

# **Lenguajes de marcas y sistemas de gestión de información**

Juan Carlos Moreno Pérez  
Sergio Luis González Ruiz



# Índice

<b>PRESENTACIÓN .....</b>	9
---------------------------	---

<b>1. RECONOCIMIENTO DE LAS CARACTERÍSTICAS DE LENGUAJES DE MARCAS .....</b>	11
Objetivos .....	11
Mapa conceptual .....	12
Glosario .....	12
1.1. Introducción .....	13
1.2. Clasificación y características comunes de los lenguajes de marcas .....	15
1.2.1. Clasificación de los lenguajes de marcas .....	15
1.2.2. Características comunes .....	16
1.3. Identificación de ámbitos de aplicación .....	18
1.4. XML: estructura y sintaxis .....	18
1.5. Herramientas de edición .....	20
1.5.1. Bloc de notas Windows .....	20
1.5.2. Gedit .....	21
1.5.3. Notepad++ .....	21
1.6. Elaboración de documentos XML bien formados .....	22
1.7. Utilización de espacios de nombres en XML .....	24
Resumen .....	26
Ejercicios prácticos .....	27
Autoevaluación .....	27
<b>2. UTILIZACIÓN DE LENGUAJES DE MARCAS EN ENTORNOS WEB .....</b>	29
Objetivos .....	29
Mapa conceptual .....	30
Glosario .....	30
2.1. Introducción .....	30

<b>2.2. Herramientas de diseño web .....</b>	31
2.2.1. KompoZer .....	31
2.2.2. Adobe Dreamweaver CC .....	31
2.2.3. Bluefish .....	32
2.2.4. HTML Online Editor .....	32
<b>2.3. Identificación de etiquetas y atributos de HTML .....</b>	32
2.3.1. Estructura de un documento HTML .....	32
2.3.2. Versiones de HTML .....	34
2.3.3. Etiquetas y atributos HTML .....	36
<b>2.4. Hojas de estilo CSS .....</b>	55
2.4.1. Introducción a CSS .....	56
2.4.2. Versiones de CSS .....	57
2.4.3. Sintaxis básica .....	57
2.4.4. Maneras de aplicar estilos a un documento HTML .....	69
<b>2.5. XHTML .....</b>	72
2.5.1. Versiones .....	72
2.5.2. Diferencias sintácticas y estructurales con HTML .....	74
2.5.3. Ventajas de XHTML sobre HTML .....	77
<b>Resumen .....</b>	78
<b>Ejercicios prácticos resueltos .....</b>	79
<b>Ejercicios prácticos .....</b>	81
<b>Autoevaluación .....</b>	83
<b>3. APLICACIÓN DE LOS LENGUAJES DE MARCAS A LA SINDICACIÓN DE CONTENIDOS .....</b>	85
<b>Objetivos .....</b>	85
<b>Mapa conceptual .....</b>	86
<b>Glosario .....</b>	86
<b>3.1. Introducción .....</b>	87
<b>3.2. Ventajas y ámbitos de aplicación .....</b>	88
<b>3.3. Tecnologías y estándares de canales de contenidos .....</b>	89
3.3.1. RSS .....	90
3.3.2. Atom .....	90
<b>3.4. Estructura y elementos de los canales de contenidos .....</b>	91
3.4.1. Estructura y elementos de RSS 2.0 .....	91
3.4.2. Estructura y elementos de Atom .....	101
<b>3.5. Creación, validación y comprobación de funcionalidades de los canales de contenidos .....</b>	108
3.5.1. Creación de un feed RSS .....	108
3.5.2. Validación y comprobación .....	108
<b>3.6. Utilización de herramientas .....</b>	111
3.6.1. Herramientas de creación y edición de feeds RSS .....	111
3.6.2. Agregadores o lectores de feeds .....	114
<b>3.7. Directorios de canales de contenidos .....</b>	119
<b>Resumen .....</b>	120
<b>Ejercicios prácticos resueltos .....</b>	122
<b>Ejercicios prácticos .....</b>	123
<b>Autoevaluación .....</b>	123

<b>4. DEFINICIÓN DE ESQUEMAS Y VOCABULARIOS EN XML .....</b>	125
<b>Objetivos .....</b>	125
<b>Mapa conceptual .....</b>	126
<b>Glosario .....</b>	126
<b>4.1. Introducción .....</b>	126
<b>4.2. Definición de la estructura y sintaxis de documentos XML .....</b>	127
4.2.1. Instrucciones de procesamiento o prólogo .....	129
4.2.2. Marcas o etiquetas .....	130
4.2.3. Elementos .....	131
4.2.4. Atributos .....	132
4.2.5. Comentarios .....	133
4.2.6. Sección CDATA .....	133
<b>4.3. Creación, asociación y elementos de DTD y XSD .....</b>	134
4.3.1. DTD (definición de tipo de documento) .....	136
4.3.2. XML Schema XSD .....	145
<b>4.4. Herramientas de creación y edición XML .....</b>	161
4.4.1. XML Copy Editor .....	162
4.4.2. Altova XMLSpy .....	162
4.4.3. XMLSpear .....	162
4.4.4. XML Notepad .....	163
4.4.5. Online XML Editor .....	163
<b>4.5. Validación .....</b>	163
<b>Resumen .....</b>	166
<b>Ejercicios prácticos resueltos .....</b>	169
<b>Ejercicios prácticos .....</b>	172
<b>Autoevaluación .....</b>	176
<b>5. CONVERSIÓN Y ADAPTACIÓN DE DOCUMENTOS XML .....</b>	177
<b>Objetivos .....</b>	177
<b>Mapa conceptual .....</b>	178
<b>Glosario .....</b>	178
<b>5.1. Introducción .....</b>	178
<b>5.2. Técnicas de transformación de documentos XML .....</b>	180
5.2.1. Diferencias entre las versiones de XSLT 1.0, 2.0 y 3.0 .....	181
<b>5.3. Formatos de salida y ámbitos de aplicación .....</b>	182
<b>5.4. Descripción de la estructura y la sintaxis .....</b>	182
5.4.1. Fichero XML .....	182
5.4.2. Hoja de estilo XSL .....	183
<b>5.5. Utilización de plantillas .....</b>	184
5.5.1. <xsl:value-of> .....	185
5.5.2. <xsl:for-each> .....	185
5.5.3. Filtrando contenido .....	187
5.5.4. Ordenando el contenido .....	187
5.5.5. Selección avanzada IF .....	188
5.5.6. Selección avanzada CHOOSE .....	188
<b>5.6. Utilización de herramientas de procesamiento: verificación, depuración y generación de documentación .....</b>	189
5.6.1. Oxygen XML Editor .....	190
5.6.2. Altova Editor XSLT .....	191
5.6.3. Saxon .....	191

5.6.4. XPontus XML Editor .....	192
5.6.5. Herramientas online .....	192
<b>Resumen .....</b>	<b>193</b>
<b>Ejercicios prácticos resueltos .....</b>	<b>194</b>
<b>Ejercicios prácticos .....</b>	<b>197</b>
<b>Autoevaluación .....</b>	<b>198</b>
 <b>6. ALMACENAMIENTO DE LA INFORMACIÓN .....</b>	<b>199</b>
<b>Objetivos .....</b>	<b>199</b>
<b>Mapa conceptual .....</b>	<b>200</b>
<b>Glosario .....</b>	<b>200</b>
<b>6.1. Introducción .....</b>	<b>200</b>
<b>6.2. Sistemas de almacenamiento de información .....</b>	<b>201</b>
6.2.1. Ficheros .....	201
6.2.2. Bases de datos XML habilitadas o <i>enabled</i> .....	202
6.2.3. Bases de datos XML nativas .....	202
<b>6.3. Inserción y extracción de información en XML .....</b>	<b>203</b>
<b>6.4. Búsqueda de información en documentos XML: lenguajes de consulta y manipulación .....</b>	<b>204</b>
6.4.1. XPath .....	204
6.4.2. XQuery .....	211
<b>6.5. Almacenamiento XML nativo .....</b>	<b>212</b>
6.5.1. Instalación y ejecución de una base de datos XML nativa: eXist-db .....	212
6.5.2. BaseX .....	213
<b>6.6. Herramientas de tratamiento y almacenamiento de información en formato XML .....</b>	<b>214</b>
<b>Resumen .....</b>	<b>215</b>
<b>Ejercicios prácticos resueltos .....</b>	<b>216</b>
<b>Ejercicios prácticos .....</b>	<b>217</b>
<b>Autoevaluación .....</b>	<b>217</b>
 <b>7. SISTEMAS DE GESTIÓN DE LA INFORMACIÓN .....</b>	<b>219</b>
<b>Objetivos .....</b>	<b>219</b>
<b>Mapa conceptual .....</b>	<b>220</b>
<b>Glosario .....</b>	<b>220</b>
<b>7.1. Introducción .....</b>	<b>221</b>
7.1.1. Los CRM .....	221
<b>7.2. ERP .....</b>	<b>223</b>
7.2.1. Integración de módulos .....	225
7.2.2. Implantación, adaptación y configuración .....	231
7.2.3. Importación y exportación de información: generación de informes .....	232
<b>7.3. Minería de datos .....</b>	<b>233</b>
7.3.1. El proceso de descubrimiento de conocimiento y la minería de datos .....	234
<b>7.4. OLAP .....</b>	<b>235</b>
<b>7.5. Dashboardy scorecard: business intelligence e informes .....</b>	<b>237</b>
7.5.1. La evolución de los informes .....	237
<b>7.6. Seguridad en sistemas de gestión empresarial .....</b>	<b>237</b>
<b>Resumen .....</b>	<b>238</b>
<b>Ejercicios prácticos .....</b>	<b>240</b>
<b>Autoevaluación .....</b>	<b>240</b>

# Presentación

El módulo Lenguajes de Marcas y Sistemas de Gestión de Información es clave y por ello es el único que está presente en los tres ciclos de grado superior actualmente existentes en la familia profesional de Informática y Comunicaciones: Desarrollo de Aplicaciones Web (DAW), Desarrollo de Aplicaciones Multiplataforma (DAM) y Administración de Sistemas Informáticos en Red (ASIR). La razón es que cualquier profesional de la informática debe tener claros los conocimientos sobre lenguajes de marcas, hojas de estilo, XML o ERP, entre otros. Además, complementa los módulos de programación de DAM y DAW, tratando conceptos que el futuro programador necesitará en su práctica profesional.

Aparte de los ciclos de grado superior, este libro servirá para complementar el temario de algunos certificados de profesionalidad relacionados con la tecnología web como pueden ser:

- Confección y Publicación de Páginas Web.
- Desarrollo de Aplicaciones con Tecnologías Web.
- Sistemas de Gestión de Información.

El contenido de esta obra sirve como base para cualquier tipo de programación web y se tratan conceptos que complementan la formación de cualquier programador como:

- a) *Características de los lenguajes de marcas.* Poniendo énfasis en los lenguajes XML y HTML.
- b) *Lenguaje de marcas HTML con sus etiquetas y atributos.* También se estudiará el XHTML y se analizarán las ventajas y diferencias frente al HTML.
- c) *Hojas de estilo.* Fundamentales a la hora de crear una web, estudiando su estructura, sintaxis, reglas, selectores, propiedades, etc.
- d) *Sindicación de contenidos.* Fundamental a la hora de estar informado de las últimas novedades de los canales RSS de los sitios web mediante el uso de agregadores.
- e) *Definición de esquemas y vocabularios en XML.* Se estudiará este lenguaje de marcas en profundidad.

- f) *Conversiones y adaptaciones de documentos XML.* Para poder generar, por ejemplo, otros ficheros XML o HTML.
- g) *Almacenamiento de información en bases de datos no SQL, como son las bases de datos XML.* Se analizarán las bases de datos nativas y no nativas. El lector encontrará muchos ejemplos en lenguajes de consulta como son XQuery y XPath.
- h) *Sistemas de gestión empresarial poniendo énfasis en los ERP.* Se analizarán las características, los módulos más importantes, la minería de datos, los sistemas OLAP, dashboard, scorecard y la seguridad, etcétera.

Cuando alguien lee el título *Lenguajes de marcas y sistemas de gestión de información*, podría creer que se trata de un libro sobre HTML y hojas de estilo. Sin embargo, se desarrollan otros conceptos muy interesantes, como el XML, en muchos casos un gran desconocido, o las bases de datos XML, con sus respectivos lenguajes de consulta (XPath y XQuery).

El alumnado encontrará en esta obra una exposición clara, didáctica y con un enfoque muy práctico, de los contenidos del módulo del mismo nombre. Incluye multitud de recursos que le ayudarán a asimilar los contenidos, como ejercicios resueltos, ejemplos, tutoriales, prácticas guiadas, etc., accesibles en la edición digital a través del icono ►, así como una serie de actividades propuestas en las que podrá verificar qué ha aprendido ✓.

Para todo aquel profesional que ya posea conceptos de programación web, este libro será una ayuda indispensable si quiere ampliar sus conocimientos sobre XML, transformaciones XML y sistemas de gestión empresarial, entre otros.

No queremos terminar esta presentación sin animar al lector a que trabaje con ahínco todos los temas y conceptos del libro, puesto que la contraprestación que conseguirá es alta. No a corto plazo, pero sí a largo, ya que el conocimiento que se adquiera va a utilizarse durante toda su carrera profesional.

# Reconocimiento de las características de lenguajes de marcas

## Objetivos

- ✓ Iniciarte en el mundo de los lenguajes de marcas, identificando sus características más generales, reconociendo las ventajas que proporcionan, estudiando su clasificación y estructura e identificando los más relevantes.
- ✓ Como cada vez son más las herramientas de desarrollo que permiten crear, gestionar o manipular este tipo de documentos, es de vital importancia que conozcas el ámbito de aplicación de estos lenguajes y que, gracias a la inclusión de ejemplos reales de su uso, comprendas la necesidad de su estudio.
- ✓ Por otro lado, al ser este el primer capítulo del libro, se pretende que, mediante los distintos ejemplos y ejercicios resueltos, el contenido teórico sea más ameno, lo que te ayudará en capítulos sucesivos y te servirá como pilar a la hora de ir trabajando con los lenguajes de marcas.

## Mapa conceptual



## Glosario

**ASCII.** Sigla inglesa que significa “código estándar estadounidense para el intercambio de información”. Código de caracteres basado en el alfabeto latino creado por el Comité Estadounidense de Estándares en 1963.

**C# o C Sharp.** Lenguaje de programación de Microsoft para la plataforma .NET, de sintaxis similar a C++ o Java.

**Java EE.** Conocido como *Java Enterprise Edition*, es una plataforma de programación que se usa para el desarrollo y ejecución de aplicaciones.

**JavaScript.** Lenguaje de programación interpretado y poco tipado, usado mayormente en el lado del cliente junto con documentos HTML, haciéndolos dinámicos y mejorando su interfaz.

**RTF.** Formato de texto enriquecido desarrollado por Microsoft.

**TeX.** Sistema topográfico usado por los matemáticos, físicos e informáticos de distribución y modificación libre escrito por Donald E. Kunth.

**Unicode.** Estándar de codificación de caracteres mantenido por el UCT, que asocia a cada carácter un identificador numérico.

**Visual Studio.** Entorno de desarrollo que soporta los lenguajes C++, Java, C#, ASP.NET, etc.

**W3C (World Wide Web Consortium).** Consorcio internacional creado en 1994 para generar estándares relacionados con la red informática mundial (www).

**XMP.** Protocolo extensible de mensajería y comunicación de presencia que está basado en XML y se emplea en mensajería instantánea.

## 1.1. Introducción

El uso de internet por parte de los usuarios está creciendo en los últimos años. La gran mayoría de los internautas navegan e intercambian información a través de la red, manejando las páginas web que internamente hacen uso de los lenguajes de marcas y permitiendo que todo fluya con normalidad, proceso que es transparente para los usuarios.

Algunos de los programas instalados en el ordenador personal usan lenguajes de marcas, ya sea para almacenar información, configuración de la aplicación, etc.

Las marcas pueden representarse con distintos caracteres, como en TeX, que se usa la barra invertida para el inicio de una marca. Las más usadas son las que se expresan entre el signo menor que (<) y mayor que (>).

En los ejemplos siguientes, se aprecian dos etiquetas de marcas con valores contenidos en ellas: XML (*extensible markup language*) y HTML (*hypertext markup language*).

Etiqueta de marca XML <nombre> con el valor “Litchi”:

```
<nombre>Litchi</nombre>
```

Etiqueta de marca HTML de cabecera <h1> con el valor “Hola Mundo”:

```
<h1>Hola Mundo</h1>
```

No puede compararse un lenguaje de marcas con un lenguaje de programación como C#, Java o Pascal, pero sí permite combinar dentro de un documento lenguajes como PHP (*hyper-text preprocessor*) o JavaScript, aportando funcionalidad, tal y como hace HTML.



SABÍAS QUE...

PHP es un lenguaje de programación usado para contenido dinámico y se ejecuta en el lado servidor. Es libre, abierto y con una curva de aprendizaje baja.

El origen de estos lenguajes radica en el lenguaje de marcado generalizado estándar SGML (*standard generalized markup language*), un estándar que permite definir lenguajes de marcado, considerándose un metalenguaje. Es descendiente de GML (*generalized markup language*) de IBM. En el año 1986, el GML se convirtió en SGML (*standard generalized markup language*), lo que dio lugar a la ISO 8879.

A principios de los años noventa, Tim Berners-Lee desarrolla HTML, que permite visualizar la información de las páginas web en los distintos navegadores. Dos de los lenguajes de marcas que se verán en próximos capítulos son HTML y XML.

Una herramienta fundamental que se usa para visualizar documentos XML, HTML, XHTML, etc., es el *navegador web* o *browser* en inglés. Son aplicaciones instaladas en distintos dispositivos cuya funcionalidad es dar acceso a la web, mostrando a los usuarios la interpretación de

los lenguajes de marcas. En los últimos años, han evolucionado a gran ritmo, incorporando multitud de nuevas funciones.

### RECUERDA

- ✓ Un navegador no es lo mismo que un buscador.

Hoy día, se usan para consultar el correo electrónico y las páginas web, realizar búsquedas, acceder a la configuración de un *router*, etc.

### Navegadores más usados

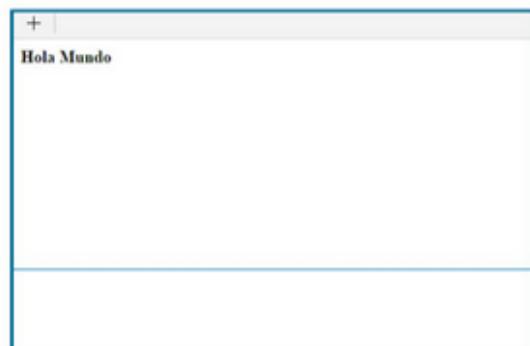


### Investiga

¿Cuántos navegadores tienes instalados en tu ordenador y qué versiones? ¿Cuántos navegadores conoces? Busca al menos cinco navegadores que no tengas en el ordenador y procede a instalar uno de ellos.

En el siguiente ejemplo se observa la estructura de un documento HTML que, al abrirlo en el navegador, muestra el mensaje “Hola Mundo”. Dicho documento puede crearse con el Bloc de notas y ha de tener la extensión .html. Las etiquetas o elementos que se aprecian en este ejemplo se verán con más detalle en el capítulo 2.

```
<html>
<head>
    <title>Página Hola Mundo</title>
</head>
<body>
    <p>
        <b>Hola Mundo</b>
    </p>
</body>
</html>
```



**Figura 1.1**  
Ejecución de código HTML  
en un navegador.

La ejecución del ejemplo da como resultado la figura 1.1, donde se aprecia que ninguna de las marcas que se han creado en el documento se muestra en el navegador. Para poder visualizar el contenido, simplemente hay que abrir el documento HTML con un navegador que se tenga instalado en el ordenador y, automáticamente, se visualizará el resultado.

## 1.2. Clasificación y características comunes de los lenguajes de marcas

En este apartado, se verá una clasificación de los lenguajes de marcas atendiendo a tres tipos distintos y mostrando ejemplos de los más usados, así como describiendo las características comunes que poseen.

### 1.2.1. Clasificación de los lenguajes de marcas

El uso de los lenguajes de marcas es muy diverso. Se utilizan para mensajería instantánea XMPP (*extensible messaging and presence protocol*), servicios web WSDL (*web services description language*), SOAP (*simple object access protocol*), páginas web HTML, WML (*wireless markup language*), sindicación de contenidos RSS (*really simple syndication*), Atom, gráficos vectoriales X3D, documentación electrónica RTF (*rich text format*), TeX, wikitexto, etc.

#### Investiga



Busca información sobre el lenguaje DocBook.

Los lenguajes de marcas suelen clasificarse en tres tipos:

- *Tipo 1. De procedimiento.* Suele emplearse para la presentación del texto, siendo visible para el usuario. Pueden usarse etiquetas para poner un título centrado, reducir el tamaño de letra, poner el texto cursivo, etc. Ejemplos de este tipo de marcado son TeX, Nroff y LaTeX.



#### SABÍAS QUE...

LaTeX es un sistema de composición de textos para crear documentos de alta calidad.

- *Tipo 2. De presentación.* Se refiere al que define el formato del texto. Aunque fácil de crear, resulta complicado de mantener o modificar. Las etiquetas de marcado, por lo general, no son visibles a los usuarios. Uno de sus usos puede ser la maquetación de documentos para que el lector pueda leerlos correctamente. Por ejemplo, Microsoft Word.

#### Ejercicio propuesto 1.1



Abre Microsoft Word y activa las marcas o caracteres ocultos.

Ayuda ➔

- *Tipo 3. Descriptivos o semánticos.* Es un marcado flexible que usa etiquetas sin especificar la manera de representarlas ni el orden. Las marcas dan información sobre su estructura y una descripción del contenido. Algunos de estos lenguajes son SGML, HTML y XML.

En el ejemplo se observa una serie de etiquetas creadas para un documento XML que almacena información relacionada con los exámenes de distintas materias de un ciclo. En este caso concreto, se han creado dos materias de las que se almacenan el nombre, el número de alumnos aprobados y suspensos y las unidades de cada una.

```
<?xml version="1.0" encoding="iso-8859-1"?>
<exámenes>
    <materia>
        <nombre>Lenguaje de marcas</nombre>
        <aprobados>15</aprobados>
        <suspensos>10</suspensos>
        <unidades>4</unidades>
    </materia>
    <materia>
        <nombre>sistemas operativos</nombre>
        <aprobados>5</aprobados>
        <suspensos>20</suspensos>
        <unidades>5</unidades>
    </materia>
</exámenes>
```

### Ejercicio propuesto 1.2



Viendo el ejemplo anterior y analizando su estructura, ¿cómo añadirías otra materia al documento XML, teniendo en cuenta que las dos materias existentes son lenguaje de marcas y sistemas operativos?

#### 1.2.2. Características comunes

No todos los lenguajes de marcas comparten las mismas características debido a la gran cantidad que puede encontrarse en el mercado actual. En este apartado, se tratarán las características comunes más importantes de estos lenguajes como puede ser el uso de texto plano, flexibilidad, interoperabilidad, etc.

##### A) *Texto plano (plain text)*

Este tipo de documentos, también llamado *texto sin formato o simple*, no permite almacenar información con formato (color, tipo de letra, negrita, tamaño, subrayados, etc.) tal y como

puede hacerlo un procesador de texto avanzado. Están formados exclusivamente por caracteres (letras, números, caracteres especiales y de control).



#### SABÍAS QUE...

El espacio en blanco, el retorno de carro y las tabulaciones se consideran caracteres.

Uno de los editores que puede encontrarse en Windows para crear un documento de texto plano es el Bloc de notas (Notepad), cuya extensión es .txt, mientras que, si se usa Linux, Gedit, Vi o Nano pueden ser los más usados.

Tal y como se ha mencionado anteriormente, al estar formado por caracteres, se emplea una codificación específica que suele ser ASCII, ISO 8859-1, Unicode o UTF-8. Estos estándares de codificación permiten representar los distintos caracteres en todos los idiomas. ISO 8859-1 se usa para la codificación de alfabeto latino (ñ, letras acentuadas, etc.) En el ejercicio resuelto 1.1, se verá que cada carácter que se escribe en un documento de texto plano ocupa un *byte* o, lo que es lo mismo, 8 bits. Hay que recordar que el espacio en blanco también se considera un carácter.

#### Ejercicio resuelto 1.1



Abre el Bloc de notas de Windows y escribe la siguiente frase: "alumno LM". ¿Cuántos bytes ocupa?

**Solución**

#### B) Interoperabilidad o independencia

Se considera el texto plano como formato universal, ya que puede abrirse y editarse desde cualquier máquina, aunque ha de tenerse en cuenta la codificación empleada. Son independientes a la plataforma usada y a cualquier sistema de representación.

#### Ejercicio propuesto 1.3



Crea un documento TXT con el Bloc de notas en Windows e intenta abrirlo en un sistema operativo distinto.

#### C) Flexibles y fáciles de crear

Simplemente, se necesita un editor de texto para poder crearlos y guardarlos en la extensión que se desee. Algunos de ellos permiten combinarse con otro lenguaje para darle mayor funcionalidad.

### Actividades propuestas



Responde si las siguientes afirmaciones son verdaderas o falsas y razona tu respuesta:

- V F 1.1. El valor de la etiqueta del siguiente ejemplo es "autor":

```
<autor>alumno</autor>
```

- V F 1.2. Al abrir la siguiente página web en el navegador, aparecerán por pantalla todas estas etiquetas

```
<html>
  <head>
    </head>
  <body>
    Hola Mundo
  </body>
</html>
```

Verificar



### 1.3. Identificación de ámbitos de aplicación

El ámbito de aplicación de los lenguajes de marcas es muy diverso, ya que permite el intercambio de datos entre distintas aplicaciones, independientemente de la plataforma usada y la tecnología en la que estén creadas. Desde un fichero XML, pueden generarse vistas como HTML, WML o PDF. Java EE usa ficheros XML para poder especificar datos de configuración. Visual Studio, a la hora de crear servicios web, genera varios documentos con estructura XML. Tanto Windows Phone como Android Studio usan esta estructura para guardar las vistas. En general, existe una gran cantidad de *software* que usa datos en formato XML para configuración o guardar información. En el mundo real, puede usarse para bases de datos, *frameworks* de desarrollo, sistemas de publicación de contenidos, definición de interfaces gráficas, etc.

### Ejemplos de aplicación de los lenguajes de marcas



### 1.4. XML: estructura y sintaxis

XML, sigla de *extensible markup language* (lenguaje de marcas extensibles), se considera un metalenguaje y surge para resolver los problemas que plantea HTML, al mostrar las etiquetas el significado de sus datos. Una de las ventajas es que no se requieren conocimientos de programación para poder crear o modificar un documento sencillo. La importancia radica en el intercambio de información de manera segura entre distintos programas, lo que permite la reutilización de contenido, crear etiquetas propias y presentar una estructura y diseño independientes.

Aunque este apartado se estudiará con más profundidad en el capítulo 4, se mostrará una introducción a su estructura y sintaxis para ir asimilando la terminología empleada.

Todo documento XML tiene estructura jerárquica arborescente y está compuesto por dos partes fundamentales: *prüfago*, que será la primera parte del documento y estará compuesto por una o varias líneas, y el *cuerpo*.

En este ejemplo el prílogo estaría compuesto por la primera línea donde se declara la versión XML, así como la codificación empleada para representar los caracteres, mientras que el resto del documento, el cuerpo, estaría compuesto por elementos anidados. Dentro de la etiqueta `<cine>`, se incluyen las distintas películas de la cartelera. En este caso, se ha introducido una película y se guarda información relativa al director, la duración y el título.

```
<?xml version="1.0" encoding="iso-8859-1"?>
<cine>
  <película>
    <director>Señor X</director>
    <duración>60</duración>
    <título> Las aventuras de los SGML</título>
  </película>
</cine>
```

**RECUERDA**

- ✓ ISO 8859-1 permite el uso de la letra ñ y de los acentos.

Las etiquetas van encerradas entre el signo menor que (`<`) y mayor que (`>`) y suelen ir emparejadas, una de apertura y otra de cierre. También existen etiquetas vacías, tal y como se aprecia en el ejemplo siguiente donde existe en la primera línea una etiqueta de apertura y cierre para el elemento `email` y una etiqueta vacía para el elemento `casado` con un atributo `v` con valor “no”. Los atributos añaden propiedades a los elementos y van dentro de la etiqueta de apertura.

```
<email>666777888@prueba.com</email>
<casado v="no" />
```

**Actividades propuestas**

Responde si las siguientes afirmaciones son verdaderas o falsas y razona tu respuesta:

- V F 1.3.** Todo documento XML tiene estructura jerárquica arborescente.
- V F 1.4.** Las etiquetas XML van encerradas entre corchetes.

