FIB

Màster en Enginyeria Informàtica (MEI)

Internet, Seguretat i Distribució de Continguts Multimèdia (ISDCM)

Colección de problemas

Applications and web services

SOLUCIONES

Curs 2024-25 Q2
Febrer 2025

Jaime Delgado
Dept. AC

Preguntas Test Cierto/Falso. Indicar si las siguientes afirmaciones son ciertas o falsas.

OSI MODEL

1. The OSI and Internet models are equal. The only difference is that both use different names.

☐ True ☐ False

Answer: False. They differ in the number of layers and the structure of some of them. Furthermore, the layers of the Internet model are not formalized.

2. The Transport layer in the OSI model is equivalent to the TCP/UDP and IP layers in the Internet model.

☐ True ☐ False

Answer: False. OSI Transport is equivalent to Internet Transport, i.e. TCP/UDP.

3. The Presentation and Session layers in the OSI model are equivalent to the Transport layer in the Internet model.

☐ True ☐ False

Answer: False. Presentation and Session in OSI are part of Application in Internet.

4. A TCP segment does not include application level information.

☐ True ☐ False

Answer: False. It includes information of all its upper levels.

5. An IP datagram (network layer data unit) may include a TCP segment (transport layer data unit).

☐ True ☐ False

Answer: True.

MIME

1. MIME was initially proposed as the solution to several limitations of the e-mail format, such as combining several kinds of content in the same message.

☐ True ☐ False

Answer: True. But more important is the ability to encode "multimedia" content in a human-readable way (compatible with ASCII systems).

2. MIME defines, apart from other features, how to combine several kinds of content in the same message.

☐ True ☐ False

Answer: True. Although the most important feature is the possibility of coding “multimedia” content in a readable manner and, therefore, compatible with ASCII systems.

3. `audio`, `image` and `video` are valid MIME content types.

☐ True ☐ False

Answer: True.

4. More than half of the existing subtypes in MIME belong to the `audio`, `image` and `video` content types.

☐ True ☐ False

Answer: False. Most of the subtypes belong to the `application` content type.

5. `image/gif` is a valid combination of MIME content type/subtype.

☐ True ☐ False

Answer: True.

6. `image` and `video` are valid MIME content types, while `example` is not.

☐ True ☐ False

Answer: False. Example is also a valid MIME content type.

7. `image` and `animation` are valid MIME content types.

☐ True ☐ False

Answer: False. **`Animation`** is not a valid MIME content type.

8. In MIME, `font` is subtype of `text`.

☐ True ☐ False

Answer: False. It is a type on its own.

9. `font` is a valid MIME content type.

☐ True ☐ False

Answer: True.

10. `base64` is inefficient is because it needs to transmit 1 byte for every 7 bits of information.

☐ True ☐ False

Answer: False. The size is increased because we need 4 bytes for every 3.

11. `base64` is inefficient is because it needs to transmit 1 extra byte for every octet.

☐ True ☐ False

Answer: False. The size is increased because we need 4 bytes for every 3.

12. `base64` is inefficient is because it needs to transmit 1 extra byte for every 3 octets.

☐ True ☐ False

Answer: True.

13. `base64` is a possible `Content-Transfer-Encoding` in MIME that allows reducing the size of the original encoding of the content.

☐ True ☐ False

Answer: False. The size is increased (we need 4 bytes for every 3).

14. base64 is a possible Content-Transfer-Encoding in MIME. However, the size of the original encoded content is multiplied by 4.

☐ True ☐ False

Answer: False. The size is increased, but we need 4 bytes for every 3.

15. `base64` is a possible `Content-Type` in MIME.

☐ True ☐ False

Answer: False. `base64` is a kind of `Content-Transfer-Encoding`.

16. `base64` is a coding mechanism in MIME.

☐ True ☐ False

Answer: True.

URL

1. `http://urn:example:animal:ferret:nose` is a valid example of a URL.

☐ True ☐ False

Answer: False. The part after `http://` is a URN.

2. `urn:example:animal:ferret:nose` is a valid example of a URN.

☐ True ☐ False

Answer: True.

3. `urn:example:animal:ferret:nose` is a valid example of a URI.

☐ True ☐ False

Answer: True. It is a URN, a particular case of a URI.

4. `urn:myapplication:element:details` is, syntactically, a valid example of URI.

☐ True ☐ False

Answer: True. It is a URN, which is a URI.

5. An IRI (Internationalized Resource Identifier) is a URI that may include non-latin characters.

☐ True ☐ False

Answer: True.

6. An IRI (Internationalized Resource Identifier) is a URI that could be used in different languages.

☐ True ☐ False

Answer: False. URIs have no language associated to. An IRI is an URI that may include non-latin characters.

7. A difference between an URL (Uniform Resource Locator) and an URN (Uniform Resource Name) is that the first one allows to locate the resource, while the second one does not.

☐ True ☐ False

Answer: True.

8. Characters “ç” or “ñ”, for example, may appear in an IRI (Internationalized Resource Identifier).

☐ True ☐ False

Answer: True. An IRI is an URI that may include non-ASCII characters.

HTTP

1. HTTP is not a protocol of type Request - Reply.

☐ True ☐ False

Answer: False. The client always sends a Request, and then the server answers with a Reply.

2. To send a file with the HTTP protocol, we use the GET method.

☐ True ☐ False

Answer: False. We need to use POST.

3. A request message of the HTTP protocol consists of a request line and the body of the message.

☐ True ☐ False

Answer: False. It also has a header.

4. A request message of the HTTP protocol consists of a request line, a header and the body of the message (if any).

☐ True ☐ False

Answer: True.

5. The HTTP’s HEAD method returns the same body that the GET method.

☐ True ☐ False

Answer: False. It only returns the header.

6. The HTTP’s POST, PUT, DELETE and PATCH methods may modify the server.

☐ True ☐ False

Answer: True.

7. HEAD, TRACE and OPTIONS are HTTP methods that do not modify the content of the server.

☐ True ☐ False

Answer: True.

8. The HTTP's POST method is the only one that may modify the server.

☐ True ☐ False

Answer: False. Also the PUT, DELETE and PATCH are allowed to modify.

9. The following header of a HTTP Request is valid:

```
GET /index.html HTTP/1.1
Host: www.example.com
```

☐ True ☐ False

Answer: True.

10. The values of the `If-Modified-Since` and `If-None-Match` elements, if any, are part of the header in a HTTP Request.

☐ True ☐ False

Answer: True.

11. The `If-Modified-Since` and `If-None-Match` elements of the HTTP Request force to download content that we already have at the client.

☐ True ☐ False

Answer: False. It is for avoiding it.

12. The element `If-Modified-Since` of a *HTTP Request* avoids downloading content already available at the client. Another way of doing this is with the element `If-None-Match`.

☐ True ☐ False

Answer: True.

13. An *Entity Tag* in a HTTP header is a *hash* of a resource identified by a URL.

☐ True ☐ False

Answer: True.

14. An *Entity Tag* in a HTTP header provides a time stamp.

☐ True ☐ False

Answer: False. An *Entity Tag* is a *hash* of a resource identified by a URL.

15. HTTP may only transfer HTML content.

☐ True ☐ False

Answer: False. The HTTP protocol may transfer any kind of content.

16. HTTP always works in *stateless* mode; i.e., a given request is independent from previous ones.

☐ True ☐ False

Answer: True.

17. HTTP may work in *stateless* mode (a given request is independent from previous ones) or not, allowing in this latter case to create dependent requests.

☐ True ☐ False

Answer: False. HTTP is always, and only, *stateless*.

18. All HTTP requests include a mandatory body.

☐ True ☐ False

Answer: False. The body is not mandatory. In fact, in the GET method, it is not allowed.

19. The `connection` HTTP Request element allows managing *pipelining*.

☐ True ☐ False

Answer: False. It is for the *persistency*.

20. The `connection` HTTP Request element allows managing *persistency*.

☐ True ☐ False

Answer: True.

21. Both a URL and a URN could be used to locate a resource to be accessed with HTTP.

☐ True ☐ False

Answer: False. Only a URL.

22. The header of a request message of the HTTP protocol consists of several elements. The specific elements to include are specified in the HTTP schema.

☐ True ☐ False

Answer: False. There is no HTTP schema. Elements are defined in the standard and in specific uses of HTTP, even "private" applications.

23. To send content with the HTTP protocol, we may use the POST method.

☐ True ☐ False

Answer: True.

24. To send content with the HTTP protocol, we normally use the POST method, but to use GET is a good alternative.

☐ True ☐ False

Answer: False. GET does not allow to add content in the body.

25. The following part of a header of a HTTP Request is correct:

```
GET /index.html HTTP/1.1
Host: www.example.com
Accept: text/html
Accept-Language: en-us
```

☐ True ☐ False

Answer: True.

26. The HTTP status codes are part of the HTTP Responses. They indicate, for example, if there has been an error at the server.

☐ True ☐ False

Answer: True.

27. The HTTP status codes are part of the HTTP Requests. They indicate, for example, if there has been an error at the server.

☐ True ☐ False

Answer: False. They are part of the HTTP Responses.

28. The HTTP status codes are part of the HTTP Requests. They indicate, for example, if there has been an error at the client.

☐ True ☐ False

Answer: False. They are part of the HTTP Response.

29. The HTTP status codes are included in the header of the HTTP Response.

☐ True ☐ False

Answer: True.

30. PUT and DELETE are methods of the HTTP protocol.

☐ True ☐ False

Answer: True.

31. The values of the `Accept` element of the HTTP Request are MIME subtypes.

☐ True ☐ False

Answer: True.

32. The value of the `Etag` element of the HTTP Response is assigned by the server.

☐ True ☐ False

Answer: True.

33. The value of the `Etag` element of the "HTTP Response" is assigned by the server by including a time stamp in the response.

☐ True ☐ False

Answer: False. It is assigned by the server, but it is a hash of the content, not a time stamp.

34. The value of the `Etag` element of the HTTP Response is assigned by the client when receiving and answer from the server.

☐ True ☐ False

Answer: False. It is assigned by the server

35. The following part of a header of a HTTP Request is correct:

```
GET /index.html HTTP/1.1 200 OK
Host: www.example.com
Accept: text/html
Accept-Language: en-us
```

☐ True ☐ False

Answer: False. The value “200 OK” in the Request Line is only part of a HTTP Response.

36. HTTP/2 allows Responses generated without the need that a client sends a previous Request.

☐ True ☐ False

Answer: True.

37. The problem of HTTP/2 is its lack of backwards compatibility with HTTP/1.1.

☐ True ☐ False

Answer: False. They are compatible.

38. HTTP/2 specifies mechanisms to improve the performance of HTTP implementations.

☐ True ☐ False

Answer: True.

39. Although there are improvements, HTTP/2 is backwards compatible with HTTP/1.1.

☐ True ☐ False

Answer: True.

