TEMA4: LENGUAJES PARA ALMACENAMIENTO Y TRANSMISIÓN DE INFORMACIÓN (XML)

Contenido del tema:

- ✓ Definición de XML
- ✓ Estructura y sintaxis de XML
- ✓ Documentos XML bien formados

Definición de XML:

XML (Lenguaje de Etiquetas Extendido) es un lenguaje de etiquetas, que estructura y guarda de forma ordenada la información.

Proporciona a webs y a aplicaciones una forma potente de guardar la información. Además, se ha convertido en un formato universal que ha sido asimilado por todo tipo de sistemas operativos y dispositivos móviles.

Al igual que en HTML un documento XML es un documento de texto, en este caso con extensión ".xml", compuesto de parejas de etiquetas, estructura en árbol, que describen una función en la organización del documento.

El proceso de creación de un documento XML pasa por varias etapas:

- Especificación de requisitos.
- Diseño de etiquetas.
- Marcado de los documentos.

Es interesante detectar necesidades futuras y crear documentos con una estructura fácilmente actualizable.

Herramientas básicas de XML:

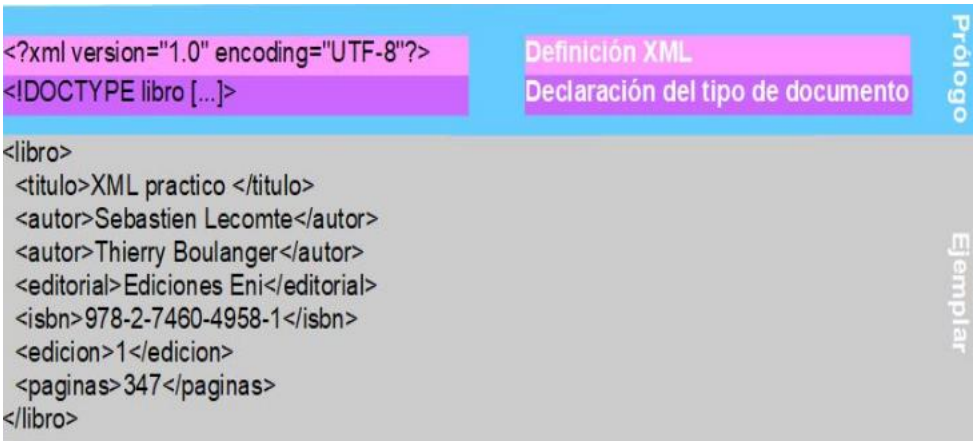
Para trabajar en XML es necesario editar los documentos y luego procesarlos, por tanto tenemos dos tipos de herramientas:

Editores: Para crear documentos XML es conveniente usar algún editor XML. Estos incluyen ayuda para la creación de otros elementos como DTD, hojas de estilo CSS o XSL,...

**Nosotros usaremos el XML Copy Editor.*

Procesadores: Para interpretar el código XML se puede utilizar cualquier navegador. Los procesadores de XML permiten leer los documentos XML y acceder a su contenido y estructura. Un procesador es un conjunto de módulos de software entre los que se encuentra un parser o analizador de XML que comprueba que el documento cumple las normas establecidas para que pueda abrirse. Estas normas pueden corresponderse con las necesarias para trabajar sólo con documentos de tipo válido o sólo exigir que el documento esté bien formado.

Los elementos: prólogo y ejemplar



Los documentos XML están formados por una parte llamada prólogo y otra parte llamada ejemplar.

Prólogo: informa al intérprete encargado de procesar el documento de todos aquellos datos que necesita para realizar su trabajo. Consta de dos partes:

DOCUMENTO XML → PRÓLOGO → DEFINICIÓN XML

→ DECLARACIÓN TIPÒ DE DOCUMENTO

→ EJEMPLAR

Definición: donde se indica la versión de XML que se utiliza, a codificación empleada para representar los caracteres y la autonomía del documento. Las 3 opciones son:

<?xml versión= “1.0” ?>
<?xml versión= “1.0” encoding=”iso-8859-1” ?>
<?xml versión= “1.0” encoding=”iso-8859-1” standalone=”yes” ?> En este caso el documento es independiente, de no ser así el atributo standalone hubiese tomado el valor "no".