Verificar

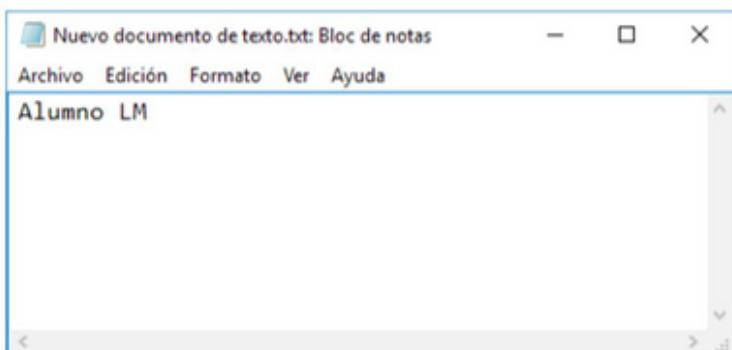
## 1.5. Herramientas de edición

En este apartado, se tratan las herramientas básicas relacionadas con la edición de páginas web, así como otras para la creación de documentos XML, estudiando algunas de mayor prestación en capítulos posteriores. Al igual que para otro tipo de *software*, pueden encontrarse tanto herramientas *offline*, que tienen que descargarse e instalarse en el ordenador personal para poder usarlas, y *online*, donde el único requisito es disponer de una conexión a internet y el uso de un navegador para acceder.

### 1.5.1. Bloc de notas Windows

El Bloc de notas de Windows (Notepad) es un editor de texto que viene con los sistemas operativos Windows que permite crear documentos de texto plano. La extensión de dichos documentos es .txt y permite realizar la codificación de caracteres en ANSI, Unicode, UTF-8.

#### Opciones del Bloc de notas de Windows



**Figura 1.2**  
Bloc de notas de Windows.

#### Ejercicio resuelto 1.2



Abre el Bloc de notas de Windows e introduce cinco nombres de plantas que conozcas. Una vez introducidos, guarda el documento en el escritorio con el nombre "plantas.txt".

#### Solución

#### Ejercicio propuesto 1.4

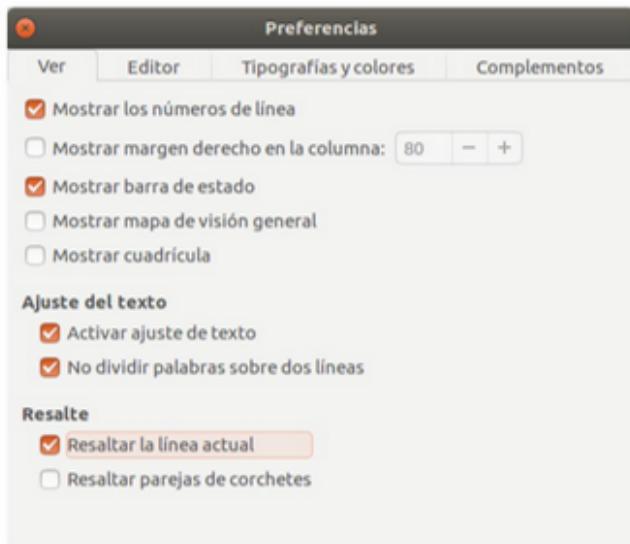


Crea un nuevo documento de texto con el Bloc de notas y escribe dos párrafos. Guarda el fichero en el escritorio y comprueba que la extensión es .txt.

#### Ayuda

### 1.5.2. Gedit

Gedit es un editor de texto de fácil uso para el entorno de escritorio GNOME, aunque también está disponible para Windows. Suele emplearse como simple bloc de notas, utilizar las características avanzadas y usarlo como entorno de desarrollo. Además de las características generales que puede tener cualquier editor de texto como copiar, guardar, abrir, imprimir, etc., permite el coloreado de sintaxis dependiendo del lenguaje de programación usado, así como la activación de la numeración de líneas, accediendo al menú Editar → Preferencias → Ver y marcando la casilla “Mostrar los números de línea”.



**Figura 1.3**  
Activación del número de líneas en Gedit de Ubuntu.

#### Ejercicio resuelto 1.3

Usando el sistema operativo Linux Ubuntu, crea un documento de texto plano con Gedit, cuyo contenido sea el nombre de tres animales, y guarda el fichero en el escritorio.

**Solución**

### 1.5.3. Notepad++

Es un potente editor de texto gratuito de licencia GPL para sistemas operativos Windows. Está escrito en C++ usando la API Win32. Permite seleccionar multitud de lenguajes de programación. Organiza el texto según el lenguaje seleccionado y muestra, entre otros aspectos, numeración de líneas, color de texto en etiquetas según la sintaxis, aumento de tamaño de letra, etc.

Entre sus características, destacan las siguientes: permite ampliar y reducir texto, autocompletar, búsqueda y reemplazo, y resaltado de sintaxis.

#### Selección del lenguaje en Notepad++

WWW

#### Recurso web

Página web de Notepad++.



## 1.6. Elaboración de documentos XML bien formados

Para poder elaborar un documento XML bien formado, es necesario seguir una serie de reglas, entre las que destacan:

- El W3C recomienda empezar por la instrucción donde se indique la versión (instrucción de procesamiento), por lo que esta ha de ser siempre la primera instrucción.
- Todo documento ha de tener una estructura jerárquica, en la que exista un único elemento raíz o principal, que será el primero que se abra y el último que se cierre.
- Las etiquetas tienen que estar anidadas correctamente unas dentro de otras, cerrándose en orden inverso al que se abren. Los elementos no vacíos tendrán etiquetas de apertura y cierre.
- Dentro de los atributos, los valores tienen que estar encerrados entre comillas dobles o simples y no podrá haber dos atributos que se llamen igual en un mismo elemento.
- A la hora de escribir las etiquetas, elementos o atributos, hay que tener en cuenta que es sensible a las mayúsculas y minúsculas. Por ejemplo, un XML de apertura de la etiqueta <casa> y uno de cierre </Casa> son distintas, ya que una empieza por mayúsculas y otra por minúsculas.
- Los nombres de etiquetas, elementos y atributos tienen que seguir una nomenclatura específica como no empezar por número, no usar caracteres especiales reservados, etc. Tienen que empezar por letras, seguidos de guiones, puntos, números u otras letras. Tampoco pueden empezar por la palabra XML.
- Los comentarios no pueden ir dentro de las etiquetas.

A continuación, se muestra un documento bien formado.

```
<?xml version="1.0" encoding="utf-8" ?>
<agenda>
    <!--vamos a crear un contacto-->
    <contacto>
        <nombre> AlumnoLM </nombre>
        <ciudad cp="29000">Málaga</ciudad>
        <teléfono n="666777888" />
        <email>666777888@prueba.com</email>
        <casado v="no" />
    </contacto>
</agenda>
```

Todo lo expuesto en este apartado puede verificarse mediante el uso de herramientas específicas, así como con un simple navegador. Si quiere saberse si un documento XML está bien formado, lo primero que hay que hacer es crearlo con el editor que se desee, guardarlo con extensión XML y abrirlo con un navegador, donde se muestra la estructura jerárquica en forma de árbol con los nodos. Si no se muestra dicha estructura, significa que el documento XML no está bien formado.

En la figura 1.4, se muestra la vista del documento XML en el navegador.

Este fichero XML no parece tener ninguna información de estilo asociada. Se muestra debajo el árbol del documento.

```
- <agenda>
  <!--vamos a crear un contacto-->
  - <contacto>
    <nombre> AlumnoLM </nombre>
    <ciudad cp="29000">Málaga</ciudad>
    <teléfono n="666777888"/>
    <email>666777888@prueba.com</email>
    <casado v="no"/>
  </contacto>
</agenda>
```

**Figura 1.4**  
Visualización del documento XML en el navegador.

Sobre el mismo documento, van a realizarse algunas modificaciones para ver el resultado que muestra el navegador:

```
<?xml version="1.0" encoding="utf-8" ?>
<agenda>
  <!--vamos a crear un contacto-->
  <contacto>
    <nombre> AlumnoLM </Nombre>
    <ciudad cp="29000">Málaga</ciudad>
    <teléfono n="666777888"/>
    <email>666777888@prueba.com</email>
    <casado v="no" />
  </contacto>
</agenda>
```

Al ejecutarlo en el navegador, aparece la información que se muestra en la figura 1.5, donde se indica que, en la línea 5, la etiqueta <nombre> no está especificada.

**Figura 1.5**  
Error obtenido al abrir el XML en el navegador.

Error de lectura XML: etiqueta sin pareja. Se esperaba: </nombre>. Ubicación: file:///C:/Users/Sergio/Desktop/XMLFile5.xml Número de línea 5, columna 25:

```
<nombre> AlumnoLM </Nombre>
-----^
```

Como puede observarse, en la línea 5, se ha añadido una etiqueta de apertura que empieza por minúsculas y de cierre con mayúsculas. Se arregla y vuelve a comprobarse si existen errores, obteniendo como resultado un fallo en la línea 8 (figura 1.6).

**Figura 1.6**  
Error obtenido al abrir el XML

Error de lectura XML: mal formado Ubicación: file:///C:/Users/Sergio/Desktop/XMLFile5.xml Número de línea 8, columna 6:

```
<email>666777888@prueba.com</email>
-----^
```

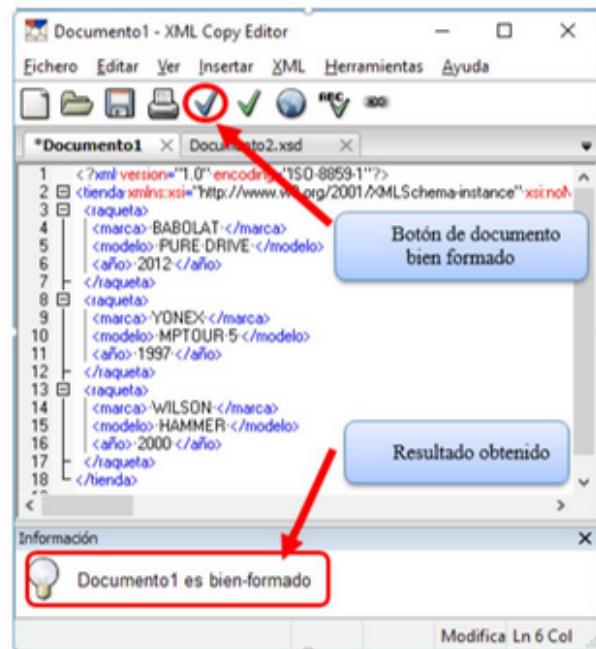
La etiqueta <email> empieza por un número. Se quita dicho número y vuelve a comprobarse si existen errores. Esta vez, se obtiene el resultado correcto.

Otra alternativa es usar XML Copy Editor, que también permite comprobar si el documento está bien formado, donde el mensaje se muestra en la parte inferior de la ventana (figura 1.7).

### Ejercicio propuesto 1.5

Comprueba si el siguiente documento XML está bien formado. Para ello, ábrelo en un navegador que tengas instalado.

```
<?xml version="1.0" encoding="utf-8" ?>
<agenda>
    <!--vamos a crear un contacto-->
    <contactos>
        <nombre> AlumnoLM </nombre>
        <ciudad cp='29000'>Málaga</ciudad>
        <teléfono n="666777888" />
        <Email>666777888@prueba.com</Email>
        <casado v="no" />
    </contacto>
</agenda>
```



**Figura 1.7**  
Comprobación de un documento bien formado con XML Copy Editor.

## 1.7. Utilización de espacios de nombres en XML

Los espacios de nombres son mecanismos que se usan para distinguir etiquetas que se denominen igual al emplear varios documentos XML, evitando de esa manera la ambigüedad. Se declara de la siguiente manera:

```
<elemento xmlns:prefijo="URI">
```

La forma de usarlos es anteponiendo el prefijo a los elementos y atributos de las etiquetas ambiguas:

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

**Ejemplo**

Se dispone de dos documentos XML distintos con un elemento *banco* que aparece tanto en el documento XML1 como en el XML2, pero con estructura y significado distintos. El primer elemento hace referencia a un banco para sentarse, mientras que el segundo, a una entidad bancaria.

Documento XML1:

```
<?xml version="1.0" encoding="iso-8859-1"?>
<banco>
    <color>marrón</color>
    <asientos>3</asientos>
</banco>
```

Documento XML 2:

```
<?xml version="1.0" encoding="iso-8859-1"?>
<banco>
    <nombre>Banco X</nombre>
    <dirección>calle Y</dirección>
    <presidente>Mr. X</presidente>
    <clientes> 154000000</clientes>
</banco>
```

Si quieren usarse los elementos de los dos XML en otro documento, se actúa de la siguiente manera, definiendo dos espacios de nombres de prefijos b1 y b2:

```
<?xml version="1.0" encoding="iso-8859-1"?>
<b1:prueba xmlns:b1="http://...documentoXML1"
             xmlns:b2="http://...documentoXML2">
    <b1:banco>
        <b1:color>marrón</b1:color>
        <b1:asientos>3</b1:asientos>
    </b1:banco>
    <b2:banco>
        <b2:nombre>Banco X</b2:nombre>
        <b2:dirección>calle Y</b2:dirección>
        <b2:presidente>Mr. X</b2:presidente>
        <b2:clientes> 154000000</b2:clientes>
    </b2:banco>
</b1:prueba>
```

## Resumen

- Clasificación de los lenguajes de marcas:
  - Tipo 1. De procedimiento.
  - Tipo 2. De presentación.
  - Tipo 3. Descriptivos o semánticos.
- Características comunes:
  - Texto plano (*plain text*).
  - Interoperabilidad o independencia.
  - Flexibles y fáciles de crear.
- Identificación de ámbitos de aplicación:
  - Páginas web.
  - Especificaciones de configuración.
  - Servicios web.
  - Diseño de las interfaces de Windows Phone y Android Studio.
- XML (estructura y sintaxis):
  - Prólogo.
  - Cuerpo.
- Herramientas de edición:
  - Bloc de notas de Windows.
  - Gedit.
  - Notepad++.
- Elaboración de documentos XML bien formados:
  - Estructura jerárquica.
  - Único elemento raíz.
  - Etiquetas anidadas correctamente.
  - Los valores de los atributos tienen que estar encerrados entre comillas.
  - Sensible a mayúsculas y minúsculas.
  - Los nombres de etiquetas, elementos y atributos tienen que seguir una nomenclatura específica.
  - Los comentarios no pueden ir dentro de las etiquetas.
- Utilización de espacios de nombres en XML:
  - Declaración:

```
<elemento xmlns:prefijo="URI" >
```

## Ejercicios prácticos



1. Averigua qué otros lenguajes de marcas se usan en la actualidad y nombra algunas de las marcas más empleadas.
2. Crea un documento XML que almacene información relacionada con animales mamíferos.
3. Comprueba si el siguiente documento XML está bien formado:

```
<?xml version="1.0" encoding="utf-8"?>
<padre>
    <hijo>
        <nombre>alumno 1</nombre>
        <apellidos>ap1 ap2</apellidos>
    </hijo>
    <hijo>
        <nombre>alumno 2</nombre>
        <apellidos>ap11 ap21</apellidos>
    </hijo>
</padre>
```

4. Realiza el proceso de instalación con capturas de pantalla de otra herramienta de edición y especifica algunas de sus características.
5. Dado el siguiente fichero XML, introduce otro libro y especifica el elemento raíz del ejemplo. Una vez añadido, comprueba si está bien formado.

```
<?xml version="1.0" encoding="utf-8"?>
<biblioteca>
    <libro>
        <título>lenguajes de marcas</título>
        <autor>Autor 1</autor>
    </libro>
</biblioteca>
```

AUTOEVALUACIÓN



# Utilización de lenguajes de marcas en entornos web

## Objetivos

- ✓ Introducirte en el uso de los lenguajes de marcas en entorno web, identificando los relacionados con la web y sus diferentes versiones.
- ✓ Analizar la estructura de un documento HTML e identificar las secciones que lo componen serán los primeros aspectos que verás, junto con el estudio de distintas herramientas para la creación de estos documentos.
- ✓ El uso de los elementos de este lenguaje es de vital importancia, por lo que describir la funcionalidad de las principales etiquetas y atributos te ayudará en su comprensión y manejo. Para dotar de estilo a dichas páginas, aprenderás las ventajas que aporta la utilización de hojas de estilo.
- ✓ Para finalizar, identificarás las semejanzas y diferencias semánticas y estructurales entre los lenguajes HTML y XHTM.

## Mapa conceptual



## Glosario

**AJAX.** Sigla de *asynchronous JavaScript and XML*. Técnica de programación web de tecnología asíncrona, creada en el 2005 por Jesse James Garrett, que permite obtener respuestas en segundo plano que vienen del servidor sin tener que recargar la página completa.

**CSS.** Sigla inglesa que significa “hojas de estilo en cascada”. Lenguaje de diseño gráfico que se emplea para establecer el diseño visual de los documentos web.

**OS/2.** Sistema operativo desarrollado por IBM, aunque en sus comienzos también participó Microsoft, cuyas siglas se interpretan como “sistema operativo de segunda generación”.

**SASS.** Sigla de *syntactically awesome style sheets*. Metalenguaje de CSS evolucionado a un lenguaje de script, SassScript.

### 2.1. Introducción

Cuando se usan los navegadores para consultar las distintas web, se aprecia una serie de páginas que contienen elementos como enlaces, texto, imágenes, vídeos, etc. Si se accede al código fuente de dicha página, se aprecia que el HTML (*hypertext markup language*) es el componente básico de la web, junto con otras tecnologías entre las que destacan CSS, AJAX, JavaScript, etc.

HTML es un lenguaje de marcado que se usa para la creación de páginas web y tuvo su origen en 1991. A día de hoy, se encuentra disponible el primer documento que se publicó.



SABÍAS QUE...

HTML se define como un lenguaje de etiquetas.

Para poder crear cualquier documento, es necesario disponer de un editor de texto plano. Existen herramientas de escritorio gratuitas y comerciales que facilitan este trabajo. Si no desea instalarse ni usarse ninguna herramienta de escritorio, existen herramientas *online* que permiten crear y editar este tipo de documentos.

De HTML, se estudiará su estructura y sintaxis, así como sus elementos principales (listas, tablas, formato de texto, párrafos, cabeceras, formularios, etc.).

Elaborar una web sin un diseño atractivo es algo impensable. En este punto, es cuando entran en juego las hojas de estilo en cascada (CSS), que describen la manera de visualizar una página web por pantalla.

## 2.2. Herramientas de diseño web

En este apartado, se dará un repaso a las herramientas de diseño web más utilizadas por los desarrolladores.

### 2.2.1. KompoZer

*Software* para el diseño web de código abierto y de fácil uso que permite la administración de un sitio FTP, así como personalizar la barra de herramientas. Trae integrado un editor de CSS y un validador HTML. Disponible en varios idiomas, permite la descarga para sistemas operativos Windows, Linux, OS/2 y Mac OSX.

### 2.2.2. Adobe Dreamweaver CC

*Software* comercial específico para diseñar y desarrollar sitios web basados en estándares con una interfaz moderna.

Entre sus características, destacan las siguientes: permite la previsualización en varios dispositivos; arquitectura 64 bits; compatible con PHP 7, JavaScript, GIT y CEF, y trae un espacio de trabajo para desarrolladores, que es compatible con procesadores de CSS como SASS, LESS y SCSS.

Fue creado inicialmente por macromedia y la empresa que lo proporciona actualmente es Adobe Systems.

WWW

### Recursos web

A través de estos enlaces podrás acceder a la página web de KompoZer, descargar la versión de prueba de Adobe Dreamweaver CC, descargar Bluefish y usar HTML Online Editor.



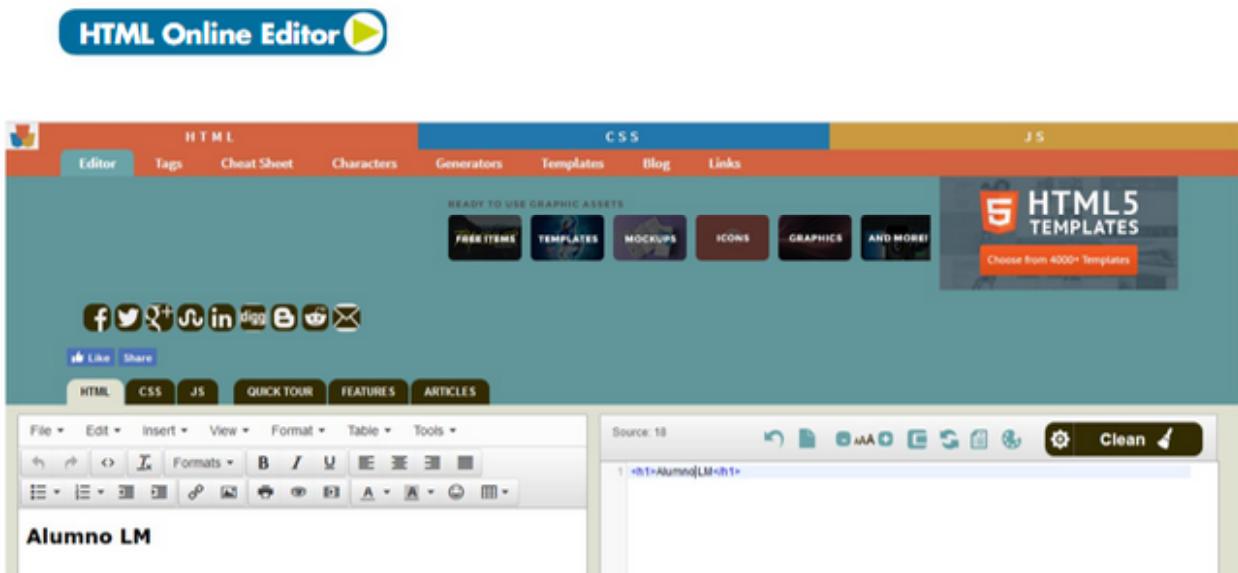
### 2.2.3. Bluefish

Bluefish es un editor web *open source* multiplataforma con licencia GNU. Aunque las primeras versiones estaban plagadas de fallos y su funcionalidad era muy justa, las versiones actuales lo hacen una alternativa fiable a las plataformas de pago.

### 2.2.4. HTML Online Editor

Programa de editor HTML WYSIWYG gratuito que permite editar código fuente en línea sin descargar ninguna aplicación. Permite trabajar con documentos HTML, CSS y JS.

El funcionamiento es muy intuitivo. En el panel derecho, el usuario va introduciendo las líneas de código HTML y, de forma concurrente, el resultado aparecerá en el panel izquierdo. También puede trabajarse sobre el documento de forma visual y el código HTML asociado aparecerá en el panel derecho.



**Figura 2.1**  
Interfaz de HTML Online Editor.

## 2.3. Identificación de etiquetas y atributos de HTML

En este apartado del capítulo, se describe cuál es la estructura del lenguaje HTML y sus etiquetas, aspecto básico en la programación de páginas web. HTML es un lenguaje de marcado que va a ser la base del módulo Lenguajes de Marcas y Sistemas de Gestión de Información.

### 2.3.1. Estructura de un documento HTML

La estructura principal de una página web está compuesta por una serie de elementos dentro de un documento de texto plano que dan un formato al contenido que lo forma. Dicho contenido es interpretado por el navegador, que muestra el resultado final.

Los navegadores también permiten ir inspeccionando el código conforme aparece el resultado. La estructura principal de una página web es la siguiente:

```
<html>
  <head>
    <title>LM</title>
  </head>
  <body>
    <!-- contenido -->
  </body>
</html>
```

La etiqueta primera de un documento es `<html>`, que define el inicio del fichero, por lo que `</html>` es la última que tiene que escribirse. Después, se encuentra la etiqueta `<head>`, donde se especifica la cabecera del documento y que puede anidar dentro otras como:

- `<title>`. Título que se muestra en la barra de título del explorador.
- `<meta>`. Sirve para el uso de metadatos y posicionar la web en los buscadores.
- `<style>`. Se utiliza en la definición de estilos.
- `<link>`. Vincula las hojas de estilo como recurso externo.

**RECUERDA**

- ✓ Pueden escribirse las etiquetas en mayúsculas o minúsculas, ya que el navegador las interpretará igual. Una etiqueta `<HTML>` será interpretada de igual manera por el navegador que si se escribe en minúsculas `<html>`.

Por último, se aprecia la cabecera `<body>` o cuerpo, que es donde se define el contenido de una página web visible en los navegadores.

En el ejemplo anterior (estructura principal de una página web), se aprecia un comentario, entre las etiquetas `<!--` y `-->`. El uso de comentarios en este tipo de documentos es muy importante, ya que ayuda a organizar el código y a ocultar texto que ya no se usa.



**SABÍAS QUE...**

En muchos documentos HTML, puede observarse que la primera línea es la declaración del tipo de documento:

```
<!DOCTYPE html...>
```

## A) Elementos HTML

La estructura básica de HTML se denomina *elemento* y está compuesto por dos propiedades: atributos y contenido. Por lo general, tiene una etiqueta de inicio (que puede contener un atributo con su valor) y otra de cierre.

Existen algunas etiquetas sin contenidos o vacías que no necesitan de etiqueta de cierre como, por ejemplo, <br>.

En la figura 2.2, se presenta un ejemplo de elemento HTML.

Etiqueta de inicio	contenido	Etiqueta cierre
<p align="center">	"center"> Hola mundo	</p>
atributo	valor	

**Figura 2.2**  
Elemento HTML.

Algunas de las etiquetas HTML que se utilizan en una página web aparecen reflejadas en la figura 2.3.



**Figura 2.3**  
Etiquetas HTML que se usan en una página.

### 2.3.2. Versiones de HTML

Las versiones de HTML están definidas por la comunidad internacional W3C (*World Wide Web Consortium*). HTML nace en el año 1991, desarrollado por el científico Tim Berners-Lee (conocido como el *padre de la web*) y era denominado como *HTML Tags*.

- *HTML 2.0*. En 1995, el Internet Engineering Task Force (IETF) creó el primer estándar oficial, y, a partir del año siguiente, el W3C se encargó de publicar los estándares de HTML.
- *HTML 3.2*. A principios de 1997, aparece esta versión, publicada por el W3C (incluyendo *applets* de Java, etc.).
- *HTML 4.0*. El 24 de abril de 1998 se publica esta versión, que presenta como novedades: tablas complejas, mejoras en los formularios, hojas de estilo, añadir *script* y mejora de accesibilidad.
- *HTML 4.01*. A finales de 1999, surge HTML 4.01 como actualización y revisión de HTML 4.0, aunque no aporta novedades relevantes.
- *HTML 5*. Publicada a finales del 2014, establece nuevos elementos, algunos similares a los anteriores, pero con un significado semántico.

## A) Declaración de tipo de documento

La declaración de tipo de documento (DTD) añade reglas sintácticas que permite definir el documento con sus reglas y atributos, permitiendo de esta manera comprobar si es válido. Entre la información que puede añadirse, destaca la versión HTML.

Debe declararse al principio del documento.

Para HTML 4.01, existen tres DTD diferentes que incluyen los siguientes campos:

1. *Estricto*. Solo pueden usarse etiquetas HTML 4.01 y no permite marcado de presentación.

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN" "http://www.w3.org/TR/html4/strict.dtd">
```

2. *Transicional*. Permite todos los elementos HTML, incluyendo los antiguos, desaconsejados o en desuso.

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/TR/html4/loose.dtd">
```

3. *Marcos o frames*. Es similar al anterior, pero usa elementos para *frames*. Se ha quedado obsoleto.

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Frameset//EN" "http://www.w3.org/TR/html4/frameset.dtd">
```

Si se analiza cada uno de los campos de una declaración, se observa que consisten en:

- *!DOCTYPE*: se usa esta cabecera para la declaración de tipo. Es la manera que tiene el navegador de saber la versión especificada.
- *HTML*: informa de cuál es el elemento raíz del documento.
- *PUBLIC*: presenta dos posibles valores para este campo, PUBLIC o SYSTEM, donde la primera de ellas indica que la declaración de tipo de documento está disponible de manera pública.
- “*-//W3C//DTD HTML 4.01 Transitional//EN*”: es el identificador público formal, que informa de la organización y contiene información diversa separada por los caracteres de la barra doble (//).
  - -: presenta dos posibles valores (– es una organización no registrada por la ISO, mientras que + es una organización registrada por la ISO).
  - *W3C*: es la organización responsable.
  - *DTD HTML 4.01 Transitional*: indica el tipo de documento que está declarándose.
  - *EN*: indica el idioma de la DTD.
- “*http://www.w3.org/TR/html4/loose.dtd*”>: indica el URL de la versión del DTD.

### 2.3.3. Etiquetas y atributos HTML

Los atributos son elementos compuestos por dos partes claramente diferenciadas (el nombre del atributo y su valor correspondiente) y se añaden en la etiqueta de inicio de un elemento HTML.

La manera de asignar un valor a un atributo es mediante el signo igual (=). El valor puede ponerse con comillas dobles, simples o sin comillas en HTML.

- *Atributo bgcolor.* Atributo para signar color usado en algunas etiquetas. La manera de asignar el color es muy diversa: en hexadecimal, mediante el nombre del color y en RGB.
  - *Hexadecimal:* el valor es “#**xxyyzz**”, siendo ese grupo de letras 3 agrupaciones numéricas con valores hexadecimales (0 1 2 3 4 5 6 7 8 9 A B C D E F). Los valores de baja intensidad se representan con 00, donde FF son los de mayor intensidad.

El color negro se representa con el código #000000, mientras que el blanco es #FFFFFF.

- **xx** es la intensidad de rojo
- **yy** es la intensidad de verde.
- **zz** es la intensidad de azul.

