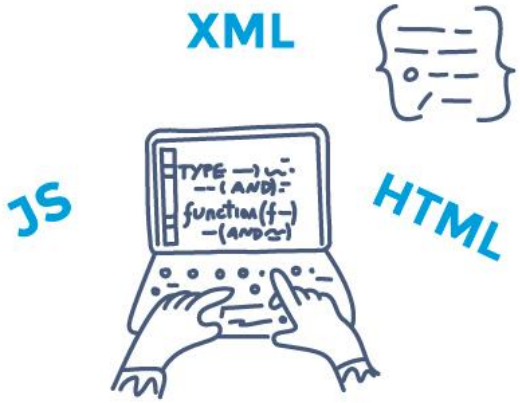
Tema 1. Introducción a los lenguajes de marcas. XML

1. Lenguajes de marcas

Un **lenguaje de marcas** es un sistema de representación de información basado en etiquetas que permite estructurar, identificar y describir datos. La idea central es separar el contenido de su presentación, para favorecer la reutilización, el intercambio y el procesado automático.



Los aspectos clave que distinguen a los lenguajes de marcas son:

- Uso de etiquetas y estructura jerárquica (en forma de árbol).
- Separación entre contenido y presentación.
- Legibilidad por humanos y máquinas.
- Independencia de plataforma y formato (texto plano).

Un ejemplo de Lenguaje de Marcas sería el siguiente, en el que vemos cómo se encierra el texto entre etiquetas para darle un significado o una representación:

undefined - ejemplo.xml

```xml
<carta>
    <fecha>22/11/2006</fecha>
    <presentacion>Estimado cliente:</presentacion>
    <contenido>bla bla bla bla …</contenido>
    <firma>Don José Gutiérrez González</firma>
</carta>
```

1.1. Evolución de los lenguajes de marcas

El lenguaje de marcado ha ido evolucionando a lo largo del tiempo:

SGML (Standard Generalized Markup Language)

Proviene de **GML** (Generalized Markup Language), un lenguaje que surgió en la década de los 70 en el seno de la empresa IBM, con el objetivo de gestionar grandes cantidades de información. Dada su gran utilidad, en 1986, la Organización Internacional de Estándares lo normalizó creando **SGML** (Standard Generalized Markup Language). Cayó en desuso, fundamentalmente, por su dificultad y el hecho de que demandaba herramientas de software demasiado costosas. En la actualidad, **ya no se utiliza**.

Lenguaje de marcado de hipertexto (HTML)

Es el lenguaje de la web, de manera que la inmensa mayoría de las páginas que existen están escritas en HTML. En 1991 la situación cambió drásticamente cuando Tim Berners-Lee, que conocía el SGML, utilizó su sintaxis para crear el HTML y compartir información entre científicos. Surgió de la necesidad de organizar, enlazar y compatibilizar la información proveniente de diferentes sistemas. Así se unieron dos estándares existentes: ASCII como codificador de caracteres y SGML para dar estructura al texto.

Básicamente, el *Hyper Text Markup Language* define los contenidos de un sitio web de forma textual y estructurada indicando al navegador cómo debe visualizarse el sitio. Su rápida expansión se debió al hecho de que era muy fácil de entender, lo que hizo que se convirtiera en un estándar general para el desarrollo de sitios y páginas web.

HTML5

Con la llegada de HTML5, no solo se incorporan nuevas etiquetas semánticas que **agregan significado a la página**, sino que también se puede añadir audio y vídeo sin recurrir a usar Flash u otro reproductor multimedia.

Otra gran ventaja de desarrollar aplicaciones HTML5 es que el resultado final es completamente accesible desde un ordenador, tableta o móvil. Los navegadores modernos más populares soportan HTML5, por lo que los usuarios pueden visualizar el contenido correctamente. Por supuesto, todo ello facilita y agiliza el proceso de programación, por lo que conocer este lenguaje de marcado es fundamental.

