Tema 1. Introducción a los lenguajes de marcas.

Tema 1. Introducción a los lenguajes de marcas

Un **lenguaje de marcado** o **lenguaje de marcas** es una forma de codificar un documento que, junto con el texto, incorpora etiquetas o marcas que contienen información adicional acerca de la estructura del texto o su presentación.

Aunque no se tenga mucha experiencia en ciencias de la información, el párrafo anterior sería un buen comienzo para este tema, pero para los más novatos quizás no quede tan claro; por lo que empezaremos por algunos asuntillos básicos para recordar o fijar conceptos, según los casos.

1. La información y su representación.

Las personas para registrar datos e informaciones, utilizamos unos sistemas de representación compuestos de una serie de símbolos o códigos adecuados. Estos sistemas de representación han ido evolucionando desde los más primitivos (símbolos gráficos, ideogramas, etc.) hasta los utilizados actualmente (alfabetos, sistemas de numeración, etc.).

Los ordenadores para tratar los datos y las informaciones, usan unos sistemas de representación que no son los que normalmente utilizamos las personas. Necesitan que dicha información esté digitalizada y por ello usan códigos binarios (un código de representación donde se usan sólo dos únicos símbolos). Normalmente se usan como símbolos los dígitos 0 y 1.

Existen muchos códigos de representación binarios dependiendo del tipo de información a representar. Por ejemplo para representar cantidades se usan sistemas de numeración binarios; para representar texto se usan sistemas de codificación de forma de que cada carácter se hace corresponder con una secuencia determinada de dígitos binarios. De igual forma, tanto el audio como el vídeo, se pueden codificar en binario mediante diferentes técnicas de muestreo de las señales analógicas.

Además, para cada tipo de información (texto, cantidades, imágenes, vídeos…), no existe un único código de representación, sino cientos de códigos aunque normalmente se usan los que están normalizados y son un estándar.

1.1. Codificación de caracteres.

La codificación de caracteres es el método que permite convertir un carácter o símbolo de un lenguaje natural (español, francés, etc.) en un símbolo o conjunto de ellos de otro sistema de representación, aplicando normas o reglas de codificación. En informática, los caracteres se codifican como un conjunto de dígitos binarios, es decir mediante secuencias de ceros y unos.

Comentaremos algunas de las normas de codificación más usuales empleadas en Informática.

1.1.1. Código ASCII.

El código ASCII (acrónimo inglés de *American Standard Code for Information Interchange* — Código Estadounidense Estándar para el Intercambio de Información), es un código de caracteres basado en el alfabeto latino para representar los caracteres del idioma inglés. Creado en 1963 por el Comité Estadounidense de Estándares (ANSI) como una refundición o evolución de los códigos utilizados entonces en telegrafía, para poner orden entre los diferentes sistemas de codificación de la época.

El código ASCII utiliza 7 bits para representar los caracteres, lo que significa que usa cadenas de bits de siete dígitos binarios (que van de 0 a 127 en base decimal) para representar información de caracteres. Con el código ASCII se pueden representar hasta 128 caracteres distintos. Esto permitía codificar todas las letras minúsculas, mayúsculas, símbolos de puntuación, especiales y hasta caracteres de control.

El código ASCII reserva los primeros 32 códigos (numerados del 0 al 31 en decimal) para caracteres de *control*: códigos no pensados originalmente para representar información imprimible, sino para controlar dispositivos (como impresoras) que usaban ASCII. Así por ejemplo, el carácter 10 representa la función "nueva línea" (*line feed*), que hace que una impresora haga avanzar el papel una línea.

Los códigos del 33 al 126 se conocen como caracteres *imprimibles*, y representan letras, dígitos, signos de puntuación y especiales. Por ejemplo, la letra A se representa es ASCII como la serie binaria 1000001, o 65 en representación decimal, y la letra B como 1000010, o 66 en decimal.

1.1.2. ASCII Extendido.

A partir del código ASCII se han desarrollado muchas variaciones con el fin de facilitar la codificación de otras lenguas diferentes al inglés pero que también usan alfabetos latinos. Todas éstas variaciones clasificadas como **ASCII Extendido** usan 8 bits para representar cada carácter, pero en todas ellas los códigos 32 a 126 coinciden con los caracteres imprimibles del código ASCII.

Como este código usa 8 bits, se dispone de 256 combinaciones distintas (128 más que con ASCII), lo que permite que se puedan ya codificar los caracteres propios de la mayoría de los lenguajes occidentales.

Las codificaciones "ASCII Extendido" que usamos en Europa Occidental son la ISO 8859-1 (también llamada Latin-1), y la ISO 8859-15 la cual añade el símbolo de Euro (€) y otros más a la ISO 8859-1. Pero hay muchas otras codificaciones "ASCII Extendido" dependiendo del conjunto de idiomas que pueda representar. Así tenemos la ISO 8859-2 para idiomas de Europa del Este, ISO 8859-5 para el alfabeto cirílico, ISO 8859-6 para el alfabeto árabe, etc.

Sin embargo, el problema de estos códigos de 8 bits es que cada uno de ellos se define para un conjunto de lenguas con escrituras semejantes y, por tanto, no dan una solución unificada a la codificación de todas las lenguas del mundo. Es decir, no son suficientes 8 bits para codificar todos los alfabetos y escrituras mundialmente conocidos.