```
<html>
<head>
    <title>LM</title>
</head>
<body bgcolor="#FF0022">
    colores
</body>
</html>
```

**Resultado en el navegador**

#### Ejercicio resuelto 2.1



Asigna valores hexadecimales para los colores rojo, verde y azul.

**Solución**

- Por nombre: red, yellow, blue, brown, silver, gray, black, white, etc.

```
<html>
<head>
    <title>LM</title>
</head>
<body bgcolor="blue">
    colores
</body>
</html>
```

**Resultado en el navegador**

- **RGB:** son colores aditivos o primarios y están basados en la combinación de píxeles de colores rojo (RED), verde (GREEN) y azul (BLUE).
  - Los colores se codifican con un número.
  - Los valores para cada color van desde 0 a 255, por lo tanto, hay 256 valores posibles para cada valor.
  - Con la unión de los tres colores en su máximo valor, se obtiene el color blanco y, con la ausencia de todos, el negro (todos a cero). El blanco será (255,255,255) y el negro será (0,0,0).
  - Permite representar  $256 * 256 * 256$  combinaciones diferentes de colores, con lo que hay 16 millones de colores.
  - Debe tenerse en cuenta que puede que no todos los navegadores admitan esta codificación de colores.

Si se mezclan los tres colores, se obtienen las siguientes combinaciones de colores secundarios:

- Verde + azul = cian.
- Rojo + azul = magenta.
- Rojo + verde = amarillo.
- Rojo + azul + verde = blanco.

```
<html>
<head>
    <title>LM</title>
</head>
<body bgcolor="rgb(0,0,255)">
    colores
</body>
</html>
```

Resultado en el navegador 

### Actividad propuesta 2.1



Señala la respuesta correcta a las siguientes preguntas:

a) ¿Con qué valores para RGB se obtiene el color blanco?

- a) (255,255,255).  
 b) (0,0,0).

b) ¿Con qué valores para RGB se obtiene el color negro?

- a) (255,255,255).  
 b) (0,0,0).

c) ¿Cuál es el valor máximo de cada color?

V    F   a) 255.

V    F   b) 256.

d) ¿Qué color es (255,0,0)?

V    F   a) Rojo.

V    F   b) Verde.

e) ¿Qué color es (0,255,0)?

V    F   a) Rojo.

V    F   b) Verde.

f) ¿Qué color es (0,0,255)?

V    F   a) Rojo.

V    F   b) Azul.

g) ¿Qué color es (128,0,0)?

V    F   a) Rojo claro.

V    F   b) Rojo oscuro.

h) ¿Qué color es (0,128,0)?

V    F   a) Verde claro.

V    F   b) Verde oscuro.

i) ¿Qué color es (0,0,128)?

V    F   a) Azul claro.

V    F   b) Azul oscuro.

Verificar



### Ejercicio resuelto 2.2



Consigue obtener los colores secundarios en RGB.

Solución



### A) Etiquetas de formato de texto, línea horizontal y salto de línea

El formato dentro de un documento es fundamental y más si se habla de ver dicho documento en un navegador web. Para ello, existe una serie de etiquetas HTML que permiten formatear el texto a gusto del usuario, dotando al documento de un diseño más atractivo y legible.

- *Etiqueta <b>*: pone el texto contenido en ella en negrita.
- *Etiqueta <u>*: se usa para subrayar el texto.
- *Etiqueta <i>*: permite poner en cursiva un texto.
- *Etiqueta <s>*: se emplea para tachar una palabra o texto.

### Actividad propuesta 2.2



¿Cuál es la etiqueta HTML para que una palabra aparezca en negrita?

- V     F    a) <s> palabra </s>.
- V     F    b) <b> palabra </b>.
- V     F    c) <n> palabra </n>.

Verificar



Las etiquetas de línea horizontal y salto de línea se emplean mucho en los documentos creados en HTML. La de salto de línea permite algunos atributos respecto al grosor y el color.

- *Etiqueta <br>*: salto de línea.
- *Etiqueta <hr>*: línea horizontal (*horizontal rules*) que permite los atributos *align* para la alineación de la línea horizontal, tomando los valores “left” o “right”, y *width*, que permite modificar el ancho.

```

<html>
<head>
    <title></title>
</head>
<body>
    <b>Ejemplos</b>
    <hr />
    <u>Lenguaje de marcas subrayado</u> <br />
    <b>Lenguaje de marcas negrita</b> <br />
    <s>Lenguaje de marcas tachado</s> <br />
    <i>Lenguaje de marcas cursiva</i><br />
    <!--ahora probamos con línea horizontal y salto
        de línea-->
    <hr width="50%" align="left" />
    Ejemplo XML
    <br>
    <br>
    <br>
    Ejemplo HTML
</body>
</html>

```

En la figura 2.4, se muestra el resultado obtenido en el ejemplo anterior.

**Figura 2.4**  
Resultado en el navegador.

### Ejemplos

Lenguaje de marcas subrayado

**Lenguaje de marcas negrita**

Lenguaje de marcas tachado

*Lenguaje de marcas cursiva*

### Ejemplo XML

### Ejemplo HTML

Etiquetas para crear índices y subíndices:

- *Etiqueta <SUP>*: para el superíndice.
- *Etiqueta <SUB>*: para el subíndice.

```
<html>
<head>
    <title>LM</title>
</head>
<body>
    H<sub>2</sub>O
    <br>
    <hr>
    FNMT<SUB>1
</body>
</html>
```

Resultado en el navegador 

- *Etiqueta <p>*: esta etiqueta se usa para crear párrafos, siendo la de cierre opcional siempre y cuando no se use para alineación del texto. La alineación se realiza con el atributo *align* cuyos valores pueden ser “left”, “center”, “justify”, “right”, aunque existe otra serie de atributos genéricos menos importantes. La etiqueta de apertura <p> actúa como salto de línea, similar a la etiqueta <br>.
- *Etiqueta <big>*: para aumentar texto.
- *Etiqueta <small>*: para disminuir texto.

```
<html>
<head>
    <title>LM</title>
</head>
<body>
    <p>Ejemplo de párrafos</p>
    <hr />
```

[.../...]

```

<p>
<p>
<p align="center">Lenguaje de marcas.</p>
<p>
<p align="right"> Lenguaje de marcas.</p>
<br>
<p>lenguaje de marcas tiene <big>más de 20 alumnos matriculados</big>
</p>
<br >
<p>lenguaje de marcas tiene <small>menos de 20 alumnos matriculados</
    small> </p>
</body>
</html>

```

**Resultado en el navegador**

- *Etiqueta <font>:* se usa para variar el tamaño, el color y la fuente de un texto. Además de los atributos genéricos, se destacan los siguientes:
  - *size:* para el tamaño de la fuente y con un valor que va del 1 al 7, donde 3 es el valor por defecto.
  - *face:* es el que especifica la fuente del texto y pueden colocarse varios nombres separados por coma.
  - *color:* especifica el color del texto.

```

<html>
<head>
    <title>LM</title>
</head>
<body>
    <p>Ejemplo de fuentes</p>
    <hr>
    <font size="6" color="blue" face="Verdana">alumnos LM</font>
</body>
</html>

```

**Resultado en el navegador**

## B) Etiquetas de encabezamientos

Estas etiquetas suelen usarse para posicionamiento web y añadir los títulos de las páginas web “recomendando que sean cortos debido al uso de palabras claves”. Esta etiqueta tiene 6 posibles valores que pueden sustituirse por la letra x, siendo 1 el de mayor tamaño y 6 el de menor <hx> </hx>.

**RECUERDA**

✓ El texto de una etiqueta <h1> es mayor que el de una etiqueta <h2>.

Suelen usarse las etiquetas <h1> y <h2> para el título principal de la página y los otros tamaños menores para subtítulos.

```
<html>
<head>
    <title></title>
</head>
<body>
    <h1>Lenguaje de marcas</h1>
    <h2>Lenguaje de marcas</h2>
    <h3>Lenguaje de marcas</h3>
    <h4>Lenguaje de marcas</h4>
    <h5>Lenguaje de marcas</h5>
    <h6>Lenguaje de marcas</h6>
</body>
</html>
```

**Resultado en el navegador**



### C) Etiquetas de listas

Existen distintos tipos de listas que pueden definirse en HTML, pero entre ellas las más importantes son las siguientes:

- *Etiqueta <ol>: listas numéricas u ordenadas.* Esta es la etiqueta que se usa para este tipo de listas, anidando dentro de ellas los elementos o líneas mediante la etiqueta <li>. Entre los atributos, destacan:
  - *type:* se usa para especificar el estilo de numeración (1,a,i,I,A). Si se omite este atributo, aparecerá como ordenada numéricamente.
  - *start:* se usa para indicar el primer número de la lista. Si no se indica, se entiende que empezará por el número 1.

```
<html>
<head>
    <title>LM</title>
</head>
<body>
    <ol type="A">
        <li>capítulo 1</li>
        <li>capítulo 2</li>
        <li>capítulo 3</li>
        <li>capítulo 4</li>
    </ol>
    <br>
    <hr>
    <ol>
        <li>capítulo primero</li>
        <li>capítulo segundo</li>
        [.../...]
```

- A. capítulo 1  
B. capítulo 2  
C. capítulo 3  
D. capítulo 4

1. capítulo primero  
2. capítulo segundo  
3. capítulo tercero  
4. capítulo cuarto

**Figura 2.5**  
Resultado en el navegador.

```

<li>capítulo tercero</li>
<li>capítulo cuarto</li>
</ol>
</body>
</html>

```

- Etiqueta `<ul>`: *listas desordenadas*. Al igual que las listas ordenadas, usa la etiqueta `<li>` para cada uno de sus elementos. El atributo que ha de destacarse es el siguiente:
- `type`: informa del tipo de viñeta que ha de usarse (“disc”, “square” o “circle”).

```

<html>
<head>
    <title>LM</title>
</head>
<body>
    <p>listas desordenadas</p>
    <hr>
    <ul type="square">
        <li>capítulo 1</li>
        <li>capítulo 2</li>
        <li>capítulo 3</li>
        <li>capítulo 4</li>
    </ul>
</body>
</html>

```

**Resultado en el navegador**

- Etiqueta `<dl>`: *Listas de definición*. Dentro de esta, se usan las etiquetas
- `<dt>`: para cada elemento
- `<dd>`: para realizar una definición de cada elemento.

```

<html>
<head>
    <title>LM</title>
</head>
<body>
    <p>listas definición</p>
    <hr>
    <dl>
        <dt>HTML</dt>
        <dd>
            HyperText Markup Language</dd>
        <dt>CSS</dt>
        <dd>
            Cascading Stylesheets</dd>
    </dl>
</body>
</html>

```

**Resultado en el navegador**

## D) Etiquetas de enlaces

Hoy en día, no se concibe una web sin hipervínculos a otras páginas o a otros apartados de esta, a la dirección de correo o a documentos. Pueden clasificarse en tres tipos: enlaces absolutos (a páginas de otras páginas web), enlaces relativos (a páginas situadas dentro del servidor local) y enlaces dentro de la misma página.

- *Etiqueta <a>*. Anchor (ancla) tiene una de apertura y otra de cierre. Entre dichas etiquetas, puede insertarse un elemento como texto, imagen u otro objeto.

Entre los atributos que posee, destacan:

- *href (hypertext reference)*: el valor que se le asigna suele ser una URL.
- *Name*: se usa para definir una ubicación dentro del documento.
- *Target*: se usa para indicar dónde desplegar la URL. Los valores posibles son:
  - *\_blank*: se usa para abrir el contenido en una nueva ventana o pestaña.
  - *\_self*: es el valor por defecto. Se abre la URL en el mismo lugar donde se ha pulsado el enlace.
  - *\_parent*: la URL se carga en el contexto padre o marco asociado.
  - *\_top*: la URL se carga en el contexto más alto y ocupa toda la ventana.

### 1. Enlaces absolutos

Son los que se usan para poner una URL completa y deben incluir el protocolo *http://* acompañado del dominio. Por lo general, las URL pertenecen a otros servidores.

```
<html>
<head>
  <title>LM</title>
</head>
<body>
<p>enlaces</p>
<hr>
  <a href="https://www.google.com">ir a Google</a>
  <br>
  <a href="https://www.google.com" target="_blank">ir a Google, abriendo
    la URL en otra ventana</a>
</body>
</html>
```

enlaces

ir a Google  
ir a Google, abriendo la URL en otra ventana

**Figura 2.6**  
 Resultado en el navegador.

## 2. Enlaces relativos

Se usan para acceder a URL alojadas en el mismo servidor, ya sea dentro o fuera del directorio donde se encuentra el enlace. La manera de actuar cuando quiere accederse a otra página dentro del mismo directorio es poner el nombre de la página junto a la extensión como valor del atributo `href`.

Si quiere accederse a una URL que está dentro de un subdirectorío, se tendrá que indicar el nombre de este junto a la URL.

Otro aspecto que puede darse es estar situado en un subdirectorío y querer acceder a una página del directorio padre. En ese caso, hay que poner dos puntos seguidos “..” por cada directorio que quiera subirse.

En el ejemplo siguiente, se muestran dos enlaces: uno situado en el mismo directorio, que accede a la página `autores.htm`, y otro situado dentro del directorio contenido, que accede al tema 1.

```
<html>
<head>
    <title>LM</title>
</head>
<body>
<p>enlaces</p>
<hr>
    <a href="autores.htm">ver los autores del libro LM</a>
    <br>
    <a href="contenido/tema 1.htm">acceder al tema 1</a>
</body>
</html>
```

## 3. Enlaces dentro de la misma página

Al disponer de páginas con mucho contenido, puede ser interesante crear un pequeño índice que acceda al contenido de ese apartado. En este punto, se verá la forma de actuar para poder crear enlaces dentro de la misma página.

Lo primero es crear los marcadores que serán las zonas de la página a las que pretende accederse. Para ello, se usa el atributo `name` de la etiqueta `<a>` con el valor del marcador que se desee.

El siguiente paso es añadir el enlace que lleve a un marcador. Para ello, dentro del atributo `href`, se pondrá el carácter almohadilla (#) seguido del nombre del marcador.

A continuación se plasma una página con información sobre tres cultivos de plantas subtropicales. Se supone que los comentarios se sustituirán por abundante información sobre cada uno de ellos. En la parte superior, se ha creado un pequeño índice que accederá a cada una de las marcas creadas dentro del documento.

```

<html>
<head>
    <title>LM</title>
</head>
<body>
    <p>enlaces</p>
    <hr>
    <a href="#Longan">ver el cultivo del longan</a><br>
    <a href="#Litchis">ver el cultivo del litchi</a> <br>
    <a href="#Carambola">ver el cultivo de la carambola</a><br>
    <h1>
        <a name="Longan"> cultivo del Longan </a>
    </h1>
    <!-- texto sobre el cultivo del longan-->
    <h1>
        <a name="Litchis">cultivo del litchi </a>
    </h1>
    <!-- texto sobre el cultivo del litchi-->
    <h1>
        <a name="Carambola">cultivo de la carambola </a>
    </h1>
    <!-- texto sobre el cultivo de la carambola-->
</body>
</html>

```

**Resultado en el navegador**

## E) Etiquetas de imágenes

La etiqueta `<img>` se usa para añadir imágenes en una página y no tiene etiqueta de cierre. Por lo general, la imagen puede situarse en la misma carpeta donde se encuentra el documento HTML o situar todas las imágenes en una carpeta concreta y acceder a ella mediante rutas absolutas o relativas.

Las extensiones de las imágenes que pueden usarse son .jpg, .bmp, .png, .gif, etc. Por defecto, siempre aparecen en la parte superior izquierda de la página web.

Los atributos más importantes son los siguientes:

- `src`: se añade el nombre de la imagen.
- `width`: se establece el ancho de la imagen.
- `height`: establece el alto de la imagen.
- `alt`: se usa para dar una breve descripción de la imagen.
- `align`: para la alineación de la imagen y puede tomar los valores:
  - `top`: se sitúa en la parte superior del texto.
  - `middle`: se sitúa centrada respecto al texto.
  - `bottom`: se alinearán con la parte inferior del texto.
  - `left`: se alinea a la izquierda.
  - `right`: se alinea a la derecha.
- `border`: se le asigna un borde con valor numérico

En este ejemplo se añaden tres imágenes de distinto tamaño.

```
<html>
<head>
    <title>LM</title>
</head>
<body>
<p>imágenes</p>
<hr>
    
    
    
</body>
</html>
```

**Resultado en el navegador**

### Ejercicio resuelto 2.3

Añade tres imágenes, pero con porcentajes:

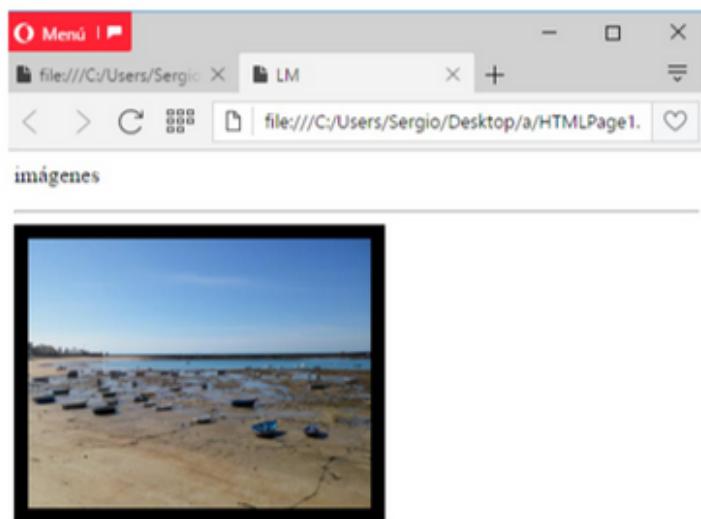
```
<html>
<head>
    <title>LM</title>
</head>
<body>
<p>imágenes</p>
<hr>
    
    
    
</body>
</html>
```

**Solución**

A continuación se muestra el uso de los atributos para el borde y la alineación.

```
<html>
<head>
    <title>LM</title>
</head>
<body>
<p>imágenes</p>
<hr>

</body>
</html>
```



**Figura 2.7**  
Resultado en el navegador.

## F) Etiquetas de tablas

- *Etiqueta <table>*: el uso de las tablas en páginas web es fundamental, pues ayuda a colocar de manera estructurada objetos dentro de ellas.
- *Etiqueta <caption>*: para añadir un título a la tabla.
- *Etiqueta <th>*: celda de cabecera en la primera fila.
- *Etiqueta <tr>*: va dentro de la etiqueta <table> y se usa para crear filas.
- *Etiqueta <td>*: se sitúa anidada dentro de una etiqueta <tr> y se usa para asignar columnas o celdas dentro de una fila.
- *Atributos*:
  - *border*: se añade un borde a la tabla.
  - *cellpadding*: para separar el borde del contenido. El valor que se le asigna es en píxeles.
  - *bgcolor*: puede añadirse a tablas, filas o columnas.

En el siguiente ejemplo se muestra una tabla con varias filas y columnas.

```

<html>
<head>
  <title>LM</title>
</head>
<body>
<p>Tablas</p>
<hr>
  <table border="5" >
    <caption>
      ejemplo completo
    </caption>
    <tr bgcolor="lime"> <td>col1</td>
      <td>col2</td>
      <td bgcolor="#3333ff">col3</td>
    </tr>
      [.../...]
  
```

```

<tr> <td>a</td>
    <td>b</td>
    <td>c</td>
</tr>
<tr>
    <td bgcolor="red">1</td>
    <td>2</td>
    <td>3</td>
</tr>
<tr>
    <td bgcolor="#009933">x</td>
    <td>y</td>
    <td>z</td>
</tr>
</table>
</body>
</html>

```

---

Tablas

---

## ejemplo completo

col1	col2	col3
a	b	c
1	2	3
x	y	z

**Figura 2.8**  
Resultado en el navegador.

- Atributos *<td>*:

- *colspan=x*: combina columnas, siendo *x* el número de columnas por incluir.
- *rowspan=x*: combina filas, siendo *x* el número de filas por incluir.
- *Align*: alineación del contenido en horizontal con los valores (“left”, “right” y “center”).
- *Valign*: alineación vertical del contenido con valores (“bottom”, “top” y “middle”), siendo esta última la que se usa por defecto.

```

<html>
<head>
    <title>LM</title>
</head>
<body>
<p>Tablas</p>
<hr>
<table>
    <tr> <td>a</td>
        <td>b</td>
        <td>c</td>
    </tr>
    <tr>
        <td>1</td>
        <td>2</td>
        <td>3</td>
    </tr>
    <tr>
        <td>x</td>
        <td>y</td>
        <td>z</td>
    </tr>
</table>
<!--con borde y combinando celdas -->

```

```
<table border>
  <tr> <td colspan="3">abc</td>
  </tr>
  <tr>
    <td rowspan="2" >1x</td>
    <td>2</td>
    <td>3</td>
  </tr>
  <tr>
    <td>y</td>
    <td>z</td>
  </tr>
</table>
</body>
</html>
```

Resultado en el navegador 

Ahora se muestra con el atributo *cellpadding* para separar el contenido del borde:

```
<html>
<head>
  <title>LM</title>
</head>
<body>
<p>Tablas</p>
<hr>
<table border="5" height="250" width="250">
  <tr> <td valign="bottom">a</td>
    <td valign="top">b</td>
    <td valign="middle">c</td>
  </tr>
  <tr>
    <td align="right">1</td>
    <td align="center">2</td>
    <td>3</td>
  </tr>
  <tr>
    <td align="left">x</td>
    <td>y</td>
    <td align="right">z</td>
  </tr>
</table>
</body>
</html>
```

Resultado en el navegador 

**Ejercicio resuelto 2.4**

Añade una imagen a una celda de una tabla, insertando algunas celdas de color.

**Solución**

**G) Formularios**

Son uno de los elementos principales que se introducen en un documento HTML. Es la manera que tienen los usuarios de interactuar con la página web mediante una serie de elementos (botones, campos de texto, casillas de verificación, etc.), y su misión es la de obtener la información que los usuarios proporcionan a la hora de utilizar algunos de sus elementos.

Aunque, normalmente, al emplear formularios se trabaja en una arquitectura cliente servidor, donde los datos obtenidos a través del formulario se envían a un servidor para ser procesados y devolver en algunos casos la información tratada, puede que se usen de manera local, aunque es lo menos habitual.

**Actividades propuestas**

Responde si las siguientes afirmaciones son verdaderas o falsas y razona tu respuesta:

**V F**

**2.3.** Las filas de una tabla HTML se definen con la marca <td>.

**V F**

**2.4.** Las listas ordenadas <ul> presentan el atributo type, que se usa para especificar el estilo de numeración (1,a,i,l,A).

**Verificar**

**Recuperación de nombre de usuario y contraseña**

Si no te acuerdas del nombre de usuario o contraseña, introduce el correo electrónico que nos facilitaste al darte de alta y lo recibirás de inmediato

Correo electrónico:

Recordar Usuario y Contraseña

**Figura 2.9**

Ejemplo de formulario de recuperación de nombre de usuario y contraseña.

Entre los formularios que pueden crearse, destacan:

- Registro de usuarios.
- Consultas.
- Libro de visitas.

- Pasarela de pago.
- Motor de búsqueda.
- Recordar contraseña.
- Identificación de usuarios.

A continuación, se presenta un ejemplo de formulario para recuperar el nombre y la contraseña de un usuario que ya ha realizado el registro previo, pero que no las recuerda.

Pueden apreciarse dos elementos de formularios: los campos de texto (donde se introduce el correo electrónico) y el botón Recordar Usuario y Contraseña, que, al pulsarse, enviará la información al servidor y devolverá al usuario los datos que ha pedido.

## 1. Estructura de los formularios

La estructura está compuesta por el elemento <form>, que actúa como contenedor de distintos controles que permiten interactuar con el usuario.

```

<html>
<head>
    <title>LM</title>
</head>
<body>
    <form>
        <!-- controles de formulario-->
    </form>
</body>
</html>

```

- *Elemento <form>*: tiene una serie de atributos que se detallan en el cuadro 2.1.

**CUADRO 2.1**  
Atributos del elemento <form>

Atributos	Descripción
<b>action</b>	Indica la acción o URI del programa que procesa la información.
<b>method</b>	Determina el método HTTP usado por el navegador y tiene dos posibles valores: <ul style="list-style-type: none"> <li>• Post: los datos se incluyen en el cuerpo del formulario.</li> <li>• Get: los datos se adjuntan en la URI después del carácter separador ?</li> </ul>
<b>enctype</b>	= "TEXT/PLAIN" indica que los datos se envían sin codificación.
<b>target</b>	Es una palabra clave que informa de dónde se mostrarán las respuestas una vez procesados los datos del formulario. Algunos valores: _blank, _parent, _top, etc.

- Elemento `<input>`: se usa para la creación de distintos tipos de controles que interactúan con el usuario. Dependiendo del valor del atributo `type`, aparecerá un control u otro.

## 2. Controles de texto

Permiten introducir información a través de unos campos de texto de una o varias líneas. Los que se estudiarán en este apartado son:

- *Texto de una sola línea*: aplicando el valor “text” al atributo `type` del elemento `<input>`, muestra un campo de texto que permite introducir una sola línea.
- *Texto de múltiples líneas*: de características similares al anterior, pero permite añadir más líneas. Puede ser interesante usarlo cuando se pide una descripción, opinión personal o un campo que vaya a ocupar más caracteres de los esperados. Para añadir un control de este tipo, se usa el elemento `<textarea>`, junto al número de filas y columnas que deben aparecer en el control.
- *Texto oculto*: aplicando el valor “password” al atributo `type` del elemento `<input>`, muestra un campo que sirve para ocultar la información y es de utilidad para añadir la contraseña.
- *Subir archivos*: puede considerarse de esta categoría y se usa como examinador para subir archivos. Aplicando el valor “file” al atributo `type` del elemento `<input>`, muestra un campo que sirve para subir documentos al servidor.

```
<html>
<head>
    <title>LM</title>
</head>
<body>
    <form action="">
        <p>
            Introduce tu nombre de usuario:
            <input type="text" name="nombre" size="45" maxlength="40" />
        </p>
        <br>
        <p>
            Introduce la contraseña:
            <input type="password" name="clave" size="45" maxlength="15" />
        </p>
        <br>
        <p>
            introduce un comentario:
            <textarea name="mensaje" rows="5" cols="50"></textarea>
        </p>
        <br>
        <p>
            subir currículo
            <input type="file" name="fichero" size="60" />
        </p>
    </form>
</body>
</html>
```

Resultado en el navegador 

### 3. Controles de selección, verificación y listas desplegables

Dependiendo del control que se use, podrá seleccionarse una o varias de las opciones que se presentan:

- *Botón de opciones*: aplicando el valor “radio” al atributo *type* del elemento `<input>`, muestra un campo que sirve para seleccionar una de las opciones que aparecen en el control. Necesitan ser agrupados para que funcionen correctamente añadiendo el mismo valor en el atributo *name*. En el atributo *value*, se añade la opción seleccionada, que debe ser diferente en cada grupo. Mediante CHECKED, se selecciona uno de ellos.
- *Control de selección o verificación*: aplicando el valor “checkbox” al atributo *type* del elemento `<input>`, muestra un campo que sirve para seleccionar una o varias opciones que aparecen en el control y, mediante CHECKED, se activa por defecto. El atributo *value* es opcional
- *Combo o listas desplegables*: es similar a los estudiados con anterioridad, pero con la peculiaridad de que, para ver las distintas opciones, hay que pulsar sobre el control y seleccionar solo una de las opciones que aparecen. Se usa el elemento `<select>` anidando cada valor que ha de elegirse dentro del elemento `<option>`.

