**Colección de Problemas Aplicaciones Distribuidas**

**Temas 1, 2 y 3**

# Ejercicios Tema 1

**Problema 1.** Indica si las siguientes afirmaciones son ciertas o falsas.

**1. Una aplicación de tipo cliente / servidor no puede ejecutar los procesos cliente y servidor en la misma máquina.**

☐ Cierto ☐ Falso

Justificación: Falso. Sí que se pueden ejecutar en la misma máquina. En ese caso se puede utilizar el interfaz localhost.

**2. application/x-www-form-urlencoded es un tipo de contenido válido.**

☐ Cierto ☐ Falso

Justificación: Cierto. Es el tipo de contenido que se utiliza para enviar el contenido de los formularios html.

**3. MIME sirve para codificar páginas html en binario.**

☐ Cierto ☐ Falso

Justificación: Falso. MIME permite enviar mensajes de correo multiparte. También se puede utilizar en otros protocolos de nivel de aplicación.

**4. La versión 0.9 de HTTP fue la primera en enviar mensajes estructurados en cabecera y contenido.**

☐ Cierto ☐ Falso

Justificación: Falso. La primera fue la 1.0.

**5. La cabecera Server pertenece al conjunto de cabeceras de petición de los mensajes HTTP.**

☐ Cierto ☐ Falso

Justificación: Falso. Pertenece al conjunto de cabeceras de respuesta.

**6. HTML5 funciona sobre dispositivos móviles.**

☐ Cierto ☐ Falso

Justificación: Cierto. Funciona tanto en Android como en dispositivos Apple.

# Ejercicios Tema 2

**Problema 1.** Indica si las siguientes afirmaciones son ciertas o falsas.

**1. Los servidores basados en J2EE siguen una estructura de directorios predefinida.**

☐ Cierto ☐ Falso

Justificación: Cierto. Fue definida por Sun y ahora mantenida por Oracle.

**2. La única manera de configurar una aplicación web en un servidor Tomcat es con el fichero web.xml.**

☐ Cierto ☐ Falso

Justificación: Falso. También se puede configurar con java annotations @.

**3. Los servlets y las jsp no tienen nada en común.**

☐ Cierto ☐ Falso

Justificación: Falso. Las jsp se tienen que convertir en servlets para poderse ejecutar.

**4. SOAP puede utilizar texto plano para enviar peticiones y respuestas.**

☐ Cierto ☐ Falso

Justificación: Falso. SOAP sólo puede utilizar XML.

**5. REST sólo puede utilizar JSON para conectarse con un servicio remoto.**

☐ Cierto ☐ Falso

Justificación: Falso. REST puede utilizar cualquier formato, JSON es sólo un formato posible.

**6. En SOAP no se puede implementar un nuevo servicio sin utilizar un registro UDDI.**

☐ Cierto ☐ Falso

Justificación: Falso. Sólo se necesita el WSDL.

**7. No se puede modificar un fichero WSDL manualmente para cambiar alguna de las características el servicio a utilizar.**

☐ Cierto ☐ Falso

Justificación: Falso. Es un fichero XML que se puede modificar con un editor de texto o de XML.

**8. Los servicios web pueden atravesar firewalls sin problema.**

☐ Cierto ☐ Falso

Justificación: Cierto. Funcionan sobre el puerto 80, que está permitido en la mayoría de firewalls para la navegación web.

**9. REST utiliza los métodos HTTP para realizar operaciones de modificación, consulta y creación**

☐ Cierto ☐ Falso

Justificación: Cierto. Utiliza los métodos POST, GET, UPDATE, DELETE.

**10. No se puede establecer una conexión a un servicio web utilizando una jsp.**

☐ Cierto ☐ Falso

Justificación: Falso. Se puede hacer igual que con un servlet aunque se programaría de forma diferente.

**11. Con SOAP es posible conectarse con un servicio web público existente.**

☐ Cierto ☐ Falso

Justificación: Cierto.

**12. Si implementamos un servicio web REST con Eclipse podemos elegir el tipo de respuesta que retornará el servicio utilizando las cabeceras HTTP del mensaje de petición.**

☐ Cierto ☐ Falso

Justificación: Cierto. Se indica en la cabecera Accept.

