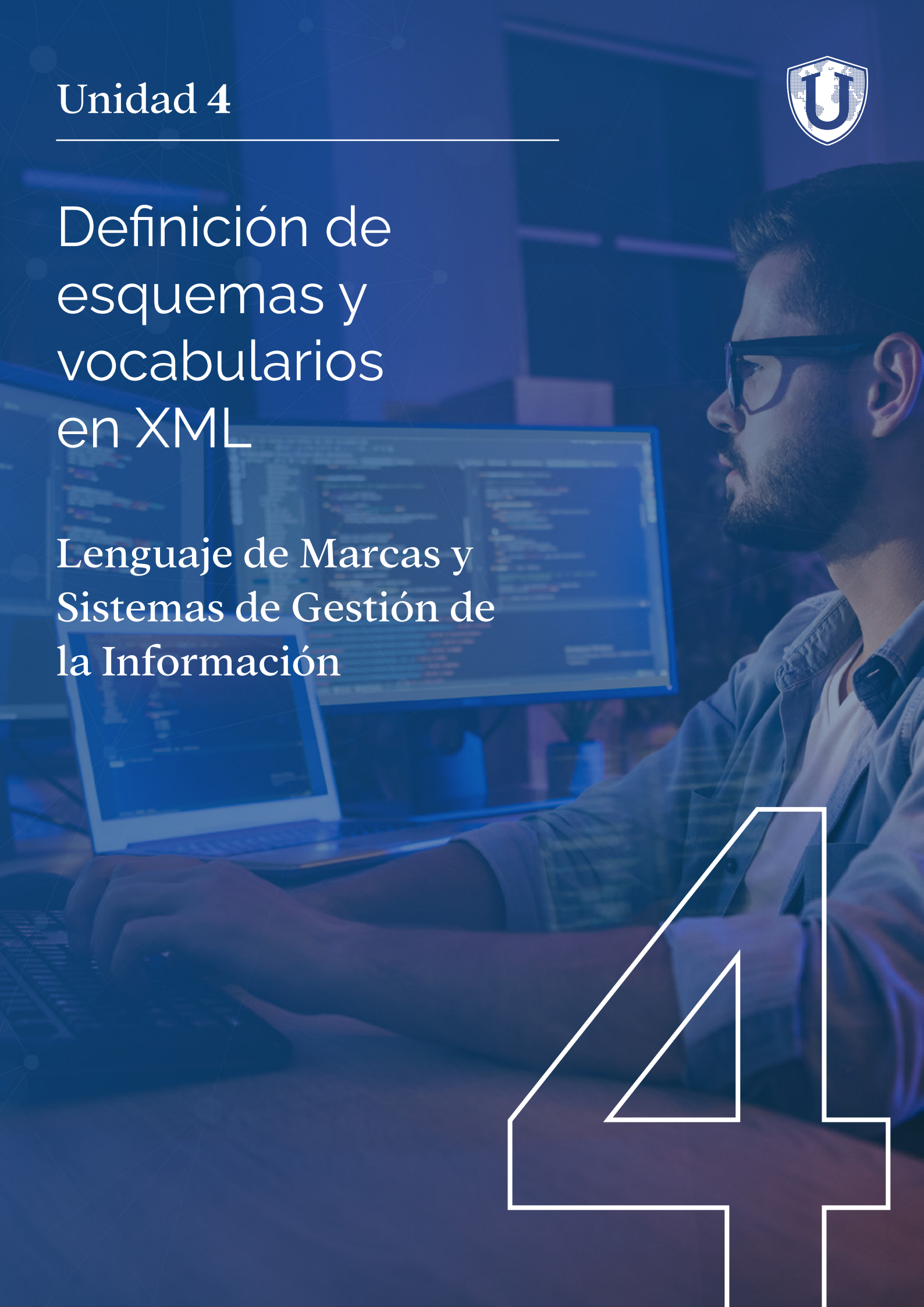


Unidad 4

Definición de esquemas y vocabularios en XML

Lenguaje de Marcas y Sistemas de Gestión de la Información

4

Índice

Introducción

En este capítulo podremos ver el lenguaje XML, extensible markup language, la cual sirve como sistema de intercambio, almacenamiento de información, actualización de software, etc.

Estudiaremos los distintos elementos que lo conforman y sus particulares características, como pueden ser su sintaxis, marcas, elementos, etc.

Aprenderemos como crear un texto XML, de forma correcta y estructurada, así como los distintos tipos de documentos usados para la modificación de los documentos XML, como son el DTD, document type definition, y el XSD, XML schema definition.

Descubriremos, no solo como crear archivos XML, sino también como validarlos y editarlos, para ello se mostrarán las diversas herramientas que tenemos a nuestro alcance con las que poder llevar a cabo esta función, desde herramientas online gratuitas, hasta descargables de pago, veremos distintos ejemplos con un breve resumen de sus características.

