MEDAC
Instituto Oficial de Formación Profesional

LENGUAJES DE MARCAS Y SISTEMA DE GESTIÓN DE INFORMACIÓN

# Definición de esquemas y vocabulario en XML

10

# ÍNDICE

# OBJETIVOS

*Conocer las características de XML.*

*Entender el concepto de validación.*

*Aprender la validación DTD.*

## / 1. Introducción y contextualización práctica

Los ficheros XML permiten la estructuración y comunicación de información en programas informáticos y su transmisión por la red. Gracias a que no dependen de ninguna plataforma, son compatibles con los diferentes sistemas que estén implementados en una red.

Debido a su rápida expansión como método utilizado para el intercambio de información, son numerosas las herramientas para el tratamiento y recuperación de estos datos. Por ello, es muy importante una buena estructuración y diseño del documento XML, que es lo que se va a estudiar en este tema.

Alberto quiere tener un inventario del software que se utiliza en la empresa WebMalagaDesign.

Para ello, habla con Gloria, estudian la información que necesitan incluir y el método a utilizar para que el documento sea válido.

Escucha el siguiente audio donde planteamos la contextualización práctica de este tema, encontrarás su resolución en el apartado Resumen y resolución del caso práctico.

*Fig. 1. Gloria y Alberto.*

*Audio Intro. "Información compartida"*
*https://bit.ly/330sq5d*

# / 2. Características y estructura de XML

En el tema 1 se hizo una introducción a XML, indicando los aspectos básicos de este lenguaje. A modo de resumen, podemos decir que:

- **XML** es un **lenguaje estructural libre** en la creación de etiquetas derivado de SGML.
- **Describe** el significado de los datos que contiene.
- **Utiliza** un esquema en forma de árbol partiendo de un nodo raíz y consta de dos partes: prólogo y cuerpo.
- **Un documento XML** está bien formado cuando cumple las reglas sintácticas definidas en la recomendación W3C.
- **Las etiquetas** deben estar emparejadas.
- **Es sensible** a mayúsculas.
- **Se usan espacios** de nombres para evitar ambigüedades.

Antes de empezar a detallar las características de XML, hay que indicar lo que no es XML, pues muchas veces surgen dudas:

- **No sustituye** a HTML.
- **No es un estándar** de comunicación.
- **No es un gestor** de base de datos.
- **No es un lenguaje** de programación.

En resumen, XML por sí mismo no hace nada. Su función es estructurar la información para que otros programas hagan uso de esa información.

Con respecto a su estructura, al igual que ocurre con HTML, para construir un documento XML, es suficiente con utilizar un editor de texto simple, porque se trabaja con texto plano. Sin embargo, hay editores avanzados que ayudan en la creación de estos documentos.

La estructura de un documento XML consta de dos partes:

- **Prólogo:** Contiene las instrucciones para procesar el documento y se divide en otras dos partes:
  - **Declaración XML:** Indica la versión, tipo de codificación y si el documento es autónomo.
  - **Declaración del tipo de documento:** Se indica el tipo de documento. Es opcional.
- **Cuerpo:** En el cuerpo se incluyen los elementos XML organizados en forma de árbol. Es el contenido en sí.

```
<?xml version="1.0" encoding="UTF-8" standalone="no" ?>
```

*Código 1. Declaración XML.*

Con el atributo **standalone** se indica si la validez del documento se va a realizar con reglas incluidas en el mismo fichero XML (*standalone="yes"*) o en un fichero externo (*standalone="no"*). La declaración del tipo de documento se detalla en próximos apartados.

# / 3. Elementos, atributos y procesamientos

La unidad fundamental en XML es el **elemento**, que se define con una etiqueta de apertura y otra de cierre. Ambas etiquetas deben estar siempre presentes, si no, el documento no estará bien formado. Los elementos pueden estar anidados y deben seguir una estructura que permita definir de manera organizada los datos a representar.

Por otro lado, un elemento XML puede incluir un **atributo** que puede complementar el contenido del elemento. No se debe abusar de los atributos, porque hacen más complejo el estudio de la información del documento XML.

Por regla general, los atributos van a representar **información** de poca envergadura, sencilla, y no va a tener es tructura. El valor del atributo va entre comillas.