40. HTTP/2 adds new methods over HTTP/1.1.

☐ True ☐ False

Answer: False.

41. HTTP/2 is split into 2 sub-levels in order to add new methods to HTTP/1.1.

☐ True ☐ False

Answer: False. The splitting into 2 levels is to keep the methods and to change the implementation on how to use TCP.

42. HTTP/2 adds new methods to HTTP/1.1, but they are optional.

☐ True ☐ False

Answer: False. There are no new methods.

XML

1. XML tags can only be defined by SDOs (Standards Development Organizations).

☐ True ☐ False

Answer: False. Every designer of an XML schema can do it.

2. Every XML designer may use XML `Name spaces`. For this purpose, the `xmlns` attribute needs to be used.

☐ True ☐ False

Answer: True.

3. A specific XML schema is written in the XML language, but the specification of the syntax to use for writing XML schemas uses another mechanism.

☐ True ☐ False

Answer: False. XML schema is also specified as an XML schema.

4. The following part of an XML instance includes "elements" and "text", but no "attributes":

```
<book>
      <title lang="en">XML</title>
      <author>John Smith</author>
      <year>2018</year>
</book>
```

☐ True ☐ False

Answer: False. "lang" is an attribute.

5. The following part of an XML instance includes a root element, "attributes" and "text", but no other "elements":

```
<book>
      <title lang="en">XML</title>
      <author>John Smith</author>
      <year>2018</year>
</book>
```

☐ True ☐ False

Answer: False. "title", "author" and "year" are elements.

6. The following part of an XML instance includes one or more "element", "attribute" and "text".

```
<book>
      <title lang="en">XML</title>
      <author>John Smith</author>
      <year>2018</year>
</book>
```

☐ True ☐ False

Answer: True.

7. The following part of an XML instance includes "attributes", "text" and "elements":

```
<book>
      <title>XML</title>
      <author>John Smith</author>
      <year>2019</year>
</book>
```

☐ True ☐ False

Answer: False. There are no "attributes".

8. The following part of an XML instance includes "text" and "elements":

```
<book>
      <title>XML</title>
      <author>John Smith</author>
      <year>2019</year>
</book>
```

☐ True ☐ False

Answer: True.

9. The following part of an XML instance includes “attributes”, “text” and “elements”:

```
<book>
      <title lang=”en”>XML</title>
      <author>John Smith</author>
      <year>2019</year>
</book>
```

☐ True ☐ False

Answer: True.

10. The following part of an XML instance includes “text” and “elements”, but no “attributes”:

```
<book>
      <title>XML</title>
      <author>John Smith</author>
      <year>2019</year>
</book>
```

☐ True ☐ False

Answer: True.

11. This fragment of an XML document is the root element of a schema:

```
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
            targetNamespace="http://www.films.org"
            xmlns="http://www.films.org">
```

☐ True ☐ False

Answer: True.

12. Given the following fragment of XML document, in `http://www.films.org` we have possible values of “films”:

```
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
            targetNamespace="http://www.films.org"
            xmlns="http://www.films.org">
```

☐ True ☐ False

Answer: False. We have a “name space”: identifiers of elements (no values).

13. The attribute `schemaLocation` of an XML document may identify the file that contains its XML schema.

☐ True ☐ False

Answer: True.

14. With XSLT, we can convert from an XML schema to a different XML schema.

☐ True ☐ False

Answer: True.

15. The purpose of XSL and XSLT is to convert from an XML schema to HTML.

☐ True ☐ False

Answer: False. It is to convert from an XML schema to a different XML schema.

16. Using XSL and XSLT we can convert from an XML schema to a different XML schema, including from an XML schema to HTML.

☐ True ☐ False

Answer: True.

17. XSL and XSLT are tools to help transforming from documents following one XML schema to another. However, it is not possible to transform from an XML schema to HTML.

☐ True ☐ False

Answer: False. It is also possible the transformation to HTML.

18. XSL and XSLT are tools to help transforming from one XML schema to another, including the transformation from a XML schema to HTML.

☐ True ☐ False

Answer: True.

19. JSON (JavaScript Object Notation) is a data interchange format standardized by the W3C (World Wide Web Consortium).

☐ True ☐ False

Answer: False. It is an IETF standard.

WEB SERVICES

1. The only difference between versions 1.1 and 2.0 of WSDL is the name: Web Services *Description* Language (for 2.0) and Web Services *Definition* Language (for 1.1).

☐ True ☐ False

Answer: False. There are other differences, mainly syntax.

2. UDDI is still widely used to make WSDL services public.

☐ True ☐ False

Answer: False. It is no longer used.

3. WSDL is used by a web service provider to allow others to implement clients to remotely access that service.

☐ True ☐ False

Answer: True.

4. WSDL allows defining bindings over different protocols such as SOAP or HTTP.

☐ True ☐ False

Answer: True. It is possible to indicate that a service is implemented over SOAP or directly over HTTP methods.

5. WSDL does not allow defining bindings directly over HTTP.

☐ True ☐ False

Answer: False. It is possible, for example, to indicate that a service is implemented over SOAP or directly over HTTP methods.

6. SOAP defines, in XML, requests and answers to be sent only over HTTP.

☐ True ☐ False

Answer: False. It is possible to use other protocols, such as SMTP.

7. *http://www.w3.org/ns/wsdl/soap* identifies the "WSDL SOAP Binding Namespace".

☐ True ☐ False

Answer: True.

8. For a given request, we usually need to send more information (more bytes) when using SOAP than when using REST.

☐ True ☐ False

Answer: True.

9. Even though using REST to implement a web service, we could use SOAP to return the responses.

☐ True ☐ False

Answer: True.

10. A service offered as REST is always accessed with the HTTP method GET.

☐ True ☐ False

Answer: False. It is also possible to use other methods.

11. A WSDL file only defines operations from a REST web service.

☐ True ☐ False

Answer: False. It defines operations of a service in general.

12. In REST, only the GET and POST HTTP methods can be used.

☐ True ☐ False

Answer: False. Also PUT or DELETE could be used.

13. SOAP is not the only way to send operations specified with WSDL.

☐ True ☐ False

Answer: True.

14. With SOAP, it is possible to connect to a public web service.

☐ True ☐ False

Answer: True.

Problema 1

En el Anexo I tenemos un fragmento de un servicio, expresado en WSDL, para buscar vídeos en función de su año de producción.

Contestar razonada y brevemente a las siguientes preguntas:

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

Al final del WSDL tenemos:

```
<wsdl:binding name="FindServiceHttpBinding" type="ns:FindServicePortType">
    <http:binding verb="POST" />
    <wsdl:operation name="searchVideosByYear">
      <http:operation location="searchVideosByYear" />
      <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>
```

Vemos, por el nombre, que esto es la especificación del binding con HTTP. Vemos además que se hace con un método POST.

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

Tenemos 8 campos de tipo string y 2 de tipo integer.

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

Definen los name spaces.

4) Justificar (en función del servicio WSDL) por qué el elemento `Body` del `Envelope` de SOAP es correcto.

En el WSDL tenemos que el tipo de la operación searchVideosByYear es:

```
<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>
```

Y en el Body tenemos el parámetro year de tipo string (valor “2012”).

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

Las operaciones disponibles y, para cada una de ellas, los mensajes de Input y Output.

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

1 nuevo schema con 2 nuevos tipos en “wsdl:types”:

```
<wsdl:types>
…
 <xs:schema  …  >

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

  <xs:element name="searchVideosByTitleResponse">
    <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>
…
  </wsdl:types>
```

2 nuevos mensajes:

```
<wsdl:message name="searchVideosByTitleRequest">
  <wsdl:part element="ns:searchVideosByTitle" name="parameters" />
</wsdl:message>

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

1 nueva operación en el port:

```
<wsdl:portType name="FindServicePortType">
…
  <wsdl:operation name="searchVideosByTitle">
    <wsdl:input
      message="ns:searchVideosByTitleRequest"
      wsaw:Action="urn:searchVideosByTitle" />
    <wsdl:output
      message="ns:searchVideosByTitleResponse"
      wsaw:Action="urn:searchVideosByTitleResponse" />
  </wsdl:operation>
</wsdl:portType>
```

1 nueva operación en cada uno de los 2 bindings:

```
<wsdl:binding name="FindServiceSoap11Binding" … />
…
  <wsdl:operation name="searchVideosByTitle">
   <soap:operation soapAction="urn:searchVideosByTitle" style="document" />
… (el resto igual)
  </wsdl:operation>
</wsdl:binding>
```

```
<wsdl:binding name="FindServiceHttpBinding" type="ns:FindServicePortType">
  <http:binding verb="POST" />
…
  <wsdl:operation name="searchVideosByTitle">
      <http:operation location="searchVideosByTitle" />
… (el resto igual)
  </wsdl:operation>
</wsdl:binding>

  </wsdl:operation>
</wsdl:binding>
```

Problema 2

El Anexo II presenta parte de un ejemplo de WSDL extraído de la web oficial del W3C. Se trata de un servicio de reservas de un hotel llamado GreatH.

Contestar razonada y brevemente a las siguientes preguntas:

1) ¿De qué versión de WSDL se trata?

WSDL 2.0. Se puede ver en varios detalles: "description", "interface y "endpoint" en vez de "definitions", "portType" y "port".

2) Como se puede observar, se ofrece una única operación para verificar la disponibilidad de habitaciones del hotel. ¿Cuáles son los parámetros de entrada y de salida? Indicar también sus tipos.

Entrada: checkInDate (xs:date), checkOutDate ("xs:date"), roomType (string).

Salida: Un entero doble. Podría ser un precio.

3) ¿Qué dos respuestas alternativas puede dar dicha operación?

Double integer (precio, por ejemplo) o string con mensaje de error.

4) ¿Qué indica el valor del atributo pattern de la operación opCheckAvailability?

pattern="http://www.w3.org/ns/wsdl/in-out"

Pregunta y respuesta obligatoria.

5) ¿Qué devolverá el sistema en caso de error (es decir, que los datos de entrada sean inválidos)?

Un string.

6) ¿En qué dirección (URL) se puede acceder al servicio cuando se implementa en un web service?

address ="http://greath.example.com/2004/reservation"/>

Suponer la siguiente especificación (extraída también de la web oficial de W3C):

- *MakeReservation*. To make a reservation, a client must provide a name, address, and credit card information, and the service will return a confirmation number if the reservation is successful. The service will return an error message if the credit card number or any other data field is invalid. Thus, the service will accept a `makeReservation` message and return a `makeReservationResponse` or `invalidCreditCardFault` message.

7) Extender los elementos types e interface del WSDL del anexo para que pueda ofrecer también esta nueva operación.