Con el fin de completar el recorrido del lenguaje XML, mostraremos la fabricación y funciones de los archivos DTD y XSD, los cuales sirven de como archivos de restricciones para el XML, así como la diferencia entre ambos.

Al finalizar esta unidad

- Estudiaremos el uso, las partes y la creación del leguaje XML.
- Describiremos los distintos tipos de documentos en el marco del XML, como los archivos DTD y XSD, y como asociándolos entre sí.
- Descubriremos las distintas herramientas con las que poder trabajar los códigos XML, ya sea para su creación o edición, así como las características avanzadas.
- Aprenderemos como validar los documentos XML con distintas herramientas online o de escritorio.

4.1. Definición de la estructura y sintaxis de documentos XML

El lenguaje de marcas XML, extensible markup language, derivado de SGML, standard generalized markup language, considerado un lenguaje de etiquetado, es un metalenguaje semejante al HTML que surge para resolver los problemas de este a partir del W3C, World Wide Web Consortium. Usa marcas no predefinidas lo que da una gran libertad para definir nuestras propias etiquetas y poder trabajar elementos específicos. Este lenguaje se ha extendido creando una multitud basados en él como son XHTML, MathML, SVG, XUL, XBL, RSS o RDF.

Imagen 1. El W3C es un consorcio internacional que genera recomendaciones y estándares que aseguran el crecimiento de la World Wide Web a largo plazo.

La visualización del código XML puede hacerse desde el navegador o desde programas especializados, como son las hojas de estilo CSS o transformaciones XSLT, entre otras.

La particularidad del código XML se encuentra en el hecho de que es compatible con una multitud de programas, lo que facilita el cambio de información, la personalización y, sobre todo, la reutilización de código, lo que ahorrará mucho tiempo y esfuerzo.

La estructura de un documento XML es jerárquica arborescente, es decir surge de un único elemento padre o raíz, el XML tan solo permite poseer una única raíz, y los hijos o ramas que surgen de dicha raíz. A su vez estas ramas actúan como una pseudoraíz para nuevas ramas, como se verá en el ejemplo. La extensión de este tipo de documentos es .xml.

```
<?xml version=”1.0” encoding=”utf-8”?>
<raíz>
        <rama>
                <subrama> Hola </subrama>
        </rama>
</raíz>
```

Comúnmente al elemento superior se le denomina padre y al inferior hijo.

Podemos ver que las líneas de código cuentan con dos partes claramente separadas:

- En primer lugar, la primera línea de código, llamada Prólogo o encabezado, marca la información necesaria con respecto a la naturaleza del propio documento, como es la versión utilizada o el tipo de codificación usado, ya sea Unicode o siguiendo las normas ISO.
- En segundo lugar, podemos encontrar el resto del documento, con una raíz inicial junto al resto de ramas dependientes de él que forman el cuerpo del documento.

4.1.1. Instrucciones de procesamiento o prólogo

El prólogo del documento XML es el que contiene la información necesaria para descifrarlo, por lo que es imprescindible para poder leerlo. En primer lugar, encontramos la versión XML usada, en segundo lugar, el encoding, el cual nos dice el tipo de código empleado para elaborar el cuerpo del documento, y, en tercer lugar, aunque no siempre se añade, encontramos el standalone, el cual, con un "yes" o "no", marca si el documente contiene fichero DTD o XSD adjuntos.

El prólogo siempre debe comenzar con los siguientes dos caracteres "<?" y termina de igual modo con los siguientes dos caracteres "?>".

Tras esta primera línea comenzaría el cuerpo del texto con la raíz.

Si el standalone es positivo a continuación se escribirá el tipo de documento adjunto, ya sea DTD o XSD.

Podemos ver a continuación un XML donde se recogen valores de razas de animales con la información correspondiente

```
<?xml version=”1.0” encoding=”iso-8859-1”?>
<perro>
	<raza>
		<nombre>dálmata</nombre>
		<origen>Dalmacia</origen>
		<color>blanco y negro</color>
		<peso_medio>24 kg</peso_medio>
		<altura_media>58 cm</altura_media>
	</raza>
</perro>
```

4.1.2. Marcas o etiquetas

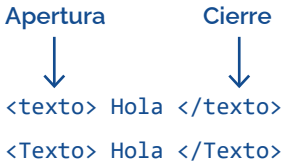
Una de las mayores ventajas del XML es, aunque se deba seguir la estructura, la capacidad de poder definir tus propias etiquetas, lo que lleva a una organización más específica en cada documento y, por tanto, más precisa.

Con el fin de que dichas etiquetas puedan ser usadas correctamente es imprescindible seguir la estructura de manera muy rígida y rigurosa. La estructura arbórea debe ir acompañada de sus correspondientes marcas, en este caso los signos de menor que "<" y mayor que ">" son imprescindibles para delimitar las etiquetas creadas. Se debe recordar que el cierre de la etiqueta se debe marcar con la propia etiqueta con una barra "/". El nombre de la etiqueta se escribirá acotado por los signos menos que y mayor que, y en el caso de cierre detrás de la barra.

