

Capítulo 2

Arquitecturas de Web Services y estándares involucrados

Al finalizar el capítulo, el alumno podrá

- Identificar los principales componentes de una arquitectura de Web Services.
- Entender las diferencias existentes entre las arquitecturas Web Services SOAP y REST.
- Conocer los estándares involucrados en el desarrollo de Web Services

Temas

1. Fundamentos de Web Services
2. SOAP- Based Web Services
3. REST- Style Web Services
4. SOAP- Based vs REST-Style Web Services

1. Fundamentos de Web Services

Fundamentos de Web Services

- Aplicaciones distribuidas desarrolladas bajo los protocolos y estándares de intercambio de información
- Implementadas en distintos lenguajes de programación
- Ejecutadas sobre múltiples plataformas
- Las principales características son:
 - Diseño modular
 - Infraestructura abierta
 - Independiente de la plataforma y lenguaje de programación

```

graph LR
    A[DEVICE A  
REQUESTOR] -- REQUEST DATA --> B[DEVICE B  
SERVICE PROVIDER]
    subgraph NETWORK
        direction TB
        C((NETWORK))
    end
    A --- C
    C --- B
    B -- RESPONSE DATA --> A
            
```

1 - 4
Copyright © Todos los Derechos Reservados - Cibertec Perú SAC.

1.1. Introducción a los Web Services

Existen múltiples definiciones sobre lo que son los Web Services (Servicios Web), lo que es sinónimo de su complejidad para dar una adecuada definición que englobe todo lo éstos implican.

En su concepto más general, se puede decir que los Web Services son aplicaciones distribuidas que se basan en una serie de protocolos y estándares para intercambiar información en redes de computadoras.

A continuación se detallan algunas de las principales características de los Web Services.

Infraestructura abierta

Los servicios Web están basados en estándares abiertos y reconocidos por la industria del software. Utilizan protocolos tales como HTTP, XML, entre otros para intercambiar información entre redes de computadoras.

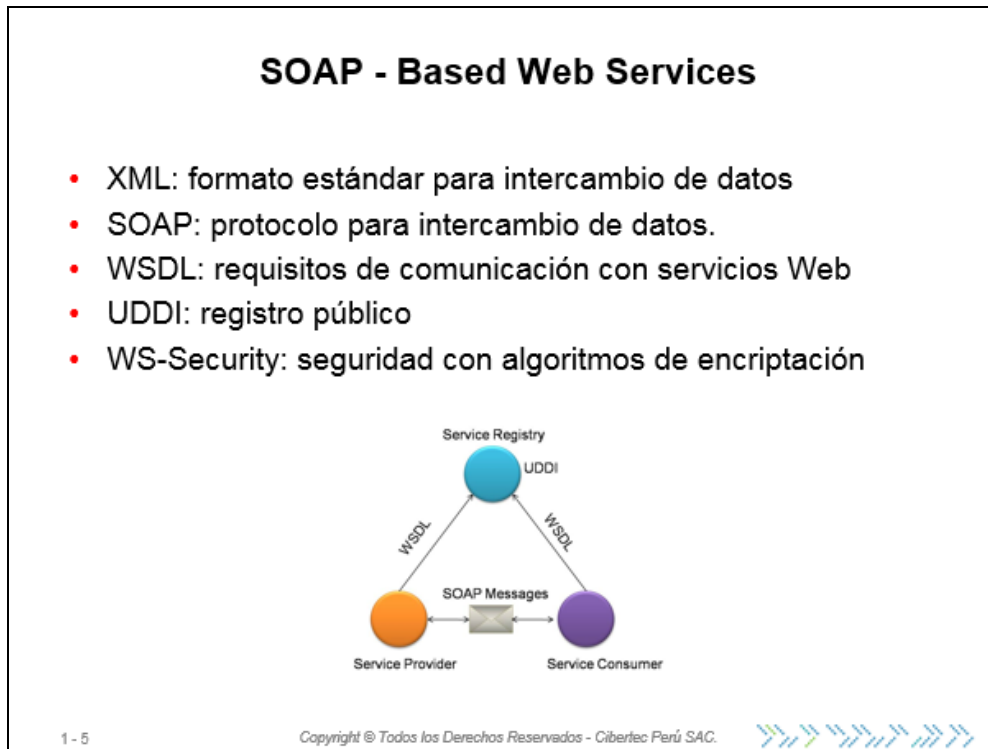
Independencia de plataforma y lenguaje de programación

Los servicios Web y sus clientes son interoperables entre sí, es decir, pueden estar desarrollados en lenguajes de programación diferentes como C/C++, C#, Java, Perl, Ruby entre otros. Cada lenguaje contiene sus librerías y frameworks que soportan el desarrollo de servicios Web.