XML

Tras el nacimiento de la web, la popularización de HTML hizo este lenguaje de marcas creciera sin orden alguno, hasta que en 1996 la *World Wide Web Consortium (W3C)* se centró en crear un estándar que deberían adoptar todos los navegadores, para así facilitar el trabajo de los desarrolladores. XML nació como parte del estándar SGML, con el objetivo de solventar las carencias de HTML en lo que a tratamiento de la información se refiere. Esto es, cuando trabajamos con HTML, no sólo el contenido se mezcla con los estilos, dependiendo la presentación del navegador que se utilice, sino que este estándar no **nos permite el intercambio de información con otros dispositivos**.

1.2. Uso de etiquetas

Los lenguajes de marcas utilizan una serie de **etiquetas** especiales intercaladas en un documento de texto sin formato. Dichas etiquetas serán posteriormente interpretadas por los intérpretes del lenguaje y ayudan al procesado del documento.

Las etiquetas se escriben encerradas entre ángulos, es decir < y >. Normalmente, se utilizan dos etiquetas: una de inicio y otra de fin para indicar que ha terminado el efecto que queríamos presentar. La única diferencia entre ambas es que la de cierre lleva una barra inclinada "/" antes del código.

```
<etiqueta>datos</etiqueta>
```

1.3. Ventajas de los lenguajes de marcas

Los lenguajes de marcas ofrecen numerosas ventajas en el tratamiento de la información, ya que **permiten organizar, estructurar y dar significado a los datos de forma clara y estandarizada**. Una de sus principales

aportaciones es la **separación entre el contenido y la presentación**, lo que facilita reutilizar la misma información en diferentes contextos y dispositivos sin necesidad de modificarla.

Además, favorecen la **interoperabilidad**, ya que al basarse en texto plano y en reglas sintácticas comunes, los documentos creados con lenguajes de marcas pueden ser compartidos, leídos y procesados por múltiples aplicaciones y sistemas sin depender de un software específico.

Otra ventaja destacada es su **flexibilidad**, al permitir combinarse con otros lenguajes, enriquecer los datos y adaptarse a distintas necesidades. Esto los convierte en una herramienta fundamental para el intercambio de información en entornos distribuidos, como Internet.

También proporcionan una **mayor accesibilidad**, puesto que los documentos escritos con lenguajes de marcas pueden ser interpretados por dispositivos muy diversos, desde navegadores web hasta lectores de pantalla, garantizando que la información esté disponible para un público más amplio. Además, cuando hablamos de información, no hablamos sólo de texto, también de gráficas, imágenes, etc.

1.4. Clasificación de los lenguajes de marcas

Los lenguajes de marcas se pueden clasificar según su propósito o finalidad en:

- Presentación: Se centran en la apariencia de los datos. Éstos suelen ocultar las etiquetas y mostrar al usuario solamente el texto con su formato. Ejemplos: HTML, Markdown, RTF, LaTeX.
- Intercambio de datos o de propósito general: Se centran en estructurar, almacenar y transportar información, sin preocuparse de la presentación. Ejemplos: XML, JSON, YAML.
- Específicos de dominio: Son **aplicaciones concretas de XML** (o de otros lenguajes) en dominios específicos, respondiendo a necesidades en un sector o área de conocimiento. Ejemplos: SVG, MathML, XHTML, RSS.

Ejemplo de un mismo dato representado en diferentes lenguajes:

HTML: `<p>Hola mundo</p>`

XML: `<mensaje>Hola mundo</mensaje>`

JSON: `{ "mensaje": "Hola mundo" }`

2. XML

XML (e**X**tensible **M**arkup **L**anguage) se define como un metalenguaje extensible de etiquetas desarrollado por el W3C, como una simplificación y adaptación de SGML, permitiendo a los desarrolladores definir la gramática de lenguajes específicos. Propuesto como un estándar para el intercambio de información estructurada entre diferentes plataformas, XML no es realmente un lenguaje en particular, sino una manera de definir lenguajes para diferentes necesidades. Su papel en la actualidad es bastante importante, ya que permite la compatibilidad entre sistemas para compartir la información de una manera segura, fiable y fácil, además de utilizarse como base para la creación de lenguajes estándares que describen diferentes tipos de datos: **SVG** (*Scalable Vector Graphics*), **MathML** (*Mathematical Markup Language* – descripción de fórmulas matemáticas), **ODT** (*Open Document Format*), **RSS** (*Really Simple Syndication*), **ePUB** (formato de libro digital), **XHTML** (versión extendida de HTML que estudiaremos próximamente), etc.

