1. Lenguajes de marcas

No hay que confundir los lenguajes de marcas con los lenguajes de programación. En un lenguaje de programación se puede trabajar con variables de diferentes tipos o hacer operaciones lógicas o matemáticas. En un lenguaje de marcas no. El lenguaje de marcas lo que hace es añadir información acerca de la estructura del texto o su presentación.

En general, se utilizan marcas con muchas finalidades. Por ejemplo, se marca un texto en negrita si se considera importante una definición en un libro; el estudiante, al tomar apuntes en clase, establece su propio código de marcas; se marca de alguna manera la presencia o ausencia de los alumnos en clase utilizando algún software específico o simplemente en un cuaderno de notas, etcétera.

> El lenguaje de marcas establece unas normas de marcado en aquellos textos que van a ser difundidos a través de la red o van a relacionarse entre ellos. Es decir, establece una codificación añadida al texto.

El siguiente ejemplo muestra un texto entre marcas con el objetivo de darle un significado concreto o una forma de representación:

```
<tema1>
    <fecha>22/05/2021</fecha>
    <titulo>Lenguajes de marcas</titulo>
    <contenidos>Los lenguajes de marcas son ...</contenidos>
    <profesor>Profesor DAM titular</profesor>
</tema1>
```

A. Tipos de lenguajes de marcas

En función de su propósito, los lenguajes de marcas se agrupan en lenguajes de marcas de presentación, de procedimiento, descriptivos o semánticos y ligeros (LML). En la siguiente tabla se resumen sus características principales y algunos ejemplos.

Tipo de lenguaje de marcas	Características	Ejemplos
De presentación	• Establecen el formato del texto, es decir, cómo lo verá quien lo vea o lea.	Ejemplo de este tipo es el utilizado desde cualquier procesador de texto, como LibreOffice, OnlyOffice o Word, que añade al texto marcas ocultas que determinan el aspecto que tendrá.
De procedimiento	• También está dedicado a la presentación del texto, pero en este caso se indica al software que lo muestra las acciones que debe realizar.	Ejemplos de este tipo son los lenguajes HTML o Hipertext Markup Language y LaTex.

	• El software que muestra el texto debe interpretar el código en el mismo orden en que aparece. • Cuando se edita el texto se muestran las marcas al usuario.	
Descriptivos o semánticos	• Utilizan marcas para describir los fragmentos de texto pero no determinan cómo será presentado el texto ni el orden en que aparecerá.	Ejemplo de este tipo son los lenguajes SGML (Standard Generalized Markup Language) y XML (Extensible Markup Language).
Ligeros (LML)	• Utilizan un tipo de formateo de texto bastante estandarizado, y su principal característica es que los archivos generados ocupan poco espacio y se editan con un editor de texto. • La reducción en el tamaño de los archivos permite optimizar el almacenamiento y supone un menor consumo de ancho de banda en su transmisión.	Ejemplos de lenguajes de marcas de este tipo son Markdown, ReStructured Text y Textile.

Tabla 1.1. *Tipos de lenguajes de marcas.*

En la sección Material de trabajo > Descarga de documentos está disponible el archivo UD1_Clasificación.pdf, que contiene otros criterios de clasificación de los lenguajes de marcas.

IMPORTANTE

El lenguaje Markdown se utiliza para documentar proyectos de software. Lo veréis con detalle en el módulo de Entornos de desarrollo.

B. Características de los lenguajes de marcas

Las características generales de los lenguajes de marcas son las siguientes:

- Los documentos, al ser en texto plano, son independientes del software utilizado, pudiendo hacer uso de cualquier editor de texto.
- Las marcas añadidas están insertadas en el propio contenido.
- Sus componentes son elementos sencillos e intuitivos, y permiten la creación de lenguajes específicos dependiendo del tipo de documento.
- Son muy versátiles y con un ámbito de utilización muy extenso: páginas web, libros, gráficos, fórmulas matemáticas, vídeos, sindicación de contenidos, etcétera.

2. Conceptos básicos

Veamos cada uno de estos componentes.

A. Etiquetas o marcas

> Las marcas son un conjunto de códigos que se añaden al texto para establecer su formato de visualización, la forma como se ha de imprimir o la estructura que tiene. Se añaden al texto, pero no forman parte de este.
>
> Cada lenguaje tiene su propia colección de marcas con su clasificación y significado.

Hay tres tipos de etiquetas:

1. **Apertura**: carácter de inicio '<' y carácter de finalización '>'. Ejemplo: .
2. **Cierre**: carácter de inicio '</' y carácter de finalización '>'. Ejemplo: .
3. **Vacía**: carácter de inicio '<' y carácter de finalización '/>'. Ejemplo: <linea/>.

Ejemplo:

```
<p>...contenido_párrafo...</p>
```

donde y son las marcas que delimitan el párrafo.

En la práctica se pueden combinar marcas pertenecientes a diferentes lenguajes de marcas. No siempre es requerida la marca de cierre o final, depende del lenguaje concreto.

Las marcas pueden ser de dos tipos: etiquetas o referencias a entidades.

En esta unidad, más adelante, se explican las referencias.

En un documento, todo lo que no son marcas es contenido. Las etiquetas se pueden escribir tanto en mayúsculas como en minúsculas, el navegador las interpreta igual. La etiqueta <HTML> se interpreta igual que <html>. En la actualidad se recomienda utilizar siempre minúsculas.

B. Elementos

> Un elemento representa una estructura lógica completa o un comportamiento concreto.

Por ejemplo, para HTML un elemento es un párrafo, una lista, una tabla, una imagen, etcétera.

La declaración de un elemento tiene una etiqueta inicial, un contenido y una etiqueta final.

Ejemplo:

```
<ul>
      <li><p>...contenido 1 de la lista...</p>
      <li><p>...contenido 2 de la lista...</p>
</ul>
```

La porción de código anterior representa una estructura de lista y, dentro de ella, dos estructuras de párrafo, cada estructura con su etiqueta inicial o de apertura y su etiqueta final o de cierre. Cada una de estas estructuras es un elemento.

Observar que la etiqueta `<li>` no tiene su correspondiente etiqueta de cierre. No es necesaria, y para `<p>` tampoco es necesaria. En estos casos se considera como cierre la aparición de otro elemento con su etiqueta, ya sea de apertura o de cierre.

Hay otros elementos que permiten omitir también la etiqueta inicial, es el caso de `<head>` y `<body>`. No hace falta escribirlas de forma explícita, porque siempre están. En general, y para HTML es su DTD (declaración de tipo de documento) el que determina si la etiqueta inicial y final son imprescindibles o no.

Un elemento es vacío si no tiene contenido. Por ejemplo, `<br>` es un elemento de salto de línea y no tiene contenido ni marca final, pero indica el comportamiento que se debe tener, es decir, cuando se encuentra este elemento se produce un salto de línea.

En HTML son elementos vacíos: `<area>`, `<base>`, `<br>`, `<col>`, `<embed>`, `<hr>`, `<img>`, `<input>`, `<link>`, `<meta>`, `<param>`, `<source>`, `<track>` y `<wbr>`.

No hay que confundir elemento con etiqueta. Los elementos son estructuras completas que contienen etiquetas o marcas.

C. Atributos

Los elementos pueden tener atributos, con unos determinados valores (entre comillas dobles " o entre comillas simples '), bien por defecto o bien asignados por el programador.

Los atributos deben ir incluidos dentro de '<' '>', y si se incluyen varias parejas atributo/valor deben ir separadas por un espacio en blanco. El orden en el que aparecen los atributos es indiferente.

Puede haber atributos en las etiquetas de apertura y en las vacías, pero no en las de cierre.

En el ejemplo siguiente, el elemento `<html>` indica el comienzo o la raíz del documento, y está siempre presente aunque no se incluya. En este caso se ha añadido el atributo `lang` (lengua) con el valor `es` (español).

```
<html lang="es">
```

En este otro ejemplo, el elemento `<meta>`, que es vacío, añade el atributo `charset`, que indica que el juego de caracteres a utilizar es utf-8.

```
<meta charset="utf-8">
```

Si se incluyen dos atributos quedaría:

```
<meta name="author" content="Nombre del autor" />
```

IMPORTANTE

Para ampliar información sobre los sistemas de codificación ASCII, UTF-8 y los atributos HTML ordenados alfabéticamente, consulta los Enlaces web en la sección Material de trabajo.

D. Metadatos

Los metadatos, en general, proporcionan información sobre los propios datos.

En una página HTML, los metadatos dan información relativa al contenido de la página como, por ejemplo, el tema, la descripción de la misma, la codificación utilizada, las palabras que describen la página, etcétera.