```
  <types>
…
      <xs:element name="makeReservation" type="tMakeReservation"/>

      <xs:complexType name="tMakeReservation">
        <xs:sequence>
          <xs:element  name="name" type="xs:string"/>
          <xs:element  name="address" type="xs:string"/>
          <xs:element  name="creditCardInformation" type="xs:string"/>
        </xs:sequence>
      </xs:complexType>

      <xs:element name="makeReservationResponse" type="xs:double"/>
      <xs:element name="invalidCreditCardError" type="xs:string"/>

    </xs:schema>

  </types>

  <interface  name = "reservationInterface" >
…
    <fault name = "invalidCreditCardFault"
            element = "ghns:invalidCreditCardError"/>

    <operation name="opMakeReservation"
            pattern="http://www.w3.org/ns/wsdl/in-out"
            style="http://www.w3.org/ns/wsdl/style/iri"
            wsdlx:safe = "true">
        <input messageLabel="In"
              element="ghns:makeReservation" />
        <output messageLabel="Out"
              element="ghns:makeReservationResponse" />
        <outfault ref="tns:invalidCreditCardFault" messageLabel="Out"/>
    </operation>

  </interface>
```

8) Si quisiéramos usar REST en vez de SOAP, ¿qué cambiaría en el diseño anterior? Indicar también que método(s) HTTP se debe(n) usar.

Necesitaríamos enviar los parámetros de las operaciones en una URL (o sea, en el Header del Request del método de HTTP). Respecto a los resultados, irían en el Body del Response del método de HTTP, codificados con SOAP u otro mecanismo alternativo.

Respecto a los métodos, deberíamos usar GET si es posible. Si seguimos el criterio de modificación (usar POST cuando se modifica la información en el servidor), GET iría bien para checkAvailability, pero sería necesario un POST para makeReservation.

Problema 3

El Anexo II presenta parte de un ejemplo de WSDL extraído de la web oficial del W3C. Se trata de un servicio de reservas de un hotel llamado GreatH.

Contestar razonada y brevemente a las siguientes preguntas:

1) El WSDL es versión 2.0. Enumerar 3 diferencias de terminología respecto a la versión 1.1.

   "description", "interface y "endpoint" en vez de "definitions", "portType" y "port".

2) ¿Dónde se indica, y con qué elemento XML, que se devuelve un mensaje de error si no se puede dar una respuesta correcta?

   En `interface`. Elemento:

```
<fault name = "invalidDataFault"
        element = "ghns:invalidDataError"/>
```

3) ¿Dónde se especifica que se usa SOAP 1.2 como formato de mensaje y HTTP como protocolo de transmisión?

   En dos atributos del elemento `binding`:

   type="http://www.w3.org/ns/wsdl/soap"

   wsoap:protocol="http://www.w3.org/2003/05/soap/bindings/HTTP/"

   (el prefijo `wsoap` indica que está especificado en la "SOAP binding extension" de WSDL 2.0).

4) En el elemento `binding` tenemos el atributo wsoap:mep (`wsoap:mep="http://www.w3.org/2003/05/soap/mep/soap-response"`), explicado en el anexo. Razonar cómo podemos usar un modelo REST con esta especificación WSDL sobre SOAP.

   Tal como se define este MEP, se deben enviar los parámetros del request con un HTTP GET sin un envelope SOAP. Por tanto, la única manera es usando la URL, es decir, un modelo REST para la pregunta, y una estructura SOAP para la respuesta.

Exercise 4

Annex III presents a WSDL fragment.

Reasoned and briefly answer the following questions:

1) How many operations and with what parameters defines this WSDL? Which XML elements contain that information?

One operation, `consulta_libres`.

Input parameters: `int id_vuelo`, `int fecha`

Output parameters: `int consulta_libresReturn`

In the element `wsdl:types`, as it is described next:

```
<wsdl:types>
<schema elementFormDefault="qualified" targetNamespace="http://vueloWS" …>
 <element name="consulta_libres">
  <complexType>

   <sequence>
    <element name="id_vuelo" type="xsd:int"/>
    <element name="fecha" type="xsd:int"/>
   </sequence>
  </complexType>
 </element>
 <element name="consulta_libresResponse">
  <complexType>
   <sequence>
```

```
        <element name="consulta_libresReturn" type="xsd:int"/>
      </sequence>
    </complexType>
  </element>
</wsdl:types>
```

2) Which element of the WSDL describes the URL in which this service operates?

http://localhost:8080/vueloWS/services/vuelo. In the element `service`.

```
<wsdl:service name="vueloService">
   <wsdl:port binding="impl:vueloSoapBinding" name="vuelo">
      <wsdlsoap:address location="http://localhost:8080/vueloWS/services/vuelo"/>
   </wsdl:port>
</wsdl:service>
```

Exercise 5

Annex IV presents a WSDL description of a Bookstore service. Annexes V and VI provide two schemas imported from the WSDL.

Reasoned and briefly answer the following questions:

PART I

1) In the `wsdl:description` tag, there are several namespace definitions. Where and for which purpose is the **`wsdlx`** namespace used?

In the wsdl:operation element. For adding a new attribute.

2) In the `wsdl:description` tag, there are several namespace definitions. Where and for which purpose is the **`msg`** namespace used?

In the wsdl:input and wsdl:output elements. For specifying the input and output operations, respectively. They are in the booklist.xsd file.

3) How could we add a second operation to the service? Which XML elements should be added and where?

We need:

- new wsdl:operation and wsdl:binding elements,
- new input and output elements in the WSDL document, and new parameters in the booklist.xsd file.

PART II

4) The attribute `wsdlx:safe` of `wsdl:operation` is defined in a WSDL extension. This attribute declares that this operation is idempotent; i.e. it does not modify the resource and can therefore be called many times with the same results. Justify if this is a restriction or an advantage if we want to implement the service as REST?

It is an advantage, since REST implies that there is no "memory" in the sequence of operations.

5) Explain the meaning of the binding in this WSDL. With this binding, is it possible to implement the service as REST?

The binding is basically:

```
<wsdl:binding name="BookListHTTPBinding"
  type="http://www.w3.org/ns/wsdl/http"
  interface="tns:BookListInterface">
 <wsdl:operation ref="tns:getBookList" whttp:method="GET"/>
</wsdl:binding>
```

so the type is “http" and the method is "GET".

We can implement the “getBookList” operation with REST, since REST works over HTTP and uses GET when the operation does not imply modifications in the server, as it is the case.

6) Propose a new `wsdl:binding` element to implement the service with SOAP?

The binding should be over SOAP. It would change to (we assume SOAP is implemented over HTTP):

```
<wsdl:binding name="BookListSOAPBinding"
  type="http://www.w3.org/ns/wsdl/soap"
  interface="tns:BookListInterface"
  protocol="http://www.w3.org/2006/01/soap11/bindings/HTTP/”>
 <wsdl:operation ref="tns:getBookList" wsoap:action="http://www.bookstore.com/getBookList/"/>
</wsdl:binding>
```

In addition, the `wsoap` namespace should be added at the beginning.

Exercise 6

Annex IV presents a WSDL description of a Bookstore service. Annexes V and VI provide “xsd” files imported from the WSDL.

Reasoned and briefly answer the following questions:

PART I

1) Which version of WDSL is used? How do you know it?

It is version 2.0, since it is a “Description” and not a “Definition”.

2) In the `wsdl:description` tag, there are several name space definitions. Explain what the purpose of the following name spaces is: `wsdl`, `tns`, `whttp`, `xs`.

xmlns:wsdl="http://www.w3.org/ns/wsdl"

The wsdl standard.

xmlns:tns="http://www.bookstore.org/booklist/wsdl"

The specific Bookstore application.

xmlns:whttp="http://www.w3.org/ns/wsdl/http"

The wsdl binding to HTTP.

xmlns:xs="http://www.w3.org/2001/XMLSchema"

The XML Schema standard.

3) How many operations, and with what input and output parameters, does this WSDL specify?

NOTE 1: For the output parameters, see the **wsdlx Schema** explanations in the Annex.

NOTE 2: The `schema` element in `booklist.xsd` includes:
"`xmlns:booksvc="http://www.bookstore.org/book/wsdl"`"

NOTE 3: The `url` attribute includes two attributes from the WSDL extensions namespace. The attributes `wsdlx:interface` and `wsdlx:binding` identify the specific WSDL interface and binding for the service.

One operation: getBookList.

Input parameters: 5 strings: author, title, publisher, subject and language.

Output parameters: A "book" element of type "BookType", which includes a string "title" and a URI "url":

```
<complexType name="bookType">
  <attribute name="title" type="string"/>
  <attribute name="url" type="anyURI"
    wsdlx:interface="booksvc:BookInterface"
    wsdlx:binding="booksvc:BookHTTPBinding"/>
```

The url attribute is a link to a book details REST Web service, which returns the details for the specific book.

4) Which is the URL of the service defined? How do you know it?

The URL of the book list service is http://www.bookstore.com/books/. It is the `address` attribute in the `endpoint` element in `service`.

5) Concerning the attributes of `wsdl:operation`, what is the meaning of the value of the attribute "pattern"?

It means that there should be both input and output operations.

PART II

6) The attribute `wsdlx:safe` of `wsdl:operation` is defined in a WSDL extension. This attribute declares that this operation is idempotent; i.e. it does not modify the resource and can therefore be called many times with the same results. Justify if this is a restriction or an advantage if we want to implement the service as REST?

It is an advantage, since REST implies that there is no "memory" in the sequence of operations.

7) Explain the meaning of the binding in this WSDL. Justify how it could help, or not, to implement the service as REST.

The type is "http" and the method is "GET". We can implement the "getBookList" operation with REST, since REST works over HTTP and uses GET when the operation does not imply modifications in the server.

8) Could we use the following URL as a REST request for this service? Why? If affirmative answer, what response should we expect from this request?

`http://www.bookstore.com/books/?subject=computers/eclipse`

Yes, we could use it. This example uses only one of the 5 possible input parameters. Furthermore, it makes use of another XML schema not introduced that provides details for books. Therefore, an answer saying this is incorrect is also valid, since we do not know the meaning of "computers/eclipse".

The response would be a list of computer books about "Eclipse".

9) What should we change in the WSDL to implement the service with SOAP?

The binding should be over SOAP. It would change to:

```
<wsdl:binding name="BookListHTTPBinding"
  type="http://www.w3.org/ns/wsdl/soap"
  interface="tns:BookListInterface">
 <wsdl:operation ref="tns:getBookList" wsoap:action="http://www.bookstore.com/getBookList/"/>
</wsdl:binding>
```

In addition, the `wsoap` name space should be added at the beginning.

10) Let's assume that we want to implement the query for selecting the books with a **form**. Which data fields do we need? Independently of the binding specified in the WSDL, which HTTP method do we need to send the form?

The ones specified for the input parameters; i.e., 5 strings: author, title, publisher, subject and language.

The form allows the server developer to indicate the HTTP method to use. If GET is used, the query will be included in the URL, while if POST is chosen, then the query will be included in the body.

Exercise 7

Annex VII presents a WSDL of a *BookList* service intended to be implemented as REST. Annex V includes a "xsd" file imported from the WSDL.

Reasoned and briefly answer the following questions:

1) What are the following URIs for?:
`http://www.w3.org/ns/wsdl"` and `http://www.bookstore.org/booklist/wsdl`?

These URIs are specified in two name spaces:

```
xmlns:wsdl="http://www.w3.org/ns/wsdl"
xmlns:tns="http://www.bookstore.org/booklist/wsdl"
```

The first one refers to the WSDL standard.

The second one refers to the specific *BookList* service.

2) What is the only operation specified in this WSDL? Identify its "`pattern`", where is its value indicated, and describe its meaning?

The only operation is `getBookList`.

Its pattern is indicated in the interface element:

```
<wsdl:interface name="BookListInterface">
     <wsdl:operation
        name="getBookList"
        pattern="http://www.w3.org/ns/wsdl/in-out"
        style="http://www.w3.org/ns/wsdl/style/iri"
        wsdlx:safe="true">
           …
     </wsdl:operation>
</wsdl:interface>
```

It means that the operation has both a Request and a Response.

3) Which of the 4 parts of the *Description* (*types, interface, binding, service*) shows that the service may be implemented as REST? In which particular line (or lines)?

In the *binding* part, since REST works over HTTP. In particular:

```
<wsdl:binding
     name="BookListHTTPBinding"
     type="http://www.w3.org/ns/wsdl/http"
     interface="tns:BookListInterface">
     …
  </wsdl:binding>
```

4) A REST service, over which HTTP methods could be implemented? In the particular case of the Annex, which methods are used? Why? Where is it specified?

A REST service may use any of the main HTTP methods, i.e. GET, POST, PUT and DELETE. In this case, however, it only uses GET, since the operation does not imply modifications in the server. It is specified in:

```
  <wsdl:binding
     name="BookListHTTPBinding"
     type="http://www.w3.org/ns/wsdl/http"
     interface="tns:BookListInterface">
    <wsdl:documentation>
       The binding for the book list service.
    </wsdl:documentation>
    <wsdl:operation ref="tns:getBookList" whttp:method="GET"/>
  </wsdl:binding>
```

5) What should we change in the WSDL to implement the service with SOAP?

The binding should be over SOAP. It would change to:

```
<wsdl:binding name="BookListHTTPBinding"
   type="http://www.w3.org/ns/wsdl/soap"
   interface="tns:BookListInterface">
  <wsdl:operation ref="tns:getBookList"
wsoap:action="http://www.bookstore.com/getBookList/"/>
</wsdl:binding>
```

In addition, the `wsoap` name space should be added at the beginning.

Exercise 8

Annex IV shows a WSDL of a `HelloService` service.

Reasoned and briefly answer the following questions:

1) Which version of WDSL is used? How do you know it?

It is version 1.1, since it uses "Definitions" and not a "Description".

2) In the `definitions` tag, there are several definitions of name spaces. Explain what the purpose of the following name spaces is: `soap`, `tns`, `xsd`.

xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"

The use of SOAP in the WSDL standard.

xmlns:tns="http://www.examples.com/wsdl/HelloService.wsdl"

The specific HelloService application.

xmlns:xsd="http://www.w3.org/2001/XMLSchema"

The XML Schema standard.

3) How many operations, and with which input and output parameters, does specify this WSDL?

One operation: sayHello.

Input parameters: 1 string: firstName.

Output parameters: 1 string: greeting.

4) Which is the URL of the defined service? How do you know it?

The URL of the Hello service is http://www.examples.com/sayHello/. It is the value of the `location` attribute in the `soap:address` element in `port` in `service`.

5) Which elements of the WSDL do we need to change, and which ones to add, if we want a second operation "SayGoodbye" (through the same "port")? This operation returns the translation of the English word "goodbye" to the language indicated in the Request.

Add 2 new "message" elements for "SayGoodbyeRequest" and "SayGoodbyeResponse".

Add 1 new operation to the "portType": "sayGoodbye".

Add a second operation to the "binding".

Minor changes (for naming consistency):

- The URL in "location". For example to: location="http://www.examples.com/SayHelloAndGoodbye/".

- targetNamespace and xmlns:tns.

6) What information provides the *binding* of this WSDL?

It is a binding to SOAP over HTTP, where also the operation is specified.

7) What should be changed in the *binding* to implement the service as REST.

We should specify the binding to HTTP, and the GET method (the service does not modify the server).

8) Assuming that we have available the second operation mentioned in question 5, is the following a valid URL to request the *SayGoodbye* service in REST?

http://www.examples.com/SayHelloAndGoodbye/SayGoodbye?language=french

If so, which answer should be provided? If not, provide a correct URL and the corresponding answer.

Yes, it may be valid assuming the changes of the URL once the second operation has been added.

The answer should be the translation to French of "goodbye", included in the body of the HTTP Response.

Exercise 9

Annex VIII shows a WSDL of a `HelloService` service.

Reasoned and briefly answer the following questions:

1) In the `definitions` tag, there are several definitions of name spaces. Explain which correspond to existing standards and which are specific for this WSDL specification.

Standard ones:

xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"

xmlns:xsd="http://www.w3.org/2001/XMLSchema"

xmlns="http://schemas.xmlsoap.org/wsdl/"

Specific ones:

xmlns:tns="http://www.examples.com/wsdl/HelloService.wsdl"

2) Which elements of the WSDL do we need to change, and which ones to add, if we want to have a second operation "SayGoodbye" (through the same "port")? This operation returns the translation of the English word "goodbye" to the language indicated in the Request.

Add 2 new "message" elements for "SayGoodbyeRequest" and "SayGoodbyeResponse".

Add 1 new operation to the "portType": "sayGoodbye".

Add a second operation to the "binding".

Minor changes (for naming consistency):

- The URL in "location". For example to: location="http://www.examples.com/SayHelloAndGoodbye/".

- targetNamespace and xmlns:tns.

3) According to the *binding* of this WSDL, over which application protocol is the SOAP message transferred?

HTTP. It is a binding to SOAP over HTTP, where also the operation is specified.

4) What should be changed in the *binding* to implement the service as REST?

We should specify the binding to HTTP, and the GET method (the service does not modify the server).

5) Assuming that we have available the second operation mentioned in question 5, is the following a valid URL to request the *SayGoodbye* service in REST?

`http://www.examples.com/SayHelloAndGoodbye/SayGoodbye?language=french`

If so, which answer should be provided? If not, provide a correct URL and the corresponding answer.

Yes, it may be valid assuming the changes of the URL once the second operation has been added.

The answer should be the translation to French of "goodbye", included in the body of the HTTP Response.

Problema 10

El Anexo IV muestra el WSDL 2.0 de un servicio `Bookstore`. El WSDL importa un schema que no está incluido.

Razonada y brevemente contestar las siguientes preguntas:

1) ¿Qué información debería haber en el XSD importado, dada la especificación original del anexo?

The elements corresponding to the types of the operations *getBookListType* and *bookListType*.

2) Tal como está especificado este servicio, justificar por qué es conveniente implementarlo como REST. Indicar los elementos del WSDL que ayudan a ello. Indicar asimismo qué faltaría añadir para tener completo el *Request* y el *Response*.

According to the "binding" element:

```
<wsdl:binding name="BookListHTTPBinding"
   type="http://www.w3.org/ns/wsdl/http"
   interface="tns:BookListInterface">
  <wsdl:documentation>
     The HTTP binding for the book list service.
  </wsdl:documentation>
  <wsdl:operation ref="tns:getBookList" whttp:method="GET"/>
</wsdl:binding>
```

The binding is directly over HTTP. This facilitates implementing the service as REST. In addition, the GET method is indicated, which is feasible since the operation does not change the server content and the parameters are passed in the URL itself. What is missing is to define how to provide the response, but it could be, for example, simply an XML document in the body of the HTTP response.

Queremos añadir una segunda operación "GetSpecificBookData" (a través de **un nuevo** "port"). Esta operación tiene como parámetro de entrada una URL que identifica al libro, y devuelve como información del libro los mismos parámetros de entrada que tiene la operación ya existente.

3) Indicar qué se debe añadir y cambiar en la especificación.

We should add a second "interface" where to include the new "operation":

```
<wsdl:interface name="SpecificBookDataInterface">
  <wsdl:operation name="getSpecificBookData"
     pattern="http://www.w3.org/ns/wsdl/in-out"
     style="http://www.w3.org/ns/wsdl/style/iri"
     wsdlx:safe="true">
    <wsdl:documentation>
       This operation returns data of a specific book.
    </wsdl:documentation>
    <wsdl:input element="msg:getSpecificBookData"/>
    <wsdl:output element="msg:bookData"/>
  </wsdl:operation>
</wsdl:interface>
```

We should also add the corresponding types in the imported schema.

Finally, the binding should be also updated.

Problema 11

El Anexo IV muestra el WSDL 2.0 de un servicio `Bookstore`. El Anexo V proporciona un schema que se importa.

Razonada y brevemente contestar las siguientes preguntas:

1) En el tag `description` hay varias definiciones de espacios de nombres. Explicar cuáles corresponden a estándares existentes y cuáles son específicas para esta especificación WSDL.

Standard ones (in the w3.org domain):

xmlns:wsdl="http://www.w3.org/ns/wsdl"

xmlns:whttp="http://www.w3.org/ns/wsdl/http"

xmlns:wsdlx="http://www.w3.org/ns/wsdl-extensions"

xmlns:xs="http://www.w3.org/2001/XMLSchema"

Specific ones (in the bookstore.org domain):

xmlns:tns=http://www.bookstore.org/booklist/wsdl

xmlns:msg="http://www.bookstore.org/booklist/xsd">

2) Tal como está especificado este servicio, ¿es mejor implementarlo con SOAP o como REST? ¿Qué deberíamos cambiar en el WSDL para que fuese mejor implementarlo de la otra forma?

According to the "binding" element:

```
<wsdl:binding name="BookListHTTPBinding"
   type="http://www.w3.org/ns/wsdl/http"
   interface="tns:BookListInterface">
  <wsdl:documentation>
     The HTTP binding for the book list service.
  </wsdl:documentation>
  <wsdl:operation ref="tns:getBookList" whttp:method="GET"/>
</wsdl:binding>
```

The binding is directly over HTTP. Therefore, we could implement the service as REST. In addition, the GET method is indicated, which is feasible since the operation does not change the server content and the parameters are passed in the URL itself. What is missing is to define how to provide the response, but it could be, for example, simply an XML document in the body of the HTTP response.

If we would like to implement the operation as SOAP, we should change the "binding" element to SOAP, including the SOAP messages.

Queremos añadir una segunda operación "GetSpecificBookData" (a través del mismo "port"). Esta operación tiene como parámetro de entrada una URL que identifica al libro, y devuelve como información del libro los mismos parámetros de entrada que tiene la operación ya existente.

3) Dar la nueva sintaxis del elemento "interface".

We should add a second "operation" (in bold):