1.1.3. UNICODE.

Como solución a estos problemas, desde 1991 se ha acordado internacionalmente utilizar la norma Unicode, que es una gran tabla, en la que se asigna un código a cada uno de los más de cincuenta mil símbolos que contiene. Los símbolos abarcan todos los alfabetos europeos, ideogramas chinos, japoneses, coreanos, muchas otras formas de escritura, y más de un millar de símbolos especiales.

Unicode define tres formas distintas para codificar los caracteres bajo el nombre **UTF** o Formato de Transformación Unicode (*Unicode Transformation Format*):

UTF-8. Usa símbolos de longitud variable (de 1 a 4 bytes por carácter según el símbolo a representar), e incluye la especificación ASCII de 7 bits, por lo que cualquier mensaje ASCII se representa sin cambios. Es la forma más usada.

UTF-16. A diferencia del UTF-8, usa 2 o 4 bytes para representar cada carácter.

UTF-32. Usa 32 bits (4 bytes) para representar cada carácter, siendo ésta la forma más sencilla de las tres aunque en realidad no se usa.

2. Archivos binarios y de texto

Sabemos que a la hora de crear documentos de texto se usa un software llamado editor o procesador de texto que permite además de escribir el texto, almacenar o recuperar el documento en/desde un archivo, en el cual los caracteres han sido codificados usando algunos de los códigos de representación de caracteres comentados anteriormente: ASCII, ISO 8859-15, UTF-8, etc.

Por otra parte, la mayoría de procesadores de textos, incluso distintas versiones del mismo fabricante, son incompatibles entre sí en el sentido de que no son capaces de entender los formatos de documentos distintos al suyo propio. El motivo es que estos documentos contienen, además del texto o mensaje, una serie de marcas propias y específicas que permiten una presentación en pantalla determinada. Por ejemplo, el efecto de presentar un texto en **negrita** se logra porque dicho texto está señalado mediante unas marcas, (marcado) para que se presente con dicho estilo.

Dichas marcas se consiguen con una serie de bits entremezclados entre el texto, creándose realmente un archivo binario. Un archivo *binario* es simplemente un archivo que almacena una tira de bits con un significado propio para el software que lo creó. Estas marcas o informaciones adicionales se denominan meta-datos (información sobre la información). Un archivo binario creado con un procesador de texto determinado, sólo es entendible por el procesador de textos que lo creó.

Por lo tanto, el problema de incompatibilidad entre distintos procesadores de texto proviene del hecho de que realmente crean archivos binarios que usan meta-datos propios para establecer formatos incompatibles entre sí. Como luego veremos, este marcado que usan los procesadores de texto se denomina “marcado de presentación”.

Hay editores de texto en donde no se permiten meta-datos para un marcado de presentación, de forma que sólo almacenan el texto en sí. Estos editores se le suelen llamar "editores planos" o "editores ASCII" debido a que inicialmente usaban el código ASCII de representación. Los archivos que se crean con ellos se denominan archivos de *texto*. Realmente, también se almacenan como una tira de bits, pero no encontraremos meta-datos entre los caracteres del texto que impidan descodificar o entender el texto.

Los editores planos pueden, por lo tanto, trabajar con cualquier archivo de texto creado con otro editor plano. Hay muchísimos editores planos para usar, de hecho todos los sistemas operativos suelen traer alguno (Bloc de notas en Windows, vi en Unix, etc.).

Resumiendo, los editores de textos "planos" se distinguen de los llamados procesadores de textos en que los primeros se usan para escribir archivos de texto (sólo texto, sin formato y sin imágenes, es decir sin diagramación); y los segundos crean archivos binarios (texto mezclado con meta-dados para establecer un marcado de presentación).

2.1. Meta-datos en archivos de texto.

La desventaja de los archivos de texto es que si queremos añadir otra información que no sea texto, en otras palabras, meta-datos lo tenemos más complicado. Por ejemplo, la mayoría de los procesadores de texto, además de almacenar un texto en su formato propio, en binario, también tienen la capacidad de almacenarlos simplemente en formato texto, pero por ejemplo, si dicho documento tiene una sección en negrita o bien una imagen, sólo se almacenará el texto, perdiéndose el formato y la imagen.

Los archivos binarios tienen la ventaja de que son compactos y se les pueden añadir meta-datos, pero carecen de la propiedad de los archivos de texto de ser intercambiables y por así decirlo "universales". En cambio los archivos de texto aunque son perfectamente intercambiables, sólo pueden almacenar texto.

¿No habría un formato ideal que combinara la universalidad de los archivos de texto con la capacidad de añadir información adicional o meta-datos de los archivos binarios? Pues sí.

Por ejemplo, algunos procesadores de texto optaron por guardar toda la información como texto, haciendo que las indicaciones de formato no se almacenen de forma binaria sino textual. Dichas indicaciones son caracteres señalados o *marcados* de manera especial para que así un programa adecuado pueda traducir dichos caracteres no como texto, sino como operaciones que finalmente mostrarán el texto del documento de una forma adecuada.

La idea del marcado procede de *marking up*, una técnica para marcar manuscritos con lápices de colores para hacer anotaciones, como por ejemplo en la tipografía que se empleaban en las imprentas. Este mismo término se ha utilizado para los documentos de texto que contienen comandos o anotaciones.

Las posibles anotaciones o indicaciones incluidas en los documentos de texto han dado lugar a *lenguajes* (entendiendo que en realidad son formatos de documento y no lenguajes en el sentido de los lenguajes de programación de aplicaciones) llamados lenguajes de marcas o de etiquetas.

Uno de los primeros intentos para combinar un formato de datos que fuera universal que admitiera meta-datos en formato texto fue el *Standard Generalized Markup Language* (SGML); es decir: Lenguaje de Marcado Generalizado Estándar.

