XML (*Extensible Markup Language*)

José Antonio Echagüe Burgos
ja_echague@yahoo.es

Tabla de Contenido

1. Lenguajes de marcas - Introducción

Cuando no existían ordenadores, la forma de referenciar en la edición de textos los atributos de formato era mediante una notación específica, desarrollada ex profeso. Se trataba de marcas añadidas a un documento original para indicar algún tipo de información. Para definirlas se utilizó un tipo de sintaxis u otra, según el caso. Se imponía por tanto desarrollar un mecanismo más o menos definido para desarrollar las marcas necesarias. Para distinguir dichas marcas del resto del texto original, se incluyen dentro de unos símbolos de uso poco frecuente dentro de los textos normales, por ejemplo:

```
Este texto es #neg#una prueba#neg#
```

La universalidad de los lenguajes de marcas utilizados, exige la utilización de una sintaxis común, que permita su compatibilidad y su correcta interpretación. Una simple lista de marcas no es suficiente. Para excluir la utilización de marcas contrapuestas es necesario definir las reglas de jerarquía que deben tener las marcas entre sí. Estos dos aspectos, a saber,

- definir las marcas y
- definir la jerarquía de utilización entre ellas,

son lo que precisamente lleva a cabo XML. La informática permite la utilización de lenguajes de marcas muy estructurados. En base a este conjunto de consideraciones surge SGML, como lenguaje que permite a su vez definir lenguajes de marcado.

SGML, *Standard Generalized Markup Language*, no es por ello un lenguaje de marcado, sino un lenguaje que permite definir lenguajes de marcas, como HTML. Surge en 1986, y se trata de una norma ISO, siendo por lo tanto un estándar.

SGML es por otra parte un lenguaje muy potente, que permite desarrollar todas las funcionalidades del mundo editorial, al que en un principio se dirigía. Desarrollado inicialmente por Goldford (IBM), fue posteriormente estandarizado por la agencia internacional ISO. Sin embargo, su gran potencia es quizás su principal desventaja fuera del mundo editorial y de las grandes empresas. En 1998 surge XML como una versión simplificada de SGML. La filosofía de SGML se basa en dos principios básicos:

- El entorno del documento, en el que hay que distinguir: contenido, estructura y tratamiento.
- Independencia de plataforma. La distinción que se establece en el entorno del documento, ha de establecerse con independencia de la plataforma.

En un lenguaje de marcado, el contenido es el contenido original que va a ser marcado, mientras que la estructura se verá reflejada en una serie de etiquetas o marcas alusivas al tipo de contenido. El tratamiento sería por su parte los diversos usos que podemos realizar con el documento marcado: imprimirlo, hacer búsquedas por texto u otros parámetros, enviarlo, convertirlo a otro formato, etc.

En cuanto al tratamiento, si bien SGML permite la posibilidad de implementar atributos relativos al tratamiento en las etiquetas de la estructura, es aconsejable no hacerlo, desde la perspectiva de separar los tres aspectos del entorno del documento mencionados anteriormente, en este caso la estructura del tratamiento. Hay por ello que tratar de evitar definir atributos de tratamiento dentro de las etiquetas, como hace por ejemplo HTML, ya que ello va contra los principios que rigen la filosofía de SGML y por extensión la de XML.

Esta separación es fácil de implementar en HTML, ya que al tratarse de etiquetas fijas, es posible referenciarlas fácilmente desde un archivo externo de definición de atributos de tratamiento (CSS). En XML, al no tratarse de un conjunto de etiquetas o marcas fijas, sino reglas para definirlas, con una semántica de

utilización limitada, no se define de ningún modo el tratamiento de las mismas. En SGML y por extensión en XML, el tratamiento se define:

- A través de un programa realizado mediante un lenguaje de programación de propósito general.
- Mediante el estándar ISO DSSSL, que permite pasar XML a otro tipo de lenguaje más adecuado para la presentación de los contenidos.

La independencia de plataforma se garantiza en SGML ya que sólo implementa caracteres ASCII, entendibles por cualquier plataforma y por cualquier editor de texto. Los caracteres ASCII garantizan por tanto una mayor perdurabilidad de los datos.

El desarrollo de SGML se basa en el uso de lo que se conoce como gramáticas independientes de contexto, que permiten desarrollar lenguajes de programación y lenguajes de definición de lenguajes de marcado.

1.1. HTML

HTML, acrónimo de *Hypertext Markup Language*, es un lenguaje de marcas de hipertexto, y un formato estándar de documentos de texto que se utiliza desde 1989 en la World Wide Web (WWW). Los documentos HTML contienen dos tipos de información: la que se muestra en pantalla y códigos (*tags* o etiquetas), transparentes al usuario, que indican cómo mostrar esa información. HTML es un subconjunto de SGML (*Standard Generalized Markup Language*, lenguaje estándar de marcado de documentos), estándar de descripción de página independiente del dispositivo.

En un documento HTML, encontramos por un lado etiquetas que indican los atributos del texto (negrita, centrado, etc.). Otras señalan al sistema cómo responder a eventos que genera el usuario, como apuntar con el ratón a un icono que representa una película y en respuesta ejecutar el programa que reproduce vídeo en formato digital, por ejemplo. La etiqueta más importante es el vínculo (link), que puede contener la URL de otro documento. Este documento puede residir en el mismo lugar Web que el documento actual o en cualquier otro ordenador de la WWW. El usuario "navega" de documento en documento seleccionando estos vínculos con el ratón. HTML también incluye marcas para rellenar formularios (*forms*), que permiten al usuario enviar la información necesaria para realizar consultas en bases de datos, comprar o solicitar un servicio.

El software que permite al usuario consultar documentos en la World Wide Web se denomina explorador (*browser*). Es el encargado de interpretar las etiquetas HTML y mostrar el documento en pantalla. La ventaja de este formato es que, al ser un estándar aceptado (al contrario que la mayoría de los procesadores de texto), cualquiera puede construir un explorador. Además, la mayoría de los lugares Web siguen este estándar, fácil de implementar, lo que ha contribuido al crecimiento exponencial de la WWW. Por otra parte, HTML evoluciona porque se crean nuevas etiquetas acompañadas de exploradores capaces de interpretarlas.

2. XML - Introducción

La menor potencia de XML respecto de SGML, hace a este lenguaje de marcado más restrictivo. Por ejemplo, en SGML el sistema cierra automáticamente las etiquetas no cerradas por error, ya que incorpora las reglas sintácticas necesarias para ello. XML generaría en este caso un error, por lo que no llegaría a abrirse el documento. Por otra parte, XML no es un estándar ISO, sino una recomendación del W3C (1998), derivada del estándar SGML. Supone una simplificación de este último para su uso en Internet.