```
<wsdl:interface name="BookListInterface">
  <wsdl:operation name="getBookList"
     pattern="http://www.w3.org/ns/wsdl/in-out"
     style="http://www.w3.org/ns/wsdl/style/iri"
     wsdlx:safe="true">
    <wsdl:documentation>
       This operation returns a list of books.
    </wsdl:documentation>
    <wsdl:input element="msg:getBookList"/>
    <wsdl:output element="msg:bookList"/>
  </wsdl:operation>

  <wsdl:operation name="getSpecificBookData"
     pattern="http://www.w3.org/ns/wsdl/in-out"
     style="http://www.w3.org/ns/wsdl/style/iri"
     wsdlx:safe="true">
    <wsdl:documentation>
       This operation returns data of a specific book.
    </wsdl:documentation>
    <wsdl:input element="msg:getSpecificBookData"/>
    <wsdl:output element="msg:bookData"/>
```

```
    </wsdl:operation>
  </wsdl:interface>
```

4) ¿Qué cambios será necesario hacer en el XSD importado?

We need to add the corresponding elements and types for the second operation:

```
<element name="getSpecificBookData" type="tns:getSpecificBookDataType">
   <annotation> <documentation>
         The request element for the get specific book data service.
   </documentation>  </annotation>
</element>

<element name="specificBookData" type="tns:specificBookDataType">
   <annotation> <documentation>
         The response element for the get specific book data service.
   </documentation> </annotation>
</element>

<complexType name="getSpecificBookDataType">
    <attribute  name="url" type="anyURI"
                wsdlx:interface="booksvc:BookInterface"
                wsdlx:binding="booksvc:BookHTTPBinding"/>
</complexType>

<complexType name="specificBookDataType">
  <sequence>
    <element name="author" type="string" minOccurs="0" maxOccurs="unbounded"/>
    <element name="title" type="string" minOccurs="0" maxOccurs="1"/>
    <element name="publisher" type="string" minOccurs="0" maxOccurs="1"/>
    <element name="subject" type="string" minOccurs="0" maxOccurs="1"/>
    <element name="language" type="string" minOccurs="0" maxOccurs="unbounded"/>
  </sequence>
</complexType>
```

There is another solution re-using the type structure of the obtained book that here is in the output parameters and in the first operation is in the input ones. Then, the "specificBookData" element would be:

```
<element name="specificBookData" type="tns:getBookListType">
   <annotation> <documentation>
         The response element for the get specific book data service.
   </documentation> </annotation>
</element>
```

And we should remove the "specificBookDataType" type.

Exercise 12

Annex IX shows a WSDL of a `Tutorial` service.

Reasoned and briefly answer the following questions:

1) Is there a complete XML Schema inside the WSDL? If so, what is their purpose?

Yes. To separate the `types` definition, which could be used independently.

2) We want to have a second operation "GetTutorialAuthor" (through a new "port"). The input should be a "TutorialID" and the output should be the name (in a unique string) of the author. Provide the syntax of the new "message" and "portType" elements needed.

We need to add 2 new "message" elements for "GetTutorialAuthorInput" and "GetTutorialAuthorOutput".

```
<message name="GetTutorialAuthorInput">
   <part name="body" element="xsd1:TutorialIDRequest"/>
</message>
<message name="GetTutorialAuthorOutput">
   <part name="body" element="xsd1:TutorialAuthorNameRequest"/>
</message>
```

We need to add 1 new "portType": "getTutorialAuthor".

```
<portType name="TutorialAuthorPortType">
   <operation name="GetTutorialAuthor">
      <input message="tns:GetTutorialAuthorInput"/>
      <output message="tns:GetTutorialAuthorOutput"/>
   </operation>
</portType>
```

3) To include the indicated second operation "GetTutorialAuthor", is there something else to change? If so, indicate the changes needed.

We need to add a second "binding" for the new port:

```
<binding name="TutorialAuthorSoapBinding" type="tns:TutorialAuthorPortType">
   <soap:binding style="document"
      transport="http://schemas.xmlsoap.org/soap/http"/>
   <operation name="GetTutorialAuthor">
      <soap:operation soapAction="http://Guru99.com/GetTutorialAuthor"/>
      <input> <soap:body use="literal"/> </input>
      <output> <soap:body use="literal"/> </output>
   </operation>
</binding>
```

We need to add a new type for the author name inside the Schema:

```
…
<element name="TutorialAuthorRequest">
   <complexType>
      <all>
         <element name="TutorialAuthorName" type="string"/>
      </all>
   </complexType>
</element>
```

We need to add the new port to the service:

```
<service name="TutorialService">
      <documentation>TutorialService</documentation>
   <port name="TutorialPort" binding="tns:TutorialSoapBinding">
      <soap:address location="http://Guru99.com/Tutorial"/>
   </port>
   <port name="TutorialAuthorPort" binding="tns:TutorialAuthorSoapBinding">
      <soap:address location="http://Guru99.com/TutorialAuthor"/>
   </port>
</service>
```

Other minor changes for naming consistency could be:

- Change the name of "Tutorial" port and binding.

- Change the name of the URL in the "location" of the Tutorial port.

4) According to the WSDL specification of the annex, is it possible to implement the service as REST? If not, what should be changed?

No. We should specify the binding to HTTP, and the GET method (the service does not modify the server).

Exercise 13

Annex X shows the WSDL of a service.

Reasoned and briefly answer the following questions:

1) 1.a) What is the difference between the 2 XML name spaces `xmlns:wsoap` and `xmlns:soap`? 1.b) For what purpose are they used?

1.a) `wsoap` corresponds to the WDSL standard, while `soap` corresponds to the SOAP standard.
1.b) `wsoap` is for the `code`, `protocol` and `mep` attributes defined to integrate SOAP in WSDL. `soap` is used for the binding to SOAP.

2) How many operations are specified? Describe their input and output parameters and types.

One only operation: `latestTutorialOperation`
The input parameter is a `latestTutorialRequest` of type "date"
The output parameter is a `latestTutorialResponse` of type "string".

3) What is the purpose of the element `invalidDateError` of type "string"?

It is defined in:
```
<fault name = "invalidDateFault"  element = "stns:invalidDateError"/>
```
It is the element to use when an error due to "invalid date" occurs.

4) One attribute of the first operation element is
`pattern="http://www.w3.org/ns/wsdl/in-out"`
Which are other possible values of this attribute?

The patterns in WSDL are in-out, in-only, robust in-only.

5) According to the **`binding`**, specify the Body of the HTTP Request that will request to the Server a latestTutorialOperation for today.

```
<?xml version=1.0"?>
<env:Envelope
     xmlns:env="http://www.w3.org/2003/05/soap-envelope/">
<env:Body xmlns:tns="http://jenkov.com/MyService"/>
  <tns:latestTutorialOperation>
    <tns:latestTutorialRequest>20/11/09</tns:latestTutorialRequest>
  </tns:latestTutorialOperation>
</env:Body>
```

6) We want to have a second operation “getUsersOfYesterday” (in the same “interface”). The purpose of the operation is to get a list of the users that used the service yesterday. 6.a) Reasoned and briefly define (no need to formally specify) the input and output parameters and their types. 6.b) Provide the syntax of the new version of the “interface” element.

6.a) Input parameter: No parameter is formally needed. The date (of yesterday) could be accepted. Output parameter: An array of strings or another means to identify users.

6.b) A second operation is added (the rest is kept):

```
<interface  name = "latestTutorialInterfaceREV" >
 <fault name = "invalidDateFault"  element = "stns:invalidDateError"/>
 <operation name="latestTutorialOperation"
   …
 </operation>
 <operation name="getUsersOfYesterday"
        pattern="http://www.w3.org/ns/wsdl/in-out"
        style="http://www.w3.org/ns/wsdl/style/iri"
        wsdlx:safe = "true">
   <input  messageLabel="In"  element="stns: getUsersOfYesterday Request" />
   <output messageLabel="Out" element="stns: getUsersOfYesterday Response" />
     <outfault messageLabel="Out" ref    ="tns:invalidDateFault" />
 </operation>

  </interface>
```

7) In order to add this second operation, do we need to change the bindings?

We keep the binding to SOAP of the same interface, but we add `wsoap:mep` information to a new `operation` element in the binding.

8) According to the WSDL specification of the annex, is it possible to implement the service as REST? If not, what should be changed?

No. Nevertheless, the binding is over a SOAP’s message exchange pattern `soap-response`, so the request could be easily done with REST.
To formally implement over REST, we should specify the binding to HTTP, and the GET method (the service does not modify the server).

Exercise 14

Reasoned and briefly answer the following questions:

1) Given the following XML element of a WSDL, **a)** how many “namespaces” does it include? Indicate their type (URN or URL) and their name. **b)** Is there any default schema? Which one, if any? **c)** What does the `targetNamespace` indicate?

```
<definitions name="BibcodeQuery"
   xmlns="http://schemas.xmlsoap.org/wsdl/"
   xmlns:xsd="http://www.w3.org/2001/XMLSchema"
   xmlns:http="http://schemas.xmlsoap.org/wsdl/http/"
   xmlns:mime="http://schemas.xmlsoap.org/wsdl/mime/"
```

```
xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
xmlns:tns="urn:ADSBibcodeQuery"
targetNamespace="urn:ADSBibcodeQuery">
```

a) There are 6 namespaces (or 7 if we consider the attribute "targetNamespace"). The first one in by default and of type URL. xsd, http, mime and wsdl are of type URL. tns is of type URN. (targetNamespace is also of type URN).

b) The default schema is the first one. As a detail, it is repeated and it is also referenced in wsdl.

c) Indicates that that is the "namespace" defined in this file.

2) Next, you find the definition of a data type and several messages. **a)** What is the relationship between the messages and the data type? **b)** How many different data types are used? **c)** Are there any optional element in the data type? If so, which one(s)?

