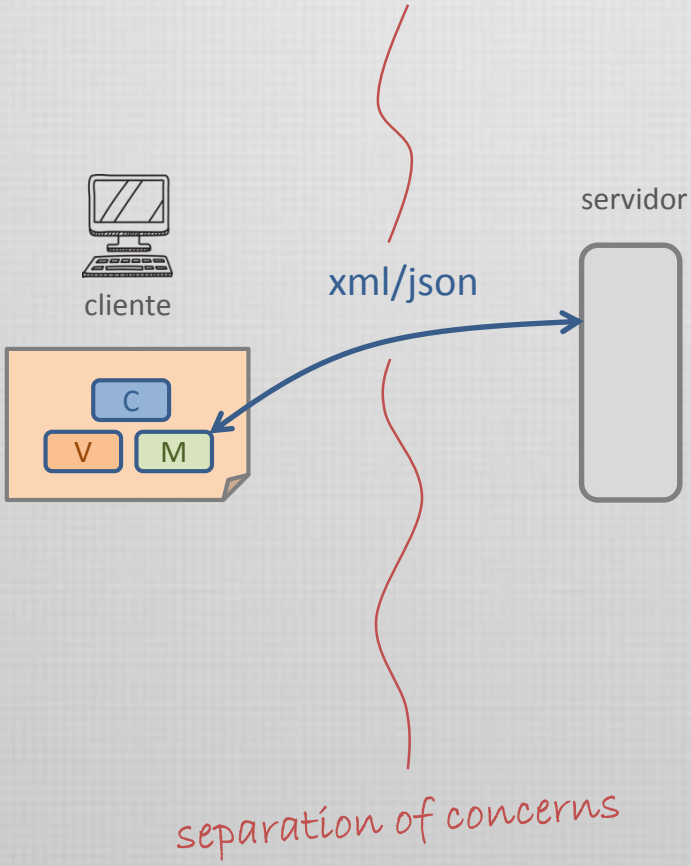
Ingeniería de Aplicaciones Web

Diego C. Martínez

Departamento de Ciencias e Ingeniería de la Computación
Universidad Nacional del Sur

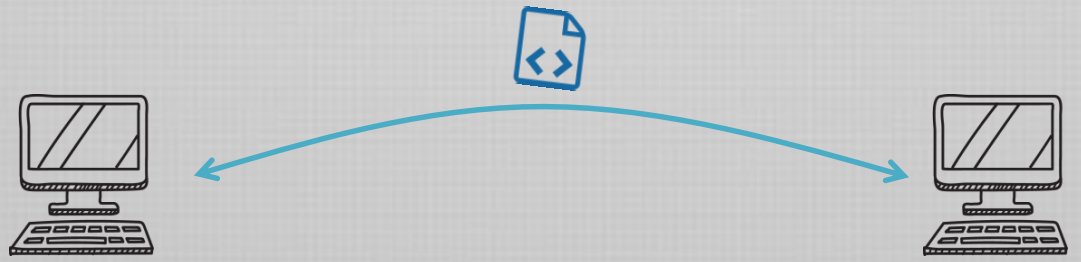
Balance cliente - servidor



¿qué es un *servicio web*?

En términos generales,
un servicio web es una aplicación accedida remotamente usando protocolos de Internet, y que utiliza XML(JSON) como mecanismo de mensajes.

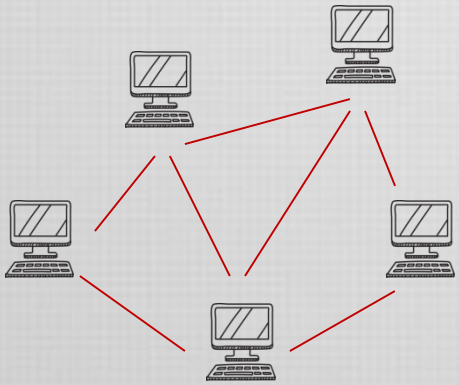
No tiene dependencias de ningún sistema operativo o lenguaje de programación.