Se debe tener en cuenta a la hora de escribir que el código XML es uno de los llamados case sensitive, lo que implica que son sensibles a las mayúsculas por lo que en los nombres de las marcas las mayúsculas también deben ser tomadas en cuenta.

Ejemplos de entradas y cierres

En el caso de encontrarse en la misma línea se abriría, se escribe el texto necesario, y se cierra añadiendo la barra.



En el caso de entradas abiertas y cerradas en distintas líneas de código deben escribirse siempre a la misma altura de tabulación, añadiendo sangrías tabuladas en función de la subordinación de la línea. Esto permite visualizar las entradas y salidas de manera fácil y simple con el fin de poder ver y corregir los posibles errores.

```
<Asunto>
        <Texto_1>Hola</Texto_1>
        <Texto_2>Adiós</Texto_2>
</Asunto>
```

Cualquier marca que contenga espacios, un número como inicio, use una barra que no sea la de cierre o no siga las reglas anteriores no será válida.

Incorrectas	Correctas	Causa
<Texto></texto>	<Texto></Texto>	Mayúsculas
<1Texto></1Texto>	<Texto1></Texto1>	Número inicial
<Texto 1></Texto 1>	<Texto_1></Texto_1>	Espacios
<Texto/1></Texto/1>	<Texto_1></Texto_1>	Uso de barra no final

Imagen 2. Tipos de errores en XML.

4.1.3. Elementos

Los elementos son el conjunto de etiquetas de entrada y cierre más lo que contenga, ya sea texto, otros elementos, atributos, etc.

El nombre de un elemento, recogido en las etiquetas de entrada y cierre, puede contener caracteres alfanuméricos, pero los únicos signos de puntuación que puede soportar son el guion "-", el guion bajo "_" y el punto ".", nunca al inicio.

En el elemento origen (<origen>Dalmacia</origen>) podemos ver sus partes principales:

- Etiqueta de apertura: <origen>
- Contenido (en este caso texto): Dalmacia
- Etiqueta de cierre: </origen>

En el caso del elemento raza podemos ver que este mismo contiene otros cinco elementos, también llamados subelementos:

```
<raza>
        <nombre>dálmata</nombre>
        <origen>Dalmacia</origen>
        <color>blanco y negro</color>
        <peso_medio>24 kg</peso_medio>
        <altura_media>58 cm</altura_media>
</raza>
```

Pueden existir elementos vacíos, sin contenido, sobre todo en casos dende se regula una estructura fija, aunque posea etiqueta sigue considerándose vacío. Estos elementos se pueden expresar de dos maneras:

+ <Texto></Texto>
+ <Texto/>
+ <Texto n="12"></Texto>
+ <Texto n="12"/>

4.1.4. Atributos

Si un elemento contiene un elemento con una propiedad específica, como un valor asignado, se debe adjuntar en la etiqueta de apertura. La forman de añadirlo sería la siguiente: Nombre del elemento, espacio, característica, signo de igual "=" y por último el valor o características entre comillas, si hubiese más atributos se añadirían con un espacio entre ellos. Como un ejemplo podemos ver los siguientes elementos.

```
<Texto Tipo="12">Hola</Texto>
<Plantas categoría="flores">Rosa</Plantas>
<Plantas categoría="arbustos"/>
```

Estos atributos añaden información al XML, pero su abuso puede ser contraproducente.

4.1.5. Comentarios

Pueden introducirse en cualquier parte del documento, en una línea independiente, con el fin de aclarar cosas, servir de recordatorio, dar información a otros lectores, etc. La sintaxis usada es la misma que la que HTML usa, iniciando con "<!--" y terminando con "-->".

```
<?xml version = "1.0" encoding = "utf-8"?>
<Libro>
       <!--Completar las entradas vacías-->
       <Título></Título>
       <Precio> 18€ </Precio>
       <Editorial></Editorial>
</Libro>
```

4.1.6. Sección CDATA

Muy semejante al comentario, permite añadir información al archivo sin que esta sea procesada, de modo que se pueda mantener en el archivo pero que no cause problemas con el código, es una buena solución para almacenar información que no se va a usar en la actualidad, pero que podría usarse en el futuro. Su sintaxis comienza con "<![CDATA[" y finaliza con "]]>", si se añaden elementos en su interior ira en líneas diferentes.

```
<![CDATA[Los archivos 1 y 2 no sirven]]>


<![CDATA[
<Texto> Hola </Texto>
]]>
```

Lo que permite guardar elementos sin que se procesen.

