Tema 4. Servicios WEB

SCS – Sistemas Cliente/Servidor
4º informática

`http://ccia.ei.uvigo.es/docencia/SCS`

octubre 2008

4.1 Servicios WEB

Un Servicio Web es un componente software que puede ser registrado, descubierto e invocado mediante protocolos estándares de Internet.

- Permiten exponer y hacer disponibles funcionalidades (servicios) de los sistemas informáticos de las organizaciones mediante tecnologías y protocolos WEB estándar.
- Cada Servicio Web se responsabilida de realizar un conjunto de funciones concretas y bien definidas
- Servicios Web actúan como componentes independientes que se pueden integrar para formar sistemas distribuidos complejos

Definición: (del *World Wide Web Consortium* [W3C])

> "Un Servicio Web (*Web Service [WS]*) es una aplicación software identificada por un URI (*Uniform Resource Identifier*), cuyas interfaces se pueden **definir**, **describir** y **descubrir** mediante documentos XML. Los Servicios Web hacen posible la interacción entre "agentes" software (aplicaciones) utilizando mensajes XML intercambiados mediante protocolos de Internet."

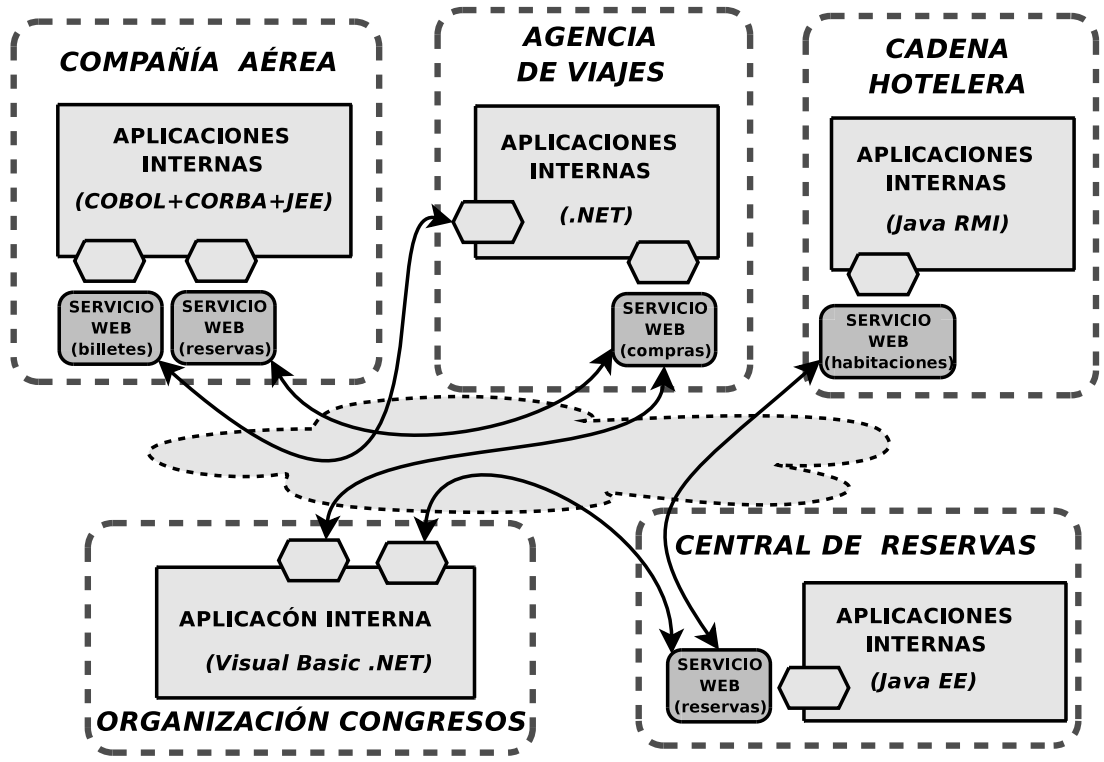
- Puntos clave: { interoperabilidad; uso de estándares abiertos; mínimo acoplamiento }
- *Interoperabilidad:* distintas aplicaciones, en lenguajes de programación diferentes, ejecutadas sobre cualquier plataforma, pueden utilizar los Servicios Web para intercambiar datos
  - La interoperabilidad se consigue mediante el uso de estándares abiertos.
  - Servicios Web se asientan sobre protololos y estándares ya existentes y muy difundidos (HTTP, XML, etc)
  - Uso de protocolos específicos extensibles $\Rightarrow$ no imponen restricciones sobre las aplicaciones a las que dan acceso ni sobre las tecnologías que las implementan (independencia de lenguaje y de plataforma)
  - OASIS y W3C: organizaciones responsables de definir la arquitectura y estándares para los Servicios Web