2.1. Documentos XML

XML es un lenguaje de marcas **orientado a la descripción**, esto es, determinadas marcas permitirán dar significado al texto sin indicar cómo representarlo en pantalla. De esta forma, un documento XML contiene datos que se autodefinen, en tanto que en un documento HTML, como veremos más adelante, datos y representación están indisolublemente unidos. XML busca dar solución al problema de **expresar información estructurada, de la manera más abstracta y reutilizable posible**. Que la información sea estructurada quiere decir que se compone de partes bien definidas, y que esas partes se componen a su vez de otras partes. De esta manera, XML se puede emplear como un lenguaje limpio que permite describir información de forma precisa, sin tener que preocuparnos en ningún momento de su representación final.

Veamos un sencillo ejemplo de documento XML.

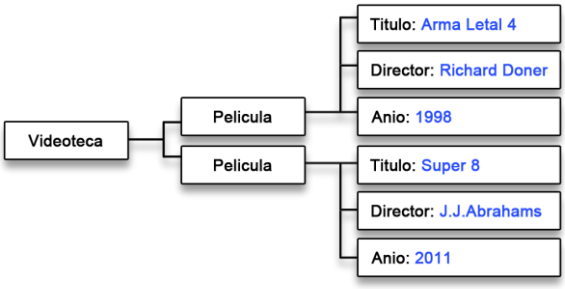
```xml
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE videoteca SYSTEM "videoteca.dtd">

<videoteca>
    <pelicula>
        <titulo>Arma Letal 4</titulo>
        <director>Richard Donner</director>
        <anio>1998</anio>
    </pelicula>
    <pelicula>
        <titulo>Super 8</titulo>
        <director>J.J.Abrahams</director>
        <anio>2011</anio>
    </pelicula>
</videoteca>
```

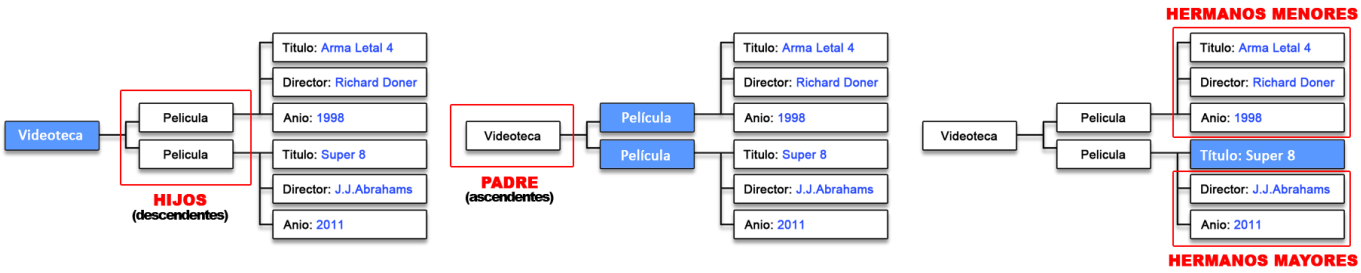
Y aquí el código del *DTD (Document Type Definition)* del documento `videoteca.dtd`:

```
<!ELEMENT videoteca (pelicula+)>
<!ELEMENT pelicula (titulo,director,anio)>
<!ELEMENT titulo (#PCDATA)>
<!ELEMENT director (#PCDATA)>
<!ELEMENT anio (#PCDATA)>
```

En definitiva, un documento XML está constituido por texto plano y **etiquetas** (marcas) que permiten clasificar al texto dotándolo de cierto significado. A diferencia de lo que ocurre con HTML, aquí no existen etiquetas predefinidas, por lo que el desarrollador podrá definir aquellas que desee según sus necesidades. Como vemos en el ejemplo anterior, la estructura del documento nos permite agrupar bajo diferentes etiquetas, creadas por el propio desarrollador, un conjunto de datos relacionados entre sí, generando una **estructura de tipo árbol** como la que vemos en la siguiente imagen.