```

<html>
<head>
    <title>LM</title>
</head>
<body>
    <form action="">
        <p>
            <b>ejemplo de botón de opciones</b><br>
            Selecciona un producto...<br>
            <input type="radio" name="productos" value="aguacate">
            El aguacate de la Axarquía<br>
            <input type="radio" name="productos" value="platano">
            El platano de Canarias<br>
            <input type="radio" name="productos" value="chirimollo">
            La chirimolla de Motril<br>
            <input type="radio" name="productos" value="longan">
            el longan del sur de China <br>
        </p>
        <br>
        <p>
            <b>Control de selección o verificación</b><br>
            Seleccione un producto favorito...<br>
            <input type="checkbox" name="a1" value="aguacate">El Aguacate<br>
            <input type="checkbox" name="a2" value="platano"> El platano<br>
            <input type="checkbox" name="a3" value="chirimollo"> El chirimollo<br>
            <input type="checkbox" name="a4" value="longan" checked>El longan
        </p>
        [.../...]
    </form>
</body>
</html>

```

```
<br>
<p>
    <b>ejemplos de listas</b>
    <br>
    Seleccione una opción...<br>
    <select name="Color">
        <option>aguacate</option>
        <option>platano</option>
        <option>chirimollo</option>
        <option>Longan</option>
    </select>
</p>
</form>
</body>
</html>
```

Resultado en el navegador 

#### 4. Controles de botones

Gracias a este tipo de controles, el usuario puede interactuar con el servidor, ya que, al presionarlo, se ejecuta un evento o acción. En este apartado, se verán los controles de botones más usados:

- *Botón de envío:* es uno de los controles que no suele faltar a la hora de crear un formulario. Aplicando el valor “submit” al atributo *type* del elemento `<input>`, muestra un botón que, al pulsarlo, envía el formulario. Otro atributo que hay que cumplimentar es *value*, cuyo valor será el texto del botón.
- *Botón de reinicio o borrado:* aplicando el valor “reset” al atributo *type* del elemento `<input>`, muestra un botón que, al pulsarlo, “limpia o restaura a valor inicial” los valores de los campos que hay en el formulario.

#### Ejercicio resuelto 2.5

Añade al ejemplo de controles de texto un botón al final de la página.

Solución 

#### 2.4. Hojas de estilo CSS

En este apartado, se estudiarán los aspectos fundamentales para poder conocer y trabajar las hojas de estilo en cascada. Se comenzará con una breve introducción, analizando las distintas versiones y su sintaxis básica (estructura, selectores, propiedades, etc.).

www

**Recurso web**

Bootstrap es un framework de código abierto para el diseño de interfaces web mediante hojas de estilo y JavaScript. Con este enlace podrás acceder a su página web.



#### 2.4.1. Introducción a CSS

Las hojas de estilo en cascada (*cascading style sheets*) o CSS son un lenguaje creado por W3C (World Wide Web Consortium) que añade diferentes estilos a documentos HTML o XHTML, dotándolos de formato mediante su uso. Se crearon para poder separar el contenido de un documento de su aspecto, lo que permite tener documentos mejor estructurados y limpios. Para separar el contenido de la presentación, se usa una serie de reglas sobre elementos que hay que declarar previamente, ya sea en la misma página o en un documento externo.

Las hojas de estilo pueden aplicarse a un documento entero, a una parte o, simplemente, a una etiqueta concreta, ya que pueden definirse varios estilos para una misma etiqueta. Permiten modificar la presentación de los elementos de un documento HTML sin tocar su código. Ahorran tiempo a la hora de definir estilos y ofrecen un abanico extenso de herramientas para el formato, más amplio que HTML.

Para poder comprender mejor el funcionamiento, puede observarse el ejemplo siguiente, donde se crean dos documentos independientes.

Primero, se aprecia la página web con código HTML.

```
<html>
<head>
    <title>LM</title>
</head>
<body>
    <h1>
        Principales cultivos subtropicales</h1>
    <ul>
        <li>Longan</li>
        <li>Aguacate</li>
        <li>Chirimollo</li>
        <li>Litchi</li>
    </ul>
</body>
</html>
```

La visualización en un navegador muestra una cabecera y una lista desordenada con elementos subtropicales, sin ningún tipo de estilo o formato (figura 2.10).

## Principales cultivos subtropicales

- Longan
- Aguacate
- Chirimollo
- Litchi

**Figura 2.10**  
Lista desordenada sin ningún formato.

Al documento HTML del ejemplo, se le añade en la cabecera la siguiente línea, que vincula el fichero CSS creado:

```
<head>
    <title>LM</title>
    <link rel="stylesheet" type="text/css" href="hoja_estilo.css"/>
</head>
```

Para aplicar estilos, se crea el siguiente fichero CSS, donde al cuerpo se le proporciona un color de fondo azul; al elemento *h1*, tipo de fuente verdana y color blanco, subrayando los elementos de la lista.

Fichero de hoja de estilos (hoja\_estilo.css) sería el siguiente:

```
body{background-color:Blue;}
h1{font-family:Verdana; color:White;}
li{text-decoration:underline}
```

El resultado de aplicar el estilo al documento se observa en la figura 2.11.

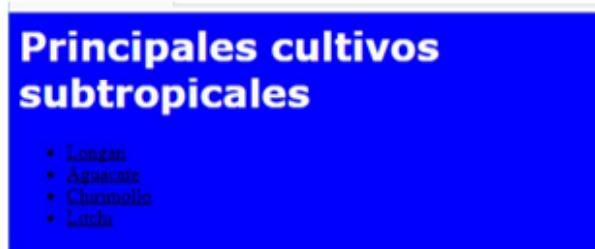


Figura 2.11

Lista desordenada aplicando la hoja de estilos.

## 2.4.2. Versiones de CSS

El W3C es el encargado de mejorar cada una de las versiones que van saliendo al mercado, dotándolas de más funcionalidad y corrigiendo algunos problemas que pudieran tener versiones antiguas.

- **CSS 1:** a finales de 1996, se publica la primera versión que permite describir formato, alineación y atributos de texto, así como las propiedades de las fuentes (tamaño, color, tipo, etc.). Actualmente, no se usa.
- **CSS 2:** incluye casi la totalidad de CSS 1. A mediados de 1998, se publica la segunda recomendación oficial de W3C. Añade como mejora para las capas <DIV> el tipo de posicionamiento y nivel.
- **CSS 2.1:** elimina errores y funcionalidades en desuso de versiones anteriores.
- **CSS 3:** usa módulos que añaden nueva funcionalidad a CSS 2, preservando de esa manera la compatibilidad con versiones anteriores. Surge a finales del 2011.

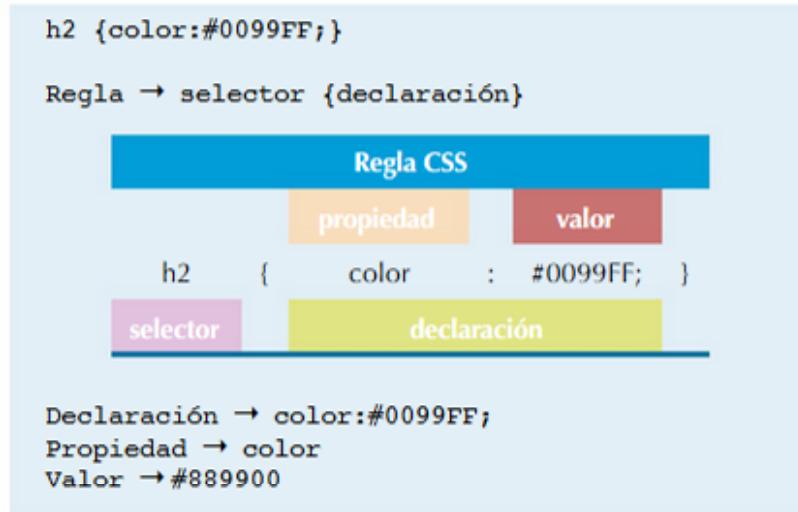
## 2.4.3. Sintaxis básica

En este apartado, se verán los aspectos más importantes que han de tenerse en cuenta para poder trabajar con hojas de estilo en cascada.

## A) Reglas CSS

Las reglas son los pilares fundamentales a la hora de crear estilos a un documento. En este apartado, se describirá cada uno de sus elementos tratando un ejemplo concreto.

- *Regla*. Es una declaración sobre un estilo de uno o varios elementos. Está compuesta por dos partes diferenciadas: un selector y una declaración.



### Ejercicio propuesto 2.1



Guiándote con el ejemplo anterior, crea una regla CSS similar y distingue el selector, la declaración, la propiedad y el valor de esta.

- *Selector*. Pueden ser uno o varios los elementos a los que se aplica la regla CSS. Dependiendo de la declaración que se añada, afectará a uno o a varios elementos del documento creado. En este ejemplo, el selector es `h2` y a todos los elementos `h2` del documento HTML o XHTML les afectará la declaración realizada. Para definir el estilo de un selector, se escribe la etiqueta seguida de la lista de declaraciones encerradas entre llaves.

Un selector puede tener varias declaraciones separadas por el carácter punto y coma (;), tal y como se aprecia en la actividad propuesta 2.5, a la hora de declarar el selector `h1`, siendo el último carácter de punto y coma opcional.

- *Declaración*. Está compuesta por dos partes diferenciadas, propiedad y valor, separadas por el carácter dos puntos (:), es donde se especifica el estilo del selector o elemento y puede contener varias propiedades en la misma declaración.

Así, la declaración es `color:#0099FF`.

- *Propiedad*. Permite modificar el aspecto de una característica de un selector. Ejemplos de propiedades pueden aplicarse sobre: color, fuente, márgenes, texto, clasificación, etc.
- *Valor*. Indica la asignación que se hace a la propiedad. Por ejemplo, a la propiedad `color`, pueden aplicársele valores de la paleta de Windows (negro, blanco, azul, etc.).

**Actividades propuestas**

Señala la respuesta correcta a las siguientes preguntas:

**2.5.** ¿Cuántos selectores hay en la siguiente hoja de estilos?

```
body{ background-color:Blue; }
h1{font-family:Verdana; color:White; }
li{ text-decoration:underline}
```

- a) Existen tres selectores en el documento anterior que son body, h1 y li.
- b) Existen dos selectores en el documento anterior que son body y h1.
- c) Existe un selector en el documento anterior que es body.

**2.6.** ¿Por cuántos elementos está compuesta una declaración?

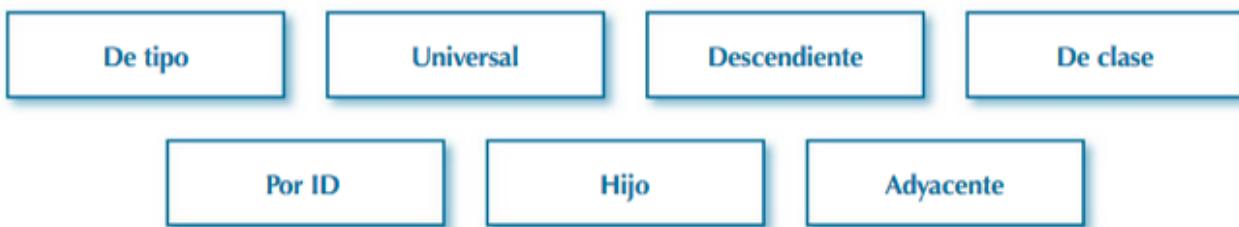
- a) Uno.
- b) Dos.
- c) Cuatro.

**2.7.** ¿Entre qué caracteres va la declaración de un selector de una regla CSS?

- a) selector { declaración}.
- b) selector [declaración ].
- c) selector (declaración ).

**Verificar****B) Selectores**

Tal y como se ha mencionado con anterioridad, cualquier elemento HTML puede ser un selector. En la figura 2.12, se observan los más usados.



**Figura 2.12**  
Tipos de selectores.

**1. Selector de tipo**

Se aplica a los elementos especificados de una página. Hay que añadir el nombre del elemento seguido de una declaración.

```
body{
    background-color:Blue;
}
```

## 2. Selector universal

Utiliza el carácter asterisco (\*) y se aplica la regla a todos los elementos de la página web. No suele usarse con frecuencia.

```
*{
    declaración
}
```

## 3. Selector descendiente

Se usa cuando un elemento es descendiente de otro, es decir, se encuentra dentro de otro (entre la etiqueta de apertura y cierre).

No hace falta que sea descendiente directo y no afectará al resto de elementos de la página siempre y cuando no desciendan del especificado.

Tal y como se aprecia a continuación, se ha creado un selector descendiente (h3 i) que afecta a los elementos *i* que se encuentren dentro de un elemento de cabecera h3 aplicándoles el formato: color de texto rojo, tipo de fuente Verdana y texto subrayado.

```
<html>
<head>
    <title>LM</title>
    <style type="text/css">
        h3 i { color: red;
            font-family:Verdana;
            text-decoration: underline;
        }
    </style>
</head>
<body>
    <h3> Cultivo del <b>Litchi</b> en la zona de la <i>Axarquía</i></h3>
    <h3> Cultivo del <b><i>longan</i></b> en Canarias</h3>
    <h3> Cultivo de <b>la carambola</b> en Málaga</h3>
    <h3> Cultivo del <i>canisté</i> en Canarias</h3>
    <h4> Cultivo de la <i>guanábana</i> en Canarias</h4>
</body>
</html>
```

**Resultado al aplicar el CSS** 

#### 4. Selector de clase

Se usa para aplicar estilos a un elemento en concreto del documento y puede definirse dentro de la etiqueta <style>, dentro del documento, así como en un fichero externo.

El nombre que se asigna a un selector de clase puede ser el que se desee, aunque suele darse uno que identifique a la regla que quiere crearse y que empiece por el carácter punto (.). Existen dos maneras de definir un selector de clase:

- NombreSelector{ declaración}.
- elemento.NombreSelector{ declaración}.

##### RECUERDA

- ✓ Para definir un selector de clase, el nombre debe empezar por el carácter punto (.).

Para poder usar el selector en un elemento, se usa el atributo *class* de las etiquetas HTML a las que quiere aplicarse el estilo. La nueva regla creada podrá aplicarse a todas las etiquetas HTML del documento que use el atributo *class* seguido del valor, que será el nombre del selector o clase sin el punto.

<etiqueta class="selector de clase">

Para poder ver el funcionamiento de un selector de clase, se muestra el ejemplo siguiente, donde se declaran dos selectores de clase (colorRojo y Color\_fondo\_azul). El primero de ellos pondrá el texto de color rojo y tipo de fuente verdana. El segundo cambiará el fondo del elemento a color azul. Estos selectores solo afectarán a los elementos del documento que añadan el atributo *class*.

```
<html>
<head>
    <title>LM</title>
<style type="text/css">
    .colorRojo
    {
        color: red;
        font-family: Verdana;
    }
    .Color_fondo_azul
    {
        background-color: Blue;
    }
</style>
```

[.../...]

```

</head>
<body>
    <h3 class="colorRojo">Cultivo del Litchi.</h3>
    <h3>Cultivo del longan en Canarias.</h3>
    <h3>Cultivo de la <b class="Color_fondo_azul">carambola.</b> </h3>
    <h3>Cultivo del <i class="colorRojo">canisté</i>.</h3>
    <h4 class="Color_fondo_azul"> Cultivo de la guanábana.</h4>
</body>
</html>

```

**Resultado utilizando la propiedad class**

En este apartado, se han visto dos maneras de definir un selector. La segunda (elemento. NombreSelector{ declaración}) se interpreta como los elementos de tipo X que contengan el atributo *class* con el valor igual al nombre del selector.

En el ejemplo siguiente, el selector (b.colorRojo) afectará a los elementos <b> que contengan el atributo *class* con valor “colorRojo”.

```

<html>
<head>
    <title>LM</title>
<style type="text/css">
    b.colorRojo
    {
        color: red;
    }
</style>
</head>
<body>
    <h3 class="colorRojo">Cultivo del Litchi.</h3>
    <h3>
        Cultivo del longan en Canarias.</h3>
    <h3>
        Cultivo de la <b class="colorRojo">carambola.</b>
    </h3>
    <h3>
        Cultivo del <i class="colorRojo">canisté</i>.</h3>
    <h4 class="colorRojo">
        Cultivo de la guanábana.</h4>
</body>
</html>

```

**Resultado al utilizar el atributo class**

## 5. Selector por ID

Similar a los selectores de clase, pero siendo la almohadilla (#) el carácter que precede al selector en vez del punto. Se usa para aplicar un estilo a un elemento concreto del documento mediante el valor del atributo *id* de los elementos, que es único.

En el ejemplo, el selector (#colorRojo) afecta a la primera cabecera <h3>, ya que el atributo *id* tiene el valor del nombre del selector sin el carácter almohadilla.

```

<html>
<head>
    <title>LM</title>
    <style type="text/css">
        #colorRojo
        {
            color: red;
        }
    </style>
</head>
<body>
    <h3 id="colorRojo">Cultivo del Litchi.</h3>
    <h3>Cultivo del longan en Canarias.</h3>
    <h3>Cultivo de la <b>carambola.</b></h3>
    <h3> Cultivo del canisté.</h3>
    <h4>Cultivo de la guanábana.</h4>
</body>
</html>

```

**Resultado con un selector por ID** 

## 6. Selector hijo

Similar al descendiente, pero, esta vez, solo afecta a hijos directos de los elementos. Para ello, se usa el signo mayor que (>).

En el siguiente ejemplo, se presenta un selector descendiente (h3>i) y un selector descendiente (h3 i). El primero de ellos pondrá de color rojo el texto de las etiquetas que sean hijas directas de la cabecera h3, mientras que, en el segundo, al ser adyacente, no hace falta que sean hijas directas de h3, aplicando un estilo al texto (subrayado y aumentado de tamaño).

```

<html>
<head>
    <title>LM</title>
    <style type="text/css">
        h3 > i {
            color: red;
        }
        h3 i {
            text-decoration:underline;
            font-size:xx-large;
        }
    </style>
</head>
<body>
    <h3> Cultivo del <b>Litchi</b> en la zona de la <i>Axarquía</i></h3>
    <h3>Cultivo del <b><i>longan</i></b> en Canarias</h3>
    <h3>Cultivo de <b>la carambola</b> en Málaga</h3>
    <h3>Cultivo del <i>canisté</i> en Canarias</h3>
    <h4>Cultivo de la <i>guanábana</i> en Canarias</h4>
</body>
</html>

```

## 7. Selector adyacente

Se usa para elementos que están seguidos de otros y son hermanos (el elemento padre de ambos es el mismo). El signo (+) se emplea para la declaración entre los elementos.

En el siguiente ejemplo, el selector adyacente es (h1 + h3), indica que se aplicará el estilo de color de texto rojo a los elementos de cabecera h3 que sean hermanos de h1 y aparece justo después de él. Hay que destacar que h1 y h3 son hermanos, ya que el padre de ambos es el elemento <body>.

```
<html>
<head>
    <title>LM</title>
    <style type="text/css">
        h1 + h3 {
            color: red;
        }
    </style>
</head>
<body>
    <h1> Cultivo del <b>Litchi</b> en la zona de la <i>Axarquía</i></h1>
    <h3>Cultivo del <b><i>longan</i></b> en Málaga</h3>
    <h3>Cultivo del <b>platano</b> en Canarias</h3>
    <h1>Cultivo de <b>la carambola</b> en Málaga</h1>
    <h2>Cultivo del <i>canisté</i> en Canarias</h2>
    <h3>Cultivo de la <i>guanábana</i> en Canarias</h3>
</body>
</html>
```

**Resultado al aplicar selectores**



### Ejercicio resuelto 2.6



Teniendo en cuenta el siguiente código HTML y CSS, ¿de qué color saldrá la palabra examen?

Documento HTML:

Documento CSS:

```
<html>
    <head>
        <title>ejemlo</title>
        <link rel="stylesheet" type="text/
            css" href="al.css" />
    </head>
    <body>
        <h1>examen</h1>
        <h2>CFGs</h2>
        <h2>ASIR</h2>
    </body>
</html>
```

```
h1 + h2{color: blue}
```

**Solución**



**Actividad propuesta 2.8**

¿Qué signo se emplea para definir estilos usando clases?

- V  F a) .clase.
- V  F b) \$clase.
- V  F c) --clase.

**Verificar**

### C) Propiedades y valores

Un selector puede contener una o varias propiedades. En los siguientes cuadros se describen las relacionadas con el texto, el tipo de letra, el posicionamiento, etc., detallando algunos valores de sobre ellas junto a su descripción.

**CUADRO 2.2**  
Propiedades de texto

Propiedad	Valores	Descripción
<i>text-decoration</i>	none, blink, line-through, overline, underline.	Decora el texto.
<i>color</i>	Los especificados (RGB o nombre color).	Color del texto.
<i>text-align</i>	center, justify, left, right.	Alineación del texto.
<i>direction</i>	ltr, rtl.	Dirección del texto.
<i>text-transform</i>	none, capitalize, lowercase, uppercase.	Convierte mayúsculas, minúsculas, etc.

**CUADRO 2.3**  
Propiedades de tipo de letra

Propiedad	Valores	Descripción
<i>font-size</i>	xx-small, x-small, small, medium, large, x-large, xx-large.	Tamaño.
<i>font-family</i>		Tipo de letra (fuente).
<i>font-style</i>	normal, italic, oblique.	Estilo fuente.

**CUADRO 2.4**  
Propiedades sobre fondos

Propiedad	Valores	Descripción
<i>background-color</i>	transparent, color.	Color de fondo.
<i>background-image</i>	none, uri.	Imagen de fondo.

**CUADRO 2.5**  
Propiedades posicionamiento

Propiedad	Valores	Descripción
<i>float</i>	none, left, right.	Posicionamiento flotante.
<i>display</i>	none, block...	Tipo de caja.
<i>position</i>	absolute, fixed, relative, static.	Modo de posicionamiento.
<i>z-index</i>	auto, número-entero.	Apilamiento.

**CUADRO 2.6**  
Propiedades de tamaño

Propiedad	Valores	Descripción
<i>height</i>	auto, distancia, porcentaje.	Altura.
<i>width</i>	auto, distancia, porcentaje.	Anchura.

### Actividad propuesta 2.9



¿Cómo se define la regla CSS para que el texto aparezca centrado?

- a) *center: true.*
- b) *align: center.*
- c) *text-align: center.*

Verificar

### D) Comentarios, agrupamiento y herencia

#### 1. Comentarios

Los comentarios son fundamentales en todo lenguaje de programación que proporcionan información de utilidad a la hora de estructurar o indicar cualquier apreciación sobre el conte-

nido. Pueden ocupar una o varias líneas, dependiendo de lo que se quiera añadir. En el caso de CSS, al igual que los comentarios en HTML, no son visibles por el navegador. Todo comentario va entre los caracteres de inicio y fin:

- Los caracteres de inicio del comentario son la barra y el asterisco (/\*).
- Los caracteres de fin de comentario son el asterisco y la barra (\*/).

```
<style type="text/css">
    h1 + h3 {
        color: red;
        /*pone de color rojo el texto*/
    }
</style>
</head>
```

### Investiga



¿Qué diferencia existe entre los comentarios CSS y HTML?

## 2. Agrupamientos

Cuando pretende aplicarse la misma declaración a distintos selectores, la manera de hacerlo es agrupando los selectores separados por el carácter coma (,).

Las reglas de estilo de un documento son las tres que se aprecian cuyos selectores (h1, h2 y h3) ponen de color rojo el texto del elemento.

```
<style type="text/css">
    h1{color: red;}
    h2{color: red;}
    h3{color: red;}
</style>
```

Existe una manera de agrupar las tres reglas en una, con su declaración correspondiente. Consiste en poner cada selector separado por el carácter coma (,) seguido de la declaración, que, en este caso, es idéntica en las tres reglas.

El agrupamiento sería:

```
h1,h2,h3{
    color: red;
}
```

### 3. Herencia

Al tener un documento HTML una estructura de árbol, los sectores anidados heredan las declaraciones de los padres. Esto se ve perfectamente con la etiqueta <body>, ya que, si se le aplica un estilo, afectará a todos los elementos del documento, debido a que es el padre del resto de etiquetas que se añaden.

En el caso de que el usuario modifique un elemento concreto mediante un selector de clase, ID u otros, prevalecerá dicho estilo en ese elemento.

```
<html>
<head>
    <title>LM</title>
    <style type="text/css">
        h1{color: red;}
        body{color:Blue;}
    </style>
</head>
<body>
    <h1> Cultivo del <b>Litchi</b> en la zona de la <i>Axarquía</i></h1>
    <h3>Cultivo del <b><i>longan</i></b> en Málaga</h3>
    <h3>Cultivo del <b>platano</b> en Canarias</h3>
    <h1>Cultivo de <b>la carambola</b> en Málaga</h1>
    <h2>Cultivo del <i>canisté</i> en Canarias</h2>
    <h3>Cultivo de la <i>guanábana</i> en Canarias</h3>
</body>
</html>
```

### E) Unidades de medida

Gracias a CSS, se consigue mayor precisión a la hora de ubicar en una zona del documento un elemento concreto y definir distancias y visibilidad de los elementos, contando con un mayor número de unidades de medida absolutas y relativas que se establecen con un valor entero o decimal que ayudan a definir los márgenes, anchura, etc., de los distintos elementos.

Algunas de las unidades absolutas se aprecian en el cuadro 2.7.

**CUADRO 2.7**

Algunas de las unidades absolutas

Puntos	Pulgadas	Centímetros	Píxeles	Milímetros
pt	in	cm	px	mm

### Cultivo del Litchi en la zona de la Axarquía

Cultivo del *longan* en Málaga

Cultivo del *platano* en Canarias

### Cultivo de la carambola en Málaga

Cultivo del *canisté* en Canarias

Cultivo de la *guanábana* en Canarias

**Figura 2.13**

Resultado al aplicar herencia.

El texto del cuerpo de la página tendrá un tamaño de letra de 8 puntos:

```
body{  
    font-size: 8pt;  
}
```

#### 2.4.4. Maneras de aplicar estilos a un documento HTML

En este apartado, van a explicarse las distintas maneras existentes de aplicar estilos a un documento HTML.

##### A) Estilos aplicados en un documento externo

Se creará un documento de hojas de estilo, cuya extensión será .css, donde se declararán las distintas reglas de la página web. Para poder vincular el archivo externo .css al documento HTML, se usa la etiqueta <link> con una serie de atributos.

Este procedimiento es mucho más elegante, ya que los distintos estilos irán añadiéndose al fichero CSS que se haya creado, ahorrando tiempo y líneas HTML y dejando un código mucho más entendible, evitando tener que definir estilos de manera individual a todas las etiquetas que se tienen en HTML.

##### RECUERDA

- ✓ La extensión de un documento de hojas de estilo es .css.

El fichero CSS que tiene que crearse será de texto plano, donde se añadirán las reglas con sintaxis CSS y se enlazará con el documento HTML mediante la etiqueta <link>, que se añade dentro de la etiqueta <head>, y una serie de atributos que se detallan a continuación.

La etiqueta <link> corresponde al tipo de etiquetas vacía al no tener contenido. Se usa para relacionar un recurso externo con el documento HTML, aunque se emplea para incluir el fichero de hojas de estilo externo, puede tener más usos.

Entre sus atributos, destacan:

- *type*: informa del tipo de contenido del fichero por enlazar. Para el uso de hojas de estilo, el valor que ha de añadirse es “text/css”, que informa de que el archivo es de texto y su sintaxis es CSS.
- *href*: URL o ruta donde se localiza el nombre del documento CSS. Es la fuente del fichero que quiere enlazarse, que puede tener una ruta absoluta o relativa.
- *rel*: informa de la relación que existe entre el documento externo y el actual. Para el caso particular de hojas de estilo, su valor es “stylesheet”, que indica que el fichero por enlazar es una hoja de estilos que aplicará el formato al documento HTML.

Etiqueta <link>:

```
<link rel="stylesheet" type="text/css" href="miestilo.css" />
```

Documento HTML:

```
<html>
<head>
    <title>LM</title>
    <link rel="stylesheet" type="text/css" href="hoja_estilo.css" />
</head>
<body>
    <h1>Estilos aplicados a una tabla</h1>
    <table border="1">
        <tr>
            <td>floración</td>
        </tr>
        <tr>
            <td>fruto</td>
        </tr>
    </table>
</body>
</html>
```

Fichero de hoja de estilos (miestilo.css):

```
td {
    background-color:Green;
    font-family:Arial Black;
}
```

## Estilos aplicados a una tabla

floración
fruto

**Figura 2.14**  
Resultado de los estilos aplicados a una tabla.

### B) Estilos aplicados en el mismo documento

El resultado que se obtuvo en el apartado anterior, puede lograrse, pero esta vez unificando todo en un único documento, distinguiendo claramente la zona de aplicación de estilos que serán efectivos para los elementos de la página web que se hayan declarado.

Para ello, se usa la etiqueta <style> dentro de la etiqueta <head>. El contenido de la etiqueta <style> estará compuesto por una serie de reglas CSS que afectarán a todo el documento.

Cualquier cambio de formato que quiera hacerse sobre uno de los elementos de la página web tendrá que realizarse en dicho apartado.

El documento HTM que se observa lleva insertados los estilos en la cabecera sin necesidad de crear un documento externo para enlazarlos. Cuando se trabaja con pocas reglas, el documento queda elegante y legible, pero si, por el contrario, el contenido de la etiqueta <style> es muy extenso, es recomendable ponerlo en un documento externo.