## Problema 2.

Se quiere diseñar una aplicación distribuida para gestionar una web de venta de artículos electrónicos. Dicha aplicación está basada en aplicaciones web J2EE, servicios web SOAP y una aplicación para dispositivos móviles.

La aplicación tiene las siguientes características:

Para poder realizar compras, los usuarios deben introducir un nombre de usuario y una contraseña. El sistema almacena también otros datos de los usuarios para poder realizar las entregas de los artículos comprados como son la dirección postal donde se realiza la entrega, dirección que debe aparecer en la factura, etc.

Una vez dentro del sistema, los usuarios disponen de las siguientes funcionalidades:

a) Compra de nuevo(s) producto(s). Cada producto tiene asociada la siguiente información: Identificador, Nombre, Precio unitario, Fabricante.

b) Devolución de producto(s) comprado(s). Los productos que se van a devolver tienen los mismos campos que en el apartado a).

c) Seguimiento de pedido(s). Cada pedido tiene los datos de los productos comprados (descritos en el apartado a), el importe total del pedido, la fecha en la que se realizó el pedido, el identificador de usuario y el identificador de pedido.

d) Actualización de datos de usuario. Username, password, NIF, nombre y apellidos, dirección postal de entrega, dirección postal de facturación, dirección de correo electrónico, teléfono de contacto. El username no se puede modificar.

Además de la versión web, se dispone de una aplicación para dispositivos móviles que permite realizar las mismas operaciones.

**Contestar razonada y brevemente a las siguientes preguntas:**

1) Indicar el número mínimo de formularios necesarios. Para cada formulario, describir brevemente los campos que contendría.

   Formularios requeridos:

   Login: Username, password, botón Submit

   Compra de productos: Id producto, cantidad, importe para cada producto que el usuario quiere comprar. Botón Submit

   Devolución de productos: Id producto, importe devolución para cada producto que el usuario quiere devolver. Botón Submit.

   Seguimiento de pedidos: Id pedido. Botón Submit.

   Actualización de datos de usuario: Todos los datos del usuario que se puedan modificar. Botón Submit.

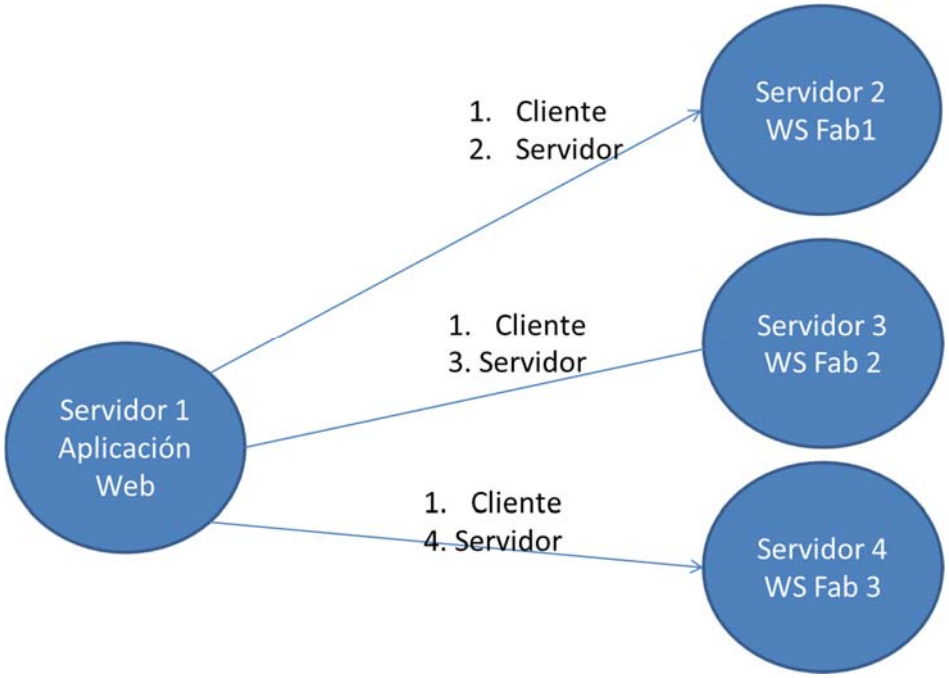
2) ¿Cuántos servlets serían necesarios en el sistema? Indicar claramente a qué formulario accedería cada uno de ellos.

