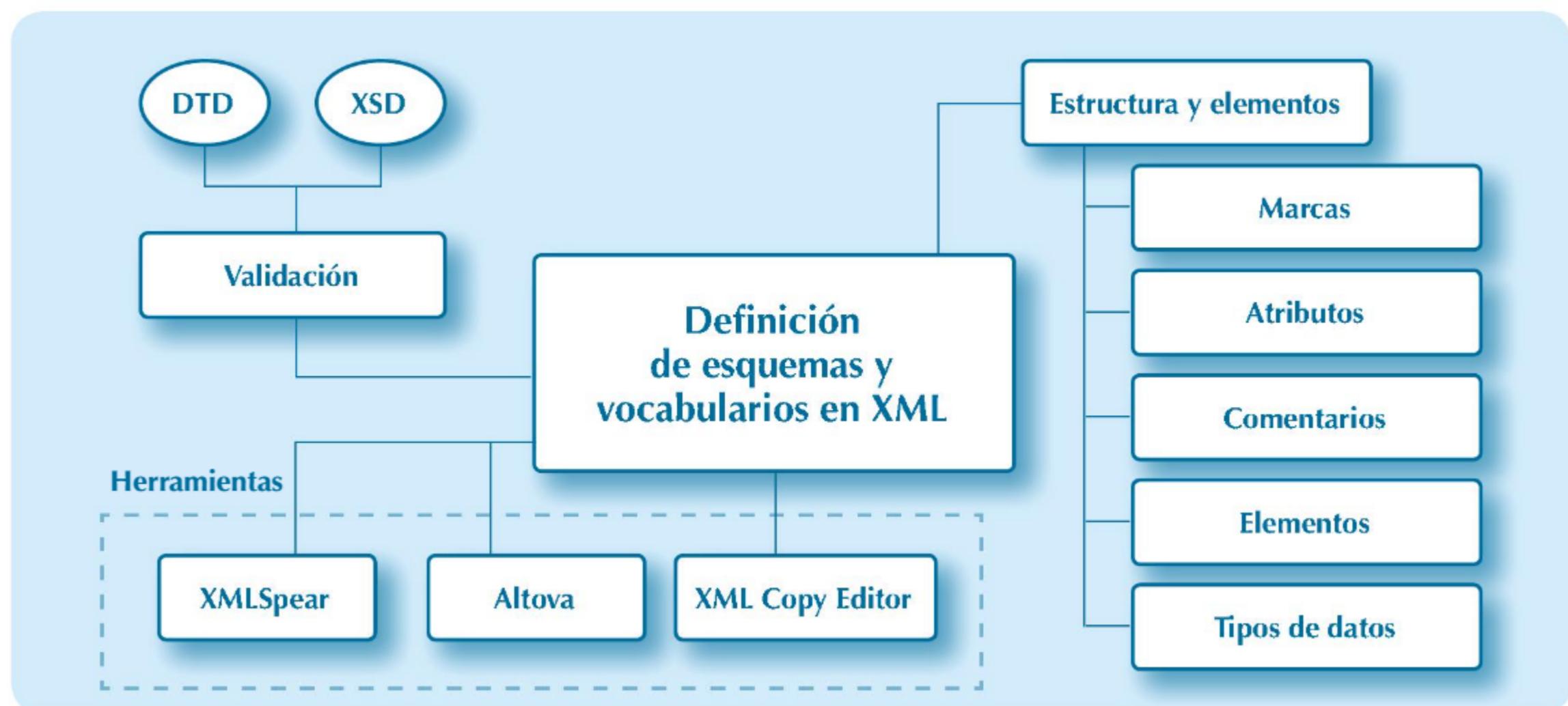


Definición de esquemas y vocabularios en XML

Objetivos

- ✓ Introducirte en el manejo de XML, analizando, creando, definiendo y profundizando en su estructura y sintaxis.
- ✓ La creación de definiciones de tipos de documentos y esquemas es el siguiente aspecto que vas a estudiar, analizando su sintaxis y estructura, así como viendo las técnicas de asociación entre XML y DTD o XSD.
- ✓ Es importante que aprendas a analizar las herramientas específicas que permiten trabajar con XML, así como validar documentos, crear sus esquemas y comprobar si están bien formados, así como trabajar con características más avanzadas.
- ✓ También es fundamental que sepas cómo validar mediante herramientas *online* o de escritorio los XML, asociando la definición de tipo de documento o esquemas.

Mapa conceptual



Glosario

JSON. Sigla de *JavaScript object notation*. Formato de texto ligero que se usa para el intercambio de datos de fácil interpretación.

Schematron. Lenguaje de validación XML inventado por Rich Jelliffe.

SGML. Sigla de *standard generalized markup language*. Estándar para lenguajes de marcado.

SOAP. Sigla de *simple object access protocol*. Protocolo de comunicación entre objetos mediante internet. Basado en XML.

Solaris. Sistema operativo Unix propiedad de Oracle para arquitecturas SPARC y x86.

WSDL. Sigla de *web services description language*. Lenguaje de descripción de servicios web. Se basa en XML.

XPath. Sigla de *XML Path Language*. Lenguaje usado para procesar documentos XML.

XQuery. Lenguaje de consultas de datos XML.

XSLT. También conocido como *transformaciones XSL (extensible stylesheet language)*, es un lenguaje que se usa para transformar documentos XML en HTML o XHTML.

4.1. Introducción

En el presente capítulo, se estudiarán aspectos relacionados con XML (*extensible markup language*), tecnología que sirve como formato de almacenamiento e intercambio de información.

Se usa para almacenamiento de datos, actualización de *software*, intercambio de información, sindicación de contenidos, etc.

Se verá su estructura y sintaxis, instrucciones de procesamiento, marcas, elementos y atributos, así como otros aspectos relevantes. Verificar que están escritos correctamente y son válidos son aspectos fundamentales a la hora de trabajar con estos documentos, ya que la validación comprueba que está bien formado siguiendo una serie de reglas y respetando, además, las normas que pueden encontrarse en la DTD (*document type definition*) o XSD (*XML schema definition*).

XML deriva de SGML (*standard generalized markup language*), siendo legible y comprensible para todos, lo que permite que usuarios sin conocimientos de lenguajes de marcas sean capaces de interpretar qué información es la que se almacena en los documentos. Mientras que las marcas de un documento HTML no aportan información relevante a los usuarios, las de XML transmiten el significado del contenido que se almacena.

Tanto para la creación como para la validación y edición, existen multitud de herramientas gratuitas, comerciales y de evaluación que permiten trabajar de manera fácil y amena con este tipo de documentos. Algunas de ellos, además de facilitar la construcción de documentos XML, permiten generar esquemas y validar y comprobar que están bien formados, así como trabajar con XSLT (*extensible stylesheet language*), XQuery, etc.

Gracias a las DTD o XSD, pueden definirse las restricciones y tipos de los atributos y elementos, así como el número de veces que aparecen o los posibles valores por defecto que pueden tomar. Estos esquemas facilitan el trabajo colaborativo al poder trabajar de manera conjunta conociendo su esquema, lo que permite, además, la validación de los documentos XML.

Aunque un documento DTD es más fácil de crear que uno XSD y siguen creándose en la actualidad, presenta una serie de limitaciones: no permite especificar restricciones como rangos concretos, número de caracteres que deben aparecer, expresiones regulares, etc. No es posible especificar tipos de datos fecha o entero positivo, no está basado en XML, no comprueba los tipos, pero puede añadirse cualquier valor en un elemento llamado *DNI*, tal y como se muestra en el ejemplo siguiente.

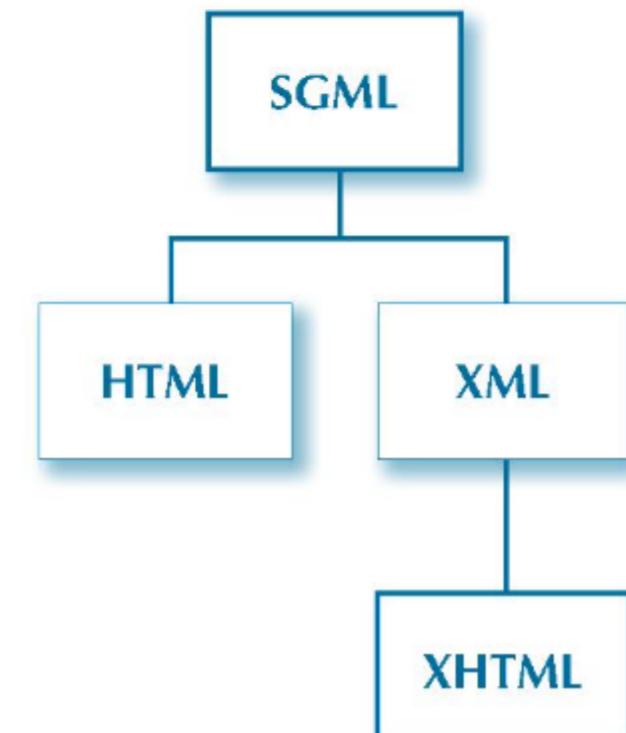


Figura 4.1
Jerarquía SGML.

```

<?xml version="1.0" encoding="iso-8859-1"?>
<usuario>
  <fechaNac>2018-10-25</fechaNac>
  <DNI>fran@librolm.com</DNI>
  <nombre>Francisco Jesús González</nombre>
</usuario>
  
```

4.2. Definición de la estructura y sintaxis de documentos XML

Tal y como se mencionó en el capítulo 1, XML es un lenguaje de marcas extensible cuyas siglas vienen de la expresión inglesa *extensible markup language*. Se considera un metalenguaje y

surge para resolver los problemas que plantea HTML, pues las etiquetas muestran el significado de sus datos.

Para poder visualizar un documento XML, puede hacerse directamente sobre el navegador, usando hojas de estilo CSS, transformaciones XSLT, etc.

La importancia radica en el intercambio de información de manera segura entre distintos programas, permitiendo así la reutilización de contenido, crear etiquetas propias y presentar una estructura y diseño independientes.



SABÍAS QUE...

Un documento XML debe crearse en texto plano.

La estructura de un documento XML es jerárquica arborescente, que contiene a un padre (raíz) único y una serie de (hijos) elementos secundarios. La raíz se sitúa en la parte superior, colgando el resto de elementos de él.

La extensión de este tipo de documentos es .xml.

```
<?xml version="1.0" encoding="utf-8"?>
<padre>
  <hijo>
    <subhijos> ... </subhijos>
  </hijo>
</padre>
```

En todo documento, se distinguen dos partes claramente diferenciadas:

1. Por un lado, se encuentra el *prüfago*, que contiene información respecto al documento creado (versión, codificación de caracteres, descripción de estructura, etc.).
2. Por el otro, se encuentra el *elemento raíz o cuerpo del documento*, que tiene que ser único y sobre el que están contenidos los demás elementos.



PARA SABER MÁS

Un elemento raíz no tiene ascendientes ni hermanos.

Ejercicio propuesto 4.1

Busca por internet un documento XML y analiza su estructura.



4.2.1. Instrucciones de procesamiento o prólogo

La primera línea por la que deben empezar los documentos XML es el prólogo, compuesto por la *versión* del documento, que indica la versión XML usada, *encoding* que informa de la codificación del documento, con posibles valores (UTF-8, ISO 8859-1, etc.), y puede ser modificado en cualquier momento. Otro posible atributo que puede contener dicha línea es *standalone*, que informa de si el documento lleva asociado un *fichero DTD o XSD*, con valores posibles “yes” o “no”.

Dichas instrucciones comienzan por los caracteres <? y terminan con los caracteres ?>. En la segunda línea, se encuentra la raíz (<árbol>).

En el caso de asociar una DTD a un XML, la segunda línea sería la definición del tipo de documento.

En el ejemplo siguiente se aprecia un XML que contiene valores sobre especies de árboles, almacenando información como nombre, variedad, origen, etc.

```
<?xml version="1.0" encoding="iso-8859-1"?>
<árbol>
<especie>
    <nombre>Litchi</nombre>
    <variedad>Kway May Pink</variedad>
    <origen>Filipinas</origen>
    <color_fruto>rojo</color_fruto>
    <maduración>agosto</maduración>
    <riego>diario</riego>
    <precio>3.5</precio>
</especie>
</árbol>
```

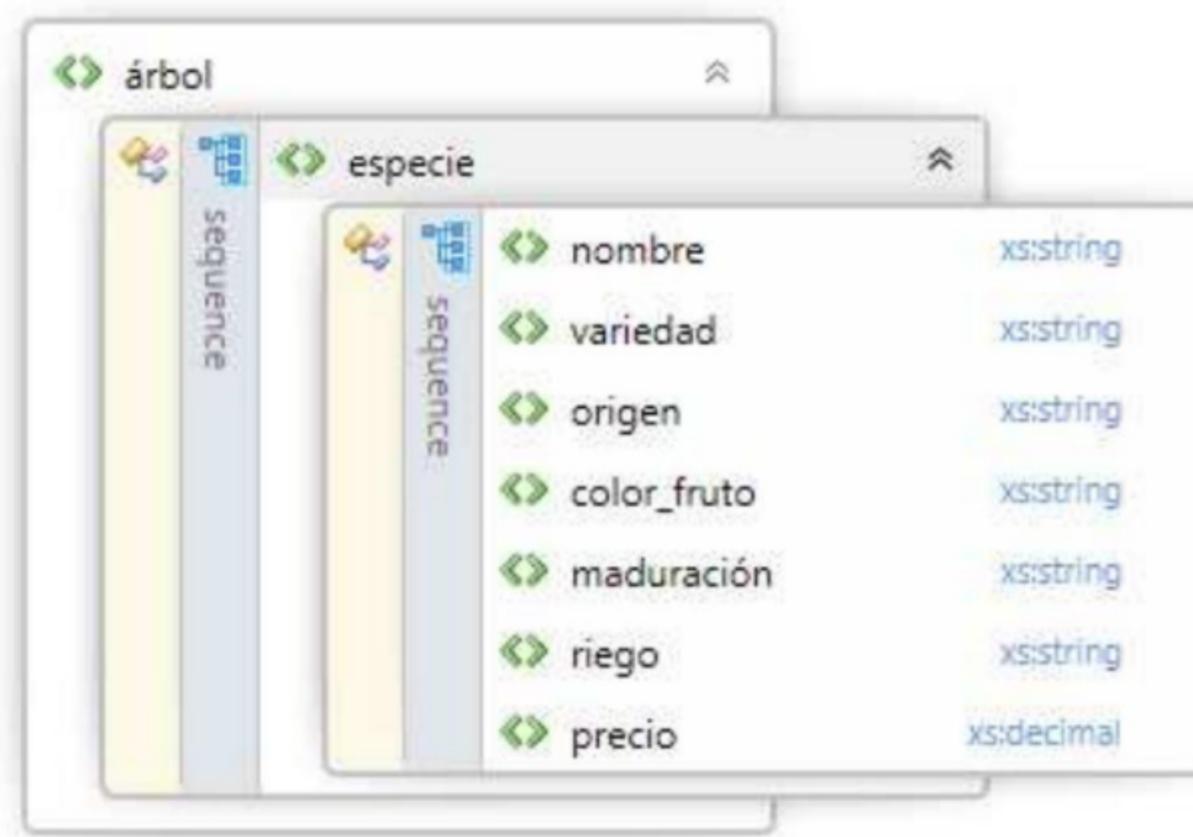


Figura 4.2
Estructura de árbol del documento XML usando Visual Studio.