Los metadatos son utilizados por los buscadores para definir la información principal de la página web (temática, descripción), y es muy importante que estén bien definidos si se quiere que en la web los buscadores indexen correctamente la página.

El elemento HTML asociado es `meta`.

Ejemplo:

```
<meta name="author" content="Nombre del autor" />
```

El metadato `meta` contiene el atributo `name`, que indica que el atributo `content` será el nombre del autor. En la unidad dedicada a HTML se verá con más detalle este elemento.

3. Evolución de los lenguajes de marcas

A continuación se da una breve introducción cronológica de los lenguajes de marcas más importantes y una comparativa entre ellos.

A. SGML

El lenguaje GML fue mejorado y ampliado, y en 1986, siguiendo la norma ISO 8879, aparece SGML o *Standard Generalized Markup Language*, estándar de lenguaje de marcado generalizado, que, en realidad, no es un lenguaje de marcas, sino un metalenguaje. Esto quiere decir que, utilizando SGML, se pueden definir otros lenguajes de marcas específicos, pero todos ellos con el objetivo común de facilitar la gestión, la publicación y el intercambio de documentos digitales con independencia del hardware y el software utilizados.

> **SGML** es un estándar internacional, no propietario, y proporciona un sistema de codificación que permite la transferencia de documentación utilizando diferentes sistemas operativos, diferente software de edición, presentación y diferentes soportes hardware para su almacenamiento e impresión sin pérdida de información.

Pero un documento SGML no incorpora nada que indique cómo va a verse en la pantalla o impreso. Para ello habrá que adjuntar al documento todo lo relativo al estilo con el que se verá.

SGML es un marco general muy abierto; de hecho, no obliga a utilizar ningún conjunto de etiquetas concretas, y eso le añade complejidad. Por ello ha quedado en desuso a nivel general muy rápidamente, pero debido a su potencia y flexibilidad ha servido de base para la creación de los lenguajes de marcas más utilizados en este momento, como son HTML, XML y HTML5.

Como resumen, las tres propiedades de SGML son:

1. Utilización de codificación descriptiva.
2. Introducción del concepto «tipo de documento».
3. Independencia del sistema utilizado para representar el documento.

Los objetos SGML pueden ser:

- **Elementos**: insertados como etiquetas y sus atributos locales. La etiqueta `<!ELEMENT` se utiliza para la declaración de un elemento.
- **Referencias a entidades**: en general son unidades de contenido. En concreto, la norma ISO 8879 define entidad como «un conjunto de caracteres que pueden ser referidos como una unidad». La etiqueta `<!ENTITY` se utiliza para la declaración de una entidad.

Un documento SGML consta de tres partes o componentes (archivos):

1. **Declaración SGML.**
2. **Declaración de tipo de documento DTD.**
3. **Instancia del documento.**

Veamos cada uno de estos componentes.

El componente **declaración SGML** indica una serie de características del documento; por ejemplo, qué conjunto de caracteres se utilizan, los delimitadores, la sintaxis del documento y, en general, la normativa aplicada a él.

Este marco formal determinado por la declaración SGML va a delimitar el alcance de **la DTD**, es decir,

la DTD debe respetar la normalización impuesta por la declaración SGML. Y si no lo hace se desencadenará una cascada de errores en el documento.

Es obligatoria su existencia, y si no aparece se toma la sintaxis por defecto, como son el conjunto de caracteres y otras características.

Un ejemplo de declaración SGML sería la siguiente:

```
<!DOCTYPE book PUBLIC "-//OASIS//DTD Docbook V3.1//EN" []>
```

Observar cómo se indica en la declaración el tipo de documento. En este caso es `book`, y en ella ya se está indicando qué DTD se debe utilizar, que es `DTD Docbook`.

Por lo tanto, con el tipo de documento se está indicando la gramática que se va a utilizar y cómo se debe utilizar. SGML no está obligando a utilizar ningún tipo de marcado específico, y es la DTD asociada al tipo de documento la que establece la gramática.

El componente declaración de tipo de documento DTD e **instancia del documento** contiene la estructura del documento, es decir, todo lo necesario para crearlo. Se trata de las también llamadas «declaraciones de documentos», que sirven para poner nombre a las marcas y especificar el contenido de cada elemento.

Lo más usual es que la DTD se almacene en un archivo aparte con extensión .dtd y desde el documento SGML se referencie. En cualquier caso, es imprescindible que exista la DTD; de lo contrario, la herramienta que comprueba si el documento SGML está bien conformado, el ***parser***, daría error.

> El ***parser*** es el programa que se encarga de comprobar que se cumplen todas las reglas. Interpreta las etiquetas, resuelve las referencias a otros archivos incluyéndolos, ejecuta las instrucciones de procesamiento, etcétera.

Por ejemplo, un documento sin título no será validado por el *parser*, ya que no cumple las especificaciones de la DTD que obliga a que siempre exista.

Ejemplo de DTD para el tipo de documento DocBook:

```
<!DOCTYPE book PUBLIC "-//OASIS//DTD DocBook V4.1//EN">
   <book lang="es">
            <bookinfo>
              <date>Abril 2021</date>
              <title>Lenguaje de marcas</title>
              <author>
                <firstname>Nombre</firstname>
                <surname>Apellido1 Apellido2</surname>
              </author>
            </bookinfo>
            <chapter id="capitulo1">
              <title>Evolución de los lenguajes de marcas</title>
              <para>
                Texto...
              </para>
              <sect1 id="seccion1">
                <title>SGML</title>
                <para>Introducción</para>
              </sect1>
            </chapter>
   </book>
```

Si el libro completo no estuviera en un solo archivo sino en varios habría que añadir la marca `ENTITY` de la siguiente forma, y en la que indicamos el nombre del archivo .sgml a incluir.

```
<!DOCTYPE book PUBLIC "-//OASIS//DTD DocBook V4.1//EN"[
    <!ENTITY capitulo2.sgml SYSTEM "capitulo2.sgml"> ]>
```

La instancia o muestra del documento es el propio documento y contiene el marcado y el texto con todos los objetos SGML necesarios siguiendo la estructura jerárquica establecida por la DTD.

Para escribir el código se pueden utilizar herramientas clásicas como Vim, Emacs, Bloc de notas, Notepad++, Gedit o también Visual Studio Code (Microsoft), que dispone de versiones para Windows, GNU/Linux y MacOS.

IMPORTANTE

Aconsejamos utilizar Visual Studio Code. En los Enlaces web está la URL de descarga, y hay información sobre la instalación y el uso básico de la herramienta.

El esquema siguiente muestra la relación entre SGML y los lenguajes de marcas que tienen su origen en él, como son HTML y XML.

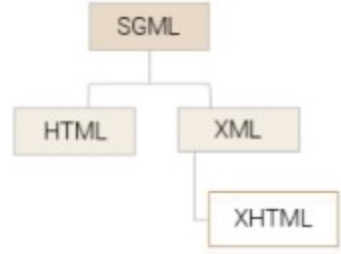


Fig. 1.1. Jerarquía de lenguajes de marcas.

B. HTML

HTML o *HiperText Markup Language*, lenguaje de marcas hipertextual, es un lenguaje de marcas que sigue el estándar SGML.

La *World Wide Web* (WWW) fue un proyecto desarrollado por Tim Berners-Lee a principios de los años noventa. El objetivo de dicho proyecto era enlazar grandes volúmenes de información ubicados en sistemas diferentes para que, en entornos científicos, se pudieran compartir datos, noticias y documentación. Para ello creó un nuevo lenguaje de marcas, HTML, que permitía la descripción de estos documentos y se basó en dos estándares que ya existían:

- **ASCII,** que representa en binario cada carácter del alfabeto latino, más una serie de caracteres de control y símbolos. Este formato de representación es entendido por los procesadores y editores de texto y, por lo tanto, permite la transferencia de información entre ordenadores.
- **SGML,** que proporciona estructura al documento, tal y como se ha explicado.

Pero HTML es mucho más sencillo que SGML, ya que solo incorpora al documento los elementos imprescindibles; eso hizo que su uso se expandiera y pasó a ser un estándar en la confección de páginas web. Pero no tiene la potencia del SGML, y además presenta algunos inconvenientes:

- Un conjunto de etiquetas limitado.
- El documento contiene tanto la estructura como el diseño.
- No permite mostrar contenido dinámico. En las últimas versiones comienza a incorporarlo.
- El mantenimiento de las páginas es costoso.
- Es muy sencillo para la presentación de información, pero no para su procesado o almacenamiento.

En la unidad 2 se estudia con detalle este lenguaje de marcas HTML con todos sus elementos.