4.2. Creación, asociación y elementos de DTD y XSD

El documento XML en ocasiones no es suficiente para nuestras necesidades, por lo que es posible adjuntarle otros archivos con las restricciones necesarias para completar el archivo. Las restricciones usadas se pueden incluir de diversas maneras, una de ellas es la inclusión de archivos DTD, los cuales pueden llegar a resultar bastante rígidos, ya que no permiten definir los elementos. Otra opción, es el uso de XML Schema, el cual resulta mucho más fácil de usar dado que supera las limitaciones del DTD. Existen otras opciones, pero son menos usadas. Para la creación de XML Schema y DTD es posible usar el programa de código abierto Trang o el programa Oxygen XML Editor para conversión de distintos esquemas.

Para que estos ficheros de restricciones, una vez elaborados, puedan ejercer su función, es preciso asociarlos al documento XML.

Podemos encontrar numerosos generadores de código, ya sea online o descargables con los que poder trabajar los esquemas DTD y SXD como, por ejemplo, Oxygen XML Editor o XML Copy Editor.

4.2.1. DTD (definición de tipo de documento)

El DTD es un documento de texto plano que se adjunta al documento XML, como un archivo de restricciones, lo que permite especificar características o prefijar normas con las que el documento XML deberá trabajar, es decir, define la estructura de un XML. Estas normas pueden incluir desde atributos hasta el número de veces que puede aparecer un elemento.

El uso de documentos de restricción, como es el DTD, se aconseja en especial si un mismo documento va a ser trabajado por diversas personas, ya que este creara homogeneidad, evitando conflictos o errores en el código, ya que sirve como unas instrucciones de uso.

Creación y elementos de una DTD

Se aconseja crear el DTD antes que el XML, de modo que desde un principio se siga una misma línea organizativa. Por lo que es imprescindible, aunque aún no se haya comenzado con el XML, tener muy claras las reglas a imponer. Los archivos resultantes siempre tendrán la terminación .dtd.

El DTD se constituye de tres componentes principales: elementos, atributos y entidades.

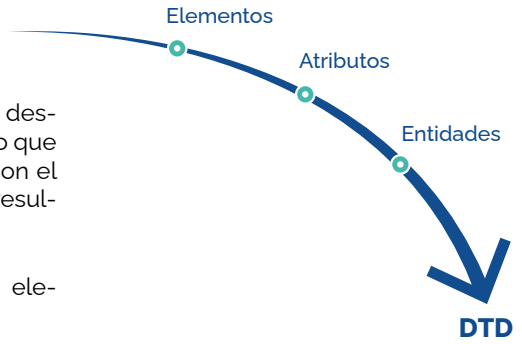


Imagen 3. Componentes de los archivos DTD.

1. Elementos

Los elementos son la base del DTD y deben de mantener una rígida estructura entre elementos superiore, padre, e inferiores, hijos. Su escritura comienza con "<!ELEMENT" y finaliza con un ">" La secuencia es la siguiente: <!ELEMENT, espacio, nombre del elemento padre, cierre con ">" o si hubiera elementos hijos, espacio, apertura de paréntesis "(", nombre del elemento hijo, coma, espacio, Segundo elemento hijo, así cuantos haya, cierre de paréntesis ")" y cierre ">". Por ejemplo:

```
<!ELEMENT plantas (flores, arbustos, helechos)>
```

A los elementos se les debe añadir una tipología específica, la cual puede ser una de estas 4 any, empty, #pcdata o #cdata, y se marca con un espacio tras el nombre del elemento y el tipo específico. Por ejemplo:

```
<!ELEMENT plantas any>
```

La descripción de los tipos son los siguientes:

- **#PCDATA:** parsed character data, el contenido se compone de texto y es analizado por un parser o analizador.
- **#CDATA:** character data, PCDATA que no puede analizarse con un parser.
- **ANY:** los elementos pueden contener cualquier valor.
- **EMPTY:** los elementos no tienen contenido, es un elemento vacío.

Si el tipo se aparece entre paréntesis indica que el contenido del elemento es analizable con un parser.

Los elementos con hijos se marcan de la siguiente manera:

```
<!ELEMENT plantas (flores, helechos>
<!ELEMENT flores (PCDATA)>
<!ELEMENT helechos (PCDATA)>
```

Podemos marcar el número de veces que un elemento aparecerá en el XML de la siguiente manera:

- **?** = El elemento aparece 0 o 1 vez.
- ***** = El elemento aparece 0, 1 o varias veces.
- **+** = El elemento aparece 1 o varias veces.
- **|** = Marca dos elementos entre los que es necesario elegir.