```
<html>
<head>
    <title>LM</title>
    <style type="text/css">
        td {
            background-color: Green;
            font-family: Arial Black;
        }
    </style>
</head>
<body>
    <h1>Estilos aplicados a una tabla</h1>
    <table border="1">
        <tr>
            <td>floración</td>
        </tr>
        <tr>
            <td>fruto</td>
        </tr>
    </table>
</body>
</html>
```

Tal y como se aprecia, la única diferencia con el documento HTML del apartado anterior es la marcada en el cuadro azul, donde se ha creado un selector td con dos declaraciones (background-color: Green; font-family: Arial Black;).

El estilo aplicado a las columnas de la tabla hará que aparezcan con un color de fondo verde y el tipo de letra arial black.

### C) *Estilos aplicados a las etiquetas de un documento*

Es conveniente usarlos cuando pretenden tenerse dos elementos iguales, pero hay que aplicarles un formato distinto para cada uno.

Un ejemplo podría ser tener dos etiquetas <h3> y que la primera se muestre con un color de texto gris y la segunda, azul. Por otro lado, a cada una de las celdas de la tabla que se muestra, se le pondrá un color de fondo diferente. Esto no sería posible tal y como se ha visto en los apartados anteriores. Para ello, se usa el atributo style en cada una de las etiquetas que quiera ponerse.

```

<html>
<head>
    <title>LM</title>
</head>
<body>
    <h3 style="color: Gray;">
        aplica estilo a la etiqueta h1 y pone el texto de color gris</h3>
    <h3 style="color: blue;">
        aplica estilo a la etiqueta h1 y pone el texto de color azul</h3>
    <table border="1">
        <tr>
            <td style="background-color:Yellow;">
                floración
            </td>
        </tr>
        <tr>
            <td style="background-color:Olive;">
                fruto
            </td>
        </tr>
    </table>
</body>
</html>

```

**Resultado de estilos aplicados a las etiquetas de un documento** 

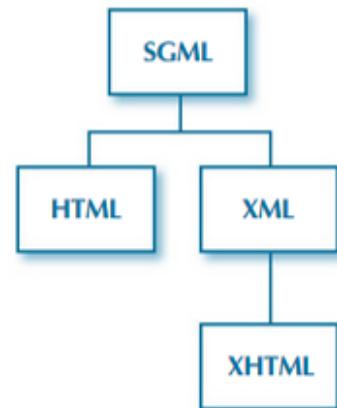
## 2.5. XHTML

Se considera XHTML, sigla de *extensible hypertext markup language* (lenguaje de marcado de hipertexto extensible) como la adaptación de HTML en XML, pues mantiene las características de HTML e incluye normas XML. Tal y como se aprecia en la figura 2.33, HTML desciende de SGML, mientras que XHTML lo hace de XML.

### 2.5.1. Versiones

En la actualidad, existen varias versiones de XHTML:

- **XHTML 1.0.** Surge en el año 2000, es la primera de todas y se considera similar a HTML 4.0 reformulado, cuya finalidad es que, gracias a ella, se use como lenguaje de contenidos. Esta versión presenta tres definiciones distintas para DTD (definición del tipo de documento) y es compatible con algunas versiones de HTML.
- **XHTML 1.1.** Esta versión no es compatible con versiones anteriores de HTML. Se considera una versión basada en módulos que sirven de base para futuros documentos extendidos, quitando características que estaban ya en desuso.



**Figura 2.15**  
Jerarquía de los  
lenguajes de marcas.

```

<?xml version="1.0" encoding="UTF-8" ?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN" "http://www.w3.org/TR/
  xhtml11/DTD/xhtml11.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en">
<head>
  <title>LM</title>
</head>
<body>
</body>
</html>

```

**CUADRO 2.8**  
Módulo XHTML 1.1

<b>Módulo de lista</b>	dl, dt, dd, ol, ul, li.
<b>Módulo de formularios</b>	button, fieldset, form, input, label, legend, select, optgroup, option, textarea.
<b>Módulo de estructura</b>	body, head, html, title.
<b>Módulo de tablas</b>	caption, col, colgroup, table, tbody, td, tfoot, th, thead, tr.
<b>Módulo de imagen</b>	img.

- **XHTML Básico.** Su objetivo es usarla por dispositivos de información que empleen acceso web como (teléfonos móviles, sistemas de navegación de coches, televisiones, relojes inteligentes, lectores de libros digitales, etc.) Presenta una versión más simplificada que las anteriores, que incluye un grupo reducido de módulos.

**CUADRO 2.9**  
Módulos XHTML Básico

<b>Módulo básico de formulario</b>	form, input, label, select, option, textarea.
<b>Módulo base</b>	base.
<b>Módulo de enlace</b>	link.
<b>Módulo de hipertexto</b>	a.
<b>Módulo básico de tablas</b>	caption, table, td, th, tr.

WWW

## Recurso web

Con este enlace puedes acceder a la guía de referencia de XHTML (W3C).



## 2.5.2. Diferencias sintácticas y estructurales con HTML

A la hora de crear documentos XHTML, no existe una gran diferencia con HTML, pero, para que esté bien formado, es necesario que cumpla una serie de reglas sintácticas heredadas, tal y como se ha visto en el apartado 1.4.

### RECUERDA

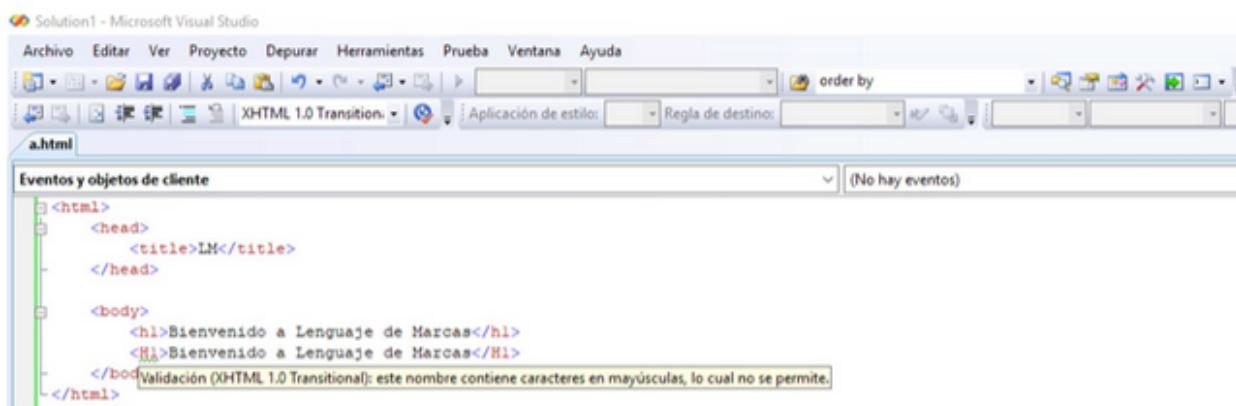
- ✓ A la hora de añadir elementos o atributos al documento, hay que hacerlo en minúsculas.

XML es sensible a las mayúsculas y minúsculas, por lo que es necesario añadir los elementos y atributos en minúsculas. Mientras la opción 1 del ejemplo 1 es correcta, ya que todos los nombres de los elementos están en minúsculas, la 2, a la hora de validarla, informará de que el nombre de la etiqueta contiene caracteres en mayúsculas, lo cual no se permite en XHTML y sí en HTML.

### EJEMPLO 1

- |   |  |
|---|--|
| 1 | <input checked="" type="checkbox"/> <h1>Bienvenido a Lenguaje de Marcas</h1> |
| 2 | <input type="checkbox"/> <H1>Bienvenido a Lenguaje de Marcas</H1>            |

El error puede visualizarse usando Visual Studio.



**Figura 2.16.**

Resultado del ejemplo anterior en XHTML.

### RECUERDA

- ✓ Los atributos tendrán obligatoriamente un valor que deberá ir entre comillas simples o dobles, aunque se prefieren estas últimas.

Ya sean valores de texto, números o alfanuméricos, deberán ir entrecomillados. Como ejemplo, se ha usado la etiqueta de imagen con tres atributos: texto alternativo, fuente y ancho de la imagen. La opción 1 del ejemplo 2 es correcta, ya que todos los valores de los atributos van con comillas, mientras que la opción 2 no lo es porque el valor del atributo *width* va sin comillas, aunque sí sería válida en HTML.

#### EJEMPLO 2

- 1    
2  

#### RECUERDA

- ✓ Todas las etiquetas o elementos tienen que estar cerrados, incluyendo las etiquetas vacías.

En HTML, la gran mayoría de etiquetas tienen una de apertura y otra de cierre, aunque existen algunas, como las de comienzo de nueva línea o de imagen, en las que no es necesario su cierre.

En el ejemplo 3, la opción número 1 es la correcta al estar en minúsculas y cerrada, mientras que a la opción 2 le falta el cierre, aunque es válida en HTML, tal y como se ha mencionado.

#### EJEMPLO 3

- 1  <br />  
2  <br>

A la hora de anidar etiquetas, hay que tener en cuenta que no deben solaparse (tienen que cerrarse conforme a cómo se abren). En el caso primero del ejemplo 4, se aprecia que las etiquetas de subrayado y cursiva no se solapan, siendo esta la manera correcta.

En el caso 2, se observa que, para poner el texto en cursiva, entre la etiqueta de apertura y la de cierre, aparece la etiqueta cerrada de subrayado, inválido en XHTML y válido en HTML.

#### EJEMPLO 4

- 1    
2  

Aunque los atributos minimizados no son los más usados, el nombre debe coincidir con su valor. Todos los atributos tienen que llevar un valor. Aunque la segunda opción del ejemplo 5 es válida en HTML, para XHTML, hay que añadir un valor al atributo *selected* con el mismo nombre, tal y como se ve en la primera opción.

#### EJEMPLO 5

```

1  <select>
    <option value="longan">Longan</option>
    <option value="carambola">Carambola</option>
    <option value="Litchi">Litchi</option>
    <option value="Rambutan" selected="selected">Rambutan</option>
</select>

2  <select>
    <option value="longan">Longan</option>
    <option value="carambola">Carambola</option>
    <option value="Litchi">Litchi</option>
    <option value="Rambutan" selected>Rambutan</option>
</select>

```

Respecto a la estructura, no varía mucho a la de un documento HTML y su mayor diferencia es la declaración del DTD para validar el documento, que es obligatoria. El elemento raíz tiene que ser la etiqueta <html>, que tendrá como atributos *xmlns*, que indica el espacio nominal XHTML (<http://www.w3.org/1999/xhtml>).

Al heredar XHTML de XML, comparte parte de su sintaxis y normas, por ello se utiliza DTD, sigla de *document type definition* o (definición del tipo de documento). Dentro de un DTD, se encuentra una serie de reglas que deben cumplirse. Se hablará con más detenimiento de este concepto en el capítulo 4.

```

<Declaración DTD>
<html ...>
  <head>
    <title>LM</title>
    ...
  </head>
  <body>
    <!-- contenido -->
  </body>
</html>

```

En la versión 1.0 de XHTML, existen tres posibilidades para la declaración del DTD (estricto, transitorio y *frameset*), incluyendo alguna de las tres fuera de la etiqueta <html> mediante la declaración DOCTYPE, pues esta es la primera que hay que añadir a la hora de crear un nuevo documento.

La declaración estricta se usa en el caso de no emplear elementos de estilo, pues es la de transición la que sí los usa. En el caso de usar marcos dentro del documento, se usará la declaración *frameset*.

**CUADRO 2.10**  
Declaración del DTD

Tipo DTD	Declaración
Estricto	<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
Transitorio	<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
Frameset	<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Frameset//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-frameset.dtd">

En el ejemplo siguiente se aprecia un documento XHTML versión 1.0 con una declaración de transición, que permite elementos en desuso.

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" >
<head>
  <title>LM</title>
</head>
<body>
</body>
</html>
```

Aunque no es obligatorio, algunos autores usan como primera línea una declaración XML, necesaria para la codificación de caracteres:

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

### 2.5.3. Ventajas de XHTML sobre HTML

XHTML presenta una serie de ventajas respecto a HTML, entre las que destacan:

- A los documentos XHTML pueden agregárseles distintas aplicaciones (*scripts y applets*).
- Pueden validarse con herramientas XML.
- Compatible con los navegadores antiguos.
- Pueden presentar mayor rendimiento.
- Formato abierto.
- Facilidad de mantener y editar.

## Resumen

- Herramientas de diseño web:
  - Código abierto: KompoZer, Bluefish y HTML Online Editor.
  - Comerciales: Dreamweaver.
- HTML 5, publicado a finales del 2014, es un lenguaje que establece nuevos elementos, algunos similares a los anteriores, pero con un significado semántico.

La declaración de tipo de documento (DTD) añade reglas sintácticas que permiten definir el documento con sus reglas y atributos, de esta manera puede comprobarse si el documento es válido.

  - Atributos HTML: bgcolor.
  - Etiquetas:
    - *Etiqueta <b>*: pone el texto contenido en ella en negrita.
    - *Etiqueta <u>*: Se usa para subrayado del texto.
    - *Etiqueta <i>*: permite poner en cursiva un texto.
    - *Etiqueta <s>*: si quiere tacharse una palabra o texto, esta es la etiqueta debe usarse.
    - *Etiqueta <br>*: salto de línea.
    - *Etiqueta <hr>*: línea horizontal.
    - *Etiqueta <SUP>*: para el superíndice.
    - *Etiqueta <SUB>*: para el subíndice.
    - *Etiqueta <p>*: se usa para crear párrafos.
    - *Etiqueta <big>*: para aumentar texto.
    - *Etiqueta <small>*: para disminuir texto.
    - *Etiqueta <font>*: fuente de texto.
    - *Etiqueta <h1>...<h6>*: encabezados.
    - *Etiqueta <ol>*: listas numéricas u ordenadas.
    - *Etiqueta <ul>*: listas desordenadas.
    - *Etiqueta <dl>*: listas de definición.
    - *Etiqueta <dt>*: listas de definición.
    - *Etiqueta <a>*: enlaces y ancla. | *Etiqueta <img>*: imágenes.
    - *Etiqueta <table>*: el uso de las tablas en páginas web es fundamental, pues ayuda a colocar de manera estructurada objetos dentro de ellas.
    - *Etiqueta <caption>*: para añadir título a la tabla.
    - *Etiqueta <th>*: celda de cabecera, en la primera fila.
    - *Etiqueta <tr>*: va dentro de la etiqueta *<table>* y se usa para crear filas.
    - *Etiqueta <td>*: se sitúa anidada dentro de una etiqueta *<tr>* y se usa para asignar columnas o celdas dentro de una fila.
  - *Etiqueta <form>*: etiqueta para crear formularios.

- *Elemento <input>*: para la creación de distintos tipos de controles que interactúan con el usuario.
- CSS, sigla de *cascading style sheets* (hojas de estilo en cascada), es un lenguaje creado por W3C (World Wide Web Consortium) que añade diferentes estilos a documentos HTML o XHTML, dotándolos de formato mediante su uso.  
Las hojas de estilo pueden aplicarse a un documento entero, a un parte o, simplemente, a una etiqueta concreta

Regla CSS		
	propiedad	valor
selector	declaración	
h2	{ color	: #0099FF; }

- *Tipos de selectores*:
  - De tipo.
  - De clase.
  - Por ID.
  - Universal.
  - Hijos.
  - Descendientes.
  - Adyacentes.
- *Agrupamientos*: cuando pretende aplicarse la misma declaración a distintos selectores, se agrupan los selectores separados por el carácter coma (,).
- *Herencia*: los sectores anidados heredan las declaraciones de los padres.
- *Unidades de medida*: puntos, pulgadas, centímetros, píxeles y milímetros.
- Se considera que el XHTML, sigla de *extensible hypertext markup language* (lenguaje de marcado de hipertexto extensible) es la adaptación del HTML en XML.

## Ejercicios prácticos resueltos

1. Realiza una página web para gestionar incidencias, donde añadas en la parte superior el nombre del alumno usando un elemento de cabecera, un campo de texto para el nombre y otro de varias líneas para describir la incidencia. Al finalizar, se añadirán dos botones: uno para enviar los datos y otro para limpiar el formulario.

El diseño puede ser similar al siguiente:

**Nombre Alumno:**

**Incidencias**

Introduzca su nombre:  
Ataulfo

introduzca la incidencia

Enviar datos Borrar datos

**Solución**

2. Crea la siguiente tabla, añadiendo el nombre del alumno, línea horizontal, etc. Respeta los colores que aparecen (que sean similares):

**números de registros**

1524	4785	7415
3256	1452	1458 2563
7485		7412
3695		7485

**Solución**

3. Elabora la siguiente lista ordenada teniendo en cuenta que aparezcan números romanos de manera automática:

- 
- Listas ordenadas**
- Asignaturas de ciclo**
- I. Sistemas operativos monopuesto
  - II. Seguridad informática
  - III. Redes locales
  - IV. Ofimática
  - V. Fundamentos

**Solución**

## Ejercicios prácticos



1. Crea una página web (relacionada con una tienda de móviles) llamada *e1.html*.

Inserta:

- Título de la página: Móviles XX.
- En el cuerpo de la página:
  - Nombre del alumno: en etiqueta *<h1>*.
  - Línea horizontal.
  - Ciclo, ASIR/DAM/DAW: en etiqueta *<h2>*.
  - Curso: en un párrafo.
  - Imagen de un teléfono llamada *LG1.jpg*.

2. Elabora una nueva lista que contenga el siguiente catálogo de móviles:

1. LG G3
2. Xiaomi
3. Samsung Galaxy
4. iPhone 6

3. Diseña la siguiente tabla

Móviles último modelo		
Categorías	Categoría 1	Categoría A

4. Aplica los siguientes estilos a la página web principal en un documento diferente llamado *estilos.css*.

- Página:
  - Color de fondo: azul.
  - Tipo de letra: verdana.
- H1:
  - Color de texto: blanco.

5. Realiza la siguiente tabla:

Partido	Votos	Escaños
P1	1000	1
P2	5000	6
P3	2000	3
P4	1500	2

6. Dado el siguiente código sin estilos:

```
<html>
<head>
    <title>Hoja 3</title>
</head>
<body>
    <h1> Nombre del alumno:</h1>
    <br>
    <p>
        Información sobre los productos que hay en la zona</p>
    <br>
    <table border="1">
        <tr>
            <td>
                longan
            </td>
            <td>
                litchi
            </td>
        </tr>
        <tr>
            <td>
                carambola
            </td>
            <td>
                aguacate
            </td>
        </tr>
    </table>
    <br>
</body>
</html>
```

Aplica estilo al documento teniendo en cuenta los siguientes aspectos de estilo:

- Tabla:
    - Color de texto: rojo.
    - Tipo de letra: verdana.
    - Color de fondo: lime.
  - Párrafos:
    - Color de texto: verde.
  - Cabeceras de texto h1:
    - Color de texto: purple.
    - Texto subrayado.
    - Poner en mayúsculas la primera letra de cada palabra.
7. Añade el color de fondo a un documento HTML en (RGB, hexadecimal y nombre del color).

8. Crea una página HTML que contenga una etiqueta de enlace, encabezados, barra horizontal, una tabla de 4 filas y 5 columnas con contenido, cambiando el color de fondo a la última fila.
9. Dada la siguiente página web, añade dos asignaturas más al *checkbox*:

```

<html>
<head>
    <title>Ejercicio </title>
</head>
<body>
    <h3>
        Formulario</h3>
    <form action="#" method="post">
        Asignatura
        <br />
        <br>
        <input name="CB1" type="checkbox" value="Redes" checked="true" />
        OACE
        <br>
        <input name="CB2" type="checkbox" value="BBDD" />
        EEE<br>
        <input name="CB3" type="checkbox" value="Sistemas Operativos" />
        Seguridad informática
        <br>
    </form>
</body>
</html>

```

10. Elabora el siguiente horario, cambiando el nombre de algunos módulos:

	Lunes	Martes	Miércoles	Jueves	Viernes		
8:00-9:00	OACE	EEE	SI	Redes	SO		
9:00-10:00		LM					
10:00-11:00	Seguridad	FOL	BBDD				
11:00-11:30	RECREO						
11:30-12:00	ISO	BBDD	FH	OACE	Redes		
12:00-13:00	LM			OACE			
13:00-14:00	Redes						

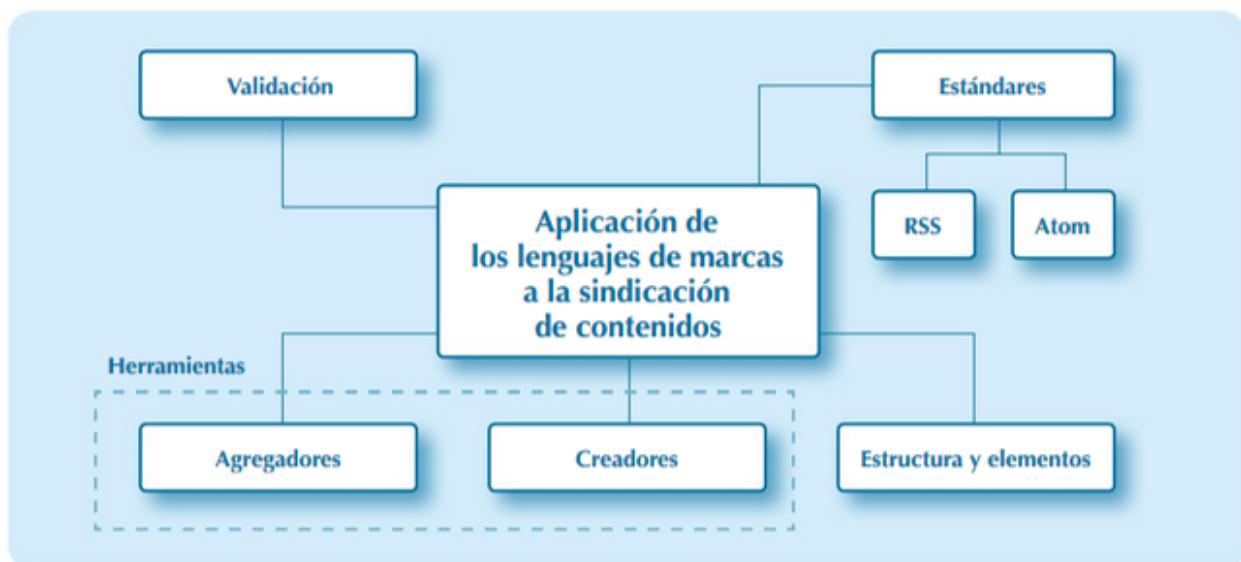


# Aplicación de los lenguajes de marcas a la sindicación de contenidos

## Objetivos

- ✓ Introducirte en la aplicación de los lenguajes de marcas a la sindicación de contenidos, trabajando y profundizando en la tecnología usada.
- ✓ Analizar los estándares usados e identificar la estructura y elementos que lo componen será uno de los primeros aspectos que verás, junto con las ventajas que aporta su uso.
- ✓ La creación y validación será otro pilar fundamental que estudiarás, ya sea de manera manual o con ayuda de software específico, comprobando la funcionalidad y acceso a los canales.
- ✓ Para finalizar, se hace uso de una serie de herramientas que te ayudarán a organizar y leer los distintos canales.

## Mapa conceptual



## Glosario

**Freeware.** Viene de la expresión inglesa *free software*. Consiste en distribuir un *software* con derechos de autor, pero sin costo alguno.

**FTP.** Protocolo de transferencia de archivos descrito en RFC 141 de 1971.

**GUID.** Denominado *identificador único global*. Número implementado por Microsoft que se emplea en algunas aplicaciones *software*. En el contexto de RSS, no hay una regla para su sintaxis.

**Netscape.** Empresa de *software* destacada por el conocido navegador Netscape Navigator.

**RFC.** Sigla de *request for comments*, que significa “petición de comentarios”. Son publicaciones sobre el funcionamiento de internet, redes, protocolos, etc. Se redactan en inglés y tienen un número asignado.

**URI.** Cadena de caracteres que se conoce como identificador de recursos uniforme. El objetivo de dicha cadena es identificar los recursos de manera particular, constando de una serie de elementos claramente diferenciados.

**UserLand.** Compañía de los EE. UU. dedicada al *software* y fundada por Dave Winer en 1988.

**Weblog.** También conocido como *blog*. Sitio web de carácter personal donde los usuarios publican una serie de entradas (anotaciones, historias, artículos, etc.) de manera fácil e intuitiva, sin tener ningún conocimiento de HTML. Presenta una estructura cronológica.

**Web 2.0.** También conocida como *web social*. Compuesta por sitios web que comparten información, donde los usuarios van interactuando y colaborando entre ellos. Concepto surgido en el 2004 que trata de diferenciarse de la web 1.0, compuesta por sitios web tradicionales y sujetos pasivos. Algunos ejemplos son: wikis, blogs, servicios y aplicaciones web, etcétera.

### 3.1. Introducción

Lo más importante cuando se accede a una página web es el contenido que puede encontrarse en ella, principal razón por la que muchos usuarios la visitan. Los internautas, por norma general, tienen que consultar de manera periódica un cierto número de páginas que le interesan y guardarlas en marcadores o favoritos para poder acceder de manera asidua, y de esa manera comprobar las novedades que ofrecen.

La web 2.0 permite a los usuarios confiar, colaborar, comunicarse entre ellos, compartir información y recursos. No tienen el rol de simples consumidores, ahora se considera que son creadores de contenidos y se les permite una gestión eficiente de la información.

La *sindicación de contenidos* también se conoce con el término de *redifusión web* y se usa principalmente para recibir información actualizada de medios de comunicación, titulares de las últimas noticias de un periódico digital, artículos de blog, empresas, etc., de los cuales el usuario está interesado en recibir de manera continuada y actualizada las últimas novedades o noticias. De esa manera, no es necesario que un usuario vaya a buscar la información, sino que la información acude al usuario.

Por norma general, cuando una persona está interesada en conocer las últimas novedades publicadas en la web de diversos sitios, tiene que visitarlas en todo momento para comprobar la información nueva que ha salido. Gracias a la sindicación, los contenidos se distribuyen automáticamente al usuario mediante *canales* de manera similar a como se comprueba el correo electrónico. Para que pueda hacerse uso de esos canales, es necesario un *software* de escritorio o web llamado *agregador* (lector de canales o noticias.) Estos lectores de noticias consultan de manera periódica los canales y acceden a la información actualizada del archivo RSS.



SABÍAS QUE...

Los usuarios que usan canales de sindicación de contenidos y reciben periódicamente sus noticias se denominan *suscriptores*.

Existen diferentes actores que intervienen en la sindicación de contenidos:

- *Autores de los artículos*: son los que van creando los artículos actualizados.
- *Página web*: donde se alojan los artículos.
- *Webmaster*: se encarga de mantener y administrar el sitio web.
- *Contenido creado*: texto, multimedia, etc.
- *Software*: para leer y organizar las noticias.
- *Consumidores de contenidos*: que, simplemente, tienen que filtrar la información que desean obtener.



PARA SABER MÁS

Los gestores de contenidos facilitan los mecanismos de publicación, generando los archivos RSS o Atom de manera automática. Entre los más importantes, destacan Joomla, MediaWiki, WordPress, Drupal y Moodle.

### 3.2. Ventajas y ámbitos de aplicación

Cuando un usuario quiere acceder a las noticias más actuales de diferentes fuentes (recursos educativos, noticias de un periódico, productos comerciales, eventos, blogs, etc.), lo hace escribiendo la URL (*uniform resource locator*) de la página deseada en la barra de direcciones del navegador, mediante el uso de favoritos o marcadores o, simplemente, usando un buscador.

Esto hace que dedique un cierto tiempo en filtrar las noticias que realmente son de su interés.

Gracias a la sindicación de contenidos, los usuarios acceden a la información que les interesa en menor tiempo y filtrando lo que realmente es relevante para ellos.

Aunque existe un número elevado de usuarios que no alcanzan a entender su finalidad (el punto débil de la sindicación es su falta de uso), existe una serie de ventajas que la sindicación de contenidos presenta y se destacan a continuación:

- No es necesario visitar periódicamente la web sobre la que quieren conocerse sus últimas novedades, titulares o noticias, gracias a que los archivos RSS se actualizan automáticamente, produciendo un ahorro de tiempo.
- Se obtiene información más precisa que si se usaran buscadores.
- Permite realizar una búsqueda rápida de noticias.
- Puede hacerse un filtro de la información que desea recibirse, evitando de esa manera que lleguen noticias menos relevantes.
- Se evitan *spam*, virus y publicidad.
- Al usar sindicación de contenidos, aumenta el tráfico de la web que los genera.
- Dar de alta o baja una suscripción se realiza de manera rápida.
- Acceder en tiempo real a titulares de distintas fuentes de noticias.
- Se usa texto plano o simple para el formato de los datos, logrando rapidez a la hora de ser transmitido. Este tipo de documentos carece de formato tipográfico y contiene caracteres legibles por los humanos (letras, ciertos caracteres especiales y de control y números).
- Rapidez a la hora de obtener los resultados lo que permite usarlos en teléfonos móviles, tabletas, etc.
- Aporta un valor añadido al sitio web, ofreciendo el servicio a los usuarios.