Actualmente, unos de los lenguajes de marcas más usado y exitoso es HTML *HyperText Markup Language* (Lenguaje de Marcado de Hipertexto). Utiliza muchos de los conceptos de SGML consiguiéndose un lenguaje de marcas universal para mostrar información y poder enlazarlas con otras informaciones. La idea fundamental con la que se creó fue que cualquier documento HTML pudiera mostrarse con cualquier aplicación capaz de entender HTML (básicamente exploradores web) y que, además, pudiera mostrar la información enlazada mediante enlaces (hiperenlaces).

3. Los lenguajes de marcas.

Como ya hemos comentado, los lenguajes de marcas, también denominados lenguajes de marcado, son aquellos que combinan la información que contiene un documento, generalmente texto, con marcas textuales o anotaciones relativas a la estructura del texto y/o a la forma de presentarlo.

Por lo tanto, en un documento existen distintos niveles de información. Por un lado están los datos que conforman el contenido de un documento y, por otro, una información superpuesta al contenido o meta-datos, que es lo que constituye el etiquetado, marcado o *mark up* (caracteres de etiquetado).

En general los lenguajes de marcado cumplen con dos objetivos esenciales a la hora de diseñar y procesar un documento digital:

a) Especificar las operaciones tipográficas y las funciones que debe ejecutar el programa visualizador sobre dichos elementos. Estas operaciones tipográficas son instrucciones de formato que se aplican a cada uno de los elementos de un documento digital como, por ejemplo, imprimir un título en negrita y a un determinado tamaño.

b) Estructurar un documento separando el texto en los elementos de los que se compone, como por ejemplo un párrafo, un capítulo, una sección, una nota, un encabezamiento, etc.

3.1. Tipos de lenguajes de marcas.

Por regla general, se distinguen dos tipos básicos de lenguajes de marcado:

a) Lenguaje de marcado de procedimiento o procesado: Las anotaciones o marcas de los lenguajes de procedimiento describen la forma y el significado de las operaciones tipográficas que van a ser aplicadas a cada uno de los elementos del documento. Por ejemplo, una regla de un lenguaje de procedimiento indicaría que el título de la sección de un texto debe ser impreso en una sola línea con una fuente de seis puntos más grande que el resto del texto, con objeto de que los lectores puedan inferir que es el título.

Se refiere, pues, a la apariencia física o formato (fuente, estilo de letra, tamaño, etc.) tanto del documento en pantalla como del documento impreso. Lenguajes de marcado de procedimientos son *nroff*, *troff*, *TeX*. Este tipo de marcado se ha usado extensivamente en aplicaciones de edición profesional, manipulados por tipógrafos cualificados, ya que pueden llegar a ser extremadamente complejos.

b) Lenguaje de marcado estructural o descriptivo: En los lenguajes estructurales las marcas o anotaciones únicamente describen la estructura lógica del documento digital y/o la descripción semántica del contenido, no su tipografía. Se utilizan etiquetas para describir los fragmentos de texto, pero sin especificar cómo deben ser representados, o en que orden. Lenguajes expresamente diseñados para generar marcado descriptivo son el *SGML*, y el *XML* (una adaptación y simplificación de SGML).

Por lo tanto la descripción estructural se basa en especificar la estructura lógica del contenido (títulos, párrafos de texto normal, enumeraciones, definiciones, citas, etc.).

Hay lenguajes de marcado que permiten un marcado tanto de procedimiento como estructural. En HTML, la descripción se basa en especificar en el texto la estructura lógica del contenido (títulos, párrafos de texto normal, enumeraciones, definiciones, citas, listas, etc.) así como los diferentes efectos que se quieren dar (especificar los lugares del documento donde se debe poner cursiva, negrita, o un gráfico determinado) y dejar que luego la presentación final de dicho hipertexto se realice por el explorador o navegador.

En general los lenguajes de marcado siguen una sintaxis basada en el uso de marcas o **etiquetas**: una etiqueta indica el principio de un elemento, y otra etiqueta el final del mismo. Así, una pareja de marcas o etiquetas encierran el texto al que afectan. Veamos un ejemplo basado en HTML:

```
<p>En este párrafo, una palabra aparecerá en <b>negrita</b>, otra en <i>cursiva</i>  y otras en
<b><i>negrita y cursiva</i></b> </p>
```

El contenido HTML anterior que presentaría un explorador sería:

En este párrafo, una palabra aparecerá en **negrita**, otra en *cursiva* y otras en ***negrita y cursiva***

En el ejemplo anterior, todo el texto está encerrado entre las marcas o etiquetas . Estas dos marcas indican que el texto enmarcado constituye un párrafo. Vemos como indica el inicio de la marca y el final. De igual forma, las etiquetas sirven para formatear texto en negrita y las etiquetas para formatearlo en cursiva. Por regla general, las etiquetas o marcas aparecen por parejas con el formato <etiqueta> </etiqueta>.

3.2. Historia y evolución de los lenguajes de marcas.

Aunque ya hemos comentado algunos lenguajes de marcas, los repasaremos y añadiremos algunos lenguajes más junto a las tecnologías relacionadas con varios de ellos.

3.2.1. SGML.

Allá por los años 70, en IBM surgió la necesidad de cómo almacenar grandes cantidades de información acerca temas diversos. Con este fin, los expertos de IBM encabezados por Charles Goldfarb diseñaron GML (*Generalized Markup Language*), un lenguaje con el que poder clasificar todo tipo de información y escribirla para que se pudiera procesar luego adecuadamente de forma automática.

