Desenvolvimento de Aplicações Distribuídas

SOA e Web Services

Geanderson Esteves dos Santos

Pontifícia Universidade Católica de Minas Gerais
Instituto de Ciências Exatas e Informática

DAD (2019/01)

Tópicos

- Apresentação da disciplina
- Introdução
- Desafios e características
  - Arquitetura
  - Comunicação
  - Nomeação
  - Controle de tempo e sincronismo
  - Transação e controle de concorrência
  - Segurança
- Arquitetura Orientada a Serviços (SOA) e Web Services
- Sistemas de arquivos distribuídos
- Aplicações móveis
- Seminários

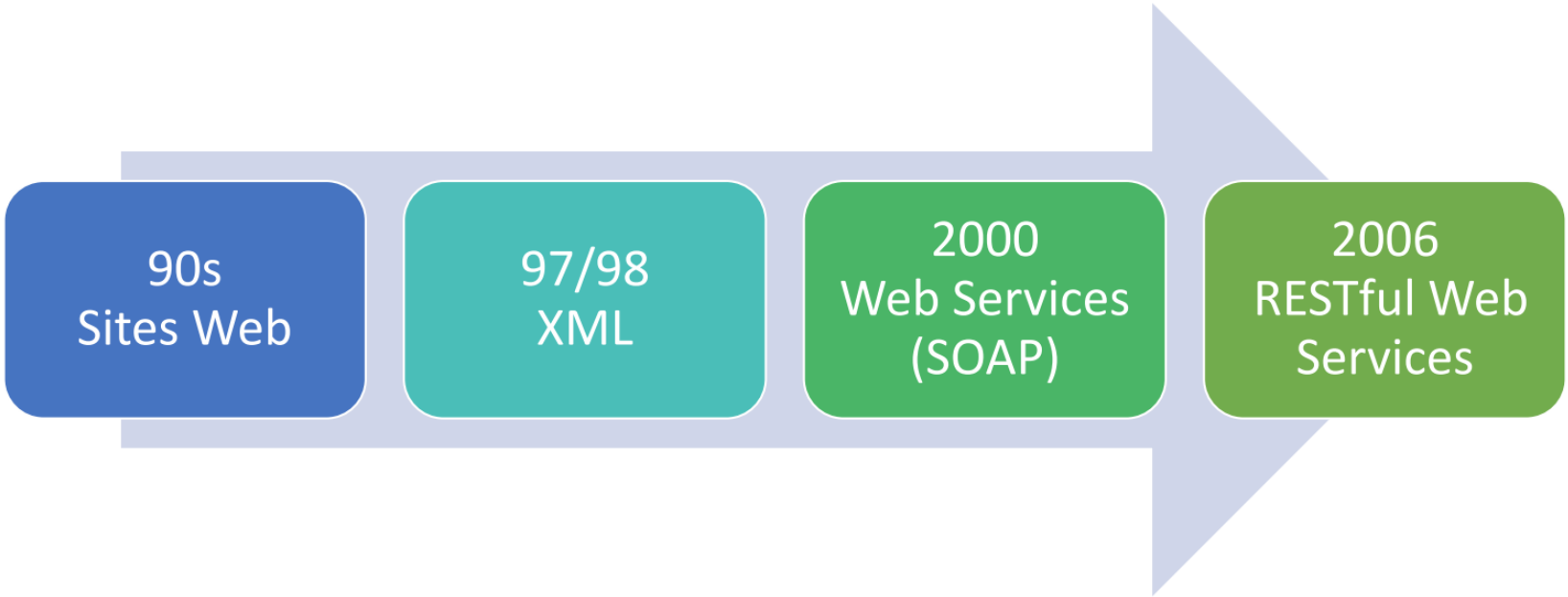
SOA e Web Services

- Introdução
  - Conceitos de Web Services
  - Características
  - Histórico e Evolução
- SOAP
  - Arquitetura
  - SOAP – Simple Object Access Protocol
  - WSDL – Web Services Description Language
  - UDDI - Universal Description and Discovery Interface
- SOA – Arquitetura orientada a Serviços
- REST
- Ferramentas
- Comparativo REST x SOAP

Conceitos de Web Server e Web Services

- O termo *'web server'* (servidor web) e *'web services'* não podem ser confundidos: um *web server* provém um básico serviço HTTP, enquanto que um *web service* provém um serviço baseado em operações definidas por uma interface
- O conceito de Web Services foi introduzido em 1999 quando Bill Gates apresentou o BizTalk
- Em seguida, o BizTalk foi renomeado para .NET

Histórico e Evolução



Conceitos de Web Services

Conceitos de Web Services

W3C - World Wide Web Consortium

"Um sistema de software projetado para suportar comunicação interoperável maquina-a-maquina através de uma rede."

