

MP_0373.
**Lenguajes de marcas y sistemas de
gestión de información**

**UF3. Lenguajes de Almacenamiento y
transmisión de información**

3.2. XML

Índice

☰	Objetivos	3
☰	Partes de un documento XML	4
☰	Jerarquía de elementos	7
☰	Documentos bien formados	8
☰	Herramientas	11
☰	DTD	13
☰	Elementos	16
☰	Atributos	19
☰	Entidades	22
☰	Notación	24
☰	XML Schema	25
☰	Definición	27
☰	Elementos simples	28
☰	Atributos y restricciones	30
☰	Resumen	31

Objetivos

Los objetivos de esta unidad son los siguientes:

- 1 Identificar las tecnologías relacionadas con la definición de documentos XML.
- 2 Analizar la estructura y sintaxis específica utilizada en la descripción.
- 3 Crear descripciones de documentos XML.
- 4 Utilizar descripciones en la elaboración y validación de documentos XML.
- 5 Conocer el esquema XML (XSD, *XML Schema Definition*).

Partes de un documento XML

Recuerda que XML no es un lenguaje de etiquetas, sino un conjunto de reglas para definir lenguajes de etiquetas, como XHTML.

Las etiquetas en XML se definen entre paréntesis angulares “< ... >”. Los documentos XML se componen de etiquetas y de contenido.

- El contenido está formado por los datos o textos sobre los que se aplican las marcas.
- Las etiquetas son la forma fundamental de los documentos de marcado.

Existen seis tipos de etiquetas:

Elementos y atributos

Son las partes en las que se divide el documento. Crean una estructura jerárquica para indicar qué elementos contienen otros elementos.

Se suele utilizar el parentesco para identificarlo (hijo de..., padre de..., hermano de...).

Los atributos son características de una etiqueta, pertenecen a la propia etiqueta que lo contiene y no son heredables.

Referencia de entidad

Las referencias de entidad se utilizan para escribir en el contenido del documento caracteres que son difíciles de escribir o no están en el teclado. Para ello se utilizan 3 formas:

- Por el nombre de entidad. Siempre debe empezar con & y terminar con el punto y coma. Por ejemplo, el símbolo de libra se puede poner como £. Al procesar el documento que contenga esa expresión aparecerá el nombre de libra. La forma de escribirlo sería:
&nombre;
- Por el nombre decimal. Funciona de forma similar al anterior, pero en lugar de indicar el nombre del carácter se introduce su codificación en ASCII. La forma de escribirlo sería:
&#número;
- Por el valor hexadecimal. Funciona de forma similar, pero aquí se indica el número en formato hexadecimal.
&#xnúmero;

Comentarios

Los comentarios son elementos que no son ni contenido ni ningún elemento de marcado. Permiten hacer anotaciones para que sea más legible y entendible el documento XML sin procesar.

Para indicar un elemento se comienza con “`<!--`” y se terminan con “`-->`”. Un comentario no puede estar dentro de una etiqueta `<...>`.

`<!-- Esto es un ejemplo y no tendrá repercusión cuando se procese el documento. -->`

Instrucciones de procesamiento

Las instrucciones de procesamiento son etiquetas que proporcionan información, datos o instrucciones a una aplicación. Cuando se utiliza el documento estas instrucciones serán interpretadas por la aplicación que procesa el XML. Al igual que los comentarios, no forman parte de la presentación de XML.

Para escribir una instrucción de procesamiento se utiliza la siguiente forma:

`<?Nombre instrucción ?>`

El *nombre* identifica la aplicación que debe interpretar la instrucción. Los nombres que comienzan con XML se reservan para las instrucciones de normalización de XML.

CDATA

Las secciones **CDATA** identifican una sección en la que no se interpretarán los caracteres. Se utiliza para poder mostrar caracteres en el documento procesado que son expresiones o palabras reservadas en XML.

El formato para escribir una sección CDATA es:

<! [CDATA[*caracteres reservados*]]>

Declaraciones de tipo de documento

Las declaraciones de tipo de documento (DTD) sirven para la descripción de la estructura y la sintaxis del documento. Aquí se definirán las etiquetas que se van a usar en la estructura del documento, qué elementos pueden incluir otros elementos, etc.

