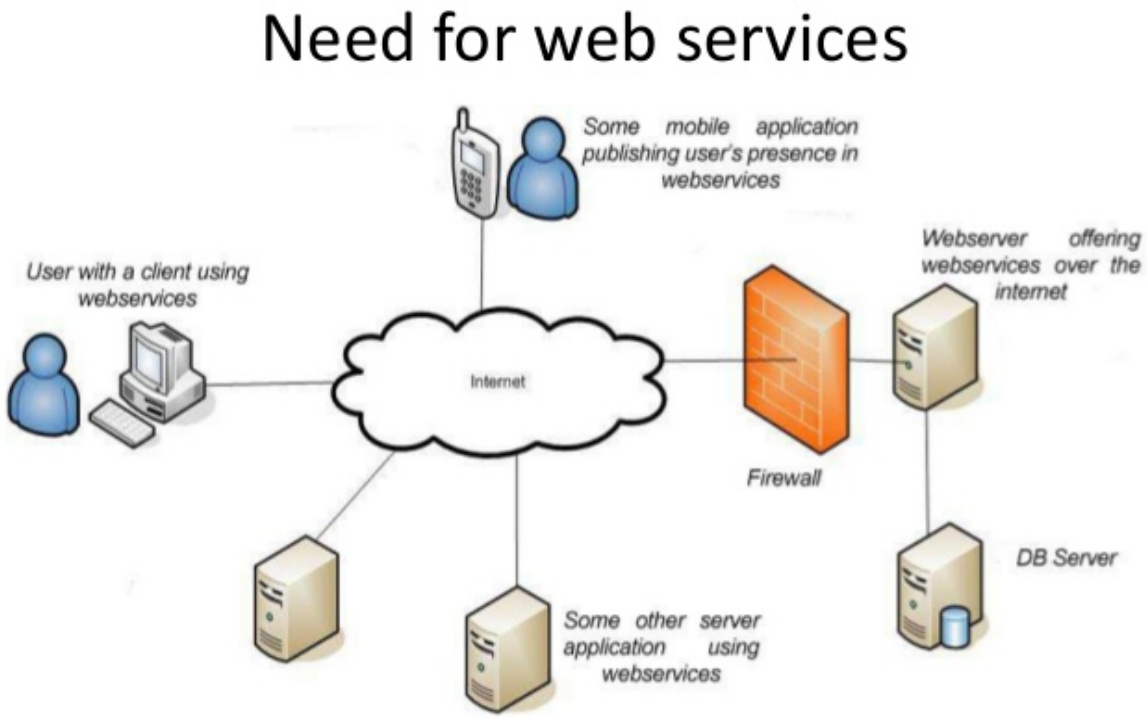
Serviços dos correios que calcula frete ou busca cep:

- Comunica-se via protocolos internet (normalmente HTTP);
- Envia e recebe dados formatados como documentos XML.



Características esperadas:

- Forneçam serviços a clientes dispersos na Web
- Interfaces bem definidas e auto-descritas
- Não sejam bloqueados por firewalls;
- Empreguem tecnologia aberta, independente de linguagem e de plataforma



Tecnologias empregadas por Web Services

- XML (eXtensible Markup Language): formato padrão para troca de dados
- SOAP: protocolo utilizado na interação com os serviços Web
- WSDL (Web Services Description Language): utilizada para descrever os serviços Web
- UDDI (Universal Description, Discovery and Integration): permite localizar serviços na rede
- REST: arquitetura para web services ;
- Restful: Implementação de protocolo que segue a arquitetura REST;
- Arquivos JSON (JavaScript Object Notation): formato para troca de dados

Arquivo XML

- Extensible Markup Language (XML) é usada para descrever dados. O padrão XML é uma forma flexível de criar formatos de informação e compartilhar eletronicamente dados estruturados por meio da Internet pública, bem como por meio de redes corporativas. Exemplo:

```
<?xml version="1.0" encoding="UTF-8"?>
<agenda>
      <contato id="01">
            <nome>Maria da Silva</nome>
            <endereco>Rua Ficticia, 32, Bairro Ficticio,
Ficticia City</endereco>
            <telefone>21 - 32658978</telefone>
            <email>maria.silva@provedor.com.br</email>
      </contato>
      <contato id="02">
            <nome>João da Silva</nome>
            <endereco>Rua Inventada, 12, Bairro Inventado,
Inventada City</endereco>
            <telefone>11 - 24569865</telefone>
            <email>joao.silva@provedor.com.br</email>
      </contato>
</agenda>
```

Simple Object Access Protocol (SOAP)

- É um padrão da indústria, criado por um consórcio formado pela Microsoft, juntamente com IBM, Lotus e UserLand.
- HTTP e XML;
- Utiliza documentação em arquivos WSDL;
- Determina um conjunto de regras de codificação para os tipos de dados.

O envelope SOAP é composto de duas partes, SOAP header e o SOAP body. O primeiro é utlizado para passagem de mensagens de controle enquanto o segundo para os dados transmitidos.