Diseño modular

Los servicios Web están diseñados para ser modulares en diseño, es decir, nuevos servicios pueden ser generados e integrados en diferentes capas y coexistir con servicios existentes.

2. SOAP- Based Web Services



2.1. El protocolo SOAP (Simple Object Access Protocol)

SOAP es un protocolo de acceso a los web services.

- Permite establecer la comunicación entre las aplicaciones que publican o consumen los “Web Services”.
- SOAP especifica el formato de los mensajes.
- Es independiente de plataforma y lenguaje de programación.
- El mensaje SOAP está compuesto por un Envelope (sobre), cuya estructura está formada por los siguientes elementos: Header (cabecera) y Body (cuerpo).

Los requerimientos básicos para llevar una buena comunicación son los siguientes:

- Formato : SOAP+XML
- Transporte : Tecnologías Web (http, smtp,etc)
- Interfaces : WSDL
- Descubrimiento y búsqueda : UDDI
- Seguridad : WS-Security

La estructura básica de un mensaje SOAP consiste de:

- Un elemento “Envelope”, que es el elemento raíz de todo mensaje SOAP que identifica un documento XML como un mensaje SOAP.
- Un elemento opcional “Header”
 - El “Header”, de estar presente, debe ser el primer elemento del “Envelope”.
 - Provee un mecanismo de extensión que permite incluir información extra en mensajes SOAP (seguridad, transacciones, etc.).
 - Puede contener varios “header blocks” que son una forma de agrupar lógicamente la información.
- Un elemento requerido “Body”: Contiene la información a ser intercambiada entre el cliente y servicio. En el “Body”, típicamente se especifica lo siguiente:
 - Una solicitud para efectuar cierta operación
 - La respuesta a cierta solicitud que puede ser un resultado o un error (fault)
- Un elemento “Fault”. Éste indica la ocurrencia de un error en el procesamiento del mensaje.
 - Comprensión del mensaje: Error de sintaxis
 - Infraestructura: ejemplo HTTP 500 Internal Server Error
 - Negocio: por ejemplo, no existe saldo en la cuenta X

SOAP Request :

```
<?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>
    <m:ObtenerPrecio
xmlns:m="http://www.w3schools.com/prices">
      <m:Elemento>Manzanas</m:Elemento>
    </m:ObtenerPrecio>
  </soap:Body>
</soap:Envelope>
```

Un diseño basado en SOAP es adecuado cuando:

- Se establece un contrato formal para la descripción de la interfaz que el servicio ofrece: el Lenguaje de Descripción de Servicios Web (WSDL) permite describir con detalles el servicio Web.
- La arquitectura debe abordar requerimientos complejos no funcionales. Muchas especificaciones de servicios web abordan tales requisitos y establecen un vocabulario común para ellos.
- La arquitectura necesita manejar procesamiento e invocación asíncrona. En estos casos, la infraestructura proporcionada por APIs como JAX-WS, junto con la asincronía por el lado cliente permitirán el soporte de estas características.

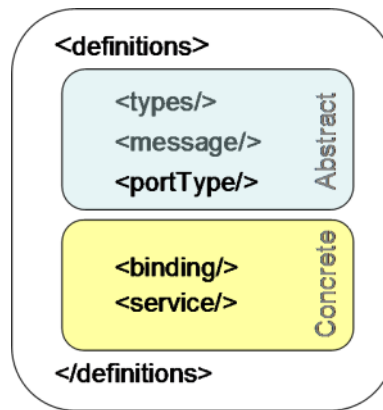
2.2. El Lenguaje de descripción de servicios web (WSDL)

Permite que un servicio y un cliente establezcan un acuerdo en lo que se refiere a los detalles de transporte de mensajes y su contenido, a través de un documento procesable por dispositivos.

En dicho documento, se describe un web service.

- ¿Dónde está ubicado el servicio?
- ¿Qué operaciones tiene?
- ¿Qué mensajes de entrada/salida recibe/responde?
- ¿Cuál es la estructura de los mensajes?
- ¿Qué protocolos de transporte hay que utilizar?

Un documento WSDL puede ser dividido en dos partes: concreta y abstracta. Las partes concretas siempre pueden ser definidas en dos o más archivos, donde la parte concreta importa de la parte abstracta. Esta separación permita la máxima reusabilidad y flexibilidad al definir los servicios como se muestra en la siguiente figura.



La descripción abstracta

Describe de forma general la estructura de la interfaz del web service, que incluye operaciones, parámetros y tipos de datos abstractos.

Los cuatro elementos XML que componen la descripción abstracta son los siguientes:

- `<wsdl:types>`: Encapsula todas las definiciones abstractas de tipos de datos.
- `<wsdl:message>`: Representa de forma abstracta los parámetros de entrada y salida para una operación.
- `<wsdl:portType>`: Es el contenedor de todas las operaciones abstractas y describe una interfaz específica del servicio.
- `<wsdl:operation>`: Está embebido dentro del `portType` y especifica las operaciones y sus parámetros de entrada/salida.