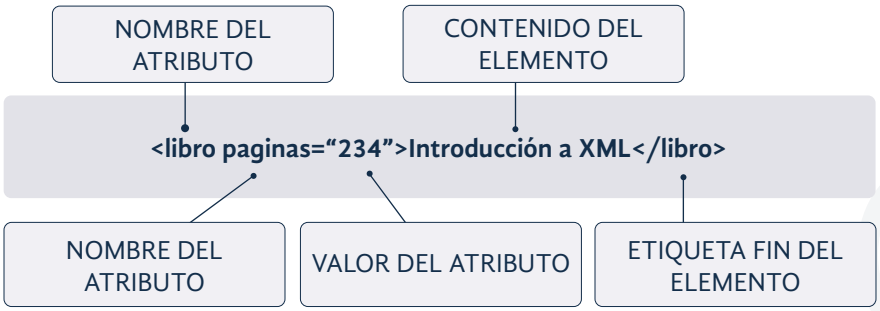



*Fig. 2. Elemento XML.*

Un elemento XML puede no tener contenido, pero sí constar de atributos, aunque no es lo habitual. En ese caso, se puede representar con la etiqueta de elemento vacío **(<elemento/>).**

Con respecto al **procesamiento**, nos estamos refiriendo a que un documento XML puede incluir ciertas instrucciones que van a permitir procesar dicho documento, y por tanto, poder mostrar la información de manera personalizada. Estas instrucciones están en el prólogo y delimitadas por <?...?>.

Entre estas instrucciones, cabe destacar la especificación de hoja de estilo XSLT, que permite la transformación del documento XML (y que estudiaremos en próximos temas), o la inclusión de CSS, que aplicará estilos a los elementos XML.

La aplicación de estos estilos es muy similar a como se aplica en HTML, pero tiene algunas diferencias:

- Los atributos *style, class* y los elementos <style> o <link> no se usan en XML.
- Los objetos ya están identificados por la propia etiqueta.
- La única instrucción de procesamiento a incluir sería:

```
<?xml-stylesheet type="text/css" href=urlhojadeestilo?>
```

*Vídeo 1. "CSS en XML"*
*https://bit.ly/2EuztZP*

## 3.1. Ejemplo de uso de CSS en XML

En el siguiente ejemplo se muestra el uso de CSS en un XML.

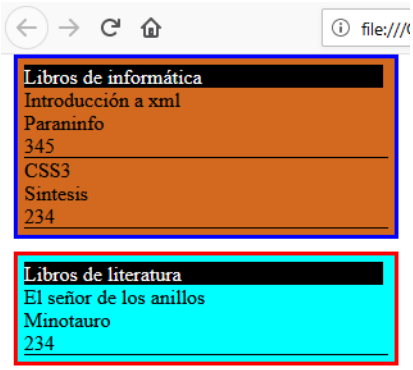
```xml
<?xml version="1.0" encoding="UTF-8"
      standalone="no"?>
<?xml-stylesheet type="text/css"
href="estilos_xml.css"?>

<biblioteca ciudad="Malaga">
  <informatica>
    <libro>
      <nombre>Introducción a xml</nombre>
      <editorial>Paraninfo</editorial>
      <paginas>345</paginas>
    </libro>
    <libro>
      <nombre>CSS3</nombre>
      <editorial>Sintesis</editorial>
      <paginas>234</paginas>
    </libro>
  </informatica>
  <literatura>
    <libro>
      <nombre>El señor de los anillos</nombre>
      <editorial>Minotauro</editorial>
      <paginas>234</paginas>
    </libro>
  </literatura>
</biblioteca>
```

*Código 2. XML con css asociadoy visualización en el navegador.*

```css
*{  box-sizing: border-box;
    display:block;}
informatica{ background-color:chocolate;
    width:300px;
    padding:5px;
    margin-left:10px;
    margin-bottom:10px;
    border:solid 3px blue;
    width:300px;}
informatica:before{display:block;
    content:"Libros de informática";
    color:white;
    background-color:black;
    width:280px;}
literatura{
    background-color:cyan;
    padding:5px;
    margin-left:10px;
    margin-bottom:10px;
    border:solid 3px red;
    width:300px;}
literatura:before{
    display:block;
    content:"Libros de literatura";
    color:white;
    background-color:black;
    width:280px;}
libro{border-bottom:solid 1px black;}
```

*Código 3. Hoja de estilos que se aplica al XML y visualización en el navegador.*




# / 4. Caso práctico 1: "Diseño XML, y XML con CSS"

**Planteamiento:** A Alberto le encargan dos tareas, a priori, muy diferentes:

- Almacenar la información de algunos correos que ha recibido en formato XML.
- Incluir información meteorológica en la web de la empresa. Los datos meteorológicos están en XML.