   Uno por formulario, es decir, 5 servlets.

3) Para mantener la información de los productos a la venta lo más actualizada posible, la aplicación web contacta con un servicio web que muestra la información “oficial” del producto ofrecida por el fabricante. ¿Cuántos ficheros WSDL serían necesarios para que nuestra aplicación fuera cliente de todos ellos? Dibujar un diagrama de servidores, suponiendo que trabajamos con 3 fabricantes distintos, indicando qué aplicación o servicio hay en cada uno de ellos.

   Sería necesario 1 fichero WSDL por cada fabricante, en nuestro caso, 3.

   Diagrama servidores:




4) Definir las tablas que serían necesarias para almacenar la información de un pedido en una base de datos relacional. Indicar claramente el nombre de la tabla, los campos que contendría y el tipo de cada campo.

Tabla pedido (Id_pedido integer, Fecha date, Id_usuario integer, Importe_total float)

Tabla producto_pedido (Id_pedido integer, Id_producto integer, Cantidad integer)

Tabla producto (Id_producto integer, Nombre varchar, Precio float, Fabricante varchar)

5) ¿Es necesario realizar alguna modificación en los servlets para que se pueda conectar la aplicación para dispositivos móviles? Justificar la respuesta.

Al ofrecer la misma funcionalidad no sería necesario realizar ninguna modificación. En el diseño de pantallas de la aplicación móvil se debería optimizar el uso de la pantalla en los distintos formularios de la aplicación.

**Problema 3.**

En el Anexo I tenemos un fragmento del WSDL de flickr:

**Contestar razonada y brevemente a las siguientes preguntas:**

1) Especificar una petición al servicio activity.userPhotos. Indicar claramente los campos que contendría y el tipo de los mismos.

La petición es una extensión del tipo authenticatedFlickrRequest y contiene los siguientes campos:

```
<xs:complexType>
       <xs:complexContent>
              <xs:extension base="authenticatedFlickrRequest">
                <xs:sequence>
                 <xs:element name="timeframe" type="xs:string" minOccurs="0"/>
                 <xs:element name="per_page" type="xs:integer" minOccurs="0"/>
                 <xs:element name="page" type="xs:integer" minOccurs="0"/>
                </xs:sequence>
              </xs:extension>
       </xs:complexContent>
</xs:complexType>
```

2) Especificar el elemento `Binding` para la operación anterior sobre HTTP. Suponer que se envía la información con el comando POST de HTTP.

```
<wsdl:binding name="flickrHTTPBinding" type="tns:flickrHTTP" >
```

```
    <http:binding verb="POST" />
      <wsdl:operation ref="tns:flickr.activity.userPhotos"/>
            <wsdl:input>
                  <mime:content part="parameters" type="application/xml" />
            </wsdl:input>
            <wsdl:output>
                  <mime:content part="parameters" type="application/xml" />
            </wsdl:output>
      </wsdl:operation>
</wsdl:binding>
```

3) ¿Qué estructura tiene la respuesta que nos devuelven las operaciones activity.userPhotos y photos.Search? Indicar claramente qué información contiene la respuesta.

```
<xs:complexType>
    <xs:choice>
            <xs:element name="err" form="unqualified">
                  <xs:complexType>
                  <xs:complexContent>
                  <xs:extension base="xs:anyType">
                    <xs:attribute name="code" type="xs:string"/>
                    <xs:attribute name="msg" type="xs:string"/>
                  </xs:extension>
                  </xs:complexContent>
                  </xs:complexType>
            </xs:element>
            <xs:any maxOccurs="unbounded"/>
    </xs:choice>
</xs:complexType>
```

Se trata de una estructura de tipo choice que distingue entre una respuesta correcta y una errónea. En caso de error devuelve un código y un mensaje. En caso correcto, puede devolver cualquier cosa (any).

4) ¿Qué elemento nos falta en el WSDL para indicar dónde podemos encontrar el servicio?