Algunos ejemplos de esto se pueden ver en la siguiente línea DTD.

```
<!DOCTYPE inventario [
<!ELEMENT inventario (plantas)+>
<!ELEMENT plantas (especie, (precio | no_disponible))>
<!ELEMENT especie (#PCDATA) #REQUIRED>
<!ELEMENT precio (#PCDATA)>
<!ELEMENT no_disponible (#PCDATA)>
]>
```

Que aparecerán en el XML como:

```
<inventario>
<plantas>
              <especie>Rosa</especie>
              <precio>1 €</precio>
       </plantas>
       <plantas>
              <especie>clavel</especie>
              <precio>0.75 €</precio>
       </plantas>
<plantas>
              <especie>Tulipán</especie>
              <no_disponible>N/D</no_disponible>
       </plantas>
</inventario>
```

Que se acabaría viendo de la siguiente manera:

Rosa 1 €

clavel 0.75 €

Tulipán N/D

2. Atributos

Los atributos en una DTD no son muy flexibles por lo que, salvo para puntualidades o necesidad, no se recomienda. En caso de necesitar añadir más de un atributo es preciso separarlos mediante espacios, tabulaciones o cambios de líneas. Su elaboración se iniciará con "<!ATTLIST" seguirá de la siguiente manera: espacio, elemento, espacio, atributo, espacio, tipo, espacio, valor y cierre con ">".

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

↓

```
<!ATTLIST nombre dni CDATA #FIXED>
```

Tipo	Definición
CDATA	Valor alfanumérico.
Enumeraciones	Especifica varios valores *
ENTITY	El atributo ya está predefinido
IDREF	Refiere a otro elemento
IDREFS	Refiere a un conjunto (1|2|3)
ID	Contenido único e inequívoco
NOTATION	Con datos que no son XML
ENTITIES	Conjunto o lista de entidades
NMTOKENS	Con el puedes definir valor del atributo con ciertos caracteres

Imagen 4. Tipos de los atributos.

```
*<!ATTLIST libro volumen(1|2|3|4) CDATA #REQUIRED>
```

Valor	Definición
#FIXED	Valor fijo e inflexible
#IMPLIED	El atributo es opcional
#DEFAULT	El valor debe especificarse, pero no es fijo
#REQUIRED	El atributo es obligatorio

Imagen 5. Valores de los atributos.

Así, iel ejemplo <!ATTLIST nombre dni CDATA #FIXED>, implica que el elemento nombre lleva asociado un atributo, el dni, que se deberá escribir en un valor alfanumérico, el cual será fijo y no se podrá modificar.

3. Entidades

Las entidades son usadas para referenciar elementos prefijados en un DTD, que sirven como marcadores de posición. En el DTD se marcan con la apertura "<!ENTITY" y se cierra con ">", con el valor entre comillas, quedando de la siguiente manera.

```
<!ENTITY nombre “valor”>      →      <!ENTITY email “1234@gmail.com”>
```

Que en el XML aparecerá con el nombre de la entidad precedido por "&" y seguido por ";", quedando un elemento XML que referencie una entidad de DTD de la siguiente manera:

```
<Texto> &email; </Texto>      →      1234@gmail.com
```

De este modo añadiéndole varias entidades, separadas por espacios, podemos preconfigurar acciones y respuestas:

```
<!ENTITY email586 “1234@gmail.com”>
<!ENTITY error “Se ha producido un error en la cuenta”>
<!ENTITY alarma “Atención:”>
```

DTD ↑ ↓ ↓ XML

```
<suceso> &alarma; &error; &email586; </suceso>
```

Que quedaría de la siguiente forma:

Atención: Se ha producido un error en la cuenta 1234@gmail.com

Una segunda parte de este entorno es la entidad de caracteres. W3 Consortium define entidad de caracteres como: "La entidad de documento sirve como la raíz de la entidad árbol y el punto de partida de un procesador XML." Son referencias predefinidas a ciertos caracteres. Podemos encontrar tres tipos de entidades:

Caracteres predefinidos		Caracteres numéricos			Caracteres con nombre	
Significado	Código	Significado	Código decimal	Código hexadecimal	Significado	Código
Ampersand	&	&	&	&	Á	Aacute
Comilla simple	'	'	'	'	ù	ugrave
Mayor que	>	>	>	>		
Menor que	<	<	<	<		
Comillas dobles	"	"	"	"		

Imagen 6. Ejemplos de diferentes entidades.

Estos caracteres permiten usar dentro de las etiquetas caracteres especiales, de modo que en el texto la línea se vería como:

```
<Texto>&lt&lt casa &gt&gt</Texto>  →  << casa >>
```

Asociación DTD en XML

Las maneras dos maneras existentes de vincular un DTD al XML es introducirlo dentro de este último o adjuntarlo como un archivo externo, ya sea en el mismo servidor o en otro externo. Con el fin de manipulación y claridad se prefiere el uso de la segunda forma, ya que permite la consulta y manipulación del DTD, sin pasar por todo el XML, y a su vez permite usar un mismo DTD para distintos XML, en este caso en particular, esto permite que, ante un error, solo se necesite una única corrección, en lugar de la ardua tarea de localizar el error en cada uno de los DTD insertos.