El problema del tratamiento se ha solucionado en XML aplicando la misma filosofía de uso de SGML. XML facilita el tratamiento ya que los programas desarrollados en un lenguaje de programación de propósito

general pueden ser ayudados mediante la utilización de APIs[1], partes de código ya escritas que implementan las funcionalidades básicas, para así poder concentrarse en la funcionalidades avanzadas. De esta forma, el tratamiento de documentos con XML no parte de cero, sino que ya están disponibles una serie de APIs aplicables a XML:

DOM (*Document Object Model,* recomendación de W3C)

Da una representación orientada a objetos, basada en árboles, del documento XML, estableciendo nodos de información (datos o metadatos).

SAX (*Simple API for XML*)

David Megginson. Tiene menos capacidad que el DOM. Recorre el documento y "avisa" cuando encuentra una marca y cuando encuentra contenido (*eventos*).

Ambas APIs cargan el documento en un formato determinado. Por otra parte, el paso de documentos XML a HTML (transformación de etiquetas XML en etiquetas HTML) se realiza mediante el estándar **XSL** (*Extended Stylesheets Language*), cuyo núcleo es **XSLT** (*Extended Stylesheets Language Transformations*). XSL es XSLT más un lenguaje para pasar etiquetas XML a un formato de presentación HTML.

3. Estructura y DTD de un documento XML

En un documento XML debemos distinguir tres aspectos claramente diferenciados:

1. Contenido.
2. Estructura.
3. Tratamiento.

En XML la estructura se fija, al igual que en el resto de lenguajes de marcado, mediante una sintaxis de etiquetas, con sus correspondientes cierres. Las marcas que podemos utilizar y su sintaxis de utilización se especifican en una **DTD** (*Document Type Definition*), que contiene las reglas para aplicar XML a documentos de un determinado tipo. Es la gramática que define el conjunto de marcas y sus relaciones. Define las marcas con las que podemos marcar un documento, a partir de una marca raíz. La DTD realmente define el documento XML.

Por tanto, XML implica manejar por una parte la definición de marcas y sus relaciones mediante DTDs, y por otra marcar los documentos con ese conjunto de etiquetas. De forma análoga, XHTML es la DTD que define los elementos y sus relaciones en HTML, para hacerlos acordes con XML.

Otra forma de definir las marcas y su sintaxis de utilización, al margen de la DTD, es mediante **XML Schemas**. Se trata de una sintaxis alternativa, más potente que la DTD, para definir las marcas y su gramática de uso.

La estructura del documento en lenguaje XML viene definida por las DTDs aplicables, que, como se ha dicho, definen las marcas y sus relaciones y atributos. El contenido de un documento XML viene definido por la instancia, que es el contenido original más las marcas XML. La DTD mas la instancia es lo que conocemos como documento XML. La DTD son las reglas de utilización de las marcas, y la instancia el contenido original mas dichas marcas:

Documento XML = DTD + Instancia / **Instancia** = contenido original + marcas

[1] API, *Application Programming Interface*. Una API "proporciona llamadas a funciones y procedimientos que proporcionen métodos y mecanismos para" la manipulación.

El tratamiento del documento XML se realiza mediante un lenguaje de programación de propósito general, ayudado por DOM y SAX, o mediante la aplicación de una transformación XSLT. La definición de DTDs y la conformación de la instancia de un documento XML contempla los siguientes elementos:

- Declaraciones: declaración de elementos, declaración de tipo de documento.
- Atributos de los elementos.
- Entidades: sirven para tener partes del documento fuera de él y luego referenciarlas.
- Instrucciones de procesamiento: llamada (señal) a la herramienta que procesa el lenguaje para que lleve a cabo una acción.
- Notaciones: invocaciones a programas capaces de entender símbolos de aplicación específica.

4. Definición de DTDs - Elementos

4.1. Declaración de elemento

Los elementos definen las marcas de nuestro lenguaje y sus relaciones. Es lo que va a permitir definir las marcas que tendremos dentro de nuestro lenguaje específico. La sintaxis para declarar elementos dentro de una DTD es:

```
<!ELEMENT nombre contenido>
```

Por ejemplo:

```
<!ELEMENT libro (intro,caps)>
```

Qué puede ser nombre y qué puede ser contenido:

- Nombre. Será el nombre del elemento y equivalentemente el nombre de la marca.
- Contenido. Indica en la instancia del documento el contenido del elemento, que es equivalente a lo que podrá aparecer, a nivel de instancia, entre la etiqueta `<nombre>` y `</nombre>`.

4.1.1. Tipos de contenidos

El contenido puede ser:

- Datos carácter.
- Otros elementos.
- Mixto.
- Vacío (`EMPTY`).
- Cualquier elemento de la DTD (`ANY`).

Datos carácter

Se indica entre corchetes de la siguiente forma:

```
(#PCDATA)
```

Así por ejemplo:

```
<!ELEMENT saludo (#PCDATA)>
```

Entre `<saludo>` y `</saludo>` podrán aparecer caracteres, es decir, el elemento `<saludo>` contiene o puede contener caracteres.

Otros elementos

Ejemplo:

```
<!ELEMENT profesor (nombre)>
<!ELEMENT nombre (#PCDATA)>
```

A nivel de instancia:

```
<profesor><nombre>Antonio Fuentes</nombre></profesor>
```

Mixto

Declaración formal:

```
<!ELEMENT nombre (#PCDATA|nombreE1|...|nombreEn)>
```

Por ejemplo:

```
<!ELEMENT texto (#PCDATA|nota|referencia)*>

<texto>
       <nota>es un ejemplo </nota>Esto es un ejemplo de
       <referencia>Antonio</referencia> para que lo entendáis
       <nota>eso espero</nota> Adiós.
</texto>
```

Vacío (EMPTY)

Se especifica mediante la palabra `EMPTY`. Se utiliza cuando se quieren poner atributos al elemento pero sin contenido. Por ejemplo:

```
<!ELEMENT foto EMPTY>

<foto></foto> ó <foto/>
```

Una única etiqueta que indica el elemento `EMPTY` y su cierre puede escribirse de dos formas:

```
<nombre></nombre> ó <nombre/>
```

Un elemento `#PCDATA` también permite poner etiquetas de esa forma:

```
<!ELEMENT n (#PCDATA)>

<n>Hola</n> ó </n>
```

El elemento `#PCDATA` también permite no poner nada. Esta forma de indicar los cierres de las etiquetas de los elementos no suele utilizarse, para mantener la compatibilidad de XML con SGML.

Cualquier elemento de la DTD (ANY)

Se especifica mediante la palabra `ANY`. Por ejemplo:

```
<!ELEMENT ejemplo (todos,eMail)>
<!ELEMENT todos ANY>
<!ELEMENT eMail (origen,destino,CC*,asunto,mensaje)>
```

```
<ejemplo>
         <todos>Hola a <CC>Pepe</CC>
          esto es un ejemplo <mensaje>tonto</mensaje>
          pero útil.</todos>
</ejemplo>
```

Nota: Si pusiéramos `<eMail>` entre `<ejemplo>` y `</ejemplo>`, entonces habría que especificar en la instancia, de forma obligatoria, todos los elementos que componen `<eMail>` en la DTD.