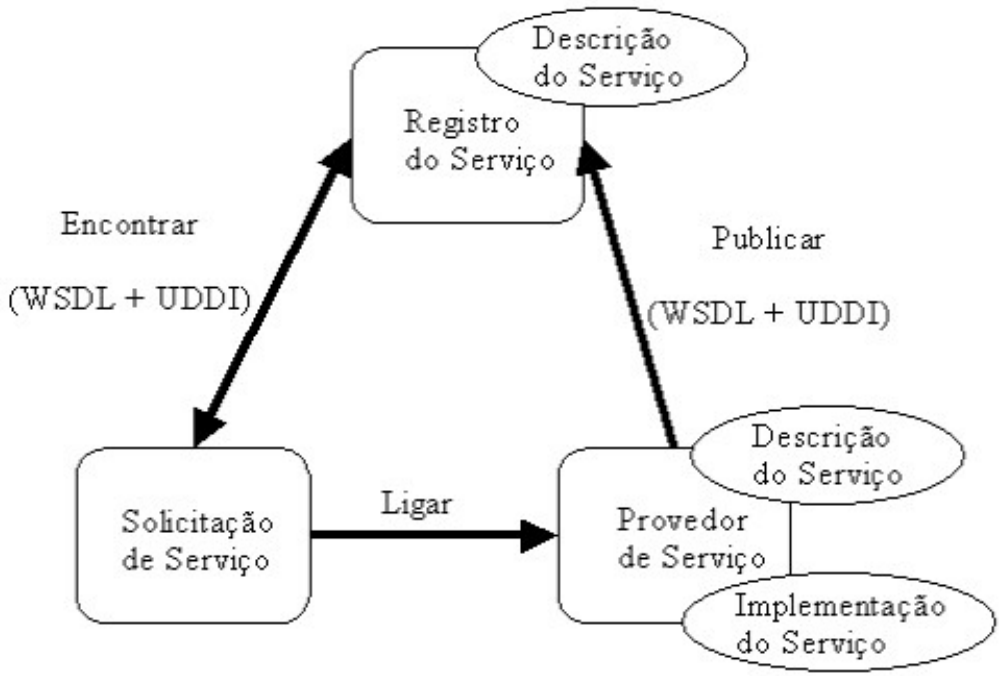
Request exemplo: Pedir uma temperatura de uma cidade.

```
POST http://www.webservicex.com/globalweather.asmx HTTP/1.1
Host: www.webservicex.com
Content-Length: 341
Content-Type: application/soap+xml

<?xml version="1.0"?>
<soap:Envelope xmlns:soap="http://www.w3.org/2003/05/soap-envelope">
  <soap:Header></soap:Header>
  <soap:Body>
    <ns1:GetWeather xmlns:ns1='http://www.webserviceX.NET'>
       <ns1:CityName>Montreal</ns1:CityName>
       <ns1:CountryName>Canada</ns1:CountryName>
    </ns1:GetWeather>
  </soap:Body>
</soap:Envelope>
```

Response exemplo:

```
HTTP/1.1 200 OK
Cache-Control: private, max-age=0
Content-Type: application/soap+xml; charset=utf-8
Server: Microsoft-IIS/7.0
Date: Wed, 28 Oct 2015 13:36:31 GMT
Content-Length: 802

<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:soap="http://www.w3.org/2003/05/soap-envelope">
  <soap:Body>
    <GetWeatherResponse xmlns="http://www.webserviceX.NET">
          <GetWeatherResult>&lt;?xml version="1.0" encoding="utf-16"?&gt;
          &lt;CurrentWeather&gt; &lt;Location&gt;Montreal-Est, Canada
          (CWPQ) 45-38N 070-33W&lt;/Location&gt; &lt;Time&gt;Mar 26, 2008 -
          12:00 PM EST / 2008.03.26 1700 UTC&lt;/Time&gt; &lt;Wind&gt;
          from the W (260 degrees) at 16 MPH (14 KT) gusting to 23 MPH (20
          KT):0&lt;/Wind&gt; &lt;Temperature&gt; 39 F (4
          C)&lt;/Temperature&gt;
          &lt;Status&gt;Success&lt;/Status&gt;
          &lt;/CurrentWeather&gt;
         </GetWeatherResult>
    </GetWeatherResponse>
  </soap:Body>
</soap:Envelope>
```

- Um web service é organizado em torno de uma arquitetura, contendo três elementos principais:
  - Registro de Serviço;
  - Fornecedor de Serviços;
  - Solicitante de Serviços.



Quando vamos trabalhar com um novo WebService, algumas informações são bem relevantes:

- Qual é o endereço do webservice?
- Qual são comandos a serem executados?
- Qual são os métodos de entrada e saída de dados?
- Quais os tipos de dados aceitos?

As respostas para estas perguntas podem ser encontradas em um arquivo de Linguagem de Descrição de Serviços Web (WSDL) em formato XML.

Linguagem de Descrição de Serviços Web (WSDL)

- WSDL é uma descrição em formato XML de um Web Service que utilizará SOAP como protocolo.
- Informa como cada serviço deve ser invocado: quais os parâmetros e tipo de dados de cada parâmetro são esperados, e qual o tipo de dado do retorno será enviado como resposta.
- Descreve como encontrá-los.