- Pueden verse como una *evolución de los mecanismos RPC*
  - Uso de protocolos estándar de internet (HTTP, SMTP) como mecanismo para el transporte de los mensajes (invocación, respuesta, ...)
    - Mensajes intercambiados se encapsulan dentro de mensajes HTTP (o SMTP)
    - Evitan problemas con firewalls y filtrado de puertos no privilegiados
    - Para la red el tráfico de Servicios Web es tráfico HTTP (o SMTP) normal
  - Uso de lenguajes basados en XML
    - Los mensajes intercambiados son representados en documentos XML
    - Servicios y métodos remotos son descritos en documentos XML

Escenario típico: Integración de un conjunto de aplicaciones de distintas empresas/organizaciones

- Aplicaciones distribuidas convencionales se basan en el uso de un *middleware* común y centralizado (ORBs en CORBA, RMI en Java, etc)
- Serv. Web permiten superar esa restricción { no exigen *middleware* único común / *middleware* abierto y no centralizado
- Servicios Web ofrecen un punto de entrada a los sistemas de información locales
  - Encapsulan una o más aplicaciones ofreciendo un interfaz único accesible por la Web
  - Ofrecen un interfaz público y estable, independiente de su implementación concreta
  - Facilitan la automatización de las interacciones entre los procesos internos de una organización con el exterior

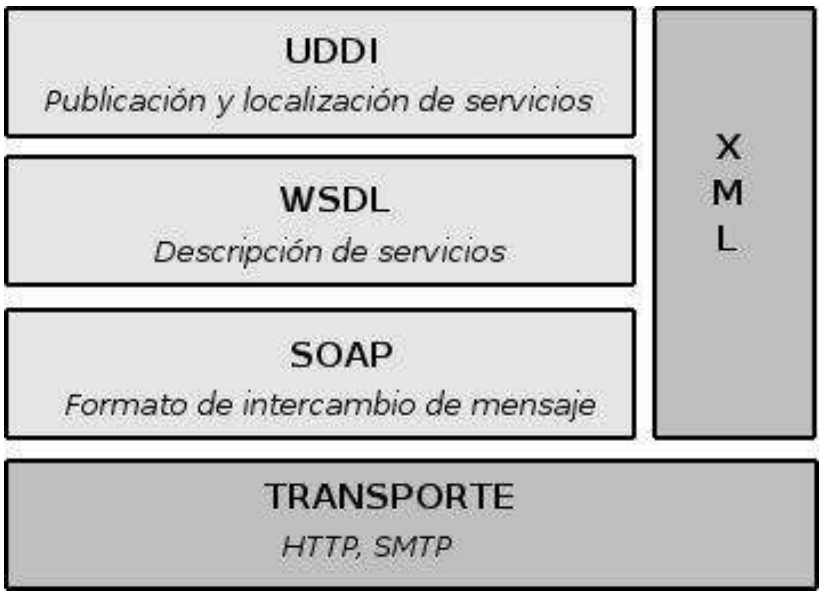
Ejemplo: uso de Servicios Web



Concepto clave: servicio

- Un **servicio** es un procedimiento, un método o un objeto con una interfaz estable y pública que puede ser invocado por un cliente
  - Los Servicios Web amplian esa idea para permitir que esa invocación se realize a través de internet empleando protocolos Web estándar ya existentes
- **Arquitectura Orientada a Servicios (SOA)**
  - Aproximación al diseño de aplicaciones complejas basada en:
    - la identificación de los servicios que ofrecerá
    - la definición de esos servicios
    - la organización de las interacciones entre esos servicios
  - Importancia de las interfaces
    - Descripción rigurosa de las interfaces
    - Tratamiento automático para generar código de implementación
    - **Idea base:** desarrollar el sistema a partir de las interfaces
  - Los servicios ofrecen operaciones a los clientes que deben ser invocadas en un orden determinado para lograr el objetivo deseado
    - Servicios simples *vs.* serv. compuestos (implementados invocando otros servicios)
    - Necesidad de especificar reglas que gobiernen el intercambio (conversación) entre servicios $\Rightarrow$ protocolos de negocio
    - BPEL: *Business Process Execution Language for Web Services*
      - ◇ Lenguaje (derivado de XML) que permite especificar las interacciones entre servicios (normalmente Servicios Web)
      - ◇ Soporta la composición de servicios simples para crear servicios compuestos

4.1.1 Arquitectura básica de protocolos de Servicios Web



Elementos necesarios para la definición de Servicios Web

1. Sintaxis común para todas la especificaciones $\Rightarrow$ uso de XML
   - XML: *eXtensible Markup Language*
     - Estándar para la definición de lenguajes de marcas
     - Flexible y extensible
   - Metalenguaje usado en Servicios Web para especificar los lenguajes y protocolos necesarios
     - Permite definición de lenguajes para $\begin{cases} \text{describir servicios} \\ \text{representar mensajes intercambiados} \end{cases}$
2. Mecanismos de interacción entre extremos $\Rightarrow$ uso de SOAP (*Simple Object Access Protocol*)
   - Necesidad de un formato de mensajes neutro, abierto y extensible
     - representación de mensajes de invocación (argumentos) y respuesta (valor retorno) como documentos XML
   - Especificación del modo de interacción: $\begin{cases} \text{síncrono (RPC: petición-respuesta)} \\ \text{asíncrono (petición)} \end{cases}$
   - Mapeo de los mensajes en el protocolo de transporte (HTTP, SMTP)