Ejercicio propuesto 4.2

Añade dos especies más al ejemplo del documento XML anterior.





Ejercicio resuelto 4.1

Dado el siguiente XML, usa otro programa que permita visualizar el contenido del XML en forma de árbol:

```
<?xml version="1.0" encoding="iso-8859-1"?>
<árbol>
  <especie>
    <nombre>Litchi</nombre>
    <variedad>Kway May Pink</variedad>
    <origen>Filipinas</origen>
    <color_fruto>rojo</color_fruto>
    <maduración>agosto</maduración>
    <riego>diario</riego>
    <precio>3.5</precio>
  </especie>
</árbol>
```

Solución



4.2.2. Marcas o etiquetas

En un documento XML, el usuario puede definir sus propias marcas o etiquetas con la finalidad de guardar la información de la manera que estime oportuna, separando el contenido de la estructura.

Las marcas son el pilar fundamental de estos lenguajes. Se usan los signos menor que (<) y mayor que (>) para definirlas. Los nombres de las marcas deben describir los datos que contienen. Cabe destacar que, en XML, no existe un número finito de marcas, sino que, para poder realizar una representación de datos adecuada, cada usuario puede definir las marcas que estime oportunas.



SABÍAS QUE...

Las marcas no pueden contener espacio en blanco, comillas simples o dobles, punto y coma, signo de porcentaje, etc.

Existen tres tipos de etiquetas o marcas:

1. Las de apertura como <casa>.
2. Las de cierre como </casa>, que usan el carácter barra (/) para indicar fin de marcado.
3. Etiquetas vacías, siendo estas últimas las que carecen de contenido como <casa/>.

Viendo el ejemplo siguiente, las marcas pueden ser las siguientes:

- De apertura <nombre> <variedad>.
- De cierre </nombre> </variedad>.

```
<nombre>Litchi</nombre>
<variedad>Kway May Pink</variedad>
```

A continuación se observan ejemplos de marcas no válidas, la primera de ellas al contener un espacio en blanco y la segunda, una barra (/).

```
<nombre producto>Litchi</nombre producto>
<variedad/precio>Kway May Pink</variedad/precio>
```

Las marcas son sensibles a las mayúsculas y minúsculas. Para XML de apertura de la etiqueta <casa> y de cierre </Casa>, son distintas, ya que una empieza por mayúsculas y otra por minúsculas.



SABÍAS QUE...

Case sensitive es una expresión que se aplica a los textos en informática para indicar que no es lo mismo escribir un carácter en mayúsculas que en minúsculas.



Actividades propuestas

Responde si las siguientes afirmaciones son verdaderas o falsas y razona tu respuesta:

V F 4.1. ¿Es XML un *case sensitive*?

V F 4.2. XML deriva de XHTML.

Verificar



4.2.3. Elementos

Compuestos por etiquetas de inicio, de fin, escritas con el mismo nombre y todo el contenido que haya entre ambas (ya sea texto u otros elementos). En el cuadro 4.6, un elemento puede ser <nombre>Litchi</nombre>, cuyas partes son las siguientes:

- Etiqueta de apertura: <nombre>.
- Etiqueta de cierre: </nombre>.
- Contenido del elemento: Litchi.
- Elemento nombre: <nombre>Litchi</nombre>.

Hay que resaltar que el elemento *especie* del ejemplo de la página 131 contiene 7 elementos hijo o subelementos: nombre, variedad, origen, color, fruto, maduración, riego y precio.

Aunque no es muy normal, pueden existir elementos vacíos, sin contenido alguno como el siguiente:

```
<casado></casado>
```

Un elemento vacío puede ser uno que no tenga etiqueta de cierre con o sin atributos:

```
<teléfono n="666777888" />
```

Ejercicio propuesto 4.3



Cuenta los elementos que hay en el siguiente documento XML:

```
<feed xmlns="http://www.w3.org/2005/Atom">
    <title>Árboles subtropicales</title>
    <subtitle>cultivo del aguacate</subtitle>
</feed>
```

Hay que tener en cuenta una serie de consideraciones a la hora de crear un elemento:

- Los elementos tienen que estar anidados correctamente unos dentro de otros, cerrándose en orden inverso al que se abren.
- Los elementos no vacíos tendrán etiquetas de apertura y cierre.
- Los nombres de los elementos y atributos tienen que seguir una nomenclatura específica como no empezar por número, no usar caracteres especiales reservados, etc. Tienen que empezar por letras o los caracteres guion bajo (_) o dos puntos (:), seguidos de guiones, puntos, números u otras letras. Tampoco pueden empezar por la palabra XML.

4.2.4. Atributos

Se usan para asignar propiedades asociadas a los elementos. Se sitúan dentro de una etiqueta de inicio o de apertura. Se especifican con el par NombreAtributo="valor", pudiendo añadirse varios separados por un espacio en blanco. Los valores tienen que estar encerrados entre comillas dobles o simples y no podrá haber dos atributos llamados iguales en un mismo elemento.

Dichos atributos proporcionan información adicional sobre los elementos. Aunque los atributos pueden usarse perfectamente en XML, no es recomendable abusar de ellos, ya que un uso excesivo podría hacer que el documento fuese menos entendible para el usuario.

En el siguiente ejemplo, el elemento *nombre* tiene el atributo *dni* cuyo valor es "77777777J".

```
<nombre dni="77777777J"> AlumnoLM </nombre>
```

Ejercicio propuesto 4.4

Crea un elemento que contenga dos atributos.

**Ejercicio resuelto 4.2**

Dado el siguiente fichero XML de una agenda personal, transforma los elementos *ciudad* e *email* en atributos.

```
<?xml version="1.0" encoding="i-
so-8859-1"?>
<agenda>
    <contacto nombre="alumno">
        <ciudad>Málaga</ciudad>
        <teléfono n="666777888"/>
        <email>a@librolm.com</email>
    </contacto>
</agenda>
```

Solución

4.2.5. Comentarios

Los comentarios sirven de ayuda cuando el documento es extenso o se pretende que todo se entienda con claridad. Pueden introducirse en cualquier parte, excepto dentro de las etiquetas, otros comentarios o declaraciones. Tienen la misma sintaxis que los comentarios HTML, comenzando por `<!--` y terminando con `-->`.

En el ejemplo siguiente, la línea 3 sería un comentario.

```
<?xml version="1.0" encoding="utf-8" ?>
<agenda>
    <!--vamos a crear un contacto-->
    <contacto nombre="alumno LM">
        <ciudad>Málaga</ciudad>
    </contacto>
</agenda>
```

4.2.6. Sección CDATA

Esta sección permite añadir contenido sin ser procesado o analizado. Deben aparecer dentro del elemento raíz de un documento XML. Es similar a un comentario cuya sintaxis es la siguiente:

```
<![CDATA[contenido dentro del bloque]]>
```

A continuación se muestra un ejemplo de XML con sección CDTA:

```
<?xml version="1.0" ?>
<asignaturas>
  <foro>
    <![CDATA[el <b>examen</b>, será el día]]>
  </foro>
</asignaturas>
```

Actividades propuestas



Responde si las siguientes afirmaciones son verdaderas o falsas y razona tu respuesta:

4.3. Los atributos se añaden dentro de la etiqueta de cierre.

4.4. Los comentarios en XML tienen la misma sintaxis que los comentarios en C#.

Verificar



Ejercicio resuelto 4.3



Es preciso mandar una tabla de manera segura y fiable a un programa cliente que recoge información sobre unos árboles subtropicales: variedad, color de fruto, maduración, etc.

Transforma el siguiente cuadro sobre las distintas especies en un documento XML que almacene dicha información:

Nombre	Variedad	Ciudad origen	Color del fruto	Maduración	Riego	Precio kg
Litchi.	Kway may pink.	Filipinas.	Rosa.	Agosto.	Diario.	3,50 €
Longan.	Champoo.	China.	Marrón.	Octubre.	Diario.	2,5 €
Litchi.	Mauritius.	Florida.	Rojo.	Agosto.	Diario.	3 €

Solución



4.3. Creación, asociación y elementos de DTD y XSD

A la hora de especificar una serie de restricciones a un documento XML, existen varias opciones que un usuario puede usar. La primera de ellas es mediante DTD, que suele ser poco flexible en las definiciones de cada uno de los elementos, ya que no es posible indicar si se trata de un elemento de tipo fecha, moneda, número etc. Estas carencias que presenta DTD se

superan usando XML Schema, siendo estas dos primeras opciones las que se estudien en este apartado. Por otra parte, existen alternativas como el uso de Relax NG (*regular language for XML next generation*) y Schematron.

Todas las opciones anteriormente vistas son métodos de validación de documentos XML, por lo que debe asociarse el fichero de restricciones al XML para que se vinculen de manera correcta.



SABÍAS QUE...

Trang es un programa de código abierto que se emplea para la generación de XML Schema y DTD mediante un documento XML. También permite convertir de DTD a XSD, y viceversa.

Existen herramientas como Oxygen XML Editor (capítulo 5) que permite, entre otros aspectos, la conversión entre distintos esquemas.

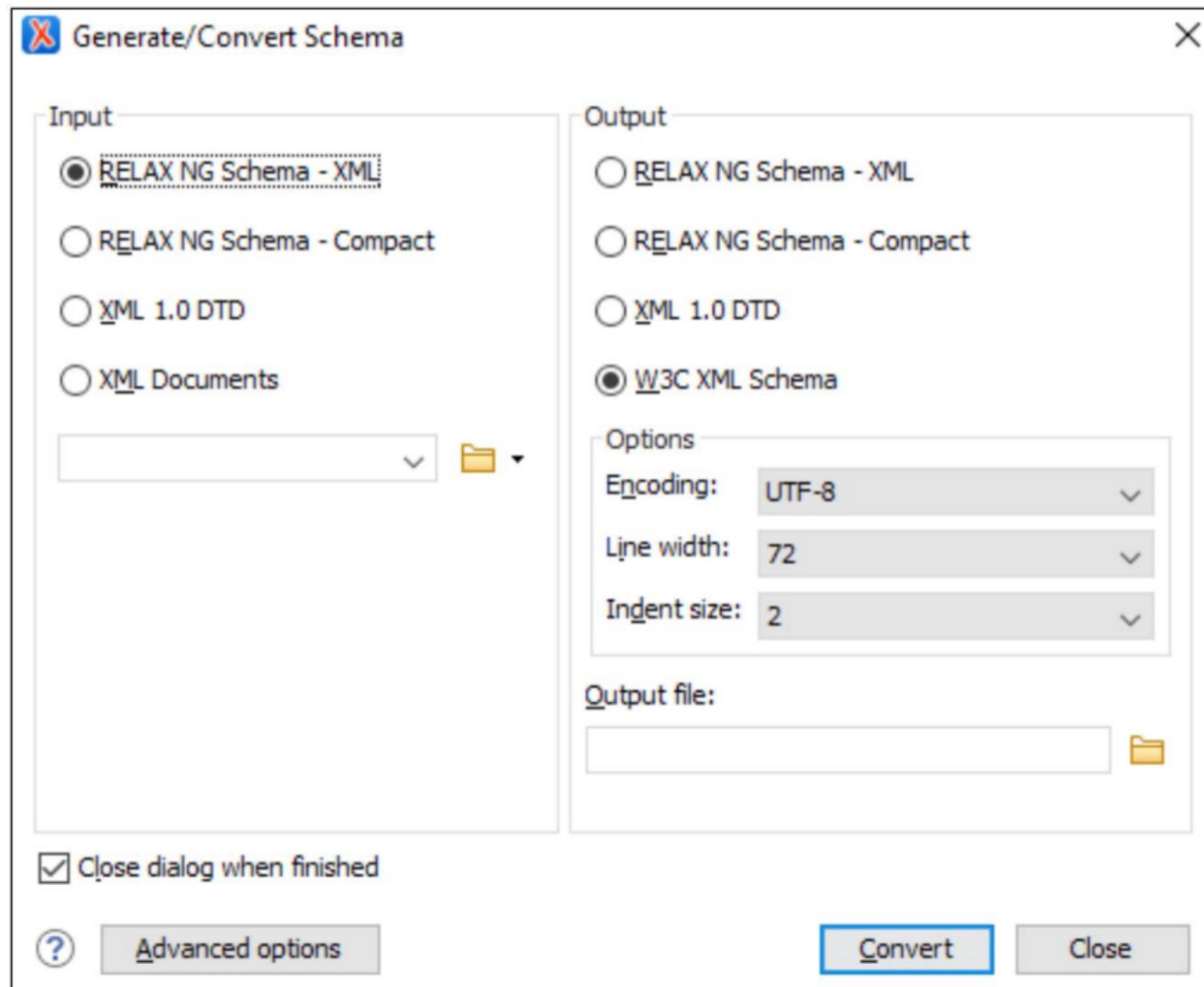


Figura 4.3
Conversión de esquemas Oxygen XML Editor.