El elemento wsdl:service que define la dirección dónde nos tenemos que conectar.

**5)** En la petición de la operación photos.Search, ¿cuántos atributos de tipo string hay? ¿Y de tipo entero? ¿Hay algún campo que sea obligatorio? Listar todos los tipos de elementos que no sean ni string ni entero.

9 campos de tipo string (incluyendo api_key). 4 de tipo integer.

No hay ningún campo obligatorio, todos tienen minOccurs=0.

unixTimeStamp, sortOrder, anyOrAll,

**Problema 4.**

En el Anexo II tenemos un fragmento de un WSDL para buscar vídeos en función de su año de producción. En el Anexo III tenemos un elemento `Envelope` de SOAP para lanzar dicha operación:

**Contestar razonada y brevemente a las siguientes preguntas:**

1) ¿Cómo sería una respuesta posible a esta pregunta? Indicar el número de elementos que podríamos tener y su estructura en campos.

Podemos tener entre 0 y n vídeos, cada uno de ellos con la siguiente información:

int catId, vidReproducciones;

String vidAutor, vidDescripcion, vidDuracion, vidFecha, vidFormato, vidId, vidImagen, vidTitulo;

2) Explicar cómo podemos saber que es posible hacer un `Binding` directamente sobre HTTP y cómo sería. ¿Dónde está la información que nos lo dice?

```
<wsdl:binding name="FindServiceHttpBinding" type="ns:FindServicePortType">
  <http:binding verb="POST" />
  <wsdl:operation name="buscarVideosByYear">
    <http:operation location="buscarVideosByYear" />
    <wsdl:input>
      <mime:content part="parameters" type="application/xml" />
    </wsdl:input>
```

```
      <wsdl:output>
        <mime:content part="parameters" type="application/xml" />
      </wsdl:output>
    </wsdl:operation>
  </wsdl:binding>
```

3) ¿Cuántos campos y de qué tipo tiene la estructura de los vídeos que nos devuelve la operación?

```
<xs:complexType name="Video">
  <xs:sequence>
      <xs:element minOccurs="0" name="catId" nillable="true" type="xs:int" />
      <xs:element minOccurs="0" name="vidAutor" nillable="true" type="xs:string" />
      <xs:element minOccurs="0" name="vidDescripcion" nillable="true" type="xs:string"/>
      <xs:element minOccurs="0" name="vidDuracion" nillable="true" type="xs:string" />
      <xs:element minOccurs="0" name="vidFecha" nillable="true" type="xs:string" />
      <xs:element minOccurs="0" name="vidFormato" nillable="true" type="xs:string" />
      <xs:element minOccurs="0" name="vidId" nillable="true" type="xs:string" />
      <xs:element minOccurs="0" name="vidImagen" nillable="true" type="xs:string" />
      <xs:element minOccurs="0" name="vidReproducciones" nillable="true" type="xs:int"/>
      <xs:element minOccurs="0" name="vidTitulo" nillable="true" type="xs:string" />
    </xs:sequence>
</xs:complexType>
```

4) ¿Qué nos indican los atributos de la etiqueta inicial del elemento `wsdl:definitions`?

Los namespaces que vamos a utilizar.

5) Justificar (en función del WSDL) porqué el elemento `Body` del `Envelope` de SOAP es correcto.

```
      <xs:element name="searchVideosByYear">
        <xs:complexType>
          <xs:sequence>
            <xs:element minOccurs="0" name="year" nillable="true" type="xs:string" />
          </xs:sequence>
        </xs:complexType>
      </xs:element>
```

La operación searchVideosByYear sólo tiene un parámetro de tipo String que es year.

6) ¿Qué se define en el elemento `wsdl:portType`?

El elemento <portType> define un servicio web, las operaciones que se pueden realizar y los mensajes que están involucrados.

7) Si quisiéramos añadir una nueva operación “Buscar vídeos por título”, ¿qué deberíamos añadir al WSDL?

Una nueva operación, definiendo la petición y la respuesta correspondientes en <wsdl:types>.

Los mensajes de petición y respuesta con <wsdl:message>.