3. Lenguaje común para describir los servicios ⇒ uso de WSDL (*Web Service Description Language*)
   - Descripción de los servicios y sus interfaces de forma estándar mediante documentos XML
   - Papel análogo al del IDL en *middleware* convencional
   - Incluye toda la información necesaria para suplir la falta de un *middleware* común centralizado
     - Especifica cada operación disponible, con sus parámetros de entrada y de salida
     - Puede usarse para generar los *stubs*/*skeleton* y las capas intermedias necesarias para escribir { clientes que invoquen los Servicios Web / servidores que los implementen
     - Especificar información sobre la localización del servicio (URIs)
4. Publicación y localización de servicios ⇒ uso de UDDI (*Universal Description, Discovery and Integration*)
   - La descripción de los servicios (documentos WSDL) se almacena en un directorio de servicios
   - UDDI especifica cómo { se publican y descubren los servicios / trabajan los directorios de servicios Web
   - Acceso al directorio UDDI mediante Servicios Web ⇒ uso mensajes SOAP
     - servidor da de alta de servicios (documentos WSDL + descripción)
     - cliente "descubre" servicios (documentos WSDL)

Algunas implementaciones

- *Java API for XML Web Services*: JAX-WS + JAX-RPC
  - Forman parte de la especificación Java EE 5 (*Java Enterprise Edition*)
    - Implementaciones incluidas en los servidores de aplicaciones Java EE
  - Implementación de referencia METRO (incluida en Java SE 6 [jdk 1.6])
- Apache AXIS2 (Java y C) [`http://ws.apache.org/axis2`]
  - La versión anterior (AXIS) incluye soporte para C++
  - `http://ws.apache.org/axis/cpp/index.html`
- Perl: SOAP::Lite [`http://www.soaplite.com`]

4.2 Protocolo SOAP: Intercambio de mensajes.

SOAP: *Simple Object Access Protocol*

- **Objetivo**: Especificar como organizar la información de forma estructurada y tipada usado XML para que sea intercambiada entre los extremos de la invocación
- Especificado por el W3C (versión actual 1.2)
  - `http://www.w3.org/2000/xp/Group/`
  - `http://www.w3c.es/Traducciones/es/TR/2003/REC-soap12-part0-20030624/`
- Derivado de protocolo XML-RPC
  - XMLR-RPC: formato XML para enviar invocaciones de métodos
    - nombre método + codificación de la lista de parámetros
    - limitado en cuanto a tipos de datos soportados

Protocolo SOAP especifica:

- Formato de mensajes común y extensible: describe cómo se organiza en forma de documentos XML la información a intercambiar
- Conjunto de normas para implementar RPC mediante mensajes SOAP
- Reglas a seguir por la entidades (cliente o servidor) que procesen los mensajes SOAP
  - indica elementos a tratar u omitir, quien debe hacerlo y los tipos de acciones a realizar
- Descripción del modo en que se envían los mensajes SOAP sobre el protocolo de transporte (HTTP o SMTP)

Características

- SOAP define un intercambio de mensajes sin estado
  - Posibilidad de soportar comunicaciones con estado añadiendo información adicional (IDs únicos) en la cabecera de los mensajes SOAP ($\approx$ *cookies*)
- Define una comunicación en una sóla dirección
  - Interacciones más complejas son gestionadas por los extremos
  - Esquemas síncronos (RPC): mensaje petición + mensaje respuesta
  - Esquemas asíncronos: sólo mensaje petición