C. XML

XML, o *eXtensible Markup Language*, lenguaje de marcas extensible, aparece en 1996 y sigue el estándar SGML, y al igual que él no es realmente un lenguaje de marcas sino un metalenguaje, es decir, permite la creación a partir de él de otros lenguajes de marcas que se adaptan a los usos.

Tiene ventajas o potencia similares a SGML, pero es menos complejo. En realidad, XML contiene un subconjunto de la funcionalidad de SGML.

Es de tipo descriptivo o semántico, y su objetivo es estructurar la información en documentos utilizando etiquetas propias y una gramática que determina la jerarquía mediante la DTD. Así, XML se puede utilizar para validar documentos que contienen fórmulas matemáticas (MathML), hojas de estilo (XSL), descripción de recursos (RDF), etc.

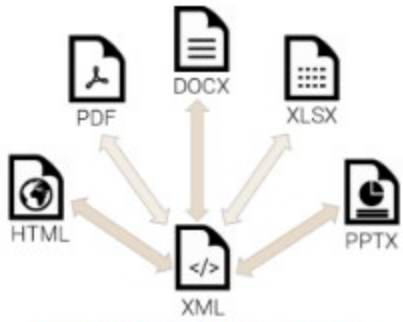


Fig. 1.2. Opciones de exportación de XML.

La figura 1.2 muestra diferentes opciones que ofrece XML para exportar documentos a otros formatos.

Las principales características de XML son:

1. Permite la creación de etiquetas propias y asignarles atributos.
2. Al incluir los metadatos en el propio documento, los motores de búsqueda de la web pueden indexarlos mejor y las búsquedas serán más efectivas.
3. La estructura del documento y su diseño están separados.
4. Permite exportar los documentos a otros formatos, como HTML, PDF y RTF, entre otros.
5. Permite compartir documentos entre diferentes sistemas operativos y plataformas, como por ejemplo páginas web y bases de datos.
6. Permite la internacionalización, ya que puede utilizar diferentes juegos de caracteres.
7. XML es un estándar abierto, es decir, se permite su uso libre sin problemas de licencias.

En el sitio web del W3C están disponibles las recomendaciones de XML publicadas por el W3C, *World Wide Web Consortium*.

En la sección Material de trabajo > Descarga de documentos está disponible el archivo UD1_Organizaciones_Internacionales.pdf, con información sobre las organizaciones internacionales desarrolladoras, entre ellas el W3C.

D. Comparativa: XML y HTML

La tabla siguiente muestra las principales características de XML y HTML de forma comparativa:

XML	HTML
Es un metalenguaje.	Es un lenguaje de marcas.

Subconjunto funcional de SGML.	Derivado de SGML.
No es un lenguaje de presentación.	Es un lenguaje de presentación.
Describe información.	Da formato y muestra información.
XML es dinámico y se usa para transferir datos.	HTML es estático, se usa para mostrar datos.
Permite la definición de nuevas marcas, es decir, puede extenderse.	Conjunto limitado de marcas predefinidas en su estándar.
Obliga a cerrar las etiquetas.	Las etiquetas pueden funcionar bien sin necesidad de cerrarlas.
XML imprime los espacios en blanco.	HTML no conserva los espacios en blanco.
Utiliza el navegador como plataforma para el desarrollo de aplicaciones.	Utiliza el navegador solo para mostrar información.
Obliga a ser ordenados.	Se puede ser desordenado.
Proporciona interoperabilidad con SGML y HTML.	Facilita la transferencia de información basada en la web.
Objetivo principal: almacenar y transferir datos.	Objetivo principal: presentar datos.

Tabla 1.1. Comparativa entre XML y HTML.

E. Comparativa: XML y SGML

Como se ha mencionado anteriormente, XML sigue el estándar SGML y, al igual que él, no es realmente un lenguaje de marcas sino un metalenguaje, es decir, permite la creación a partir de él de otros lenguajes de marcas que se adaptan a los usos que se vayan a hacer.

En la siguiente tabla se resumen las principales diferencias entre ambos lenguajes:

XML	SGML
Subconjunto de la funcionalidad de SGML pero más sencillo.	Muy complejo.
Introduce el concepto de documento "bien formado" sin requerir validación.	Requiere la validación del documento.
Muy utilizado en diferentes áreas.	Uso muy restringido por su complejidad.
Compatible con HTML.	No hay compatibilidad específica con HTML.
Formateo y estilo relativamente fáciles de aplicar.	Formateo y estilo relativamente difíciles de aplicar.

Tabla 1.2 Comparativa entre XML y SGML.

Reseñar que a lo largo de las unidades del módulo Lenguajes de Marcas se podrán utilizar indistintamente los términos «marca» y «etiqueta», teniendo ambos el mismo significado.

4. Iniciación a XML

Comenzamos describiendo el origen del lenguaje XML.

A. Origen del XML

En 1989, Tim Berners-Lee utilizó como base SGML para crear HTML, con el objetivo de ser utilizado en el laboratorio europeo CERN, pero rápidamente se extendió su uso para el intercambio de información a nivel mundial. El mismo Berners-Lee, en 1994, creó el organismo internacional W3C para hacer un seguimiento, una normalización y una ampliación del nuevo lenguaje HTML.

Pero llegó un momento en que su uso obligaba a una ampliación de aplicaciones tan dispares que el W3C decidió establecer unas reglas más generales para que se pudieran crear diferentes lenguajes de marcas en función de las necesidades específicas de los usuarios. Y esas reglas deberían permitir la compatibilidad con HTML a nivel de estructuras, sintaxis, etc. Y así fue como apareció XML en 1998, como una plataforma base para, sobre ella, generar diferentes lenguajes de marcas.

El W3C continuó el desarrollo de XML generando o incorporando todas aquellas tecnologías necesarias para procesar los documentos XML y que pudieran ser utilizadas desde cualquier lenguaje de marcas derivado de XML.

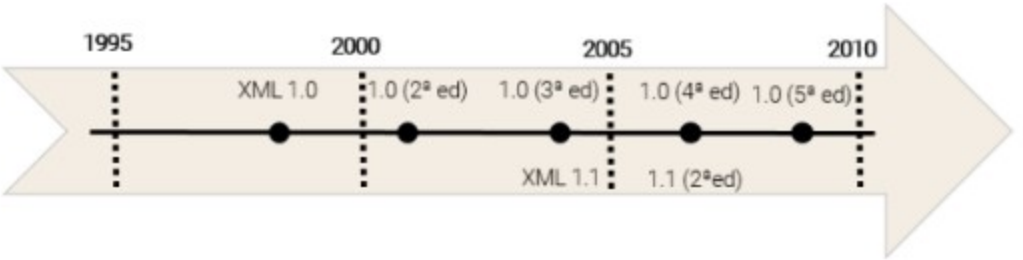


Fig. 1.1. Línea de tiempos para XML.

En el año 2000, y dado que HTML no seguía el estándar XML, el W3C creó el XHTML, que era un HTML acorde a XML. Entonces el desarrollo de HTML terminó.

Otra iniciativa del W3C fue impulsar la llamada web semántica. La semántica es la parte de la lingüística que estudia cómo los seres humanos dan sentido a las expresiones lingüísticas a través de las estructuras léxicas y los procesos mentales (Ver Enlaces web en la sección Material de Trabajo al final de la unidad).

Su objetivo es que la búsqueda de información por parte de los usuarios sea más sencilla e intuitiva, utilizando como base XML. Esta iniciativa del W3C no tuvo éxito, ya que la web no adoptó ninguna de las recomendaciones de la web semántica.

El recorrido del nuevo lenguaje XHTML fue muy corto por diferentes motivos, y en 2014 el W3C retoma el desarrollo de HTML creando las recomendaciones para HTML5, una que está basada en XML y se llama XHTML5, y otra no basada en XML llamada HTML5.

Por último, decir que en 2018 el W3C abandonó el desarrollo de XML, ya que cerró todos los grupos dedicados a ello. Como principal motivo de este cierre está el hecho de que XML es altamente exigente con el tema de los errores, es un lenguaje muy estricto en ese sentido.

De cualquier forma, XML sigue siendo un lenguaje muy utilizado en todo lo relativo al almacenamiento y la transmisión de la información, y es muy posible que el W3C lo retome en un futuro.

B. Características del metalenguaje XML

XML es un metalenguaje de tipo descriptivo o semántico, y su objetivo es estructurar la información en documentos utilizando etiquetas propias y una gramática que determina la jerarquía mediante la DTD o *Document Type Definition*.

Su potencia radica en su interoperabilidad entre diferentes sistemas operativos y aplicaciones, además de ser un estándar para el almacenamiento de información, creado y optimizado para ser utilizado en la web.