Este lenguaje gustó mucho, y el ISO (entidad que se encarga de normalizar hasta lo inimaginable) lo normalizó en el año 1986 creando el SGML (*Standard Generalized Markup Language*). Básicamente era el GML original, pero estandarizado mediante la Norma ISO 8879.

SGML es un lenguaje basado en texto que se puede usar para marcar datos, es decir para añadir meta-datos, que permiten estructurar el texto de manera lógica y también cómo debe presentarse en pantalla.

En realidad SGML no es un lenguaje con unas etiquetas concretas, sino que se trata de un lenguaje que sirve para definir lenguajes de etiquetas; o más exactamente es un lenguaje de marcado que sirve para definir formatos de documentos de texto con marcas. Entre los formatos definidos mediante SGML, sin duda HTML es el más popular.

Este marcado se realiza definiendo una serie de etiquetas válidas para cada tipo de documento, incluyendo también las reglas relativas a la estructura lógica. La forma de usar las etiquetas y las reglas que se deben cumplir para estructurar la información las veremos en un ejemplo, pero usando XML (otro lenguaje de marcas creado a partir de SGML).

Realmente, SGML es un lenguaje muy trabajado, capaz de adaptarse a un gran tipo de problemas, pero bastante complicado de utilizar. Precisamente, ésta fue una de las razones (había otras) por las que surgió XML (*e**X**tensible **M**arkup **L**anguage)*, lenguaje de marcas bastante más simple de usar.

3.2.2. HTML

HTML (*Hipertext Markup Language)* o lenguaje de marcas de hipertexto, es el lenguaje que permite la generación de documentos de hipertexto en la World Wide Web. El lenguaje HTML deriva de SGML pero, a su vez, ha tenido unos desarrollos posteriores.

Podemos definir el **hipertexto** como una manera de organizar la información de forma no secuencial. Mediante el modelo de hipertexto un sistema puede organizar y presentar los documentos en un medio informático, basado en la vinculación de documentos o fragmentos de documentales digitales (textuales o gráficos) a otros fragmentos o documentos, lo que permite acceder a la información no necesariamente de forma secuencial.

La World Wide Web, comúnmente llamada Web o WWW, nace a principios de los años 90, aunque sus orígenes se remontan mucho tiempo atrás. Fue creada por Tim Berners-Lee en el Centro Europeo de Física Nuclear (CERN) con el objetivo de servir como herramienta para la búsqueda y transmisión de información entre los científicos del CERN.

El hipertexto es la base funcional y estructural de la World Wide Web. Podríamos decir que la Web es un hipertexto de escala planetaria puesto que cualquier usuario puede poner su página en la Red y establecer enlaces a cualquiera de los documentos disponibles en ella. Por lo tanto, las páginas web se enlazan unas a otras dentro de un sitio *web* y pueden conectarse a otros sitios *web* llevando al usuario de un servidor a otro sin necesidad de teclear ninguna ruta.

HTML es un lenguaje sencillo que permite describir hipertexto. La descripción se basa en especificar en el texto la estructura lógica del contenido (títulos, párrafos de texto normal, enumeraciones, definiciones, citas, listas, etc.), así como, los diferentes efectos que se quieren dar (especificar los lugares del documento donde se debe poner cursiva, negrita, o un gráfico determinado) y dejar que luego la presentación final de dicho hipertexto se realice por medio del explorador o navegador.

El lenguaje HTML ha sido sometido y lo sigue estando a constantes cambios y a un crecimiento vertiginoso. La tarea de determinar su sintaxis recae en el Consorcio World Wide Web (W3C) (http://www.w3.org). Su objetivo es la creación de documentos estandarizados que puedan difundirse sin problemas por la red y que puedan ser interpretados por los diferentes navegadores.

Para obtener cualquier información sobre el lenguaje HTML se puede consultar el sitio web del World Wide Web Consortium (http://www.w3.org/TR/#tr_HTML) donde se publican las Recomendaciones y Especificaciones sobre este lenguaje, así como, las distintas versiones normalizadas.

Básicamente, HTML utiliza los mismos tipos de elementos que el lenguaje SGML: etiquetas, o marcas (*tags*, en inglés) que igualmente están compuestas de códigos enmarcados por los signos '<' y '>'.

Al estar HTML basado en texto plano, podemos crear documentos HTML simplemente usando cualquier editor plano. No obstante, existen aplicaciones específicas que permiten crear este tipo de documentos de manera más simple y mucho más rápida.

En conjunción con HTML, se han desarrollado las llamadas Hojas de Estilo (*Style Sheets*). Una hoja de estilo es una colección de reglas que afecta a la apariencia de un documento. Actualmente el tipo más común de hoja de estilo es la llamada Hoja de Estilo en Cascada (*Cascading Style Sheet* o *CSS*). Estas normas se refieren al modo en que aparecerá un documento en pantalla cuando el usuario utilice un navegador, controlando por ejemplo el color, el fondo, tipo de fuente, apariencia del borde, márgenes, alineación, espacio entre caracteres, etc.

La últimas versiones de HTML han ido desplazado las funciones de presentación que se realizaban en las versiones anteriores hacia las hojas de estilo, reservando el lenguaje HTML para describir la organización del contenido y dejando a CSS la manera de presentarlo.

En un principio el W3C finalizó el desarrollo de HTML en 1998 a favor de XHTML; pero en el año 2004 un grupo de empresas crearon el grupo de trabajo WHATWG (*Web Hypertext Application Technology Working Group* http://www.whatwg.org/) para continuar con el desarrollo de HTML que el W3C había finalizado.