Conceitos de Web Services

W3C - World Wide Web Consortium

"Um sistema de software projetado para suportar comunicação interoperável maquina-a-maquina através de uma rede."

Coulouris

"Web service provê uma interface de serviço que permite a clientes interagirem com servidores de uma maneira mais geral que os navegadores. Os clientes acessam as operações em uma interface de um web service por meio de requisições e respostas formatadas em XML e usualmente transmitidas sobre o protocolo HTTP."

Conceitos de Web Services

W3C - World Wide Web Consortium

"Um sistema de software projetado para suportar comunicação interoperável maquina-a-maquina através de uma rede."

Coulouris

"Web service provê uma interface de serviço que permite a clientes interagirem com servidores de uma maneira mais geral que os navegadores. Os clientes acessam as operações em uma interface de um web service por meio de requisições e respostas formatadas em XML e usualmente transmitidas sobre o protocolo HTTP."

Sebesta

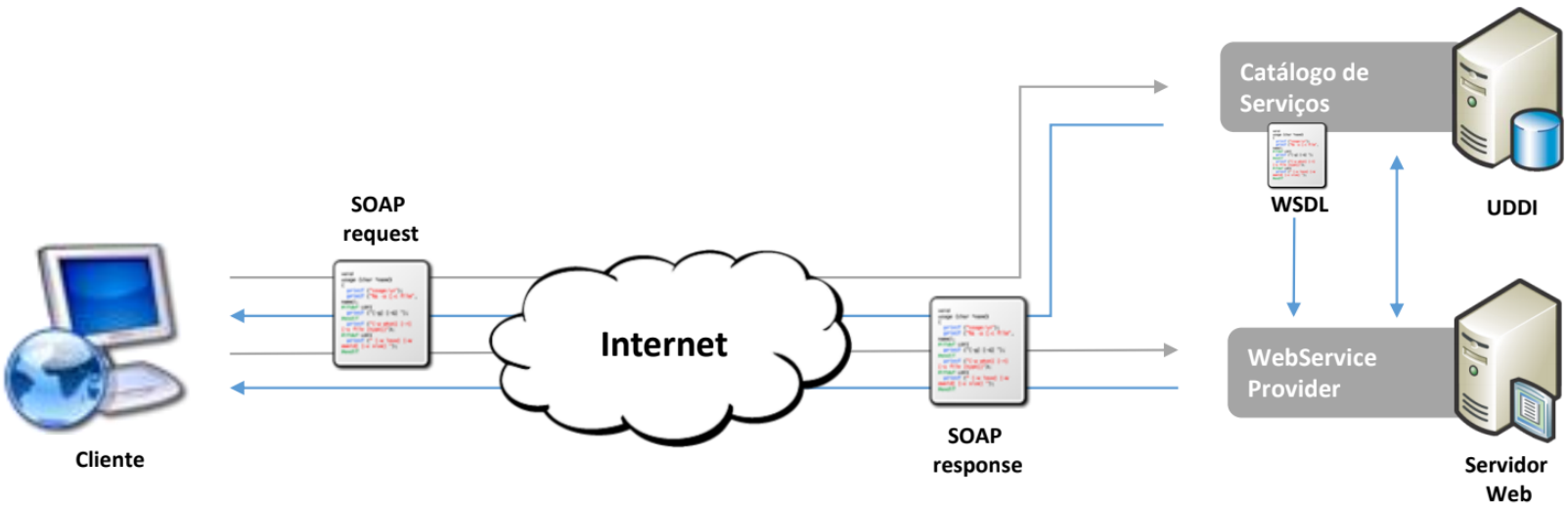
"Tecnologia usada para fornecer as tecnologias necessárias para permitir software em diferentes lugares, escritos em diferentes linguagens e residentes em diferentes plataformas sejam capazes de se comunicar."