Existen generadores de esquemas de escritorio (Oxygen XML Editor, XMLSpy, Visual Studio, XML Copy Editor, etc.) y *online* que facilitan obtener la DTD o SXD de manera automática, ahorrando así al usuario la tediosa tarea de crear XSD o DTD de documentos XML extensos.

En la figura 4.4, se aprecia una herramienta *online* donde se han añadido etiquetas XML, lo que permite obtener documentos RNC, RNG, DTD y XSD.

The XML Schema Generator creates a basic, easily adapted XML schema from an XML file.

Paste the contents of your XML file below: *

```
<?xml version="1.0" encoding="iso-8859-1"?>
<árbol>
  <especie>
    <nOMBRE>Litchi</nOMBRE>
    <variedad>Kway May Pink</variedad>
    <origen>Filipinas</origen>
    <color_fruto>rojo</color_fruto>
    <maduración>agosto</maduración>
    <riego> diario</riego>
    <precio>3.5</precio>
  </especie>
  <especie>
    <nOMBRE>Longan</nOMBRE>
    <variedad>Champoo</variedad>
    <origen> China</origen>
    <color_fruto>marrón</color_fruto>
```

Output Format: *

RNG RNC DTD XSD

Generate

This XML Schema Generator accepts one XML document and infers a schema.

It produces as output a schema written in any of the following formats:

- **RNG**: RELAX NG (XML syntax)
- **RNC**: RELAX NG (compact syntax)
- **DTD**: XML 1.0 DTD
- **XSD**: W3C XML Schema

Figura 4.4

Generador de esquemas *online* usando XML Schema Generator.

4.3.1. DTD (definición de tipo de documento)

Definición de tipo de documento o DTD es un documento de texto plano donde se especifican o definen una serie de *normas* que se usan para saber cómo tiene que crearse un fichero XML para poder validarla. Gracias a este tipo de documentos, puede especificarse el número de ocurrencias de un elemento XML y el tipo de datos, así como el orden de precedencia, entre otros aspectos.

Aunque no es obligatoria la creación de una DTD, es recomendable su uso si la información del XML va a ser compartida. De esa manera, los usuarios que trabajen con el documento XML sabrán, gracias al DTD, su definición o reglas a cumplir.

Su sintaxis es distinta a la de XML, pues no pueden definirse tipos de datos por parte de los usuarios ni insertar elementos no ordenados.

RECUERDA

- ✓ Para crear un documento XML, es recomendable previamente crear una DTD, donde se especifiquen los elementos que intervienen.

A) Creación y elementos de una DTD

Una vez que se han determinado las reglas que servirán como base para la generación de documentos XML, es el momento de empezar a construir la DTD. Para ello, se estudiarán tres componentes principales que lo forman (elementos, atributos y entidades).

Para crear este tipo de documentos, puede usarse un editor de texto plano, teniendo en cuenta que la extensión de estos documentos será .dtd.



Figura 4.5
Componentes DTD.

1. Elementos

Los elementos de una DTD son la base de su estructura. Lo primero que habría que mirar es si existe dependencia jerárquica, especificando el nodo padre y, entre paréntesis, sus hijos si los tuviera, separados por el carácter coma (,) tal y como se aprecia a continuación:

```
<!ELEMENT nombreElemento (elementos_hijo1, elemento_hijo2, ...)>
```

Por otra parte, cada uno de los elementos debe ser de un tipo específico, pudiendo distinguir entre *any*, *empty*, *#pcdata* o *#cdata*. Se especifica tal y como se muestra en el ejemplo siguiente:

```
<!ELEMENT nombreElemento Tipo>
```

El *tipo* puede ser:

- (*#PCDATA*): *parsed character data*, que indica que el contenido de ese elemento es de tipo texto y es analizado por un *parser*.
- (*#CDATA*): *character data*, similar a PCDATA con la obviedad de que el contenido de los elementos no es analizado por un *parser* y no pueden detectarse posibles entidades.
- *ANY*: los elementos pueden contener cualquier valor.
- *EMPTY*: los elementos no tienen contenido. Elemento vacío.



SABÍAS QUE...

Un *parser* es un analizador o procesador XML que comprueba si el documento está bien formado o es válido, analizando su estructura.

Para plasmar lo visto, se verá el ejemplo a continuación del cuadro 4.1 donde quiere crearse un documento DTD para el envío de correos electrónicos que contiene los campos (receptor, asunto y cuerpo).

CUADRO 4.1**Ejemplos de campos de un *email***

Receptor	Asunto	Cuerpo
r1@librolm.com	Cumpleaños	Desearte un feliz...
r2@librolm.com	Día libre	Estimado señor X,...

Un fichero DTD para un XML podría ser:

```
<!ELEMENT bdcorreo (email)+>
<!ELEMENT email (receptor,asunto,cuerpo)>
<!ELEMENT receptor (#PCDATA)>
<!ELEMENT asunto (#PCDATA)>
<!ELEMENT cuerpo (#PCDATA)>
```

■ Ocurrencia de aparición de los elementos

Si quiere especificarse que un elemento puede aparecer en el XML cero, una o varias veces, es necesario contar con una nomenclatura específica para plasmar lo que se desea. Para ello, se usa una serie de caracteres que informa de la obligatoriedad o no, así como del número de veces que puede aparecer dicho elemento.

CUADRO 4.2**Caracteres de ocurrencia de los elementos**

Carácter	Descripción
?	El elemento puede aparecer cero o una vez (opcional).
*	El elemento puede aparecer cero, una o varias veces (opcional).
+	El elemento debe aparecer una o más veces (obligatorio).
	Similar al (OR) lógica, indica que puede añadirse uno de los dos elementos que se especifiquen, pero no ambos (obligatorio):

```
<!ELEMENT NombreE (hijo1, (hijo2 | hijo3)) >
```

En el ejemplo siguiente, se observa el uso de los caracteres más (+) y pleca (|).

Se ha creado un XML que almacena componentes electrónicos donde puede aparecer el elemento precio o precio con IVA (solo uno de los dos). Para ello, se ha añadido (**precio|precioiva**), que indica que puede añadirse cualquier elemento que haya en la lista. Por otro lado, el signo + informa de que el elemento debe aparecer una o más veces.

```

<?xml version="1.0" encoding="iso-8859-1"?>
<!DOCTYPE BDcomponentes [
<!ELEMENT BDcomponentes (componente)+>
<!ELEMENT componente (nombre,(precio|precioiva),tipo+,medida)>
<!ELEMENT nombre (#PCDATA)>
<!ELEMENT precio (#PCDATA)>
<!ELEMENT precioiva (#PCDATA)>
<!ELEMENT tipo (#PCDATA)>
<!ELEMENT medida (#PCDATA)>
]>
<BDcomponentes>
  <componente>
    <nombre>resistencia</nombre>
    <precio>0.25</precio>
    <tipo>pasivo</tipo>
    <tipo>bobinadas</tipo>
    <medida>ohmios</medida>
  </componente>
  <componente>
    <nombre>condensador</nombre>
    <precioiva>0.75</precioiva>
    <tipo>pasivo</tipo>
    <medida>Faradios</medida>
  </componente>
</BDcomponentes>

```

Si ahora se modifica el ejemplo anterior y no se añade al componente el elemento *tipo*, no será válido e informará del error.

```

<?xml version="1.0" encoding="iso-8859-1"?>
<!DOCTYPE BDcomponentes [
<!ELEMENT BDcomponentes (componente)+>
<!ELEMENT componente (nombre,(precio|precioiva),tipo+,medida)>
<!ELEMENT nombre (#PCDATA)>
<!ELEMENT precio (#PCDATA)>
<!ELEMENT precioiva (#PCDATA)>
<!ELEMENT tipo (#PCDATA)>
<!ELEMENT medida (#PCDATA)>
]>
<BDcomponentes>
  <componente>
    <nombre>resistencia</nombre>
    <precio>0.25</precio>
    <tipo>pasivo</tipo>
    <tipo>bobinadas</tipo>

```

[.../...]

```

<medida>ohmios</medida>
</componente>
<componente>
  <nombre>condensador</nombre>
  <precioiva>0.75</precioiva>
  <medida>Faradios</medida>
</componente>
</BDcomponentes>

```

Para solucionar el problema, podría cambiarse la siguiente línea, modificando el signo más (+) por un asterisco (*):

```
<!ELEMENT componente (nombre,(precio|precioiva),tipo*,medida)>
```

2. Atributos

Los atributos en una DTD se usan para enriquecer o añadir información a un elemento, pero su uso no es muy recomendable, ya que no pueden contener varios valores, no son extensibles, etc. Si un elemento contiene más de un atributo, se definen uno detrás de otro, separándose mediante espacios, tabuladores o nuevas líneas. Se declaran usando la marca `<!ATTLIST` seguida del elemento que contiene el atributo, su nombre, tipo y valor, tal y como se muestra a continuación.

```
<!ATTLIST Elemento Atributo Tipo Valor>
```

Donde:

- *Elemento*: es el nombre del elemento al que quiere añadirse un atributo.
- *Atributo*: nombre del atributo por añadir.
- *Tipo*: se selecciona un tipo del cuadro 4.3.
- *Valor*: comportamiento del valor del atributo.

CUADRO 4.3
Tipos de atributos

Tipo atributo	Explicación
CDATA	El valor es alfanumérico (cadena de texto).
Enumeraciones	Permite especificar varios valores para un atributo: <code><!ATTLIST numero octal(0 1 2 3 4 5 6 7) CDATA #REQUIRED></code>
ENTITY	El atributo es una entidad que se ha definido previamente.
IDREF	Hace referencia al identificador de otro elemento.
IDREFS	Hace referencia a un conjunto de identificadores (ej1 ej2 ej3).
ID	Su contenido es único, identificando de manera única a cada elemento.

[.../...]

CUADRO 4.3 (CONT.)

NOTATION	Se especifican datos que no son XML.
ENTITIES	Conjunto o lista de entidades.
NMTOKENS	Permite especificar el valor de un atributo mediante caracteres permitidos.

Para especificar si los valores de los atributos son obligatorios u opcionales, se usan los modificadores del cuadro 4.4.

CUADRO 4.4
Valores de atributos

Valor	Significado
#FIXED	Debe especificarse un valor que es fijo y no podrá cambiarse.
#IMPLIED	El atributo es opcional.
#DEFAULT	Debe especificarse un valor por defecto que sí podrá cambiarse.
#REQUIRED	El atributo es obligatorio.

En el ejemplo siguiente se aprecia un documento XML con su DTD asociada, donde aparecen dos atributos obligatorios definidos mediante <!ATTLIST>.

```
<?xml version="1.0" encoding="iso-8859-1"?>
<!DOCTYPE bdusuarios [
    <!ELEMENT bdusuarios (usuario)+>
    <!ELEMENT usuario (nombre,email,fechaNac)>
    <!ATTLIST usuario
        clave CDATA #REQUIRED
        dni CDATA #REQUIRED>

    <!ELEMENT nombre (#PCDATA)>
    <!ELEMENT email (#PCDATA)>
    <!ELEMENT fechaNac EMPTY>
]>
<bdusuarios>
    <usuario clave="**" dni="*****">
        <nombre>Ataulfo G. Pascual</nombre>
        <email>user1@librolm.com</email>
        <fechaNac></fechaNac>
    </usuario>
    <usuario clave="****" dni="*****">
        <nombre>Francisco J. García</nombre>
        <email>user2@librolm.com</email>
        <fechaNac></fechaNac>
    </usuario>
</bdusuarios>
```

3. Entidades

Las entidades se consideran constantes y pueden usarse para abreviar texto o utilizar algunos caracteres especiales como el signo de menor que (<) cuya entidad es (<).

Pueden clasificarse en internas y externas (especificando la URL del documento que contiene la entidad).

Para crearlas dentro de una DTD, se usa la marca <!ENTITY> seguida del nombre de la entidad y su valor asociado, tal y como se muestra a continuación.

```
<!ENTITY nombre valor>
```

La manera de referenciar a una entidad es dentro del documento XML usando el carácter *ampersand* (&) que precede al nombre de la entidad terminada en el carácter punto y coma (;).

Para comprender mejor este concepto, va a crearse una serie de entidades propias en el documento DTD, usándolas en el XML, como se muestra en el ejemplo siguiente.

Las dos entidades son:

- Web con valor “<http://www.librolm.com>”.
- Cliente cuyo valor es “Se informa que el usuario ha accedido al”.