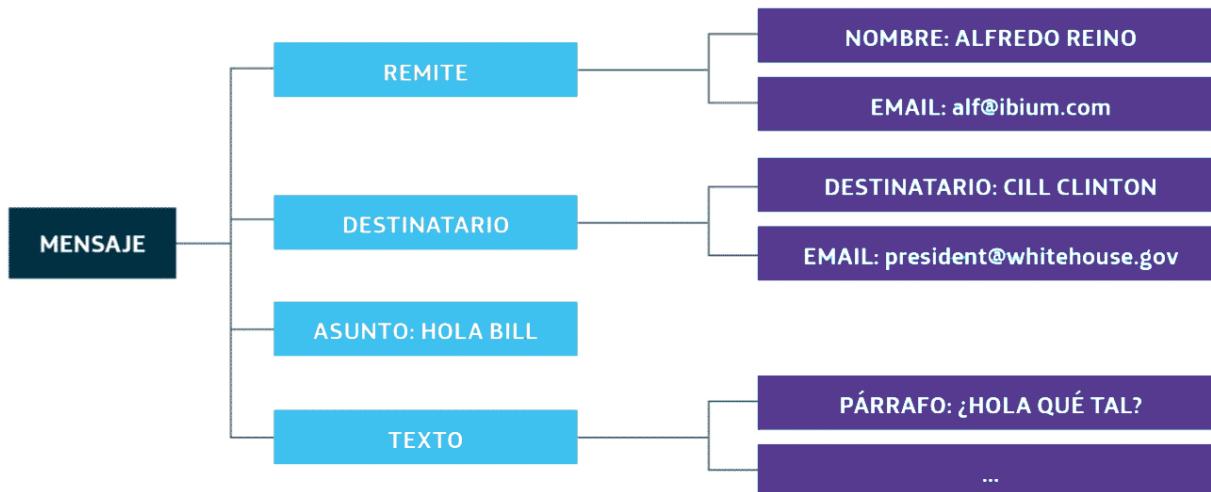
Una instrucción del DTD siempre tiene la forma <!Valor Instrucción>, donde:

- *Valor* es una palabra clave que permite identificar el elemento que se va describir.
- *Instrucción* es el código para el valor.

Más adelante veremos cómo escribir las instrucciones en la DTD del documento.

Jerarquía de elementos

El siguiente XML muestra la estructura jerárquica de elementos, representada también en la imagen.



```
<?xml version="1.0" encoding="UTF-7"?>
<!DOCTYPE mensaje SYSTEM "mensaje.dtd">
<mensaje>
  <remite>
    <nombre>Alfredo Reino</nombre>
    <email>alf@ibium.com</email>
  </remite>
  <destinatario>
    <nombre>Bill Clinton</nombre>
    <email>president@whitehouse.gov</email>
  </destinatario>
  <asunto>Hola Bill</asunto>
  <texto>
    <parrago>¿Hola qué tal? ...</parrago>
  </texto>
</mensaje>
```

Documentos bien formados

Un documento bien formado es aquel que cumple estrictamente con las siguientes normas para los elementos.

1

Los elementos deben seguir una estructura jerárquica. Cada elemento debe estar contenido dentro de otro, exceptuando el elemento raíz.

2

Los elementos deben estar alineados; deben contener tabulaciones de forma que en la columna que tenemos la etiqueta de inicio de un elemento, debe estar también la de cierre. Esto hace legible los documentos sin procesar al ojo humano.

3

Los elementos no pueden estar superpuestos. Un elemento que está contenido en otro elemento debe contener dentro de este elemento su etiqueta de inicio y final.

4

Los elementos deben estar cerrados, es decir, todos los elementos deben contener su etiqueta de inicio y de cierre.

5

Solo puede existir un elemento raíz, del que parten los demás.

6

El nombre de los elementos debe empezar siempre con un carácter no numérico y nunca puede comenzar por XML o cualquier variación, es decir, no puede empezar por xml, Xml, XML, xML, etc.

7

Hay que **respetar las mayúsculas y minúsculas**. XML es un lenguaje *case sensitive* (sensible a mayúsculas y minúsculas). No es lo mismo la etiqueta <*EtiQueta*> que la etiqueta <*eTiqueTa*>.

8

Los espacios y retornos de carro solamente son tenidos en cuenta si aparecen en el valor de un atributo.

9

No se pueden utilizar de "manera normal" los caracteres y palabras claves de XML, que son: &, <>, "".

Documento válido

La especificación XML en W3C establece que cualquier programa que procese un XML debe dejar de hacerlo si encuentra un error. Sin embargo, un documento HTML seguirá siendo procesado, aún con errores.



Un documento XML es válido si está bien formado y además cumple con las definiciones que se han mostrado en la DTD o en cualquier otro elemento de validación.