La jerarquía formada entre los elementos del árbol establece automáticamente relaciones entre padres, hijos, hermanos, ascendentes y descendientes. De esta forma, las partes del árbol que tienen hijos se las denomina **nodos intermedios**, en tanto que aquellas que no tienen se conocen como **hojas**.



Es importante conocer la sintaxis de XML para poder crear documentos bien-formados. Un documento escrito bajo estándar XML debe seguir una estructura estrictamente jerárquica, en lo referente a las etiquetas que delimitan a sus elementos.

```
<p>El lenguaje HTML <b>permite <i>esto</b></i></p>
```

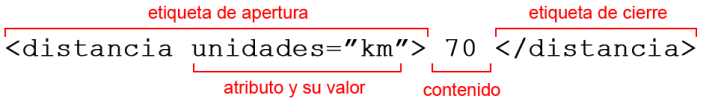
Según el estándar XML, la línea de código anterior sería incorrecta ya que éste exige que cada etiqueta esté correctamente contenida dentro de otra, teniendo en cuenta que todas deberán estar cerrarse en orden inverso al que se abrieron.

```
<p>En el lenguaje XML <b>la estructura <i>debe ser</i> jerárquica</b></p>
```

2.2. Elementos básicos y atributos

Como se introdujo al principio de la página, los **elementos básicos** de XML son las marcas o etiquetas. Éstas no son más que contenedores de información y deben aparecer en parejas, delimitando de esta manera su contenido que, a su vez puede ser vacío, contener texto, atributos u otros elementos de XML. Las etiquetas pueden completarse

con **atributos** que nos permitan definir ciertas características o propiedades de dicho elemento. Éstos están constituidos por pares *nombre-valor* que son definidos en la etiqueta de apertura.



Por ejemplo, supongamos que en el ejemplo propuesto anteriormente, deseamos indicar la calidad de cada una de las películas. Podemos hacerlo incorporando un atributo llamado `calidad`, al elemento película. De esta manera, el código XML anterior quedaría.

```
<pelicula calidad=”Muy buena”>
    <titulo>Arma Letal 4</titulo>
    <director>Richard Donner</director>
    <anio>1998</anio>
</pelicula>
```

No hay una regla fija para saber cuándo añadir atributos pero, algo que nos puede ayudar es pensar qué datos son reales y cuáles describen a los primeros. Los siguientes ejemplos contienen la misma información:

```
<carta fecha="2008-01-10">
  <a>Carlos</a>
  <de>Alba</de>
</carta>
```

```
<carta>
  <fecha>2008-01-10</fecha>
  <a>Carlos</a>
  <de>Alba</de>
</carta>
```

```
<carta>
  <fecha>
    <anio>2008</anio>
    <mes>01</mes>
    <dia>10</dia>
  </fecha>
  <a>Carlos</a>
  <de>Alba</de>
</carta>
```

Algunas cosas a tener en cuenta al utilizar atributos son:

- Los atributos **no** pueden contener múltiples valores (los elementos sí)
- Los atributos **no** pueden contener estructuras de árbol (los elementos sí)
- Los atributos **no** son fácilmente expandibles (para cambios futuros)

En XML también podemos encontrar **elementos vacíos**, estos son los que no tienen contenido y que, aunque pueden tener atributos, se abren y cierran con una sola etiqueta.

```
<separador distancia=”7” />
```

```
<separador distancia=”7”></separador>
```

Los elementos se emplean por tanto para representar jerarquías o contenido y, el orden en el que aparecen no es representativo, pudiendo encontrar a lo largo del documento XML múltiples ocurrencias de dicho elemento. Además, un elemento, etiqueta o marca puede contener atributos que modifican la información, no pudiendo aparecer un mismo atributo más de una vez en una misma etiqueta.