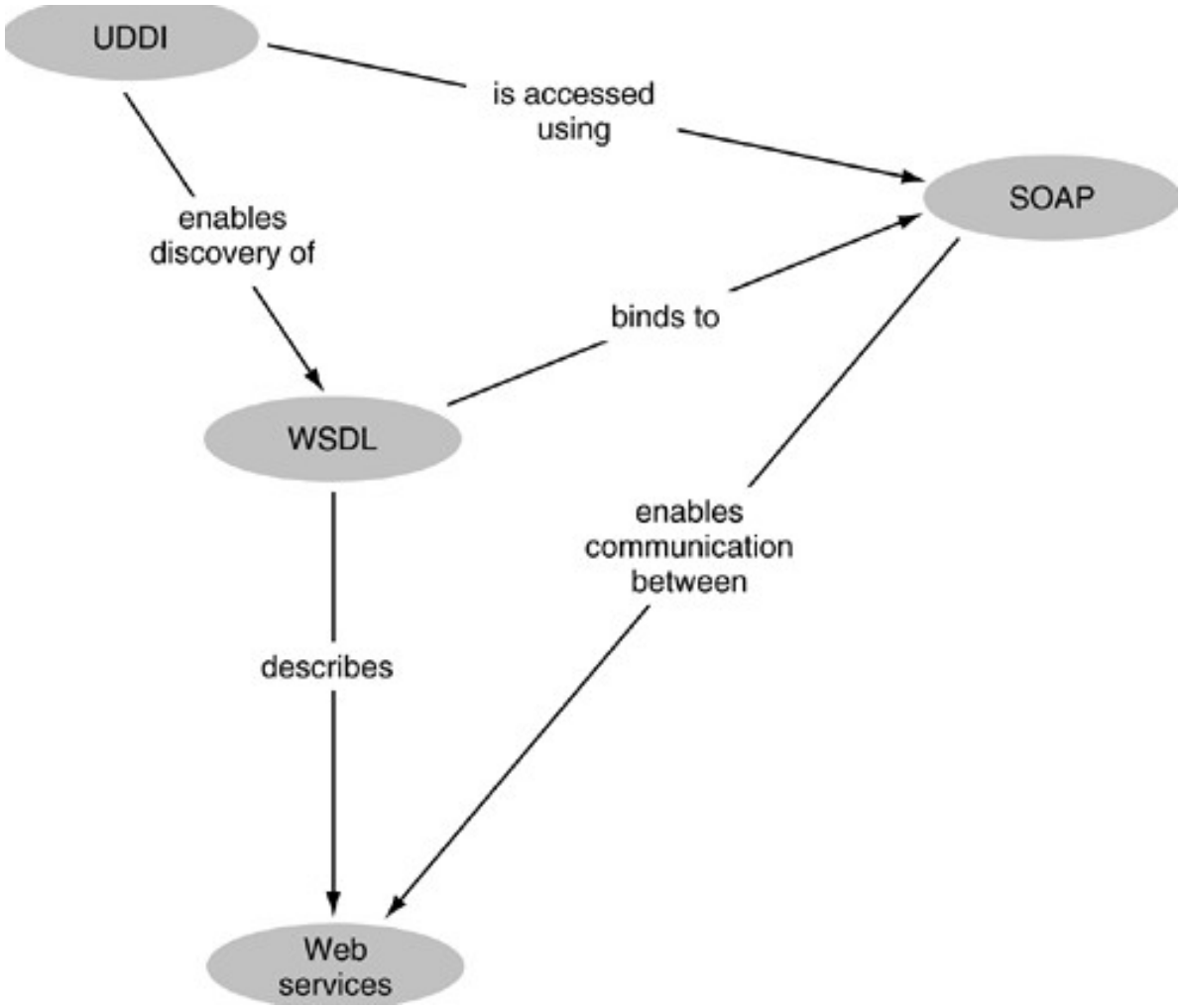
Os elementos básicos de um arquivo WSDL são:

- $< types >$: aqui deverão ser descritos os tipos de dados suportados pelo serviço em questão
- $< message >$: aqui devem ser especificados os padrões de entrada e saída de dados dos web services
- $< portType >$: aqui devem ser descritos os agrupamentos lógicos das operações. São as operações executadas pelo web service
- $< binding >$: aqui devem ser apresentados os protocolos de comunicação que os web services utilizam
- $< operation >$: região que permite a especificação das assinaturas dos métodos disponibilizados
- $< definitions >$: elemento padrão de todos os documentos WSDL. Permite efetuar descrições sobre schemas e namespaces e sobre o próprio documento WSDL.