Para poder validar un documento XML podemos utilizar diferentes métodos:

- **DTD:** lo que realiza es una comprobación del cumplimiento del DTD definido.
- **Schema:** es un lenguaje más avanzado que la DTD y permite especificar la estructura y contenidos del documento.
- **Relax NG:** es un lenguaje de esquema basado en la gramática.
- **Schematron:** utiliza expresiones de acceso.

Pueden ser descritos al principio del documento o a través de un documento externo, cuyos enlaces se describen en el prólogo del XML. Ejemplo:

```
<?xml:version="1.0"?>
```

Validador XML

Editor web para validar XML.

[VALIDADOR XML](#)

Validador HTML

Editor web para validar HTML.

[VALIDADOR HTML](#)

Ejemplo de documentos mal formados

```
<?xml: version="1.0"?>
<base_datos nombre="MiEmpresa", tipo="Oracle", propietario="user1">
  <tabla nombre="Sucursales", propietario="user1">
    <columna nombre="nSucursal", tipo="Integer">
    </columna>
  </tabla >
</base_datos>
<autor nombre="jdedef">
</autor>
```

Este documento está mal formado porque existen dos elementos raíz y no respeta las tabulaciones.

```
<?xml: version="1.0"?>
<base_datos nombre="MiEmpresa", tipo="Oracle", propietario="user1">
  <tabla nombre="Sucursales", propietario="user1">
    <columna nombre="nSucursal", tipo="Integer">
  </tabla>
</base_Datos>
</coluMNA>
```

Es un documento mal formado porque no respeta el anidamiento ni las mayúsculas y minúsculas.

Herramientas

Para poder confeccionar un documento XML y HTML de manera correcta se pueden usar herramientas que revisan la sintaxis mientras se confecciona el documento.

Entre estas herramientas existen varios editores de XML que facilitan la tarea de crear documentos válidos y bien formados, pues advierten de los fallos de sintaxis de manera automática.

Las **funciones mínimas** que debería tener un editor de XML son:

- Ser capaz de leer la DTD del documento y presentar una lista *drop-down* con todos los elementos disponibles en la propia DTD de forma enumerada; de esa forma se evita incluir elementos no definidos.

- Avisar del olvido de etiquetas que son de uso obligatorio, incluso no permitiendo que se pueda continuar con la edición y salvaguarda del documento.

Algunos editores:

- **XML Pro** de Vervet Logic (open source).
- **XMLSpy** de Altova.
- **Liquid XML Studio** de Liquid Technologies.
- **<oXigen/> XML Editor**.
- **Turbo XML** de TIBCO (Plataforma de desarrollo integrado de XML).
- **XML Notepad** de Microsoft.
- **XMLwriter** de Wattle Software.

DTD

Al igual que en SGML, en XML los DTD son **archivos de texto que encierran una definición formal** de un tipo de documento y, a la vez, tienen la función de **especificar la estructura lógica de un archivo XML**, proporcionando los nombres de los elementos, atributos y entidades que utiliza, además de **suministrar la información sobre cómo se pueden usar conjuntamente**.

Un DTD es un **conjunto de reglas sintácticas** para definir etiquetas.

Nos indica qué etiquetas se pueden usar en un documento, en qué orden deben aparecer, cuáles pueden aparecer dentro de otras, cuáles tienen atributos, etc.

Crear una definición del tipo de documento (DTD) es como crear nuestro propio lenguaje de marcado para una aplicación específica. Por ejemplo, podríamos crear un DTD que defina una tarjeta de visita.

A partir de ese DTD, tendríamos una serie de elementos XML que nos permitirían definir tarjetas de visita.

Este lenguaje formal permite a los procesadores analizar automáticamente un documento e identificar de dónde viene cada elemento y cómo se relacionan entre ellos, para que los navegadores, las hojas de estilo, los motores de búsqueda, las bases de datos o las aplicaciones de impresión, entre otras, puedan utilizarlos.

El DTD del lenguaje XML es opcional. En tareas sencillas no es necesario construir una DTD, entonces se trataría de un documento "bien formado" (*well-formed*) y si lleva DTD será un documento "validado" (*valid*).

DTD especifica la clase de documento

- Describe un formato de datos.
- Usa un formato común de datos entre aplicaciones.
- Verifica los datos al intercambiarlos.
- Verifica un mismo conjunto de datos.

DTD describe

- **Elementos:** cuáles son las etiquetas permitidas y cuál es el contenido de cada etiqueta.
- **Estructura:** en qué orden van las etiquetas en el documento.
- **Anidamiento:** qué etiquetas van dentro de cuáles.