```
<xsd:complexType name="SummaryResult">
   <xsd:sequence>
      <xsd:element name="missionName"  type="xsd:string"/>
      <xsd:element name="dataIdCount"  type="xsd:int"/>
   </xsd:sequence>
</xsd:complexType>

<message name="doGetSummary">
  <part name="ra"      type="xsd:double"/>
  <part name="dec"     type="xsd:double"/>
  <part name="radius"  type="xsd:double"/>
</message>

<message name="doGetSummaryResponse">
  <part name="return"  type="SummaryResult"/>
</message>
```

a) The first message, corresponding to a "request", defines 3 input parameters. The second one, a "response", is based in the type previously defined (which is a sequence of 2 elements).

b) There are 4 different data types: int, string, double and SummaryResult.

c) There are no optional elements, because the default value of minOccurs is 1, and it is not specified.

3) **a)** Assuming that the previous messages are sent with SOAP, what is the HTTP method to send the "getSummary" request? Why? **b)** What would happen if we were using REST?

a) SOAP requests are always sent with POST, since we need to send content (the SOAP message) in the HTTP body.

b) The REST request could be sent with GET or POST, depending on how we specify the sending of parameters (in the URL or in the content of the HTTP request).

Given the following fragments of a WSDL:

```
<types>
  <xs:schema
      xmlns:xs=         "http://www.w3.org/2001/XMLSchema"
      targetNamespace=  "http://jenkov.com/MyService/schema"
      xmlns=            "http://jenkov.com/MyService/schema">
    <xs:element name="latestTutorialRequest"
                type="typeLatestTutorialRequest"/>
    <xs:complexType name="typeLatestTutorialRequest">
      <xs:sequence>
        <xs:element  name="date"   type="xs:date"/>
```

```
      </xs:sequence>
    </xs:complexType>
    <xs:element name="latestTutorialResponse" type="xs:string"/>
    <xs:element name="invalidDateError" type="xs:string"/>
  </xs:schema>
</types>

<interface  name = "latestTutorialInterface" >
  <fault name = "invalidDateFault"  element = "stns:invalidDateError"/>
  <operation name="latestTutorialOperation"
          pattern="http://www.w3.org/ns/wsdl/in-out"
          style="http://www.w3.org/ns/wsdl/style/iri"
          wsdlx:safe = "true">
    <input    messageLabel="In"  element="stns:latestTutorialRequest" />
    <output   messageLabel="Out" element="stns:latestTutorialResponse" />
    <outfault messageLabel="Out" ref    ="tns:invalidDateFault" />
  </operation>
</interface>
```

4) **a)** To which WSDL version do they correspond? **b)** What is the purpose of the element `invalidDateError` of type "string"? **c)** Why the "types" element consists of an XML schema?

a) Version 2.0, because the element "interface" only exists in that version.
b) It is defined in:

```
<fault name = "invalidDateFault"  element = "stns:invalidDateError"/>
```

It is the element to use when an error due to "invalid date" occurs.
c) To allow re-use.

5) In WSDL 2.0 we might have an attribute in an operation element as
`pattern="http://www.w3.org/ns/wsdl/in-out"`
Which is the equivalent attribute in SOAP and which are its possible values?

Message Exchange Patterns (MEP): request-response, soap-response

6) We want to have a second operation "getStatus" in the previous "interface". The purpose of the operation is to check the service server. a) Reasoned and briefly define (no need to formally specify) the input and output parameters and their types. b) Provide the syntax of the new version of the "interface" element.

a) Input parameter: No parameter is formally needed.
Output parameter: A string with some message, or an integer code, or even a boolean.

b) A second operation is added (the rest is kept):

```
<interface  name = "latestTutorialInterfaceREV" >
 <fault name = "invalidDateFault"  element = "stns:invalidDateError"/>
 <operation name="latestTutorialOperation"
   …
 </operation>
 <operation name="getStatus"
        pattern="http://www.w3.org/ns/wsdl/in-out"
        style="http://www.w3.org/ns/wsdl/style/iri"
        wsdlx:safe = "true">
   <input  messageLabel="In"  element="stns:getStatusRequest" />
   <output messageLabel="Out" element="stns: getStatusResponse" />
     <outfault messageLabel="Out" ref    ="tns:invalidDateFault" />
 </operation>

  </interface>
```

6) In order to add this second operation, do we need to change the “types”? If so, which changes?

Yes, to add the new types for “getStatusRequest” (none in this solution) and “getStatusResponse”.

Problema 15

Contestar razonada y brevemente a las siguientes preguntas:

1) Dado el siguiente WSDL que nos da el valor actual de un stock a partir del nombre de una sociedad:

```
<?xml version="1.0" encoding="UTF-8"?>
<description   name="StockQuote"
               targetNamespace="http://example.com/stockquote/"
               xmlns="http://www.w3.org/ns/wsdl/"
               xmlns:tns="http://example.com/stockquote/"
               xmlns:xsd="http://www.w3.org/2001/XMLSchema">
   <types>
         <xsd:schema>
               <xsd:element name="getQuoteRequest">
                     <xsd:element name="ticker" type="xsd:string"/>
               </xsd:element>
               <xsd:element name="getQuoteResponse">
                     <xsd:element name="result" type="xsd:float"/>
               </xsd:element>
         </xsd:schema>
   </types>
   <interface  name="StockQuoteIF">
         <operation  name="getQuote"
                     pattern="http://www.w3.org/ns/wsdl/in-out">
               <input element="tns:getQuoteRequest"/>
               <output element="tns:getQuoteResponse"/>
         </operation>
   </interface>
</description>
```

a) ¿Qué versión de WSDL utiliza? **b)** ¿Qué *name spaces* corresponden a estándares? **c)** ¿Cuántos elementos tiene el elemento `operation`? ¿Cuáles son? **d)** Dar un posible valor alternativo para el atributo "pattern". Con este nuevo valor, ¿sería correcta ahora el resto de la especificación de "interface"? **e)** El elemento `description` tiene un elemento `types` y otro `interface`, ¿qué elementos faltan para completar el WSDL?

a) Versión 2.0 (es "description").

b) WSDL (por defecto) y XML schema.

c) Dos. Input y output.

d) "http://www.w3.org/ns/wsdl/robust-in-only" (o "http://www.w3.org/ns/wsdl/in-only"). Ahora no sería correcto el resto, puesto que no debería haber operación "output".

e) binding y service.

2) Para el WSDL anterior, añadir el elemento necesario para que lo podamos implementar con REST.

Debemos añadir un elemento binding. Por ejemplo:

```
<binding     name="StockQuoteBinding"
             interface="tns:StockQuoteInterface">
             type="http://www.w3.org/ns/wsdl/http"
      <operation    ref="tns:getQuote"
                    whttp:method="GET"/>
</wsdl:binding>
```

3) Queremos extender el WSDL anterior (incluyendo lo añadido en la pregunta 2) con una nueva operación **(en el mismo elemento `interface`)** que nos dé, también a partir del nombre de una sociedad, no sólo el valor de su stock sino también, **opcionalmente**, la hora exacta en la que se verificó el valor. **a)** ¿Qué cambios habrá que hacer en el WSDL? **b)** Dar una nueva versión, si cambia, del elemento `types`.

a) La operación se añade al elemento interface, por lo que allí estará el primer cambio. Debemos añadir también el nuevo tipo de salida (en types), pues el de entrada ya lo tenemos, pues es el mismo. Debemos añadir también la nueva operación en el elemento binding.

b) *Cambios resaltados en amarillo*:

```
<types>
  <xsd:schema>
    <xsd:element name="getQuotesRequest">
      <xsd:element name="ticker" type="xsd:string"/>
    </xsd:element>
    <xsd:element name="getQuoteResponse">
      <xsd:element name="result" type="xsd:float"/>
    </xsd:element>
    <xsd:element name="getQuoteAndTimeResponse">
      <xsd:complexType>
        <xsd:sequence>
          <xsd:element name="stockPrice" type="xsd:float"/>
          <xsd:element name="time" type="xsd:time" minOccurs="0" maxOccurs="1"/>
        </xsd:sequence>
      </xsd:complexType>
    </xsd:element>
  </xsd:schema>
</types>
```

4) **a)** Queremos solicitar para la sociedad XYZ la operación añadida en la pregunta anterior. Dar un ejemplo de URL a utilizar. **b)** ¿Con qué método HTTP se enviará la petición? **c)** Dar el contenido del `body` del HTTP `Request`.

a) El name space es http://example.com/stockquote/. Como hay dos operaciones diferentes con los mismos parámetros de entrada, deberíamos incluir el nombre de la operación para distinguirlas. Respecto al nombre del parámetro, no es necesario, pues sólo hay uno. Por tanto, un ejemplo podría ser:

http://example.com/stockquote/getQuoteAndTime/XYZ

b) Tal como hemos especificado en el binding del WSDL, la petición REST se debe enviar con GET.

c) Al usarse un get e ir toda la información en el HTTP Header, no hay body.

Exercise 16

Reasoned and briefly answer the following questions:

1) Given the following definition of a data type and several messages. **a)** What is the relationship between the messages and the data type? **b)** Are there any optional element in the data type? If so, which one(s)?

```
<xsd:complexType name="abc">
   <xsd:sequence>
      <xsd:element name="abc-def"  type="xsd:string"/>
      <xsd:element name="abc-ghi"  type="xsd:int"/>
   </xsd:sequence>
</xsd:complexType>

<message name="getData">
  <part name="a"   type="xsd:double"/>
  <part name="b"   type="xsd:string"/>
  <part name="c"   type="xsd:int"/>
</message>

<message name="getDataResponse">
  <part name="z"  type="abc"/>
</message>
```

a) The first message defines 3 parameters. The second one is based in the type previously defined (which is a sequence of 2 elements).

b) There are no optional elements, because the default value of minOccurs is 1, and it is not specified.

2) **a)** Assuming that the previous messages are sent with SOAP, what is the HTTP method to send the “getData” request? Why? **b)** What would happen if we were using REST?

a) SOAP requests are always sent with POST, since we need to send content (the SOAP message) in the HTTP body.

b) The REST request could be sent with GET or POST, depending on how we specify the sending of parameters (in the URL or in the content of the HTTP request).

Given the following fragments of a WSDL:

```
<types>
  <xs:schema
      xmlns:xs=         "http://www.w3.org/2001/XMLSchema"
      targetNamespace= "http://jenkov.com/MyService/schema"
      xmlns=           "http://jenkov.com/MyService/schema">
    <xs:element name="latestTutorialRequest"
                  type="typeLatestTutorialRequest"/>
    <xs:complexType name="typeLatestTutorialRequest">
      <xs:sequence>
        <xs:element  name="date" type="xs:date"/>
      </xs:sequence>
    </xs:complexType>
    <xs:element name="latestTutorialResponse" type="xs:string"/>
    <xs:element name="invalidDateError" type="xs:string"/>
  </xs:schema>
</types>

<interface  name="latestTutorialInterface" >
  <fault name="invalidDateFault"  element="stns:invalidDateError"/>
  <operation name="latestTutorialOperation"
          pattern="http://www.w3.org/ns/wsdl/in-out"
          style="http://www.w3.org/ns/wsdl/style/iri"
          wsdlx:safe="true">
    <input    messageLabel="In"  element="stns:latestTutorialRequest" />
    <output   messageLabel="Out" element="stns:latestTutorialResponse" />
    <outfault messageLabel="Out" ref="tns:invalidDateFault" />
  </operation>
</interface>
```

3) **a)** What is the purpose of the XML schema? **b)** How many, and with which identifiers, input and output elements are specified? **c)** What is the purpose of the “outfault” element?

d) It specifies the types used in the operations, in such a way that they could be re-used.

e) There is one input element (with message label “In”), one regular output element (with message label “Out”), and, for that output element, an “outfault” element to handle errors, as defined in:

```
<fault name = "invalidDateFault"  element = "stns:invalidDateError"/>
```

f) It is the element to use when an error due to “invalid date” occurs.

4) We want to have a second operation “getStatus” in the previous “interface”. The purpose of the operation is to check the status of the service server. **a)** Reasoned and briefly define (no need to formally specify) the input and output parameters and their types. **b)** Where in the WSDL is the new operation to be specified? **c)** Do we need to change the “types” schema? If so, which changes?

a) Input parameter: No parameter is formally needed.
Output parameter: A string with some message, or an integer code, or even a boolean.

b) The second operation is added in the "interface" element:

```
<interface  name="latestTutorialInterfaceREV" >
 <fault … />
 <operation name="latestTutorialOperation"
   …
 </operation>
 <operation name="getStatus"
             … >
   …
 </operation>
  </interface>
```

c) Yes, to add the new types for "getStatusRequest" (none in this solution) and "getStatusResponse".

ANNEX I. Especificación en WSDL de un servicio "Búsqueda de vídeos por año" *(Problema 1)*

```
<?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="searchVideosByYear">
      <http:operation location="searchVideosByYear" />
      <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>
```

ANNEX II. Ejemplo de WSDL (part of *the GreatH Web Service*) *(Problemas 2 y 3)*
(Aclaraciones al final)

```
<?xml version="1.0" encoding="utf-8" ?>
<description
    xmlns="http://www.w3.org/ns/wsdl"
    targetNamespace= "http://greath.example.com/2004/wsdl/resSvc"
    xmlns:tns= "http://greath.example.com/2004/wsdl/resSvc"
    xmlns:ghns = "http://greath.example.com/2004/schemas/resSvc"
    xmlns:wsoap= "http://www.w3.org/ns/wsdl/soap"
    xmlns:soap="http://www.w3.org/2003/05/soap-envelope"
    xmlns:wsdlx= "http://www.w3.org/ns/wsdl-extensions">

  <types>
    <xs:schema
        xmlns:xs="http://www.w3.org/2001/XMLSchema"
        targetNamespace="http://greath.example.com/2004/schemas/resSvc"
        xmlns="http://greath.example.com/2004/schemas/resSvc">

      <xs:element name="checkAvailability" type="tCheckAvailability"/>

      <xs:complexType name="tCheckAvailability">
        <xs:sequence>
          <xs:element  name="checkInDate" type="xs:date"/>
          <xs:element  name="checkOutDate" type="xs:date"/>
          <xs:element  name="roomType" type="xs:string"/>
        </xs:sequence>
      </xs:complexType>

      <xs:element name="checkAvailabilityResponse" type="xs:double"/>

      <xs:element name="invalidDataError" type="xs:string"/>

    </xs:schema>

  </types>

  <interface  name = "reservationInterface" >

    <fault name = "invalidDataFault"
            element = "ghns:invalidDataError"/>
```

```
    <operation name="opCheckAvailability"
            pattern="http://www.w3.org/ns/wsdl/in-out"
            style="http://www.w3.org/ns/wsdl/style/iri"
            wsdlx:safe = "true">
        <input messageLabel="In"
              element="ghns:checkAvailability" />
        <output messageLabel="Out"
              element="ghns:checkAvailabilityResponse" />
        <outfault ref="tns:invalidDataFault" messageLabel="Out"/>
    </operation>

  </interface>

  <binding name="reservationSOAPBinding"
          interface="tns:reservationInterface"
          type="http://www.w3.org/ns/wsdl/soap"
          wsoap:protocol="http://www.w3.org/2003/05/soap/bindings/HTTP/">

    <fault ref="tns:invalidDataFault"
      wsoap:code="soap:Sender"/>

    <operation ref="tns:opCheckAvailability"
      wsoap:mep="http://www.w3.org/2003/05/soap/mep/soap-response"/>

  </binding>

  <service name="reservationService"
       interface="tns:reservationInterface">

     <endpoint name="reservationEndpoint"
               binding="tns:reservationSOAPBinding"
               address ="http://greath.example.com/2004/reservation"/>

  </service>

</description>
```

- Aclaraciones sobre los atributos de `fault`:

`<fault name = "invalidDataFault"`

The `name` attribute defines a name for this fault. The name is required so that when an operation is defined, it can reference the desired fault by name. Fault names must be unique within an interface.

`element = "ghns:invalidDataError"/>`

The `element` attribute specifies the schema type of the fault message, as previously defined in the `types` section.

- Aclaraciones sobre atributos y elementos de `operation`:

`wsdlx:safe="true" >`

This line indicates that this operation will not obligate the client in any way, i.e., the client can safely invoke this operation without fear that it may be incurring an obligation (such as agreeing to buy something).

`<outfault ref="tns:invalidDataFault" messageLabel="Out"/>`

This associates an output fault with this operation. Faults are declared a little differently than normal messages. The `ref` attribute refers to the name of a previously defined fault in this interface -- not a message schema type directly. Since message exchange patterns could in general involve a sequence of several messages, a fault could potentially occur at various points within the message sequence. Because one may wish to associate a different fault with each permitted point in the sequence, the `messageLabel` is used to indicate the desired point for this particular fault. It does so indirectly by specifying the message that will either trigger this fault or that this fault will replace, depending on the pattern.

ANNEX III. Example of the WSDL "vuelos libres" *(Exercise 4)*

```
<wsdl:definitions … >
<wsdl:types>
  <schema elementFormDefault="qualified" targetNamespace="http://vueloWS"
          xmlns="http://www.w3.org/2001/XMLSchema">
   <element name="consulta_libres">
    <complexType>
     <sequence>
      <element name="id_vuelo" type="xsd:int"/>
      <element name="fecha" type="xsd:int"/>
     </sequence>
    </complexType>
   </element>
   <element name="consulta_libresResponse">
    <complexType>
     <sequence>
      <element name="consulta_libresReturn" type="xsd:int"/>
     </sequence>
    </complexType>
   </element>
  </schema>
 </wsdl:types>

  <wsdl:message name="consulta_libresResponse">
      <wsdl:part element="impl:consulta_libresResponse" name="parameters"/>
   </wsdl:message>

   <wsdl:message name="consulta_libresRequest">
      <wsdl:part element="impl:consulta_libres" name="parameters"/>
   </wsdl:message>

   <wsdl:portType name="vuelo">
      <wsdl:operation name="consulta_libres">
         <wsdl:input message="impl:consulta_libresRequest"
                     name="consulta_libresRequest">
       </wsdl:input>
       <wsdl:output message="impl:consulta_libresResponse"
                    name="consulta_libresResponse">
       </wsdl:output>
      </wsdl:operation>
   </wsdl:portType>

   <wsdl:binding name="vueloSoapBinding" type="impl:vuelo">
      <wsdlsoap:binding style="document"
                        transport="http://schemas.xmlsoap.org/soap/http"/>
      <wsdl:operation name="consulta_libres">
         <wsdlsoap:operation soapAction=""/>
         <wsdl:input name="consulta_libresRequest">
            <wsdlsoap:body use="literal"/>
         </wsdl:input>
         <wsdl:output name="consulta_libresResponse">
            <wsdlsoap:body use="literal"/>
         </wsdl:output>
      </wsdl:operation>
   </wsdl:binding>

   <wsdl:service name="vueloService">
      <wsdl:port binding="impl:vueloSoapBinding" name="vuelo">
         <wsdlsoap:address location="http://localhost:8080/vueloWS/services/vuelo"/>
      </wsdl:port>
   </wsdl:service>
</wsdl:definitions>
```

ANNEX IV. Bookstore WSDL (http://www.ibm.com/developerworks/library/ws-restwsdl/) *(Exercises 5, 6, 8, 10, 11)*

```
<?xml version="1.0" encoding="UTF-8"?>
<wsdl:description xmlns:wsdl="http://www.w3.org/ns/wsdl"
   targetNamespace="http://www.bookstore.org/booklist/wsdl"
   xmlns:tns="http://www.bookstore.org/booklist/wsdl"
   xmlns:whttp="http://www.w3.org/ns/wsdl/http"
   xmlns:wsdlx="http://www.w3.org/ns/wsdl-extensions"
   xmlns:xs="http://www.w3.org/2001/XMLSchema"
   xmlns:msg="http://www.bookstore.org/booklist/xsd">
  <wsdl:documentation>
     This is a WSDL description of a sample bookstore service listing for obtaining
book information.
  </wsdl:documentation>

  <wsdl:types>
    <xs:import namespace="http://www.bookstore.org/booklist/xsd"
       schemaLocation="booklist.xsd"/>
  </wsdl:types>

  <wsdl:interface name="BookListInterface">
    <wsdl:operation name="getBookList"
       pattern="http://www.w3.org/ns/wsdl/in-out"
       style="http://www.w3.org/ns/wsdl/style/iri"
       wsdlx:safe="true">
      <wsdl:documentation>
         This operation returns a list of books.
      </wsdl:documentation>
      <wsdl:input element="msg:getBookList"/>
      <wsdl:output element="msg:bookList"/>
    </wsdl:operation>
  </wsdl:interface>

  <wsdl:binding name="BookListHTTPBinding"
     type="http://www.w3.org/ns/wsdl/http"
     interface="tns:BookListInterface">
    <wsdl:documentation>
       The HTTP binding for the book list service.
    </wsdl:documentation>
    <wsdl:operation ref="tns:getBookList" whttp:method="GET"/>
  </wsdl:binding>

  <wsdl:service name="BookList" interface="tns:BookListInterface">
    <wsdl:documentation>
       The bookstore's book list service.
    </wsdl:documentation>
    <wsdl:endpoint name="BookListHTTPEndpoint"
       binding="tns:BookListHTTPBinding"
       address="http://www.bookstore.com/books/">
    </wsdl:endpoint>
  </wsdl:service>
</wsdl:description>
```

ANNEX V. booklist.xsd (http://www.ibm.com/developerworks/library/ws-restwsdl/) *(Exercises 5, 6, 7, 11)*

```
<?xml version="1.0" encoding="UTF-8"?>
<schema xmlns="http://www.w3.org/2001/XMLSchema"
   targetNamespace="http://www.bookstore.org/booklist/xsd"
   xmlns:tns="http://www.bookstore.org/booklist/xsd"
   xmlns:booksvc="http://www.bookstore.org/book/wsdl"
   xmlns:wsdlx="http://www.w3.org/ns/wsdl-extensions"
   elementFormDefault="qualified">

  <element name="getBookList" type="tns:getBookListType">
    <annotation> <documentation>
         This request element for the book list service.
   </documentation>  </annotation>
  </element>

  <element name="bookList" type="tns:bookListType">
    <annotation> <documentation>
         The response element for the book list service.
    </documentation> </annotation>
  </element>