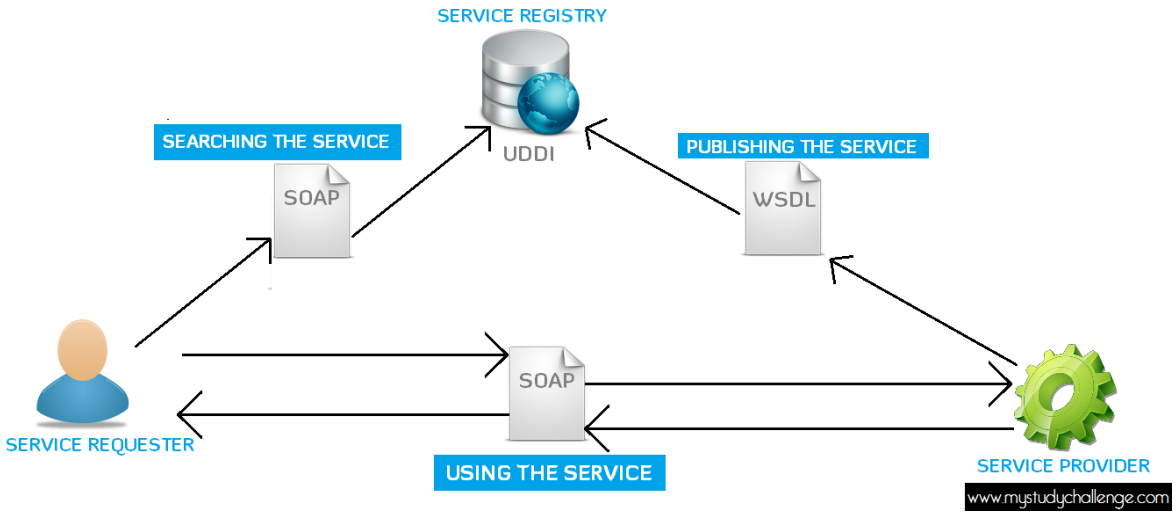
Ver arquivo book.wdsl para exemplo de arquivo.wdsl

Universal Discovery Description and Integration- UDDI

- Para fazer uma chamada a um web service, é necessário localizá-lo, descobrir a interface e semântica da sua chamada, escrever e configurar o software local para colaborar com o serviço.
- Assim, é necessário um mecanismo para publicar os serviços.
- Registro de Web services;
- Permite acesso a serviços novos de forma tão fácil quanto o acesso a antigos;


UDDI
is accessed using
SOAP
enables discovery of
binds to
WSDL
enables communication between
describes
Web services

Arquitetura SOAP



REST

- Representational State Transfer (REST). Em português, Transferência de Estado Representacional;
- Conjunto de regras que permitem a criação de um projeto com interfaces bem definidas;
- Criado por Roy Fielding em 2000,como tese de doutorado (PHD) na UC Irvine;
- Permite que diferentes procolos sejam utilizados para a troca das mensagens;
- Utiliza as funções padrão do HTTP como get, post, put, delete.
- Restful são protocolos que segem os principios do REST.

- Um recurso (Resource) é a chave da abstração no REST.
- Um recurso é qualquer coisa importante o suficiente para ser referenciado com um nome.
- O REST usa um URI (Uniform Resource Identifier) para identificar o recurso.
- Recurso é qualquer coisa que possa ter uma URI.

- JSON (JavaScript Object Notation) é um modelo para armazenamento e transmissão de informações no formato texto;
- Mais simples;
- Informações apresentadas de uma forma bem mais compacta em comparação com XML.

7) Create one contact

```
POST: /apis/fake/contact/

Request (application/json)

    body:
    {
        "full_name": "Dave Bradley",
        "email": "dave@gmail.com",
        "agenda_slug": "my_super_agenda",
        "address":"47568 NW 34ST, 33434 FL, USA",
        "phone":"7864445566"
    }
```

- Um documento ou imagem, um serviço temporal (por exemplo, "clima de hoje em Los Angeles"), uma coleção de outros recursos, um objeto não virtual (por exemplo, uma pessoa) e assim por diante . Em outras palavras, qualquer conceito que possa ser alvo de referência de hipertexto de um autor deve se enquadrar na definição de um recurso.
  - /documentos;
  - /documentos/matriculas/01;
  - /sensores/predio1/sala18/temperatura
  - http://example.com/products?color=green
  - http://example.com/orders/2007/11

Roy Fielding, CHAPTER 5 - Representational State Transfer (REST)