En 2007 el W3C y WHATWG deciden unirse para el desarrollo de una nueva especificación de HTML, lo que permite publicar en 2008 el primer borrador (*draft)* de HTML 5.

En 2009 el W3C pone fin al desarrollo de XHTML 2.0 a favor de HTML 5, siendo la última versión a día de hoy la de HTML 5.1 (https://www.w3.org/TR/html51)

3.2.3. XML.

XML (*eXtensible Markup Language*) son las siglas de Lenguaje de Etiquetado Extensible. Se trata también de un lenguaje estándar que posee una Recomendación del W3C: (http://www.w3.org/TR/REC-xml/). Con la palabra "Extensible" se alude a la no limitación en el número de etiquetas, ya que como veremos se podrán crear todas aquellas que sean necesarias.

El desarrollo de XML comenzó en 1996 tomando como partida SGML pero simplificándolo para poder trabajar de forma más sencilla. Desde entonces tuvo un desarrollo exponencial, y sólo dos años después, en febrero de 1998, fue adoptado como recomendación por el W3C, quien lanzó la versión 1.0.

XML permite definir múltiples lenguajes específicos, ya que no es un lenguaje particular sino una manera de definir lenguajes para diferentes necesidades; es decir lo que se denomina un metalenguaje.

XML no define cuales son las etiquetas ni cómo se utilizan, sino que ofrece un escaso número de reglas sintácticas para poder crear documentos. Por lo tanto, XML no es un lenguaje, sino un metalenguaje o lenguaje para definir otros lenguajes donde se especifica las etiquetas posibles, dónde colocarse y el significado de cada una de ellas. Es decir un metalenguaje que permite definir la estructura y el vocabulario del lenguaje a definir.

Al contrario de HTML, donde las etiquetas son fijas (un <h1> es siempre un encabezado de primer nivel), en XML las etiquetas sólo se usan para delimitar piezas de datos, dejando la interpretación de éstos a la aplicación que los lee.

La primera definición de XML fue la de "Sistema para definir, validar y compartir formatos de documentos en la Web", aunque como veremos su uso se ha generalizado a otros ámbitos.

La mayor ventaja de XML es que permite representar e intercambiar los datos de forma independiente a su presentación. Además, XML no sólo se aplica en Internet, sino que se propone como un lenguaje de bajo nivel para intercambio de información estructurada entre distintas plataformas. Se puede utilizar en bases de datos, hojas de cálculo, editores de texto, etc. y no sólo en la Web.

Veamos un ejemplo de cómo representar datos en XML. Supongamos que queremos guardar información sobre los datos de una serie de libros. Antes de que existiera XML posiblemente usaríamos un archivo ASCII con el siguiente contenido:

"Visual C#","Fco. Javier Ceballos","Ra-Ma","936","52,75"

"Programación en C","Luis Joyanes Aguilar","McGraw-Hill","735","45,25"

Más o menos, podemos interpretar qué datos estamos registrando y con qué orden (título del libro, autor, editorial, …). Para entender el significado de los dos últimos campos de información deberíamos aclarar que se trata del número de páginas y del precio en euros, respectivamente.

En formato XML podría quedar de la siguiente manera:

```
<biblioteca>
    <libro>
        <titulo>Visual C#</titulo>
        <autor>Fco. Javier Ceballos</autor>
        <editorial>Ra-Ma</editorial>
        <paginas>936</paginas>
        <precio>52,75</precio>
    </libro>
    <libro>
        <titulo>Programación en C</titulo>
        <autor>Luis Joyanes Aguilar</autor>
        <editorial>McGraw-Hill</editorial>
        <paginas>735</paginas>
        <precio>45,25</precio>
    </libro>
</biblioteca>
```

Vemos como los datos se autodescriben gracias al uso de etiquetas o marcas (coloreadas en azul para que resalten) que indican como así decirlo el vocabulario a utilizar. Además podemos ver la estructura de los datos comprobando cómo unos elementos pueden definirse dentro de otros. Aquí las etiquetas permiten describir como se organizan y estructuran los datos, pero no informan de cómo deben presentarse dichos datos.

El formato XML, al ser de tipo texto, permite a las personas reconocer los datos y su estructura pero, además, gracias al uso de las etiquetas y a ciertas reglas de escritura, diferentes programas pueden obtener fácilmente los datos y sus relaciones para su posterior procesamiento y/o transmisión.

Como podemos comprobar en el ejemplo, XML no es realmente un lenguaje en particular, sino una manera de definir lenguajes para diferentes necesidades. En el ejemplo hemos definido un lenguaje con ciertas etiquetas y algunas pequeñas reglas para registrar y transmitir información sobre los libros de una biblioteca. De igual forma podríamos haber definido un lenguaje en XML para registrar facturas.

Como ya hemos comentado, XML se puede usar en bases de datos, editores de texto, hojas de cálculo y casi cualquier cosa imaginable. Los principales usos de XML son los siguientes:

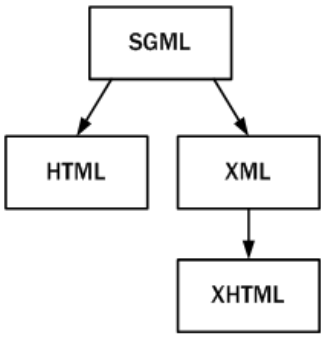
- **XML aplicado a los sitios web**: permite separar contenido y presentación, y que los mismos datos se puedan mostrar de varias formas distintas sin demasiado esfuerzo. (en pantalla, un móvil, papel, etc.)
- **XML para la comunicación entre aplicaciones:** representa los datos de forma muy simple facilitando su transmisión por la red de forma estándar. En los últimos tiempos este uso se está haciendo muy popular con el surgimiento de los *Servicios web*.
- **XML para la configuración de programas**: representándose los parámetros que personalizan una aplicación de una forma simple y estándar, en contraposición con los crípticos formatos propietarios.