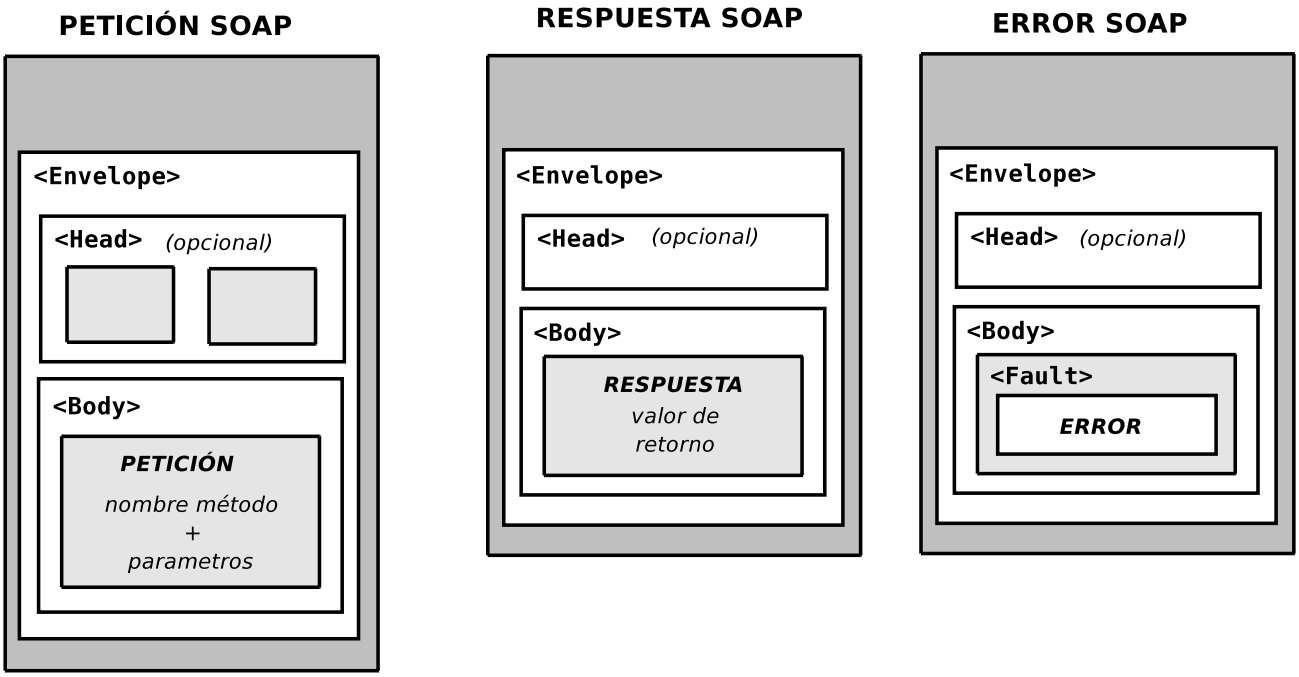
- SOAP no impone restricciones sobre la semántica de los mensajes intercambiados
  - SOAP sólo ofrece la infraestructura para transferir esos mensajes
  - El significado de los mensajes (etiquetas XML) es interpretado por los extremos
    - SOAP es independiente del modelo de datos de la aplicación
- SOAP no se encarga de cuestiones de fiabilidad, integridad de los mensajes, transacciones, seguridad, etc
  - La gestión de esos aspectos es responsabilidad de la infraestructura/aplicación que implementa el Servicio Web

Nota: La implementación de los Servicios Web es responsabilidad del extremo que lo ofrece y debe encargarse de interpretar el contenido de los mensajes SOAP

- El extremo contará con un servidor Web que recibirá la petición SOAP y la redirigirá a la implementación del servicio Web
- La implementación real del Servicio Web puede residir en un CGI (*Common Gateway Ieterface*), un Servlet, un objeto CORBA, un Componente EJB (*enterprise Java Bean*), un Procedimiento Almacenado de un SGBD, etc, ...

4.2.1 Estructura de los mensajes SOAP

- SOAP ofrece soporte para el envío de datos de aplicación arbitrarios
  - Define la estructura de un "contenedor" de mensajes XML
  - No establece restricciones sobre el contenido del mensaje ni sobre el procesamiento a realizar con él
    - XML-RPC usaba etiquetas predefinidas (diferencia con SOAP)



(a) Mensaje SOAP: *SOAP envelope* (2 partes)

- **Cabecera** (etiqueta `<Header>`): componente opcional
  - Contiene información sobre el mensaje a usar por la infraestructura de Servicios Web
    - identificadores de transacciones
    - información de seguridad (claves, certificados, etc)
    - otros: prioridades, identificadores de origen y/o destino, etc
  - Atributos *mustUnderstand actor* indican que componente debe encargarse de interpretar esa cabecera (extremos de la comunicación, puntos intermedios)
  - Puede estructurarse en bloques
- **Cuerpo** (etiqueta `<Body>`): componente obligatorio
  - Contiene información específica a usar por las aplicaciones que usan o implementan el Servicio Web
    - los extremos son los responsables de acordar el formato de la información intercambiada y de generar y/o procesar su contenido