Características de Web Services

- Utilizam o protocolo HTTP como transporte
- Podem ser utilizados em ambientes protegidos com firewall sem a abertura de portas adicionais
- É baseado em padrões abertos mantidos pelo W3C
- Permite interoperabilidade com diversas plataformas e linguagens
- Oferece baixo acoplamento das aplicações

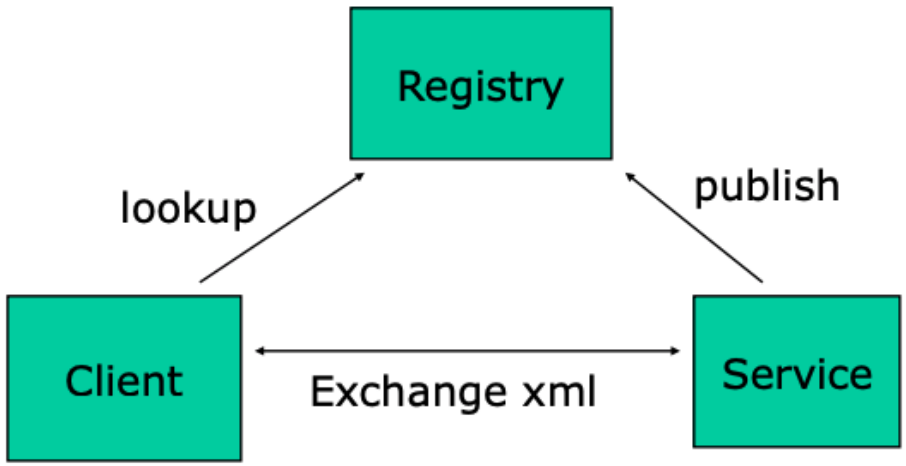
Arquitetura

- SOAP – Protocolo de troca de dados baseado em XML para envio e recebimento de mensagens na Internet
- XML – Linguagem de marcação para descrição dos dados
- WSDL – Descritor de serviços baseado em XML
- UDDI – Registro e descoberta de serviços



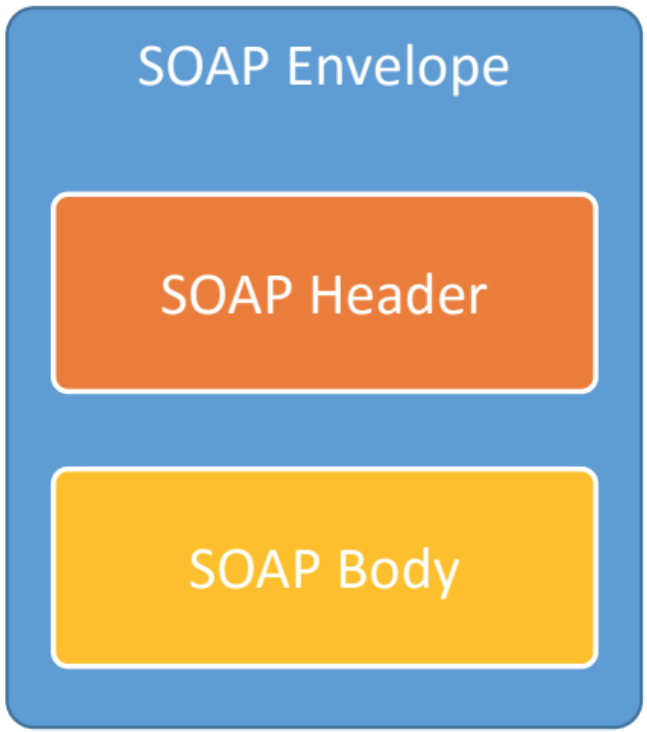
Simple Object Access Protocol (SOAP)

- Fornecem interfaces de serviços
- Comunicam usando mensagens de requisição e resposta ou outro tipo de documento XML
- Possuem uma IDL (Interface Definition Language) conhecida como WSDL (Web Service Definition Language)
- São encontrados no UDDI (Universal Directory and Discovery Service)