XML es una tecnología sencilla que tiene a su alrededor otras muchas que la complementan y la hacen mucho más grande y con unas posibilidades mucho mayores. Tiene un papel muy importante en la actualidad ya que permite la compatibilidad entre sistemas para compartir la información de una manera segura, fiable y fácil.

3.2.4. XHTML.

El XHTML (*eXtensible Hypertext Markup Language*) o Lenguaje de Etiquetado Hipertextual Extensible es una reformulación del lenguaje HTML, pero como una aplicación XML. Está normalizado, siendo la última versión la recogida en la Recomendación del World Wide Web Consortium (W3C) XHTML 2. (https://www.w3.org/TR/xhtml2) .

XHTML es una adaptación de HTML 4.01 manteniendo casi todas sus etiquetas y características, pero añade algunas restricciones y elementos propios de XML. La versión XHTML 1.1 es la más usada ya que la última publicada, la versión XHTML 2.0, supone un cambio muy importante respecto a las versiones anteriores.



El lenguaje XHTML es muy similar al lenguaje HTML. De hecho, XHTML no es más que una adaptación de HTML al lenguaje XML. Pero técnicamente, HTML es descendiente directo del lenguaje SGML, mientras que XHTML lo es del XML (que, a su vez, también es descendiente de SGML).

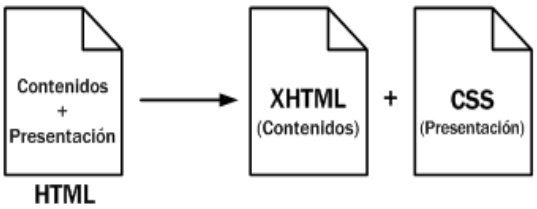
El lenguaje XHTML surgió para dar una solución ante los problemas de compatibilidad que surgían cuando se usaba un documento HTML en distintas plataformas y/o exploradores.

En realidad, los navegadores HTML son muy permisivos y aceptan cualquier entrada, ya sea correcta o no, e intentan mostrar algo perceptible a partir de lo que reciben. Esto supone que una enorme cantidad de documentos HTML sean realmente incorrectos, pero al mostrarse en el navegador sin problemas, el autor no es consciente de los errores. El gran problema es que esta permisibilidad, hace que sea realmente complicado implementar aplicaciones que lean y procesen HTML (como por ejemplo los buscadores), puesto que los documentos que se supone son HTML correctos, en realidad con frecuencia tienen muchos errores.

La finalidad de XHTML es que pueda ser usado como un lenguaje de contenidos pero que, a su vez, sea conforme a XML y, si se siguen algunas sencillas reglas, las aplicaciones que leen y procesan documentos HTML 4.01 seguirán funcionado con XHTML.

Otra ventaja de XHTML es que permite usar una serie de tecnologías que están relacionadas con XML: (XLST, XForm, etc.). Estas tecnologías, como luego veremos, están diseñadas para trabajar con formatos tipo XML, como lo es XHTML, pero no con HTML.

Además, XHTML permite separar de una forma mucho más eficaz los contenidos, del aspecto que deben presentar. Normalmente, en HTML los contenidos y el diseño pueden estar mezclados, de forma que se complica en exceso su mantenimiento.



Usando CSS, hojas de estilos en cascada, podemos separar mejor los contenidos definidos mediante XHTML, del aspecto que deben presentar esos contenidos. De esta forma, los documentos XHTML son más flexibles, ya que se adaptan mejor a las diferentes plataformas: pantallas de ordenador, pantallas de dispositivos móviles, impresoras y dispositivos utilizados por personas discapacitadas.

3.2.5. Tecnologías relacionadas con XML.

a) XML Linking Language XLL: (http://www.w3.org/XML/Linking) o *eXtensible Linking Languaje*, esto es, Lenguaje de Enlaces Extendido. Abarca un conjunto de tecnologías relacionadas con los enlaces y el direccionamiento en lenguaje XML.

- XML Linking Language (XLink): describe cómo se asocian dos o más recursos. (http://www.w3.org/TR/xlink/).
- XML Base (http://www.w3.org/TR/xmlbase/): propone un mecanismo para definir los URIs (Uniform Resource Identifier), cadenas de caracteres que identifica inequívocamente un recurso.
- XML Pointer Language (XPointer): describe cómo se localiza un recurso (http://www.w3.org/TR/xptr/).

Una de las características principales de la Web y del hipertexto es la conectividad que proporciona el uso de enlaces de todo tipo: a imágenes, sonidos, vídeos, otros párrafos dentro del propio texto, otros documentos, etc. En el lenguaje HTML existen etiquetas o marcas para cada caso (generalmente usando la etiqueta `<a>`), pero en XML los enlaces se diseñan de una forma particular.

En XML, el tratamiento de los enlaces o hipervínculos es de tal importancia que el W3C ha redactado dos especificaciones o normas para mejorar y estandarizar su tratamiento: XLink y XPointer, y a estas especificaciones hay que sumar la especificación XML Base, donde se incluye un mecanismo para identificar y localizar los recursos a los que apuntan los enlaces, es decir, el uso de URIs (Uniform Resource Identifier) o Identificador de Recursos Uniforme.