4.1.2. Relaciones que pueden existir entre los elementos

- Secuenciación.
- Opción o elección.

Secuenciación

Se indica mediante comas:

```
<!ELEMENT profesor (nombre, apellido, apellido2)>
<!ELEMENT nombre (#PCDATA)>
<!ELEMENT apellido (#PCDATA)>

<profesor>
          <nombre>Antonio</nombre>
          <apellido>Fuentes</apellido>
          <apellido2>Herrero</apellido2>
</profesor>
```

En la instancia hay que respetar el orden de la secuencia de los elementos, las relaciones jerárquicas definidas por la DTD.

Opción o elección

Se representa mediante una barra vertical.

```
<!ELEMENT empleado (fijo|porHoras)>
<!ELEMENT fijo (#PCDATA)>
<!ELEMENT porHoras (#PCDATA)>

<empleado><fijo>Antonio</fijo><empleado>
<empleado><porHoras>Antonio</porHoras></empleado>
```

Se pueden mezclar ambas formas de relación entre los elementos dentro de la DTD:

```
<!ELEMENT empleado(nombre,(fijo|porHoras)>
<!ELEMENT nombre(fijo|porHoras)>
```

4.1.3. Repetición y carácter opcional de los elementos

La repetición y el carácter opcional de los elementos se indica mediante una serie de símbolos:

`?` --> 0 ó 1
`+` --> 1 ó *n* veces.
`*` --> 0 ó *n* veces.

Por tanto, un elemento puede ser o aparecer de las siguientes formas:

- Opcional.
- Al menos uno.
- En cualquier número.

Opcional

Se representa mediante un signo de interrogación `?`.

```
<!ELEMENT persona (nombre,conyuge?)>

<persona>
         <nombre>Carlos</nombre>
         <conyuge>Ana</conyuge>
</persona>
```

O bien:

```
<persona><nombre>Carlos</nombre></persona>
```

Al menos un elemento

Esta circunstancia se indica mediante un signo `+`.

```
<!ELEMENT eMail (origen,destino+,CC,asunto,mensaje)>

<eMail>
       <origen>...</origen>
       <destino>...</destino>
       <destino>...</destino>
        ....
       <CC>...</CC>
       <asunto>...</asunto>
       <mensaje>...</mensaje>
</eMail>
```

En cualquier número

Si queremos que, siguiendo el ejemplo anterior, `<CC>` pueda aparecer o no y que, en el caso de que aparezca, lo haga en cualquier número, debemos indicar esta propiedad con un asterisco `*`.

```
<!ELEMENT eMail (origen,destino,CC*,asunto,mensaje)>

<eMail>
       <origen>...</origen>
       <destino>...</destino>
       <CC>...</CC>
       <CC>...</CC>
       <asunto>...</asunto>
       <mensaje>...</mensaje>
</eMail>
```

También podría aparecer sólo una vez `<CC>` o ninguna.

4.2. Reglas generales

No se puede definir un elemento más de una vez.

El mismo elemento no puede aparecer más de una vez como modelo de contenido de un elemento con contenido mixto. Por ejemplo:

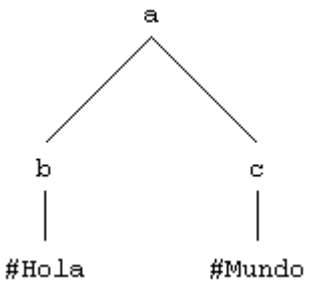
```
<!ELEMENT a (b,b)> --> sí
<!ELEMENT a (#PCDATA|b|c|b)*> --> no
```

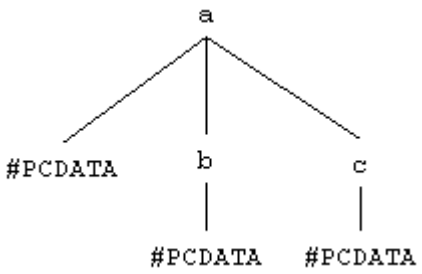
En el ultimo ejemplo nos encontramos con un elemento (a) con contenido mixto. El contenido mixto es posible por el uso del asterisco `*`.

El contenido mixto puede ser `(#PCDATA)` ó `(#PCDATA,elemento)`. Ejemplo:

```
<!ELEMENT a (b,c)>
<!ELEMENT b (#PCDATA)>
<!ELEMENT c (#PCDATA)>

<a>
   <b>Hola</b>
   <c>Mundo</c>
</a>
```

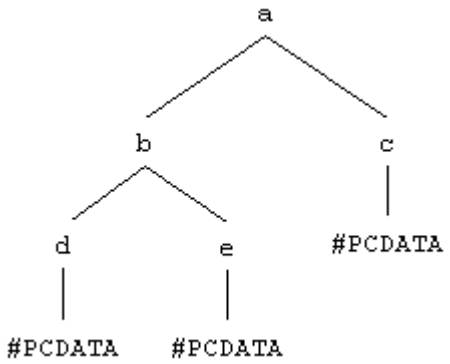




Ejemplo:

```
<!ELEMENT a (#PCDATA|b|c)*>
<!ELEMENT b (#PCDATA)>
<!ELEMENT c (#PCDATA)>

<a>Hola <b>a todo</b> el <c>mundo</c>
mundial</a>
```



Los ejemplos anteriores muestran como una definición DTD "define" una estructura de árbol del documento XML, que es de la que se sirve XSL para acceder a los nodos. Un ejemplo algo más complejo es el siguiente:

```
<!ELEMENT a (b,c)>
<!ELEMENT b (#PCDATA)>
<!ELEMENT c (d,e)>
<!ELEMENT d (#PCDATA)>
<!ELEMENT e (#PCDATA)>
```

5. Definición de DTDs - Declaración de tipo de documento

La declaración de tipo de documento vincula DTDs a instancias a través de la siguiente declaración, inserta en la cabecera del documento XML:

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

<!DOCTYPE nombreDoc [

                    <!ELEMENT nombreDoc (contenido)>
                    ...
              ]>

              instancia del documento XML

              </nombreDoc>

</xml>
```

Como se ha dicho en anteriores apartados, un documento XML está formado por la DTD y la instancia (= marcas + contenido original), relacionándose ambos elementos mediante la declaración que acabamos de definir en el código anterior. La DTD también puede estar contenida en un archivo diferente del archivo XML, en vez de en su cabecera. En este caso, la declaración incluida en la cabecera del documento XML incluye una referencia al archivo que contiene la DTD (y que tiene extensión `.dtd`) de la siguiente forma:

```
<!DOCTYPE nombreDoc SYSTEM "nombreArchivo">
```

Por ejemplo:

```
<!DOCTYPE eMail SYSTEM "email.dtd">
```

Podemos especificar si la DTD es propia o de una organización que la ha definido. En el segundo caso la declaración a incluir en la cabecera del documento XML es como sigue:

```
<!DOCTYPE nombreDoc PUBLIC "nombreArchivo">
```

6. Definición de DTDs - Atributos

Los atributos permiten capturar propiedades no estructuradas. La sintaxis de la declaración de un atributo es la siguiente:

```
<!ATTLIST nombreElemento listaAtributos>
```

En donde:

`nombreElemento`: es el nombre del elemento al cual vamos a definir un atributo o atributos.

`listaAtributos`: se puede definir más de un atributo, dejando un espacio en blanco entre cada uno de ellos. Se escribe de la siguiente manera:

```
defAtributo 1
defAtributo 2
defAtributo n
```

`defAtributo`: los elementos que hay que especificar en su definición son los que siguen:

```
nombreAtributo (tipoAtributo) valorDefecto
```

El valor por defecto es el que el sistema va a atribuir si no especificamos ninguno. Ejemplo:

```
<!ELEMENT eMail (...)>
<!ATTLIST eMail prioridad (alta|baja|media) media>
```

Con lo que:

```
<eMail>...</eMail> = <eMail prioridad="media">...</eMail>
```

6.1. Tipos de atributos

Son los siguientes:

- Cadena de caracteres (`CDATA`).
- Enumerado.
- Identificador (`ID`).
- Referencia a un identificador (`IDREF`).
- Referencia a una lista de identificadores (`IDREFS`).
- Entidad/lista de entidades (`ENTITY`/`ENTITIES`).
- Atributo `NMTOKEN`.
- Lista de nombres (`NMTOKENS`).

6.1.1. Cadena de caracteres

Se utiliza cuando no queremos restringir el atributo a priori, para que pueda ser una cadena de caracteres. Este tipo de atributo se especifica con la palabra `CDATA`.

```
<!ELEMENT tabla (...)>
<!ATTLIST tabla descripción CDATA #IMPLIED>
```

En el ejemplo anterior, el valor por defecto es `#IMPLIED`, y la utilización de esta palabra permite incluir el atributo en la instancia o dejarlo en blanco. Si utilizamos como valor por defecto la palabra `#REQUIRED`, deberemos especificar obligatoriamente los posibles valores del atributo en la instancia:

```
<tabla descripción="Tabla de proveedores">
...
</tabla>
```

Todo atributo tiene que tener especificado de manera obligatoria el valor por defecto, ya que así lo requiere la sintaxis de XML.

6.1.2. Enumerado

Se define en la DTD de la siguiente manera:

```
(valor1|valor2|...|valorn)
```

El subrayado significa que en la instancia no tiene que aparecer literalmente `valor1`, `valor2`, etc., sino algo que represente ese valor: `1`, `2`, `3` ...; `a`, `b`, `c` ... etc.

```
<!ELEMENT persona (#PCDATA)>
<!ATTLIST persona sexo CDATA valorDefecto>
```

Si la definición del atributo fuese tal cual está expresada en el ejemplo anterior, en la instancia podría aparecer como valor de atributo cualquier cadena de caracteres, lo que en este caso no tiene sentido:

```
<persona sexo="BMW">
...
</persona>
```

Para evitar esta circunstancia, especificaremos, siempre que así lo requiera el atributo, sus posibles valores en la definición de valor de atributo:

```
<!ELEMENT persona (...)>
<!ATTLIST persona sexo (m|h) "m">
```

```
<persona sexo="m">Alicia</persona>
```

6.1.3. Identificador

Este tipo de atributo trata de garantizar que en un documento no existan dos elementos con el mismo valor de atributo, cuando así lo requiera el tipo de contenido original que queramos marcar. Se especifica en la DTD con la palabra `ID`, que implica que en la instancia del documento el valor asignado al atributo tiene que ser único. Ejemplo:

```
<!ELEMENT persona (#PCDATA)>
<!ATTLIST persona DNI ID #REQUIRED>

<persona DNI="_11">Antonio</persona>
<persona DNI="_22">Ana</persona>
```

Los valores de atributos `ID` tienen una serie de restricciones:

- El valor que asignemos al atributo tiene que empezar con una letra o con un guión bajo (_).
- Sólo se puede tener un atributo `ID` por elemento.

6.1.4. Referencia a un identificador

A veces nos interesa en un elemento referenciar a otros elementos, es decir, que estos elementos referencien a otros elementos, que además estén unívocamente referenciados. Se especifica con la palabra `IDREF`. El valor de un atributo `IDREF` debe ser el valor de un atributo `ID` en el documento. Ejemplo:

```
<!ELEMENT hijo (#PCDATA)>
<!ATTLIST hijo padre IDREF #REQUIRED>
              madre IDREF #REQUIRED>

<hijo padre="d11" madre="d112">Luis</hijo>
```

Nota: En el ejemplo anterior, los valores de los atributos "`padre`" y "`madre`" (`d11` y `d112` respectivamente) serían por ejemplo valores `ID` referidos a los DNI de los atributos "`padre`" y "`madre`" de la instancia del documento. Tienen que existir por ello, obligatoriamente, sendos atributos "`padre`"y "`madre`" con los valores `ID` especificados.

Otro ejemplo:

```
<!ELEMENT a (b|c)*>
<!ELEMENT b (#PCDATA)>
<!ATTLIST b idB ID #REQUIRED>
<!ELEMENT c (#PCDATA)>
<!ATTLIST c refB IDREF #REQUIRED>

<a><b idB="x11">Hola</b>
...
<c refB="x11">pepe</c></a>
```

En las relaciones `IDREF`, el nombre de los atributos no importa, puede ser cualquiera. En el ejemplo anterior, es el valor de atributo `refB="x11"` el que nos lleva o referencia al elemento `b` con valor de atributo `idB="x11"`.

6.1.5. Referencia a una lista de identificadores

Se especifica con la palabra `IDREFS`. Ejemplo:

```
<!ELEMENT hijo (#PCDATA)>
<!ATTLIST hijo padres IDREFS #REQUIRED>

<hijo padres="d11 d111">Luis</hijo>
```

Los valores `IDREFS` se separan en la instancia por blancos.

6.1.6. Entidad/lista de entidades

Se especifica mediante la palabra `ENTITY`, y tiene que ser el valor de una entidad no analizada declarada en la DTD (ver ejemplo en el apartado correspondiente a las entidades). Por su parte, una lista de entidades se especifica mediante la palabra `ENTITIES`.

6.1.7. Atributo NMTOKEN

Un atributo `NMTOKEN` es una cadena de caracteres (*token*) formada por letras, dígitos, guiones, puntos, comas ó guiones bajos (*underscore*), es decir, una cadena sin blancos. Ejemplo:

```
<!ELEMENT libro (...)>
<!ATTLIST libro ISBN NMTOKEN #REQUIRED>

<libro ISBN="1-234T6-784-0">
...
</libro>
```

6.1.8. Lista de palabras (*name tokens*)

Este tipo de atributo se especifica mediante `NMTOKENS`. Las palabras de la lista se separan mediante espacios. Ejemplo:

```
<! ELEMENT cliente EMPTY>
<! ATTLIST cliente numeroDeReferencia NMTOKENS #REQUIRED>
```

6.2. Tipos de valores por defecto de un atributo

`#REQUIRED`	Expresa la obligatoriedad de dar un valor al atributo.
`#IMPLIED`	Su uso permite que se pueda incluir o no el valor del atributo.

`"valorAtributo"`	Valor de un enumerado o de un `CDATA` referidos a los valores de un atributo.
	`<!ATTLIST persona sexo (h|m) "m">`
	`nacion CDATA "Italia"> etc.`

`#FIXED valorAtributo`	Especifica que se trata de un valor de atributo fijo.
	`<!ATTLIST documento autor CDATA #FIXED "Ant">`
	En la instancia:
	`<documento>...</documento> ó`
	`<documento autor="Ant">...</documento>`

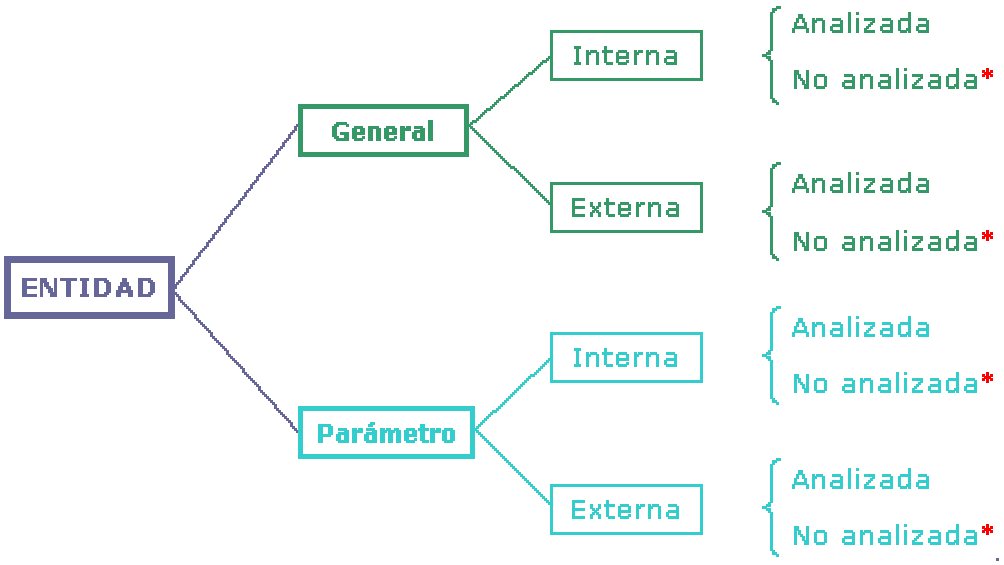
6.3. Observaciones generales

- Un elemento puede tener más de un atributo.
- El valor por defecto para un atributo de tipo `ID` tiene que ser `#IMPLIED` ó `#REQUIRED`.

7. Definición de DTDs - Entidades

Las entidades son una forma de nombrar datos *no* xml que se incluyen en los documentos XML, por ejemplo, una imagen, un vídeo, etc. Los tipos de entidades se establecen en base a tres características distintas:

- Entidades generales/parámetro.
- Entidades internas/externas.
- Entidades analizadas/no analizadas.



Nota: Los tipos de entidades señaladas con un asterisco * son tipos no permisibles.

Entidades generales/parámetro

Las entidades generales van a nombrar datos a introducir en la instancia del documento XML. Las entidades parámetro son por su parte una forma de nombrar datos xml a incluir en la DTD.

Entidades internas/externas

Las entidades internas definen su contenido (datos a incluir en el documento) en el propio documento XML. En el caso de las entidades externas, los datos a incluir se encuentran en un archivo externo.

Entidades analizadas/no analizadas

Las entidades analizadas contienen datos xml, se sustituyen y se analizan esos datos. Las entidades no analizadas contienen datos *no* xml (imágenes, vídeo, audio, etc., es decir, datos *no* carácter) que no se sustituyen ni analizan.

7.1. Tipos permisibles de entidades

7.1.1. Declaración de entidad general interna analizada

Por *general* entendemos que se trata de datos a sustituir en la instancia, y por "analizada" que son datos xml.

```
<!ENTITY nombreEntidad "datos">
```

La referencia a una entidad general en la instancia se realiza de la siguiente forma:

```
&nombreEntidad;
```

Ejemplo:

```
<!ENTITY ucm "Universidad Complutense de Madrid">

<texto>Antonio trabaja en la &ucm; y le gusta</texto>
```

7.1.2. Declaración de entidad general externa analizada

En este tipo de entidades, los datos, en vez de definirse en el propio documento XML, se definen de forma externa, en un archivo independiente.

```
<!ENTITY nombreEntidad SYSTEM/PUBLIC "URI">
```

El URI (*Uniform Resource Identifier*) puede ser:

- La ruta de un archivo.
- Una URL (*Universal Resource Locator*).

Ejemplo:

```
<!DOCTYPE texto [
      <!ELEMENT texto (#PCDATA)>
      <!ENTITY ucm SYSTEM "<\data\ucm.txt">
]>

<texto>Trabajo en la &ucm;</texto>
```

Otro ejemplo:

```
<!DOCTYPE libro [
      <!ELEMENT libro (capitulo+)>
      <!ELEMENT capitulo (cap1,cap2)>
      ...
      <!ENTITY cap1 SYSTEM "capitulo1.txt">
      <!ENTITY cap2 SYSTEM "capitulo2.txt">
]>

<libro>&cap1;
       &cap2;
</libro>
```

Nota: la inclusión de la entidad debe corresponderse con lo que cada elemento admite como contenido (otros elementos, `PCDATA`, etc.).

7.1.3. Declaración de entidad general externa no analizada

Se trata de invocar a programas que sepan manejar los datos *no* xml, siendo este programa el nombre de la notación. El nombre de la notación, definido en la DTD, es por ello el que se va a encargar de manejar los datos *no* xml.

```
<!ENTITY nombreEntidad SYSTEM "URI" NDATA nombreNotacion>
```

Ejemplo:

```
<!ENTITY foto1 SYSTEM "foto1.gif" NDATA gif>
<!NOTATION gif SYSTEM "visorGif.exe">
<!ELEMENT imagen EMPTY>
<!ATTLIST imagen archivo ENTITY #REQUIRED>

<texto>Aquí viene una foto <imagen archivo="foto1"/></texto>
```

En el ejemplo anterior, `gif` es la notación encargada de manejar la entidad `foto1`. En la práctica no se aplica el anterior procedimiento, ya que generalmente las aplicaciones, por sí solas, saben o no saben manejar los datos *no* xml. Si no saben hacerlo, la anterior declaración es demasiado simple para que el analizador invoque al archivo capaz de manejar la entidad. Lo normal es por ello hacer la siguiente declaración y aplicación de esta a la instancia, delegando en la aplicación o analizador xml la capacidad de manejar o no los datos *no* xml:

```
<!ELEMENT imagen EMPTY>
<!ATTLIST imagen archivo CDATA #REQUIRED>

<texto>Aquí viene una foto <imagen archivo="foto1.gif"/></texto>
```

7.1.4. Declaración NOTATION

Proporciona un programa que el analizador XML pasa a la aplicación para que manipule datos *no* xml.