Un documento XML se guarda en un archivo .xml y es un archivo de texto cuyo contenido son parejas de etiquetas con estructura de árbol que encierran la información propiamente. Por lo tanto, este archivo .xml puede ser editado con cualquier editor de texto incluso en la nube utilizando, por ejemplo, XML Viewer o XMLGrid, que además son gratuitos.

Hay que tener en cuenta que, aunque el archivo .xml es un archivo de texto, no es exactamente un archivo .txt. Para la conversión de un archivo XML a otros formatos, y si la propia herramienta con la que se generó no lo permite, hay que hacerlo utilizando algún tipo de software específico o servicio en línea.

En Material de trabajo > Descarga de documentos tienes disponibles enlaces a diferentes conversores en el archivo UD1_Conversores_XML.pdf.

Además de las características básicas de XML ya enumeradas, ahora se van a completar con otras pero enfocadas a la web:

- Un documento XML tiene que estar «bien formado», es decir, tiene que cumplir unas reglas, y si no las cumple el *parser* lo rechazará.
- Un documento XML, además de las reglas del lenguaje, debe cumplir las reglas del esquema. El esquema define qué elementos se pueden utilizar, cómo se pueden anidar y tipos de atributos que se pueden incluir. Solo así será un documento válido.
- Al ser un lenguaje extensible se puede adaptar a diferentes requerimientos. Por lo tanto, su lista de etiquetas no es ni fija ni cerrada.
- Su objetivo es la descripción semántica del texto, no la presentación del mismo.
- Al incluir los metadatos en el propio documento, los motores de búsqueda de la web pueden indexarlos mejor y las búsquedas serán más efectivas.

C. Marcado del documento

El marcado de un documento se hace para estructurar su contenido.

Para colocar las marcas hay que tomar una parte del texto que tenga un significado concreto. La marca asociada se encierra entre los caracteres "<" y ">", que siempre se interpretan como código.

Una marca es, simplemente, una secuencia lineal de caracteres con una estructura concreta.

EJEMPLO 1

Revisión de etiquetas

Volvemos al código XML del caso práctico 3 y se propone revisar las diferentes etiquetas y la estructura del código.

```
<?xml version="1.0" encoding="utf-8"?>
<!DOCTYPE libro>
<libro>
    <titulo>Lenguaje de marcas: XML</titulo>
    <autor>Nombre autor</autor>
    <editorial>McGraw-Hill</editorial>
    <isbn>978-2-7360-4965-1</isbn>
    <depositolegal>M-8672-2021</depositolegal>
    <paginas>270</paginas>
</libro>
```

Solución

La línea de código `<titulo>Lenguaje de marcas: XML</titulo>` incorpora dos etiquetas, una de apertura (< >) y otra de cierre (</ >), igual que en HTML. Son etiquetas que se incluyen para delimitar una porción de texto del documento que identifica el título. Como alternativa se puede utilizar también `<` para representar '<' y `>` para '>'.
En el ejemplo se observa que el documento XML tiene un nodo raíz y de él cuelgan todos los elementos, pudiendo presentar cuantas anidaciones se requieran de estos elementos y creando una estructura en árbol. En el ejemplo solo existe un anidamiento, siendo el nodo raíz el elemento `<libro>`.

Elementos, atributos y valores

El elemento es la unidad de información semántica y consta de una etiqueta o marca de apertura '<' y otra de cierre '>'.

Un elemento consta de:

1. El nombre del elemento o identificador.
2. Uno o varios atributos con sus valores respectivos.

Los atributos permiten añadir propiedades a los elementos del documento. Pero no añaden nada a la estructura lógica del documento XML y no pueden contener otros elementos y/o atributos.

```
<elemento atributo="valor">contenido</elemento>
```

que, en el ejemplo concreto, añadiendo el atributo `'id'`, será:

```
<titulo id="XML">Lenguaje de marcas: XML</titulo>
```

El nombre del atributo debe ir dentro de la etiqueta de apertura, separado por un espacio del nombre del elemento y seguido de '='. El valor asignado irá entre comillas dobles.

El orden de los atributos dentro de la etiqueta es indiferente, pero no se deben repetir.

Cuando un elemento no tiene contenido se llama elemento vacío y se representa <elemento_vacio/>.

A continuación, y como resumen, se indican unas reglas mínimas para la formación de elementos. De esta forma, el procesador XML no dará error.

Respecto al nombre del elemento:

1. No puede empezar por un dígito, signo o la cadena 'xml'.
2. Solo puede contener letras y dígitos, pero no acentos o espacios.
3. Puede contener algunos caracteres como el guion '-', el guion bajo '_' o el punto '.'.
4. Se diferencia entre mayúsculas y minúsculas.

Respecto a la estructura:

1. Un documento XML debe tener un único elemento raíz.
2. Se permite la anidación de elementos siempre que estén correctamente abiertos y cerrados. Es decir, no se puede cerrar un elemento si tiene anidado otro elemento abierto.
3. El texto contenido en un elemento no puede contener los caracteres '<', '>', '&', '', ''.

La tabla siguiente muestra las cadenas que sustituyen a estos caracteres en el caso de que sea necesaria su inclusión en el texto.

Carácter	Cadena	Carácter	Cadena	Carácter	Cadena
>	>	&	&	'	'
<	<	"	"		

Tabla 1.1. Cadenas que sustituyen a caracteres.

EJEMPLO 2

Identificar atributos

El siguiente código muestra un documento XML en el que algunos de sus elementos contienen atributos. Identificar cuáles son esos elementos. ¿Qué se verá en el navegador?

```
<?xml version="1.0" encoding="utf-8" ?>
<!DOCTYPE biblioteca >
<biblioteca>
      <ejemplar tipo_ejem="libro" titulo="Lenguaje de marcas: XML"
editorial="McGraw-Hill">
      <tipo> <libro isbn="978-2-7360-4965-1" edicion="1"
paginas="270"></libro> </tipo>
      <autor nombre="Nombre autor"></autor>
      <autor nombre="Nombre Apellido1 Apellido2" funcion="traductor"></autor>
      <prestado lector="Fulanito">
      <fecha_pres dia="1" mes="ene" año="2021"></fecha_pres>
      <fecha_devol dia="31" mes="ene" año="2021"></fecha_devol>
      </prestado>
      </ejemplar>
</biblioteca>
```

Solución

Crear un archivo .xml con el código dado y comprobar que en el navegador lo que se verá es el propio código XML. Vemos que los elementos aparecen coloreados en ciruela, los nombres de los atributos en rojo y sus valores en azul.

Como se observa en el código, los atributos se definen y dan valor dentro de una etiqueta de inicio o de elemento vacío, a continuación del nombre del elemento o de la definición de otro atributo y 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, '<'.

IMPORTANTE

Hay que tener en cuenta que XML diferencia entre mayúsculas y minúsculas en los identificadores y los atributos.
Se pueden (y deben) añadir comentarios a los documentos XML. El intérprete XML no los tiene en cuenta pero añaden información que facilitará la posterior lectura y comprensión de dicho documento. Estos se incluyen entre las cadenas "<!--" y "-->".

D. Partes de un documento XML

Un documento XML contiene las siguientes partes:

1. Prólogo

No es obligatorio, pero si se incluye debe ir al comienzo del documento. Consta de la declaración XML y la declaración de tipo de documento.

Declaración XML (instrucciones de procesamiento).

Un documento XML comienza con la declaración XML:

```
<?xml version="1.0" encoding="utf-8" ?>
```

Antes del elemento declaración XML no se puede añadir otro elemento, comentarios o espacios en blanco.

Al ser opcional, los documentos SGML y HTML pueden ser procesados como si fueran XML.

Las funciones de la declaración XML son tres:

1. **Declara que es un documento XML y la versión de XML utilizada.** Para ello se incluye:

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

   donde el carácter '<' seguido del interrogante de cierre (?) se interpreta como código y le indica que es una declaración o una instrucción de procesamiento, no de un elemento normal.

   Es importante conocer la versión del lenguaje utilizada por si, en el futuro, con nuevas versiones, hubiese algún tipo de incompatibilidad.

2. **Declara el tipo de codificación de caracteres utilizada.**

   ```
   <?xml version="1.0" encoding="iso-8859-1" ?>
   ```

   que, en el ejemplo, es ISO-8859-1, que permite el uso de acentos o la 'ñ' española, aunque por defecto utiliza UTF-8 (conjunto de caracteres universal).