El servicio y las operaciones con <wsdl:portType>.

Los bindings a los distintos tipos de transporte con <wsdl:binding>.

# Ejercicios Tema 3

**Problema 1.** Indica si las siguientes afirmaciones son ciertas o falsas.

**1. Un vídeo de youtube es un tipo de contenido monomedia.**

☐ Cierto ☐ Falso

Justificación: Falso. Los vídeos de youtube incluyen al menos imagen/vídeo y audio.

**2. PCM es un mecanismo que permite codificar audio.**

☐ Cierto ☐ Falso

Justificación: Cierto.

**3. MP3 es un formato de audio que se basa en PCM para hacer la codificación de audio.**

☐ Cierto ☐ Falso

Justificación: Falso. Utiliza un mecanismo que comprime más el audio.

**4. Con un gráfico vectorial es posible dibujar objetos geométricos.**

☐ Cierto ☐ Falso

Justificación: Cierto. Este tipo de gráficos se basa en formas geométricas.

**5. El formato de fichero JPEG permite indicar las dimensiones de la imagen que contiene.**

☐ Cierto ☐ Falso

Justificación: Cierto. Son dos de los campos que contiene.

**6. Los vídeos que se representan como secuencias de imágenes completas ocupan menos espacio que los vídeos que utilizan distintos tipos de imágenes.**

☐ Cierto ☐ Falso

Justificación: Falso. Los vídeos que utilizan codificación temporal o espacial pueden ocupar mucho menos al no enviar todas las imágenes.

**7. Sólo hay contenedores multimedia de tipo específico.**

☐ Cierto ☐ Falso

Justificación: Falso. Hay contenedores específicos y genéricos.

**8. El ISO Base Media File Format permite guardar contenido definido únicamente por MPEG.**

☐ Cierto ☐ Falso

Justificación: Falso. Es un formato genérico que permite almacenar contenido de distinto tipo.

**9. MPEG-DASH y MPEG-21 son formatos de contenedores multimedia definidos por MPEG.**

☐ Cierto ☐ Falso

Justificación: Falso. DASH es un protocolo para enviar contenido utilizando streaming adaptativo.

**10. Dentro de un MPEG-21 DI se pueden expresar metadatos EBUCore.**

☐ Cierto ☐ Falso

Justificación: Cierto. MPEG-21 DI es un contenedor genérico que permite incluir cualquier tipo de información, aunque no esté basada en MPEG.

**11. Con Dublin Core sólo se pueden definir metadatos sobre la localización de un contenido.**

☐ Cierto ☐ Falso

Justificación: Falso. Dublin Core permite definir metadatos genéricos sobre un contenido como pueden ser el título, la fuente, los derechos, etc.

**12. JPSearch es un esquema de metadatos orientado a educación, igual que IEEE LOM.**

☐ Cierto ☐ Falso

Justificación: Falso. JPSearch está orientado a imágenes.

**13. Existen esquemas de metadatos específicos para programas de televisión.**

☐ Cierto ☐ Falso

Justificación: Cierto. EbuCore es uno de ellos.