2.2. Estructura del documento XML

Aunque no es obligatorio, es aconsejable que un documento XML comience con una línea que describa la versión de la especificación utilizada y el tipo de documento. En ocasiones también se incluye en esta definición el atributo **`standalone`**, cuyo valor puede ser **`yes`** o **`no`**, y especificará si un documento depende o no de otros, como una DTD.

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

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

Seguidamente, hemos de realizar la declaración que define qué tipo de documento estamos creando, para que pueda ser procesado correctamente. Es decir, indicamos una **DTD** (*Document Type Definition*) válida, donde se definen las reglas que debe cumplir nuestro documento XML. En nuestro caso, si nos fijamos en el ejemplo anterior, indicamos que el contenido de nuestro archivo XML se basa en la estructura definida en la DTD contenida en el archivo `videoteca.dtd`, cuyo elemento raíz es `videoteca`.

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

A continuación, en el documento encontramos una estructura jerárquica de elementos que, a su vez pueden contener a otros elementos, estar asociados directamente con cierto contenido, o bien ser elementos vacíos. Un elemento de contenido es, por ejemplo:

```
    <pelicula>Arma Letal 4</pelicula>
```

2.3. Principales características de un documento XML bien formado

XML es un lenguaje estricto y es sensible a mayúsculas y minúsculas, por lo que si se define un elemento de una u otra manera, siempre tendremos que referirnos a él de la misma forma. Al emplear XML será necesario asignar nombre a las estructuras, tipos de elemento, entidades, etc. En este caso, debemos seguir lo exigido por el estándar que nos dice que: “**Un nombre debe empezar con una letra, o un guión bajo, continuándose con letras, dígitos, guiones, rayas, dos puntos o puntos**.” Está terminante prohibido que el nombre de un elemento empiece por la cadena XML en alguna de sus variantes (empleando mayúsculas y/o minúsculas).

Por lo tanto, las principales características de XML, que deberemos tener muy en cuenta a la hora de crear documentos bien formados son:

a. Todos los elementos XML deben cerrarse.

```
<p>This is a paragraph.</p>
<br />
```

b. XML es sensible a mayúsculas y minúsculas.

```
<mensaje>This is correct</mensaje>
<mensaje>This is NOT correct</Mensaje>
```

c. Los elementos XML deben estar siempre correctamente anidados.

```
<b><i>This text is bold and italic</i></b>
```

d. Un documento XML tiene **un único elemento raíz**.

```
<raiz>
  <hijo>
    <subhijo>.....</subhijo>
  </hijo>
</raiz>
```

e. Todos los atributos de un elemento XML deben encerrar su valor entre comillas.

```
<carta date="12/11/2007">
  <a>Carlos</a>
  <de>Alba</de>
</carta>
```

f. Los comentarios en XML comienzan por ***<!--*** y terminan por ***-->***. No pueden aparecer antes de la especificación del documento, versión y tipo de codificación.

```
<!-- This is a comment -->
```

g. Los espacios en blanco (tabulador, nueva línea, retorno de carro y espacio) se conservan en las secciones **PCDATA** (*Parsed Character Data*) que encontramos como contenido de las etiquetas. Como valor de un atributo, los espacios en blanco adyacentes se condensarán en uno sólo.

XML:	Hello	Tove
HTML:	Hello Tove	

h. XML utiliza 5 entidades predefinidas para representar los caracteres: **'** (***'***), **"** (***"***), **<** (***<***), **>** (***>***), **&** (***&***).

```
<message>salary &lt; 1000</message>
```

Un documento XML con una sintaxis correcta se denomina "bien formado". Sin embargo, un documento XML validado contra una DTD es "bien formado" y "válido".