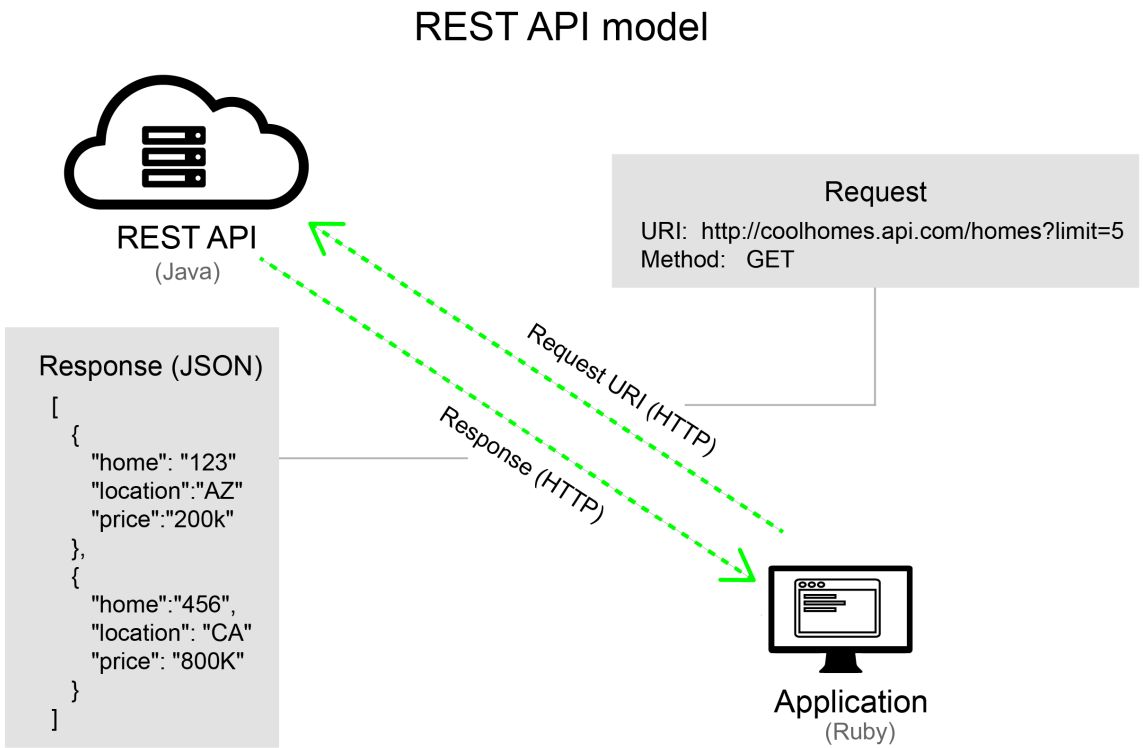
Principios:

- Um protocolo cliente/servidor sem estado: cada mensagem HTTP contém toda a informação necessária para compreender o pedido. Como resultado, nem o cliente e nem o servidor necessitam gravar nenhum estado das comunicações entre mensagens.
- Um conjunto de operações bem definidas que se aplicam a todos os recursos de informação;
- Uma sintaxe universal para identificar os recursos. No sistema REST, cada recurso é unicamente direcionado através da sua URI.
- O uso de hipermídia, tanto para a informação da aplicação como para as transições de estado da aplicação: a representação deste estado em um sistema REST são tipicamente HTML ou XML.
- Como resultado disto, é possível navegar com um recurso REST a muitos outros, simplesmente seguindo ligações sem requerer o uso de registros ou outra infraestrutura adicional.

- HATEOAS ou Hypermedia As the Engine Of Application State é um modelo simples e elegante com que uma API REST pode ser desenhada para que a mesma permita que aplicações naveguem entre seus recursos através de links e URLs sem a necessidade de conhecimento denso prévio sobre a API.

```
HTTP/1.1 200 OK
Content-Type: application/vnd.acme.account+json
Content-Length: ...

{
    "account": {
        "account_number": 12345,
        "balance": {
            "currency": "usd",
            "value": 100.00
        },
        "links": {
            "deposit": "/accounts/12345/deposit",
            "withdraw": "/accounts/12345/withdraw",
            "transfer": "/accounts/12345/transfer",
            "close": "/accounts/12345/close"
        }
    }
}
```

Rest


REST API model
REST API
(Java)
Request
URI: http://coolhomes.api.com/homes?limit=5
Method: GET
Request URI (HTTP)
Response (HTTP)
Response (JSON)
[
{
"home": "123"
"location":"AZ"
"price":"200k"
},
{
"home":"456",
"location": "CA"
"price": "800K"
}
]
Application
(Ruby)

Exemplo de requisições

- GET http://mybookmarks.com/bookmarks: Obter a lista de favoritos
- GET http://mybookmarks.com/bookmarks/13: Obter um favorito em específico.
- POST http://mybookmarks.com/bookmarks ”title´´ : ”Google´´, ”url´´ : ”http://google.com´´: Incluir um novo favorito.
- DELETE http://mybookmarks.com/bookmarks/1: Apagar um favorito.

´

REST Representational State Transfer	SOAP Simple Object Access Protocol
Estilo de arquitetura de software	Protocolo de comunicação
Abordagem mais simples	Maior complexidade
Menos burocratica	Especificação bem definidas
Fortemente relacionado ao HTTP	HTTP como meio de transporte
Muitos recursos (URI's)	Poucas (URI's)
Métodos uniformes	Muitos métodos customizados
Suporte Json, XML, YAML e etc...	Suporte XML
Pode ler cache	Invoca Serviços chamando o método RPC
Fácil integração com javascript	
Amplamente e frequentemente usado	Desempenho não é tão grande comparado ao REST

REST como um cartão postal e SOAP como uma carta:

Exemplos:

- https://towardsdatascience.com/creating-restful-apis-using-flask-and-python-655bad51b
- https://gist.github.com/davidthor/44e192169ecde2d95d6f9eda45638f12
- https://pythonhelp.wordpress.com/2014/07/25/acessando-apis-rest-com-python/