Un URI es una cadena caracteres que identifica inequívocamente un recurso (servicio, página, documento, dirección de correo electrónico, enciclopedia, etc.). Normalmente, estos recursos son accesibles en una red o sistema y necesitan ser identificados de forma única.

b) XSL o Extensible Stylesheet Language. No se trata de un único lenguaje, sino de toda una familia de recomendaciones del World Wide Web Consortium (http://www.w3.org/Style/XSL/) para expresar Hojas de Estilo en lenguaje XML. XSL consta de tres partes:

- XSL Transformations (XSLT): un lenguaje para transformar documentos XML a otros formatos. (http://www.w3.org/TR/xslt y http://www.w3.org/TR/xslt20/)
- XML Path Language (XPath), un lenguaje de expresión usado por XSLT para acceder o referirse a partes de un documento XML. (XPath se usa también en la especificación XLink). (http://www.w3.org/TR/xpath y http://www.w3.org/TR/xpath20/)
- XSL Formatting Objects (XSL-FO): permite especificar el formato visual con el cual se quiere presentar un documento XML. Es usado principalmente para generar documentos PDF. (http://www.w3.org/TR/xsl)

A pesar de la existencia de las CCS u Hojas de Estilo en Cascada que sirven para definir las presentaciones de los documentos en la Web, se ha creado otra forma específica para las presentaciones en XML. EL XSL o eXtensible Stylesheet Language define o implementa un lenguaje de estilo para usarlo con los documentos que están escritos conforme a XML.

Así, un documento XML puede usar para su presentación o estilo tanto CSS como XSL pero, al estar este último incluido dentro de la familia del lenguaje XML, usa su sintaxis. Actualmente con XSL se logran mejoras en la presentación sobre todo cuando los datos son tablas u organigramas.

Además, XSL dispone de un lenguaje de transformación de documentos: XSLT. Con XSLT se puede transformar un documento XML en otro o, incluso, en otros formatos. Por ejemplo, puede usarse para transformar datos XML en documentos HTML/CSS y alojarlos luego en un servidor Web o bien para conseguir los datos en un formato texto.

Aparte de las tecnologías nombradas anteriormente, existen otras múltiples relacionadas con XML. Algunas de las que se nombran a continuación se estudiarán más adelante:

DTD	Document Type Definition	Definición del esquema de documentos XML.
XSD	XML Schema	Definición del esquema de documentos XML (en XML).
XQuery	XML Query language	Mecanismo para hacer consultas en documentos XML.
XMLEnc	XML Encryption	Criptografía para documentos XML.
XMLDSig	XML-Signature	Firmas digitales en XML.
XKMS	XML Key Management	Gestión de claves en XML.
DOM	Document Object Model	API para crear, acceder y modificar documentos XML.
SAX	Simple API for XML	API para trabajar con documentos XML.
XForms	XML Forms	Formularios web en XML.
Data Island		Inclusión de datos XML en un documento HTML.
Data Binding		Generación automática de HTML a partir de XML.

3.2.6. Otros lenguajes de marcas.

- **Tex** y **LaTeX**. En la década de los 70 Donald Knuth creó Tex para generar documentos científicos utilizando una tipografía y capacidades que fueran iguales en cualquier ordenador, y que presentara una gran calidad en los resultados. TeX usa comandos para crear documentos pero necesita un programa capaz de convertir el archivo TeX a un formato de impresión.

  El éxito de TeX produjo numerosos derivados de los cuales el más popular es LaTeX. Se trata de un lenguaje que intenta simplificar a TeX, usando menos comandos pero manteniendo su estructura tipográfica. Es muy utilizado para producir documentos con expresiones científicas de gran calidad. La idea es que los científicos se centren en el contenido y no en la presentación.

Ejemplo de código LaTeX:

```
\documentclass[12pt]{article}
\usepackage{amsmath}
\title{\Ejemplo}
\begin{document}
Este es el texto ejemplo de \LaTeX{}
Con datos en \emph{cursiva} o \textbf{negrita}.
Ejemplo de f\'ormula
\begin{align}
E &= mc^2
\end{align}
\end{document}
```

- **RTF**. Es el acrónimo de *Rich Text Format*, un lenguaje ideado por Microsoft en 1987 para producir documentos de texto que incluyan anotaciones de formato. En entorno Windows es muy utilizado como formato de intercambio entre distintos procesadores por su potencia. El procesador de texto Word Pad lo utiliza como formato nativo.

- **PostScript**. Es un lenguaje de descripción de páginas; quizás el más popular. Permite crear documentos en los que se dan indicaciones sobre cómo mostrar información en el dispositivo final. John Warnock inició su desarrollo en 1976, lo continuó en la empresa Xerox hasta que en 1982 el propio Warnock funda Adobe Systems en donde se continúa su desarrollo.

  En realidad es como un lenguaje de programación que indica la forma en que se debe mostrar la información la cual puede incluir texto y el tipo de letra del mismo, píxeles individuales y formas vectoriales (líneas, curvas). Sus posibilidades son muy amplias.

- **JSON**. Abreviatura de *JavaScript Object Notation*. Se trata de una notación de datos procedente del lenguaje *JavaScript* con el fin de compactar y evaluar la información que normalmente usan las aplicaciones web en formato XML. A partir del año 2002 se le empezó a dar soporte por parte de los navegadores y su fama ha sido tal que compite claramente con XML.