Declaración del tipo de documento: define qué tipo de documento estamos creando para ser procesado correctamente:

<!DOCTYPE Nombre_del_ejemplar ...> El nombre_del_ejemplar ha de ser idéntico al del ejemplar del documento XML.

Ejemplar: contiene los datos reales del documento. Están delimitados por una etiqueta de apertura y una etiqueta de cierre. En realidad, el ejemplar es el elemento raíz de un documento XML. Todos los demás elementos de un documento XML están contenidos en el mismo.

```
<?xml version="1.0" encoding="UTF-8"?>
<libro>
  <titulo>XML practico </titulo>
  <autor>Sebastien Lecomte</autor>
  <autor>Thierry Boulanger</autor>
  <editorial>Ediciones Eni</editorial>
  <isbn>978-2-7460-4958-1</isbn>
  <edicion>1</edicion>
  <paginas>347</paginas>
</libro>
```

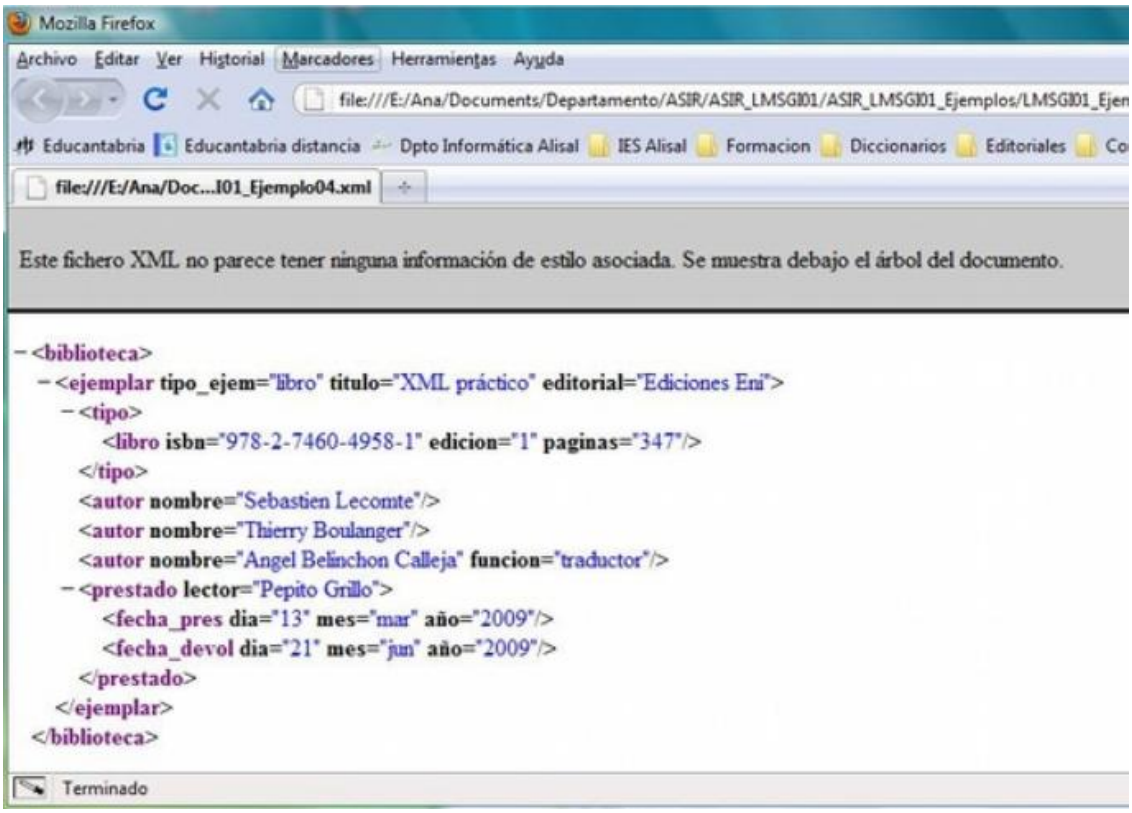
El ejemplar es el elemento <libro>, que a su vez está compuesto de los elementos <titulo>, <autor>, <editorial>, <isbn>, <edicion> y <paginas>.