**Nudo:** Lo habla con Gloría, y para la organización de la información del correo, diseñan cómo sería la estructura del XML. Los elementos que quieren incluir son los siguientes:

- **Correo:** El elemento raíz que incluirá el resto de datos.
- **Remitente:** Se tiene que incluir el nombre y correo.
- **Destinatario:** Se tiene que incluir el nombre y correo.
- **Asunto:** El asunto del correo.
- **Mensaje:** El mensaje en sí, estructurado en párrafos.

Para implementar el apartado de información meteorológica, Gloria comenta que a partir de los datos XML, se podría crear un fichero CSS y aplicar un estilo con iconos meteorológicos para aportar un aspecto visual atractivo.

**Desenlace:**

```xml
<?xml version="1.0" encoding="UTF-8"?>
<correo>
  <remitente>
    <nombre>Pedro Martínez</nombre>
    <email>pmartinez@gmail.com</email>
  </remitente>
  <destinatario>
    <nombre>Gloria Ruiz</nombre>
    <email>gruiz@gmail.com</email>
  </destinatario>
  <asunto>Envío de documentos</asunto>
  <mensaje>
    <parrafo>Hola Gloria</parrafo>
    <parrafo>Te envío los documentos que me
    solicitaste el otro día.</parrafo>
    <parrafo>Por favor, revísalos y me
    comentas algo.</parrafo>
    <parrafo>Un saludo.</parrafo>
  </mensaje>
</correo>
```

*Código 4. Documento XML para correos.*

```xml
<?xml version="1.0" encoding="utf-8"?>
<?xml-stylesheet href="meteo.css"
type="text/css" ?>
<temperatura>
  <lugar>
    <poblacion>Málaga</poblacion>
    <fecha>26/06/2020</fecha>
    <maxima>Máxima: 24 ºC</maxima>
    <minima>Mínima: 16 ºC</minima>
  </lugar>
  <previsiones>
    <prevision estado="nubes">
      <hora>8:00 horas</hora>
      <grados>16 ºC</grados>
    </prevision>
    <prevision estado="sol">
      <hora>16:00 horas</hora>
      <grados>24 ºC</grados>
    </prevision>
    <prevision estado="lluvia">
      <hora>20:00 horas</hora>
      <grados>20 ºC</grados>
    </prevision>
  </previsiones>
</temperatura>
```

*Código 5. Datos meteorológicos en XML.*

**Solución:** La solución completa puedes consultar en el anexo del tema.

# / 5. Validación XML

La validación XML va a permitir comprobar si un documento XML se ajusta estructuralmente a un conjunto de reglas definidas.

El concepto de **validación** no hay que confundirlo con el concepto de documento bien formado. Un documento está **bien formado** cuando cumple las reglas sintácticas definidas para XML (por ejemplo, cerrar etiquetas, emparejar etiquetas, etc.) y es válido cuando cumple las reglas de validación asociadas. Un documento válido siempre va a estar bien formado.

Existen dos formas de validar un documento XML:

- **DTD (*Document Type Definition*):** metalenguaje que permite la descripción del formato que tienen que cumplir los datos del documento XML.
- **XML *Schema*:** lenguaje basado en XML que mejora las características de los DTD para realizar la validación y soporta la definición de tipos de datos.

En este tema, nos centraremos en la forma DTD, y en el siguiente profundizaremos sobre XML *Schema*.

Un DTD va a describir la estructura de un documento XML indicando cuáles son los elementos que tienen que aparecer, el orden, número de apariciones, etc.

Lo primero que hay que referenciar cuando se trabaja con DTD es la indicación de donde se encuentran las reglas de validación y las características del DTD. Para ello, se tiene la siguiente clasificación:

- **Ubicación:**
  - **Interno:** Las reglas están en el propio XML.

- **Externo:** Las reglas están en un fichero externo indicado por su URL.
- **Mixto:** Dispone de reglas internas y externas.