   También se puede incluir la hoja de estilo utilizada para la visualización del documento. Por ejemplo:

   ```
   <?xml version="1.0" encoding="utf-8" ?>
   <?xml-stylesheet type="text/xsl" href="template.xsl"?>
   ```

3. **Declara la autonomía del documento.** Utilizando el atributo `standalone` se indica si el documento tiene (no) o no tiene (yes) dependencia de un esquema externo. Si no depende

(`standalone="no"`), el documento sigue las reglas de XML. Si depende (`standalone="yes"`), el documento sigue un DTD externo.

Se incluye en la declaración de la siguiente forma:

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

Declaración de tipo de documento

Con la declaración de tipo de documento se está indicando qué tipo de documento se está creando. Siempre comienza por la cadena:

```
<!DOCTYPE Nombre_tipo ...>
```

Si la definición de tipo de documento DTD se incluye en el propio documento, la sintaxis es:

```
<!DOCTYPE Nombre_tipo [… declaraciones …] >
```

Ejemplo:

```
<!DOCTYPE  libro  SYSTEM "http://localhost/libro.dtd">
```

En otra unidad se volverá sobre el DTD ampliando y completando todos los detalles relativos a dicho elemento.

2. Cuerpo del documento o ejemplar

El cuerpo del documento contiene los datos o el texto del documento XML. Su estructura es una serie de elementos anidados que son los que contienen los bloques de texto. Como ya se ha dicho, cada elemento lleva una marca de apertura y una de cierre del elemento, y cada elemento puede llevar atributos.

En un documento XML no puede haber ningún texto, dato o información que no esté contenido en un elemento.

En general, se busca que los nombres asignados a las marcas o etiquetas sean significativos, es decir, que tengan una relación con el contenido descrito.

Por último, y como resumen, se incluyen las reglas básicas que debe seguir la estructura de un documento XML.

En un documento XML:

1. Los elementos que lo componen deben tener estructura de árbol, es decir, jerárquica.
2. Solo puede haber un elemento raíz.
3. Los elementos deben estar correctamente anidados.
4. Los elementos no se deben superponer entre ellos.

Fig. 1.2. Estructura jerárquica de un libro.

El esquema de la figura 1.4 muestra un ejemplo de dicha estructura jerárquica para el caso de un libro con varios autores, un precio y varios descriptores:

Es muy importante conocer la estructura jerárquica de la información antes de la creación del documento XML. Ayudará a poder manejarla más tarde.

E. Documento XML bien formado

Se dice que un documento XML está bien formado cuando cumple las siguientes reglas o normas sintácticas definidas en la recomendación del W3C:

- Debe tener definido un prólogo con la declaración XML completa.
- Solo debe existir un único elemento raíz, que anide al resto de elementos del documento.
- Las instrucciones de procesamiento son opcionales.
- Un elemento debe comenzar y terminar con la misma etiqueta.
- Los atributos de los elementos son opcionales.
- Los valores de los atributos deben ir entre comillas dobles (") o comillas simples (').
- Se diferencia entre mayúsculas y minúsculas. En XML no es lo mismo <etiqueta> que <ETIQUETA>.

IMPORTANTE

No hay que confundir el concepto de «documento bien formado» con el de «documento válido». Un documento decimos que está bien formado cuando cumple las reglas antes descritas, y un documento válido, además de estar bien formado, tiene que cumplir las reglas que establece su DTD o esquema.

El concepto de esquema se estudia en otra unidad, en la que se verán todos los mecanismos de validación.

Creación de un recetario de cocina

ENUNCIADO

Vamos a crear un documento válido en XML que permita estructurar la información de varias recetas de concina de la página web de un restaurante.

Para cada receta habrá que indicar: su nombre, el tipo de receta (entrantes, sopas, ensaladas, carnes, pescados, postres), la dificultad, el tiempo total, el número de raciones, las calorías por ración, los ingredientes, y la descripción del procedimiento.

A continuación, se incluye una receta del tipo «postre» como ejemplo:

Bizcocho de calabaza (8 raciones)

Ingredientes:

- 250 gr calabaza asada limpia.
- 200 gr panela.
- 3 huevos.
- 70 gr aceite de girasol.
- 250 gr harina repostería.
- 3 sobres dobles de gaseosas.

Preparación

1. Forrar un molde rectangular de 18 x 30 cm aprox.
2. Batir la calabaza asada con la panela.
3. Añadir los huevos y seguir batiendo hasta que la panela esté disuelta.
4. Añadir el aceite y seguir batiendo hasta que quede mezclado.
5. Añadir la harina y los sobres blancos. Remover con cuidado y sin que queden grumos.
6. Añadir los sobres azules. Remover hasta que queden mezclados.
7. Verter la mezcla en el molde ya preparado. Espolvorear la superficie con azúcar y canela.
8. Hornear a 175ºC durante 25 minutos con horno precalentado.

Tiempo de resolución: 1 h aprox.

CLAVES DE RESOLUCIÓN

Recuerda que el documento siempre debe comenzar por el prólogo, que consta de:

- La declaración XML:

```
<?xml version="1.0" encoding="utf-8" standalone="yes"?>
```

- La declaración del tipo de documento:

```
<!DOCTYPE recetario>
```

La primera de ellas es imprescindible.

A continuación, ya se puede comenzar a escribir las diferentes etiquetas que van a identificar los elementos de que se compone el archivo. En nuestro caso ya sabemos que la raíz es 'recetario' y luego vendrán las diferentes recetas que, en nuestro caso, será solo una.

SOLUCIÓN

Utilizando Visual Studio Code, introducir el código XML siguiente.

```
<?xml version="1.0" encoding="utf-8" standalone="yes"?>
<!DOCTYPE recetario>
<recetario>
    <receta>
        <nombre >Bizcocho de calabaza</nombre >
        <tipo>Postre</tipo>
        <raciones>8</raciones >
        <dificultad>fácil</dificultad>
        <tiempo>35 minutos</tiempo>
        <calorias>250</calorias>
        <ingredientes>
            <ingrediente cantidad="250" unidad="gramos" nombre="calabaza asada limpia"/>
            <ingrediente cantidad="200" unidad="gramos" nombre="panela" />
            <ingrediente cantidad="3" unidad="unidades" nombre="huevos"/>
            <ingrediente cantidad="70" unidad="gramos" nombre="aceite de girasol"/>
            <ingrediente cantidad="250" unidad="gramos" nombre="harina reposteria"/>
            <ingrediente cantidad="3" unidad="sobres dobles" nombre="gaseosas"/>
            </ingredientes>
        <preparación>
            <paso>0. Forrar un molde rectángular de 18*30cm aprox.</paso>
            <paso>1. Batir la calabaza asada con la panela.</paso>
            <paso>2. Añadir los huevos y seguir batiendo hasta que la panela este
disuelta.</paso>
            <paso>3. Añadir el aceite y seguir batiendo hasta que quede mezclado.</paso>
            <paso>4. Añadir la harina y los sobres blancos. Remover con cuidado y sin que
queden grumos.</paso>
            <paso>5. Añadir los sobres azules. Remover hasta que queden mezclados.</paso>
            <paso>6. Verter la mezcla en el molde ya preparado. Espolvorear la superficie
con azúcar y canela.</paso>
            <paso>7. Hornear a 175ºC durante 25' con horno precalentado.</paso>
        </preparacion>
    </receta>
</recetario>
```

De la misma forma, se podrían introducir más recetas de diferentes tipos, para elaborar un menú.

Evolución de la informática

ENUNCIADO

Es un ejercicio de obtención de un documento XML a partir de datos. Crea un documento válido en XML que permita organizar diferentes hechos históricos relacionados con la informática.

La tabla siguiente los recoge de forma resumida y el documento XML generado debe reflejar todos los hechos, junto con la información proporcionada: año, mes, día, el hecho en sí mismo, su autor y su descripción.