Las referencias se hacen mediante:

- &cliente;
- &web;

```

<?xml version="1.0" encoding="iso-8859-1"?>
<!DOCTYPE bdusuarios[
    <!ELEMENT bdusuarios (usuario)+>
    <!ELEMENT usuario (email,nombre,suceso)>
    <!ELEMENT email (#PCDATA)>
    <!ELEMENT nombre (#PCDATA)>
    <!ELEMENT suceso (#PCDATA)>
    <!ENTITY web "http://www.librolm.com">
    <!ENTITY cliente "Se informa que el usuario ha accedido al">
]>
<bdusuarios>
    <usuario>
        <email>user1@librolm.com</email>
        <nombre>Ataulfo G. Pascual</nombre>
        <suceso> &cliente; registro en la página &oZz; </suceso>
    </usuario>
    <usuario>
        <email>user2@librolm.com</email>
        <nombre>Francisco J. García</nombre>
        <suceso> &cliente; contenido de &oZz; para consultar el libro X</
            suceso>
    </usuario>
</bdusuarios>
```

Al visualizarlo en un navegador, la referencia a la entidad del documento XML se transforma en el valor asociado (figura 4.6).

```

<?xml version="1.0" encoding="iso-8859-1" ?>
<!DOCTYPE bdusuarios (View Source for full doctype...)>
- <bdusuarios>
  - <usuario>
    <email>user1@librolm.com</email>
    <nombre>Ataulfo G. Pascual</nombre>
    <suceso>Se informa que el usuario ha accedido al registro en la
      página http://www.librolm.com</suceso>
  </usuario>
  - <usuario>
    <email>user2@librolm.com</email>
    <nombre>Francisco J. García</nombre>
    <suceso>Se informa que el usuario ha accedido al contenido de
      http://www.librolm.com para consultar el libro X</suceso>
  </usuario>
</bdusuarios>

```

Figura 4.6
Visualización de entidades XML en el navegador.



Actividades propuestas

Responde si las siguientes afirmaciones son verdaderas o falsas y razona tu respuesta:

- V F 4.5.** Los componentes principales de una DTD son (elementos, atributos y entidades).
- V F 4.6.** Los atributos en una DTD se declaran con la marca <!ENTITY.



B) Asociación DTD en XML

Existen dos maneras de vincular un fichero DTD a un XML. La primera de ellas es ubicarlo en el mismo documento XML y la segunda usando un fichero externo que puede estar en el mismo servidor o uno externo. Aunque las dos posibilidades son aceptadas, es recomendable usar la segunda opción por temas de claridad, compartición y manipulación, ya que, si una DTD se usa para distintos XML y hubiera que hacer modificaciones, habría que cambiar varios documentos en vez de uno.

RECUERDA

- ✓ Puede declararse una DTD en el mismo XML, así como en un fichero externo.

Para declarar una DTD en el mismo documento, la segunda línea del XML será <!DOCTYPE seguida del elemento raíz y, entre corchetes, los distintos elementos, tal y como se muestra en el ejemplo del ejemplo:

```

<?xml version="1.0" encoding="iso-8859-1"?>
<!DOCTYPE bdcorreo [
    <!ELEMENT bdcorreo (email)+>
    <!ELEMENT email (receptor,asunto,cuerpo)>
    <!ELEMENT receptor (#PCDATA)>
    <!ELEMENT asunto (#PCDATA)>
    <!ELEMENT cuerpo (#PCDATA)>
]>
<bdcorreo>
    <email>
        <receptor>r1@librolm.com</receptor>
        <asunto>Cumpleaños</asunto>
        <cuerpo>Desearte un feliz..</cuerpo>
    </email>
    <email>
        <receptor>r2@librolm.com</receptor>
        <asunto>Día libre</asunto>
        <cuerpo>Estimado señor X,...</cuerpo>
    </email>
</bdcorreo>

```

Si, por el contrario, el usuario desea declarar la DTD en un documento externo, simplemente tendrá que crear dos ficheros independientes. El primero para la DTD siguiendo las reglas que van a detallarse a continuación y el segundo para el XML.

Para comenzar a declarar una *DTD en un fichero independiente*, simplemente hay que introducir los distintos elementos que lo forman. En el documento XML, la llamada al DTD se realiza mediante la instrucción o marca `<!DOCTYPE` seguida del elemento raíz y el nombre del fichero DTD entre comillas.

Entre el nombre del documento y el elemento raíz, suele ponerse si la definición es privada (SYSTEM), siendo esta la manera más usada o pública (PUBLIC) definida por organismos, lo que obliga a poner un campo más llamado *identificador FPI (formal public identifier)*.

Para ver una definición de DTD independiente, se aprecia la asociación del DTD en el XML en la segunda línea:

```

<?xml version="1.0" encoding="iso-8859-1"?>
<!DOCTYPE bdcorreo SYSTEM "midtd.dtd">
<bdcorreo>
    <email>
        <receptor>r1@librolm.com</receptor>
        <asunto>Cumpleaños</asunto>
        <cuerpo>Desearte un feliz..</cuerpo>
    </email>
    <email>
        <receptor>r2@librolm.com</receptor>
        <asunto>Día libre</asunto>
        <cuerpo>Estimado señor X,...</cuerpo>
    </email>
</bdcorreo>

```

El fichero DTD creado se llamará *midtd.dtd* y será el del ejemplo siguiente:

```
<!ELEMENT bdcorreo (email)+>
<!ELEMENT email (receptor,asunto,cuerpo)>
<!ELEMENT receptor (#PCDATA)>
<!ELEMENT asunto (#PCDATA)>
<!ELEMENT cuerpo (#PCDATA)>
```

RECUERDA

La primera línea de un documento XML posee un atributo llamado *standalone* con los siguientes valores:

- Si se usa DTD externa, el valor es “no”.
- Si no se usa DTD externa, el valor es “yes”.

Ejercicio resuelto 4.4



Dado el DTD para crear distintas especies de árboles, obtén un XML asociado:

```
<!ELEMENT árbol (especie)+>
<!ELEMENT especie (nombre,variedad,(origen)+,color_fruto,
    (maduración)+,(riego)?,(precio)?)>
<!ELEMENT nombre (#PCDATA)>
<!ELEMENT variedad (#PCDATA)>
<!ELEMENT origen (#PCDATA)>
<!ELEMENT color_fruto (#PCDATA)>
<!ELEMENT maduración (#PCDATA)>
<!ELEMENT riego (#PCDATA)>
<!ELEMENT precio (#PCDATA)>
```

Solución

4.3.2. XML Schema XSD

El lenguaje XML Schema o XSD (*XML schema definition*) surgido en 1998 y se usa como alternativa a documentos DTD, al no cubrir este las expectativas que se necesitan, ya que no permite especificar tipos de elementos o atributos más complejos, así como añadir otras restricciones más complejas.

Entre las ventajas que presenta, destacan:

- Más definición de tipos básicos.
- Mayor control sobre el número de ocurrencias de los elementos.
- Sintaxis similar a XML.

Los XML Schema son ficheros planos con extensión .xsd que permiten definir los elementos y atributos que componen un XML, así como sus tipos, el orden de aparición, el número de veces que puede repetirse cada elemento, valores por defecto, etc.

A) Creación y elementos de un XSD

La estructura de todo documento XSD es similar a la de un XML, pero presentando el elemento raíz `<xs:schema>`. Dicho elemento contiene una serie de atributos, entre los que destacan los siguientes:

- `xmlns:alias`. Se especifica el espacio de nombres de donde provienen los elementos y tipos usados, cuyo valor tiene que ser “<http://www.w3.org/2001/XMLSchema>”. El alias suele ser `xs` o `xsd`, aunque, mientras se use la misma en todo el documento, no importa el nombre que se le asigne. Hacen referencia a las etiquetas del espacio de nombres.
- `elementFormDefault`. A la hora de declarar los elementos, hay que indicar si debe añadirse el espacio de nombres delante, cuyos posibles valores son “qualified” y “unqualified”.
- `attributeFormDefault`. A la hora de declarar los atributos, hay que indicar si debe añadirse el espacio de nombres delante, cuyos posibles valores son “qualified” y “unqualified”, que indica que el primero de ellos debe ser obligatorio. Por defecto, se usa la segunda opción.
- `targetNamespace`. Se especifica el espacio de nombres de los elementos definidos.
- `versión`. Se especifica la versión del documento de esquema.

Aunque no es obligatorio, la primera línea puede ser la declaración del XML. En la segunda, tal y como se muestra a continuación, se define el esquema, con la especificación de los distintos elementos. El alias en este caso es `xs`.

```

<?xml version= "1.0" encoding= "utf-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
    elementFormDefault="qualified">
    ...
    Elementos
    ...
</xs:schema>

```

1. Elementos

Puede realizarse una distinción entre los tipos de elementos de un XSD. Están los elementos simples y complejos. Elementos *sin hijos ni atributos o simples* se definen con la marca `<xs:element>`, conteniendo solamente valores de un determinado tipo.

```
<xs:element Nombre Tipo Valor/>
```

Donde:

- *Nombre*: se representa mediante el atributo *name*, asignándole el valor que se desee.
- *Tipo*: se usa el atributo *type*, que permite especificar el tipo de elemento (ver cuadro 4.6).
- *Valor*: puede ser un valor por defecto (*default*) que permite ser modificado o un valor fijo obligatorio (*fixed*) que no podrá cambiarse. Cada uno de ellos irá acompañado de su correspondiente asignación.

A continuación, se aprecia un elemento XSD llamado *color* de tipo cadena de caracteres o texto y valor por defecto “azul”.

```
<xs:element name="color" type="xs:string" default="azul"/>
```

■ Indicadores de ocurrencias

Indica las veces que un elemento debe aparecer o repetirse mediante dos atributos que informan del número máximo y mínimo de veces que puede aparecer.

CUADRO 4.5
Indicadores de ocurrencias de elementos

Atributo	Significado
maxOccurs	Se especifica un valor con el número máximo de veces que puede aparecer el elemento, cuyo valor por defecto es 1. Para especificar un número de veces ilimitado, el valor es “unbounded”.
minOccurs	Se especifica un valor con el número mínimo de veces que puede aparecer el elemento, cuyo valor por defecto 1. Para especificar un número de veces ilimitado, el valor es “unbounded”.

■ Elementos complejos

Contienen otros elementos y atributos. Dependiendo del contenido, pueden darse ejemplos de elementos con hijos, con contenido vacío y atributos, con contenido no vacío y atributos, entre otros.

Se definen con el tipo de datos `<xs:complexType>`, seguido de un indicador de orden y los elementos simples, que puede variar dependiendo del contenido que tenga.

```

<xs:element name="nombreElementoCompuesto">
  <xs:complexType>
    <Indicador_Orden>
      Elementos / atributos
    </ Indicador_Orden>
  </xs:complexType>
</xs:element>

```

Donde:

- <xs:complexType>: se usa para definir elementos de tipo complejo.
- *Indicador_Orden*: en este apartado, hay que especificar un indicador de orden de aparición de los distintos elementos, que puede ser tres distintos:
 1. <xs:sequence>: se especifican los hijos del elemento principal, siguiendo la secuencia u orden de aparición indicada.
 2. <xs:choice>: se especifica el elemento hijo que puede aparecer de entre todos. Solo uno de ellos.
 3. <xs:all>: los hijos pueden aparecer en cualquier orden una sola vez, la secuencia aparecerá en el orden que se desee.

Para ver con detalle este tipo de elementos, va a realizarse una serie de ejemplos donde irán definiéndose los XDS de los XSM.

En el siguiente ejemplo, se declara un XSD con elementos complejos mediante referencias. La secuencia de elementos que debe aparecer es *nombre*, *email* y *fechaNac*, en ese orden. La referencia se realiza con el atributo *ref* y cuyo valor es el nombre del elemento situado en la parte inferior, donde se especifica el tipo.

```

<xs:schema elementFormDefault="qualified" xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <xs:element name="usuario">
    <xs:complexType>
      <xs:sequence>
        <xs:element ref="nombre"/>
        <xs:element ref="email"/>
        <xs:element ref="fechaNac"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
  <xs:element name="nombre" type="xs:string"/>
  <xs:element name="email" type="xs:string"/>
  <xs:element name="fechaNac" type="xs:date"/>

```

El ejemplo anterior puede realizarse sin referencias, como se muestra a continuación, donde se observa un elemento compuesto (*usuario*) y tres elementos simples (*nombre*, *email* y *fechaNac*).

```

<xs:schema elementFormDefault="qualified" xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <xs:element name="usuario">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="nombre" type="xs:string"/>
        <xs:element name="email" type="xs:string"/>
        <xs:element name="fechaNac" type="xs:date"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
</xs:schema>

```

Un XML correcto a la hora de validarla podría ser el siguiente:

```

<?xml version="1.0" encoding="iso-8859-1"?>
<usuario>
  <nombre>Francisco J. García</nombre>
  <email>user2@librolm.com</email>
  <fechaNac>2018-10-25</fechaNac>
</usuario>

```

Si, por el contrario, el usuario se equivoca a la hora de construir el XML siguiendo el esquema, que dará error al validar. Este aspecto se aprecia a continuación, donde se ha cambiado de orden el segundo y el tercer elemento.

```

<?xml version="1.0" encoding="iso-8859-1"?>
<usuario>
  <nombre>Francisco J. García</nombre>
  <fechaNac>2018-10-25</fechaNac>
  <email>user2@librolm.com</email>
</usuario>

```

El indicador de orden `<xs:choice>` permite especificar uno de los elementos hijos que aparecen, pero solo uno de ellos. En el ejemplo siguiente, se pretende que el usuario se identifique con su nombre de usuario o *email*, con uno de los dos.

```

<xs:element name="usuario">
  <xs:complexType>