- **Carácter del DTD:**
  - **Privado:** El uso del DTD es privado y se indica con la palabra reservada *SYSTEM*.
  - **Público:** El DTD es de uso público, se indica con la palabra reservada PUBLIC y debe de ir acompañada de un FPI (*Formal Public Identifier*).

Para indicar el tipo de documento con el que se va a trabajar, se usa la etiqueta <!DOCTYPE>. Las combinaciones más usuales son:

- **DTD interno**

```
<!DOCTYPE elemento_raiz [reglas]>
```

- **DTD externo y privado**

```
<!DOCTYPE elemento_raiz SYSTEM URL>
```

- **DTD externo y público**

```
<!DOCTYPE elemento_raiz PUBLIC FPI URL>
```

El elemento_raiz es común a todas las combinaciones e identifica el nodo raíz del árbol XML.

*Audio 1. "DTD externo"*
*https://bit.ly/33eqBlp*

# / 6. Tipos de elementos DTD: Elementos

Las declaraciones de marcado que se pueden incluir en un documento DTD se listan a continuación:

- **Elemento (<!ELEMENT>):** Permite definir los elementos permitidos e indica cuál será su contenido.
- **Atributos (<!ATTLIST>):** Información adicional sobre un elemento.
- **Entidades (<!ENTITY>):** Caracteres con un significado especial.

Veremos, a continuación, cada tipo por separado.

Comencemos por la declaración de un tipo: elemento. Esta debe seguir la siguiente sintaxis:

```
<!ELEMENT nombre_elemento modelo_contenido>
```

- **nombre_elemento:** Elemento que aparece en el fichero XML y hay que indicarlo en el DTD sin los signos < y >.
- **modelo_contenido:** Puede contener reglas, datos de cualquier tipo o elementos descendientes.

Las reglas pueden ser:

- **ANY:** El elemento puede tener cualquier contenido. No debe de aparecer en el DTD definitivo y se usa para que valide en todos los casos.
- **EMPTY:** Describe un elemento vacío que no tiene descendientes, pero sí puede tener atributos.

Destacar que con la declaración **(#PCDATA)** permite contener cualquier texto (cadena de caracteres).

Con respecto a descendientes, un elemento puede contener otros elementos descendientes que se indican entre paréntesis. Destacamos:

- **Cardinalidad:** Número de veces que aparece el elemento descendiente. Se indica con los siguientes símbolos que se han de escribir a continuación del elemento.
  - **?:** El elemento aparece 0 o 1 vez.
  - ***:** El elemento aparece 0 a N veces.
  - **+:** El elemento aparece 1 a N veces.
- **Secuencia:** Se indican los elementos que deben de aparecer en el fichero XML. Se usan las siguientes reglas:
  - **X, Y:** Indica que el elemento X tiene que aparecer a continuación de Y.
  - **(X|Y):** Indica que aparece el elemento X o el elemento Y, pero no los dos al mismo tiempo.
  - **Combinación:** Se pueden combinar los símbolos de secuencia y los de cardinalidad.

## 6.1. Ejemplo DTD

Veamos a continuación un ejemplo sobre una biblioteca:

```xml
<?xml version="1.0" encoding="UTF-8"
      standalone="no" ?>
<!DOCTYPE biblioteca SYSTEM "biblio.dtd">
<biblioteca>
  <libro>
    <nombre>Introducción a xml</nombre>
    <autor>Pedro Martínez</autor>
    <autor>Alberto Ruiz</autor>
    <editorial>Paraninfo</editorial>
    <paginas>345</paginas>
    <precio>45</precio>
  </libro>
  <libro>
    <nombre>CSS3</nombre>
    <autor>Luis Ruiz</autor>
    <editorial>Sintesis</editorial>
    <paginas>234</paginas>
    <precio_con_iva>45</precio_con_iva>
  </libro>
  <libro>
    <nombre>El señor de los anillos</nombre>
    <autor>Tolkien</autor>
    <paginas>234</paginas>
    <precio>34</precio>
  </libro>
</biblioteca>
```

*Código 6. XML.*

```
<!ELEMENT biblioteca (libro)+>
<!ELEMENT libro (nombre,autor+,editorial?,
                paginas, (precio|precio_con_iva))>
<!ELEMENT nombre (#PCDATA)>
<!ELEMENT autor (#PCDATA)>
<!ELEMENT editorial (#PCDATA)>
<!ELEMENT paginas (#PCDATA)>
<!ELEMENT precio (#PCDATA)>
<!ELEMENT precio_con_iva (#PCDATA)>
```