Evolución de la informática resumida

Año	Mes	Día	Hecho	Autor	Descripción
3500 A.C.			Ábaco		Es el dispositivo más antiguo utilizado para realizar operaciones aritméticas.
1646	Marzo	8	Calculadora universal	Leibnitz	Máquina que suma, resta, multiplica, divide y hace raíces cuadradas.
1792	Marzo	14	Máquina analítica	Charles Babbage y Ada Lovelace	Máquina que resolvía funciones y obtenía tablas.
1852	Abril	12	Varias máquinas	Torres Quevedo	Construyó una máquina para jugar al ajedrez y una calculadora.
1936	Septiembre	15	Máquina de Turing	Alan Turing	Máquina capaz de resolver todo tipo de problemas con la solución algorítmica. Con Turing se inició la teoría matemática de la computación.
1943			Ordenador digital electrónico	Maurice V. Wilkes	El primer ordenador del mundo fue el EDSAC, desarrollado en la Universidad de Cambridge.
1958			Segunda generacion de ordenadores	IBM	Reemplaza las válvulas de vacío por los transistores. Son más pequeños y consumen menos electricidad.
1963			Código ASCII	ANSI	Nace a partir de reordenar y expandir el conjunto de símbolos y caracteres utilizados en aquel momento en telegrafía por la compañía Bell.
1964			Chips	Jack S. Kilby y Robert Noyce	Estructura de pequeñas dimensiones de material semiconductor, sobre la que se fabrican circuitos electrónicos.
1969			ARPANET	MIT (Massa-chusetts Institute of Technology)	Red de computadoras construida en 1969 como un medio para enviar datos militares y conectar grupos de investigación, a través de los Estados Unidos..
1981			MS-DOS	Microsoft	Es un sistema operativo de disco para el IBM PC y sistemas compatibles. Este sistema operativo

					dominó el mercado de los ordenadores personales entre 1985 y 1995.
1990			World Wide Web	Tim Berners Lee	Es un sistema de distribución de documentos de hipertexto o hipermedios interconectados y accesibles vía Internet.
1991			Núcleo Linux	Linus Torvalds	Linux es el núcleo de un sistema operativo libre, basado en Unix. Linux está licenciado bajo la GPL v2 y está desarrollado por colaboradores de todo el mundo.
1997	Mayo	7	Pentium 2	INTEL	El Pentium II es un microprocesador con arquitectura x86 diseñado por Intel.
2019	Enero		Ordenadores cuánticos	IBM	Se basan en los principios de la mecánica cuántica. En lugar de utilizar bits utilizan qubits, que pueden tener los dos estados de procesamiento al mismo tiempo, o ninguno. Esta tecnología mejora mucho la velocidad de procesamiento.

CLAVES DE RESOLUCIÓN

Hay que entrar en Visual Studio Code, crear el esqueleto del archivo XML asignando el nombre `evolucion` a la raíz y creando un bloque con la etiqueta `hecho` para cada evento histórico de la tabla.

Se deben contemplar todos los campos, incluso aunque estén algunos vacíos. Una vez creado el bloque, seleccionar, copiar y pegar e ir sustituyendo los datos por los correspondientes a cada entrada de la tabla. Lo único que se requiere es un poco de paciencia. La tabla es larga pero es importante reflejar todos los eventos porque más adelante se utilizará este archivo XML para realizar otro tipo de tareas.

SOLUCIÓN

```
<?xml version="1.0" encoding="utf-8"?>
<evolucion>
  <hecho nombre="Abaco" descripcion="Es el dispositivo más antiguo utilizado para
realizar operaciones aritméticas.">
    <autor></autor>
    <fecha>
     <dia></dia>
     <mes></mes>
     <anyo>3500AC</anyo>
   </fecha>
  </hecho>
  <hecho nombre="Calculadora universal" descripcion="Máquina que suma, resta,
multiplica, divide y hace raices cuadradas.">
   <autor>Leibnitz</autor>
   <fecha>
    <dia>8</dia>
    <mes>Marzo</mes>
    <anyo>1646</anyo>
  </fecha>
 </hecho>
```

```
    <hecho nombre="Máquina analítica" descripcion="Máquina que resolvía funciones y obtenía
tablas.">
    <autor>Charles Babbage y Ada Lovelace</autor>
    <fecha>
      <dia>14</dia>
      <mes>Marzo</mes>
      <anyo>1792</anyo>
    </fecha>
  </hecho>
    <hecho nombre="Varias máquinas" descripcion="Construyó una máquina para jugar al
 ajedrez y una calculadora.">
    <autor>Torres Quevedo</autor>
    <fecha>
      <dia>12</dia>
      <mes>Abril</mes>
      <anyo>1852</anyo>
    </fecha>
  </hecho>
   <hecho nombre="Máquina de Turing" descripcion="Máquina capaz de resolver todo tipo de
 problemas con la solución algorítmica. Con Turing se inició la teoría matemática de la
 computación.">
    <autor>Alan Turing</autor>
    <fecha>
      <dia>15</dia>
      <mes>Septiembre</mes>
      <anyo>1936</anyo>
    </fecha>
  </hecho>
   <hecho nombre="Ordenador digital electrónico" descripcion="El primer ordenador del
 mundo fue el EDSAC, desarrollado en la Universidad de Cambridge.">
    <autor>Maurice V. Wilkes</autor>
    <fecha>
      <dia></dia>
      <mes></mes>
      <anyo>1943</anyo>
    </fecha>
  </hecho>
 <hecho nombre="Segunda generación de ordenadores" descripcion="Reemplaza las válvulas de
 vacío por los transistores. Son más pequeños y consumen menos electricidad.">
    <autor>IBM</autor>
    <fecha>
      <dia></dia>
      <mes></mes>
      <anyo>1958</anyo>
    </fecha>
  </hecho>
   <hecho nombre="Código ASCII" descripcion="Nace a partir de reordenar y expandir el
 conjunto de símbolos y caracteres utilizados en aquel momento en telegrafía por la
 compañía Bell.">
    <autor>ANSI</autor>
    <fecha>
      <dia></dia>
      <mes></mes>
      <anyo>1963</anyo>
    </fecha>
  </hecho>
   <hecho nombre="Chips" descripcion="Estructura de pequeñas dimensiones de material
 semiconductor, sobre la que se fabrican circuitos electrónicos.">
    <autor>Jack S. Kilby y Robert Noyce</autor>
    <fecha>
      <dia></dia>
      <mes></mes>
      <anyo>1964</anyo>
    </fecha>
```

```
 </hecho>
  <hecho nombre="ARPANET" descripcion="Red de computadoras construida como un medio para
enviar datos militares y conectar grupos de investigación, a través de los Estados
Unidos.">
    <autor>MIT (Massachusetts Institute of Technology)</autor>
    <fecha>
      <dia></dia>
      <mes></mes>
      <anyo>1969</anyo>
   </fecha>
  </hecho>
  <hecho nombre="MS-DOS" descripcion="Es un sistema operativo de disco para el IBM PC y
sistemas compatibles. Este sistema operativo dominó el mercado de los ordenadores
personales entre 1985 y 1995.">
    <autor>Microsoft</autor>
    <fecha>
      <dia></dia>
      <mes></mes>
      <anyo>1981</anyo>
   </fecha>
  </hecho>
   <hecho nombre="World Wide Web" descripcion="Es un sistema de distribución de
documentos de hipertexto o hipermedios interconectados y accesibles via Internet.">
    <autor>Tim Berners Lee</autor>
    <fecha>
      <dia></dia>
      <mes></mes>
      <anyo>1990</anyo>
    </fecha>
   </hecho>
   <hecho nombre="Núcleo Linux" descripcion="Linux es el núcleo de un sistema operativo
libre, basado en Unix. Linux está licenciado bajo la GPL v2 y está desarrollado por
colaboradores de todo el mundo.">
    <autor>Linus Torvalds</autor>
    <fecha>
      <dia></dia>
      <mes></mes>
      <anyo>1991</anyo>
    </fecha>
   </hecho>
   <hecho nombre="Pentium 2" descripcion="El Pentium II es un microprocesador con
   arquitectura x86 diseñado por Intel.">
    <autor>INTEL</autor>
    <fecha>
      <dia>7</dia>
      <mes>Mayo</mes>
      <anyo>1997</anyo>
    </fecha>
   </hecho>
   <hecho nombre="Ordenadores cuánticos" descripcion="Se basan en los principios de la
mecánica cuántica.En lugar de utilizar bits utilizan qubits, que pueden tener los dos
estados de procesamiento al mismo tiempo, o ninguno. Esta tecnologia mejora mucho la
velocidad de procesamiento.">
    <autor>IBM</autor>
    <fecha>
      <dia></dia>
      <mes>Enero</mes>
      <anyo>2019</anyo>
   </fecha>
  </hecho>
   </evolucion>
```

Documentos XML bien formados

ENUNCIADO

No debes confundir el concepto de «documento XML bien formado» con el de «documento XML válido».

- Un documento XML decimos que está bien formado cuando cumple las reglas sintácticas establecidas por el W3C.
- Un documento XML decimos que es válido si, además de estar bien formado, cumple las reglas que establece su DTD o esquema.

A continuación, se incluye un documento XML y debes determinar si está bien formado o no. En el caso de que no esté bien formado, debes decir qué habría que modificar para que sí lo estuviera.