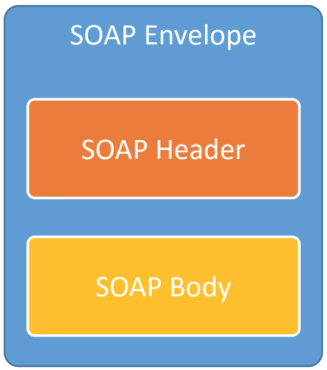
Simple Object Access Protocol (SOAP)

- Protocolo de troca de dados baseado em XML para envio e recebimento de mensagens na Internet
- Independente de plataforma ou linguagem



Estrutura da Mensagem SOAP

- Envelope
  - Define o início e o final da mensagem
  - É obrigatório
- Header (Cabeçalho)
  - Traz atributos opcionais da mensagem utilizada para o seu processamento
  - É opcional
- Body (Corpo)
  - Possui o conteúdo da mensagem em formato XML
  - É obrigatório



Estrutura da Mensagem SOAP – Requisição

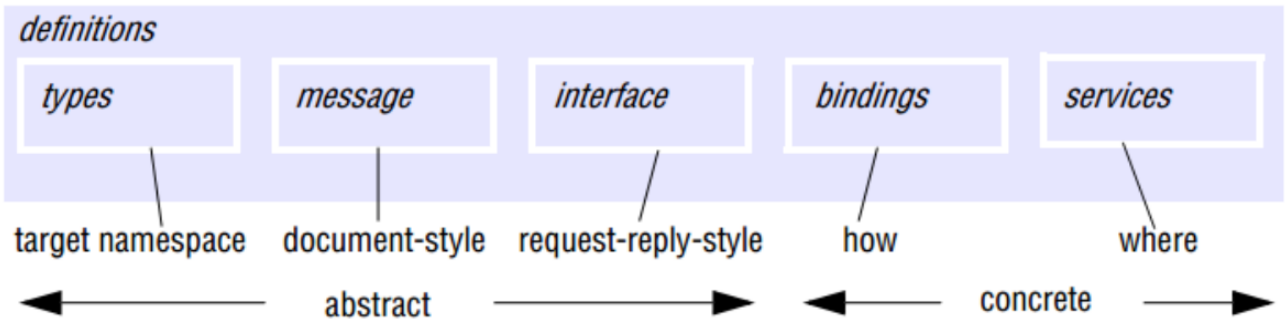
```
POST /InStock HTTP/1.1
Host: www.example.org
Content-Type: application/soap+xml; charset=utf-8
Content-Length: nnn

<?xml version="1.0"?>
<soap:Envelope
  xmlns:soap="http://www.w3.org/2001/12/soap-envelope"
  soap:encodingStyle="http://www.w3.org/2001/12/soap-encoding">

    <soap:Header>
     <t:Transaction xmlns:t="URI“ soap:mustUnderstand="1" > 5
     </t:Transaction>
    </soap:Header>

    <soap:Body xmlns:m="http://www.example.org/stock">
     <m:GetStockPrice>
       <m:StockName>IBM</m:StockName>
     </m:GetStockPrice>
    </soap:Body>

</soap:Envelope>
```

Estrutura da Mensagem SOAP – Resposta

```
POST /InStock HTTP/1.1
Host: www.example.org
Content-Type: application/soap+xml; charset=utf-8
Content-Length: nnn

<?xml version="1.0"?>
<soap:Envelope
  xmlns:soap="http://www.w3.org/2001/12/soap-envelope"
  soap:encodingStyle="http://www.w3.org/2001/12/soap-encoding">

    <soap:Header>
     <t:Transaction xmlns:t="URI“ soap:mustUnderstand="1" > 5
     </t:Transaction>
    </soap:Header>

    <soap:Body xmlns:m="http://www.example.org/stock">
     <m:GetStockPrice>
       <m:StockName>IBM</m:StockName>
     </m:GetStockPrice>
    </soap:Body>

</soap:Envelope>
```