*Código 7. DTD ( biblio.dtd)*

Analizando el código expuesto, podemos destacar:

- **El DTD** es externo y privado. El fichero que contiene el DTD es biblio.dtd.
- **El elemento raíz** es biblioteca y se ha indicado en <!DOCTYPE>, al igual que la referencia al fichero biblio.dtd.
- **La biblioteca** contiene libros y cada libro consta de nombre, autor, editorial, páginas y precio, en ese orden.
- **Autor** puede haber uno o más, por ello se indica el símbolo +.
- **La editorial** aparece o no. Por ello se indica con el símbolo ?.
- **Los libros** pueden aparecer con el precio sin IVA o con IVA. Para ello se utiliza el símbolo |.

En el tema 1 se vio como con **XML Copy Editor** se podía ver si un documento estaba bien formado. Pues bien, este programa también permite validar un XML.

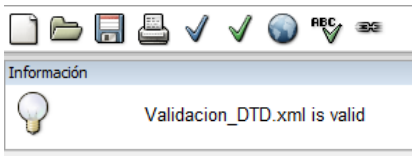



*Fig. 3. Validación del XML a través del DTD.*

Con el DTD indicado se pueden comprobar si los documentos XML son válidos o no. Si, por ejemplo, se intenta crear un XML que no contenga 'Autor' estaría bien formado, pero ya no sería válido. Te animamos a que quites el autor del último libro y observar que el XML ya no sería válido.

# / 7. Tipos de elementos DTD: Atributos

Con la declaración de atributos, incluimos información adicional en un elemento. La sintaxis para incluir un atributo sería:

```
<!ATTLIST elemento atributo tipo_atributo valor>
```

Donde:

- **elemento:** sería el elemento que aparece en el fichero XML y hay que indicarlo en el DTD sin los signos < y >.
- **atributo:** nombre del atributo.
- **tipo_atributo:** es un tipo de la tabla 1.
- **valor:** se indica el valor del atributo o la característica del atributo. Los valores se enumeran en la tabla 2.

| Propiedad | Significado |
|---|---|
| CDATA | Cadena de texto. Valor alfanumérico. |
| Enumeraciones | Permite indicar varios valores º |
| ENTITY | Entidad definida previamente. |
| ENTITIES | Conjunto de entidades. |
| IDREF | Se referencia el identificador de otro elemento. |
| IDREFS | Referencia un conjunto de identificadores. |
| ID | Identificador único para identificar elementos. |
| NOTATION | Se indican datos que no es XML. |
| NMTOKEN | Un nombre sin espacios en blanco. |
| NMTOKENS | Lista de nombres. |

*Tabla 1. Lista de atributos.*

| Propiedad | Significado |
|---|---|
| Valor | El valor por defecto del atributo entre comillas. |
| #REQUIRED | El atributo es obligatorio. |
| #IMPLIED | El atributo es opcional. |
| #FIXED valor | El atributo es obligatorio y se le asigna un valor por defecto que es el único valor que puede tener. |

*Tabla 2. Comportamiento de los atributos.*

Para indicar el comportamiento de los atributos se pueden usar los valores siguientes:

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<!DOCTYPE empleados
[
  <!ELEMENT empleados (empleado)+>
  <!ELEMENT empleado (nombre,edad)>
  <!ELEMENT nombre (#PCDATA)>
  <!ELEMENT edad (#PCDATA)>
  <!ATTLIST empleado
     sexo (v|h) #REQUIRED
     sueldo CDATA #IMPLIED>
]>