SOAP

SOAP

SOAP → *Simple Object Access Protocol*



intercambio de
información estructurada y tipada
entre puntos de un
ambiente distribuído descentralizado

Es un lenguaje que define el
formato de mensajes

independiente de la plataforma
basado en XML
con facilidades de extensión

--------→ *intercomunicación de aplicaciones*

SOAP

1998 — Microsoft, *IBM, Lotus (y otras)*
Primera versión, enfocada en definir un sistema de tipos

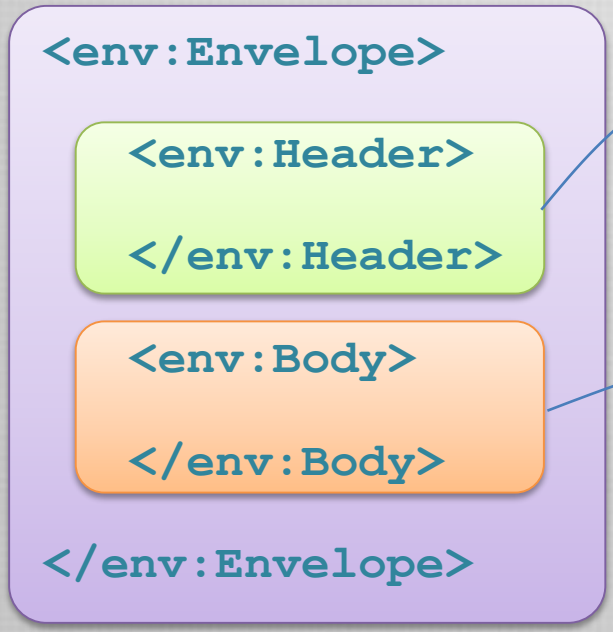
1999 — Primera especificación - nombre SOAP.
basado en *XML Schema Type System + headers de protocolos*
"vendor wars"

2003 — *SOAP - W3C Recommendation*

2007 — *SOAP 1.2 - Messaging Framework*
No es más un acrónimo.

SOAP - estructura

La estructura general de un mensaje SOAP es simple.
Se lo denomina SOAP Envelope y *es un documento XML sin DTD ni PI*



Información de control, formada por elementos denominados *header blocks*. (descripciones del mensaje, instrucciones de procesamiento, etc)
Es opcional

Transporta la información que es relevante para los extremos de la comunicación.
Es obligatorio

SOAP en HTTP

SOAP es lo suficientemente general como para usar otros protocolos de Internet.

En el contexto de los servicios web, los mensajes SOAP serán enviados via HTTP.

El mensaje HTTP apropiado es POST.

```
POST /soap HTTP/1.1
Host: localhost
Connection: Keep-Alive
User-Agent: PHP-SOAP/5.3.1
Content-Type: application/soap+xml; charset=utf-8
Content-Length: 471

<?xml version="1.0" encoding="UTF-8"?>
<env:Envelope xmlns:env="http://www.w3.org/2003/05/soap-envelope"
      xmlns:ns1="http://example.localhost/index/soap"
      xmlns:xsd="http://www.w3.org/2001/XMLSchema"
      xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
      xmlns:enc="http://www.w3.org/2003/05/soap-encoding">
<env:Body>
     <ns1:getLibro env:encodingStyle="http://www....soap-encoding">
          <param0 xsi:type="xsd:string">1456594788</param0>
     </ns1:getLibro>
</env:Body>
</env:Envelope>
```

getLibro(1456594788)

SOAP - ejemplo

```
<s:Envelope xmlns:s="http://www.w3.org/2001/06/soap-envelope">

 <s:Header>
   <m:transaction xmlns:m="soap-transaction" s:mustUnderstand="true">
     <transactionID>1234</transactionID>
   </m:transaction>
 </s:Header>

 <s:Body>
     <n:purchaseOrder xmlns:n="urn:OrderService">
          <from>
               <person>Juan Perugia</person>
          </from>
          <to>
               <person>Steven Spielberg</person>
          </to>
          <order>
               <quantity>1</quantity>
               <item>Casette Video</item>
          </order>
     </n:purchaseOrder>
</s:Body>

</s:Envelope>
```