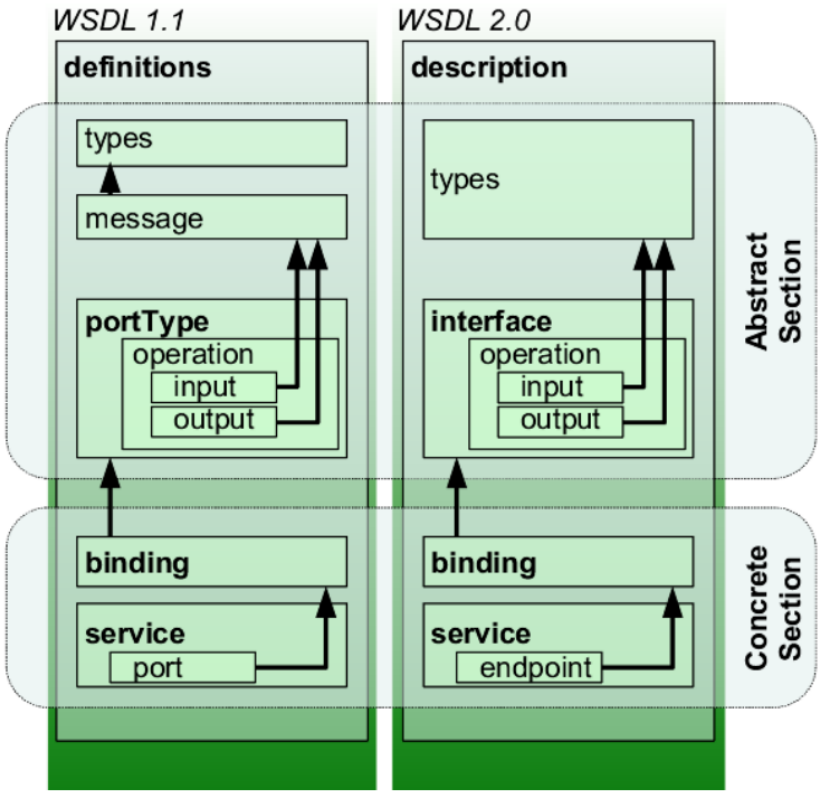
Web Services Description Language (WSDL)

- Linguagem de descrição de um web service baseada em XML
- Define os seguintes objetos:
  - **Tipos de dados** suportados pelo serviço
  - Padrão de **mensagens** de entrada e saída
  - Protocolos de comunicação permitidos pelo Web Service (**binding**)
  - Serviços e portas de comunicação onde **operações** são disponibilizadas pelo Web Service
  - **Definições** sobre schemas e namespaces utilizados no contexto do Web Service



Web Services Description Language (WSDL) - Exemplo

```
<message name="getTermRequest">
 <part name="term" type="xs:string"/>
</message>

<message name="getTermResponse">
 <part name="value" type="xs:string"/>
</message>

<portType name="glossaryTerms">
 <operation name="getTerm">
  <input message="getTermRequest"/>
  <output message="getTermResponse"/>
 </operation>
</portType>
```

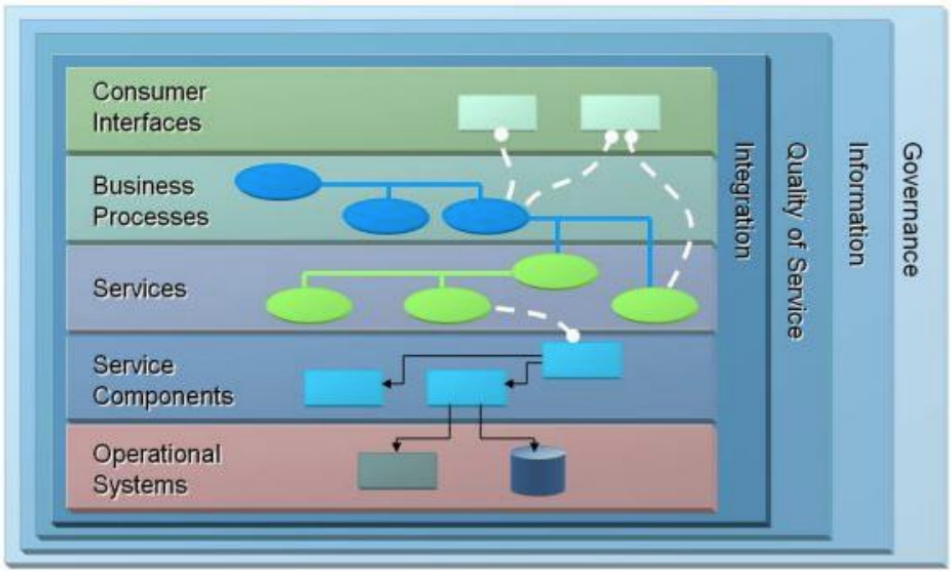


Universal Description and Discovery Interface (UDDI)