La descripción concreta

Se asocia a una descripción abstracta, una dirección de red concreta, un protocolo de comunicación y estructuras de datos concretas.

Los tres elementos XML que componen la descripción concreta son los siguientes:

- `<wsdl:binding>`: Asocia un `—portType`, sus mensajes y operaciones, a un protocolo de transporte y un formato de mensaje.
- `<wsdl:service>`: Es un contenedor de elementos `—port`.
- `<wsdl:port>`: Especifica la dirección de red para un determinado "binding".

La estructura del documento WSDL describe a un servicio web que emplea

Elemento	Definición
<code><types></code>	Los tipos de datos utilizados por el servicio web Para una máxima neutralidad en la plataforma, WSDL emplea la sintaxis de esquemas XML para definir los tipos de datos.
<code><message></code>	Define los elementos de datos de una operación utilizados por el servicio web. Cada mensaje puede consistir de una o más partes. Dichas partes pueden ser comparadas con los parámetros de una invocación a una función en un lenguaje de programación tradicional.
<code><portType></code>	Es el elemento más importante dentro del documento. Describe un servicio web, las operaciones que pueden ejecutarse y los mensajes involucrados. Este elemento puede compararse con una librería de funciones o métodos en cualquier otro lenguaje de programación.
<code><binding></code>	Los protocolos de comunicación utilizados por el servicio web en cada port.

```

<definitions name="StockQuote"
targetNamespace=http://example.com/stockquote.wsdl
xmlns:tns=http://example.com/stockquote.wsdl xmlns:xsd1=http://example.com/stockquot
e.xsd xmlns:soap=http://schemas.xmlsoap.org/wsdl/soap/
xmlns="http://schemas.xmlsoap.org/wsdl/">
  <message name="GetLastTradePriceInput">
    <part name="body" element="xsd1:TradePriceRequest"/>
  </message>
  <portType name="StockQuotePortType">
    <operation name="GetLastTradePrice"> </operation>
  </portType>
  <binding name="StockQuoteSoapBinding" type="tns:StockQuotePortType"> </binding>
  <service name="StockQuoteService">
    <port name="StockQuotePort" binding="tns:StockQuoteBinding">
      <soap:address location="http://example.com/stockquote"/>
    </port>
  </service>
</definitions>

```


3. REST- Style Web Services

REST - Style Web Services

- HTTP: protocolo de comunicación
- REST: estilo arquitectónico basado principalmente en HTTP para transferencia de hypermedia en sistemas distribuidos
- URI: dirección de acceso a la representación de un recurso
- MIME Types: identificador compuesto de dos (2) partes que describe el formato de un recurso en internet

1 - 11

Copyright © Todos los Derechos Reservados - Cibertec Perú S.A.C.



3.1. REST (REpresentational State Transfer)

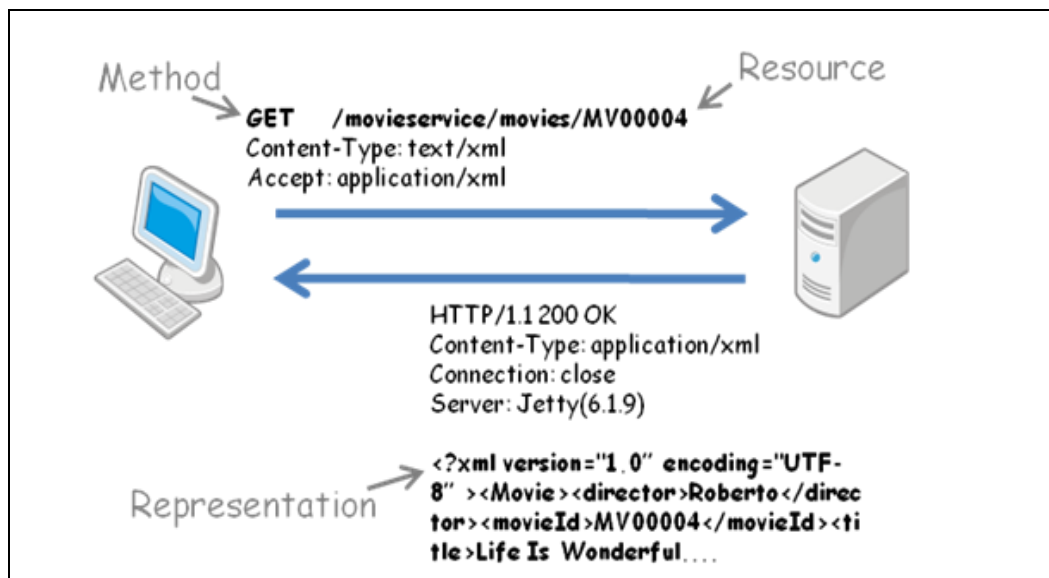
Está compuesto por una colección de principios de arquitectura de software para construir sistemas distribuidos basados en mecanismos que definen y acceden a recursos como internet.