Para validar el documento XML, usamos el **Document Type Definition** o DTD (en español "definición de tipo de documento"), el cual define los tipos de elementos, atributos y entidades permitidas, y puede expresar algunas limitaciones para combinarlos. Los documentos XML que se ajustan a su DTD son denominados válidos.

2.4. Parser

Se conoce con el nombre de **parser** o **analizador XML**, a la aplicación que se encarga de leer un documento XML y determinar la estructura y propiedades de los datos contenidos en él, generando el árbol jerárquico asociado. Definimos un **parser validador** como aquel que, además se encarga de comprobar que el código XML que conforma el documento, cumple unas determinadas reglas propuestas por el desarrollador, comprobando de esta forma la semántica del documento e informando de posibles errores. Son muchas las herramientas existentes para el manejo de XML y sus tecnologías (DTD, XSD, XPATH, XSLT, etc.), entre todas ellas destacamos: XMLSpy, <oXygen/>, XML Copy Editor, XMLPad Pro Edition, Exchanger XML Editor y Liquid XML Editor.

2.5. Espacios de nombres

En ocasiones, en un mismo documento XML, se producen conflictos de nombres entre elementos o atributos con diferentes definiciones. Esto supone un problema para el parser que no sabría cómo manejar etiquetas iguales. Los **espacios de nombres** o **namespaces** evitan este problema, indicando en cada etiqueta el contexto para cada una de ellas, permitiendo de esta manera el poder diferenciarlas. En resumen, un espacio de nombres supone una recomendación del W3C para eliminar ambigüedades entre elementos y atributos de un documento XML.

```
<?xml version=”1.0” encoding="UTF-8" ?>
<!DOCTYPE videoteca SYSTEM "videoteca.dtd">

<videoteca>
    <titulo>Videoteca de Alex</titulo>
    <pelicula>
        <titulo>Arma Letal 4</titulo>
        <director>Richard Donner</director>
        <anio>1998</anio>
    </pelicula>
</videoteca>
```

Observamos en el ejemplo anterior que la etiqueta **`titulo`** se utiliza en contextos diferentes: el primero de ellos da título a la videoteca y el segundo se corresponde con el título de la película en particular. Esto, como se dijo anteriormente, supone un conflicto para el parser.

La solución a este problema consiste en agregar un prefijo al nombre de la etiqueta que identifique el propietario de la misma.

```
<?xml version=”1.0” encoding="UTF-8" ?>
<!DOCTYPE videoteca SYSTEM "videoteca.dtd">

<videoteca>
    <video:titulo>Videoteca de Alex</video:titulo>
    <video:pelicula>
        <ficha:titulo>Arma Letal 4</ficha:titulo>
        <ficha:director>Richard Donner</ficha:director>
        <ficha:anio>1998</ficha:anio>
    </video:pelicula>
</videoteca>
```

Cuando usamos prefijos en XML, se debe definir un **espacio de nombres** para el prefijo. Utilizamos el atributo **`xmlns`** (xml namespace) en la etiqueta de apertura de un elemento. La declaración del espacio de nombres sigue la siguiente sintaxis:

```
xmlns:prefix="URI"
```

donde **URI** es el identificador único del recurso que define un nombre lógico para el espacio de nombres, resolviendo de esta manera posibles problemas de duplicidades.

Cuando se define un espacio de nombres para un elemento, todos los elementos secundarios con el mismo prefijo se asocian con el mismo espacio de nombres. A esto se le denomina **ámbito** de declaración del espacio de nombres.