**ANEXO I**. Extracto WSDL Operaciones en flickr

```
<wsdl:types>
       <xs:complexType name="flickrRequest">
         <xs:sequence>
           <xs:element name="api_key" type="xs:string"/>
                 <xs:any minOccurs="0" maxOccurs="unbounded"/>
           </xs:sequence>
       </xs:complexType>
       <xs:complexType name="signedFlickrRequest">
             <xs:complexContent>
                   <xs:extension base="flickrRequest">
                     <xs:sequence>
                          <xs:element name="api_sig" type="xs:string"/>
                     </xs:sequence>
                   </xs:extension>
             </xs:complexContent>
       </xs:complexType>
       <xs:complexType name="authenticatedFlickrRequest">
             <xs:complexContent>
                   <xs:extension base="signedFlickrRequest">
                     <xs:sequence>
                          <xs:element name="auth_token" type="xs:string"/>
                     </xs:sequence>
                   </xs:extension>
             </xs:complexContent>
       </xs:complexType>
…
       <xs:element name="activity.userPhotos">
       <xs:complexType>
             <xs:complexContent>
                   <xs:extension base="authenticatedFlickrRequest">
                     <xs:sequence>
                      <xs:element name="timeframe" type="xs:string" minOccurs="0"/>
                      <xs:element name="per_page" type="xs:integer" minOccurs="0"/>
                      <xs:element name="page" type="xs:integer" minOccurs="0"/>
                     </xs:sequence>
                   </xs:extension>
             </xs:complexContent>
       </xs:complexType>
       </xs:element>
…
       <xs:element name="photos.search">
       <xs:complexType>
             <xs:complexContent>
                   <xs:extension base="flickrRequest">
                     <xs:sequence>
                       <xs:element name="user_id" type="xs:string" minOccurs="0"/>
                       <xs:element name="tags" type="xs:string" minOccurs="0"/>
                       <xs:element name="tag_mode" type="anyOrAll" minOccurs="0"/>
                       <xs:element name="text" type="xs:string" minOccurs="0"/>
                       <xs:element name="min_upload_date" type="unixTimeStamp"
                         minOccurs="0"/>
                       <xs:element name="max_upload_date" type="unixTimeStamp"
                         minOccurs="0"/>
                       <xs:element name="min_taken_date" type="unixTimeStamp"
                         minOccurs="0"/>
                       <xs:element name="max_taken_date" type="unixTimeStamp"
```

```
                    minOccurs="0"/>
                  <xs:element name="license" type="xs:string" minOccurs="0"/>
                  <xs:element name="sort" type="sortOrder" minOccurs="0"/>
                  <xs:element name="privacy_filter" type="xs:integer"
                    minOccurs="0"/>
                  <xs:element name="bbox" type="xs:string" minOccurs="0"/>
                  <xs:element name="accuracy" type="xs:integer"
                    minOccurs="0"/>
                  <xs:element name="machine_tags" type="xs:string"
                    minOccurs="0"/>
                  <xs:element name="machine_tag_mode" type="anyOrAll"
                    minOccurs="0"/>
                  <xs:element name="group_id" type="xs:string" minOccurs="0"/>
                  <xs:element name="extras" type="xs:string" minOccurs="0"/>
                  <xs:element name="per_page" type="xs:integer"
                    minOccurs="0"/>
                  <xs:element name="page" type="xs:integer" minOccurs="0"/>
                </xs:sequence>
              </xs:extension>
        </xs:complexContent>
  </xs:complexType>
  </xs:element>
…
  <xs:element name="rsp">
  <xs:complexType>
        <xs:choice>
              <xs:element name="err" form="unqualified">
                    <xs:complexType>
                    <xs:complexContent>
                    <xs:extension base="xs:anyType">
                      <xs:attribute name="code" type="xs:string"/>
                      <xs:attribute name="msg" type="xs:string"/>
                    </xs:extension>
                    </xs:complexContent>
                    </xs:complexType>
              </xs:element>
              <xs:any maxOccurs="unbounded"/>
        </xs:choice>
  </xs:complexType>
  </xs:element>

  <wsdl:operation name="flickr.activity.userPhotos" wsdlx:safe="true">
        <wsdl:input element="activity.userPhotos"/>
        <wsdl:output element="rsp"/>
  </wsdl:operation>
…
  <wsdl:operation name="flickr.photos.search">
        <wsdl:input element="photos.search"/>
        <wsdl:output element="rsp"/>
  </wsdl:operation>
```