Los **atributos** permiten añadir propiedades a los elementos de un documento. Los atributos no pueden organizarse en ninguna jerarquía, no pueden contener ningún otro elemento o atributo, y no puede haber dos atributos con el mismo nombre en un mismo elemento.

No se debe utilizar un atributo para contener información susceptible de ser dividida.

```
<?xml version="1.0" encoding="UTF-8" standalone="yes" ?>
<biblioteca>
  <ejemplar tipo_ejem="libro" titulo="XML práctico" editorial="Ediciones Eni">
    <tipo>
      <libro isbn="978-2-7460-4958-1" edicion="1" paginas="347"></libro>
    </tipo>
    <autor nombre="Sebastien Lecomte"></autor>
    <autor nombre="Thierry Boulanger"></autor>
    <autor nombre="Angel Belinchon Calleja" funcion="traductor"></autor>
    <prestado lector="Pepito Grillo">
      <fecha_pres dia="13" mes="mar" año="2009"></fecha_pres>
      <fecha_devol dia="21" mes="jun" año="2009"></fecha_devol>
    </prestado>
  </ejemplar>
</biblioteca>
```

Al abrir el documento anterior con el navegador Firefox obtenemos:



Vemos que los elementos aparecen coloreados en ciruela, los nombres de los atributos en negro y sus valores en azul.

Como se observa en el ejemplo, los atributos se definen y dan valor dentro de una etiqueta de inicio o de elemento vacio, a continuación del nombre del elemento o de la definición de otro atributo siempre separado de ellos por un espacio. Los valores del atributo van precedidos de un igual que sigue al nombre del mismo y tienen que definirse entre comillas simples o dobles.

Los nombres de los atributos han de cumplir las mismas reglas que los de los elementos, y no pueden contener el carácter menor que, "<".

Definición de la sintaxis de documentos XML:

En los documentos de lenguajes de marcas, la distribución de los elementos está jerarquizada según una estructura de árbol, lo que implica que es posible anidarlos pero no entrelazarlos.

Hemos visto que en los elementos el orden es importante, ¿lo es también para los atributos? En este caso el orden no es significativo. Lo que hay que tener presente es que no puede haber dos atributos con el mismo nombre.

Sabemos que los atributos no pueden tener nodos que dependan de ellos, por tanto solo pueden corresponder con hojas de la estructura de árbol que jerarquiza los datos.

¿Significa esto que todas las hojas van a ser atributos? Pues no, es cierto que los atributos son hojas, pero las hojas pueden ser atributos o elementos.

En ese caso, ¿qué criterios podemos utilizar para decidir si un dato del documento que se pretende estructurar ha de representarse mediante un elemento o un atributo? Aunque no siempre se respetan, podemos usar los siguientes criterios:

- El dato será un elemento si cumple alguna de las siguientes condiciones:
  - Contiene subestructuras.
  - Es de un tamaño considerable.
  - Su valor cambia frecuentemente.
  - Su valor va a ser mostrado a un usuario o aplicación.
- Los casos en los que el dato será un atributo son:
  - El dato es de pequeño tamaño y su valor raramente cambia, aunque hay situaciones en las que este caso puede ser un elemento.
  - El dato solo puede tener unos cuantos valores fijos.
  - El dato guía el procesamiento XML pero no se va a mostrar.

Documentos XML bien formados:

Para no dar error y poder ser interpretado por un procesador, el documento XML ha de cumplir ciertas normas:

- Debe existir un elemento raíz, y sólo uno.

-Todos los elementos tienen una etiqueta de inicio y otra de cierre. En el caso de que en el documento existan elementos vacíos, deben cerrarse: />. Es decir, <elemento> </elemento> puede sustituirse por: <elemento/>

-Deben estar correctamente anidados (el zapato y el calcetín).

-Los nombres de las etiquetas de inicio y de cierre de un elemento han de ser idénticos, respetando las mayúsculas y minúsculas. Pueden ser cualquier cadena alfanumérica que no contenga espacios y no comience ni por el carácter dos puntos, ":", ni por la cadena "xml" ni ninguna de sus versiones ("XML", "XmL", "xML",...).