Virtualização e máquinas virtuais

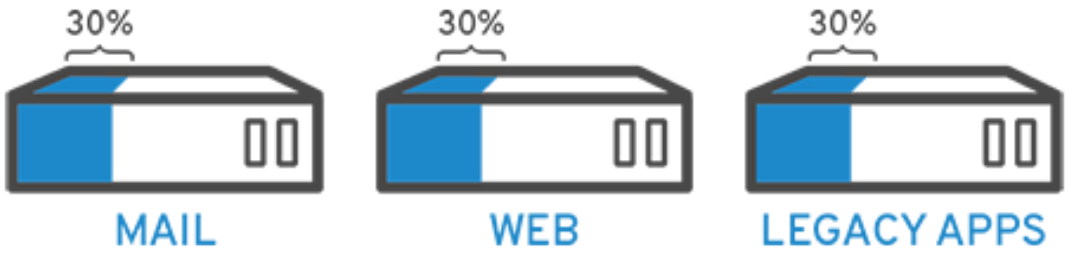
História

- No final da década de 1960 e durante a década de 1970, a maior parte do trabalho de virtualização e hipervisor foi vista em computadores mainframe desenvolvidos pela IBM, para uso na construção de sistemas de compartilhamento de tempo, teste de novas ideias de sistema operacional ou mesmo na exploração de novos conceitos de hardware;
- O aspecto da virtualização permitiu aos programadores implantar e depurar sem comprometer a estabilidade do sistema de produção principal e sem ter que implantar sistemas de desenvolvimento adicionais caros.

Virtulização:
Virtualização é o processo de criação de uma representação com base em software, ou virtual de:

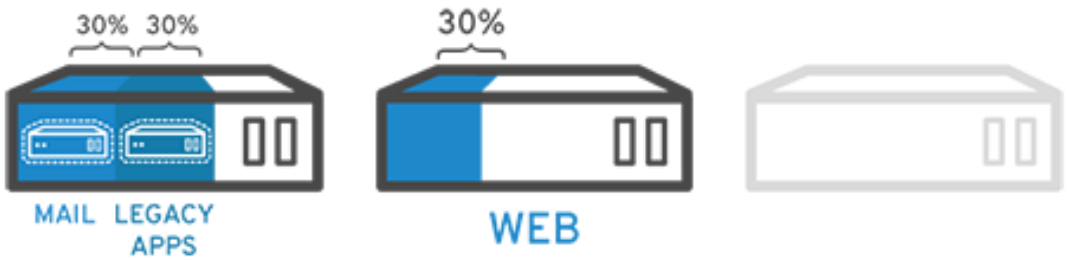
- Aplicativos virtuais;
- Servidores;
- Armazenamento;
- Redes.

Situação real de virtualização:



Servidores subutilizados

Situação real de virtualização:



A virtualização permite que o usuário use toda a capacidade de uma máquina física ao proporcionar a distribuição dos recursos entre diversos usuários ou ambientes.

A virtualização de servidores permite que vários sistemas operacionais sejam executados em um único servidor físico como máquinas virtuais altamente eficientes. Os principais benefícios incluem:

- Redução dos custos operacionais;
- Mais rapidez na implantação de cargas de trabalho;
- Melhor desempenho dos aplicativos;
- Maior disponibilidade do servidor;
- Eliminação da complexidade e da proliferação de servidores;
- Redução das emissões de carbono da empresa.

Maquina virtual

Um sistema de computadores virtual é chamado de "máquina virtual" (VM, pela sigla em inglês): um contêiner de software rigidamente isolado que contém um sistema operacional e aplicativos. Cada VM autocontida é completamente independente. Colocar múltiplas VMs em um único computador permite que vários sistemas operacionais e aplicativos sejam executados em um só servidor físico ou "host".

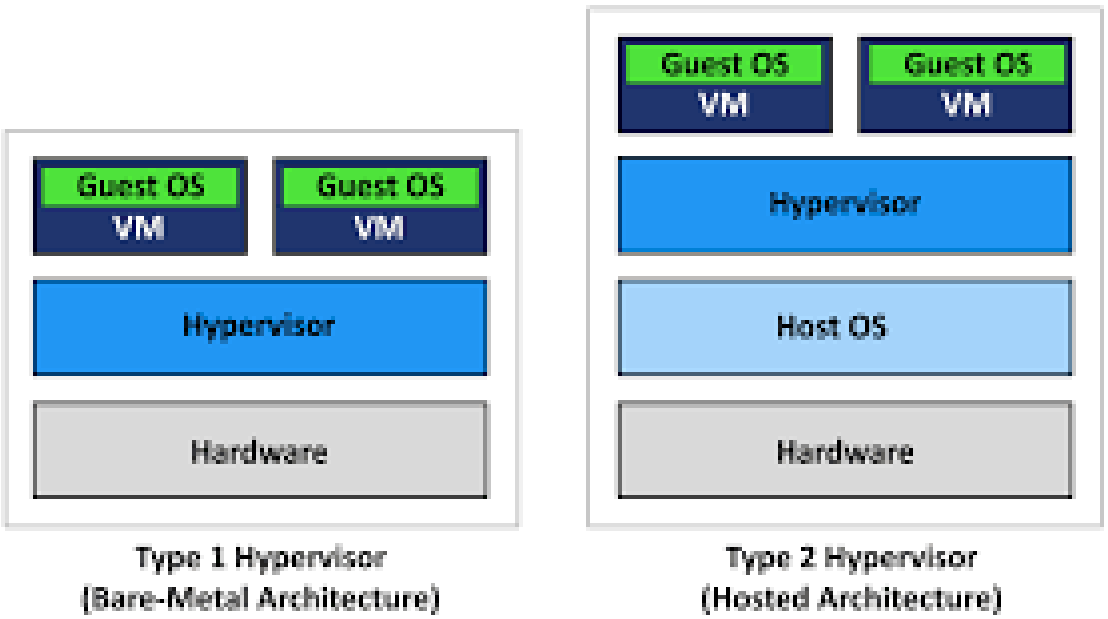
Principais propriedades das máquinas virtuais As VMs apresentam as características a seguir, que oferecem vários benefícios.