```

[.../...]

```

<xs:choice>
    <xs:element name="nombreU" type="xs:string"/>
    <xs:element name="email" type="xs:string"/>
</xs:choice>
</xs:complexType>
</xs:element>
</xs:schema>

```

Un posible XML válido sería:

```

<?xml version="1.0" encoding="iso-8859-1"?>
<usuario>
    <email>user2@librolm.com</email>
</usuario>

```

Un XML que no cumple a la hora de validar porque se han añadido dos elementos en vez de seleccionar uno de la lista sería el que se presenta en el ejemplo siguiente:

```

<?xml version="1.0" encoding="iso-8859-1"?>
<usuario>
    <nombreU>Sergio</nombreU>
    <email>user2@librolm.com</email>
</usuario>

```

El indicador de orden <xs:all>: permite especificar la secuencia de elementos en cualquier orden, aunque sin poder añadir dentro de él <xs:choice> o <xs:sequence>. La especificación se muestra en el ejemplo siguiente:

```

<xs:schema elementFormDefault="qualified" xmlns:xs="http://www.w3.org/2001/XMLSchema">
    <xs:element name="usuario">
        <xs:complexType>
            <xs:all>
                <xs:element name="nombre" type="xs:string"/>
                <xs:element name="email" type="xs:string"/>
                <xs:element name="fechaNac" type="xs:date"/>
            </xs:all>
        </xs:complexType>
    </xs:element>
</xs:schema>

```

Algunos de los XML válidos al tener en cuenta el XSD del ejemplo anterior son los que se muestran a continuación, donde aparecen distintas alternativas dependiendo del número de elementos que haya dentro de <xs:all>.

```
?xml version="1.0" encoding="iso-8859-1"?
<usuario>
    <nombre>Francisco J. García</nombre>
    <fechaNac>2018-10-25</fechaNac>
    <email>user2@librolm.com</email>
</usuario>
```

```
<?xml version="1.0" encoding="iso-8859-1"?>
<usuario>
    <fechaNac>2018-10-25</fechaNac>
    <nombre>Francisco J. García</nombre>
    <email>user2@librolm.com</email>
</usuario>
```

```
<?xml version="1.0" encoding="iso-8859-1"?>
<usuario>
    <fechaNac>2018-10-25</fechaNac>
    <email>user2@librolm.com</email>
    <nombre>Francisco J. García</nombre>
</usuario>
```

Si pretende crearse un XML que no cumpla con las especificaciones del XSD, bastaría con eliminar uno de los elementos de la lista que tiene que aparecer:

```
<?xml version="1.0" encoding="iso-8859-1"?>
<usuario>
    <fechaNac>2018-10-25</fechaNac>
    <email>user2@librolm.com</email>
</usuario>
```

Tal y como se ve en el XML del ejemplo siguiente, donde se muestra un dato complejo con texto y atributos, existe un elemento *nombre* con el atributo *DNI* y contenido Eduardo.

```
<?xml version="1.0" encoding="iso-8859-1"?>
<nombbre dni="11111111K">Eduardo</nombbre>
```

Un esquema asociado al ejemplo anterior sería el que se presenta a continuación:

```

<?xml version="1.0" encoding="iso-8859-1"?>
<xss:schema elementFormDefault="qualified" xmlns:xss="http://www.w3.org/2001/
  XMLSchema">
  <xss:element name="nombre">
    <xss:complexType>
      <xss:simpleContent>
        <xss:extension base="xss:string">
          <xss:attribute name="dni" type="xss:string"/>
        </xss:extension>
      </xss:simpleContent>
    </xss:complexType>
  </xss:element>
</xss:schema>

```

Ejercicio resuelto 4.5



Crea un XSD para un XML de un elemento vacío con atributo como el siguiente:

```

<?xml version="1.0" encoding="iso-8859-1"?>
<nombre dni="11111111K"></nombre>

```

Solución

2. Tipos de datos XSD

XML Schema dispone de numerosos tipos de datos simples predefinidos incorporados para los elementos o atributos, siendo los más comunes los que se observan en el cuadro 4.6

CUADRO 4.6
Tipos de datos predefinidos XSD

Tipo	Descripción
decimal	Número decimal.
boolean	Un único valor lógico (1, 0).
dateTime	Fecha y hora en formato (AAAA-MM-DD T HH:MM:SS).
string	Texto o cadena de caracteres.
date	Fecha: año-mes-día (aaaa-mm-dd).
time	Hora: hh:mm:ss.
integer	Número entero positivo o negativo.
positiveInteger	Número entero positivo.
long	Entero de 64 bits.
short	Entero de 16 bits.

En el cuadro 4.7, se aprecia un elemento nombre de tipo cadena de caracteres (xs:string).

CUADRO 4.7
Aplicando tipo cadena de caracteres a un elemento

Elemento XSD	<xs:element name="nombre" type="xs:string"/>
Ej. XML	<nombre>Francisco J. García</nombre>

■ Creación de tipos de datos simples personalizados

En ocasiones, es interesante crear tipos de datos nuevos personalizados, dotándolos de mayor precisión y eficiencia. Estos pueden ser asignados a elementos o atributos, siendo reutilizados en el esquema XDS.

Para definir un tipo simple nuevo, se usa la siguiente estructura <xs:simpleType>, cuyo atributo *name* servirá para asignárselo al elemento e indicar que es de un tipo simple nuevo. Dentro de un tipo simple, pueden añadirse restricciones o facetas, listas, uniones, etc.

La sintaxis puede apreciarse en el ejemplo siguiente:

```
<xs:simpleType name="Tcat1">
  ...
</xs:simpleType>
```

■ Tipos de datos complejos

Ya vistos con los elementos complejos, se muestran en el primer ejemplo el XML de usuarios y en el segundo la DTD asociada.

```
<?xml version="1.0" encoding="iso-8859-1"?>
<usuario>
  <nombre>Francisco J. García</nombre>
  <email>user2@librolm.com</email>
  <fechaNac>2018-10-25</fechaNac>
</usuario>
```

```
<?xml version="1.0" encoding="iso-8859-1"?>
<xs:schema attributeFormDefault="unqualified" elementFormDefault="qualified"
  xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <xs:element name="usuario" type="Tuser" />
  <xs:complexType name="Tuser" >
    <xs:sequence>
      <xs:element name="nombre" type="xs:string" />
      <xs:element name="email" type="xs:string" />
      <xs:element name="fechaNac" type="xs:date" />
    </xs:sequence>
  </xs:complexType>
</xs:schema>
```

3. Atributos

Poseen la misma funcionalidad que los atributos de un elemento DTD. Los elementos con atributos se consideran elementos complejos, declarándose como tipos simples. Intentan dotarlos de más información usando la siguiente sintaxis.

Existe la posibilidad, al igual que en los elementos, de añadir restricciones a sus tipos.

```
<xs:attribute NombreA TipoA USO ValorA/>
```

Donde:

- *NombreA*: se representa mediante el atributo *name*, asignándole el valor que se desee.
- *TipoA*: se usa el atributo *type*, que permite especificar el tipo de elemento.
- *USO*: se especifica la opcionalidad o no del atributo mediante *use*, que puede tener los valores: “prohibited”, que indica que es prohibido; “required” para indicar que es obligatorio, y “optional”, que informan de que es opcional y es el valor por defecto.
- *ValorA*: puede ser un valor por defecto (*default*) o un valor fijo u obligatorio (*fixed*) con su correspondiente asignación.

```
<xs:attribute name="dni" type="xs:integer" use="required"/>
```

4. Restricciones o facetas

Si quisiera ponerse una restricción (también llamadas *facetas*) a un atributo o elemento, hasta ahora, con DTD no podía especificarse. XML Schema permite especificar esta característica controlando en mejor medida los datos introducidos en el XML. En el cuadro 4.8, se aprecian diversas facetas junto con su descripción.

CUADRO 4.8

Restricciones o facetas

Faceta	Descripción
length	Contiene un atributo <i>value</i> donde se especifica un valor de longitud fija.
minLength	Contiene un atributo <i>value</i> donde se especifica el valor de longitud mínima.
maxLength	Contiene un atributo <i>value</i> donde se especifica el valor de longitud máxima.
whiteSpace	Contiene un atributo <i>value</i> donde se especifica un valor que puede ser (“preserve”, “replace” y “collapse”). Trata el uso de los espacios en blanco, retornos de carro, etc., que puedan aparecer.
maxExclusive	Contiene un atributo <i>value</i> donde se especifica el valor, que es menor al especificado.
minExclusive	Contiene un atributo <i>value</i> donde se especifica el valor, que es mayor al especificado.

[.../...]

CUADRO 4.8 (CONT.)

minInclusive	Contiene un atributo <i>value</i> donde se especifica el valor, que es mayor o igual al especificado.
maxInclusive	Contiene un atributo <i>value</i> donde se especifica el valor, que es menor o igual al especificado.
totalDigits	Contiene un atributo <i>value</i> donde se especifica el valor, que es el número máximo de dígitos de un número teniendo en cuenta los decimales.
fractionDigits	Contiene un atributo <i>value</i> donde se especifica el valor, que es el número máximo de dígitos de decimales de un número.
enumeration	Contiene un atributo <i>value</i> donde se especifica uno de los valores admitidos de la lista.
pattern	Contiene un atributo <i>value</i> donde se especifica un rango de caracteres admitidos o expresión regular.

Para comprender mejor este apartado, se presenta el XML siguiente, que almacena datos relativos a varios productos.

```
<?xml version="1.0" encoding="iso-8859-1"?>
<bdproductos xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:no-
NamespaceSchemaLocation="zax.xsd">
  <producto>
    <nombre>teclado</nombre>
    <código>C001</código>
    < categoría>A</ categoría>
    <iva>21</iva>
    <precio>15</precio>
    <descuento>10</descuento>
  </ producto>
  <producto>
    <nombre>monitor</nombre>
    <código>C002</código>
    < categoría>B</ categoría>
    <iva>21</iva>
    <precio>147.2</precio>
    <descuento>9</descuento>
  </ producto>
</bdproductos>
```

El esquema asociado es el que aparece a continuación, donde algunos de los tipos de los elementos son tipos simples nuevos personalizados, con una serie de restricciones cada uno de ellos.

Para definir una restricción sobre un elemento, puede hacerse de varias maneras.

```

<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema" elementFormDefault="qualified">
  <xs:element name="bdproductos">
    <xs:complexType>
      <xs:sequence>
        <xs:element maxOccurs="unbounded" ref="producto"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
  <xs:element name="producto">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="nombre" type="xs:string"/>
        <xs:element name="código" type="Tlongitudentre"/>
        <xs:element name="categoría" type="Tcat1"/>
        <xs:element name="iva" type="xs:integer"/>
        <xs:element name="precio" type="Tdecimal"/>
        <xs:element name="descuento" type="Tdescuento"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
</xs:schema>
Tipos simples
...
  
```

■ Restricción numérica

Si desea especificarse que el descuento puede tener el valor entre 1 y 10, ambos incluidos, va a crearse un tipo simple llamado *ndescuento*, que permite introducir descuentos con el rango de valores [1..10]. Si, en el XML, se introduce en ese elemento un contenido no perteneciente al rango, estará bien formado, pero no será válido.

Lo primero es crear un tipo simple con una restricción. Para añadir una restricción, se usa el elemento *xs:restriction*, que define las restricciones del tipo. Dispone de un atributo *base*, que indica el tipo de datos.

```

<xs:simpleType name="Tdescuento">
  <xs:restriction base="xs:integer">
    <xs:minInclusive value="1"/>
    <xs:maxInclusive value="10"/>
  </xs:restriction>
</xs:simpleType>
  
```

El código tendrá una longitud entre 1 y 4.

```

<xs:simpleType name="Tlongitudentre">
  <xs:restriction base="xs:string">
    <xs:minLength value="1"/>
    <xs:maxLength value="4"/>
  </xs:restriction>
</xs:simpleType>
  
```

Después, se crea la restricción para que el precio tenga un total de 4 dígitos y hasta 2 decimales.

```
<xs:simpleType name="Tdecimal">
  <xs:restriction base="xs:decimal">
    <xs:totalDigits value="4"/>
    <xs:fractionDigits value="2"/>
  </xs:restriction>
</xs:simpleType>
```

■ Restricción longitud del contenido

Es preciso especificar que el código tendrá 4 caracteres.

```
<xs:simpleType name="Tlongitud">
  <xs:restriction base="xs:string">
    <xs:length value="4"/>
  </xs:restriction>
</xs:simpleType>
```

■ Restricción de enumeración

Hay que limitar el contenido de un elemento o atributo a un conjunto de valores posibles. Por ejemplo, la categoría puede valer A, B, C, D y E:

```
<xs:simpleType name="Tcat">
  <xs:restriction base="xs:string">
    <xs:enumeration value="A"/>
    <xs:enumeration value="B"/>
    <xs:enumeration value="C"/>
    <xs:enumeration value="D"/>
    <xs:enumeration value="E"/>
  </xs:restriction>
</xs:simpleType>
```

■ Restricción mediante expresiones regulares o patrones

Otra manera de presentar el ejemplo anterior es mediante el uso de patrones, donde el valor permitido es uno de los del rango de valores permitidos. Algunas de las expresiones regulares o patrones que pueden incluirse son:

- [0-9]: número del 0 al 9.
- [a-z]: letra minúscula perteneciente a ese intervalo.