Para declarar una DTD inserta, se seguiría el siguiente esquema: La segunda línea tras el prólogo se escribiría de la siguiente forma "<!DOCTYPE" seguido de espacio, el elemento raíz, espacio, y, entre corchetes, los distintos elementos, en las siguientes líneas y con el cierre de ">":

```
<?xml version=”1.0” encoding=”iso-8859-1”?>
<!DOCTYPE flora [
<!ELEMENT flora (plantas)+>
<!ELEMENT plantas (flores, precio)>
<!ELEMENT flores (#PCDATA)>
<!ELEMENT helechos (#PCDATA)>
]>
<flora>
        <plantas>
                <flores>Rosa</flores>
                <precio>1€</precio>
        </plantas>
        <plantas>
                <flores>Clavel</flores>
                <precio>0,75€</precio>
        </plantas>
</flora>
```

En el caso

Para asociar un archivo DTD, se debe crear por separado en un documento externo, es decir, deberá crear dos ficheros independientes, uno para el DTD y otro para el XML.

En el DTD solo hay que introducir los elementos que lo forman, por otro lado, en el XML, para asociarlo, es necesario adjuntar, detrás del prólogo, la línea “<!DOCTYPE”, espacio, el elemento raíz, espacio “SYSTEM” o “PUBLIC”, dependiendo de si es un DTD privado o público, espacio y el nombre del fichero DTD entre comillas.

Como ejemplo si el XML anterior hubiera contenido el DTD adjunto y no inserto, y este se llamase “inventario 2”, más la extensión .dtd, el resultado hubiese sido el siguiente:

```
<!DOCTYPE flora SYSTEM “inventario 2.dtd”›
```

Si se tratase de un DTD público tras “PUBLIC” se debe adjuntar, separado por espacios y entre comillas, el identificador FPI, formal public identifier, del archivo DTD.

4.2.2. XML Schema XSD

Como alternativa al DTD, podemos encontrar el XSD, XML schema definition, el cambio se debe a que los archivos XSD permiten especificar tipos de elementos y añadir restricciones y atributos más complejos. Además de lo ya mencionado es destacable que su sintaxis es similar a XML y permite un mayor control sobre las ocurrencias de los elementos que los archivos DTD. Con él es más fácil realizar conversiones de datos.

Los XSD son archivos de texto plano con la extensión .xsd.

Creación y elementos de un XSD

La raíz del archivo XSD será "<xs:schema>" y el resto poseerá una estructura semejante a XML. Este archivo raíz puede contener uno de los siguientes atributos:

- **xmlns:alias.** Se especifica de donde provienen los elementos y tipos usados, "http://www.w3.org/ 2001 / XMLSchema". El alias puede ser xs o xsd, mientras se use el mismo en todo el documento no importa, ejemplo:

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

  Por lo tanto, las siguientes líneas siempre comenzarán con el prefijo "<xs:"

- **elementFormDefault.** Aclara si se debe añadir los valores "qualified" y "unqualified" a los elementos. Por defecto "qualified":

  ```
  elementFormDefault="qualified"
  ```

  Lo que indica que los elementos utilizados por el documento XML que aparezcan en este esquema deben estar calificados para el espacio de nombres.

- **attributeFormDefault.** Aclara si se debe añadir los valores "qualified" y "unqualified" a los atributos.

- **targetNamespace.** Se especifica de donde se extraen los nombres de los elementos definidos, "https://www.w3schools.com".

- **version.** Se especifica la versión del documento de esquema.

Los atributos siempre se separarán de su valor con un igual (=) y estos valores irán entre comas.

Un ejemplo, sin elementos, es el siguiente:

```
<?xml version="1.0"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
targetNamespace="https://www.w3schools.com"
xmlns="https://www.w3schools.com"
elementFormDefault="qualified">

...
</xs:schema>
```

1. Elementos

En los archivos XSD el elemento schema siempre será la raíz, y por tanto se abrirá y cerrará con él. Este elemento contendrá los atributos ya mencionados.

Podemos encontrar elementos simples, que solo pueden contener texto, sin atributos u otros elementos subordinados. Que aparecerá como:

```
<xs:elemento nombre=”X” tipo=”Y”>
</xs:elemento>
```

En este elemento la “X” sería el nombre del elemento e “Y” el tipo de datos que el elemento contiene.

Los tipos más comunes de elementos son los siguientes:

- xs:string
- xs:decimal
- xs:integer
- xs:boolean
- xs:date
- xs:time

```
<xs:element name=”Inscripción” type=”xs:date”/>
```

Se les puede añadir un valor predeterminado o fijo a los elementos simples. Este valor puede ser por defecto (default) que permite modificaciones o un valor fijo obligatorio (fixed) que no las permite.

```
<xs:element name=”flores” type=”xs:string”
default=”Rosa”/>
```

Los atributos son opcionales de forma predeterminada, salvo que se use el atributo “use” seguido por su valor, los atributos obligatorios:

Se pueden añadir restricciones a los atributos con la entrada <xs:restriction base”atributo”>. Puedes consultar más información sobre este asunto en: https://www.w3schools.com/xml/schema_facets.asp

- **Indicadores de ocurrencias:**

  La cantidad de veces que puede o debe aparecer un elemento en el XML.