- Particionamento
  - Execução de diversos sistemas operacionais em uma máquina física.
  - Divisão de recursos do sistema entre máquinas virtuais.
- Isolamento
  - Fornecimento de isolamento de falhas e segurança no nível do hardware.
  - Preservação do desempenho com controles avançados de recursos.
- Encapsulamento
  - Gravação do estado integral da máquina virtual em arquivos.
  - Facilidade para mover e copiar máquinas virtuais (tão fácil quanto mover e copiar arquivos).
- Independência de hardware
  - Aprovisionamento ou migração de qualquer máquina virtual para qualquer servidor físico.

hipervisor

- Uma fina camada de software que dissocia as máquinas virtuais do host e aloca dinamicamente os recursos de computação em cada uma dessas máquinas, conforme necessário.
- Para executar VMs, todos os hipervisores precisam de alguns componentes no nível do sistema operacional, como gerenciador de memória, agendador de processos, stack de entrada/saída (E/S), drivers de dispositivo, gerenciador de segurança, um stack de rede e muito mais.
- O hipervisor fornece a cada máquina virtual os recursos que foram alocados e gerencia a programação de recursos da VM em comparação com os físicos.

- Os hipervisors de tipo 1, chamados "bare metal", são executados diretamente no hardware do host;
- Os hipervisors de tipo 2, conhecidos como "hospedados", são executados como uma camada de software em um sistema operacional, como outros programas de computador.



- Um hipervisor tipo 1, também chamado de hipervisor nativo ou bare-metal, é executado diretamente no hardware do host para gerenciar sistemas operacionais guest. Ele ocupa o lugar de um sistema operacional host, e os recursos da máquina virtual são programados diretamente no hardware pelo hipervisor.
- Este tipo é mais comum em datacenters corporativos ou outros ambientes baseados em servidor.
- KVM, Microsoft Hyper-V e VMware vSphere são exemplos de hipervisores tipo 1. A KVM foi integrada ao kernel do Linux em 2007.

- O hipervisor tipo 2, também chamado de hosted, é executado em um sistema operacional convencional como uma camada de software ou aplicação.
- Ele funciona abstraindo sistemas operacionais guest do sistema operacional host. Os recursos de máquina virtual são operados em um sistema operacional host, que é executado no hardware.
- Esse tipo de hipervisor é mais adequado para usuários individuais que desejam executar vários sistemas operacionais em um computador pessoal.
- VMware Workstation e Oracle VirtualBox são exemplos de hipervisores tipo 2.

Segurança

- As máquinas virtuais fornecem um ambiente que é isolado do resto do sistema. Então, independentemente do que estiver sendo executado na máquina virtual, não haverá interferência alguma no que estiver sendo executado no hardware host.
- Testar um software que possa ser prejudicial ao seu computador em uma máquina virtual;
- Como as máquinas virtuais são isoladas, mesmo se uma delas estiver comprometida, não haverá impacto em todo o sistema.
- No entanto, se o próprio hipervisor estiver comprometido, isso pode afetar todas as máquinas virtuais que ele gerencia, tornando os dados em cada uma delas vulneráveis.

Virtualização de redes

- A virtualização de redes refere-se à separação de recursos de rede tradicionalmente fornecidos em hardware para o software;
- A virtualização de redes pode combinar várias redes físicas em uma rede virtual baseada em software ou dividir uma rede física em redes virtuais separadas e independentes;
- Vlan;
- Redes definidas por software (SDN);
- Virtualização de funções da rede (NFV);

Virtualização de redes

- A virtualização de redes refere-se à separação de recursos de rede tradicionalmente fornecidos em hardware para o software;
- A virtualização de redes pode combinar várias redes físicas em uma rede virtual baseada em software ou dividir uma rede física em redes virtuais separadas e independentes;
- Vlan;
- Redes definidas por software (SDN);
- Virtualização de funções da rede (NFV);

Iremos abordar este tópico em mais detalhes nas próximas aulas.