-El contenido de los elementos no puede contener la cadena "]]>" por compatibilidad con SGML. Además no se pueden utilizar directamente los caracteres mayor que, >, menor que, <, ampersand, &, dobles comillas, ", y apostrofe, ‘. En el caso de tener que utilizar estos caracteres se sustituyen por las siguientes cadenas:

- > por >
- < por <
- & por &
- " por "
- ' por '

-Para utilizar caracteres especiales, como £, ©, ®,... hay que usar las expresiones &#D; o &#H; donde D y H se corresponden respectivamente con el número decimal o hexadecimal correspondiente al carácter que se quiere representar en el código UNICODE. Por ejemplo, para incluir el carácter de Euro, €, se usarían las cadenas € o &#x20AC.

-Los nombres de las etiquetas han de ser autodescriptivos, lo que facilita el trabajo.

-Los atributos deben de ir entre comillas.

-Los documentos XML pueden tener comentarios, <!--" y "--> en cualquier posición en el documento salvo antes del prólogo o dentro de una etiqueta.

-Todos los documentos XML deben verificar las reglas sintácticas que define la recomendación de W3C para el estándar XML.

Un documento **está bien formado** cuando sigue las reglas de XML. Un documento **es válido** si sigue las normas del vocabulario que tiene asociado.

¿Cuáles son los errores del siguiente documento XML?

```
<?XML VERSION="1.0" encoding="UTF-8" standalone="yes" ?>
<biblioteca>
  <ejemplar tipo_ejem='libro' titulo='XML práctico' editorial='Ediciones Eni'>
    <tipo>
      <libro isbn='978-2-7460-4958-1' edicion= paginas='347'></libro>
    </tipo>
    <autor nombre='Sebastien Lecomte'></autor>
    <autor nombre='Thierry Boulanger'></autor>
    <autor nombre='Angel Belinchon Calleja' funcion='traductor'></autor>
    <prestado lector='Pepito Grillo'>
      <fecha_pres dia='13' mes='mar' año='2009'></fecha_pres>
      <fecha_devol>
    </prestado>
  </ejemplar>
</biblioteca>
```

- [x] Utiliza mayúsculas en la definición de la versión XML.
- [ ] No utiliza el código de caracteres adecuado.
- [ ] Los valores de los atributos no están entre comillas dobles.
- [x] Hay algún atributo vacío.
- [x] La etiqueta <fecha_devol> no se cierra.
- [ ] Se usan mayúsculas en los datos del documento.

¿ Está "bien formado" el siguiente documento XML?

```
<?xml version="1.0"?>
<mensaje>
   <destinatario>Tomas</destinatario>
   <remitente>Juan</remitente>
   <asunto>
   <contenido> No olvides ir a recogerme al aeropuerto mañana por la mañana!</contenido>
</mensaje>
```

○ Verdadero ◉ Falso

Correcto

Ni está cerrada la etiqueta <asunto> ni el prólogo tiene una declaración xml completa.

Sección CDATA:

En esta sección se puede añadir contenido que no será procesado o analizado y es similar a un comentario. Deben aparecer dentro del elemento raíz del documento XML. Por ejemplo:

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

Un ejemplo de un documento XML con sección CDATA:

```
<?xml version="1.0"?>
<mensaje>
  <![CDATA[
  <destinatario>Tomas</destinatario>
  <remitente>Juan</remitente>]]>
  <contenido> No olvides ir a recogerme al aeropuerto mañana por la mañana!</contenido>
</mensaje>
```

TEMA4: LENGUAJES PARA ALMACENAMIENTO Y TRANSMISIÓN DE INFORMACIÓN (DTD)

Contenido del tema:

- ✓ Definición de DTD
- ✓ Declaración de tipos de elementos terminales
- ✓ Declaración de tipos de elementos no terminales
- ✓ Declaración de atributos para los elementos

Definición de XML:

Los DTD (Definiciones de Tipo de Documento), especifican la estructura de un documento XML. Es decir, indican qué elementos pueden aparecer en un documento y dónde, así como el contenido y los atributos que puede tener cada uno. El documento XML debe cumplir las restricciones que se hayan impuesto en el DTD para ser válido.

¿Cuáles son los inconvenientes de los DTD?