  - maxOccurs: Siendo por defecto 1, especifica el número máximo de veces que un elemento puede aparecer, en caso de no poseer limitación su valor sería “unbounded”.
  - minOccurs: Siendo por defecto 1, especifica el número mínimo de veces que un elemento debe aparecer, en caso de no poseer limitación su valor sería “unbounded”.

> Elementos complejos

Pueden ser elementos vacíos o contener elementos hijos, con contenido y contenido vacío, atributos, etc.

Los elementos vacíos aparecen de la siguiente forma:

```
<xs:element name=”Texto”/>
```

La aparición de estos elementos se marca con “<xs:complexType>” el indicador de orden y los elementos que contenga. El indicador Orden especifica el orden de aparición de los distintos elementos. Puede ser de tres tipos:

- <xs:sequence>: todos los elementos hijos según la secuencia indicada.
- <xs:choice>: solo uno de los indicados.
- <xs:all>: todos sin importar el orden.

```
<xs:element name=”flora”>
      <xs:complexType>
            <xs:sequence>
                  <xs:element name=”Flores” type=”xs:string” default=”Rosa”/>
                  <xs:element name=”Arbustos” type=”xs:string” default=”Cardo”/>
            </xs:sequence>
      </xs:complexType>
</xs:element>
```

2. Tipos de datos XSD

Los tipos de datos más comunes son los siguientes:

Tipo de dato	Descripción
decimal	Números decimales
boolean	Valor lógico
dateTime	Año-mes-díaThora:minuto:segundo (se incluye una “T” intercalada)
string	Cadena de caracteres, texto
date	Año-mes-día
time	Hora:minuto:segundo
integer	Número entero tanto positivo como negativo

Imagen 7. Tipo de datos.

> Creación de tipos de datos simples personalizados

En el caso de necesitar crear un nuevo tipo de dato se usará la construcción <xs:simpleType> con el atributo "name", con la que podremos crear un nuevo tipo de dato a nuestro gusto, especificando las características y restricciones que queramos añadirle.

```
<xs:simpleType name="geolocalización">
…
</xs:simpleType>
```

3. Restricciones o facetas

Las restricciones, o facetas, se pueden asociar tanto a tributos y elementos, aunque ya se han visto se ampliarán a continuación:

Restricción	Descripción del valor
length	Una longitud fija
minLenght	Una longitud mínima
maxLenght	Una longitud máxima
maxExclusive	Valor menor al especificado
minExclusive	Valor mayor al especificado
maxInclusive	Valor menor o igual al especificado
minInclusive	Valor mayor o igual al especificado
totalDigits	Número máximo de dígitos, incluidos decimales
fractionDigits	Número máximo de dígitos de decimales de un número
enumeration	Se selecciona uno de los valores admitidos en una lista predefinida
pattern	se especifica un rango de caracteres admitidos

Imagen 8. Restricciones XSD

> Restricción numérica

A la hora de crear restricciones las más simples son las numéricas, de modo que, si queremos crear una restricción de 0 a 100, decimales ambos incluidos, con tres dígitos máximo y dos decimales, debemos hacerlo de la siguiente manera:

```
<xs:simpleType name="Limitación de notas">
  <xs:restriction base="xs:integer">
        <xs:minInclusive value="0"/>
        <xs:maxInclusive value="100"/>
  </xs:restriction>
</xs:simpleType>
<xs:simpleType name="Limitación de decimales
de notas">
  <xs:restriction base="xs:decimal>
        <xs:totalDigits value="5"/>
        <xs:fractionDigits value="2"/>
  </xs:restriction>
</xs:simpleType>
```

> Restricción longitud del contenido

Para restringir la limitación del contenido usaremos el siguiente código, en este caso para introducir un DNI.

```
<xs:simpleType name="DNI">
  <xs:restriction base="xs:string">
        <xs:length value="9"/>
  </xs:restriction>
</xs:simpleType>
```

> Restricción de enumeración

Para limitar las posibles elecciones de valores se realizaría de la siguiente forma:

```
<xs:simpleType name="Estado">
  <xs:restriction base="xs:string">
        <xs:enumeration value="No empezado"/>
        <xs:enumeration value="En proceso"/>
        <xs:enumeration value="Parado"/>
        <xs:enumeration value="Completado"/>
  </xs:restriction>
</xs:simpleType>
```

- **Restricción mediante expresiones regulares o patrones**

  Los patrones, o expresiones regulares, nos permiten elegir una opción de un rango de valores y van precedidos por "pattern". Los más utilizados son:

```
<xs:simpleType name="Número de participantes">
  <xs:restriction base="xs:integer">
        <xs:pattern value="[0-6]"/>
  </xs:restriction>
</xs:simpleType>
```

Patrones	Descripción
[0-9]	Permite elegir uno de los caracteres que engloban, incluidos los extremos. En el caso de las letras esta elección se realizará en minúscula o mayúscula en función de lo que estipule el patrón.
[a-z]	
[A-Z]	
[aeiou]	Uno de los caracteres que aparecen
[^aeiou]	Uno de los caracteres que no aparecen
[X]	El contenido que lo precede se repetirá las veces indicadas por X: [5], [10]…

Imagen 9. Patrones y expresiones regulares.