<empleados>
    <empleado sexo="v" sueldo="1800">
        <nombre>Alberto Ruiz</nombre>
        <edad>56</edad>
    </empleado>
    <empleado sexo="h" sueldo="1600">
        <nombre>Gloria Martín</nombre>
        <edad>35</edad>
    </empleado>
</empleados>
```

*Código 8. Uso de atributos.*

# / 8. Tipos de elementos de DTD: Entidades

Las entidades las podemos considerar constantes que se declaran con: <!ENTITY>, para luego poder utilizarlas en el fichero XML.

La sintaxis sería:

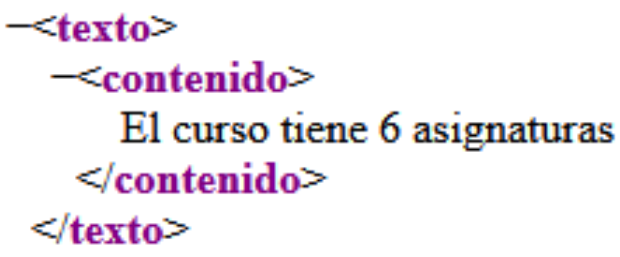
- **<!ENTITY nombre_entidad valor>**

Una vez declarada la entidad, para poderla usar en el fichero XML, se tiene que utilizar el carácter *ampersand* (&) seguido del nombre de la entidad y finalizando con punto y coma (;). Posteriormente, el procesador XML sustituirá el valor indicado en el propio documento XML. En el siguiente ejemplo se muestra su uso:

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<!DOCTYPE texto
[
  <!ELEMENT texto (contenido)>
  <!ENTITY asignaturas "6">
  <!ELEMENT contenido (#PCDATA)>
]>
<texto>
   <contenido>El curso
              tiene &asignaturas;
              asignaturas </contenido>
</texto>
```

*Código 9. Uso de entidades.*

Si lo abrimos en el navegador, obtendríamos la siguiente salida:




*Código 10. Visualización en el navegador.*

Como el XML no tiene asociado ningún fichero CSS, la salida que se muestra en el navegador es la estándar. Como se observa, se ha sustituido la entidad '&asignaturas' por el valor incluido en <!ENTITY>.

Existen **cinco entidades predefinidas** en el lenguaje:

- <: Es el signo <.
- >: Es el signo >.
- ": Son las comillas rectas dobles, ".
- ': Es el apóstrofe o comilla simple, '.
- &: Es el ampersand, &.

No obstante, DTD presenta ciertas limitaciones y por eso su uso se ha limitado a determinados ámbitos, siendo relegada por la validación XSD, que se estudiará en el siguiente tema. Entre las limitaciones principales caben destacar:

- El DTD no es un documento XML.
- No se pueden asignar tipos de datos a los elementos.
- La cardinalidad es muy limitada.

*Vídeo 2. "Validación DTD"*
*https://bit.ly/334DGNQ*

# / 9. Caso práctico 2: "Cursos"

**Planteamiento:** En WebMalagaDesign se van a impartir una serie de cursos, y Gloria, para tener ordenada la información de cada uno de ellos, piensa en crear una estructura XML. Además, quiere crear un DTD para que cuando se estén introduciendo nuevos cursos, se amolden a la estructura XML creada y se pueda saber si son documentos válidos o no.

**Nudo:** Gloria se reúne con Alberto y estudian los datos a incluir. Después de la reunión, llegan a las siguientes conclusiones:

- Tiene que haber un elemento raíz 'listacursos' (con uno o más cursos).
- El curso está identificado con un código.
- Un curso tiene uno o más profesores.
- Un profesor tiene DNI, un nombre y un apellido, puede que tenga segundo apellido o no.
- Un curso tiene cero o más alumnos. Se ha creado el curso, pero aún no hay alumnos.
- Todo alumno tiene un DNI, un nombre y un apellido, puede que tenga segundo apellido o no.
- El contenido del curso debe incluir la descripción, las horas y puede aparecer o no los créditos.

**Desenlace:**

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE listacursos [
        <!ELEMENT listacursos (curso+)>
        <!ELEMENT curso (profesor+,alumno*,contenido)>
         <!ATTLIST curso codigo CDATA  #REQUIRED>
        <!ELEMENT profesor (dni,nombre,ap1,ap2?)>
        <!ELEMENT alumno (dni,nombre,ap1,ap2?)>
        <!ELEMENT contenido (descripcion,horas,creditos?)>

        <!ELEMENT  dni (#PCDATA)>
        <!ELEMENT  nombre (#PCDATA)>
        <!ELEMENT  ap1 (#PCDATA)>
        <!ELEMENT  ap2 (#PCDATA)>
        <!ELEMENT  descripcion (#PCDATA)>
        <!ELEMENT  horas (#PCDATA)>
        <!ELEMENT  creditos (#PCDATA)>
]>
<listacursos>
    <curso codigo="LM">
```