Los principales son:

- Su sintaxis no es XML.
- No soportan espacios de nombres.
- No definen tipos para los datos. Solo hay un tipo de elementos terminales, que son los datos textuales.
- No permite las secuencias no ordenadas.
- No es posible formar claves a partir de varios atributos o elementos.
- Una vez que se define un DTD no es posible añadir nuevos vocabularios.

La DTD puede estar incluida en el propio documento, ser un documento externo o combinarse ambas.

- Cuando están dentro del documento XML se ubican entre corchetes después del nombre del ejemplar en el elemento <!DOCTYPE>.

```
<!DOCTYPE nombre_ejemplar[
... declaraciones ...
]>
```

- Cuando la DTD está en un documento externo, definimos el DTD externo en un fichero de texto plano con extensión dtd y se especifica un URI donde pueden localizarse las declaraciones. Además, en el documento XML, se puede incluir el atributo standalone="no" en la declaración de XML.

    - Si sólo va a ser utilizada por una única aplicación, la sintaxis es la siguiente:

```
<!DOCTYPE nombre_ejemplar SYSTEM "URI">
```

    - Si la DTD está en un documento externo y, si va a ser utilizada por varias aplicaciones, también se especifica un identificador público:

```
<!DOCTYPE nombre_ejemplar PUBLIC "id_publico" "URI">
```

Declaración de tipos de elementos terminales:

Los tipos terminales son aquellos elementos que se corresponden con hojas de la estructura de árbol, es decir, aquellos que no contienen más elementos.

La declaración de tipos de elementos está formada por la cadena "<!ELEMENT" separada por, al menos un espacio del nombre del elemento XML que se declara, y seguido de la declaración del contenido que puede tener dicho elemento.

Esta declaración de contenido puede tener los siguientes valores:

EMPTY: Indica que el elemento no contiene nada.

```
<!ELEMENT A EMPTY>
```

ANY: Permite que el contenido del elemento sea cualquier cosa (texto y otros elementos). Un ejemplo de definición de un elemento de este tipo es:

```
<!ELEMENT A ANY>
```

(#PCDATA): Indica que el elemento puede contener datos de tipo carácter exceptuando los siguientes: <, &,]], >. Si es de este tipo, el elemento A tendrá una definición como:

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

Declaración de tipos de elementos no terminales:

Una vez que sabemos el modo de definir las hojas de un árbol de datos veamos cómo definir sus ramas, es decir los elementos que están formados por otros elementos. Para definirlos utilizamos referencias a los grupos que los componen tal y como muestra el ejemplo:

```
<!ELEMENT A (B, C)>
```

En este caso se ha definido un elemento A que está formado por un elemento B seguido de un elemento C.

Ocurrencias de los elementos

¿Y qué sucede cuando un elemento puede aparecer en el documento varias veces? ¿hay que indicarlo de algún modo? Pues sí, y también hay que indicar cuándo un elemento puede no aparecer. Para ello usamos los siguientes operadores, que nos permiten definir la cardinalidad de un elemento:

?: Indica que el elemento es opcional. En el ejemplo, el subelemento trabajo es opcional.

```
<!ELEMENT telefono (trabajo?, casa )>
```

+: Indica que el elemento aparece al menos una vez. En el ejemplo, definimos un elemento formado por el nombre de una provincia y otro grupo (cp, ciudad), que puede aparecer una o varias veces.

```
<!ELEMENT provincia (nombre, (cp, ciudad)+ )>
```

*: Indica que un elemento aparece cero, una o varias veces. En el ejemplo, el grupo (cp, ciudad) puede no aparecer o hacerlo varias veces.

```
<!ELEMENT provincia (nombre, (cp, ciudad)* )>
```

|: Se utiliza entre opciones, indicando que hay que elegir entre los elementos separados por este operador. En el ejemplo, se podrá elegir entre cp o ciudad, pero no ambos.

```
<!ELEMENT provincia (nombre, (cp | ciudad) )>
```

Crea un DTD correspondiente a la siguiente estructura de datos de un documento XML:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE alumno SYSTEM "alumno.dtd">
<alumno>
    <nombre>Olga</nombre>
    <direccion>El Percebe 13</direccion>
</alumno>
```

El fichero **alumno.dtd**:

```
<!ELEMENT alumno (nombre,  direccion)>
<!ELEMENT nombre  (#PCDATA)>
<!ELEMENT direccion (#PCDATA)>
```

Declaración de atributos para los elementos:

Ya sabemos cómo declarar elementos, ahora veamos el modo de declarar los atributos asociados a un elemento. Para ello utilizamos la cadena <!ATTLIST seguida del nombre del elemento asociado al atributo que se declara, luego el nombre de éste último seguido del tipo de atributo y del modificador. Este elemento puede usarse para declarar una lista de atributos asociada a un elemento, o repetirse el número de veces necesario para asociar a dicho elemento esa lista de atributos, pero individualmente.

Al igual que los elementos no todos los atributos son del mismo tipo, los más destacados son:

Enumeración: el atributo solo puede tomar uno de los valores determinados dentro de un paréntesis y separados por el operador |.

```
<!ELEMENT fecha (#PCDATA)>
<!ATTLIST fecha dia_semana (lunes|martes|miércoles|jueves|viernes|sábado|domingo) #REQUIRED>
```

CDATA: se utiliza cuando el atributo es una cadena de texto.

```
<!ELEMENT fecha (#PCDATA)>
<!ATTLIST fecha dia_semana CDATA #REQUIRED>
```

ID: permite declarar un atributo identificador en un elemento. Este valor ha de ser único en el documento. Además los números no son nombres válidos en XML, por tanto no son un identificador legal de XML

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

IDREF: permite hacer referencias a identificadores. En este caso, el valor del atributo ha de corresponder con el de un identificador de un elemento existente en el documento.

```
<!ELEMENT persona (#PCDATA) >
<!ATTLIST persona dni ID #REQUIRED>
<!ELEMENT prestamo (#PCDATA) >
<!ATTLIST prestamo persona IDREF #REQUIRED>
```

NMTOKEN: permite determinar que el valor de un atributo ha de ser una sola palabra compuesta por los caracteres permitidos por XML: letras, dígitos, y los caracteres punto ".", guion "-", subrayado "_" y dos puntos ":" y no se permiten espacios.

```
<!ELEMENT persona (#PCDATA) >
<!ATTLIST persona fechanac NMTOKEN #REQUIRED>
```

También hay que declarar si el valor de un atributo es obligatorio o no usando los modificadores:

- #IMPLIED: determina que el atributo sobre el que se aplica es opcional.
- #REQUIRED: determina que el atributo tiene carácter obligatorio.
- #FIXED: permite definir un valor fijo para un atributo y no puede cambiarse.
- Literal: permite definir un valor por defecto para un atributo que puede cambiarse.

Crea un DTD correspondiente a la siguiente estructura de datos de un documento XML:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE alumno SYSTEM "alumno.dtd">
<alumno edad="15">
    <nombre>Olga</nombre>
    <apellidos>Velarde Cobo</apellidos>
    <direccion>El Percebe 13</direccion>
</alumno>
```

```
<!ELEMENT alumno (nombre, apellidos, direccion)>
<!ATTLIST alumno edad CDATA #REQUIRED>
<!ELEMENT nombre (#PCDATA)>
<!ELEMENT apellidos (#PCDATA)>
<!ELEMENT direccion (#PCDATA)>
```

Un documento **está bien formado** cuando sigue las reglas de XML. Un documento **es válido** si sigue las normas del vocabulario que tiene asociado.

Como ejemplo, utilizaremos el siguiente XML **pedido.xml** que guarda los pedidos de clientes de una tienda de Internet:

```
<?xml version="1.0" encoding="UTF-8" ?>
<pedido numero="26" dia="2017-12-01">
<cliente codigo="20">
  <nombre> Felipe </nombre>
  <apellido> Garcia </apellido>
</cliente>
<articulos>
  <articulo>
    <descripcion> Yoda Mimobot USB Flash Drive 8GB </descripcion>
    <cantidad> 5 </cantidad>
    <precio> 38.99 </precio>
  </articulo>
  <articulo>
    <descripcion> Darth Vader Half Helmet Case for iPhone </descripcion>
    <cantidad> 2 </cantidad>
    <precio> 29.95 </precio>
  </articulo>
</articulos>
<total valor="254.85"/>
</pedido>
```

El objetivo es crear el correspondiente DTD para validar el XML, al que llamaremos por ejemplo **"pedido.dtd"**. Analicemos los elementos del XML y crearemos el DTD poco a poco:

La raíz del documento es el elemento <pedido>, que tiene tres hijos:

```
<!ELEMENT pedido (cliente, articulos, total)>
```

También tiene dos atributos, **numero** y **dia**, que sólo pueden tener valores sin espacios y, por tanto, serán **NMTOKEN**. Son datos obligatorios por motivos fiscales.

```
<!ATTLIST pedido numero NMTOKEN #REQUIRED>
<!ATTLIST pedido dia NMTOKEN #REQUIRED>
```

Una vez definido el primer nivel podemos pasar a definir los otros. Por un lado el elemento **<cliente>**. Supondremos que no todos los clientes tienen el atributo código. Sólo los clientes existentes:

```
<!ELEMENT cliente (nombre, apellido)>
<!ATTLIST cliente codigo NMTOKEN #IMPLIED>
```

Por otra parte el elemento **<total>**, que está vacío pero tiene un atributo:

```
<!ELEMENT total EMPTY>
<!ATTLIST total valor CDATA #REQUIRED>
```

También el elemento **<articulos>**, que contendrá una lista de los artículos que ha comprado el cliente. Como no tendría sentido hacer un pedido sin artículos el modificador que se utilizará será **+.**

```
<!ELEMENT articulos (articulo+)>
```

Por su parte, desarrollar **<articulo>** tampoco llevará muchos problemas:

```
<!ELEMENT artículo (descripcion, cantidad, precio)>
```

Para terminar sólo quedan los elementos que contienen datos y el archivo **pedido.dtd** resultante será:

```
<!ELEMENT pedido (cliente, articulos, total)>
<!ATTLIST pedido numero NMTOKEN #REQUIRED>
<!ATTLIST pedido dia NMTOKEN #REQUIRED>

<!ELEMENT cliente (nombre, apellido)>
<!ATTLIST cliente codigo NMTOKEN #IMPLIED>
<!ELEMENT nombre (#PCDATA)>
<!ELEMENT apellido (#PCDATA)>

<!ELEMENT articulos (articulo+)>
<!ELEMENT articulo (descripcion, cantidad, precio)>
<!ELEMENT descripcion (#PCDATA)>
<!ELEMENT cantidad (#PCDATA)>
<!ELEMENT precio (#PCDATA)>

<!ELEMENT total EMPTY>
<!ATTLIST total valor CDATA #REQUIRED>
```

Y añadiremos en **pedido.xml** en la segunda línea, el siguiente código:

```
<!DOCTYPE pedido SYSTEM "pedido.dtd">
```

Si, en cambio, se desea incluir el DTD en el mismo documento pedido.xml, quedaría así:

```
<?xml version="1.0" encoding="UTF-8" ?>
<!DOCTYPE pedido [
<!ELEMENT pedido (cliente, articulos, total)>
<!ATTLIST pedido numero NMTOKEN #REQUIRED>
<!ATTLIST pedido dia NMTOKEN #REQUIRED>

<!ELEMENT cliente (nombre, apellido)>
<!ATTLIST cliente codigo NMTOKEN #IMPLIED>
<!ELEMENT nombre (#PCDATA)>
<!ELEMENT apellido (#PCDATA)>

<!ELEMENT articulos (articulo+)>
<!ELEMENT articulo (descripcion, cantidad, precio)>
<!ELEMENT descripcion (#PCDATA)>
<!ELEMENT cantidad (#PCDATA)>
<!ELEMENT precio (#PCDATA)>

<!ELEMENT total EMPTY>
<!ATTLIST total valor CDATA #REQUIRED>
]>

<pedido numero="26" dia="2017-12-01">
<cliente codigo="20">
  <nombre> Felipe </nombre>
  <apellido> Garcia </apellido>
</cliente>
<articulos>
  <articulo>
    <descripcion> Yoda Mimobot USB Flash Drive 8GB </descripcion>
    <cantidad> 5 </cantidad>
    <precio> 38.99 </precio>
  </articulo>
  <articulo>
    <descripcion> Darth Vader Half Helmet Case for iPhone </descripcion>
    <cantidad> 2 </cantidad>
    <precio> 29.95 </precio>
  </articulo>
</articulos>
<total valor="254.85"/>
</pedido>
```