SOAP - ejemplo

```
<?xml version='1.0' ?>
<env:Envelope xmlns:env="http://www.w3.org/2003/05/soap-envelope">
 <env:Header>
  <m:reservation xmlns:m="http://travelcompany.example.org/reservation"
          env:role="http://www.w3.org/2003/05/soap-envelope/role/next"
           env:mustUnderstand="true">
        <m:reference>uuid:093a2da1-q345-ba5d-pqff98fe8j7d</m:reference>
        <m:dateAndTime>2001-11-29T13:20:00.000-05:00</m:dateAndTime>
  </m:reservation>
  <n:passenger xmlns:n="http://mycompany.example.com/employees"
          env:role="http://www.w3.org/2003/05/soap-envelope/role/next"
           env:mustUnderstand="true">
        <n:name>Åke Jógvan Øyvind</n:name>
  </n:passenger>
 </env:Header>
 <env:Body>
  <p:itinerary xmlns:p="http://travel.example.org/reservation/travel">
   <p:departure>
     <p:departing>New York</p:departing>
     <p:arriving>Los Angeles</p:arriving>
   </p:departure>
   <p:return>
     <p:departing>Los Angeles</p:departing>
     <p:arriving>New York</p:arriving>
   </p:return>
  </p:itinerary>
  </env:Body>
</env:Envelope>
```

SOAP – ejemplo - fault

```
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas..../envelope/">
<SOAP-ENV:Body>
<SOAP-ENV:Fault>
     <faultcode>ns1:DBError</faultcode>
     <faultstring>A database error has occurred</faultstring>
     <detail>
          <ns1:DBUnavailableFault>
          <DBMessage>Unable to connect to database</DBMessage>
          <RetryInMinutes>60</RetryInMinutes>
          </ns1:DBUnavailableFault>
     </detail>
</SOAP-ENV:Fault>
</SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

WSDL

WSDL es una gramática en XML utilizada para describir servicios web
Significa Web Services Description Language.
No es estrictamente necesario utilizar WSDL, aunque es lo recomendable.

El documento WSDL debe definir
el servicio completo,
como una colección de operaciones
que reciben y/o retornan datos estructurados.
Esta descripción es *abstracta*

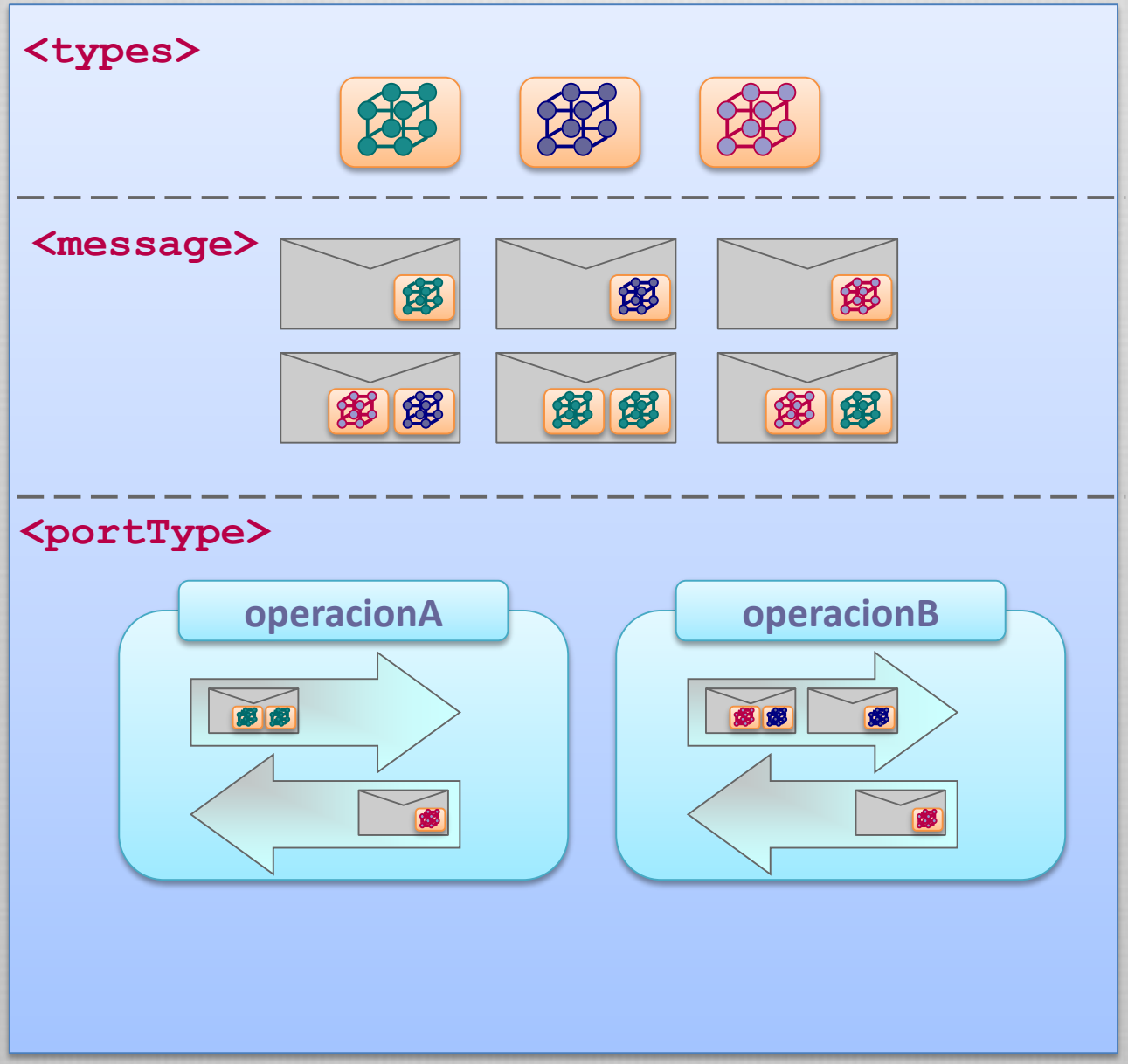
Naturalmente, es necesario describir la *forma de comunicación*.
El documento WSDL debe definir, además,
qué tipo de protocolo de transporte se utilizará y
cuál es el formato de los mensajes que se intercambiarán.
Esta la parte *concreta* de la descripción del servicio

El nivel de abstracción buscado y la modularidad en las descripciones hace que los documentos WSDL sean algo extensos y confusos.

WSDL – partes