Para este caso práctico extendido utilizaremos el software XML Copy Editor. Es software libre, y dispone de versiones para GNU/Linux, Windows y sistemas Mac OS. Lo puedes descargar en el siguiente enlace: https://sourceforge.net/projects/xml-copy-editor/files/.

```
<?xml version="1.0" encoding="UTF-8"?>
<planetas>
      <planeta interior>
          <nombre>Mercurio</nombre>
          <diametro 0,382 />
       </planeta>
      <planeta interior>
            <nombre>Venus</nombre>
            <diametro 0,949 />
      </planeta>
      <planeta interior>
            <nombre>La Tierra</nombre>
           <diametro 1,00 />
      </planeta>
      <planeta interior>
           <nombre>Marte</nombre>
            <diametro 0,53 />
      </planeta>
      <planeta exterior>
           <nombre>Jupiter</nombre>
           <diametro 11,2 />
     </planeta>
    <planeta exterior>
           <nombre>Sarturno</nombre>
          <diametro 9,41 />
      </planeta>
      <planeta exterior>
          <nombre>Urano</nombre>
          <diametro 3,98 />
    </planeta>
    <planeta interior>
```

```
            <nombre>Neptuno</nombre>
          <diametro 3,81 />
      </planeta>
</planetas>
```

CLAVES DE RESOLUCIÓN

1. Descarga el software e instala en el ordenador la versión correspondiente. Ejecuta la aplicación.
2. Crea un nuevo documento XML. Guárdalo como archivo con extensión .xml, asignandole un nombre.
3. Copia y pega el código correspondiente. Vuelve a guardar el archivo.
4. Si hubiera algún error, XML Copy Editor nos avisaría en la barra inferior, indicando el tipo de error, la línea y la columna donde está situado.

SOLUCIÓN

Los errores detectados hacen referencia a la NO utilización de atributos en las etiquetas identificativas.

Por ejemplo, no podemos escribir `<planeta interior>`, sino que lo correcto sería `<planeta tipo="interior">`.

Para los diámetros pasa exactamente lo mismo. Habría que añadir un atributo para identificar el diámetro. Por ejemplo:

```
<diametro referencia="3,81"/>
```

Otra opción hubiera sido escribirlo de esta forma:

```
<diametro>3,81</diametro>
```

APOYO MULTIMEDIA

En el vídeo siguiente encontrarás claves adicionales para resolver este Caso Práctico.

El archivo con la solución del caso práctico es planetas_bienformado.xml y lo puedes descargar en la pantalla situada justo a continuación del vídeo.

5. Espacio de nombres en XML

Si se quiere eliminar problemas derivados de una posible repetición de nombre o atributo, la solución pasa por asignar a cada vocabulario XML un espacio de nombres, es decir, un espacio semántico único. De esta forma desaparecen posibles ambigüedades al trabajar conjuntamente con documentos XML que han utilizado los mismos nombres de elementos o atributos para representar cosas diferentes.
Por ejemplo:

```
<capital>
      <pais>Francia</pais>
      <nombre>París</nombre>
</capital>
<capital>
      <bonos>
      <cantidad>10000 €</cantidad>
      </bonos>
      <letras>
      <precio>1000 €</precio>
      </letras>
</capital>
```

Si en un documento XML se incluyen los elementos <capital> hay un conflicto de nombres.
Por lo tanto, los espacios de nombres en XML se utilizan para:

- Diferenciar entre elementos y atributos de vocabularios diferentes con diferentes significados pero que comparten nombre.
- Agrupar todos los elementos y atributos relacionados de un documento XML para que el software pueda reconocerlos con facilidad.

La utilización de espacios de nombres XML es una recomendación del W3C.
Cuando en un documento XML hay conflicto de nombres de etiquetas a estas etiquetas, para referenciarlas se les añade un prefijo que especifica a qué contexto pertenece esa etiqueta.
Esto es:

```
<prefijo:etiqueta></prefijo:etiqueta>
```

Pero previamente hay que declarar el espacio de nombres para cada una de esas etiquetas conflictivas.
Declarar un espacio de nombres consiste en asociar un índice con el URI asignado al espacio de nombres. Para ello se utiliza el atributo reservado **`xmlns`**, y se inserta entre el prólogo y el ejemplar del documento XML de la forma siguiente:

```
<elemento xmlns:nombre_prefijo="URL">
<capital xmlns:capital="http://localhost/UD1/capital">
<contab xmlns:capital="http://localhost/UD1/contab">
```

Donde:

- El espacio de nombres se inicia con la palabra clave **xmlns**.
- La palabra nombre_prefijo es el prefijo del espacio de nombres. En el ejemplo, '`capital`'.
- La URI es el identificador del espacio de nombres. Actúa como un identificador único.

El prefijo actúa como un alias, de forma que todos los nombres de elementos que lleven ese prefijo pertenecen a ese espacio de nombres.

De esta forma, cuando nos referimos en el documento a `capital` habrá que indicarlo con su prefijo correspondiente:

```
<capital:capital>
```

Espacios de nombres XML

INTRODUCCIÓN TEÓRICA

En XML un espacio de nombres es una colección de nombres (nombres de elementos y atributos) identificados por un URI.

Un URI (Uniform Resource Identifier) o identificador de recursos uniforme, es una cadena de caracteres que identifica los recursos de una red de forma unívoca.

Los nombres de un espacio de nombres XML aparecen en forma de nombres que contienen un prefijo de espacio de nombre, los dos puntos (:) y la parte local. Si no se indica un prefijo, se utiliza el espacio de nombres por defecto.

El prefijo puede utilizar cualquier cadena de caracteres siempre que no comience por xml (ya sea en mayúsculas o minúsculas).

Recuerda que los espacios de nombres en XML se utilizan para:

- Diferenciar entre elementos y atributos de vocabularios diferentes, con distintos significados, pero que comparten nombre.
- Agrupar todos los elementos y atributos relacionados de un documento XML, para que el software pueda reconocerlos con facilidad.

IMPORTANTE

Los URI especificados en un documento XML no tienen porqué contener nada, su función es ser únicos.

ENUNCIADO

Se quiere crear un documento XML que represente los préstamos de libros a personas en una biblioteca.

- De la persona que toma el libro se guardará su nombre, apellidos, correo electrónico y DNI.
- De los libros prestados se guardará el título, año de edición, autor (nombre y apellidos) y número de páginas.

CLAVES DE RESOLUCIÓN

- Vemos que se repiten campos que, en este caso, tienen el mismo significado semántico, pero no tiene por qué ser así siempre. Por ejemplo, podríamos estar hablando de 'cartas' con el sentido de las que enviamos como correo y 'cartas' como las del menú de un restaurante.
- Para evitar la ambigüedad en el nombre y apellidos, hay que introducir los espacios de nombres.

```
        <pais>Francia</pais>
        <nombre>París</nombre>
    </capital:capital>
    <contab:capital>
        <bonos>
        <cantidad>10000 €</cantidad>
        </bonos>
        <letras>
        <precio>1000 €</precio>
        </letras>
    </contab:capital>
```

Así no hay confusión sobre a qué `capital` nos referimos en cada caso y a qué espacio de nombres corresponde.

El URI, *Uniform Resource Identifier*, o identificador uniforme de recursos, es realmente el nombre del espacio de nombres y no se interpreta como una dirección web sino como una cadena de texto que identifica el espacio de nombres (el *parser* XML lo trata como una cadena de texto).

Es importante diferenciar entre URI y URL.

URI (Uniform Resource Identifier): es una cadena de caracteres que identifica los recursos de una red de forma unívoca.

URL (Uniform Resource Locator): es un identificador de recursos uniforme URI cuyos recursos referidos pueden cambiar, esto es, la dirección puede apuntar a recursos variables en el tiempo.

El concepto de espacio de nombres XML se retomará en otra unidad de forma más ampliada, así como la definición de vocabularios XML en la DTD o en el Schema.