Sintaxis

El documento XML suele comenzar por una instrucción de procesamiento para el procesador de XML. En esta instrucción se indica el tipo de normalización que sigue el documento según el estándar XML y la codificación.

```
<?XML version="1.0" encoding="UTF-8"?>
```

Después suele comenzar el DTD, que puede ser referenciado por un enlace o descrito manualmente en el propio documento. Para definir el DTD utilizamos la sintaxis:

```
<!DOCTYPE[  
    definiciones  
>
```

Donde en *definiciones* se incluyen todas las declaraciones que puede contener el DTD. Si se referencia a un estándar externo se utiliza la palabra SYSTEM de la siguiente forma:

```
<!DOCTYPE etiqueta SYSTEM "documento.dtd">
```

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE ligaDeFutbol [
  <!ELEMENT ligaDeFutbol (partido*)>
  <!ELEMENT partido EMPTY>
  <!ATTLIST partido localNombre CDATA #REQUIRED>
  <!ATTLIST partido localGoles CDATA #REQUIRED>
  <!ATTLIST partido visitanteNombre CDATA #REQUIRED>
  <!ATTLIST partido visitanteGoles CDATA #REQUIRED>
]>

<ligaDeFutbol>
  <partido localNombre="Nottingham Presa" localGoles="0" visitanteNom-
bre="Inter de Mitente" visitanteGoles="1" />
  <partido localNombre="Vodka Juniors" localGoles="3" visitanteNombre="S-
parta da Risa" visitanteGoles="3" />
  <partido localNombre="Water de Munich" localGoles="4" visitanteNom-
bre="Esteaua es del grifo" visitanteGoles="2" />
</ligaDeFutbol>
```

Ejemplo de XML con DTD.

Elementos

Define todos los elementos que podemos encontrar a lo largo del documento, es decir define las etiquetas del documento.

<!ELEMENT tag(tag1, tag2)>

Si queremos describir un elemento que contenga otro elemento pondremos entre paréntesis los elementos que se pueden introducir.

<!ELEMENT tag(tag1 | tag2, tag3 +, tag4 *)>

- Si está permitido que aparezca más de una vez pondremos el signo "+" detrás del nombre.
- Si un elemento es opcional pondremos "?".
- Si un elemento es opcional y puede repetirse pondremos "*".
- Con "|" indicamos que el elemento es opcional, pero tendremos que elegir entre uno de ellos.

Para los elementos que no contienen a otros elementos indicaremos si están vacíos (*Empty*), cualquier contenido (*Any*) o solo datos ([#PCDATA]).

Ejemplos:

```
<!ELEMENT texto #PCDATA>
<!ELEMENT salto EMPTY>
<!ELEMENT extra ANY>
```

<!ELEMENT elemento (#PCDATA, elemento1)>

También se permite el uso de elementos mixtos.

```
<?xml version="1.0"?>
<!DOCTYPE note [
  <!ELEMENT note (to,from,heading,body)>
  <!ELEMENT to (#PCDATA)>
  <!ELEMENT from (#PCDATA)>
  <!ELEMENT heading (#PCDATA)>
  <!ELEMENT body (#PCDATA)>
]>
<note>
  <to>Tove</to>
  <from>Jani</from>
  <heading>Reminder</heading>
  <body>Don't forget me this weekend</body>
</note>
```

Ejemplo.

En el ejemplo anterior podemos ver que el DTD define las etiquetas que estarán dentro de la etiqueta `<note>`, no permitiendo el uso de las no indicadas (*to,from,heading,body*).

También definimos que cada una de las etiquetas contendrá datos (#PCDATA).

De este modo, si falta alguno de los elementos indicados o el contenido no es el indicado, el XML no se validará.

Saber más

En este enlace encontrarás más ejemplos de aplicación de ELEMENT.

[ELEMENTOS](#)

Atributos

Los atributos permiten añadir información adicional a los elementos de un documento. La principal diferencia entre los elementos y los atributos es que los atributos no pueden ser anidados (no tienen hijos).

Se usan para añadir información corta, sencilla y desestructurada a una etiqueta.

<!ATTLIST Elemento NomAtributo Tipo Valpredeterminado>

Con los atributos podemos especificar cuál será el contenido adicional del elemento.

Para definir el tipo del atributo que contendrá el elemento tenemos:

CDATA

Los atributos CDATA (*character data*) son los más sencillos, y pueden contener casi cualquier cosa. **!CDATA** hace que los analizadores lo ignoren.

ID

Nombre que identifica inequívocamente a la etiqueta.

IDREF

Sirve para referenciar una etiqueta que se ha definido en el ID.

ENTITY

(entities) Es el valor de una entidad o entidades separadas por coma.

NMTOKEN

(nmtokens) Se parecen a CDATA, pero solo aceptan denominar cosas con caracteres validos (letras, números, puntos, guiones, subrayados y los dos puntos).

Lista de nombres

Si se especifica una lista de nombres separados por comas indica que al usar el atributo se debe seleccionar uno de esa lista.

También podemos definir el tipo de valores predeterminado para los elementos:

**#REQUIRED**

Indica que el atributo es requerido siempre que se use la etiqueta.

**#VALOR**

Indica un valor que se usará por defecto si no se especifica el valor del atributo.

**#IMPLIED**

Indica que no se quiere que el atributo coja el valor por defecto.

**#FIXED**

Indica que el atributo siempre tiene el valor, aunque no se especifique.