#### RECUERDA

- ✓ Un fichero de texto plano carece de formato (*cursiva*, color del texto, negrita, subrayado, tipo de letra, etcétera).

#### • Ámbito de aplicación

Son cada vez más los sitios web que presentan iconos RSS para que los usuarios se suscriban a sus canales y estar de esa manera informados de las últimas novedades. Pueden encontrarse en blogs, revistas y periódicos digitales, organismos oficiales, empresas, bibliotecas y en el ámbito de la educación, entre otros.



SABÍAS QUE...

- ✓ Algunos programas usan RSS para conocer si su última versión ha salido y de esa manera actualizarse.
- ✓ RSS se usa también para distribuir podcasts.

En la actualidad, algunas empresas ofrecen sus agregadores disponibles para varios dispositivos como pueden ser ordenadores personales con distintos sistemas operativos, así como teléfonos móviles.

### 3.3. Tecnologías y estándares de canales de contenidos

Los formatos o estándares más usados para la sindicación de contenidos en la web son RSS y, en menor medida, Atom. Están basados en tecnología XML (*extensible markup language*), lo que permite que la información que se encuentra disponible en algunos sitios web se difundida de manera rápida. En realidad, estos formatos pueden usarse en cualquier web que actualice su contenido de manera periódica.

Para que la información que un autor plasma en su artículo llegue a los consumidores, es necesario disponer de diferentes tecnologías y arquitecturas que sirven de nexo de unión entre ambos. La manera de distribuir contenido es mediante la generación de archivos o *feeds*, siendo los más comunes (XML, RSS, RDF y Atom).

Estándares sindicación de contenidos	
ATOM	RSS

**Figura 3.1**  
Estándares de sindicación de contenidos.



PARA SABER MÁS

El estándar de sindicación de contenidos más usado es RSS.

Por lo general, cuando una web quiere difundir su contenido actualizado, incluye los iconos RSS o Atom con la dirección web del canal.

Un *feed*, también conocido como *fuente web* o *canal*, es un servicio que se encuentra en algunos sitios web o blogs. Suelen situarse en las páginas web para que los usuarios se suscriban al canal. Es un documento que incluye una sintaxis específica que contiene información relativa al autor, título, resumen, enlace, etc., del canal y está codificado en XML. Por lo general, se usan algunos de los iconos que se muestran en la figura 3.2.



**Figura 3.2**  
Iconos usados en sindicación de contenidos.

RECUERDA

- ✓ Un *feed* también se conoce como *fuente web* o *canal*.



### Ejercicio resuelto 3.1

Localiza los iconos de sindicación de contenidos del sitio web que deseas.

**Solución**

#### 3.3.1. RSS

La sindicación realmente simple o RSS (*really simple syndication*) para su versión 2.0 es el estándar usado para distribuir o *sindicar* noticias o información de contenidos que se encuentra en la web. Se considera un lenguaje derivado de XML y define una manera elegante y fácil de compartir contenido web. RSS ha ido evolucionando y cambiando las siglas de significado en las diferentes versiones tal y como se detalla a continuación.

En 1997, nace RSS a manos de Dave Winer (estadounidense desarrollador de *software* y fundador de varias compañías como Living Videotext, UserLand y Small Picture), que diseñó el formato de sindicación de contenidos XML para su weblog.

Netscape desarrolla en 1999 RSS 0.90 y RSS 0.91, sigla de *rich site summary* (resumen óptimo del sitio.)

En el año 2000, la compañía de *software* UserLand de Dave Winer, con sede en los Estados Unidos y fundada en 1988, lanza la especificación oficial de RSS 0.91 (que permite definir hasta 15 ítems dentro de un canal) y RSS 0.92 en el año 2000, similar a la anterior, pero que permite añadir un número ilimitado de ítems dentro del canal.

O'Reilly, en el año 2000, desarrolla RSS 1.0, sigla de *RDF site summary* (resumen RDF del sitio), usando formato RDF y espacios de nombres.



SABÍAS QUE...

La versión RSS 1.0 también se conoce como *RDF*.

Dave Winer desarrolla en el 2002 RSS 2.0 después de dejar UserLand, siendo esta versión la más empleada, fácil de usar y entender.

#### 3.3.2. Atom

Se considera Atom como un formato de sindicación de metadatos y contenido web que, al igual que RSS, se basa en XML. El propósito de su desarrollo fue corregir algunos aspectos de RSS 2.0, mejorando las partes más problemáticas o complejas. Desarrollado por Internet Engineering Task Force (IETF) y publicado en RFC (Request for Comments) 4287 y RFC 5023.

La versión 1.0 apareció en el 2005 y presenta mejoras respecto a RSS en algunas cuestiones técnicas como poder indicar el tipo de noticia (texto plano, HTML o contenido audiovisual.)

**Investiga**

Busca algunas diferencias entre RSS y Atom.



### 3.4. Estructura y elementos de los canales de contenidos

A la hora de crear un formato RSS o Atom, hay que seguir una serie de normas. Cada uno de estos estándares presenta una estructura diferenciada con un número de elementos obligatorios y opcionales que contribuyen a su creación de manera correcta. En este apartado, se estudiarán la estructura y los elementos, poniendo ejemplos de cada uno de ellos.

#### 3.4.1. Estructura y elementos de RSS 2.0

Los ficheros RSS o *feed* RSS que se encuentran en algunos sitios web, por lo general, se crean de manera automática y están compuestos por una sintaxis autodescriptiva y simple, que contiene distintos elementos entre los que destacan:

- Título.
- Resumen.
- Enlace o hipervínculo.
- Autor.

En el siguiente ejemplo se observa la estructura principal de un documento RSS, que no es más que un fichero XML con una serie elementos específicos.

```
<?xml version="1.0" encoding="utf-8"?>
<rss version="2.0">
    <channel>
        <!--en este apartado se agregan las propiedades del canal -->
        <!--opcionales y obligatorios -->
        <item>
            <!--contenido de la 1º entrada-->
        </item>
        <!--contenido de la 2º entrada -->
        <item>
        </item>
        <!--contenido de la última entrada -->
        <item>
        </item>
    </channel>
</rss>
```

Siguiendo como guía la estructura del ejemplo anterior, la primera línea es la declaración XML con los atributos: versión con valor igual a 1.0 y codificación de caracteres usada UTF-8.

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

En la línea número dos, se encuentra la raíz del documento y se declara el tipo por usar. En este caso, indica que es un documento RSS con el atributo versión igual a 2.0, que será la versión usada para todos los ejemplos y ejercicios del capítulo.

```
<rss version="2.0">
```

La raíz contiene un solo hijo etiquetado `<channel>` o *canal*, que se usa para describir la fuente mediante subelementos que aportan información relevante del canal. Dentro de los subelementos o elementos hijo, existen 3 obligatorios y 16 opcionales.

```
<channel>
```

Los elementos vistos hasta ahora se representan en el siguiente ejemplo.

```
<?xml version="1.0" encoding="UTF-8" ?>
<rss version="2.0">
    <channel>
        <!-- contenido -->
    </channel>
</rss>
```

#### RECUERDA

- ✓ Solo tiene que haber un elemento `<channel>` en un documento RSS.

#### Actividad propuesta 3.1



¿Cuántos subelementos obligatorios yopcionales tiene el elemento *canal* de un RSS 2.0?

- a) Tiene 3 subelementos obligatorios y 12 opcionales.
- b) Tiene 1 subelemento obligatorio y 16 opcionales.
- c) Tiene 3 subelementos obligatorios y 16 opcionales.

**Verificar**

#### A) Elementos requeridos de `<channel>`

El elemento `<channel>` tiene tres subelementos, o elementos hijo, directos *obligatorios*:

1. El título o nombre del canal se define con el elemento `<title>` y, en ocasiones, coincide con el nombre del sitio web.
2. El enlace o hipervínculo al canal se define con el elemento `<link>`, introduciendo la URL que apunta a un recurso asociado como una página web siguiendo el esquema de una URI (*uniform resource identifier*).
3. Una descripción corta del canal mediante `<description>`.

En el ejemplo siguiente se aprecia la estructura de los elementos obligatorios del canal.

```
<?xml version="1.0" encoding="UTF-8" ?>
<rss version="2.0">
  <channel>
    <title>Página web LM</title>
    <link>https://www.librolm.com</link>
    <description>página web sobre lenguaje de marcas</description>
  </channel>
</rss>
```

## B) Elementos secundarios `<channel>`

Existe una serie de elementos no obligatorios o secundarios que pueden añadirse al canal, siendo algunos de ellos relevantes para dotar al *feed* de mayor información. Entre ellos, destacan los que se muestran en la figura 3.3.



**Figura 3.3**  
Elementos secundarios del canal RSS.

- *Elemento `<category>`:* se declaran una o varias categorías del contenido para el canal creado, teniendo un único atributo opcional *domain*.

```
<category domain=""> árboles/subtropicales </category>
```

- *Elemento `<docs>`:* se especifica la URL donde se encuentra la documentación del formato usado del *feed*.

```
<docs>http://blogs.law.harvard.edu/tech/rss</docs>
```

- *Elemento <ttl>*: deben especificarse los minutos que el *feed* puede permanecer almacenado en la memoria caché antes de ser actualizado.

```
<ttl>60</ttl>
```

- *Elemento <pubDate>*: se especifica la última fecha de publicación del contenido del *feed* en formato RFC 822.

```
<pubDate>Wed, 19 Dec 2018 15:38:16 +0100</pubDate>
```

- *Elemento <image>*: se especifica el ícono que representa el canal, que puede ser GIF, PNG o JPEG. Tiene tres subelementos obligatorios y tres optionales:

— *Obligatorios*:

- URL de la imagen <url>.
- Título o texto de la imagen <title>, con su descripción. Se usa en el atributo *ALT* de la etiqueta <img> de HTML.
- Enlace al sitio web <link>.

— *Opcionales*:

- Descripción que especifica el texto en el atributo de título HTML del enlace alrededor de la imagen <description>.
- Alto de la imagen. Por defecto, es 31, siendo el valor máximo 400 <height>.
- Ancho de la imagen. Por defecto, el valor es 88, siendo el valor máximo 144 <width>.



SABÍAS QUE...

El ancho y alto de una imagen se especifica en píxeles.

En este ejemplo se muestra una imagen JPG de ancho 88 y alto 50.

```
<image>
  <url>http://www.librolm.com/arboles/imagenes/subt.jpg</url>
  <title>Árboles Subtropicales</title>
  <link>http://www.librolm.com/arboles.aspx</link>
  <description>Árboles Subtropicales</description>
  <width>88</width>
  <height>50</height>
</image>
```

- *Elemento <copyright>*: se especifican los derechos de autor del contenido publicado.

```
<copyright>Copyright 2018 libroLM</copyright>
```

- Elemento `<lastBuildDate>`: se introduce la fecha y la hora de la última modificación del contenido del *feed* en formato RFC 822.

```
<lastBuildDate>Tue, 19 Dec 2017 16:31:42 +0100</lastBuildDate>
```

- Elemento `<rating>`: se especifica el puesto o clasificación PICS (*platform for internet content selection*) del canal. Originalmente, se usaba para que los padres o maestros controlaran el acceso a internet de los niños.



- Elemento `<skipDays>`: se especifican los días en los que el agregador no debe actualizar el *feed*, y pueden especificarse hasta 7 subelementos `<day>` donde se detallan los días de la semana.

```
<skipDays>
  <day>Saturday</day>
  <day>Sunday</day>
</skipDays>
```

- Elemento `<skipHours>`: se especifican las horas en las que el agregador no debe actualizar el *feed*, y pueden introducirse hasta 24 subelementos `<hour>` donde se declaran las horas (de 0 a 23).

```
<skipHours>
  <hour>0</hour>
  <hour>1</hour>
  <hour>2</hour>
  <hour>3</hour>
  <hour>4</hour>
  <hour>5</hour>
  <hour>6</hour>
</skipHours>
```

- *Elemento <generator>*: debe contener una cadena que indica el programa que ha creado el fichero RSS

```
<generator>FeedForAll v2.0 (2.0.4.0)</generator>
```

- *Elemento <language>*: se especifica el lenguaje en el que está escrito el feed. RSS soporta múltiples lenguajes que pueden agregarse en este elemento. Se agrega mediante un código corto que indica el lenguaje empleado en el canal. Debe ser un código de idiomas permitido por el World Wide Web Consortium.

### Ejemplos de códigos de idiomas para las especificaciones de RSS

- *Elemento <cloud>*: se usa para registrar procesos en la nube y recibir notificaciones inmediatas de las actualizaciones del canal. Se especifica un servicio web que admite la interfaz *rssCloud*, que puede implementarse en HTTP-POST, XML-RPC o SOAP 1.1.

```
<cloud domain="rpc.sys.com" port="80" path="/RPC2" registerProcedure="my-
Cloud.rssPleaseNotify" protocol="xml-rpc" />
```

- *Elemento <managingEditor>*: se especifica la dirección de correo del editor para posibles consultas relacionadas con la edición.

```
<managingEditor>editor1@librolm.com</managingEditor>
```

- *Elemento <webMaster>*: se especifica el correo electrónico del *webmaster* o responsable técnico.

```
<webMaster>wb@librolm.com</webMaster>
```

- *Elemento <textInput>*: se especifica un campo de texto de entrada en el canal que contiene cuatro elementos hijo obligatorios:
  1. *<description>*: descripción de la entrada de texto. Máximo 500 caracteres.
  2. *<link>*: URL del script CGI que procesa el ingreso de texto.
  3. *<title>*: etiqueta del botón enviar. Máximo 100 caracteres.
  4. *<name>*: nombre del objeto de texto.


**PARA SABER MÁS**

La mayoría de los agregadores ignoran el elemento <textInput>.

### C) Elementos de <item>

El canal tiene tres elementos hijo directos *obligatorios* y cero o varios elementos <item>, donde se guarda información relativa a artículos o novedades del canal, por lo que este es un elemento de vital importancia.

**RECUERDA**

- ✓ Que un feed no tenga elementos <item> no significa que no sea válido.

- *Elemento <item>*: se especifica el contenido visible del canal y consta de 10 subelementos opcionales. Hay que reseñar que existen tres elementos secundarios recomendados, que son título, enlace y descripción de la noticia o artículo. Aunque antes se ha mencionado que todos los elementos son opcionales, es preciso que el título o descripción de un ítem aparezca para que este se valide correctamente.



**Figura 3.4**  
Elementos de <item>.

- *Elemento <title>*: título del ítem.
- *Elemento <link>*: se especifica el hipervínculo del ítem mediante URL.

```

<item>
  <title>XHTML</title>
  <link>https://www.librolm.com/xhtml.html</link>
</item>
  
```

- *Elemento <guid>*: se introduce una cadena que actúa como identificador global único para cada artículo. Contiene un atributo opcional con valor verdadero si se especifica una URL.

```
<guid isPermaLink="false">DC1EF2AE-7B32-4C1E-9989-EF4E090238AB</guid>
```

- *Elemento <enclosure>:* añade un archivo multimedia y tiene tres atributos:

1. URL del elemento multimedia que ha de agregarse (*url*).
2. Tamaño en *bytes* del archivo (*length*).
3. Tipo del archivo multimedia (*type*).

```
<item>
  <title>polinización del chirimollo</title>
  <enclosure url="http://www.librolm.com/media/polinizacion.wmv" length="6215" type="video/x-ms-wmv" />
</item>
```

- *Elemento <author>:* el contenido del elemento es una dirección de correo del autor del artículo.

```
<author>autor1@librolm.com</author>
```

- *Elemento <source>:* especifica el canal RSS del que proviene el artículo y contiene un atributo obligatorio (*url*).

```
<item>
  <title>Cultivo del Longan</title>
  <source url="http://www.cultivossubtropicales.com/longan.xml">
    cultivo longan Florida
  </source>
</item>
```

- *Elemento <comments>:* se especifica una URL para establecer comentarios del artículo. Suele usarse en weblogs.

```
<item>
  <title>Cultivo del longan</title>
  <comments>http://www.librolm.com/arboles/foro.aspx</comments>
</item>
```

- *Elemento <pubDate>:* se especifica la última fecha de publicación del ítem.

```
<pubDate>Tue, 19 Dec 2018 15:31:35 +0100</pubDate>
```

- Elemento <category>: es igual que la categoría del canal, pero se especifica la categoría de cada ítem, y pueden definirse varias.

```
<item>
    <title>Cultivo del Zapote</title>
    <category>subtropicales</category>
    <category>Rutaceae</category>
</item>
```

- Elemento <description>: el contenido de este elemento suele ser una breve descripción o resumen del artículo, las primeras líneas, primeros párrafos, o artículo completo, aunque esta última opción es menos usual. El formato puede ser texto plano o usar HTML, este último puede ser introducido mediante dos maneras diferentes: la primera, codificando los caracteres < y > por &lt; y &gt;, y la segunda, usando un bloque CDATA, donde se pone una palabra de la descripción en negrita usando la etiqueta <strong>.

A continuación se muestra un código HTML en el que se usa un bloque CDATA, donde se pone una palabra de la descripción en negrita usando la etiqueta <strong>.

```
<item>
    <title>Cultivo del Zapote</title>
    <description><![CDATA[En este apartado se trata la siembra, fertilización y poda del <strong>Zapote</strong>]]></description>
    <link>http://www.librolm.com/arboles/zapote.aspx</link>
</item>
```



**Figura 3.5**  
Visualización del contenido de un ítem en un agregador.

### Ejercicio resuelto 3.2



Crea una descripción de un ítem que contenga una palabra en negrita usando codificación de la entidad HTML.

**Solución**

**RECUERDA**

Al derivar RSS de XML, hay que tener en cuenta que:

- Todos los elementos deben tener una etiqueta de cierre.
- Los elementos son sensibles a las mayúsculas.
- Los elementos deben estar anidados correctamente.
- Los valores del atributo siempre deben citarse.

En el ejemplo siguiente se presenta un RSS 2.0 con algunos de sus elementos.

```
<?xml version="1.0" encoding="UTF-8" ?>
<rss version="2.0">
<channel>
    <title>Página web LM</title>
    <link>https://www.librolm.com</link>
    <description>página web sobre lenguaje de marcas</description>
    <item>
        <title>CSS</title>
        <link>https://www.librolm.com/css.html</link>
        <description>Tutorial rss</description>
    </item>
    <item>
        <title>XML</title>
        <link>https://www.librolm.com/xml.html</link>
        <description>Tutorial xml</description>
    </item>
    <item>
        <title>XHTML</title>
        <link>https://www.librolm.com/xhtml.html</link>
        <description>Tutorial xhtml</description>
    </item>
</channel>
<!-- se han añadido tres items -->
</rss>
```

#### • Comentarios RSS

Como ya se ha indicado en el capítulo 2, una línea de comentario sería:

<!-- comentario -->

**Ejercicio propuesto 3.1**

Tomando como base un RSS del capítulo, añade más elementos opcionales al canal y a los ítems.

**RECUERDA**

- ✓ Si quiere añadirse al elemento *descripción* de un ítem contenido en formato HTML sin ser procesado, se usa el siguiente bloque:

<![CDATA[...]]>

**Actividades propuestas**

Responde si las siguientes afirmaciones son verdaderas o falsas y razona tu respuesta:

**V F 3.2.** En RSS 2.0, <skipDays> es un elemento secundario de <item>.

**V F 3.3.** En RSS 2.0, <category> puede ser un elemento secundario de <item> y <channel>.

**Verificar**

### 3.4.2. Estructura y elementos de Atom

La estructura para un fichero ATOM 1.0 es similar a la de uno RSS:

```
<?xml version="1.0" encoding="utf-8"?>
<feed xmlns="http://www.w3.org/2005/Atom">
    <!--en este apartado se agregan los subelementos del Feed -->
    <!--opcionales y obligatorios y recomendados -->
    <entry>
        <!-- elementos entrada 1 -->
    </entry>
    <entry>
        <!-- elementos entrada 2 -->
    </entry>
</feed>
```

A la hora de crear un documento Atom 1.0, se construirá con un elemento raíz <feed>, que contiene metadatos y entradas. La raíz tiene que tener siempre el atributo *xmlns*, que define el espacio de nombre de este tipo de documentos con el valor “<http://www.w3.org/2005/Atom>”, para que pueda ser validado.

Hay que tener en cuenta que los elementos donde el usuario introduce un texto como <title>, <subtitle>, <right>, <content> y <sumary> contienen un atributo opcional *type* con tres posibles valores (“text”, “html” y “xhtml”), siendo el primero de ellos el valor por defecto.

### A) Elementos de <feed> requeridos, recomendables y opcionales

El elemento <feed> es similar al elemento <channel> de RSS y está compuesto por una serie de metadatos y entradas que pueden ser obligatorios, opcionales y recomendables.

- *Elemento requerido <id>*: identifica el canal utilizando una URI, que puede ser un dominio personal.

```
<id>http://www.librolm.com/</id>
```

- *Elemento requerido <title>*: Título del canal, que debe ser obligatorio y legible para los usuarios. Normalmente, se corresponde con el nombre de la página web pertinente.

```
<title>árboles tropicales</title>
```

- *Elemento requerido <updated>*: indica la última vez que se ha modificado el feed de una manera significativa. Deben cumplir con el formato de fecha RFC 3339.



SABÍAS QUE...

- ✓ El formato de fecha es Año-Mes-DíaTHora:Minutos:Segundos zona horaria
- ✓ Para la zona horaria, puede usarse el carácter Z, que indica hora universal.
- ✓ Se usa el carácter T para separar el día de la hora.

```
<updated>2018-04-13T10:25:02Z</updated>
```

- *Elemento recomendado <link>*: se especifica una página web relacionada. Tiene un atributo obligatorio y variosopcionales que se detallan a continuación:

- *Atributo obligatorio href*: se especifica mediante una URI el recurso o página web usada.
- *Atributo opcional rel*: puede contener una URI completa con un valor predefinido, que puede ser alguno de la siguiente lista (*alternate*, *enclosure*, *related*, *self* y *via*), siendo el primero de ellos el valor por defecto.
- *Atributo opcional type*: indica el tipo de recurso.
- *Atributo opcional hreflang*: se especifica el idioma del recurso al que se hace referencia.
- *Atributo opcional title*: título.
- *Atributo opcional length*: se especifica en *bytes* el tamaño del recurso.

```
<link rel="self" type="application/atom+xml" hreflang="es"
      href="http://librolm.com/atom/downloadservice.xml" title="Enlace al mismo
      documento"/>
```

- *Elemento recomendado <author>*: especifica el autor, empresa o entidad del *feed*. Contiene varios elementos hijo:
  - *Elemento requerido <name>*: nombre del autor o entidad.
  - *Elemento opcional <email>*: correo electrónico del autor o empresa.
  - *Elemento opcional <uri>*: página web de la empresa o autor.

```
<author>
  <name>autor1</name>
  <email>autor1@librolm.com</email>
  <uri>http://www.librolm.com/</uri>
</author>
```

- *Elemento opcional <subtitle>*: se especifica una descripción o un subtítulo legible para el usuario. Tal y como se ha mencionado con anterioridad, contiene un atributo opcional *type* con tres posibles valores (“text”, “html” y “xhtml”).

```
<subtitle>Árboles tropicales en Málaga</subtitle>
```

- *Elemento opcional <rights>*: permite añadir derechos de autor del *feed*. Contiene un atributo opcional *type* con tres posibles valores (“text”, “html” y “xhtml”), siendo el primero de ellos el valor por defecto.

```
<rights> © 2007 TutorialsPoint.com </rights>
```

- *Elemento opcional <contributor>*: se especifica el nombre de un colaborador o empresa del *feed*. Hay que reseñar que pueden especificarse varios colaboradores. Al igual que el elemento *<autor>*, contiene varios elementos hijo:
  - *Elemento requerido <name>*: nombre del colaborador o entidad.
  - *Elemento opcional <email>*: correo electrónico del colaborador o empresa.
  - *Elemento opcional <uri>*: página web de la empresa o colaborador.

```
<contributor>
  <name>José Antonio López López</name>
</contributor>
```

- *Elemento opcional <generator>*: se especifica el software utilizado para crear el canal. Contiene dos atributos opcionales (“uri” y “version”).

```
<generator uri http://www.miweb.com/ " version="1.0">herramienta usada</generator>
```

- *Elemento opcional <logo>*: se añade una imagen que proporciona una identificación visual para el canal.

```
<logo>/mi_logo.jpg</logo>
```

- *Elemento opcional <icon>*: se especifica una imagen pequeña que identifique el canal.

```
<icon>/mi_icono.jpg</icon>
```

- *Elemento opcional <category>*: se especifica la categoría a la que pertenece el feed. Hay que tener en cuenta que, dentro de un <feed>, pueden especificarse múltiples elementos de este tipo. Dispone de tres atributos (*term*, *scheme* y *label*), siendo el primero de ellos obligatorio y el resto,opcionales.

```
<category term="categoría del feed" />
```

## B) Elementos de <entry> requeridos, recomendados yopcionales

Las entradas dentro de un canal se definen mediante el elemento <entry>. Un canal puede contener uno o más elementos de entrada:

- *Elemento requerido <id>*: identificador de la entrada mediante URI.

```
<id>http://www.librolm.com/1978</id>
```

- *Elemento requerido <title>*: se agrega el título de la entrada. Este valor no debe estar en blanco.

```
<title>El cultivo del aguacate</title>
```

- *Elemento requerido <updated>*: se especifica la fecha de última modificación de la entrada. Cada una de ellas tendrá un valor diferente para este elemento.

```
<updated>2018-01-19T19:25:03-05:00</updated>
```

- *Elemento recomendado <content>*: se especifica el contenido de la entrada mediante un hipervínculo o escribiéndolo directamente. Debe proporcionarse si no hay un enlace alternativo o si no se encuentra el elemento <summary>.

```
<content>contenido </content>
```

- *Elemento opcional <summary>*: se especifica un resumen o extracto de la entrada.

```
<summary>resumen de la entradasummary>
```

- *Elemento opcional <rights>*: se especifican los derechos de autor de la entrada. Al igual que otros elementos estudiados, contiene un atributo opcional *type* con tres posibles valores (“text”, “html” y “xhtml”).

```
<rights type="html">Todos los derechos reservados &lt;em&gt; LibroLm.com &lt;/em&gt; </rights>
```

- *Elemento opcional <contributor>*: se especifica el nombre de uno o varios colaboradores o empresas de la entrada. Al igual que el elemento <autor>, contiene varios elementos hijo, siendo el primero de ellos obligatorio.
  - *Elemento <name>*: nombre del autor o entidad.
  - *Elemento <email>*: correo electrónico del autor o empresa.
  - *Elemento <uri>*: página web de la empresa o autor.

```
<contributor>
  <name>Francisco González</name>
  <email>francisco.gonzalez@librolm.com</email>
</contributor>
```

- *Elemento opcional <published>*: se especifica la fecha de creación de la entrada.

```
<published>2018-01-13T12:55:10Z</published>
```

- *Elemento opcional <source>*: se especifican metadatos si la entrada es una copia.

```
<source>
  <id>http://www.miweb.com/</id>
  <title>cultivo de aguacates en Colombia</title>
  <updated>2018-05-20T12:34:05Z</updated>
  <rights>© 2018 todos los derechos reservados MiWeb.</rights>
</source>
```

- *Elemento recomendado <author>*: se especifica al autor o autores de la entrada. Dispone de un elemento obligatorio <name> y dos opcionales (<uri> e <email>), donde se añaden la página web y el correo electrónico del autor.

```
<author>
  <name>José A. López López</name>
  <email>jose.lopez@librolm.com</email>
</author>
```

- *Elemento recomendado <link>*: se especifica la URL de la entrada. El tipo de relación se define mediante el atributo *rel*. Es similar al elemento <link> del <feed> visto con anterioridad.

```
<link rel="alternate" type="text/html" href="http://www.librolm.com/conte-
nidol" hreflang="es" />
```

- *Elemento opcional <category>*: se especifican una o varias categorías a las que pertenece la entrada. Tiene tres atributos (*term*, *scheme* y *label*), siendo el primero de ellos obligatorio y el resto, opcional.

```
<category term="deporte" />
```

A continuación se presenta un documento Atom con algunos elementos.

```
<?xml version="1.0" encoding="utf-8"?>
<feed xmlns="http://www.w3.org/2005/Atom">
    <title>Árboles subtropicales</title>
    <subtitle>cultivo del aguacate</subtitle>
    <link href="http://www.librolm.com/" />
    <updated>2018-01-14T22:45:58Z</updated>
    <author>
        <name>José A. González</name>
        <email>jose.gonzalez@librolm.com</email>
    </author>
    <contributor>
        <name>Carmen Ruiz</name>
    </contributor>
    <id>http://www.librolm.com/</id>
    <entry>
        <title>Aguacates en la Axarquía</title>
        <author>
            <name>José A. López López</name>
            <email>jose.lopez@librolm.com</email>
        </author>
        <link href="http://librolm.com/aguacates/001/a/atom001"/>
        <id>http://www.librolm.com/aguacate</id>
        <updated>2018-01-15T08:40:25Z</updated>
        <summary>Este cultivo se da en la zona de...</summary>
        <contributor>
            <name>Francisco Pascual</name>
            <email>francisco.pascual@librolm.com</email>
        </contributor>
    </entry>
</feed>
```

### Ejercicio propuesto 3.2

Siguiendo como base la estructura detallada, crea un documento Atom que contenga algunos de los elementos vistos en este apartado.