```
<definitions xmlns="http://schemas.xmlsoap.org/wsdl/">

  <types>
     <!-- definiciones de tipos utilizados -->
  </types>

  <message>
     <!-- definicion abstracta de los datos que se transmiten -->
  </message>

  <portType>
     <!-- operaciones abstractas + input y output de mensajes -->
  </portType>

  <binding>
     <!-- protocolo concreto y formato de datos especifico -->
  </binding>

  <service>
     <!-- ubicaciones y bindings del servicio -->
  </service>

</definitions>
```

WSDL – definiciones *abstractas*



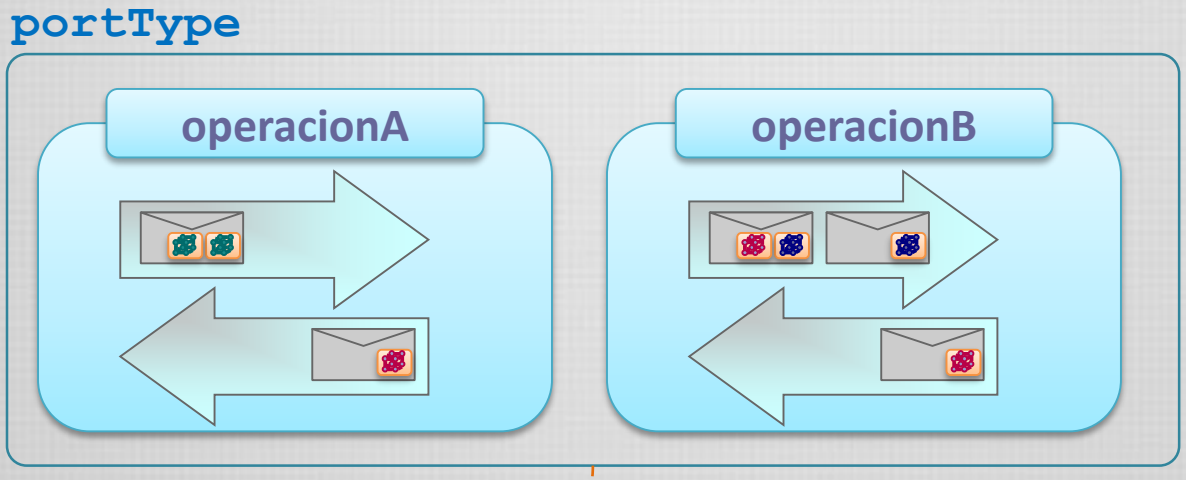
WSDL – definiciones *abstractas - tipos*