SOLUCIÓN

El dcumento XML correspondiente, sin espacios de nombres y bien formado sería el siguiente:

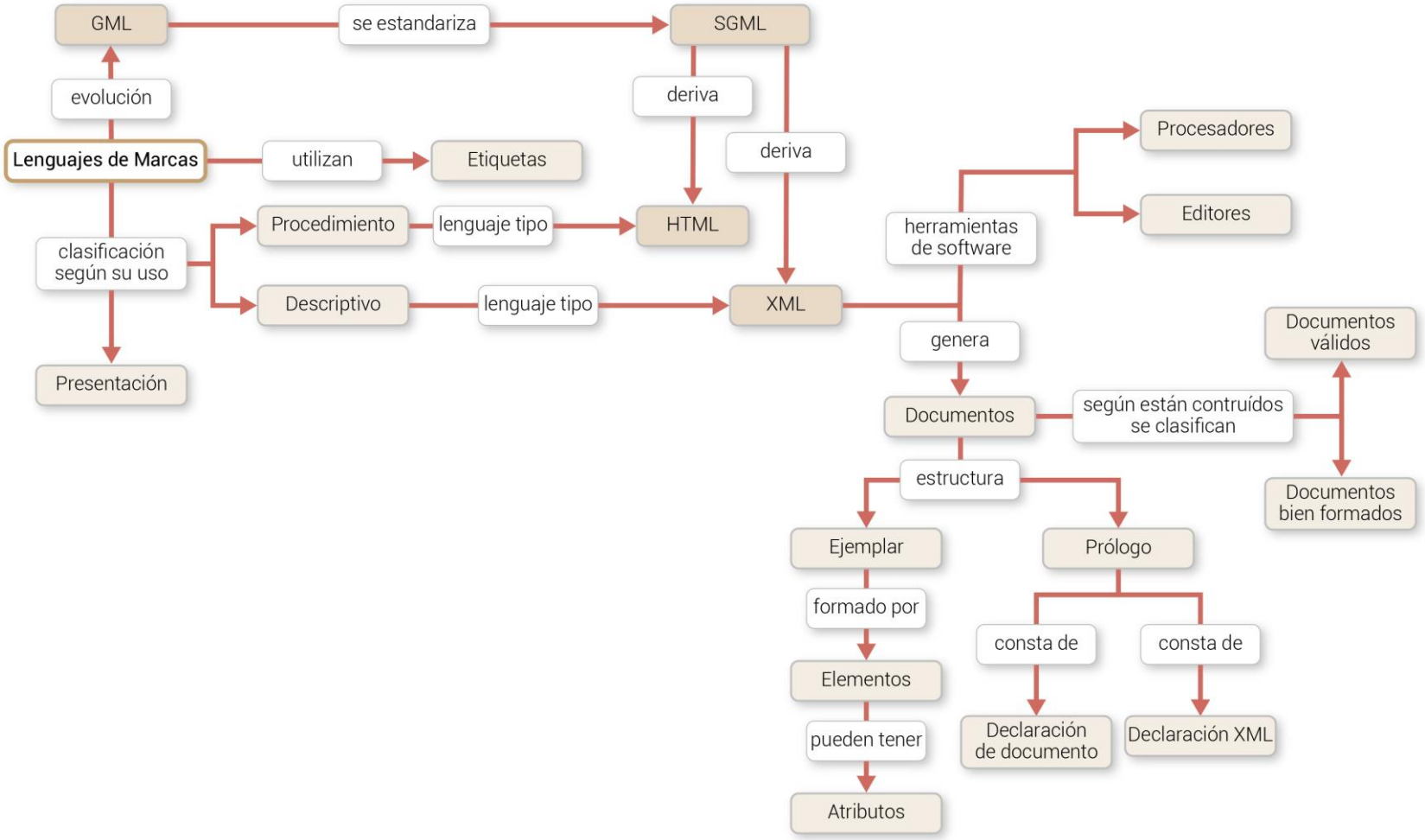
```
<?xml version="1.0" encoding="ISO-8859-1" standalone="no"?>
<prestamos>
    <prestamo>
        <persona>
            <nombre>Juan</nombre>
            <apellidos>Pérez García</apellidos>
            <dni>20345678</dni>
            <email>juanperez@servidor.com</email>
            <fecha>22/07/2021</fecha>
        </persona>
   <libro>
        <titulo>El asombroso viaje de Pomponio Flato</titulo>
        <autor>
            <nombre>Eduardo</nombre>
            <apellidos>Mendoza Garriga</apellidos>
        </autor>
        <edicion>2008</edicion>
        <paginas>192</paginas>
   </libro>
  </prestamo>
</prestamos>
```

Insertamos ahora dos espacios de nombres, uno para la persona que toma el préstamo y otro para el autor del libro. Observa que la declaración del espacio de nombres no se ha hecho en el elemento raíz, sino junto a las etiquetas afectadas.

La solución completa quedaría de la siguiente forma:

```
<?xml version="1.0" encoding="utf-8" standalone="no"?>
<prestamos>
    <prestamo>
        <p:persona xmlns:p="http://localhost/UD1/persona">
            <p:nombre>Juan</p:nombre>
            <p:apellidos>Pérez García</p:apellidos>
            <p:dni>20345678</p:dni>
            <p:email>juanperez@servidor.com</p:email>
            <p:fecha>22/07/2021</p:fecha>
        </p:persona>
   <libro>
        <titulo>El asombroso viaje de Pomponio Flato</titulo>
        <a:autor xmlns:a="http://localhost/UD1/autor">
            <a:nombre>Eduardo</a:nombre>
            <a:apellidos>Mendoza Garriga</a:apellidos>
        </a:autor>
        <edicion>2008</edicion>
        <paginas>192</paginas>
   </libro>
  </prestamo>
</prestamos>
```

CARACTERÍSTICAS DE LOS LENGUAJES DE MARCAS



1 UNIDAD

Otros criterios de clasificación de los lenguajes de marcas

En función de su ámbito de utilización, los lenguajes de marcas se pueden agrupar en:

- RTF. Rich Text Format, formato de texto enriquecido. Desarrollado por Microsoft en 1987, permite el intercambio de documentos de texto entre distintos procesadores de texto e incluso entre sistemas operativos incompatibles.
- TeX. Lenguaje de marcas de tipo tipográfico utilizado en entornos académicos en trabajos matemáticos, físicos o informáticos.
- Wikitexto. Lenguaje de marcas dedicado a la creación de páginas wiki, como por ejemplo la Wikipedia.
- DocBook. Lenguaje de marcas de tipo semántico que permite generar documentos con una estructura lógica independientemente de su formato. Estos documentos pueden ser publicados en diferentes formatos, aplicando la plantilla correspondiente y sin necesidad de modificar el documento original.

En función de las tecnologías de Internet implicadas, los lenguajes de marcas se pueden agrupar en:

- HTML. Lenguaje de marcas utilizado para la creación de páginas web.
- XHTML. eXtensible HyperText Markup Language, lenguaje de marcas que intenta seguir las normas XML y permite la compatibilidad con gran cantidad de formatos de imágenes, vídeos e incluso con lenguajes de scripts o guiones.
- RSS. Really Simple Syndication, lenguaje de marcas que permite compartir contenidos de la web y sigue el estándar XML. En concreto, se utiliza para la difusión, desde una fuente de contenidos, de información a usuarios que se han suscrito.

Por último, están los lenguajes de marcas especializados. Entre ellos:

- MathML. Mathematical Markup Language, es utilizado en páginas web, junto con XHTML, para poder mostrar fórmulas matemáticas e intercambiar información entre software matemático. Basado en XML.
- VoiceXML. Voice Extended Markup Language, lenguaje de marcas para la implementación de sistemas de diálogo, es decir, permite describir servicios de voz independientemente de la aplicación utilizada.
- MusicXML. Lenguaje de marcas utilizado en notación musical y que permite el intercambio de partituras entre distintos editores.

1 UNIDAD

Conversores XML

A continuación se incluye una lista de conversores XML disponibles.

1. Se puede hacer la conversión en línea de forma gratuita desde varios enlaces:

- Code Beautify: https://codebeautify.org
- Online Convert Free: https://onlineconvertfree.com/es/convert/xml
- Aconvert: https://www.aconvert.com/es/format/xml
- PDF MaLL: https://pdfmall.com/es/xml-to-pdf

2. Se pueden utilizar herramientas específicas como:

- Altova MapForce: https://www.altova.com
- Advanced XML Converter: http://www.xml-converter.com

1 UNIDAD

Organizaciones internacionales desarrolladoras de estándares

Las principales organizaciones internacionales implicadas son:

- W3C (World Wide Web Consortium): https://www.w3c.es.
- ISO (International Standard Organization): https://www.iso.org.

El W3C es un consorcio internacional creado en 1994 por Tim Berners-Lee cuyo objetivo es la creación de recomendaciones y estándares, entre ellos los relativos a los lenguajes de marcas, pero sin llegar a la certificación.

Los estándares para XML, HTML, XHTML, CSS, RDF y muchos otros fueron desarrollados por el W3C.

Estos estándares y recomendaciones son los que garantizan la interoperabilidad de todo el software relacionado con la web. Para ello, W3C monta grupos de trabajo, cuyos componentes son miembros de la organización y también invitados expertos, y que elaboran las recomendaciones que luego serán aprobadas por el consorcio.

La Organización Internacional para Normalización (ISO) creó la norma ISO 8879: 1986, que establecía el estándar para definir lenguajes generalizados de marcado para documentos, en concreto SGML.

Puedes consultar información adicional y actualizada en español sobre los estándares web en https://www.w3c.es/estandares.