### Actividades propuestas

Responde si las siguientes afirmaciones son verdaderas o falsas y razona tu respuesta:

**V** **F**

**3.4.** A la hora de crear un documento Atom 1.0, se construirá con un elemento raíz `<tree>` que contiene metadatos y entradas.

**V** **F**

**3.5.** En Atom 1.0, el elemento recomendado `<author>` del `<feed>` especifica el autor, empresa o entidad y contiene, entre otros, el elemento hijo `<email>`.

**Verificar**

### 3.5. Creación, validación y comprobación de funcionalidades de los canales de contenidos

Por lo general, en un blog, gestor de contenidos o wiki, los ficheros RSS se generan de manera automática. Muchos *feeds* se crean usando lenguajes como PHP, Python, Perl, etc. Una vez creados, es recomendable validarlos para comprobar que la estructura y elementos son los correctos.

#### 3.5.1. Creación de un feedRSS

Si desea crearse un *feed* de manera manual, existen varias opciones: la primera de ellas es usar uno de los programas vistos en el capítulo, donde, mediante una serie de ventanas, el usuario tiene que ir llenando los campos del canal o de los distintos ítems y, al finalizar, se genera el *feed* RSS. La segunda opción es crearlo desde cero, usando un editor de código XML como puede ser Notepad++.

Para esta segunda opción, se tendrán en cuenta la estructura y los elementos de RSS. De manera genérica, los pasos que han de seguirse para RSS son

1. Saber qué temática y contenido va a incluirse en el canal, así como el número de artículos o ítems que van a componerlo. Es conveniente emplear un poco de tiempo para organizar de manera adecuada el contenido que ha de crearse.
2. Crear un documento XML con formato RSS en un editor de texto plano o usar una plantilla.
3. Especificar la etiqueta de inicio de XML y la versión de RSS que va a usarse.
4. Añadir datos relativo al canal, obligatorios u opcionales como el título, URL del canal, imagen, descripción del mismo.
5. Insertar los artículos o ítems que se deseen, incluyendo en cada uno elementos como GUID, autor, comentarios, descripción, título, etc.
6. Guardar el *feed* RSS con la extensión .rss o .xml, siendo la primera de ellas la más usada.
7. Validar el *feed* para comprobar que cumple con las especificaciones. Para ello, pueden usarse algunas de las técnicas que se muestran en el apartado 3.5.2.
8. Publicar el *feed* subiendo el documento creado a un sitio web, mediante FTP o un gestor de archivos. No hay que olvidarse de que, al hacerlo de manera manual, hay que actualizar el *feed* cada vez que quiera agregarse nuevo contenido.



#### SABÍAS QUE...

Si quiere enlazarse un RSS a una imagen del sitio web, hay que insertar la siguiente línea de código HTML, para ello se usan los elementos <img> y <a> con sus respectivos atributos:

```
<a type="application/rss+xml" href="canal.rss"> </a>
```

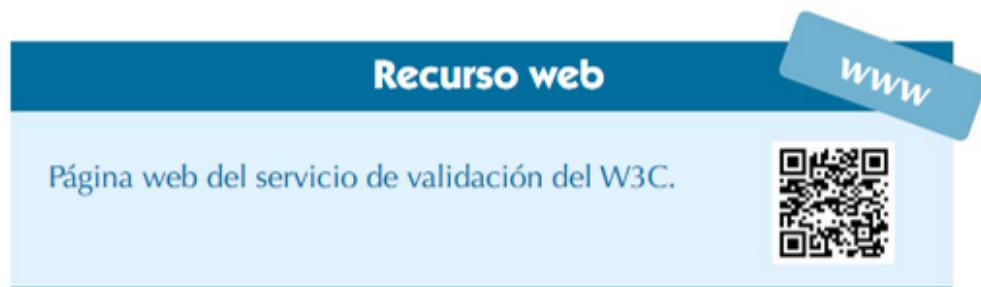
#### 3.5.2. Validación y comprobación

A la hora de crear manualmente o mediante herramientas un documento RSS, es recomendable validarlos para comprobar que todo es correcto, para ello se dispone de varios métodos.

Para comprobar su funcionalidad, una vez validado y subido el *feed* al servidor, simplemente, hay que usar uno de los agregadores descritos en este capítulo y comprobar que el contenido se visualiza de manera correcta.

### A) Servicio de validación online del W3C

Es un servicio gratuito que verifica la sintaxis de las fuentes RSS o Atom, lo que permite su validación por URI o por entrada directa, escribiendo el código.



El fichero RSS que ha de validarse, modificando algunas líneas para comprobar posibles errores, es el siguiente:

```
<?xml version="1.0" encoding="UTF-8" ?>
<rss version="2.0">
    <channel>
        <title>Página web LM</title>
        <link>https://www.librolm.com</link>
        <description>página web sobre lenguaje de marcas</description>
        <item>
            <title>CSS</title>
            <link>https://www.librolm.com/css.html</link>
            <description>Tutorial rss</description>
        </item>
        <item>
            <title>XML</title>
            <link>https://www.librolm.com/xml.html</link>
            <description>Tutorial xml</description>
        </item>
        <item>
            <title>XHTML</title>
            <link>https://www.librolm.com/xhtml.html</link>
            <description>Tutorial xhtml</description>
        </item>
    </channel>
    <!-- se han añadido tres items -->
</rss>
```

En la figura 3.6, se ha usado la validación por entrada directa, pegando el código en la ventana que presenta. Se pulsa sobre la opción Check y se obtiene el resultado de la validación. Si hay un fallo, aparece el número de línea y columna donde se encuentra el error. Puede obtenerse una explicación sobre cómo arreglarlo si se pulsa el enlace *help*.

El resultado obtenido no es un documento validado, ya que se han cambiado las líneas 3 y 4 del fichero por validar, tal y como puede apreciarse.

### Sorry

This feed does not validate.  
[line 3, column 0: Undefined rss element: channl \[help\]](#)  
<channl>  
[line 4, column 23: XML parsing error: <unknown>:4:23: mismatched tag \[help\]](#)  
<titl>Página web LM</titl>

### Source:

```

01.  <?xml version="1.0" encoding="UTF-8" ?>
02.  <rss version="2.0">
03.  <channl>
04.  [<titl>Página web LM</titl>
05.  <link>https://www.librolm.com</link>
```

**Figura 3.6**

Informe de errores de la página web del servicio de validación del W3C.

Se arreglan las líneas 3 y 4 cambiando <channl> por <channel> y <titl> por <title> y vuelve a validarse. En este caso, se comprueba que es correcto.

Además de mostrar los posibles errores, puede ofrecer algunos mensajes de avisos o recomendaciones sobre compatibilidad, como definir un elemento <guid> dentro de una entrada. Otra recomendación que da es añadir un espacio de nombres y un enlace Atom.

### Ejemplos

## B) Feed Validator

Validador de canales que funciona para RSS y Atom. Las versiones que soporta para RSS son RSS 0.90, 0.91, 0.92, 0.93, 0.94, 1.0 y 2.0. A la hora de añadir la URL, si el validador encuentra cualquier problema de sintaxis, resaltarán dónde se produce e indicará una posible solución.

Mensaje de error

### Sorry

This feed does not validate.

- [line 23, column 1143: Unexpected type attribute on thumbnail element \(38 occurrences\) \[help\]](#)  
... "366" type="image/jpeg"></media:content><media:thumbnail url="http://e00 ...

**Figura 3.7**

Mensaje de error del servicio de validación *online* de Feed Validator.

**Recurso web**

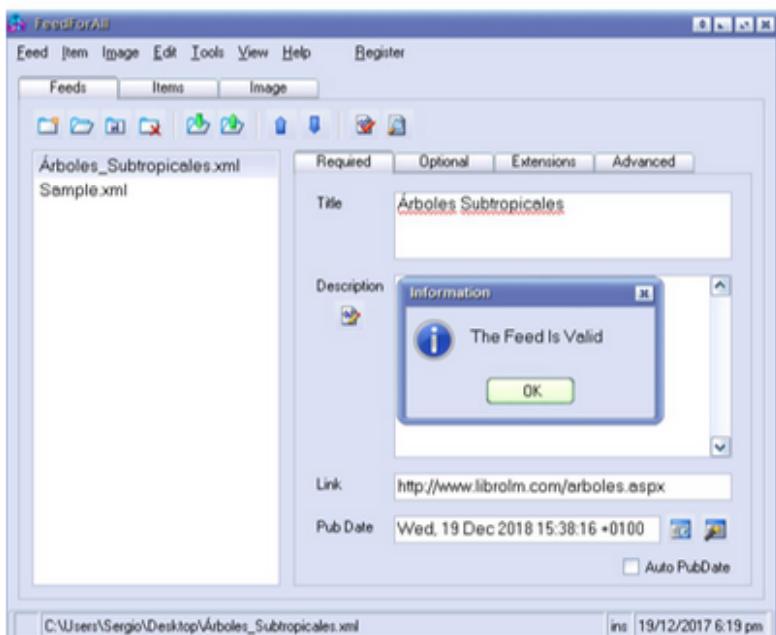
Página web del servicio de validación de Feed Validator.

www



### C) Herramienta software FeedForAll

Algunos programas de creación de canales de contenidos traen la opción de validar tal y como se aprecia en el programa FeedForAll, que la incorpora dentro del menú Herramientas.



**Figura 3.8**  
Validación usando el software FeedForAll.

### Ejercicio propuesto 3.3

Comprueba si un documento RSS que no contenga elementos <ítem> puede ser válido.



## 3.6. Utilización de herramientas

Existen diversas herramientas que permiten crear de manera fácil y amena feeds RSS sin tener conocimientos previos de XML. Para ello, presentan una serie de campos que el usuario tiene que ir llenando.

Una vez creados los canales RSS, hay que disponer de herramientas lectoras de feeds o agregadores para poder hacer uso de ellos.

Gracias a los feeds RSS, se incrementa el número de lectores de una página web manteniendo a los usuarios informados de todas las novedades.

### 3.6.1. Herramientas de creación y edición de feeds RSS

A la hora de crear los ficheros RSS, existen varias posibilidades: que los gestores de contenidos los elaboren de manera automática, que el usuario los cree de manera manual o usando herramientas específicas para tal fin, que será las vistas en este apartado.



### PARA SABER MÁS

WordPress ya viene preparado para la sindicación de contenidos y permite feeds RSS 1.0, 0.92, 2.0 y Atom.

#### A) FeedForAll

Software de evaluación para la creación, edición, validación y publicación de contenidos mediante RSS. Presenta una interfaz fácil de usar a la hora de crear nuevos canales RSS 2.0 y, mediante el uso de un asistente, irá mostrando cada uno de los campos que han de llenarse o, usando la ventana principal, las diferentes zonas con los elementos por cumplimentar.

¿Cómo usar FeedForAll?

#### Ejercicio resuelto 3.3



Usando la herramienta FeedForAll, crea un canal con distintos ítems usando los elementos que estimes oportuno. Muestra el código resultante.

Solución

#### B) RSS Builder

Herramienta *freeware* desarrollada por Win Bokkers y usada para crear ficheros RSS. Presenta un diseño elegante e intuitivo que es muy fácil de usar, ya que los distintos elementos se añaden mediante campos que el usuario tiene que ir llenando (enlaces, título del canal, autor, etc.). Para poder publicar el RSS en la página web, incorpora un cliente FTP, además de un editor HTML para el campo Descripción.

¿Cómo usar RSS Builder?

#### Ejercicio resuelto 3.4



Usando la herramienta RSS Builder, crea un fichero con distintos ítems, usando los elementos que estimes oportuno. Muestra el código resultante.

Solución

**Ejercicio propuesto 3.4**

Usando la herramienta RSS Builder, elabora un RSS con tres ítems de la temática que deseas.

**C) ListGarden**

Programa de código abierto escrito en lenguaje Perl, de fácil uso para crear y mantener feeds RSS. Al contrario que los vistos con anterioridad, usa una interfaz ejecutada sobre un navegador web.

No requiere conocimientos previos de XML o formato de RSS y está disponible para Windows, Mac OS X y Linux. El documento XML resultante puede almacenarse en local o remotamente mediante FTP. Puede usarse para feeds RSS de podcast, registros de cambios en una web, weblog feed, etc.

**¿Cómo trabajar con ListGarden?****Ejercicio resuelto 3.5**

Usando la herramienta ListGarden, crea un fichero con distintos ítems, usando los elementos que estimes oportuno. Muestra el código resultante.

**Solución****D) Absolute RSS Editor**

Programa para la creación y edición de fuentes RSS 2.0 y podcasts para iTunes. Sin conocimientos de sintaxis XML, pueden rellenarse los campos que van apareciendo en las distintas ventanas. Incorpora una herramienta de validación que informa de posibles errores que puedan surgir. Una vez terminado un feed RSS, puede obtenerse una vista previa del código XML y publicar en la web.

**E) RSS Feed Creator**

Este software de evaluación, desarrollado por Webvigour, permite crear y editar ficheros RSS para agregar a la página web. Dispone de un editor de canales RSS y una opción para crear podcasts. Presenta un diseño visual e intuitivo a la hora de usarlo, que permite a los usuarios crear sus propios ficheros RSS sin tener conocimientos de XML.

www

## Recursos web

- ✓ A través de los siguientes enlaces podrás acceder a las páginas de descarga de FeedForAll, ListGarden, Absolute RSS Editor RSS Feed Creator (usa la versión 2.0):



### Actividades propuestas



Responde si las siguientes afirmaciones son verdaderas o falsas y razona tu respuesta:

- V F 3.6.** La herramienta RSS Builder incluye un editor HTML para el elemento *descripción*.
- V F 3.7.** Feed Validator es un validador de canales que funciona para RSS y Atom.

**Verificar**

### 3.6.2. Agregadores o lectores de feeds

Un agregador o lector de RSS o *feeds* es un *software* que se usa para leer y administrar noticias de un blog, medios de comunicación, revistas, sitios web, etc., a las cuales un usuario se ha suscrito previamente. El lector de *feed* recoge las novedades de las páginas web que el usuario previamente ha configurado y son de su interés.

En la actualidad, la gran mayoría de navegadores presentan un lector de fuentes RSS, pero existe una serie de herramientas de escritorio que permiten la suscripción de fuentes en los formatos vistos (Atom y RSS).

El agregador actúa como intermediario entre el usuario y las páginas que difunden las noticias. Para poder acceder a su contenido, hay que indicarle la dirección web del archivo fuente en los formatos correspondientes.



#### SABÍAS QUE...

Los agregadores actualizan los cambios que se producen en la web de manera transparente para el usuario.



**Figura 3.9**  
Fuentes para los agregadores.

Los agregadores permiten: organizar las noticias de manera personalizada, buscar información y controlar los contenidos que han sido leídos (de manera similar a como se consulta el correo electrónico).

Existen varios tipos de agregadores y usar uno u otro depende de las necesidades que cada usuario tenga.

### A) Agregadores online o web

Los agregadores en línea son aplicaciones alojadas en sitios web. La principal ventaja que presentan es que no hace falta instalar una aplicación para poder usarlos. Otra de las ventajas es que puede accederse a ellos desde cualquier dispositivo que disponga de una conexión a internet.



SABÍAS QUE...

Google Reader fue un lector de referencia, pero cerró en el año 2013.

- *Feedly*: aplicación web, aunque también disponible para dispositivos móviles, que permite gestionar feeds, accediendo al contenido actualizado de la web en diferentes formatos. Requiere de un registro previo que puede realizarse con una cuenta propia del programa, de Facebook o de Google, entre otras. Admite suscripción a múltiples sitios web para poder consultar sus últimas novedades. Ofrece la posibilidad de visualizar la información como si fuera una revista, lista de artículos o tarjetas.
- *Inoreader*: lector RSS y Atom disponible como aplicación web y para dispositivos móviles. Desarrollado en el 2013 por Innologic y programado en PHP, presenta una interfaz amena para el usuario. Requiere un registro previo, que puede realizarse con una cuenta de Facebook, Google o un registro en la misma plataforma. Permite etiquetar y organizar de manera automática los artículos, presentando suscripciones ilimitadas de manera gratuita.
- *Digg Reader*: aplicación web disponible también para dispositivos móviles. De fácil manejo, presenta dos modos de lectura de las noticias. Permite el registro con Facebook, Twitter y cuenta de Google.

#### ¿Cómo usar Digg Reader?



- *The Old Reader*: es un agregador de noticias muy fácil de usar y rápido creado por Anton Tolchanov, Elena Bulygina y Dmitry Krasnoukhov en el 2013. Es gratuito si no se llega a los 100 feeds. Si desean tenerse más, existe una versión premium con mejores características. Existe una aplicación para móviles gratuita y para las plataformas más importantes

#### ¿Cómo usar The Old Reader?



www

## Recursos web

Con estos enlaces podrás acceder a Feedly, Inoreader, Digg Reader y The Old Reader.



### Ejercicio propuesto 3.5

Suscríbete a la web que deseas usando un agregador *online*.

### B) Agregadores de escritorio

Los agregadores de escritorio son aplicaciones que hay que instalar sobre un sistema operativo de un ordenador personal, aunque algunos de ellos permiten su instalación en otros dispositivos como tabletas o teléfonos móviles. Presentan una interfaz gráfica muy intuitiva, separada por bloques, similar a la de los programas de correo electrónico.

Por un lado, aparecen las suscripciones que un usuario ha realizado y, por otro, una zona donde se accede al contenido de estas para su lectura.

Algunos de ellos se detallan a continuación:

- *NewsFire*: lector de noticias RSS para Mac Os X que permite feeds RSS o Atom.
- *RSSReader*: *freeware* de agregación y sindicación de contenido para leer noticias RSS 0.9x, 1.0 y 2.0 y Atom 0.1, 0.2 y 0.3, disponible para sistemas operativos Windows. Trabaja en segundo plano para recopilar las noticias.
- *NewzCrawler*: *software* de versión trial que incluye lector de RSS y Atom. Agregador de noticias disponible para sistemas operativos Windows.
- *FeedDemon*: lector de RSS para sistemas operativos Windows que presenta una interfaz de usuario muy amigable. Creado y desarrollado por Nick Bradbury, organiza las suscripciones en etiquetas o carpetas, permitiendo eliminar así RSS antiguos que quedan obsoletos.

#### ¿Cómo usar FeedDemon?

- *FeedReader*: completo lector de noticias compatible con RSS y Atom que dispone de versiones *online* y de escritorio.

#### ¿Cómo usar FeedReader?

- *Liferea (Linux Feed Reader)*: es un agregador de noticias de *software* libre bajo licencia GNU para Linux. Es compatible con los formatos Atom, RSS y RDF que soporta po-

*dcasts*. Es fácil de usar para escritorio GNOME. Al igual que otros programas similares, permite cambiar la frecuencia de actualización de los canales.

#### Ejercicio resuelto 3.6

Instala Liferea en Linux y agrega varios canales.

**Solución** 



#### Ejercicio propuesto 3.6

Añade una fuente de archivo local en Liferea y comprueba que se visualiza de manera correcta.



- *Omea Reader*: lector de RSS y Atom gratuito desarrollado por JetBrains, que, además de ser un agregador de grupos de noticias, actúa como gestor de tareas y marcadores, libreta de direcciones, etc. De fácil uso, permite una perfecta clasificación y organización de los canales añadidos. El software presenta tres paneles claramente diferenciados: el panel de la izquierda, donde aparecen los canales que el usuario va añadiendo; el central, donde se muestran las noticias de los canales, y el panel derecho, donde se abre el contenido de cada una de las noticias.

#### Recursos web

*www*

Con estos enlaces podrás acceder a NewsFire, RSSReader, NewzCrawler, FeedDemon, FeedReader, Liferea y Omea Reader.



#### Ejercicio resuelto 3.7

Agrega dos RSS en Omea Reader.

**Solución** 



### C) Agregadores en navegador web

Gracias a las últimas versiones de los navegadores más importantes, puede accederse al contenido de los RSS sin instalar *software* adicional. Algunos ejemplos se muestran a continuación:

- *Mozilla Firefox*: al igual que otros navegadores, permite la gestión de canales RSS. Cuando un usuario localiza un ícono de canal de noticias RSS o Atom y pulsa sobre él, aparece una nueva ventana que permite que el usuario pueda suscribirse.

#### Suscripción a un canal con Mozilla Firefox

- *Opera*: este navegador permite agregar fuentes RSS y Atom relacionadas con noticias de revistas, periódicos u otras que contengan información de interés para el usuario, recibiendo resúmenes de las distintas noticias de un modo similar a recibir correos electrónicos.

#### Suscripción a un canal con Opera

#### Ejercicio resuelto 3.8



Utiliza el navegador Internet Explorer para agregar una fuente RSS.

#### Solución



#### Ejercicio propuesto 3.7

Suscríbete a la página web que deseas usando Opera.

### D) Agregadores en clientes de correo electrónico

El uso de clientes de correo electrónico está muy extendido en la actualidad, pues permite acceder al correo sin tener que abrir el navegador. Además de poder consultar el correo, algunos clientes permiten la suscripción a canales de noticias RSS.

Uno de los clientes de correo más conocidos es Outlook, pero existen otras alternativas como Mozilla Thunderbird y Zimbra Desktop. La suscripción a web de noticias, blogs o cualquier otro contenido puede realizarse mediante los programas citados anteriormente. Gracias a la suscripción, el usuario estará informado de las últimas novedades que vayan publicándose.

#### 1. Canal RSS en Outlook Express

En la actualidad, existe una multitud de fuentes RSS que están compuestas por titulares, un breve resumen y un enlace al lugar de origen, entre otros datos.

**Agregar una fuente RSS en Outlook Express Opera** **2. Canal RSS en Mozilla Thunderbird**

Mozilla Thunderbird es una aplicación de correo gratuita de Mozilla que, entre sus características, dispone de un lector RSS, filtros de correo y detección de *spam*, así como grupos de noticias integrados.

**Configurar canales RSS en Mozilla Thunderbird** **Ejercicio propuesto 3.8**

Usando Mozilla Thunderbird, agrega el siguiente canal:  
[http://estaticos.elmundo.es/elmundo/rss/union\\_europea.xml](http://estaticos.elmundo.es/elmundo/rss/union_europea.xml)

**Recurso web**  
www

Consulta el enlace para aprender cómo suscribirse a noticias y blogs con Mozilla Thunderbird.

**3.7. Directorios de canales de contenidos**

Cuando quiere darse a conocer un canal de noticias u otro tipo de documento RSS, puede incluirse dicho canal en directorios específicos, permitiendo así que los ficheros RSS estén disponibles para los usuarios. Uno de los requisitos que hay que cumplir es registrar el RSS en el directorio, lo que permite clasificar y realizar búsquedas de canales de manera eficiente.

**Recurso web**  
www

Accede a Buscazoom, un ejemplo de directorio web de noticias y canales RSS.



**Actividades propuestas**

Responde si las siguientes afirmaciones son verdaderas o falsas y razona tu respuesta:

**V F 3.8.** Un agregador también puede denominarse *herramienta de diseño*.

**V F 3.9.** Mozilla Thunderbird es una aplicación de correo gratuita de Mozilla que, entre sus características, dispone de un lector RSS.

**Verificar****Resumen**

- Ventajas y ámbitos de aplicación de los lenguajes de marcas a la sindicación de contenidos:
  - Información más precisa.
  - Búsqueda rápida de noticias.
  - Valor añadido al sitio web.
  - Alta o baja de suscripción rápida.
  - Aumenta el tráfico de la web.
  - Se evitan *spam*, virus y publicidad.
  - Uso en distintos dispositivos.
- Tecnologías y estándares de canales de contenidos:
  - RSS.
  - Atom.
- Estructura y elementos de los canales de contenidos:
  - Estructura RSS:

```
<?xml version="1.0" encoding="utf-8"?>
<rss version="2.0">
  <channel>
    <!--en este apartado se agregan las pripiedades del canal -->
    <!--opcionales y obligatorios -->
    <item>
      <!--contenido de la 1º entrada-->
    </item>
    <!--contenido de la 2º entrada -->
    <item>
    </item>
    <!--contenido de la última entrada -->
    <item>
    </item>
  </channel>
</rss>
```

- Estructura Atom:

```
<?xml version="1.0" encoding="utf-8"?>
<feed xmlns="http://www.w3.org/2005/Atom">
    <!--en este apartado se agregan los subelementos del Feed -->
    <!--opcionales y obligatorios y recomendados -->
    <entry>
        <!-- elementos entrada 1 -->
    </entry>
    <entry>
        <!-- elementos entrada 2 -->
    </entry>
</feed>
```

- Elementos RSS:

- Elementos hijo de <channel>:

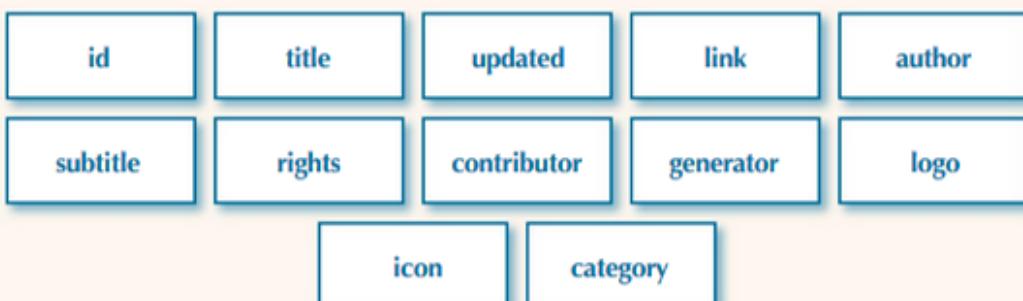


- Elementos hijo de <item>:



- Elementos Atom:

- Elementos hijo de <feed>:



- Elementos hijo de <entry>:



- Creación y validación de *feeds*:
  - Creación en editor de texto plano.
  - Validación:
    - Servicio de validación *online* del W3C.
    - Feed Validator.
    - Herramienta software FeedForAll.
- Herramientas de creación y edición de *feeds RSS*:
  - FeedForAll.
  - RSS Builder.
  - ListGarden.
  - Absolute RSS Editor.
- Agregadores o lectores de *feeds*:
  - Agregadores online o web.
  - Agregadores de escritorio.
  - Agregadores en navegador web.
  - Agregadores en clientes de correo electrónico.

## Ejercicios prácticos resueltos

1. Crea un fichero RSS en Notepad++ que contenga tres ítems.

**Solución**

2. Elabora una tabla comparativa que contenga algunos elementos de Atom 1.0 y RSS 2.0.

**Solución**

3. Usando la herramienta Absolute RSS Editor, genera un fichero con distintos ítems, usando los elementos que estimes oportuno. Muestra el código resultante.

**Solución**

## Ejercicios prácticos



1. Accede a varias páginas web y localiza el enlace RSS o Atom. ¿En qué parte de la página se encuentra? ¿Trae los dos formatos?
2. Localiza un documento RSS de la web y comenta su estructura y elementos que lo componen.
3. Especifica los elementos obligatorios y, por lo menos, tres secundarios de <channel> para un fichero RSS, poniendo un ejemplo de cada uno.
4. Crea en Notepad++ un documento RSS donde el agregador no actualice el feed los viernes, sábados ni domingos.
5. Utilizando Notepad++, genera un documento RSS con una descripción de ítems donde se use HTML.
6. Valida, usando algunas de las herramientas vistas, el siguiente documento RSS, enumerando y corrigiendo los posibles fallos hasta que el documento sea válido:

```
<?xml version="1.0" encoding="utf-8"?>
<rss version="2.0">
<channel>
    <title>Manual LM</title>
    <link>http://www.librolm.com</link>
    <description>Apuntes XML ciclos ASIR DAM DAW</description>
    <language>es-es</language>
    <webMaster1>wm@librolm.com</webMaster1>
    <item>
        <title>Esquema HTML</title>
        <author>autor2@librolm.com</author>
        <category>informática</category>
    </item>
    <item>
        <title>Esquema XHTML</title>
        <author>autor1@librolm.com</author>
        <category>informática</category>
    </item>
    <item>
        <title>Esquema XML</title>
        <author>autor@librolm.com</author>
        <category>informática</category>
    </item>
</channel>
</rss>
```

7. Crea un documento Atom que contenga algunos elementos vistos en el capítulo.
8. Elabora un fichero RSS sobre la temática que deseas, usando el programa RSS Builder y muestra el contenido del fichero generado.
9. Instala un agregador de escritorio y agrega algunas fuentes.