```
<!NOTATION nombre Notación PUBLIC/SYSTEM "URI">
```

Ejemplo:

```
<!NOTATION gif SYSTEM "visorGif.exe">
```

7.1.5. Declaración de entidad parámetro interna analizada

```
<!ENTITY %nombreEntidad "datos">
```

Ejemplo:

```
<!ENTITY %atrComunes "archivo" CDATA #REQUIRED
                     "autor" CDATA #REQUIRED>
<!ELEMENT datos (...)>
<!ATTLIST datos %atrComunes;>
```

La referencia a la entidad parámetro dentro de la DTD se hace por ello con:

```
%nombreEntidad;
```

7.1.6. Declaración de entidad parámetro externa analizada

```
<!ENTITY %nombreEntidad SYSTEM "URI">
<!ENTITY %atrComunes SYSTEM "atrCom.txt">
<!ELEMENT datos (…)>
<!ATTLIST datos %atrComunes;>
```

7.2. Inserción de referencias a caracteres

Se utilizan para insertar caracteres que no están en el teclado, y se hace de la siguiente forma:

- Caracteres binarios: `&#64;`
- Caracteres hexadecimales: `&#x41;`

Nota: `64` y `41` se han puesto a modo de ejemplo, pueden ser cualquier otro código de los admitidos.

Ejemplo. Para poner:

```
<texto> 45 < 211 </texto>
```

Escribiremos:

```
<texto> 45 &#60; 211 </texto>
```

7.2.1. Entidades predefinidas

Las entidades predefinidas no es necesario que sean definidas por el usuario, ya que el sistema, como su propio nombre indica, las tiene predefinidas. Son las siguientes:

Binario	Entidad predefinida	Hexadecimal
`&`	&	`&#38;`
`<`	<	`&#60;`
`>`	>	`&#62;`
`&apos;`	'	`&#39;`
`&quot;`	"	`&#34;`

8. DTDs - Subconjunto interno y subconjunto externo

El estándar XML permite tener dos DTDs: una interna y otra externa. La interna complementa a la externa. En la DTD interna podemos definir únicamente:

- Declaraciones de entidades.
- Declaraciones de atributos.

Tiene mayor preferencia lo que se defina de forma interna que aquello definido externamente. Ejemplo: tenemos la DTD "`libro.dtd`" que contiene:

```
<!ELEMENT libro (capitulo+)>
<!ELEMENT capitulo (cap1,cap2,...,capn)>
...
```

Esta DTD externa no contiene declaraciones `ENTITY`. En la instancia tenemos por su parte el siguiente subconjunto interno:

```
<!DOCTYPE SYSTEM "libro.dtd"
       [ <!ENTITY cap1 SYSTEM "cap1.txt">
         <!ENTITY cap2 "cap2.txt">
]>

<libro>&cap1;
       &cap2;
</libro>
```

Si quisiéramos añadir nuevos capítulos al elemento "`libro`", no tendremos que modificar la DTD externa, que contendrá el esquema general de la estructura del documento XML, sino simplemente al subconjunto interno de la DTD interna de dicho documento, y hacer la modificaciones oportunas.

9. Instrucciones de procesamiento

Dan una serie de instrucciones a la aplicación que está procesando el documento XML para que lleve a cabo alguna acción especial. Por ejemplo:

```
<?print ship?>
```

Se incluyen dentro de la instancia del documento. Aunque se pueden poner, no se debería hacerlo, al ir contra la filosofía de SGML y XML, marcar el documento independientemente del tratamiento. La sintaxis de estas instrucciones es:

```
<?aplicación instrucción?>
```

10. Prólogo XML - Declaración XML

Un documento XML está formado por un prólogo seguido de una instancia seguido de otros, como por ejemplo una instrucción de procesamiento. El prólogo está formado por:

- La declaración de tipo de documento, que contiene la DTD (incluida físicamente o referenciada).
- Declaración XML (antes de la DTD).

El aspecto de la declaración XML es como sigue:

```
<?xml version="1.0" encoding="ISO8859-1" standalone="yes/no"?>
```

- `encoding` es la definición de codificación, juego de caracteres que está utilizando el documento XML. Por defecto es UTF-8. En el ejemplo anterior, `ISO8859-1` es el estándar ISO que recoge el juego de caracteres extendido.
- `standalone` indica si hay declaraciones de marcado externas al documento.

La declaración XML habitual adopta por ello la siguiente forma:

```
<?xml version="1.0" encoding="ISO8859-1"?>
```

Si la DTD contiene caracteres "extraños" o extendidos, esta declaración también hay que incluirla en la misma, poniéndola delante de la definición de la DTD. Por ejemplo:

```
<?xml version="1.0" encoding="ISO8859-1"?>
<!ELEMENT á(#PCDATA)>

<?xml version="1.0" encoding="ISO8859-1"?>
<!DOCTYPE á SYSTEM "a.dtd">
...
<á>Café</á>
```

11. Secciones CDATA

Permiten incluir en la instancia caracteres y etiquetas de marcado que no se corresponden con etiquetas de marcado de la DTD o que no queremos que se identifiquen con etiquetas de marcado. Adoptan la siguiente sintaxis de utilización:

```
<![CDATA [contenido]]>
```

Por ejemplo, es incorrecto poner:

```
<texto>La etiqueta de negrita es <b></texto>
```

La correcta inclusión del ejemplo anterior en un documento XML sería la siguiente:

```
<texto>
<![CDATA [La etiqueta de negrita <b> es utilizada...]]>
</texto>
```

12. Comentarios

Se pueden incluir tanto en la instancia del documento como en la DTD, mediante la siguiente sintaxis:

```
<!-- comentario -->
```

Ejemplo:

```
<!ELEMENT a (PCDATA)>
<!-- Esto es un ejemplo -->
<!ELEMENT b (#PCDATA)>
```

13. Espacios en blanco

Hay que diferenciar entre:

- Los que aparecen como contenido de un elemento con valor `PCDATA`: blanco, tabulador, salto de línea.
- Espacios en blanco entre etiquetas:

```
<a>
   <b>Hola</b>
   <c>Mundo</c>  = <a><b>Hola</b><c> Mundo</c></a>
</a>
```

La forma de indicar a la aplicación que procesa el documento XML que tenga en cuenta los espacios en blanco es mediante el atributo `xmlispace`:

```
<!DOCTYPE a [
<!ELEMENT a (b+)>
<!ATTLIST a xmlispace (default|preserve)”preserve”>
]>
```

En cuyo caso,

```
<a>
   <b>Hola</b>
   <c>Mundo</c>
</a>
```

no es igual a:

```
<a><b>Hola</b><c> Mundo</c></a>
```

14. Documentos válidos y documentos bien formados

Un documento bien formado es aquel que respeta la sintaxis de XML:

`<a tipo="grande>` `-->` no está bien formado.
`<a tipo="grande">` `-->` bien formado.

`<a><b></a></b>` `-->` no está bien formado.
`<a><b></b></a>` `-->` bien formado.

Un documento puede estar bien formado y no tener DTD. Por su parte, documento válido es un documento bien formado en el que además la instancia se ajusta a la DTD correspondiente. Un documento válido tiene que ser necesariamente un documento bien formado.

Esta diferenciación se permite para que existan documentos XML sin DTD, de cara a su distribución por Internet, cuando se sobreentiende que la DTD es de dominio público o ya la tiene el destinatario del documento. Por otra parte, un documento XML sin DTD no sirve para nada. La instrucción de procesamiento XML sirve precisamente para que un documento XML sin instancia pueda ser bien interpretado por una aplicación.

Bibliografía

DuCharme B., XML *The Annotated Specification*, Prentice Hall, 1999.

Extensible Markup Language (XML) 1.0, W3C Recommendation 10-February-1998.
Disponible en: http://www.w3.org/TR/1998/REC-xml-19980210 (obsoleta)

Extensible Markup Language (XML) 1.0 (Second Edition) W3C Recommendation 6 October 2000
Disponible en: http://www.w3.org/TR/REC-xml

Goldfarb C.F., The SGML Handbook, Oxford University Press, 1990.

HTML 4.01 Specification, W3C Recommendation 24 December 1999.
Disponible en: http://www.w3.org/TR/html4/

Ide N., Véronis J., *Text Encoding Initiative: Background and Context*, Kluwer Academic Publishers, 1995.

XML Path Language (XPath) Version 1.0, W3C Recommendation 16 November 1999.
Disponible en: http://www.w3.org/TR/xpath

XSL Transformations (XSLT) Version 1.0, W3C Recommendation 16 November 1999.
Disponible en: http://www.w3.org/TR/xslt

Ejemplos de DTD con su aplicación en la instancia

Ejemplo nº 1

DTD del documento XML

```
<?xml version="1.0" encoding="ISO8859-1"?>
<!ELEMENT dniS (dni+)>
<!ELEMENT dni (datos1, datos2)>

<!ELEMENT datos1 (numeroDNI, nombre, ap1, ap2, expedido, valido, firma, foto)>
<!-- Podría ponerse como un elemento #PCDATA y sin atributo -->
<!ELEMENT numeroDNI EMPTY>
<!ATTLIST numeroDNI dni ID #REQUIRED>
<!ELEMENT nombre (#PCDATA)>
<!ELEMENT ap1 (#PCDATA)>
<1ELEMENT ap2 (#PCDATA)>
<!ELEMENT expedido (dia, mes, ano)>
<!ELEMENT dia (#PCDATA)>
<!ELEMENT mes (#PCDATA)>
<!ELEMENT ano (#PCDATA)>
<!ELEMENT valido (dia, mes, ano)>
<!ELEMENT firma EMPTY>
<!ATTLIST firma archivo ENTITY #REQUIRED>
<!ELEMENT foto EMPTY>
<!ATTLIST foto archivo ENTITY #REQUIRED>

<!ELEMENT datos2 (nacioEn, provincia, el, hijoDe, sexo, domicilio, localidad,
provincia, equipo)>
<!-- Hay demasiadas localidades como para restringirlo como un atributo enumerado -->
<!ELEMENT nacioEn (#PCDATA)>
<!-- Hay demasiadas provincias como para restringirlo como un atributo enumerado -->
<!ELEMENT provincia (#PCDATA)>
<!ELEMENT el (dia, mes, ano)>
<!ELEMENT hijoDe (padre, madre)>
<!ELEMENT padre (#PCDATA)>
<1ELEMENT madre (#PCDATA)>
<!ELEMENT sexo EMPTY>
<!ATTLIST sexo tipo (v|h) #REQUIRED>
<!ELEMENT domicilio (via, numero, piso)>
<!ELEMENT via (#PCDATA)>
<!ATTLIST via tipo (calle|plaza|avda) "calle">
<!ELEMENT numero (#PCDATA)>
<!ELEMENT piso (#PCDATA)>
<!ELEMENT localidad (#PCDATA)>
<!ELEMENT equipo (#PCDATA)>
```

Instancia del documento XML

```
<?xml version="1.0" encoding="ISO8859-1"?>
<!DOCTYPE dniS SYSTEM "dnis.dtd"
      [ <!NOTATION gif SYSTEM "">
        <!ENTITY foto123456789 SYSTEM "foto123456789.gif" NDATA gif>
        <!ENTITY firma123456789 SYSTEM "firma123456789.gif" NDATA gif>
        <!ENTITY foto987654321 SYSTEM "foto987654321.gif" NDATA gif>
```

```
        <!ENTITY firma987654321 SYSTEM "firma987654321.gif" NDATA gif>
      ]>
<dniS>
      <dni>
          <datos1>
                  <numeroDNI dni="_123456789"/>
                  <nombre>Luis</nombre>
                  <ap1>Sánchez</ap1>
                  <ap2>Pérez</ap2>
                  <expedido>
                      <dia>2001</dia>
                      <mes>2001</mes>
                      <año>2001</ano>
                  </expedido>
                  <valido>
                      <dia>2001</dia>
                      <mes>2001</mes>
                      <ano>2006</ano>
                  </valido>
                  <firma archivo="firma987654321"/>
                  <foto archivo="foto987654321"/>
          </datos1>
          <datos2>
                  <nacioEn>Madrid</nacioEn>
                  <provincia>Madrid</provincia>
                  <el>
                      <dia>01</dia>
                      <mes>O1</mes>
                      <ano>1961</ano>
                  </el>
                  <hijoDe>
                      <padre>Luis</padre>
                      <madre>Pepa</madre>
                  </hijoDe>
                  <sexo tipo="v"/>
                  <domicilio>
                      <via tipo="calle">Mayor</via>
                      <numero>58</numero>
                      <piso>3 A</piso>
                  </domicilio>
                  <localidad>Madrid</localidad>
                  <provincia>Madrid</provincia>
                  <equipo>1425406D8</equipo>
          </datos2>
      </dni>
      <dni>
          <datos1>
                  <numeroDNI dni="_987654321"/>
                  <nombre>Ana</nombre>
                  <ap1>Martos</ap1>
                  <ap2>Juarez</ap2>
                  <expedido>
                      <dia>02</dia>
                      <mes>02</mes>
                      <ano>2001</ano>
                  </expedido>
                  <valido>
                      <dia>O2</dia>
                      <mes>O2</mes>
                      <ano>2006</ano>
                  </valido>
                  <firma archivo="firma987654321"/>
                  <foto archivo="foto987654321"/>
          </datos1>
          <datos2>
```