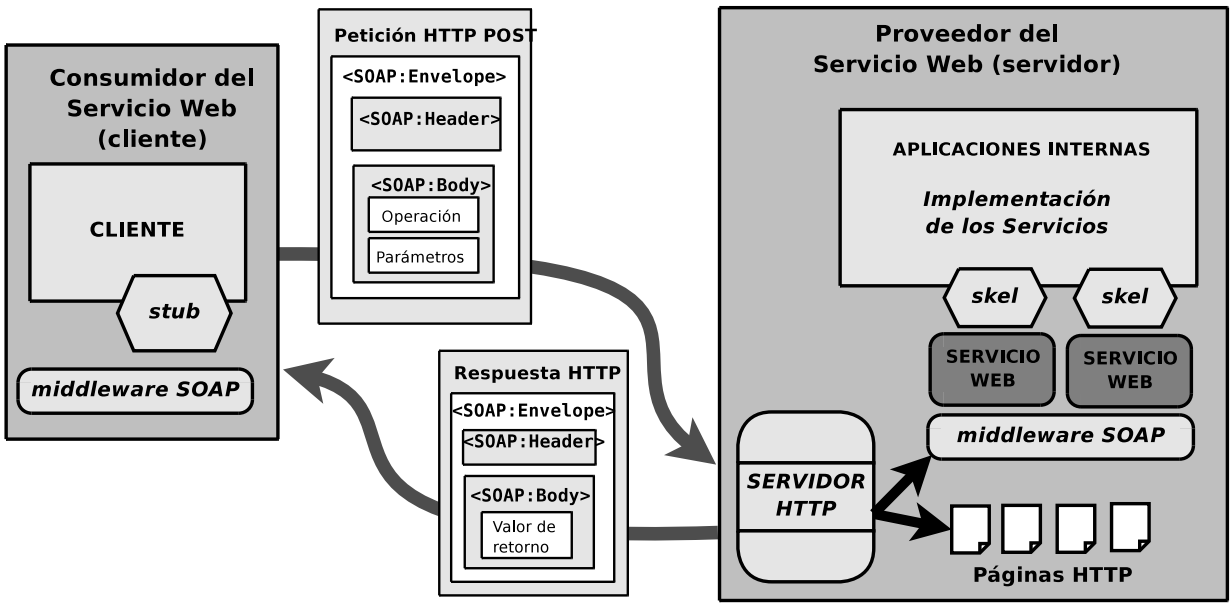
- Puede estructurarse en bloques
  - Usualmente los bloques mapean una invocación o respuesta RPC en formatro XML
    - Esquema usual: codificación de parámetros y valor de retorno
    - Suele hacerse uso de los tipos de datos definidos en la especificación *XML Schema*
    - *XML Schema* permite representar en XML { tipos simples (enteros, reales,...) / tipo compuestos (estructuras, arrays,...) }
  - Bloque especial **fault**: usado para representar errores en el procesamiento de mensajes SOAP
    - identificación del error (código de fallo)
    - explicación legible del fallo
    - origen del fallo

- SOAP no establece ningua restricción en el contenido y estructura de los bloques incluidos en ambas secciones
  - Aunque sí existen recomendaciones y prácticas comunmente aceptadas

(b) *Bindings* SOAP

- La especificación SOAP es independiente del protocolo de transporte usado para transferir los mensajes
  - SOAP sólo define un contenedor de "mensajes" y la forma de encapsularlos en el protocolo de transporte que se use
- ***binging* SOAP**: descripción de cómo se envía un mensaje SOAP utilizando un protocolo de tansporte determinado
  - Incluye la forma de especificar las direcciones de origen y destino
- El *binding* típico de SOAP hace uso de mensajes HTTP para encapsular mensajes SOAP
  - Peticiones HTTP POST: { se envía el mensaje SOAP en el cuerpo de la petición HTTP / la respuesta HTTP contiene un mensaje SOAP (respuesta o *Fault*) }
- El estándar también define *bindings* de SOAP sobre mensajes del protocolo SMTP (e-mail)
- La dirección de destino toma la forma de una URL en *binding* sobre HTTP (o el campo `to:` en *bindings* SMTP)

Ejemplo mensajes SOAP



Petición SOAP sobre un mensaje POST de HTTP

```
POST /WeatherForecast.asmx HTTP/1.1
Host: www.webservicex.net
Content-Type: text/xml; charset=utf-8
Content-Length: 446
SOAPAction: "http://www.webservicex.net/GetWeatherByPlaceName"

<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
               xmlns:xsd="http://www.w3.org/2001/XMLSchema"
               xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Head>
      <userID>010243</userID>
      <transactionID>02394800231</transactionID>
  </soap:Head>
  <soap:Body>
    <GetWeatherByPlaceName xmlns="http://www.webservicex.net">
      <PlaceName>Las Vegas</PlaceName>
    </GetWeatherByPlaceName>
  </soap:Body>
</soap:Envelope>
```

Respuesta SOAP sobre una respuesta HTTP

```
HTTP/1.1 200 OK
Content-Type: text/xml; charset=utf-8
Content-Length: 1637

<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
               xmlns:xsd="http://www.w3.org/2001/XMLSchema"
               xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Body>
    <GetWeatherByPlaceNameResponse xmlns="http://www.webservicex.net">
      <GetWeatherByPlaceNameResult>
        <Latitude>36.17208</Latitude>
        <Longitude>115.122368</Longitude>
        <AllocationFactor>0.033507</AllocationFactor>
        <FipsCode>32</FipsCode>
        <PlaceName>LAS VEGAS</PlaceName>
        <StateCode>NV</StateCode>
        <Details>
          <WeatherData>
            <Day>Sunday, December 7, 2008</Day>
            <WeatherImage> ... </WeatherImage>
            ...
            <MaxTemperatureC>9</MaxTemperatureC>
            <MinTemperatureC>1</MinTemperatureC>
          </WeatherData>
          ...
          <WeatherData>
            <Day>Thursday, December 11, 2008</Day>
            <WeatherImage> ... </WeatherImage>
            ...
            <MaxTemperatureC>17</MaxTemperatureC>
            <MinTemperatureC>5</MinTemperatureC>
          </WeatherData>
        </Details>
      </GetWeatherByPlaceNameResult>
    </GetWeatherByPlaceNameResponse>
  </soap:Body>
</soap:Envelope>
```