## **ANEXO II**. WSDL Búsqueda de vídeos por año

```
<?xml version="1.0" encoding="UTF-8" standalone="no" ?>

<wsdl:definitions
    xmlns:ax21="http://model/xsd" xmlns:ns="http://test"
    xmlns:ns1="http://org.apache.axis2/xsd" xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
    xmlns:soap12="http://schemas.xmlsoap.org/wsdl/soap12/"
    xmlns:wsaw="http://www.w3.org/2006/05/addressing/wsdl"
    xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/" xmlns:xs="http://www.w3.org/2001/XMLSchema"
    targetNamespace="http://test">


  <wsdl:types>
    <xs:schema
        xmlns:ax22="http://model/xsd" attributeFormDefault="qualified"
        elementFormDefault="qualified" targetNamespace="http://test">


      <xs:element name="searchVideosByYear">
        <xs:complexType>
          <xs:sequence>
            <xs:element minOccurs="0" name="year" nillable="true" type="xs:string" />
          </xs:sequence>
        </xs:complexType>
      </xs:element>


      <xs:element name="searchVideosByYearResponse">
        <xs:complexType>
-         <xs:sequence>
            <xs:element
                maxOccurs="unbounded" minOccurs="0"
                name="return" nillable="true" type="ax21:Video" />
          </xs:sequence>
        </xs:complexType>
      </xs:element>
    </xs:schema>


    <xs:schema
        attributeFormDefault="qualified" elementFormDefault="qualified"
        targetNamespace="http://model/xsd">


      <xs:complexType name="Video">
        <xs:sequence>
          <xs:element minOccurs="0" name="catId" nillable="true" type="xs:int" />
          <xs:element minOccurs="0" name="vidAutor" nillable="true" type="xs:string" />
          <xs:element minOccurs="0" name="vidDescripcion" nillable="true" type="xs:string" />
          <xs:element minOccurs="0" name="vidDuracion" nillable="true" type="xs:string" />
          <xs:element minOccurs="0" name="vidFecha" nillable="true" type="xs:string" />
          <xs:element minOccurs="0" name="vidFormato" nillable="true" type="xs:string" />
          <xs:element minOccurs="0" name="vidId" nillable="true" type="xs:string" />
          <xs:element minOccurs="0" name="vidImagen" nillable="true" type="xs:string" />
          <xs:element minOccurs="0" name="vidReproducciones" nillable="true" type="xs:int" />
          <xs:element minOccurs="0" name="vidTitulo" nillable="true" type="xs:string" />
        </xs:sequence>
      </xs:complexType>
    </xs:schema>


  </wsdl:types>


  <wsdl:message name="searchVideosByYearRequest">
    <wsdl:part element="ns:searchVideosByYear" name="parameters" />
  </wsdl:message>


  <wsdl:message name="searchVideosByYearResponse">
    <wsdl:part element="ns:searchVideosByYearResponse" name="parameters" />
  </wsdl:message>


  <wsdl:portType name="FindServicePortType">
    <wsdl:operation name="searchVideosByYear">
      <wsdl:input
          message="ns:searchVideosByYearRequest" wsaw:Action="urn:searchVideosByYear" />
      <wsdl:output
          message="ns:searchVideosByYearResponse"
          wsaw:Action="urn:searchVideosByYearResponse" />
    </wsdl:operation>
  </wsdl:portType>


  <wsdl:binding name="FindServiceSoap11Binding" type="ns:FindServicePortType">
    <soap:binding style="document" transport="http://schemas.xmlsoap.org/soap/http" />
```

```
    <wsdl:operation name="searchVideosByYear">
      <soap:operation soapAction="urn:searchVideosByYear" style="document" />
      <wsdl:input>
        <soap:body use="literal" />
      </wsdl:input>
      <wsdl:output>
        <soap:body use="literal" />
      </wsdl:output>
    </wsdl:operation>
  </wsdl:binding>

  <wsdl:binding name="FindServiceHttpBinding" type="ns:FindServicePortType">
    <http:binding verb="POST" />
    <wsdl:operation name="buscarVideosByYear">
      <http:operation location="buscarVideosByYear" />
      <wsdl:input>
        <mime:content part="parameters" type="application/xml" />
      </wsdl:input>
      <wsdl:output>
        <mime:content part="parameters" type="application/xml" />
      </wsdl:output>
    </wsdl:operation>
  </wsdl:binding>

…

</wsdl:definitions>
```

**ANEXO III**. Elemento Envelope de SOAP para lanzar la operación del WSDL del Anexo I.

```
<soapenv:Envelope
            xmlns:q0="http://test"
            xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
            xmlns:xsd="http://www.w3.org/2001/XMLSchema"
            xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">

      <soapenv:Body>
            <q0:searchVideosByYear>
                  <q0:year>2012</q0:year>
            </q0:searchVideosByYear>
      </soapenv:Body>

</soapenv:Envelope>
```