  Se trata realmente de una notación, y no se considera técnicamente un lenguaje de marcas, ya que no hay diferencia en el texto a través de etiquetas, sino que se basa en que el texto se divide en datos y metadatos. El símbolo de los dos puntos separa el metadato del dato. Por otro lado los símbolos de llave y corchete permiten agrupar de manera correcta los datos. Ejemplo de JSON:

```
{
   "menu":
   {
       "id": "file",
       "value": "File",
       "popup":
       {
        "menuitem":
            [
                {"value": "New", "onclick": "CreateNewDoc()"},
                {"value": "Open", "onclick": "OpenDoc()"},
                {"value": "Close", "onclick": "CloseDoc()"}
            ]
       }
   }
}
```

3.2.7. Lenguajes de marcado especializados basados en XML.

Como vimos con el ejemplo de XML para registrar datos sobre libros, XML nos permitió crear nuestro propio conjunto de etiquetas con un significado específico para describir nuestros datos. Según el ejemplo, cuando una aplicación programada para ello lea el documento XML, entenderá que <biblioteca> hace referencia a un conjunto de libros, y que <libro> representa a un solo libro. Lo mismo sucede con las etiquetas que describen la estructura de un libro: (<titulo>, <autor>, <editorial>, etc.).

Gracias a la flexibilidad de XML, la industria, las organizaciones o las empresas, pueden definir sus propias etiquetas para estructurar y compartir la información. Algunas de estas especificaciones han llegado a ser tan útiles y populares que incluso el W3C las ha adoptado como un estándar.

Una vez que dicha especificación XML llega a ser un estándar, los desarrolladores pueden usarla en sus aplicaciones y sistemas. A continuación se presenta una lista de lenguajes de marcado especializados.

- **RSS**. (**Really Simple Syndication)**. Permite generar y producir contenidos que se obtienen por suscripción. Se utiliza fundamentalmente para generar noticias. Es una de las aplicaciones XML más utilizada.
- **Atom**. Es un formato semejante al anterior pensado también para distribuir información desde una web.
- **ePUB**. Formato de libro digital que se ha convertido en un estándar de facto por la gran implantación que está teniendo en todos los dispositivos de lectura digitales. En realidad es un archivo comprimido (ZIP) que contiene tres documentos XML que son los que especifican la estructura y contenido del documento.
- **DITA**. **Darwin Information Typing Architecture**, se utiliza para producir documentos técnicos y está siendo cada vez más popular. Tiene capacidad para incluir numerosos metadatos y para poder adaptarse a las necesidades de las entidades que utilicen el lenguaje. Permite dividir en temas reutilizables los documentos.
- **MathML**. Pensado para representar documentos con expresiones matemáticas.
- **ODF.** Es un formato abierto que pretende ser un estándar como formato de intercambio entre documentos ofimáticos (hojas de cálculo, procesadores de texto, presentaciones, diagramas,…). Su popularidad aumentó notablemente cuando fue adoptado por el software **Open Office**.
- **OSDF**. **Open Software Description Format**, sirve para describir la tecnología y los componentes con los que se ha desarrollado un determinado software, a fin de facilitar el proceso de instalación.
- **OOXML. Office Open XML.** Formato rival de ODF, propiedad de Microsoft y utilizado como formato XML para el software **Microsoft Office**. Pretende también ser un estándar.
- **RDF**, **Resource Description Format,** Sirve para desarrollar documentos que describan recursos. Se trata de un proyecto ya antiguo para definir modelos de metadatos. Se basa en los modelos conceptuales como el modelo **entidad/relación** o los **diagramas de clases**, aunque actualmente se utiliza fundamentalmente para describir recursos web.
- **SMIL**. **Synchronized Multimedia Integration Language**, Lenguaje sincronizado de integración multimedia. Utilizado para producir fundamentalmente presentaciones de TV en la web.
- **SOAP**. **Simple Object Access Protocol**. Protocolo estándar de comunicación entre objetos software utilizado para comunicar con Servicios Web.
- **SVG**. **Scalable Vector Graphics,** gráficos de vectores escalables. Permite definir imágenes vectoriales pensadas para ser publicadas en una página web.
- **VoiceXML**. Se utiliza para representar diálogos vocales con los que se pueda interactuar.
- **WSDL**. **Web Services Description Language**. Lenguaje para definir Servicios Web.

Otros lenguajes de marcas basados en XML, menos conocidos, pero agrupados por sectores en donde se utilizan son:

Contabilidad	XFRML (Extensible Financial Reporting Markup Language), SMBXML (Small and Medium Sized Business XML)
Entretenimiento	SMDL (Standard Music Description Language), ChessGML (Chess Game Markup Language), BGML (Board Game Markup Language)
Relación con clientes	CIML (Customer Information Markup Language), NAML (Name/Address Markup Language), vCard
Educación	TML (Tutorial Markup Language), SCORM (Shareable Courseware Object Reference Model Initiative), LMML (Learning Material Markup Language)

Software	OSD (Open Software Description), PML (Pattern Markup Language), BRML (Business Rules Markup Language)
Fabricación	SML (Steel Markup Language)
Ordenadores	SML (Smart Card Markup Language), TDML (Timing Diagram Markup Language)
Energía	PetroXML, ProductionML, GeophysicsML
Multimedia	MML (Music Markup Language), X3D (Extensible 3D)
Matemáticas	OpenMath

La lista podría ser más extensa, pero con la expuesta anteriormente podemos hacernos una idea de la cantidad de lenguajes de marcado especializados que están normalizados para un uso de forma estándar.