4.3 Protocolo WSDL: Descripción de servicios.

WSDL: *Web Services Description Language*

- *Middleware* convencional: lenguajes de definición de interfaces (IDL)
- Servicios Web necesitan una descripción de interfaces más rica e independiente de la plataforma
  1. Definición de operaciones: nombre, argumentos (nombre + tipo), valor retorno
  2. Definición de mecanismos de interacción (*bindings*)
     - Sistemas distribuidos convencionales usan siempre el mismo middleware
     - En Servicios Web cada servicio se puede servir con distintos protocolos (HTTP, SMTP)
  3. Especificación de la localización del servicio (URI del servicio)
     - Localización a donde enviar los mensajes SOAP
- Descripción del Servicio Web está basada en documentos XML conforme a la especificación WSDL
  - `http://www.w3.org/2002/ws/desc/`

4.3.1 Usos de WDSL

1. Como lenguaje de descripción de interfaz del servicio (IDL)
   - Describe el interfaz que implementa el Servicio Web (contrato entre cliente y servidor)
   - Indica cómo interactuar con el servicio { operaciones definidas; datos a enviar y devolver; formato mensajes + protocolo transferencia }
2. Como entrada para compiladores/generadores de *stubs* y/o *skeleton*

   Las aplicaciones no escriben ni procesan directamente los mensajes XML SOAP
   - A partir de la definición WSDL del Servicio Web se generan los elementos adicionales necesarios para que { servidor implemente el Servicio Web; cliente realice la llamada al Servicio Web }
     - *stubs* y/ *skeletons*
     - Objetos y/o procedimientos para serializar los datos en el formato de los mensajes SOAP usados por cada Servicio Web concreto
   - Ejemplos { `wsimport` en Java JAX-WS; `WSDL2java` en Apache Axis }

   También es habitual contar con **generadores WSDL** que crean la especificación WSDL a partir de las implementaciones de los Servicios Web
   - `wsgen` de Java: crea fichero WSDL a partir del código Java de una clase

4.3.2 Elementos de una especificación WSDL

- Documento WSDL describe un servicio Web como una colección de *ports* (puertos) [extremos de la comunicación (*end points*)] capaces de intercambiar mensajes
  - Cada *port* tiene { una definición abstracta (*port type*) [operaciones y mensajes] / una definición concreta (*binding*) [protocolos]
- **Elemento** `<definitions>`**:** raíz del documento WSDL
  - Usado para declarar espacios de nombres (*XML namespace*)
- Parte *abstracta*
  - Equivalente al lenguaje IDL de sistemas distribuidos convencionales
  - Describe de forma abstracta los servicios Web en términos de operaciones y mensajes

  elemento `<types>`**:** Define los *tipos* y *estructuras de datos* de los datos intercambiados

  elemento `<message>`**:** Define los *mensajes* (datos que van a ser intercambiados), indicando su nombre y su contenido
    - Un mensaje para un servicio concreto contendrá un conjunto de *partes* [elemento `<part>`]
    - Cada *parte* está caracterizadas por un nombre y un tipo (definido en la sección *types*)