```
<?xml version="1.0"?>
<definitions name="StockQuote"   ...namespaces...>

<types>
  <schema ...namespaces xsd...>

   <element name="TradePriceRequest">
      <complexType>
         <all>
            <element name="tickerSymbol" type="string"/>
         </all>
      </complexType>
   </element>

   <element name="TradePrice">
      <complexType>
         <all>
            <element name="price" type="float"/>
         </all>
      </complexType>
   </element>

  </schema>
</types>
```

WSDL – definiciones *abstractas* – *mensajes + portTypes*

```
<message name="GetLastTradePriceInput">
  <part name="body" element="xsd1:TradePriceRequest"/>
</message>

<message name="GetLastTradePriceOutput">
  <part name="body" element="xsd1:TradePrice"/>
</message>


<portType name="StockQuotePortType">
  <operation name="GetLastTradePrice">
    <input message="tns:GetLastTradePriceInput"/>
    <output message="tns:GetLastTradePriceOutput"/>
  </operation>
</portType>
```

WSDL – definiciones concretas – *bindings*



binding

Servicio provisto

Transporte y formato

SOAP	MIME	SOAP	XX
HTTP	SMTP	SMTP	YY

Una vez elegido el formato, debe indicarse cómo se formatea cada operación y sus mensajes.
binding portType - bindings operaciones - bindings mensajes

WSDL – definiciones concretas – *bindings y service*

```
<binding name="StockQuoteSoapBinding" type="tns:StockQuotePortType">

  <soap:binding style="document"
                transport="http://schemas.xmlsoap.org/soap/http"/>

  <operation name="GetLastTradePrice">
    <soap:operation soapAction="http://example.com/GetLastTradePrice"/>
      <input>
        <soap:body use="literal"/>
      </input>
      <output>
        <soap:body use="literal"/>
      </output>
  </operation>