```
                    <nacioEn>Madrid</nacioEn>
                    <provincia>Madrid</provincia>
                    <el>
                        <dia>O4</dia>
                        <mes>O4</mes>
                        <ano>1974</ano>
                    </el>
                    <hijoDe>
                        <padre>Javier</padre>
                        <madre>Eva</madre>
                    </hijoDe>
                    <sexo tipo="h"/>
                    <domicilio>
                        <via tipo="avda">Ciudad de Barcelona</via>
                        <numero>1</numero>
                        <piso>14</piso>
                    </domicilio>
                    <localidad>Madrid</localidad>
                    <provincia>Madrid</provincia>
                    <equipo>745754H5S4</equipo>
            </datos2>
        </dni>

</dniS>
```

Ejemplo nº 2

DTD del documento XML

```
<?xml version="1.0" encoding="ISO8859-1"?>
<!ELEMENT libro (portada, indice, capitulos, referencias)>
<!ELEMENT portada (titulo, autores)>
<!ELEMENT titulo (#PCDATA)>
<!ELEMENT autores (autor+)>
<!ELEMENT autor (nombre, apellido)>
<!ELEMENT nombre (#PCDATA)>
<!ELEMENT apellido (#PCDATA)>
<!-- El índice está formado por una lista de referencias a capítulos -->
<!ELEMENT indice (capituloI+)>
<!ELEMENT capituloI (#PCDATA)>
<!ATTLIST capituloI refA IDREF #REQUIRED>
<!ELEMENT capitulos (capitulo+)>
<!-- Cada capítulo tiene un ID para poder referenciarlas desde el índice -->
<!ELEMENT capitulo (nombre, texto)>
<!ATTLIST capitulo id ID #REQUIRED
          autor CDATA #IMPLIED>
<!ELEMENT texto (#PCDATA | ref)*>
<!-- Las ref referencian a las referencias concretas -->
<!ELEMENT ref (#PCDATA)>
<!ATTLIST ref refA IDREF #REQUIRED>
<!ELEMENT referencias (referencia+)>
<!-- Cada referencia está identificada para poder referenciarlas desde el texto -->
<!ELEMENT referencia (titulo, autores, publicacion)>
<!ATTLIST referencia id ID #REQUIRED>
<!ELEMENT publicacion (#PCDATA)>
```

Instancia del documento XML

```
<?xml version="1.0" encoding="ISO8859-1"?>
<!DOCTYPE libro SYSTEM "libro.dtd">
<libro>
      <portada>
            <titulo>Un día en la vida del cangrejo tibetano</titulo>
            <autores>
```

```
                  <autor>
                        <nombre>Antonio</nombre>
                        <apellido>Navarro</apellido>
                  </autor>
            </autores>
      </portada>
      <indice>
            <capituloI refA="manana">Por la mañana</capituloI>
            <capituloI refA="mediodia">Al mediodía</capituloI>
            <capituloI refA="tarde">Por la tarde</capituloI>
            <capituloI refA="noche">Por la noche</capituloI>
      </indice>
      <capitulos>
            <capitulo id="manana">
                  <nombre>Por la mañana en la vida del cangrejo tibetano</nombre>
                  <texto>La mañana es uno de los momentos de mayor actividad del
cangrejo tibetano. Como diría <ref refA="Sanchez00"> Sánchez</ref>: "al cangrejo que
madruga, Dios le ayuda"...
                  </texto>
            </capitulo>
            <capitulo id= "mediodia">
                  <nombre>Al mediodía en la vida del cangrejo tibetano</nombre>
                  <texto>Al mediodía el cangrejo...</texto>
            </capitulo>
            <capitulo id="tarde">
                  <nombre>Por la tarde en la vida del cangrejo tibetano</nombre>
                  <texto>Por la tarde el cangrejo...</texto>
            </capitulo>
            <capitulo id="noche">
                  <nombre>Por la noche en la vida del cangrejo tibetano</nombre>
                  <texto>Es durante la noche cuando el cangrejo tibetano lleva a cabo
sus ritos de apareamientos. La pareja de cangrejos utilizan diversas técnicas [<ref
refA="Sanchez00"/>] en función de
                  </texto>
            </capitulo>
      </capitulos>
      <referencias>
            <referencia id="Sanchez00">
                  <titulo>El cangrejo tibetano</titulo>
                  <autores>
                        <autor>
                              <nombre>Marta</nombre>
                              <apellido>Sanchez</apellido>
                        </autor>
                  </autores>
                  <publicacion>14 Congreso sobre fauna tibetana. Valdemoro, 28.07.00-
03.08.00, Madrid, Spain
                  </publicacion>
            </referencia>
      </referencias>
</libro>
```

Ejemplo nº 3

DTD del documento XML

```
<?xml version="1.0" encoding="ISO8859-1"?>
<!ELEMENT biblioteca (libros, autores, fondos)>
<!-- Cada libro está identificado por su ISBN y referencia a sus autores -->
<!ELEMENT libros (libro+)>
<!ELEMENT libro (titulo, autoresL)>
<!ATTLIST libro ISBN ID #REQUIRED>
<!ELEMENT titulo (#PCDATA)>
<!ELEMENT autoresL EMPTY>
```

```
<!ATTLIST autoresL refA IDREFS #REQUIRED>
<!ELEMENT autores (autor+)>
<!ELEMENT autor (#PCDATA)>
<!ATTLIST autor id ID #REQUIRED>
<!-- El fondo relaciona a cada libro con su código de biblioteca -->
<!ELEMENT fondos (fondo+)>
<!ELEMENT fondo EMPTY>
<!ATTLIST fondo libro IDREF #REQUIRED
                codigo ID #REQUIRED>
```

Instancia del documento XML

```
<?xml version="1.0" encoding="ISO8859-1"?>
<!DOCTYPE biblioteca SYSTEM "biblioteca.dtd">
<biblioteca>
      <libros>
            <libro ISBN="_1">
                  <titulo>El cangrejo tibetano</titulo>
                  <autoresL refA="a1"/>
            </libro>
            <libro ISBN="_2">
                  <titulo>El mejillón holandés</titulo>
                  <autoresL refA="a2"/>
            </libro>
      </libros>
      <autores>
            <autor id="a1">Antonio Navarro</autor>
            <autor id="a2">Marta Sánchez</autor>
      </autores>
      <fondos>
            <!-- Hay tres ejemplares del primer libro, y dos del segundo -->
            <fondo libro="_1" codigo="b1"/>
            <fondo libro="_1" codigo="b2"/>
            <fondo libro="_1" codigo="b3"/>
            <fondo libro="_5" codigo="b4"/>
            <fondo libro="_5" codigo="b5"/>
      </fondos>
</biblioteca>
```