```
        <profesor>
            <dni>30515455K</dni>
            <nombre>Tomás</nombre>
            <ap1>Martínez</ap1>
            <ap2>Ruiz</ap2>
        </profesor>
        <alumno>
                <dni>33354697A</dni>
                <nombre>Silvia</nombre>
                <ap1>Ruiz</ap1>
        </alumno>
        <contenido>
                <descripcion> Introducción a
                los lenguajes de marcas</descripcion>
                <horas>200</horas>
                <creditos>4</creditos>
        </contenido>
    </curso>
'listacursos>
```

*Código 11. DTD y XML para la lista de cursos.*

# / 10. Resumen y resolución del caso práctico de la unidad

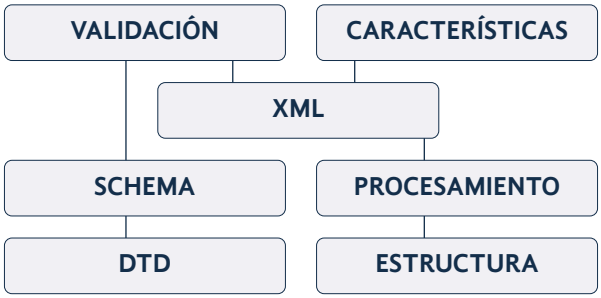



*Fig. 4. Esquema del tema.*

En este tema, hemos visto las características fundamentales del lenguaje de marcado XML, y su uso principal como contenedor de información. Una buena estructura es fundamental para que el procesamiento posterior sea óptimo.

El XML se puede procesar con una hoja de estilos para darle formato y también se le puede asociar un DTD o un SCHEMA para saber si el documento es válido. El documento de validación se puede considerar como una plantilla a la que se tiene que amoldar el XML.

**Resolución del caso práctico inicial**

A continuación, se muestra la resolución del caso práctico inicial, en donde se planteaba la necesidad de elaborar un inventario del software utilizado en la empresa Web Málaga Design.

```
<?xml version="1.0" encoding="utf-8" standalone="yes" ?>
<!DOCTYPE software [
  <!-- DEFINICIÓN DE ELEMENTOS -->
  <!ELEMENT software (programa)+>
  <!ELEMENT programa (titulo,(español|ingles),
  precio,categoria,descripcion,
  departamentos,web?,caratula?)>
  <!ELEMENT titulo (#PCDATA)>
  <!ELEMENT ingles EMPTY>
  <!ELEMENT español EMPTY>
  <!ELEMENT precio (#PCDATA)>
  <!ELEMENT categoria (#PCDATA)>
  <!ELEMENT descripcion (#PCDATA)>
  <!ELEMENT departamentos (departamento)+>
  <!ELEMENT departamento (#PCDATA)>
  <!ELEMENT web (#PCDATA)>
  <!ELEMENT caratula (#PCDATA)>
  <!-- Definición de atributos -->
  <!ATTLIST programa codigo CDATA #REQUIRED>
  <!ATTLIST programa version CDATA #REQUIRED>
  <!ATTLIST programa año CDATA #IMPLIED>
  <!ATTLIST programa
    sistema_operativo (windows|mac) "windows">
]>
<software>
  <programa codigo="p1" version="15.0" año="2015"
            sistema_operativo="mac">
    <titulo>Adobe Photoshop</titulo>
    <ingles />
    <precio>120€</precio>
    <categoria>Diseño</categoria>
    <descripcion>
        Programa para el diseño de elementos gráficos
        para la web. Dispone de numeros plugin´s
        que ayudan en el trabajo de creación de imágenes.
    </descripcion>
    <departamentos>
      <departamento>diseño</departamento>
      <departamento>informática</departamento>
    </departamentos>
    <web>http://www.adobe.es</web>
    <caratula>caratulas/photoshop.jpg</caratula>
  </programa>
</software>
```

*Código 12. XML con un programa de ejemplo.*

**Solución:** La solución completa con el DTD la puedes encontrar en el anexo de la unidad.

# / 11. Webgrafía

Servicio de validación online: Permite a través de W3C validar un documento XML.

Ejemplo completo XML+DTD+XSD (Parte 1): Desarrollo completo de un ejercicio con validación DTD+XSD. La parte XSD la puedes dejar para verla en el tema siguiente.

Ejemplo completo XML+DTD+XSD (Parte 2): Desarrollo completo de un ejercicio con validación DTD+XSD. La parte XSD la puedes dejar para verla en el tema siguiente.