REST ofrece un estilo simple, interoperable y flexible de contruir “Web Services” sin adherirse a una tecnología en particular.

El término REST es muchas veces empleado para describir un estilo de transmisión de datos basado en HTTP sin agregar capas semánticas ni de manejo de sesiones.

En la mayoría de casos, REST se basa en los siguientes estándares para su implementación:

- Formato: Tipos MIME tales como text/html, application/xml, etc.
- Identificación de recursos : URI (Uniform Resource Identifier)
- Transporte: HTTP
- Manipulación de recursos: Basado en métodos del protocolo HTTP tales como GET, PUT, POST, DELETE



Los principios de arquitectura sobre los que basa REST son definidos como:

Identificación de recursos

La unidad mínima de abstracción para los datos en REST es el llamado “recurso”, cada recurso debe ser identificado y ubicado mediante una URI (Uniform Resource Identifier).

Representación de los recursos

La interacción con los servicios se realiza mediante representación del mismo servicio. Es decir, un recurso es referenciado por una URI y puede tener múltiples formatos. Diferentes plataformas necesitan diferentes formatos. Por ejemplo, los navegadores necesitan HTML, Javascript necesita JSON y Java puede necesitar XML o algún otro tipo de formato.

Hypermedia como un mecanismo de estados de aplicación (HATEOAS)

El estado actual de una aplicación web debería ser capturada en uno o más documentos de hipertexto, y residir tanto en el cliente como en el servidor. El servidor conoce sobre el estado de sus recursos, aunque no intenta seguirle la pista las sesiones individuales de los clientes. Esta es la misión del navegador, el sabe como navegar de recurso en recurso y recoger información que él necesita o cambiar el estado que necesita cambiar.



4. SOAP - Based vs REST - Style Web Services

SOAP - Based vs REST- Style Web Services

- Es la primera decisión a tomar para construcción un servicio web. SOAP-Based Web Services es un caso especial que corresponde al REST-Style Web Services, con la diferencia que están basados únicamente en HTTP y SOAP+XML
- Las ventajas y desventajas de cada una de las arquitecturas pueden ser favorables para algunos sistemas, pero no para otros, por lo tanto, la decisión dependerá estrictamente de los requerimientos y limitaciones de las aplicaciones

1 - 14 Copyright © Todos los Derechos Reservados - Cibertec Perú S.A.C. 

4.1. SOAP vs REST

A continuación se muestra un cuadro comparativo con información acerca de las características más relevantes de REST y SOAP.

REST	SOAP
<ul style="list-style-type: none"> • Expone “recursos” que simbolizan información. • Utiliza métodos HTTP: (GET/PUT/POST/DELETE) • Se enfatiza en la comunicación punto a punto sobre HTTP. • Se enfatiza en no mantener estados en la comunicación para promover bajo acoplamiento. 	<ul style="list-style-type: none"> • Expone operaciones que representan lógica de negocio. • Utiliza únicamente HTTP POST para la transferencia de información. • Utiliza HTTP GET para la descripción del servicio o WSDL. • Soporta únicamente XML y datos adjuntos. • Soporta el manejo de la comunicación con y sin estado.
<ul style="list-style-type: none"> • Puede optimizar el consumo de recursos y ancho de banda ya que los mensajes son más ligeros (no necesariamente utiliza SOAP y XML para el intercambio de información). • El cliente no necesita información de enrutamiento a partir de la URI inicial. 	<ul style="list-style-type: none"> • Ha sido utilizado ampliamente para el desarrollo de web services y cuenta con mayor cantidad de tecnologías asociadas. • Se puede asociar con servicios empresariales como <ul style="list-style-type: none"> – Alta disponibilidad con WS-Reliable Messaging (WS-RM) – Transacciones con WS-Atomic Transactions (WS-AT) – Seguridad con WS-Security

¿ Cual de los dos seleccionar?

Es la primera decisión en la construcción de un servicio web, por lo que es importante comprender las características, ventajas y desventajas de cada una.

Los SOAP-based “*web services*” son con un caso especial de REST-style web services con la diferencia que están basados únicamente en HTTP y SOAP+XML.

En conclusión, las ventajas y desventajas de cada una de las arquitecturas pueden ser favorables para algunos sistemas pero no para otros, por lo tanto, la decisión dependerá estrictamente de los requerimientos y limitaciones de las aplicaciones.