```
<!ATTLIST curso
director CDATA #REQUIRED
horario (mañana | tarde | noche)
#IMPLIED
instalaciones (Exes | externas)
"Exes">
```

Ejemplo de aplicación ATTLIST.

Saber más

En este enlace encontrarás más ejemplos de ATTLIST.

[ATTLIST](#)

Entidades

Las entidades permiten colocar trozos de contenido en cualquier lugar del documento haciendo referencia a su nombre. Para definir una entidad usamos la etiqueta ***!ENTITY***.

Existen diferentes entidades:

- Internas.
- Externas y parámetros.

Entidades internas

También llamadas **macros o constantes de texto**. Las entidades internas son las que se asocian a una cadena de caracteres. Se referencian únicamente desde el propio fichero, de ahí su nombre, internas.

`<!ENTITY tag "Jaun Perez">`

Son abreviaturas que se usan dentro del documento para simplificar. Cuando se procese el documento estas abreviaturas serán sustituidas por el nombre completo. Para definirlas se usa:

```
<!ENTITY tag "texto sustituyo">
```

Para referenciar la entidad dentro del documento usamos "&nombre_entidad", donde cada vez que usemos &ISO en el documento aparecerá el nombre completo.

```
<!ENTITY ISO "International Organization for Standardization">
```

Entidades externas y parámetros

Las **entidades externas** son similares a las internas, con la diferencia de que obtienen el contenido fuera del documento, pudiendo ser incluso imágenes o archivos de sonido. Se utiliza la palabra SYSTEM para indicar que es externo:

```
<!ENTITY entity-name SYSTEM "URI/URL">
```

Los **parámetros** solo se pueden referenciar en la DTD del documento. Se utiliza % antes del nombre para referenciar y se definen de la forma:

```
<!ENTITY %parametro "texto">
```

Saber más

En este enlace puedes ampliar la información sobre !ENTITY.

[!ENTITY](#)

Notación

Este tipo de atributo permite al autor declarar que su valor se ajusta a una notación declarada.

```
<!NOTATION nombre SYSTEM "notación">
```

```
<!ATTLIST mensaje fecha NOTATION (ISO-DATE | EUROPEAN-DATE) #REQUIRED>
```

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<!DOCTYPE frutas [
  <!ELEMENT frutas (fruta)*>
  <!ELEMENT fruta EMPTY>
  <!ATTLIST fruta foto ENTITY #REQUIRED>

  <!ENTITY manzana SYSTEM "manzana.gif" NDATA gif>
  <!ENTITY naranja SYSTEM "naranja.gif" NDATA gif>

  <!NOTATION gif SYSTEM "image/gif">
]>

<frutas>
  <fruta foto="manzana"/>
  <fruta foto="naranja"/>
</frutas>
```

Ejemplo XML con DTD en el propio documento.

XML Schema

Un esquema, al contrario que un DTD, puede definir tipos de datos, lo cual es claramente beneficioso en el intercambio de datos, objetos o bases de datos.

Schema es más potente que el DTD clásico de XML, ya que permite más flexibilidad. Es un estándar recomendado por el W3C (Word Wide Web Consortium).

El DTD o declaración del tipo de documento y los esquemas proporcionan la **gramática** para una clase de documentos XML.

Estos mecanismos se utilizan para la llamada validación, tanto estructural como formal del documento, esto es, enviar un documento a un destinatario junto con las condiciones que deben cumplir los documentos.