- $[A-Z]$: letra mayúscula perteneciente a ese intervalo.
- $[aeiou]$: un carácter de los que aparecen entre los corchetes.
- $[^aeiou]$: un carácter que no aparezca entre corchetes.
- $\{X\}$: las llaves indican que tiene que repetirse X veces el contenido que haya delante.

```

<xs:simpleType name="Tcat1">
    <xs:restriction base="xs:string">
        <xs:pattern value="[A-E]" />
    </xs:restriction>
</xs:simpleType>

```

El XSD con todas las restricciones es el siguiente:

```

<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
    elementFormDefault="qualified">
    <xs:element name="bdproductos">
        <xs:complexType>
            <xs:sequence>
                <xs:element maxOccurs="unbounded" ref="producto" />
            </xs:sequence>
        </xs:complexType>
    </xs:element>
    <xs:element name="producto">
        <xs:complexType>
            <xs:sequence>
                <xs:element name="nombre" type="xs:string"/>
                <xs:element name="código" type="Tlongitudentre" />
                <xs:element name="categoría" type="Tcat1" />
                <xs:element name="iva" type="xs:integer" />
                <xs:element name="precio" type="Tdecimal" />
                <xs:element name="descuento" type="Tdescuento" />
            </xs:sequence>
        </xs:complexType>
    </xs:element>
    <xs:simpleType name="Tdescuento">
        <xs:restriction base="xs:integer">
            <xs:minInclusive value="1" />
            <xs:maxInclusive value="10" />
        </xs:restriction>
    </xs:simpleType>
    <xs:simpleType name="Tdecimal">
        <xs:restriction base="xs:decimal">
            <xs:totalDigits value="4" />
            <xs:fractionDigits value="2" />
        </xs:restriction>
    </xs:simpleType>
    <xs:simpleType name="Tcat">

```

[.../...]

```

<xs:restriction base="xs:string">
    <xs:enumeration value="A"/>
    <xs:enumeration value="B"/>
    <xs:enumeration value="C"/>
    <xs:enumeration value="D"/>
    <xs:enumeration value="E"/>
</xs:restriction>
</xs:simpleType>
<xs:simpleType name="Tcat1">
    <xs:restriction base="xs:string">
        <xs:pattern value="[A-E]"/>
    </xs:restriction>
</xs:simpleType>
<xs:simpleType name="Tlongitud">
    <xs:restriction base="xs:string">
        <xs:length value="4"/>
    </xs:restriction>
</xs:simpleType>
<xs:simpleType name="Tlongitudentre">
    <xs:restriction base="xs:string">
        <xs:minLength value="1"/>
        <xs:maxLength value="4"/>
    </xs:restriction>
</xs:simpleType>
</xs:schema>

```

Ejercicio resuelto 4.6

Elabora un patrón para DNI.

Solución 



5. Documentación de esquemas

La creación de un esquema puede ser algo complejo, por lo que introducir datos relacionados con la autoría, utilidad, derechos de autor, etc., es un aspecto que ha de tenerse en cuenta. Podría pensarse que, con poner comentarios, valdría, pero existen elementos destinados a tal fin.

Si necesita dotarse de información a los esquemas para que pueda ser consultada por otros usuarios, se usan las siguientes instrucciones:

- *xs:annotation*: especifica los comentarios dentro de un esquema que actúan como documentación de este. Tiene un atributo opcional (*id*) y dos elementos hijos (*appinfo* o *documentation*).
- *xs:appinfo*: se usa para especificar la información de la aplicación. Tiene un atributo opcional (*source*).

- xs:documentation: *añade comentarios en un esquema. Debe ir dentro de xs:annotation y tiene dos atributos optionales (source y xml:lang).*

```
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
<xs:annotation>
  <xs:appinfo>Lenguaje de marcas</xs:appinfo>
  <xs:documentation xml:lang="es">
    comentarios sobre
  </xs:documentation>
</xs:annotation>
...

```

Actividades propuestas



Responde si las siguientes afirmaciones son verdaderas o falsas y razona tu respuesta:

- V F 4.7.** La faceta *pattern* se usa para añadir restricciones mediante expresiones regulares.
- V F 4.8.** En los elementos complejos, existe la marca <xs:choice>, que indica que los hijos pueden aparecer en cualquier orden una sola vez, la secuencia aparecerá en el orden que se desee.

Verificar

B) Asociación XSD en XML

La asociación de un documento XSD a uno XML se realiza mediante un espacio de nombres con una serie de atributos:

- *xmlns*: definir el espacio de nombres, indicando que van a usarse los elementos definidos en la URL especificada. Los elementos del esquema llevarán el prefijo que se defina, en este caso, será xs.

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

Las referencias a los esquemas pueden realizarse mediante los siguientes atributos:

- *noNamespaceSchemaLocation*: se emplea cuando no se utilizarán espacios de nombres en el documento. Se usará un fichero con el esquema:

`xs:noNamespaceSchemaLocation = "longan.xsd"`

- *schemaLocation*: se emplea cuando se utilizan explícitamente los nombres de los espacios de nombres en las etiquetas.

Tal y como se aprecia en el ejemplo siguiente, el documento XML de bases de datos de alumnos hace referencia a un fichero de espacio de nombres previamente creado, xsdalumnos.xsd, el cual servirá para que sea validado el XML.

```
<?xml version="1.0" encoding="iso-8859-1"?>
<bdalumnos xmlns:xs="http://www.w3.org/2001/XMLSchema-instance" xs:noNa-
    mespaceSchemaLocation="xsdalumnos.xsd">
    <alumno>
        <nombre>José Antonio</nombre>
        <padre>José Luis</padre>
        <madre>Carmen</madre>
        <edad>18</edad>
        <ciudad>Málaga</ciudad>
    </alumno>
    <alumno>
        <nombre>Francisco Jesús</nombre>
        <padre>José</padre>
        <madre>María</madre>
        <edad>25</edad>
        <ciudad>15</ciudad>
    </alumno>
</bdalumnos>
```

El esquema xsdalumnos.xsd es:

```
<?xml version="1.0" encoding="iso-8859-1"?>
<xs:schema attributeFormDefault="unqualified" elementFormDefault="qualified"
    xmlns:xs="http://www.w3.org/2001/XMLSchema">
    <xs:element name="bdalumnos">
        <xs:complexType>
            <xs:sequence>
                <xs:element maxOccurs="unbounded" name="alumno">
                    <xs:complexType>
                        <xs:sequence>
                            <xs:element name="nombre" type="xs:string" />
                            <xs:element name="padre" type="xs:string" />
                            <xs:element name="madre" type="xs:string" />
                            <xs:element name="edad" type="xs:unsignedByte" />
                            <xs:element name="ciudad" type="xs:string" />
                        </xs:sequence>
                    </xs:complexType>
                </xs:element>
            </xs:sequence>
        </xs:complexType>
    </xs:element>
</xs:schema>
```

4.4. Herramientas de creación y edición XML

Existen multitud de herramientas que permiten crear y editar de manera fácil y amena documentos XML. Aunque las gratuitas permiten crear, editar y validar estos documentos y crear

esquemas, así como comprobar si están bien formados, las de licencia propietaria permiten la descarga de una versión de evaluación con un periodo determinado de tiempo y presentan muchas más prestaciones.

Algunas de ellas son Stylus Studio, Oxygen XML, Exchanger XML Editor, EditiX XML Editor, Liquid Studio y XMLBlueprint, entre otras.

4.4.1. XML Copy Editor

Software con licencia GNU disponible para plataformas Windows y Linux, permite la validación DTD. Resalta la sintaxis y permite plegado y la terminación de etiquetas.

4.4.2. Altova XMLSpy

Es uno de los editores de XML más vendidos, con avanzadas funciones de modelado, edición, transformación y depuración. Se permite su compra o realizar una evaluación de 30 días. Entre sus características, se destaca: Editor XML, cliente y depurador SOAP, editor gráfico de esquemas XML y DTD, editor WSDL gráfico, vista texto XML, vista cuadrícula XML, editor JSON y JSON Schema, analizador y generador de expresiones XPath, integración con Visual Studio y Eclipse, edición, depuración y generación de perfiles XQuery, etc.

4.4.3. XMLSpear

Editor XML libre que permite la validación en tiempo real. Construido en Java y disponible para todas las plataformas. Puede crearse un documento XML desde cero o abrir uno y añadir o eliminar nodos. Algunas características que han de destacarse: panel de búsqueda XPath; soporte de catálogo XML; traducciones XSLT; validación completa del esquema; editor de árbol para insertar, repetir y eliminar nodos; validación en tiempo real contra esquema o DTD durante la edición, etc.

La pantalla principal, en la parte inferior, permite ver el código en forma de elemento o tabla, lo que resulta de gran utilidad para tener una visión general del XML.

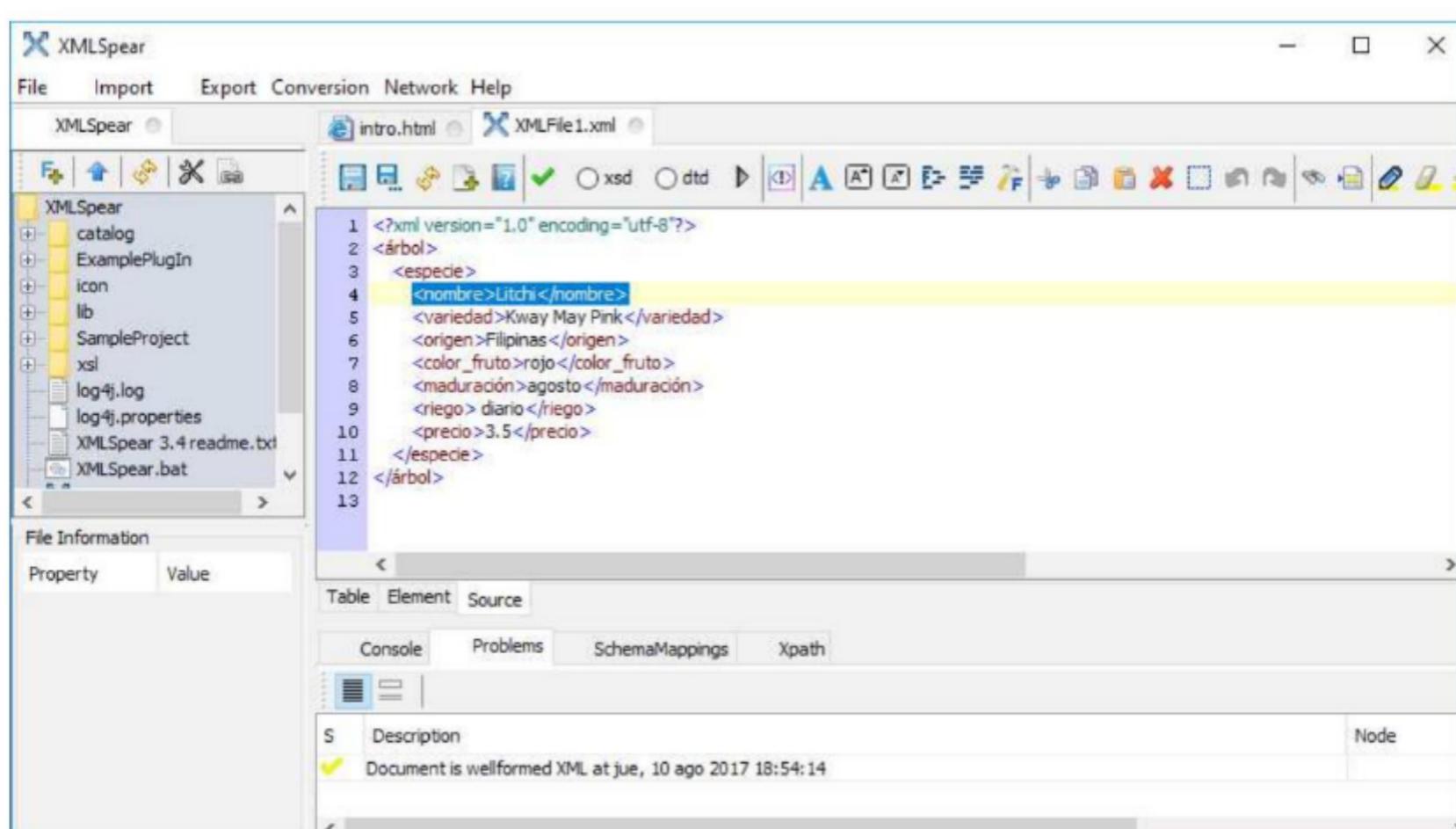


Figura 4.7. Interfaz de XMLSpear.

4.4.4. XML Notepad

Editor de XML con dos partes diferenciadas (código y estructura de árbol) que permite editar y visualizar documentos XML de manera fácil y amena.

4.4.5. Online XML Editor

Editor de XML que muestra la información en forma de árbol. El usuario va escribiendo las etiquetas en el panel izquierdo y, de manera automática, va apareciendo el árbol a mano derecha.

Recursos web

www

A través de estos enlaces podrás acceder a XML Copy Editor, Altova XMLSpy, XMLSpear, XML Notepad y Online XML Editor.



4.5. Validación

El método de validación es un proceso que permite comprobar que el contenido de un fichero XML presenta una estructura adecuada y cumple una serie de normas (tipo de elementos y atributos que pueden añadirse, orden de aparición, etc.).

Para construir un documento bien formado o sintácticamente correcto, hay que seguir las normas vistas en el capítulo 1. Un documento válido tiene que estar bien formado y cumplir una serie de requisitos especificados en una estructura DTD, XML Schema o mediante Relax NG.

RECUERDA

- ✓ Un documento puede estar bien formado, pero no ser válido. En cambio, si es válido, seguro que está bien formado.

Las herramientas que se encargan de realizar este proceso se denominan *validadores* o *parsers*. Puede validarse de varias maneras, como se detalla a continuación.

A) Usando línea de comando

XMLStarlet es una herramienta de línea de comandos que permite validar, transformar, editar, consultar, ficheros XML para sistemas operativos (Linux, Solaris, Windows, Mac OS X, etc.).