</binding>

<service name="StockQuoteService">
   <documentation>Mi servicio web SOAP</documentation>
   <port name="StockQuotePort" binding="tns:StockQuoteBinding">
       <soap:address location="http://example.com/stockquote"/>
   </port>
</service>
```

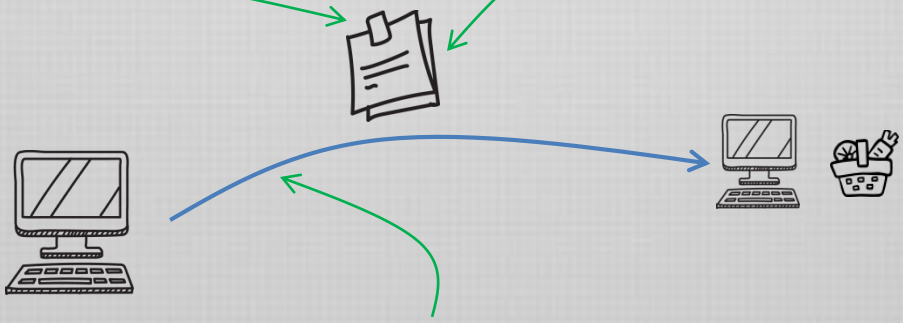
REST

REST

REST significa Representational State Transfer.

No es un estándar como los anteriores, sino una abstracción de elementos de una arquitectura dentro de un sistema hipermedial distribuido.

Básicamente, traslada la configuración de un *request*, a las tecnologías ya provistas por la web.



MIME type
text/xml
application/json

XML / JSON
El formato elegido para la transferencia de los recursos solicitados

HTTP
Las acciones a realizar: GET, HEAD, POST, PUT y DELETE.

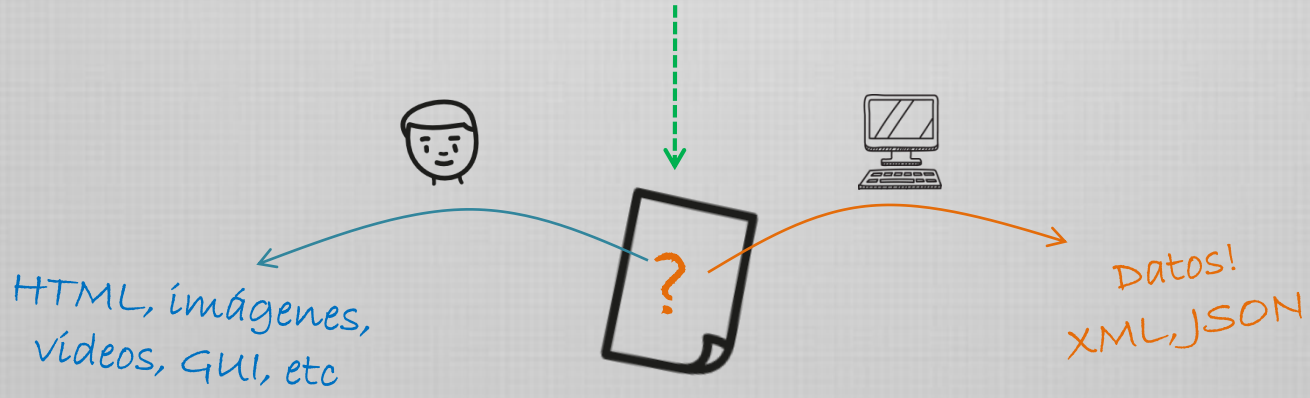
URI
El identificador del recurso que representa el servicio web

REST

REST ve el mundo como una colección de recursos. → *abstracción de "datos"*

identificables por
medio de URI o URL

http://unhost.com/ligafutbol/equipos/deportivoX



REST – mensajes HTTP

REST asigna un rol especial a los mensajes HTTP, consistente con su uso web.

GET *Retrieve*

Recupera la representación de un recurso

HEAD *Retrieve*

Recupera metadatos de la representación de un recurso

POST *Create*

Estrictamente, crea un recurso.
Se usa también para actualizar un recurso o borrarlo

PUT *Update*

Actualiza un recurso.
Usualmente reemplazado por el método POST

DELETE *Delete*

Elimina un recurso.
Usualmente reemplazado por el método POST

REST – URLs

http://www.autopartes.com/partes

```
<?xml version="1.0"?>
<p:partes xmlns:p="http://www...." xmlns:xlink="http://...">
  <p:autoparte id="ABC1" xlink:href="http://www.autopartes.com/partes/ABC1"/>
  <p:autoparte id="DEF2" xlink:href="http://www.autopartes.com/partes/DEF2"/>
  <p:autoparte id="GEW3" xlink:href="http://www.autopartes.com/partes/GEW3"/>
  <p:autoparte id="KLM1" xlink:href="http://www.autopartes.com/partes/KLM1"/>
</p:partes>
```

http://www.autopartes.com/partes/DEF2

```
<?xml version="1.0"?>
<p:partes xmlns:p="http://www...." xmlns:xlink="http://...">
  <p:autoparte>
      <p:id>DEF2</p:id>
      <p:desc>paragolpe delantero</p:desc>
      <p:auto>Fiat 600</p:auto>
      <p:stock>5</p:stock>
  </p:autoparte>
</p:partes>
```

REST – URLs

http://maps.googleapis.com/maps/api/geocode/xml?address=Bahia Blanca