4. Documentación de esquemas

Con el fin de que su consulta y archivo se facilite XSD incluye entradas con las que marcar la autoría de un documento, estas son:

- **xs:annotation:** determina que comentarios de un esquema contienen su documentación, para facilitar esta labor cuenta con el atributo opcional "id" y los elementos hijos "appinfo o documentation".
- **xs:documentation:** dentro de xs:annotation añade comentarios en un esquema, permite la inclusión de los atributos opcionales (source y xml:lang).
- **xs:appinfo:** especifica la información de la aplicación, permite el atributo opcional "source".

```
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema ">
  <xs:annotation>
        <xs:appinfo>Curso de informática</xs:appinfo>
        <xs:documentation xml:lang="es">
        Este curso está destinado a…
        </xs:documentation>
  </xs:annotation>
…
</xs:schema>
```

Asociación XSD en XML

Los XSD se pueden asociar al espacio de nombres desde donde hemos extraído lo nombres de elementos atributos, etc. ya construidos. Esto se realiza en su raíz con el atributo "xmlns":

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

En el caso de su asociación con el XML se debe realizar con los siguientes atributos:

- **noNamespaceSchemaLocation:** en el caso de que no se usen espacios de nombres, pero si un fichero de esquema previamente establecido, cuyo nombre se colocará entre comillas.

```
xs:noNamespaceSchemaLocation = "inventario.xsd"
```

- **schemaLocation:** se usan los nombres de los espacios de nombres en las etiquetas.

4.3. Herramientas de creación y edición XML

Las herramientas con las que se puede elaborar un XML son numerosas, tanto descargables como online, gratuitamente, las de pago además de permitir su creación suelen poseer un verificador de errores que facilitará mucho la tarea en documentos extensos.

Algunas de las más conocidas son: Stylus Studio, Oxygen XML o Exchanger XN1L Editor.

4.3.1. XML Copy Editor

Disponible tanto para plataformas Windows como Linux, este software con licencia GNU, también conocida como General Public License, permite la validación DTD. Destaca en el resaltado de la sintaxis, el plegado de los elementos y la terminación de etiquetas.

4.3.2. Altova XMLSpy

Es el editor de XML más vendidos. Destacando por sus avanzadas funciones de modelado, edición, transformación y depuración es considerado uno de los más completos y actualizados. Algunas de sus elementos característicos son:

- Editor JSON y Editor de esquemas JSON.
- Transformación JSON con XPath, XSLT, XQuery.
- Editar documentos XML.
- Validación XML smartFix y corrección de errores.
- Editor de esquemas XML.
- Editor XSLT.
- Depurador y perfilador XSLT.
- Optimizador de velocidad XSL.

4.3.3. XMLSpear

XMLSpear es un editor XML gratuito, construido en Java y disponible para todas las plataformas, con validación en tiempo real. Algunas de sus características son las siguientes:

- Panel de búsqueda Xpath (versión 3.0).
- Compatibilidad con catálogos XML (versión 3.0).
- Traducciones XSLT (versión 3.0).
- Validación completa del esquema mediante DOM3 en Xerces 2.9.0.
- Editor de árbol para insertar, repetir (versión 3.0) y eliminar nodos.
- Validación en tiempo real contra esquema o DTD durante la edición.
- Resolución interactiva de ubicaciones de esquemas o DTD.

4.3.4. XML Notepad

XML Notepad es un completo editor de XML con una interfaz muy intuitiva de doble ventana, texto y estructura de árbol, lo cual permite visualizar y editar documentos en XML con mayor facilidad. Entre sus características más particulares destacan:

- El soporte para drag-and-drop.
- La opción de "undo" infinito
- La posibilidad de personalizar fuentes y colores.

4.3.5. Online XML Editor

Editor de XML con forma de arbórea de doble ventana, donde el usuario escribe las etiquetas en el panel izquierdo y, al escribir, va apareciendo en forma de árbol a mano derecha. Su manejo es simple e intuitivo y como su nombre indica es una herramienta online gratuita.

4.4. Validación

La validación permite "poner a prueba" los archivos XML con el fin de detectar posibles fallos o carencias. A los programas que se encargan de esta labos se les conoce como validadores o parsers.

Las maneras de validación son las siguientes:

Usando línea de comando

Usados de manera online. Entre las múltiples herramientas de este tipo que podemos encontrar en la web destaca la herramienta XMLStarlet.

Aplicaciones de escritorio para validar

Son múltiples las aplicaciones descargables que nos permiten, introduciendo el archivo XML, realizar un proceso de validación con el que poder detectar los errores encontrados en el documento, de modo que podamos verlos y corregirlos. Los más destacados son:

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

www.universae.com