WWW

Recursos web

A través de los siguientes enlaces podrás acceder a dos herramientas *online* de validación: Validator W3 y XML Validation.



B) Aplicaciones de escritorio para validar

Existen numerosas aplicaciones de escritorio del ámbito de XML que ofrecen la opción de comprobar si un documento está bien formado y es válido. Para ello, simplemente, hay que introducir el documento XML y acceder a las opciones que se deseen. Algunos editores XML que ofrecen esta posibilidad son:

- XML Copy Editor.
- Stylus Studio.
- Liquid Studio.
- Oxygen XML Editor.
- EditiX XML Editor.

En la figura 4.8, se observan las opciones de validar y comprobar si un documento XML está bien formado usando la herramienta XML Copy Editor.

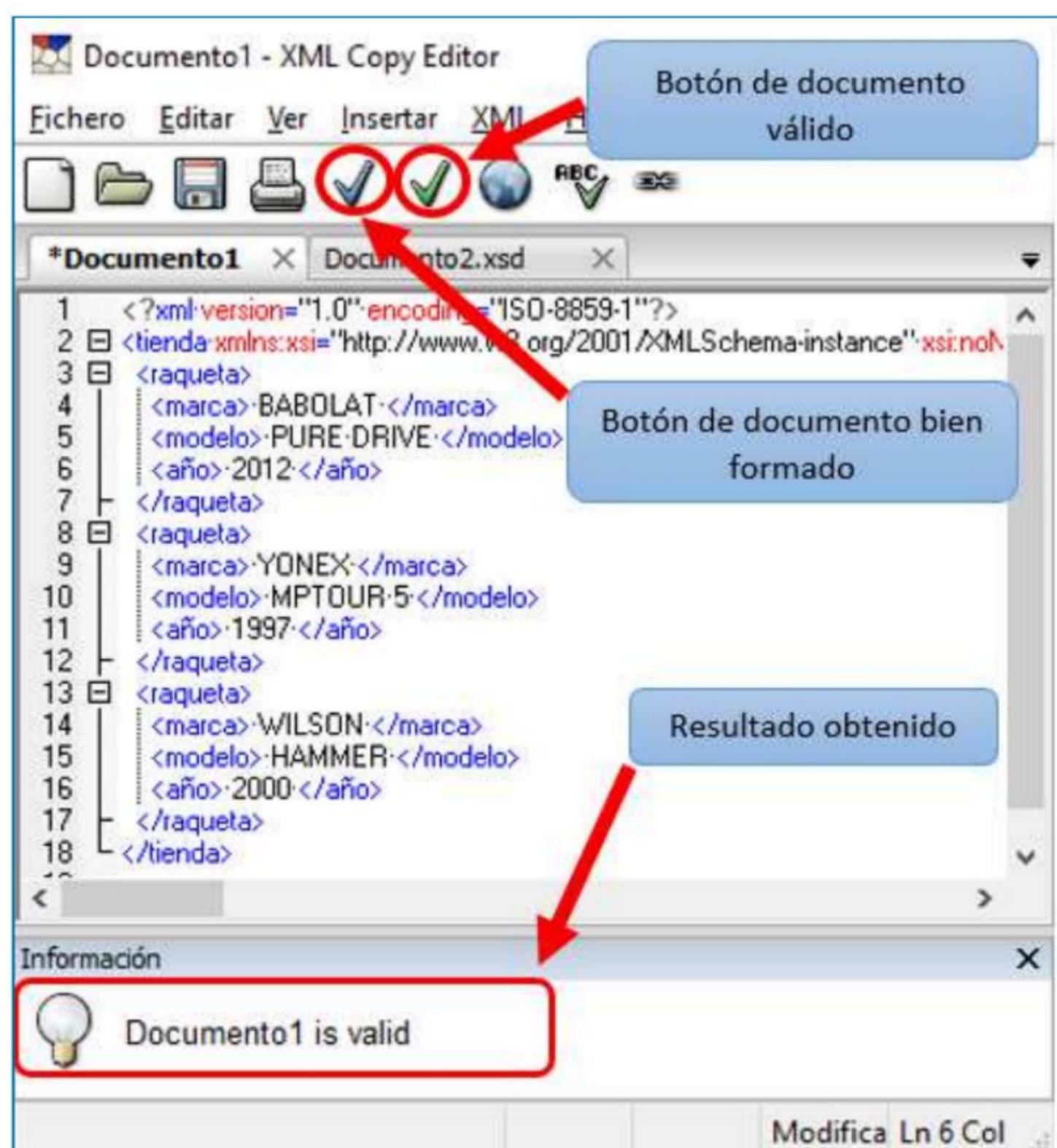


Figura 4.8
Validar un documento empleando XML Copy Editor.

Ejercicio resuelto 4.7



Valida el siguiente XML usando Liquid Studio.

XML:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<tienda>
  <raqueta>
    <marca>BABOLAT</marca>
    <modelo>PURE DRIVE</modelo>
    <año>2012</año>
  </raqueta>
  <raqueta>
    <marca>YONEX</marca>
    <modelo>MPTOUR 5</modelo>
    <año>1997</año>
  </raqueta>
  <raqueta>
    <marca>WILSON</marca>
    <modelo>HAMMER</modelo>
    <año>2000</año>
  </raqueta>
</tienda>
```

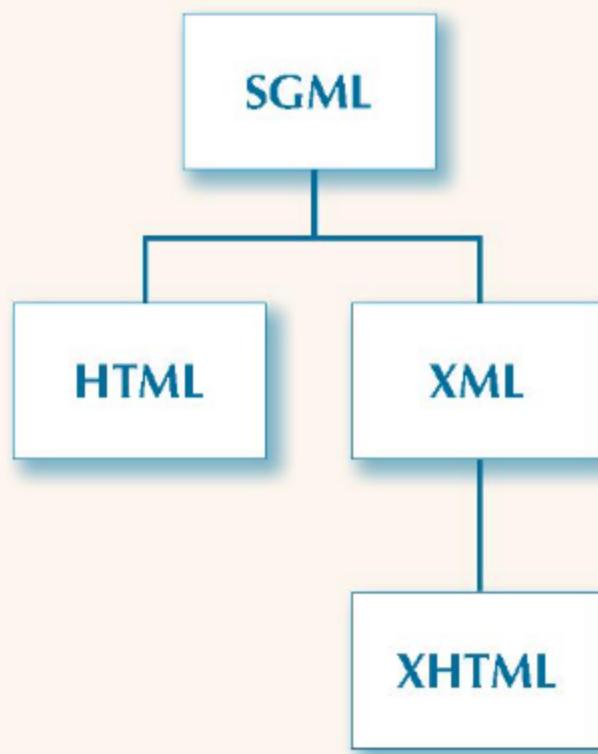
XSD:

```
<?xml version="1.0" encoding="iso-8859-1"?>
<xs:schema attributeFormDefault="unqualified" elementFormDefault="qualified" xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <xs:element name="tienda">
    <xs:complexType>
      <xs:sequence>
        <xs:element maxOccurs="unbounded" name="raqueta">
          <xs:complexType>
            <xs:sequence>
              <xs:element name="marca"
                type="xs:string" />
              <xs:element name="modelo"
                type="xs:string" />
              <xs:element name="año"
                type="xs:unsignedShort" />
            </xs:sequence>
          </xs:complexType>
        </xs:element>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
</xs:schema>
```

Solución

Resumen

■ Origen XML:



■ Estructura XML y sintaxis:

- El prólogo:
 - Versión.
 - *Encoding*.
- Elemento raíz o cuerpo:

```

<?xml version="1.0" encoding="utf-8"?>
<padre>
  <hijo>
    <subhijos> ... </subhijos>
  </hijo>
</padre>
  
```

■ En las marcas o etiquetas, se usan los signos menor que (<) y mayor que (>) para definirlas. Existen tres tipos de etiquetas o marcas:

1. Las de apertura.
2. Las de cierre.
3. Etiquetas vacías.

- Los atributos se especifican con el par NombreAtributo="valor".
- Los comentarios se expresan comenzando por <!-- y terminando con -->.
- Sección CDATA:

```
<![CDATA[ contenido dentro del bloque ]]>
```

- Creación, asociación y elementos de DTD y XSD.
- DTD (definición de tipo de documento):

— Elementos:

```
<!ELEMENT nombreElemento (elementos_hijo1, elemento_hijo2,...)>
```

— Atributos:

```
<!ATTLIST Elemento Atributo Tipo Valor>
```

— Entidades:

```
<!ENTITY nombre valor>
```

— Asociación DTD en XML:

- Interna:

```
<!DOCTYPE raíz [
    elementos
]>
```

- Externa:

```
<!DOCTYPE bdcorreo SYSTEM "midtd.dtd">
```

■ XML Schema XSD:

— Estructura:

```
<?xml version = "1.0" encoding = "utf-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema" elementForm-
Default="qualified">
    ...
    Elementos
    ...
</xs:schema>
```

— Elementos:

```
<xs:element Nombre Tipo Valor"/>
```

— Tipos de datos XSD:

Tipo	Descripción
<i>decimal</i>	Número decimal.
<i>boolean</i>	Un único valor lógico (1, 0).
<i>dateTime</i>	Fecha y hora en formato (AAAA-MM-DD T HH:MM:SS).
<i>string</i>	Texto o cadena de caracteres.
<i>date</i>	Fecha: año-mes-día (aaaa-mm-dd).
<i>time</i>	Hora: hh:mm:ss.
<i>integer</i>	Número entero positivo o negativo.
<i>positiveInteger</i>	Número entero positivo.
<i>long</i>	Entero de 64 bits.
<i>short</i>	Entero de 16 bits.

— Tipos de datos XSD:

Tipo	Descripción
<i>decimal</i>	Número decimal.
<i>boolean</i>	Un único valor lógico (1, 0).
<i>dateTime</i>	Fecha y hora en formato (AAAA-MM-DD T HH:MM:SS).
<i>string</i>	Texto o cadena de caracteres.
<i>date</i>	Fecha: año-mes-día (aaaa-mm-dd).
<i>time</i>	Hora: hh:mm:ss.
<i>integer</i>	Número entero positivo o negativo.
<i>positiveInteger</i>	Número entero positivo.
<i>long</i>	Entero de 64 bits.
<i>short</i>	Entero de 16 bits.

— Atributos:

`<xs:attribute NombreA TipoA USO ValorA"/>`

— Restricciones o facetas:

Faceta	Descripción
<i>length</i>	Contiene un atributo <i>value</i> donde se especifica un valor de longitud fija.
<i>minLength</i>	Contiene un atributo <i>value</i> donde se especifica el valor de longitud mínima.
<i>maxLength</i>	Contiene un atributo <i>value</i> donde se especifica el valor de longitud máxima.
<i>whiteSpace</i>	Contiene un atributo <i>value</i> donde se especifica un valor que puede ser (“preserve”, “replace” y “collapse”). Trata el uso de los espacios en blanco, retornos de carro, etc., que puedan aparecer.
<i>maxExclusive</i>	Contiene un atributo <i>value</i> donde se especifica el valor, que es menor al especificado.
<i>minExclusive</i>	Contiene un atributo <i>value</i> donde se especifica el valor, que es mayor al especificado.
<i>minInclusive</i>	Contiene un atributo <i>value</i> donde se especifica el valor, que es mayor o igual al especificado.
<i>maxInclusive</i>	Contiene un atributo <i>value</i> donde se especifica el valor, que es menor o igual al especificado.
<i>totalDigits</i>	Contiene un atributo <i>value</i> donde se especifica el valor, que es el número máximo de dígitos de un número teniendo en cuenta los decimales.
<i>fractionDigits</i>	Contiene un atributo <i>value</i> donde se especifica el valor, que es el número máximo de dígitos de decimales de un número.
<i>enumeration</i>	Contiene un atributo <i>value</i> donde se especifica uno de los valores admitidos de la lista.
<i>pattern</i>	Contiene un atributo <i>value</i> donde se especifica un rango de caracteres admitidos o expresión regular.

— Asociación XSD en XML:

```
<?xml version="1.0" encoding="iso-8859-1"?>
<bdalumnos xmlns:xs="http://www.w3.org/2001/XMLSchema-instance"
    xs:noNamespaceSchemaLocation="xsdalumnos.xsd">
```

■ Herramientas de creación y edición XML:

- XML Copy Editor.
- Altova XMLSpy.
- XMLSpear.
- XML Notepad.
- Online XML Editor.

■ Validación:

- Usando línea de comando.
- Herramientas *online*.
- Aplicaciones de escritorio.

Ejercicios prácticos resueltos

1. Cuenta el número de elementos del siguiente código XML:

```
<item>
    <title>Agregadores</title>
    <link>http://www.librolm.com/html</link>
    <description>Apuntes agregadores</description>
    <guid>B4BA124D-D535-4830-B28C-E0F0BD49D80E</guid>
    <pubDate>Tue, 26 Dec 2017 17:02:39 GMT</pubDate>
</item>
```

Solución 

2. Contesta a las siguientes preguntas:

- ¿Está bien formado el siguiente XML teniendo en cuenta su esquema?
- ¿Es válido el siguiente XML teniendo en cuenta su esquema?

XML:

```
<?xml version="1.0" encoding="iso-8859-1"?>
<bdproductos xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:noNamespaceSchemaLocation="zax.xsd">
    <producto>
        <nombre>teclado</nombre>
        [.../...]
```

```

<código>C001</código>
<iva>21</iva>
< categoría>A</ categoría>
<precio>15</precio>
<descuento>10</descuento>
</ producto>
< producto>
    < nombre>monitor</ nombre>
    < código>C002</ código>
    < categoría>B</ categoría>
    <iva>21</iva>
    <precio>147.2</precio>
    <descuento>9</descuento>
</ producto>
</ bdproductos>

```

Esquema:

```

<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema" elementFormDefault="qualified">
    <xs:element name="bdproductos">
        <xs:complexType>
            <xs:sequence>
                <xs:element maxOccurs="unbounded" ref="producto"/>
            </xs:sequence>
        </xs:complexType>
    </xs:element>
    <xs:element name="producto">
        <xs:complexType>
            <xs:sequence>
                <xs:element ref="nombre"/>
                <xs:element ref="código"/>
                <xs:element ref=" categoría"/>
                <xs:element ref="iva"/>
                <xs:element ref="precio"/>
                <xs:element ref="descuento"/>
            </xs:sequence>
        </xs:complexType>
    </xs:element>
    <xs:element name="nombre" type="xs:NCName"/>
    <xs:element name="código" type="Tlongitudentre"/>
    <xs:element name=" categoría" type="Tcat1"/>
    <xs:element name="iva" type="xs:integer"/>
    <xs:element name="precio" type="Tdecimal"/>
    <xs:element name="descuento" type="Tdescuento"/>
    <xs:simpleType name="Tdescuento">
        <xs:restriction base="xs:integer">
            <xs:minInclusive value="1"/>
            <xs:maxInclusive value="10"/>
        </xs:restriction>
    </xs:simpleType>
    <xs:simpleType name="Tdecimal">

```

[.../...]