```
<?xml version="1.0" encoding="UTF-8"?>
<GeocodeResponse>
    <status>OK</status>
    <result>
        <type>locality</type>
        <type>political</type>
        <formatted_address>
                Bahia Blanca, Buenos Aires, Argentina
        </formatted_address>
        <address_component>
            <long_name>Bahía Blanca</long_name>
            <short_name>Bahía Blanca</short_name>
            <type>administrative_area_level_2</type>
            <type>political</type>
         </address_component>
...
```

REST – URLs

http://services.explorecalifornia.org/rest/tours.php?packageid=5

```
<?xml version="1.0"?>
<tours>
  <tour>
    <tourId>14</tourId>
      <tourTitle>2 Days Adrift the Salton Sea</tourTitle>
      <packageId>5</packageId>
      <packageTitle>From Desert to Sea</packageTitle>
        <description>
            The Salton Sea is saltier than the Pacific,
            an unusual feat for inland body of water.
            And even though its salinity has risen over
            the years, due in part to lack …
```

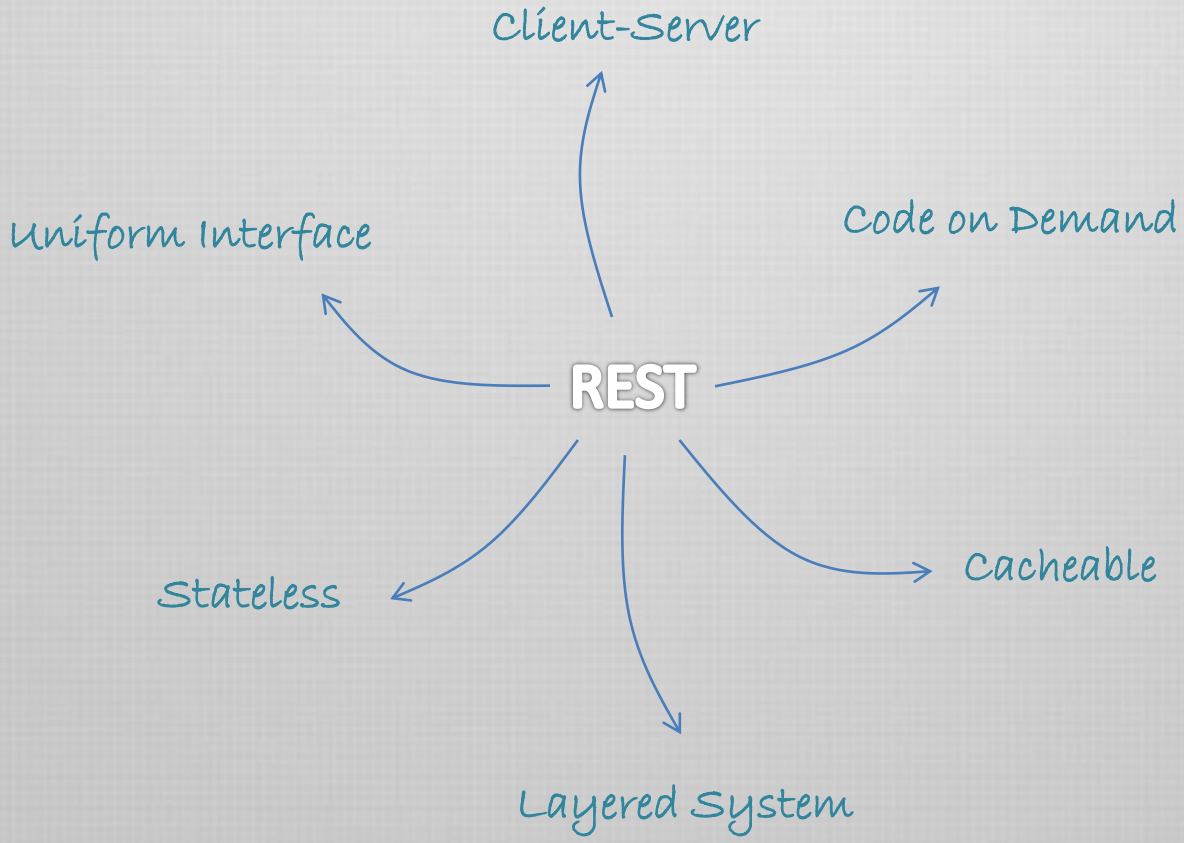
REST – URLs