Virtualização de desktops

- A virtualização de desktop é uma tecnologia que permite aos usuários executar o carregamento de uma estação de trabalho para acessar o desktop de um dispositivo conectado remotamente ou localmente.
- Isso separa o ambiente do desktop e seus aplicativos do dispositivo cliente físico usado para acessá-lo.
- Os desktops e os aplicativos virtualizados também podem ser fornecidos com rapidez e facilidade a filiais, funcionários externos e terceirizados e trabalhadores móveis, utilizando tablets iPad e Android.

Virtualização de desktops

- Infraestrutura de desktop virtual (VDI) é a hospedagem de ambientes de desktop em um servidor central;
- É uma forma de virtualização de desktop, uma vez que as imagens do desktop específico são executadas em máquinas virtuais (VMs) e entregues aos clientes finais por rede;
- Os clientes finais devem estar constantemente conectados ao servidor de gerenciamento centralizado para manter o acesso aos desktops virtualizados que hospedam.

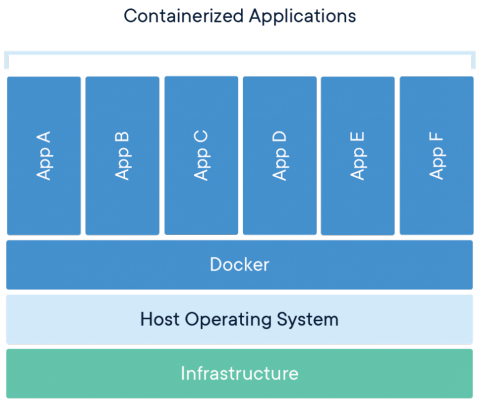
Benefícios da vitualização de desktops

- Gerenciamento de recursos;
- Trabalho remoto;
- Segurança.

Computação na nuvem

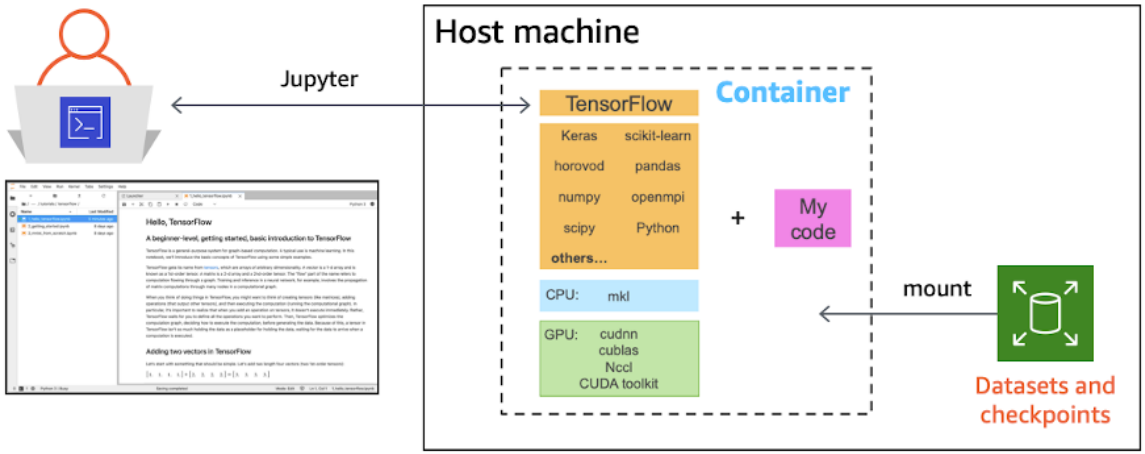
Docker ou contâiner:

- Um contêiner é uma unidade padrão de software que empacota o código e todas as suas dependências para que o aplicativo seja executado de forma rápida e confiável de um ambiente de computação para outro.
- Uma imagem de contêiner do Docker é um pacote de software leve, autônomo e executável que inclui tudo o que é necessário para executar um aplicativo: código, ferramentas do sistema, bibliotecas do sistema e configurações.



`https://www.docker.com/`

Exemplo visão computacional



- Dockerfile é um documento de texto que contém todos os comandos que um usuário pode chamar na linha de comando para montar uma imagem.
- O uso docker build de usuários pode criar uma compilação automatizada que executa várias instruções de linha de comando em sucessão.

```
FROM tensorflow/tensorflow:latest-gpu-py3

## ensure locale is set during build
ENV LANG C.UTF-8

RUN pip install keras && \
    pip install uproot && \
    pip install jupyter && \
    pip install matplotlib && \
    pip install papermill && \
    mkdir /afs

RUN apt-get update && apt-get install -y nano

COPY . /btagging
WORKDIR /btagging
```

Instalação:

- https://docs.docker.com/engine/install/ubuntu/
- https://docs.docker.com/docker-for-windows/install/
- https://docs.docker.com/docker-for-windows/install-windows-home/
- docker online https://labs.play-with-docker.com/

Exemplos docker Python e links interessantes

- https://www.tutorialspoint.com/docker/docker_containers.htm
- https://hub.docker.com/_/python
- https://docs.docker.com/language/python/build-images/
- https://ebasso.net/wiki/index.php?title=Docker:_Criando_a_Docker_Image_com_o_Python_e_Flask