```

<xs:restriction base="xs:decimal">
    <xs:totalDigits value="4"/>
    <xs:fractionDigits value="2"/>
</xs:restriction>
</xs:simpleType>
<xs:simpleType name="Tcat">
    <xs:restriction base="xs:string">
        <xs:enumeration value="A"/>
        <xs:enumeration value="B"/>
        <xs:enumeration value="C"/>
        <xs:enumeration value="D"/>
        <xs:enumeration value="E"/>
    </xs:restriction>
</xs:simpleType>
<xs:simpleType name="Tcat1">
    <xs:restriction base="xs:string">
        <xs:pattern value="[A-E]"/>
    </xs:restriction>
</xs:simpleType>
<xs:simpleType name="Tlongitud">
    <xs:restriction base="xs:string">
        <xs:length value="4"/>
    </xs:restriction>
</xs:simpleType>
<xs:simpleType name="Tlongitudentre">
    <xs:restriction base="xs:string">
        <xs:minLength value="1"/>
        <xs:maxLength value="4"/>
    </xs:restriction>
</xs:simpleType>
</xs:schema>

```

Solución 

3. Dado el siguiente documento XML, donde se almacenan datos relacionados de una librería con libros de informática, crea el documento DTD asociado.

XML:

```

<?xml version="1.0" encoding="iso-8859-1"?>
<librería>
    <libro>
        <título>Lenguaje de marcas</título>
        <autor>Autor2</autor>
        <autor>Autor1</autor>
        <ISBN></ISBN>
        <páginas>158</páginas>
        <precio>15</precio>
    </libro>
    <libro>
        <título>XML</título>
        <autor>Autor1</autor>
        <ISBN></ISBN>
        <páginas>278</páginas>
        <precio>30</precio>
    </libro>
</librería>

```

¿Qué modificación harías en el documento DTD para que puedan añadirse varios precios, teniendo en cuenta que puede haber libros en papel y *online*?

Solución 

4. Pon un ejemplo de los siguientes elementos:

- Elementos sin atributos con datos.
- Elementos con atributos y datos.
- Elementos con atributos sin datos.

Solución 

Ejercicios prácticos

1. Realiza una DTD para describir un conjunto de artículos, sabiendo que hay que almacenar el nombre, precio, código y fabricante.
2. Obtén el DTD y XSD del siguiente documento XML:

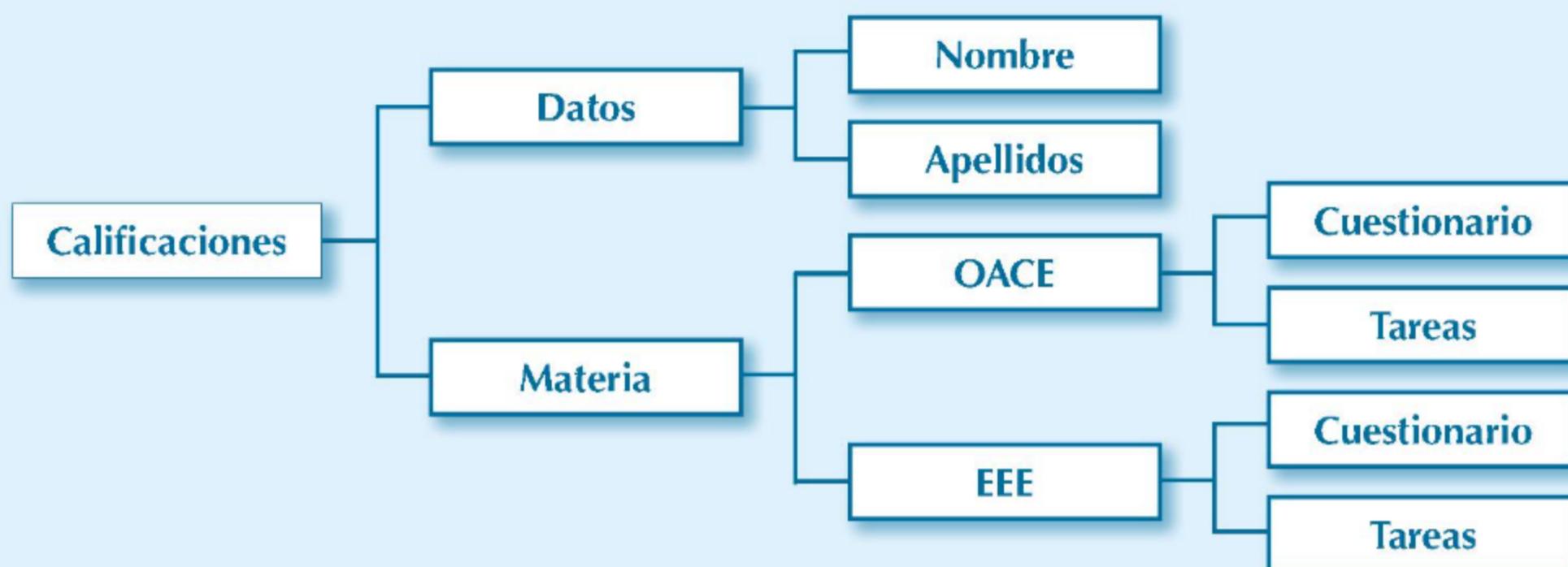
```
<?xml version="1.0" encoding="iso-8859-1"?>
<vivero>
    <especie siembra="2018">
        <nombre>Litchi</nombre>
        <precio moneda="euro">25</precio>
        <variedad>Kway May Pink</variedad>
        <origen>Filipinas</origen>
        <color_fruto>rojo</color_fruto>
        <color_fruto>rosa</color_fruto>
        <otros_datos>
            <maduración>agosto</maduración>
            <riego> diario</riego>
        </otros_datos>
    </especie>
    <especie siembra="2017">
        <nombre>Longan</nombre>
        <precio moneda="euro">10</precio>
        <variedad>Champoo</variedad>
        <origen> China</origen>
        <color_fruto>marrón</color_fruto>
        <otros_datos>
            <maduración>octubre</maduración>
            <riego> diario</riego>
        </otros_datos>
    </especie>
    <especie siembra="2016">
        <nombre>Litchi</nombre>
        <precio moneda="euro">21</precio>
        <variedad> mauritius</variedad>      [.../...]
```

```

<origen>Florida</origen>
<color_fruto>rojo</color_fruto>
<otros_datos>
    <maduración>agosto</maduración>
    <riego> diario</riego>
</otros_datos>
</especie>
</vivero>

```

3. Quiere implementarse en un documento XML la estructura de las calificaciones de los alumnos de un centro de Málaga, entre los que hay que guardar: datos personales (nombre y apellidos), materia (OACE y EEE) con sus respectivas tareas y cuestionarios, tal y como se muestra en la siguiente figura:



Los datos que han de introducirse son:

ALUMNOS						
Datos		Materia				
		OACE		EEE		
Nombre	Apellidos	Cuestionario	Tareas	Cuestionario	Tareas	
Felipe	Pascual	4	8	7	6	
María	García	9	2	9	6	

Una vez terminado, añade otra firma, completando con los valores que estimes oportuno.

4. Dado el siguiente documento XML, determina sus etiquetas, elementos y atributos:

```

<?xml version="1.0" encoding="utf-8" ?>
<agenda>
    <contacto nombre="alumno LM">
        <ciudad>Málaga</ciudad>
        <teléfono n="666777888"/>
        <email>666777888@prueba.com</email>
    </contacto>
</agenda>

```

5. Dado el siguiente fichero DTD, construye un documento XML con algunos datos:

```
<!ELEMENT Móviles (Movil+)
<!ELEMENT Movil (Modelo, precio, Fabricante, color)>
<!ELEMENT Modelo (#PCDATA)>
<!ELEMENT precio (#PCDATA)>
<!ELEMENT Fabricante (#PCDATA)>
<!ELEMENT color (#PCDATA)>
```

6. Quiere almacenarse la siguiente información en un XML, que contiene datos relativos a socios de un gimnasio, sabiendo que *género* e *id* son atributos.

- a) Crea el XML donde se almacena dicha información.

Persona		Nombre y apellidos	Teléfono	Email	Socio
Género	ID				
Femenino	001	Segundo Pascual	6XXXXXXXXX 95XXXXXXXX	User1@gmail.com User2@gmail.com User3@gmail.com	Sí
Masculino	002	Obdulia Toledo		Carmen@uma.es	No

- b) Elabora un fichero DTD asociado, para lo cual has de tener en cuenta que:

- Un usuario puede tener más de un *email* (+).
- Puede haber más de un teléfono por usuario, o ninguno (teléfono)*.
- Puede introducirse más de un usuario (persona)+.

7. Dado el siguiente XML:

```
<agenda>
  <contacto>
    <nombre_apellidos>Francisco Ruiz González</nombre_apellidos>
    <telefono>600222xxx</telefono>
    <edad>39</edad>
    <email>curro@gmail.com</email>
    <direccion>
      <calle>cómpeta 16</calle>
      <cod_post>12345</cod_post>
      <provincia>Málaga</provincia>
    </direccion>
  </contacto>
</agenda>
<agenda>
  <contacto>
    <nombre_apellidos>Francisco Ruiz González</nombre_apellidos>
    <telefono>600222xxx</telefono>
    <edad>39</edad>
    <email>curro@gmail.com</email>
  [.../...]
```

```

<direccion>
    <calle>cómpeta 16</calle>
    <cod_post>12345</cod_post>
    <provincia>Málaga</provincia>
</direccion>
</contacto>
<contacto>
    <nombre_apellidos>Pepe Escudero</nombre_apellidos>
    <telefono>954122xxx</telefono>
    <telefono>602222xxx</telefono>
    <edad>25</edad>
    <direccion>
        <calle>cristo</calle>
        <cod_post>29785</cod_post>
        <provincia>Málaga</provincia>
    </direccion>
</contacto>
<contacto>
    <nombre_apellidos>Carmen Ruiz García</nombre_apellidos>
    <edad>18</edad>
    <email>carmen@gmail.com</email>
    <email>carmen1@gmail.com</email>
    <email>carmen2@gmail.com</email>
    <direccion>
        <calle>canalejas</calle>
        <cod_post>25985</cod_post>
        <provincia>Cádiz</provincia>
    </direccion>
</contacto>
</agenda>

```

Completa el siguiente fichero DTD:

```

<!ELEMENT agenda (contacto_____)>
<!ELEMENT contacto (nombre_apellidos, telefono__, edad, email__, di-
    reccion)>
<!ELEMENT nombre_apellidos (#PCDATA)>
<!ELEMENT telefono (_____)>
<!ELEMENT edad (#PCDATA)>
<!ELEMENT email (#PCDATA)>
<!ELEMENT direccion (calle, cod_post, _____)>
<!ELEMENT calle (#PCDATA)>
<!ELEMENT cod_post (_____)>
<!ELEMENT provincia (#PCDATA)>

```

8. Dado el siguiente esquema XSD:

```

<?xml version="1.0" encoding="utf-8"?>
<xs:element name="agenda">
    <xs:complexType>
        [.../...]

```

```
<xs:sequence>
  <xs:element maxOccurs="unbounded" name="contacto">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="nombre_apellidos" type="xs:s-
          tring"/>
        <xs:element minOccurs="0" maxOccurs="2"
          name="telefono" type="xs:unsignedInt" />
        <xs:element name="edad" type="xs:unsignedByte" />
        <xs:element minOccurs="1" maxOccurs="3" na-
          me="email"
          type="xs:string" />
      </xs:sequence>
    </xs:complexType>
  </xs:element>
</xs:sequence>
</xs:complexType>
</xs:element>
</xs:schema>
```

Responde a las siguientes preguntas, argumentando el resultado:

- a) ¿Podrías añadir 7 contactos más a la agenda?
 - b) Si un usuario deja el campo teléfono sin rellenar, ¿estaría bien?
 - c) El usuario Ataulfo dispone de dos teléfonos móviles y uno fijo, ¿podrían añadirse todos en el XML?
 - d) ¿Cuántos correos electrónicos, como máximo y mínimo, hay que introducir?
 - e) ¿De qué tipo es el campo edad?
9. Enumera las principales diferencias entre DTD y XML.

AUTOEVALUACIÓN