```
POST http://services.explorecalifornia.org/rest/tours.php HTTP/1.0
Host: services.explorecalifornia.org
Accept: text/xml
Content-Type: text/xml
Content-Length: 165

<?xml version="1.0"? Encoding="utf-8"">
<parameters>
        <packageid>5<packageid>
</parameters>
```

REST constraints

"an architectural style consisting of the set of constraints applied to elements within the architecture"

Roy Fielding



REST constraints

“an architectural style consisting of the set of constraints applied to elements within the architecture”

REST

REST constraints

"an architectural style consisting of the set of constraints applied to elements within the architecture"

REST → Uniform Interface

principio de ingeniería: generalidad en la interfaz
simplifica la arquitectura
las implementaciones se desacoplan de los servicios que proveen

identificación de recursos

manipulación de recursos por su representación

mensajes auto descriptivos

Hypermedia as the Engine of Application State

REST constraints

"an architectural style consisting of the set of constraints applied to elements within the architecture"

REST → Uniform Interface

identificación de recursos

manipulación de recursos por su representación

mensajes auto descriptivos

Hypermedia as the Engine of Application State

REST constraints

"an architectural style consisting of the set of constraints applied to elements within the architecture"

REST → Uniform Interface

identificación de recursos

manipulación de recursos por su representación

mensajes auto descriptivos

Hypermedia as the Engine of Application State

Los recursos son identificados por medio de un URI
Los recursos están conceptualmente separados de sus representaciones.

REST constraints

"an architectural style consisting of the set of constraints applied to elements within the architecture"

REST → Uniform Interface

identificación de recursos

manipulación de recursos por su representación

mensajes auto descriptivos

Hypermedia as the Engine of Application State

Una representación = datos + metadatos
Tipo de datos = "media type"
La representación del recurso R puede variar en el tiempo $\longrightarrow M_R(t)$
La representación de un recurso puede ser el estado actual de un recurso o el estado deseado.

REST constraints

"an architectural style consisting of the set of constraints applied to elements within the architecture"

REST → Uniform Interface

identificación de recursos

manipulación de recursos por su representación

mensajes auto descriptivos

Hypermedia as the Engine of Application State

Los mensajes deben contener toda la información necesaria para procesar el mensaje.
(Internet Media types, cache policies, etc)

REST constraints

"an architectural style consisting of the set of constraints applied to elements within the architecture"

REST → Uniform Interface

identificación de recursos

manipulación de recursos por su representación

mensajes auto descriptivos

Hypermedia as the Engine of Application State

HATEOAS

El modo de comunicación distribuída es hipermedial

Recursos desde y hacia los clientes.

Recursos linkeados (incluyen API)

Control de cache

REST constraints

"an architectural style consisting of the set of constraints applied to elements within the architecture"

Cada *request* del cliente debe contener toda la información necesaria para comprender el *request*

No participa ningún contexto almacenado en el servidor

favorece escalabilidad

REST constraints

"an architectural style consisting of the set of constraints applied to elements within the architecture"

Los datos deben ser implícitamente o explícitamente identificados como cacheables o no-cacheables

reduce tráfico de red y favorece performance
favorece escalabilidad

REST constraints

"an architectural style consisting of the set of constraints applied to elements within the architecture"

"separation of concerns"

A los clientes no les incumbe aspectos de almacenamiento de datos
A los servidores no les incumbe el estado del usuario o su interfaz

servidores y clientes reemplazables
favorece escalabilidad

REST constraints

"an architectural style consisting of the set of constraints applied to elements within the architecture"

Arquitecturas en los extremos basadas en capas jerárquicas
Cada capa tiene su responsabilidad
Independencia de sustratos

reduce complejidad
favorece escalabilidad
riesgo de overhead

REST constraints

"an architectural style consisting of the set of constraints applied to elements within the architecture"

Los servidores puede extender la funcionalidad del cliente
El servidor transfiere lógica al cliente (scripts, applets, etc)
Es el único requerimiento opcional

simplifica la complejidad del cliente
clientes extensibles