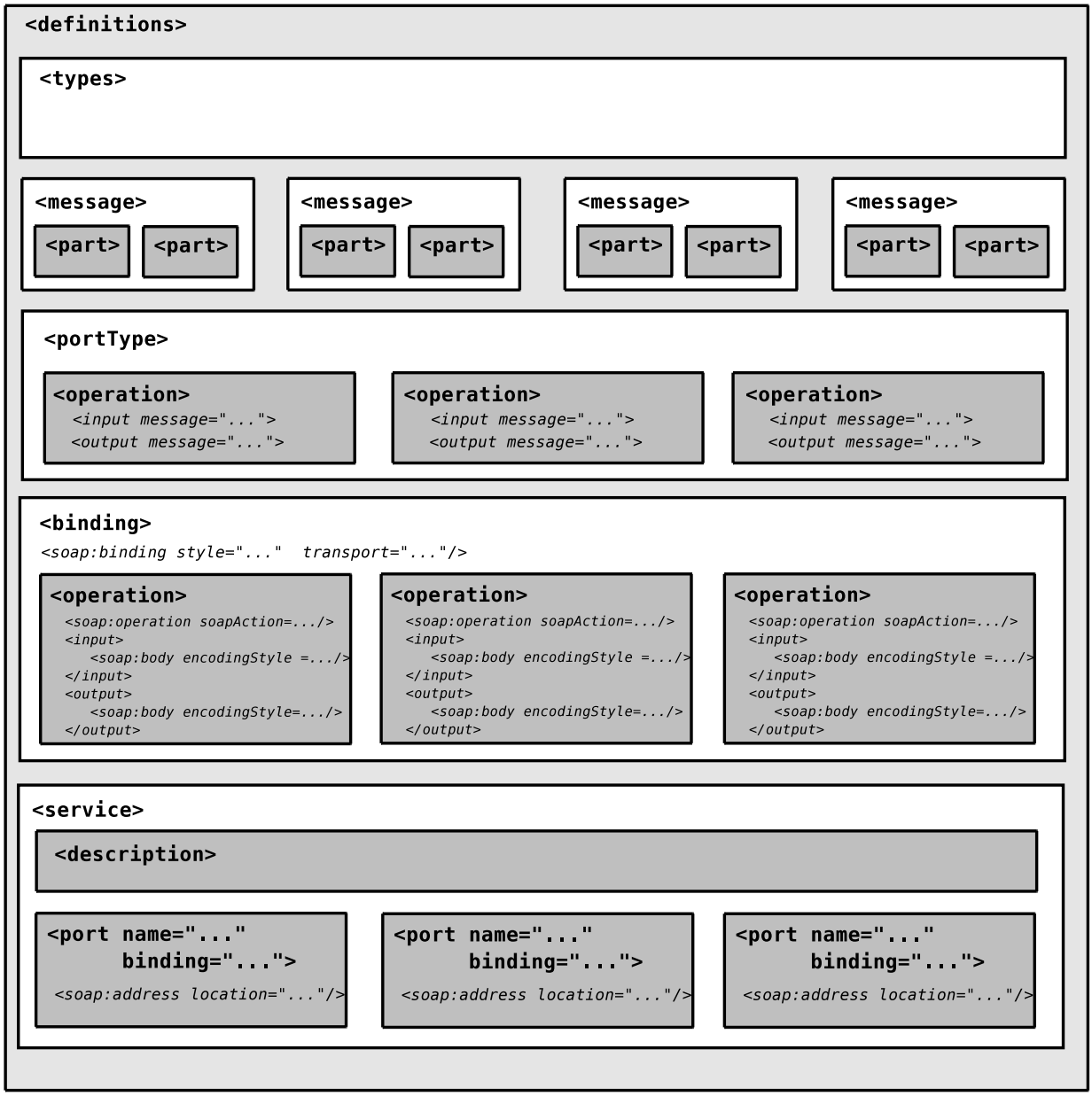
  elemento `<portType>`**:** Define grupos de operaciones (*ports*, puertos)
  Para cada operación (elemento `<operation>`) se le asigna un nombre y se especifica el intercambio de mensajes [4 tipos]
    - *one-way*: cliente invoca servicio enviando un único mensaje (asíncrono)
    - *notifications*: servidor envían un mensaje (asíncrono)
    - *request-response*: servidor recibe petición y responde (síncrono)
    - *solicit-response*: servidor invoca y espera respuesta (síncrono)
- Parte *concreta*

  Se encarga de definir (concretar) la instancia real del servicio
  - Especifica aspectos relativos al uso del servicio Web { puertos que implementa / protocolos de transporte usados / dirección donde se ubica / codificación de mensajes usada

  elemento `<bindings>`**:** Asocia a un grupo de operaciones (*portType*) una especificación de la codificación de mensajes y el protocolo de transporte (atributo *transport*) a utilizar.
    - Informa a los usuarios del Servicio Web (clientes o servidores) de los protocolos a usar, de como estructurar los mensajes XML y de lo que se espera recibir al enviar un mensaje
    - WSDL permite *bindings* para SOAP, HTTP GET, HTTP POST y MIME (SMTP)
    - En el caso de usar SOAP como mecanismo de intercambio de mensajes el *binding* contiene toda la información necesaria para construir y procesar automáticamente los mensajes SOAP (atributo *encodingStyle*)

elemento `<service>`**:** Especifica una agrupación lógica de puertos (*port*)

- Opcionalmente puede incluir una descripción textual del servicio (elemento `<description>`)
- Cada puerto (elemento `<port>`) especifica un punto final (*endpoint*) [destino final de la comunicación]
  - Asocia la información de los *bindings* (conjuntos de operaciones) a la dirección (URI) donde se accederá a sus implementaciones
  - *port* $\approx$ *portType* + *binding* + localización