- Qualquer organização que planeja distribuir suas aplicações por meio de web services preferirá o uso de um diretório de serviços para prover acesso ao web service
- Essa Funcionalidade é implementada por meio de UDDI
- Serviço de diretório que mantem referência para os web services registrados
- Assim, organizações podem registrar e buscar por web services
- Os serviços providos podem ser acessados por meio do nome (‘white pages’) ou por meio do atributo (‘yellow pages’)
- UDDI fornece uma API para checagem (‘lookup’) dos serviços registrados

Service Oriented Architecture (SOA)

- Estilo de arquitetura de software cujo princípio fundamental está baseado em funcionalidades implementadas por aplicações e disponibilizadas na forma de serviços.
- **Camadas**
  - Interface de Usuário
  - Processos de Negócios
  - Serviços
  - Componentes de Serviços
  - Sistemas Operacionais



REpresentational State Transfer (REST)

- É um estilo arquitetural para construção de web services escaláveis que define um conjunto de regras.
- **Regras REST**
  - Cliente e Servidor
  - Sem estado (stateless) → servidor não armazena sessão do cliente
  - Interface Uniforme → permite que qualquer componente que entenda o protocolo de aplicação HTTP use o serviço
  - Permite cache (cacheable)
  - Sistema em camadas

Exemplos de APIs REST

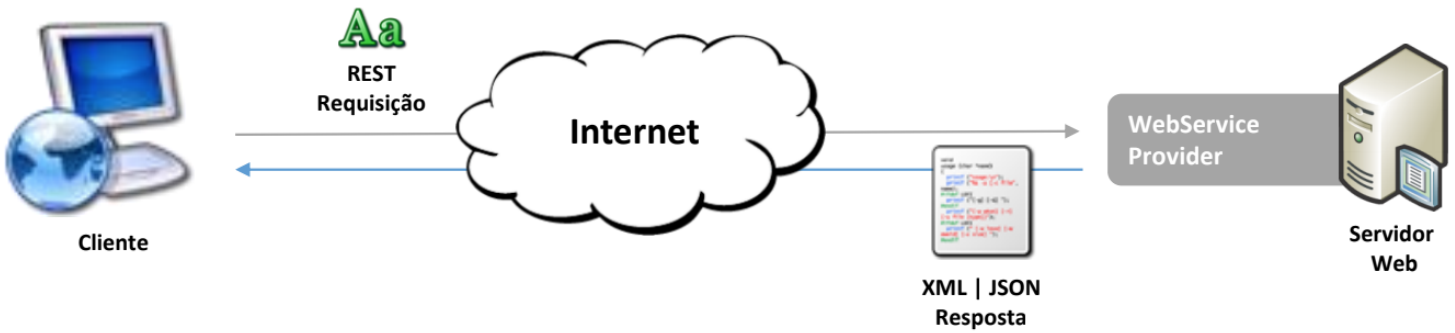
- Google
  - Site da Documentação: https://developers.google.com/apis-explorer
  - Exemplo Google Maps: http://maps.googleapis.com/maps/api/geocode/json?address=brasil&sensor=true
- Facebook (The Graph API)
  - Site da Documentação: https://developers.facebook.com/docs/graph-api
  - Exemplo – Obter uma foto do usuário: https://graph.facebook.com/536510179872296/picture?type=small
- Twitter
  - Site da Documentação: https://dev.twitter.com/rest/public
- GitHub
  - Site da Documentação: https://developer.github.com/v3/

Regras REST – Cliente e Servidor

- O cliente se preocupa com a apresentação para o servidor e o estado da aplicação
- O servidor se preocupa com o armazenamento dos dados e a lógica do negócio