  <complexType name="getBookListType">
    <sequence>
      <element name="author" type="string" minOccurs="0" maxOccurs="unbounded"/>
      <element name="title" type="string" minOccurs="0" maxOccurs="1"/>
      <element name="publisher" type="string" minOccurs="0" maxOccurs="1"/>
      <element name="subject" type="string" minOccurs="0" maxOccurs="1"/>
      <element name="language" type="string" minOccurs="0" maxOccurs="unbounded"/>
    </sequence>
  </complexType>

  <complexType name="bookListType">
    <sequence>
      <element name="book" type="tns:bookType" minOccurs="0" maxOccurs="unbounded"/>
    </sequence>
  </complexType>

  <complexType name="bookType">
    <attribute name="title" type="string"/>
    <attribute name="url" type="anyURI"
       wsdlx:interface="booksvc:BookInterface"
       wsdlx:binding="booksvc:BookHTTPBinding"/>
  </complexType>
</schema>
```

ANNEX VI wsdlx Schema (http://www.w3.org/ns/wsdl-extensions/) *(Exercises 5 and 6)*

```
<xs:schema targetNamespace="http://www.w3.org/ns/wsdl-extensions"
attributeFormDefault="qualified" elementFormDefault="qualified"
finalDefault="" blockDefault="">

  <xs:attribute name="safe" type="xs:boolean">
…
  </xs:attribute>

  <xs:attribute name="interface" type="xs:QName">
    <xs:annotation> <xs:documentation>
This attribute may be used to annotate element or attribute definitions to indicate
that the content refers to Web service that implements the specified interface.
    </xs:documentation> </xs:annotation>
  </xs:attribute>

  <xs:attribute name="binding" type="xs:QName">
    <xs:annotation> <xs:documentation>
  This attribute may be used to annotate element or attribute definitions to indicate
that the content refers to Web service that implements the specified binding.
    </xs:documentation> </xs:annotation>
  </xs:attribute>

</xs:schema>
```

ANNEX VII. BookList WSDL (http://www.ibm.com/developerworks/library/ws-restwsdl/) *(Exercise 7)*

```
<wsdl:description xmlns:wsdl="http://www.w3.org/ns/wsdl"
   targetNamespace="http://www.bookstore.org/booklist/wsdl"
   xmlns:tns="http://www.bookstore.org/booklist/wsdl"
   xmlns:whttp="http://www.w3.org/ns/wsdl/http"
   xmlns:wsdlx="http://www.w3.org/ns/wsdl-extensions"
   xmlns:xs="http://www.w3.org/2001/XMLSchema"
   xmlns:msg="http://www.bookstore.org/booklist/xsd">
 <wsdl:documentation>
     This is a WSDL description of a sample bookstore service listing for obtaining
book information.
  </wsdl:documentation>

  <wsdl:types>
    <xs:import namespace="http://www.bookstore.org/booklist/xsd"
       schemaLocation="booklist.xsd"/>
  </wsdl:types>

  <wsdl:interface name="BookListInterface">
    <wsdl:operation name="getBookList"
       pattern="http://www.w3.org/ns/wsdl/in-out"
       style="http://www.w3.org/ns/wsdl/style/iri"
       wsdlx:safe="true">
      <wsdl:documentation>
         This operation returns a list of books.
      </wsdl:documentation>
      <wsdl:input element="msg:getBookList"/>
      <wsdl:output element="msg:bookList"/>
    </wsdl:operation>
  </wsdl:interface>

  <wsdl:binding name="BookListHTTPBinding"
     type="http://www.w3.org/ns/wsdl/http"
     interface="tns:BookListInterface">
    <wsdl:documentation>
       The binding for the book list service.
    </wsdl:documentation>
    <wsdl:operation ref="tns:getBookList" whttp:method="GET"/>
  </wsdl:binding>

  <wsdl:service name="BookList" interface="tns:BookListInterface">
    <wsdl:documentation>
       The bookstore's book list service.
    </wsdl:documentation>
    <wsdl:endpoint name="BookListHTTPEndpoint"
       binding="tns:BookListHTTPBinding"
       address="http://www.bookstore.com/books/">
    </wsdl:endpoint>
  </wsdl:service>
</wsdl:description>
```

ANNEX VIII. HelloService WSDL (https://www.tutorialspoint.com/wsdl/wsdl_example.htm) *(Exercises 8 and 9)*

```
<definitions name="HelloService"
   targetNamespace="http://www.examples.com/wsdl/HelloService.wsdl"
   xmlns="http://schemas.xmlsoap.org/wsdl/"
   xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
   xmlns:tns="http://www.examples.com/wsdl/HelloService.wsdl"
   xmlns:xsd="http://www.w3.org/2001/XMLSchema">

   <message name="SayHelloRequest">
      <part name="firstName" type="xsd:string"/>
   </message>

   <message name="SayHelloResponse">
      <part name="greeting" type="xsd:string"/>
   </message>

   <portType name="Hello_PortType">
      <operation name="sayHello">
         <input message="tns:SayHelloRequest"/>
         <output message="tns:SayHelloResponse"/>
      </operation>
   </portType>

   <binding name="Hello_Binding" type="tns:Hello_PortType">
      <soap:binding style="rpc"
         transport="http://schemas.xmlsoap.org/soap/http"/>
      <operation name="sayHello">
         <soap:operation soapAction="sayHello"/>
         <input>
            <soap:body
               encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
               namespace="urn:examples:helloservice"
               use="encoded"/>
         </input>

         <output>
            <soap:body
               encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
               namespace="urn:examples:helloservice"
               use="encoded"/>
         </output>
      </operation>
   </binding>

   <service name="Hello_Service">
      <documentation>WSDL File for HelloService</documentation>
      <port binding="tns:Hello_Binding" name="Hello_Port">
         <soap:address
            location="http://www.examples.com/SayHello/" />
      </port>
   </service>
</definitions>
```

ANNEX IX. Tutorial Service in WSDL (https://www.guru99.com/wsdl-web-services-description-language.html)

(Exercise 12)

```
<?xml version="1.0"?>
<definitions name="Tutorial"
       targetNamespace=http://Guru99.com/Tutorial.wsdl
       xmlns:tns=http://Guru99.com/Tutorial.wsdl
       xmlns:xsd1=http://Guru99.com/Tutorial.xsd
       xmlns:soap=http://schemas.xmlsoap.org/wsdl/soap/
       xmlns="http://schemas.xmlsoap.org/wsdl/">

   <types>
      <schema targetNamespace=http://Guru99.com/Tutorial.xsd
        xmlns="http://www.w3.org/2000/10/XMLSchema">
         <element name="TutorialNameRequest">
            <complexType>
               <all>
                  <element name="TutorialName" type="string"/>
               </all>
            </complexType>
         </element>
         <element name="TutorialIDRequest">
            <complexType>
               <all>
                  <element name="TutorialID" type="number"/>
               </all>
            </complexType>
         </element>
      </schema>
   </types>

   <message name="GetTutorialNameInput">
      <part name="body" element="xsd1:TutorialIDRequest"/>
   </message>
   <message name="GetTutorialNameOutput">
      <part name="body" element="xsd1:TutorialNameRequest"/>
   </message>

   <portType name="TutorialPortType">
      <operation name="GetTutorialName">
         <input message="tns:GetTutorialNameInput"/>
         <output message="tns:GetTutorialNameOutput"/>
      </operation>
   </portType>

   <binding name="TutorialSoapBinding" type="tns:TutorialPortType">
      <soap:binding style="document"
         transport="http://schemas.xmlsoap.org/soap/http"/>
      <operation name="GetTutorialName">
         <soap:operation soapAction="http://Guru99.com/GetTutorialName"/>
         <input> <soap:body use="literal"/> </input>
         <output> <soap:body use="literal"/> </output>
      </operation>
   </binding>

 <service name="TutorialService">
       <documentation>TutorialService</documentation>
    <port name="TutorialPort" binding="tns:TutorialSoapBinding">
       <soap:address location="http://Guru99.com/Tutorial"/>
    </port>
 </service>
</definitions>
```

ANNEX X. Service in WSDL (http://tutorials.jenkov.com/wsdl/overview.html#full-wsdl-example) *(Exercise 13)*

```
<?xml version="1.0" encoding="utf-8" ?>
<description
    xmlns=           "http://www.w3.org/ns/wsdl"
    targetNamespace= "http://jenkov.com/MyService"
    xmlns:tns=       "http://jenkov.com/MyService"
    xmlns:stns =     "http://jenkov.com/MyService/schema"
    xmlns:wsoap=     "http://www.w3.org/ns/wsdl/soap"
    xmlns:soap=      "http://www.w3.org/2003/05/soap-envelope"
    xmlns:wsdlx=     "http://www.w3.org/ns/wsdl-extensions" >

  <types>
    <xs:schema
        xmlns:xs=        "http://www.w3.org/2001/XMLSchema"
        targetNamespace= "http://jenkov.com/MyService/schema"
        xmlns=           "http://jenkov.com/MyService/schema">
      <xs:element name="latestTutorialRequest"
                  type="typeLatestTutorialRequest"/>
      <xs:complexType name="typeLatestTutorialRequest">
        <xs:sequence>
          <xs:element  name="date"   type="xs:date"/>
        </xs:sequence>
      </xs:complexType>
      <xs:element name="latestTutorialResponse" type="xs:string"/>
      <xs:element name="invalidDateError" type="xs:string"/>
    </xs:schema>
  </types>

  <interface  name = "latestTutorialInterface" >
    <fault name = "invalidDateFault"  element = "stns:invalidDateError"/>
    <operation name="latestTutorialOperation"
            pattern="http://www.w3.org/ns/wsdl/in-out"
            style="http://www.w3.org/ns/wsdl/style/iri"
            wsdlx:safe = "true">
      <input    messageLabel="In"  element="stns:latestTutorialRequest" />
      <output   messageLabel="Out" element="stns:latestTutorialResponse" />
      <outfault messageLabel="Out" ref    ="tns:invalidDateFault" />
    </operation>
  </interface>

  <binding name="latestTutorialSOAPBinding"
          interface="tns:latestTutorialInterface"
          type="http://www.w3.org/ns/wsdl/soap"
          wsoap:protocol="http://www.w3.org/2003/05/soap/bindings/HTTP/">
    <fault ref="tns:invalidDateFault" wsoap:code="soap:Sender"/>
    <operation ref="tns:latestTutorialOperation"
      wsoap:mep="http://www.w3.org/2003/05/soap/mep/soap-response"/>
  </binding>

  <service
       name     ="latestTutorialService"
       interface="tns:latestTutorialInterface">
     <endpoint name ="latestTutorialEndpoint"
            binding ="tns:latestTutorialSOAPBinding"
            address ="http://jenkov.com/latestTutorial"/>
  </service>

</description>
```