En la actualidad los esquemas se utilizan en mayor medida que los DTD dentro de XML.

Un problema que presentan los DTD es que no siguen una **sintaxis XML**, sino una propia. Un grupo de desarrolladores ha propuesto una alternativa a los DTD, llamada esquemas ("schemas").

Básicamente, un esquema establece las reglas de un documento e indica qué etiquetas se pueden usar, sus atributos, las relaciones entre etiquetas, etc.

Ejemplo

En el siguiente ejemplo podemos ver que mediante el *Schema*, no solo definimos los elementos que contendrá nuestro documento, sino que también estamos definiendo el tipo de datos.

```
<?xml version="1.0"?>
<xsschema xmlns:xs="http://www.w3.org/2001/XMLSchema">

<xs:element name="note">
<xs:complexType>
<xs:sequence>
<xs:element name="to" type="xs:string"/>
<xs:element name="from" type="xs:string"/>
<xs:element name="heading" type="xs:string"/>
<xs:element name="body" type="xs:string"/>
</xs:sequence>
</xs:complexType>
</xs:element>
```

Ejemplo de *XML Schema Definition (XSD)*.

Definición

Al definir el **schema** es necesario escribir una serie de atributos:

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

Esto indica que los elementos utilizados corresponden al espacio de nombres (*namespace*) definido en esa dirección. No es el objetivo de este módulo describir los espacios de nombres, pero podemos decir que para que no haya ambigüedad entre las etiquetas descritas, siempre se define un espacio de nombres al que pertenecen.

Dentro de un espacio de nombres no puede haber dos etiquetas con el mismo nombre.

Al igual que en la DTD, es posible hacer una referencia a un esquema o escribirlo manualmente en el prólogo del documento.

Para referenciarlo utilizamos los atributos **xsi** de **xmlns:xsi** y **xsi:schemalocation**. En este último pondríamos el archivo o dirección web donde esté definido el nuevo *schema*. Puedes ver un ejemplo a continuación.

```
xmlns:xsi= http://www.w3.org/2001/XMLSchema-instance xsi:schemaLocation=  
"c:/archivo.xsd"
```

Elementos simples

Un elemento simple es el que contiene solo datos de contenido, no contiene ningún otro elemento. Se describe con la etiqueta:

```
<xs:element name="nombre_elemento" type="tipo_datos"/>
```

En *schema* cuando se define un elemento se debe poner tanto el nombre como el tipo de datos que contiene.

Se han definido varios tipos de datos, incluso pueden definirse otros tipos de datos nuevos descritos por el usuario. Los más comunes son:

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

El **elemento** puede incluir atributos o restricciones. Si no incluyera ninguna de estas utilizaríamos la forma de sintaxis abreviada que hemos visto; abre con "<" y cierra al final de la línea con "/>."

Si tuviera alguno de estos elementos se debe definir una nueva etiqueta, indicando que el elemento es simple:

```
<xss:element name="nombre_elemento">
<xss:simpleType>
```

Atributos y restricciones

Para definir atributos se utiliza una sintaxis similar a la que tenemos para los elementos.

```
<xs:attribute name="nombre_atributo" type="tipo"/>
```

Se pueden poner valores por defecto y fijos con los atributos *fixed* y *default*, así como elegir si es requerido o no con el atributo *use*.

```
<xs:attribute name="nombre_atributo" type="tipo" use="required" default="valor"/>
<xs:attribute name="nombre_atributo" type="tipo" fixed="valor"/>
```

En cuanto a las restricciones, existen gran número de restricciones que se indican con una etiqueta. Las más utilizadas son la de valor máximo y mínimo, la longitud del campo de datos, o la elección entre una lista de valores de atributo:

```
<xs:restriction base="xs:string">
  <xs:length value=" numero de caracteres"/>
</xs:restriction>
<xs:restriction base="xs:string">
  <xs:length value="8"/>
</xs:restriction>
<xs:restriction base="xs:integer">
  <xs:minInclusive value="valor_maximo"/>
  <xs:maxInclusive value="valor minimo"/>
</xs:restriction>
```

Resumen

Has terminado la lección, veamos los puntos más importantes que hemos tratado.

En esta unidad hemos aprendido:

- Las partes que conforman un documento XML y la jerarquía de sus elementos.
- La validación y/o definición de un documento mediante las reglas de las DTD.
- La validación de los documentos mediante *schema* para documentos en los que queremos insertar reglas más estrictas, como el tipo de datos de los diferentes nodos.



PROEDUCA