Benefícios

- Portabilidade da interface de usuário (Desktop, Mobile, API)
- Escalabilidade (Múltiplos servidores e clientes)



Regras REST – Interface uniforme

- Identificação de recursos via URI: `http://graph.facebook.com/536510179872296`
- Manipulação de recursos via representações
- Mensagens auto descritivas
- Hipertexto como o mecanismo de estado da aplicação

Benefícios

- Simplifica a implementação
- Prove desacoplamento da arquitetura
- Permite que cada parte (cliente e servidor) evoluam independentemente

Regras REST – Interface uniforme

- Identificação de recursos via URI → **API Facebook**
- Requisição
  `http://graph.facebook.com/geanderson.esteves`
- Resposta (JSON)


```
{
  "id": "1047518425438133",
  "name": "Geanderson Esteves",
  "last_name": "Esteves",
  "first_name": "Geanderson"
}
```

Regras REST – Interface uniforme

- Hipertexto como o mecanismo de estado da aplicação (HATEOAS)

Resposta p/ saldo de 100,00

```
GET /account/12345 HTTP/1.1

HTTP/1.1 200 OK
<?xml version="1.0"?>
<account>
    <account_number>12345</account_number>
    <balance currency="usd">100.00</balance>
    <link rel="deposit" href="/account/12345/deposit" />
    <link rel="withdraw" href="/account/12345/withdraw" />
    <link rel="transfer" href="/account/12345/transfer" />
    <link rel="close" href="/account/12345/close" />
</account>
```

Resposta p/ saldo de -25,00

```
GET /account/12345 HTTP/1.1

HTTP/1.1 200 OK
<?xml version="1.0"?>
<account>
    <account_number>12345</account_number>
    <balance currency="usd">-25.00</balance>
    <link rel="deposit" href="/account/12345/deposit" />
</account>
```

Regras REST – Permite cache

- As respostas podem ser armazenadas em cache (No cliente ou em Proxies reversos)
- As respostas devem definir se permitem ou não o cache

Benefícios

- Aumenta eficiência (Menos requisições)
- Aumenta performance (Menor latência entre cliente e servidor)
- Permite maior escalabilidade

Regras REST – Sistema em camadas

- Permite uma arquitetura composta em camadas
- Cada componente não pode ver além da camada seguinte
- Servidores intermediários podem atuar provendo escalabilidade
  - Servidores de Cache
  - Balanceadores de carga

Benefícios

- Simplifica o cliente
- Permite maior escalabilidade (load balance, Cache)

SOA e Web Services – Ferramentas

Cliente HTTP

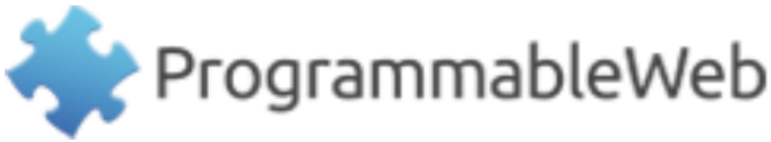
- SOAP UI
- curl
- Postman

Documentação de APIs:

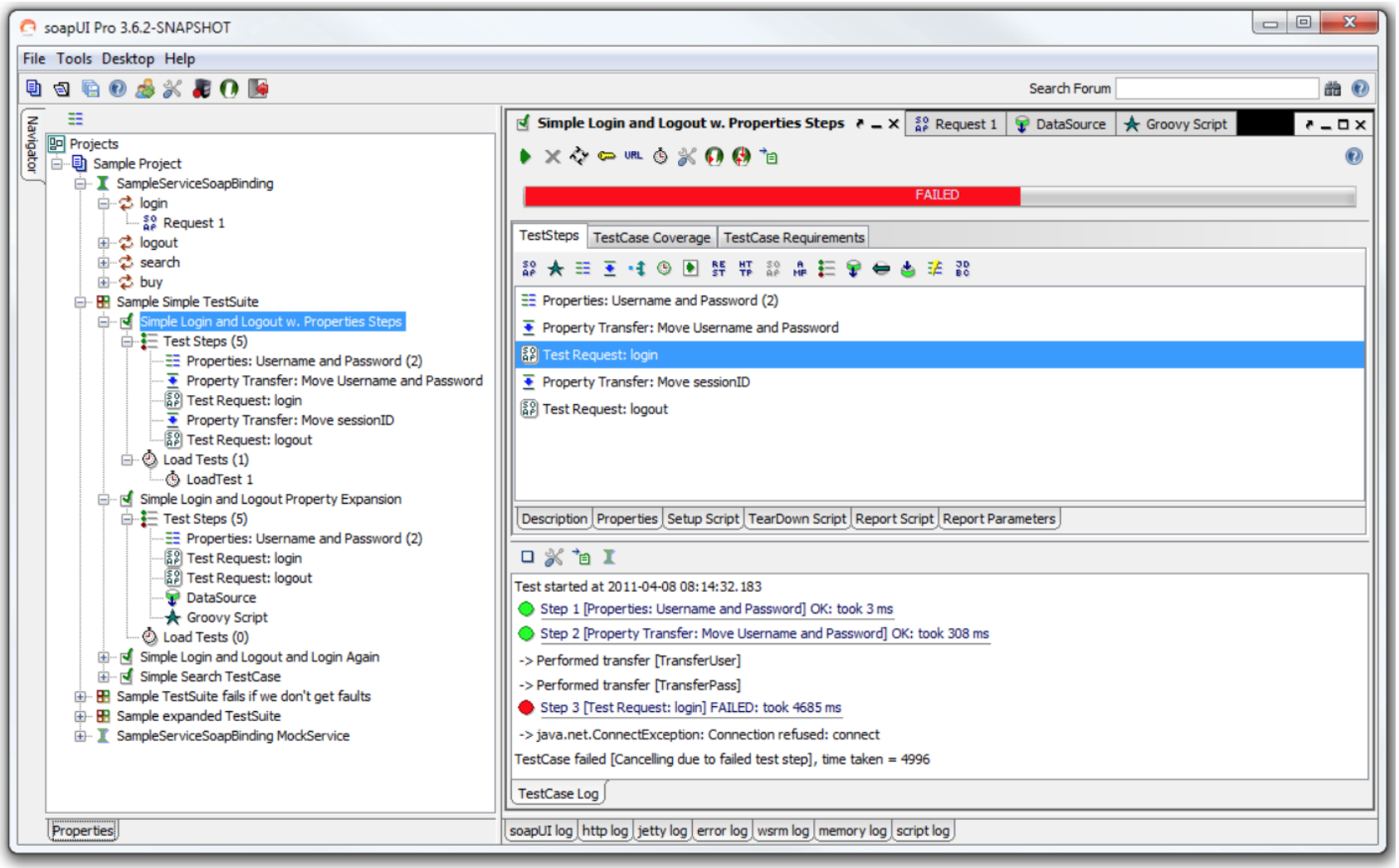
- Apigee
- ProgrammableWeb

Cache:

- Varnish Cache
- nginx
- Yslow (ferramenta par testar cache)
- Apache mod_cache



SOA e Web Services – Ferramentas – SOAPUI



Requisição SOAP

```
<?xml version="1.0"?>
<soap:Envelope xmlns:soap="http://www.w3.org/2001/12/soap-envelope"
    soap:encodingStyle="http://www.w3.org/2001/12/soap-encoding">
  <soap:body pb="http://www.acme.com/phonebook">
    <pb:GetUserDetails>
      <pb:UserID>12345</pb:UserID>
    </pb:GetUserDetails>
  </soap:Body> </soap:Envelope>
```

Requisição REST

```
GET http://www.acme.com/phonebook/UserDetails/12345
```

XML + SOAP vs. JSON + REST

Item	SOAP	REST
Estrutura	Protocolo baseado em XML	Protocolo baseado no estilo arquitetural
Comunicação de Dados	Utiliza um padrão de mensagens baseado em XML	Utiliza XML ou JSON para envio e recebimento dos dados
Comunicação	Invoca serviços por meio de HTTP e RPC (remote procedure call)	Baseado 100% em HTTP e nas URIs
Formato da Mensagem	Resultado codificado no padrão SOAP	Resultado facilmente interpretado por um humano
Compatibilidade com Javascript	Complexa	Simplificada
Desempenho	Médio	Alto