Ejemplo documento WSDL

```
<wsdl:definitions targetNamespace="http://www.webservicex.net">

<wsdl:types>
   <s:schema targetNamespace="http://www.webservicex.net">
      <s:complexType name="WeatherForecasts">
        <s:sequence>
           <s:element minOccurs="1" maxOccurs="1" name="Latitude" type="s:float"/>
           <s:element minOccurs="1" maxOccurs="1" name="Longitude" type="s:float"/>
           ...
           <s:element minOccurs="0" maxOccurs="1" name="Details"
                                                  type="tns:ArrayOfWeatherData"/>
        </s:sequence>
      </s:complexType>
      <s:complexType name="ArrayOfWeatherData">
        <s:sequence>
           <s:element minOccurs="0" maxOccurs="unbounded" name="WeatherData"
                                                          type="tns:WeatherData"/>
        </s:sequence>
      </s:complexType>
      <s:complexType name="WeatherData">
        <s:sequence>
           <s:element minOccurs="0" maxOccurs="1" name="Day" type="s:string"/>
           ...
           <s:element minOccurs="0" maxOccurs="1" name="MaxTempC" type="s:string"/>
           <s:element minOccurs="0" maxOccurs="1" name="MinTempC" type="s:string"/>
        </s:sequence>
      </s:complexType>
      ...
      <s:element name="GetWeatherByPlaceName">
        <s:complexType>
           <s:sequence>
             <s:element minOccurs="0" maxOccurs="1" name="PlaceName" type="s:string"/>
           </s:sequence>
        </s:complexType>
      </s:element>
      <s:element name="GetWeatherByPlaceNameResponse">
        <s:complexType>
           <s:sequence>
              <s:element minOccurs="1" maxOccurs="1" name="GetWeatherByPlaceNameResult"
                                                     type="tns:WeatherForecasts"/>
           </s:sequence>
        </s:complexType>
      </s:element>
   </s:schema>
</wsdl:types>
```

```
<wsdl:message name="GetWeatherByPlaceNameSoapIn">
   <wsdl:part name="parameters" element="tns:GetWeatherByPlaceName"/>
</wsdl:message>

<wsdl:message name="GetWeatherByPlaceNameSoapOut">
   <wsdl:part name="parameters" element="tns:GetWeatherByPlaceNameResponse"/>
</wsdl:message>


<wsdl:portType name="WeatherForecastSoap">
   <wsdl:operation name="GetWeatherByPlaceName">
      <wsdl:input message="tns:GetWeatherByPlaceNameSoapIn"/>
      <wsdl:output message="tns:GetWeatherByPlaceNameSoapOut"/>
   </wsdl:operation>
</wsdl:portType>


<wsdl:binding name="WeatherForecastSoap" type="tns:WeatherForecastSoap">
   <soap:binding transport="http://schemas.xmlsoap.org/soap/http" style="document"/>

   <wsdl:operation name="GetWeatherByPlaceName">
      <soap:operation soapAction="http://www.webservicex.net/GetWeatherByPlaceName"
                      style="document"/>
      <wsdl:input>
         <soap:body use="literal"/>
      </wsdl:input>
      <wsdl:output>
         <soap:body use="literal"/>
      </wsdl:output>
   </wsdl:operation>
</wsdl:binding>


<wsdl:service name="WeatherForecast">
   <documentation>
      Get one week weather forecast for valid zip code or Place name in USA
   </documentation>

   <wsdl:port name="WeatherForecastSoap" binding="tns:WeatherForecastSoap">
      <soap:address location="http://www.webservicex.net/WeatherForecast.asmx"/>
   </wsdl:port>
</wsdl:service>

</wsdl:definitions>
```

4.4 Protocolo UDDI: Publicación de servicios.

UDDI: (*Universal Description, Discovery and Integration*)

- Protocolo para interaccionar con un servidor (registro UDDI) que proporciona operaciones (vía SOAP) para registrar y buscar (descubrir) Servicios Web
- Cada servicio se registra dando su nombre, una descripción del servicio (URL de su WSDL, una descripción textual, etc.)
- Uso de UDDI mediante APIs de programación basadas en SOAP
  - El API de UDDI está especificada con WSDL $\Rightarrow$ uso de mensajes SOAP
  - Alta (publicación) de Servicios Web por parte de los servidores
  - Localización (descubrimiento) de Servicios Web por parte de los cliente
- Finalidad:
  - Ofrecer soporte para encontrar información sobre servicios web y poder construir clientes
  - Facilitar el enlace dinámico, permitiendo consultar referencias y acceder a servicios de interés en tiempo de ejecución (descubir servicios e invocarlos "al vuelo")
- Especificación: `http://uddi.xml.org`

Tipos de información ofrecida

Páginas blancas : Identificador y dirección de contacto de la empresa/organización que publica el Servicio Web

Páginas amarillas : Descripciones de los Servicios Web ofrecidos usando diferentes tipos de categorizaciones (taxonomías)

- *NAICS-North American Industry Classification System*, *UNSPSC-Universal Standard Products and Services Classification*, etc

Páginas verdes : Info. técnica sobre los servicios web (URL de descarga del WSDL)