Los espacios de nombres también se pueden declarar en el elemento raíz XML:

```
<raiz xmlns:h="http://www.w3.org/TR/html4/"
xmlns:f="https://www.w3schools.com/furniture">

<h:table>
  <h:tr>
    <h:td>Apples</h:td>
    <h:td>Bananas</h:td>
  </h:tr>
</h:table>

<f:table>
  <f:name>African Coffee Table</f:name>
  <f:width>80</f:width>
  <f:length>120</f:length>
</f:table>

</raiz>
```

3. Otros lenguajes XML

Al comienzo del tema mencionamos algunos lenguajes, basados en XML, que se utilizan para propósitos específicos. Entre ellos encontramos los siguientes:

a. **SVG (Scalable Vector Graphics)**. Auspiciado por el W3C, SVG se define como un lenguaje de marcas para la representación de gráficos vectoriales bidimensionales, que actualmente están soportados por cualquier navegador. Almacena gráficos vectorizados escalables en formato texto, lo que, no sólo permite editarlo con cualquier aplicación de edición de textos, sino que también da lugar a archivos muy compactos. Además, soporta hojas de estilo y pueden generarse dinámicamente en un servidor web, como respuesta a la interacción con el usuario. Este formato es utilizado por numerosas aplicaciones gráficas, como Adobe Illustrator, Corel Draw, Inkscape, Mayura Draw, etc.
b. **WML (Wireless Markup Language)**. Es un lenguaje de marcas utilizado para representar la información que se visualiza en dispositivos móviles y asistentes digitales que utilicen tecnología **WAP** (Wireless Application Protocol - estándar utilizado para comunicaciones inalámbricas).

```
<?xml version="1.0" ?>
<!DOCTYPE wml PUBLIC "-//WAPFORUM//DTD WML 1.1//EN"
                     "http://www.wapforum.org/DTD/wml_1.1.xml">
<wml>
   <card id="MiPrimerWML" title="Primer WML">
      <p align="center">
         Mi primer ejercicio en WML
      </p>
   </card>
</wml>
```

c. **RSS (Really Simple Syndication)**. Se utiliza para sindicar o compartir contenido en la web, difundiendo información actualizada a los usuarios que se hayan suscrito previamente a la fuente de contenidos. Un archivo RSS contiene información básica sobre las novedades del sitio, como título, fecha de publicación o descripción. Generalmente se utilizan agredadores, esto es, aplicaciones web o de escritorio, diseñadas específicamente para la lectura de dichos contenidos, encargadas además de aplicar un estilo al contenido, presentándolos de forma atractiva al usuario. Alguno de los agregadores más utilizados son Flipboard, Feedly, Menéame, Scoop.it, Divúlgame, etc.

d. **DocBook**. Además de una aplicación, es un lenguaje utilizado para crear documentación, generalmente de tipo técnico, utilizando un formato neutro independiente de su presentación que permite especificar tanto contenido, como la estructura.

```
1  <?xml version="1.0" encoding="UTF-8" standalone="no"?>
2  <!DOCTYPE bookarticlearticlearticlearticle PUBLIC "-//OASIS//DTD DocBook XML V4.5//EN" "../
   docbook-xml-4.5/docbookx.dtd">
3  <book>
4  == Ejemplo Docbook Book
5          <chapter>
6  === Primer capítulo
7
8          === Primera seccion del primer capitulo
9              Ipsum reversus ab viral inferno, nam rick grimes malum cerebro. De carne lumbering
               animata corpora quaeritis. Summus brains sit, morbo vel maleficia? De apocalypsi
               gorger omero undead survivor dictum mauris...
10
11         === Segunda seccion del primer capitulo
12             Hi mindless mortuis soulless creaturas, imo evil stalking monstra adventus resi
               dentevil vultus comedat cerebella viventium. Qui animated corpse, cricket bat max
               brucks terribilem incessu zombie...
13
14         </chapter>
15 </book>
16
```

Bibliografía

- Lenguajes de Marcas. XML. Validación de documentos
  Jorge Sánchez Asenjo

- Lenguajes de Marcas y Sistemas de Gestión de la Información
  Juan Manuel Castro Ramos, José Ramón Rodríguez Sánchez
  Editorial Garceta

- Internet Encyclopedia
  Hossein Bidgoli
  California State University

- W3Schools