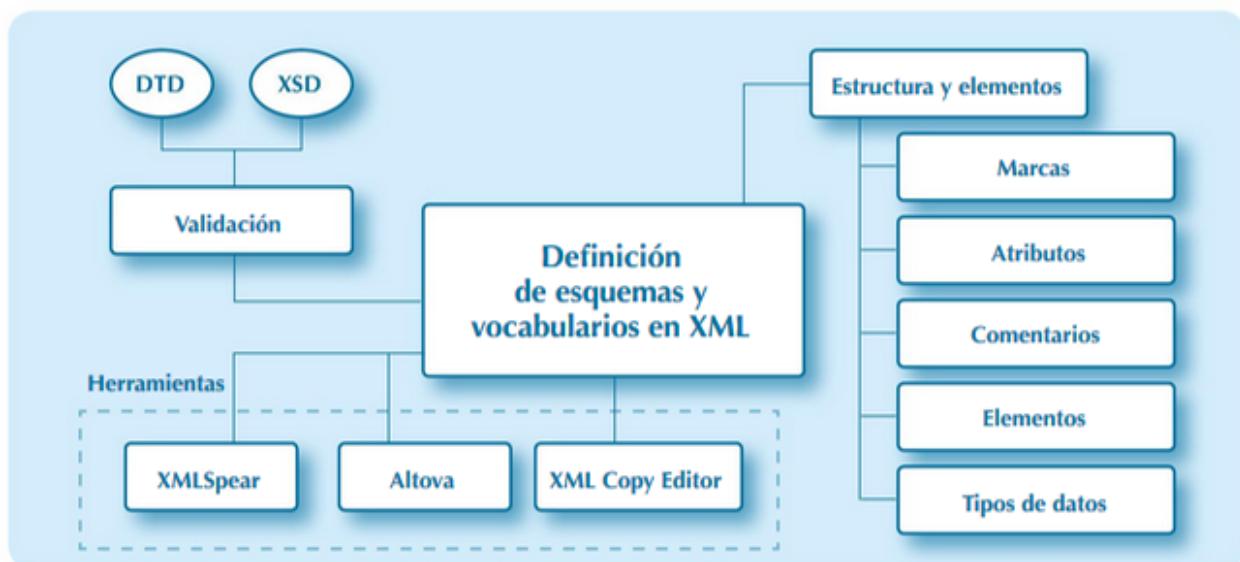


# Definición de esquemas y vocabularios en XML

## Objetivos

- ✓ Introducirte en el manejo de XML, analizando, creando, definiendo y profundizando en su estructura y sintaxis.
- ✓ La creación de definiciones de tipos de documentos y esquemas es el siguiente aspecto que vas a estudiar, analizando su sintaxis y estructura, así como viendo las técnicas de asociación entre XML y DTD o XSD.
- ✓ Es importante que aprendas a analizar las herramientas específicas que permiten trabajar con XML, así como validar documentos, crear sus esquemas y comprobar si están bien formados, así como trabajar con características más avanzadas.
- ✓ También es fundamental que sepas cómo validar mediante herramientas *online* o de escritorio los XML, asociando la definición de tipo de documento o esquemas.

## Mapa conceptual



## Glosario

**JSON.** Sigla de *JavaScript object notation*. Formato de texto ligero que se usa para el intercambio de datos de fácil interpretación.

**Schematron.** Lenguaje de validación XML inventado por Rich Jelliffe.

**SGML.** Sigla de *standard generalized markup language*. Estándar para lenguajes de marcado.

**SOAP.** Sigla de *simple object access protocol*. Protocolo de comunicación entre objetos mediante internet. Basado en XML.

**Solaris.** Sistema operativo Unix propiedad de Oracle para arquitecturas SPARC y x86.

**WSDL.** Sigla de *web services description language*. Lenguaje de descripción de servicios web. Se basa en XML.

**XPath.** Sigla de *XML Path Language*. Lenguaje usado para procesar documentos XML.

**XQuery.** Lenguaje de consultas de datos XML.

**XSLT.** También conocido como *transformaciones XSL (extensible stylesheet language)*, es un lenguaje que se usa para transformar documentos XML en HTML o XHTML.

### 4.1. Introducción

En el presente capítulo, se estudiarán aspectos relacionados con XML (*extensible markup language*), tecnología que sirve como formato de almacenamiento e intercambio de información.

Se usa para almacenamiento de datos, actualización de *software*, intercambio de información, sindicación de contenidos, etc.

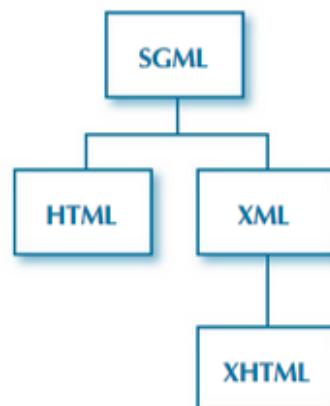
Se verá su estructura y sintaxis, instrucciones de procesamiento, marcas, elementos y atributos, así como otros aspectos relevantes. Verificar que están escritos correctamente y son válidos son aspectos fundamentales a la hora de trabajar con estos documentos, ya que la validación comprueba que está bien formado siguiendo una serie de reglas y respetando, además, las normas que pueden encontrarse en la DTD (*document type definition*) o XSD (*XML schema definition*).

XML deriva de SGML (*standard generalized markup language*), siendo legible y comprensible para todos, lo que permite que usuarios sin conocimientos de lenguajes de marcas sean capaces de interpretar qué información es la que se almacena en los documentos. Mientras que las marcas de un documento HTML no aportan información relevante a los usuarios, las de XML transmiten el significado del contenido que se almacena.

Tanto para la creación como para la validación y edición, existen multitud de herramientas gratuitas, comerciales y de evaluación que permiten trabajar de manera fácil y amena con este tipo de documentos. Algunas de ellos, además de facilitar la construcción de documentos XML, permiten generar esquemas y validar y comprobar que están bien formados, así como trabajar con XSLT (*extensible stylesheet language*), XQuery, etc.

Gracias a las DTD o XSD, pueden definirse las restricciones y tipos de los atributos y elementos, así como el número de veces que aparecen o los posibles valores por defecto que pueden tomar. Estos esquemas facilitan el trabajo colaborativo al poder trabajar de manera conjunta conociendo su esquema, lo que permite, además, la validación de los documentos XML.

Aunque un documento DTD es más fácil de crear que uno XSD y siguen creándose en la actualidad, presenta una serie de limitaciones: no permite especificar restricciones como rangos concretos, número de caracteres que deben aparecer, expresiones regulares, etc. No es posible especificar tipos de datos fecha o entero positivo, no está basado en XML, no comprueba los tipos, pero puede añadirse cualquier valor en un elemento llamado *DNI*, tal y como se muestra en el ejemplo siguiente.



**Figura 4.1**  
Jerarquía SGML.

```

<?xml version="1.0" encoding="iso-8859-1"?>
<usuario>
  <fechaNac>2018-10-25</fechaNac>
  <DNI>fran@librolm.com</DNI>
  <nombre>Francisco Jesús González</nombre>
</usuario>
  
```

## 4.2. Definición de la estructura y sintaxis de documentos XML

Tal y como se mencionó en el capítulo 1, XML es un lenguaje de marcas extensible cuyas siglas vienen de la expresión inglesa *extensible markup language*. Se considera un metalenguaje y

surge para resolver los problemas que plantea HTML, pues las etiquetas muestran el significado de sus datos.

Para poder visualizar un documento XML, puede hacerse directamente sobre el navegador, usando hojas de estilo CSS, transformaciones XSLT, etc.

La importancia radica en el intercambio de información de manera segura entre distintos programas, permitiendo así la reutilización de contenido, crear etiquetas propias y presentar una estructura y diseño independientes.



#### SABÍAS QUE...

Un documento XML debe crearse en texto plano.

La estructura de un documento XML es jerárquica arborescente, que contiene a un padre (raíz) único y una serie de (hijos) elementos secundarios. La raíz se sitúa en la parte superior, colgando el resto de elementos de él.

La extensión de este tipo de documentos es .xml.

```
<?xml version="1.0" encoding="utf-8"?>
<padre>
    <hijo>
        <subhijos> ... </subhijos>
    </hijo>
</padre>
```

En todo documento, se distinguen dos partes claramente diferenciadas:

1. Por un lado, se encuentra el *prüfago*, que contiene información respecto al documento creado (versión, codificación de caracteres, descripción de estructura, etc.).
2. Por el otro, se encuentra el *elemento raíz o cuerpo del documento*, que tiene que ser único y sobre el que están contenidos los demás elementos.



#### PARA SABER MÁS

Un elemento raíz no tiene ascendientes ni hermanos.



#### Ejercicio propuesto 4.1

Busca por internet un documento XML y analiza su estructura.

#### 4.2.1. Instrucciones de procesamiento o prólogo

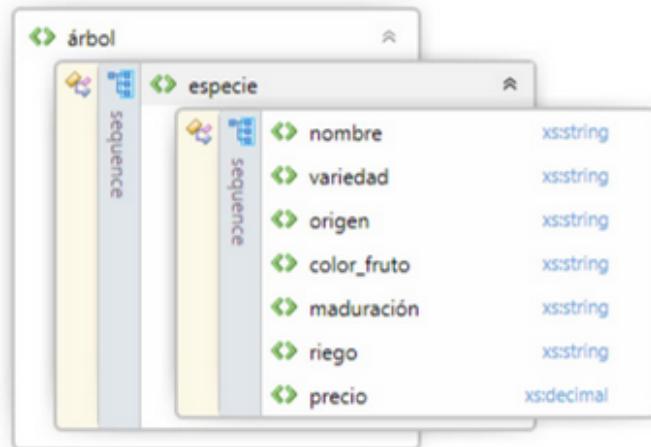
La primera línea por la que deben empezar los documentos XML es el prólogo, compuesto por la *versión* del documento, que indica la versión XML usada, *encoding* que informa de la codificación del documento, con posibles valores (UTF-8, ISO 8859-1, etc.), y puede ser modificado en cualquier momento. Otro posible atributo que puede contener dicha línea es *standalone*, que informa de si el documento lleva asociado un *fichero DTD o XSD*, con valores posibles “yes” o “no”.

Dichas instrucciones comienzan por los caracteres <? y terminan con los caracteres ?>. En la segunda línea, se encuentra la raíz (<árbol>).

En el caso de asociar una DTD a un XML, la segunda línea sería la definición del tipo de documento.

En el ejemplo siguiente se aprecia un XML que contiene valores sobre especies de árboles, almacenando información como nombre, variedad, origen, etc.

```
<?xml version="1.0" encoding="iso-8859-1"?>
<árbol>
<especie>
    <nombre>Litchi</nombre>
    <variedad>Kway May Pink</variedad>
    <origen>Filipinas</origen>
    <color_fruto>rojo</color_fruto>
    <maduración>agosto</maduración>
    <riego>diario</riego>
    <precio>3.5</precio>
</especie>
</árbol>
```



**Figura 4.2**  
Estructura de árbol del documento XML usando Visual Studio.

#### Ejercicio propuesto 4.2

Añade dos especies más al ejemplo del documento XML anterior.



### Ejercicio resuelto 4.1



Dado el siguiente XML, usa otro programa que permita visualizar el contenido del XML en forma de árbol:

```
<?xml version="1.0" encoding="iso-8859-1"?>
<árbol>
  <especie>
    <nombre>Litchi</nombre>
    <variedad>Kway May Pink</variedad>
    <origen>Filipinas</origen>
    <color_fruto>rojo</color_fruto>
    <maduración>agosto</maduración>
    <riego>diario</riego>
    <precio>3.5</precio>
  </especie>
</árbol>
```

**Solución**

#### 4.2.2. Marcas o etiquetas

En un documento XML, el usuario puede definir sus propias marcas o etiquetas con la finalidad de guardar la información de la manera que estime oportuna, separando el contenido de la estructura.

Las marcas son el pilar fundamental de estos lenguajes. Se usan los signos menor que (<) y mayor que (>) para definirlas. Los nombres de las marcas deben describir los datos que contienen. Cabe destacar que, en XML, no existe un número finito de marcas, sino que, para poder realizar una representación de datos adecuada, cada usuario puede definir las marcas que estime oportunas.



#### SABÍAS QUE...

Las marcas no pueden contener espacio en blanco, comillas simples o dobles, punto y coma, signo de porcentaje, etc.

Existen tres tipos de etiquetas o marcas:

1. Las de apertura como <casa>.
2. Las de cierre como </casa>, que usan el carácter barra (/) para indicar fin de marcado.
3. Etiquetas vacías, siendo estas últimas las que carecen de contenido como <casa/>.

Viendo el ejemplo siguiente, las marcas pueden ser las siguientes:

- De apertura <nombre> <variedad>.
- De cierre </nombre> </variedad>.

```
<nombre>Litchi</nombre>
<variedad>Kway May Pink</variedad>
```

A continuación se observan ejemplos de marcas no válidas, la primera de ellas al contener un espacio en blanco y la segunda, una barra (/).

```
<nombre producto>Litchi</nombre producto>
<variedad/precio>Kway May Pink</variedad/precio>
```

Las marcas son sensibles a las mayúsculas y minúsculas. Para XML de apertura de la etiqueta <casa> y de cierre </Casa>, son distintas, ya que una empieza por mayúsculas y otra por minúsculas.



SABÍAS QUE...

*Case sensitive* es una expresión que se aplica a los textos en informática para indicar que no es lo mismo escribir un carácter en mayúsculas que en minúsculas.



### Actividades propuestas

Responde si las siguientes afirmaciones son verdaderas o falsas y razona tu respuesta:

V F 4.1. ¿Es XML un *case sensitive*?

V F 4.2. XML deriva de XHTML.

Verificar

### 4.2.3. Elementos

Compuestos por etiquetas de inicio, de fin, escritas con el mismo nombre y todo el contenido que haya entre ambas (ya sea texto u otros elementos). En el cuadro 4.6, un elemento puede ser <nombre>Litchi</nombre>, cuyas partes son las siguientes:

- Etiqueta de apertura: <nombre>.
- Etiqueta de cierre: </nombre>.
- Contenido del elemento: Litchi.
- Elemento nombre: <nombre>Litchi</nombre>.

Hay que resaltar que el elemento *especie* del ejemplo de la página 131 contiene 7 elementos hijo o subelementos: nombre, variedad, origen, color, fruto, maduración, riego y precio.

Aunque no es muy normal, pueden existir elementos vacíos, sin contenido alguno como el siguiente:

```
<casado></casado>
```

Un elemento vacío puede ser uno que no tenga etiqueta de cierre con o sin atributos:

```
<teléfono n="666777888" />
```

#### Ejercicio propuesto 4.3



Cuenta los elementos que hay en el siguiente documento XML:

```
<feed xmlns="http://www.w3.org/2005/Atom">
    <title>Árboles subtropicales</title>
    <subtitle>cultivo del aguacate</subtitle>
</feed>
```

Hay que tener en cuenta una serie de consideraciones a la hora de crear un elemento:

- Los elementos tienen que estar anidados correctamente unos dentro de otros, cerrándose en orden inverso al que se abren.
- Los elementos no vacíos tendrán etiquetas de apertura y cierre.
- Los nombres de los elementos y atributos tienen que seguir una nomenclatura específica como no empezar por número, no usar caracteres especiales reservados, etc. Tienen que empezar por letras o los caracteres guion bajo (\_) o dos puntos (:), seguidos de guiones, puntos, números u otras letras. Tampoco pueden empezar por la palabra XML.

#### 4.2.4. Atributos

Se usan para asignar propiedades asociadas a los elementos. Se sitúan dentro de una etiqueta de inicio o de apertura. Se especifican con el par NombreAtributo="valor", pudiendo añadirse varios separados por un espacio en blanco. Los valores tienen que estar encerrados entre comillas dobles o simples y no podrá haber dos atributos llamados iguales en un mismo elemento.

Dichos atributos proporcionan información adicional sobre los elementos. Aunque los atributos pueden usarse perfectamente en XML, no es recomendable abusar de ellos, ya que un uso excesivo podría hacer que el documento fuese menos entendible para el usuario.

En el siguiente ejemplo, el elemento *nombre* tiene el atributo *dni* cuyo valor es "77777777J".

```
<nombre dni="77777777J"> AlumnoLM </nombre>
```

## Ejercicio propuesto 4.4

Crea un elemento que contenga dos atributos.



## Ejercicio resuelto 4.2



Dado el siguiente fichero XML de una agenda personal, transforma los elementos *ciudad* e *email* en atributos.

```
<?xml version="1.0" encoding="i-
so-8859-1"?>
<agenda>
  <contacto nombre="alumno">
    <ciudad>Málaga</ciudad>
    <teléfono n="666777888" />
    <email>a@librolm.com</email>
  </contacto>
</agenda>
```

**Solución**

#### 4.2.5. Comentarios

Los comentarios sirven de ayuda cuando el documento es extenso o se pretende que todo se entienda con claridad. Pueden introducirse en cualquier parte, excepto dentro de las etiquetas, otros comentarios o declaraciones. Tienen la misma sintaxis que los comentarios HTML, comenzando por `<!--` y terminando con `-->`.

En el ejemplo siguiente, la línea 3 sería un comentario.

```
<?xml version="1.0" encoding="utf-8" ?>
<agenda>
  <!--vamos a crear un contacto-->
  <contacto nombre="alumno LM">
    <ciudad>Málaga</ciudad>
  </contacto>
</agenda>
```

#### 4.2.6. Sección CDATA

Esta sección permite añadir contenido sin ser procesado o analizado. Deben aparecer dentro del elemento raíz de un documento XML. Es similar a un comentario cuya sintaxis es la siguiente:

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

A continuación se muestra un ejemplo de XML con sección CDTA:

```
<?xml version="1.0" ?>
<asignaturas>
    <foro>
        <![CDATA[el <b>examen</b>, será el día]]>
    </foro>
</asignaturas>
```

### Actividades propuestas



Responde si las siguientes afirmaciones son verdaderas o falsas y razona tu respuesta:

**V F 4.3.** Los atributos se añaden dentro de la etiqueta de cierre.

**V F 4.4.** Los comentarios en XML tienen la misma sintaxis que los comentarios en C#.

**Verificar**

### Ejercicio resuelto 4.3



Es preciso mandar una tabla de manera segura y fiable a un programa cliente que recoge información sobre unos árboles subtropicales: variedad, color de fruto, maduración, etc.

Transforma el siguiente cuadro sobre las distintas especies en un documento XML que almacene dicha información:

Nombre	Variedad	Ciudad origen	Color del fruto	Maduración	Riego	Precio kg
Litchi.	Kway may pink.	Filipinas.	Rosa.	Agosto.	Diario.	3,50 €
Longan.	Champoo.	China.	Marrón.	Octubre.	Diario.	2,5 €
Litchi.	Mauritius.	Florida.	Rojo.	Agosto.	Diario.	3 €

**Solución**

### 4.3. Creación, asociación y elementos de DTD y XSD

A la hora de especificar una serie de restricciones a un documento XML, existen varias opciones que un usuario puede usar. La primera de ellas es mediante DTD, que suele ser poco flexible en las definiciones de cada uno de los elementos, ya que no es posible indicar si se trata de un elemento de tipo fecha, moneda, número etc. Estas carencias que presenta DTD se

superan usando XML Schema, siendo estas dos primeras opciones las que se estudien en este apartado. Por otra parte, existen alternativas como el uso de Relax NG (*regular language for XML next generation*) y Schematron.

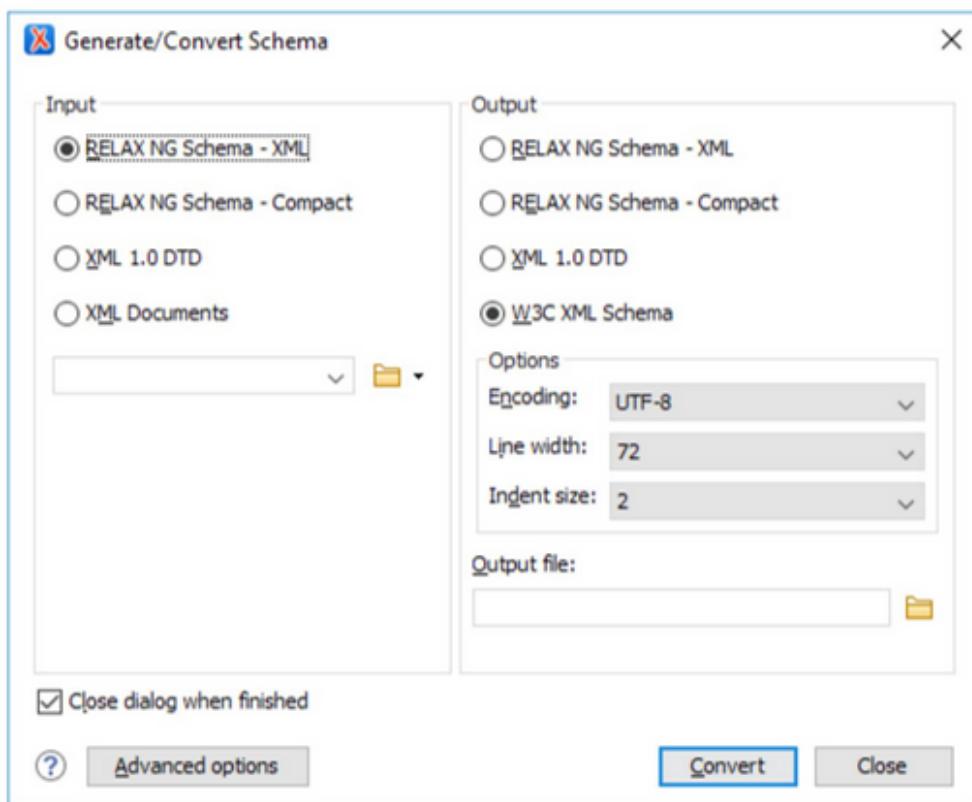
Todas las opciones anteriormente vistas son métodos de validación de documentos XML, por lo que debe asociarse el fichero de restricciones al XML para que se vinculen de manera correcta.



#### SABÍAS QUE...

Trang es un programa de código abierto que se emplea para la generación de XML Schema y DTD mediante un documento XML. También permite convertir de DTD a XSD, y viceversa.

Existen herramientas como Oxygen XML Editor (capítulo 5) que permite, entre otros aspectos, la conversión entre distintos esquemas.



**Figura 4.3**  
Conversión de esquemas Oxygen XML Editor.

Existen generadores de esquemas de escritorio (Oxygen XML Editor, XMLSpy, Visual Studio, XML Copy Editor, etc.) y *online* que facilitan obtener la DTD o SXD de manera automática, ahorrando así al usuario la tediosa tarea de crear XSD o DTD de documentos XML extensos.

En la figura 4.4, se aprecia una herramienta *online* donde se han añadido etiquetas XML, lo que permite obtener documentos RNC, RNG, DTD y XSD.

**XML Schema Generator**

The XML Schema Generator creates a basic, easily adapted XML schema from an XML file.

Paste the contents of your XML file below: \*

```
<?xml version="1.0" encoding="iso-8859-1"?>
<árbol>
  <especie>
    <nOMBRE>Litchi</nOMBRE>
    <variedad>Kway May Pink</variedad>
    <origen>Filipinas</origen>
    <color_fruto>rojo</color_fruto>
    <maduración>agosto</maduración>
    <riego> diario</riego>
    <precio>3.5</precio>
  </especie>
  <especie>
    <nOMBRE>Longan</nOMBRE>
    <variedad>Champoo</variedad>
    <origen> China</origen>
    <color_fruto>marrón</color_fruto>
```

Output Format: \*

RNG  RNC  DTD  XSD

**Generate**

This XML Schema Generator accepts one XML document and infers a schema.

It produces as output a schema written in any of the following formats:

- **RNG:** RELAX NG (XML syntax)
- **RNC:** RELAX NG (compact syntax)
- **DTD:** XML 1.0 DTD
- **XSD:** W3C XML Schema

**Figura 4.4**

Generador de esquemas *online* usando XML Schema Generator.

### 4.3.1. DTD (definición de tipo de documento)

Definición de tipo de documento o DTD es un documento de texto plano donde se especifican o definen una serie de *normas* que se usan para saber cómo tiene que crearse un fichero XML para poder validarla. Gracias a este tipo de documentos, puede especificarse el número de ocurrencias de un elemento XML y el tipo de datos, así como el orden de precedencia, entre otros aspectos.

Aunque no es obligatoria la creación de una DTD, es recomendable su uso si la información del XML va a ser compartida. De esa manera, los usuarios que trabajen con el documento XML sabrán, gracias al DTD, su definición o reglas a cumplir.

Su sintaxis es distinta a la de XML, pues no pueden definirse tipos de datos por parte de los usuarios ni insertar elementos no ordenados.

**RECUERDA**

- ✓ Para crear un documento XML, es recomendable previamente crear una DTD, donde se especifiquen los elementos que intervienen.

## A) Creación y elementos de una DTD

Una vez que se han determinado las reglas que servirán como base para la generación de documentos XML, es el momento de empezar a construir la DTD. Para ello, se estudiarán tres componentes principales que lo forman (elementos, atributos y entidades).

Para crear este tipo de documentos, puede usarse un editor de texto plano, teniendo en cuenta que la extensión de estos documentos será .dtd.



**Figura 4.5**  
Componentes DTD.

### 1. Elementos

Los elementos de una DTD son la base de su estructura. Lo primero que habría que mirar es si existe dependencia jerárquica, especificando el nodo padre y, entre paréntesis, sus hijos si los tuviera, separados por el carácter coma (,) tal y como se aprecia a continuación.

```
<!ELEMENT nombreElemento (elementos_hijo1, elemento_hijo2, ...)>
```

Por otra parte, cada uno de los elementos debe ser de un tipo específico, pudiendo distinguir entre *any*, *empty*, *#pcdata* o *#cdata*. Se especifica tal y como se muestra en el ejemplo siguiente.

```
<!ELEMENT nombreElemento Tipo>
```

El *tipo* puede ser:

- (*#PCDATA*): *parsed character data*, que indica que el contenido de ese elemento es de tipo texto y es analizado por un *parser*.
- (*#CDATA*): *character data*, similar a PCDATA con la obviedad de que el contenido de los elementos no es analizado por un *parser* y no pueden detectarse posibles entidades.
- *ANY*: los elementos pueden contener cualquier valor.
- *EMPTY*: los elementos no tienen contenido. Elemento vacío.



SABÍAS QUE...

Un *parser* es un analizador o procesador XML que comprueba si el documento está bien formado o es válido, analizando su estructura.

Para plasmar lo visto, se verá el ejemplo a continuación del cuadro 4.1 donde quiere crearse un documento DTD para el envío de correos electrónicos que contiene los campos (receptor, asunto y cuerpo).

**CUADRO 4.1**  
Campos de un *email*

Receptor	Asunto	Cuerpo
r1@librolm.com	Cumpleaños	Desearte un feliz...
r2@librolm.com	Día libre	Estimado señor X,...

Un fichero DTD para un XML podría ser:

```
<!ELEMENT bdcorreo (email)+>
<!ELEMENT email (receptor,asunto,cuerpo)>
<!ELEMENT receptor (#PCDATA)>
<!ELEMENT asunto (#PCDATA)>
<!ELEMENT cuerpo (#PCDATA)>
```

## ■ Ocurrencia de aparición de los elementos

Si quiere especificarse que un elemento puede aparecer en el XML cero, una o varias veces, es necesario contar con una nomenclatura específica para plasmar lo que se desea. Para ello, se usa una serie de caracteres que informa de la obligatoriedad o no, así como del número de veces que puede aparecer dicho elemento.

**CUADRO 4.2**  
Caracteres de ocurrencia de los elementos

Carácter	Descripción
?	El elemento puede aparecer cero o una vez (opcional).
*	El elemento puede aparecer cero, una o varias veces (opcional).
+	El elemento debe aparecer una o más veces (obligatorio).
	Similar al (OR) lógica, indica que puede añadirse uno de los dos elementos que se especifiquen, pero no ambos (obligatorio):

```
<!ELEMENT NombreE (hijo1, (hijo2 | hijo3)) >
```

En el ejemplo siguiente, se observa el uso de los caracteres más (+) y pleca (|).

Se ha creado un XML que almacena componentes electrónicos donde puede aparecer el elemento precio o precio con IVA (solo uno de los dos). Para ello, se ha añadido (**precio|precioiva**), que indica que puede añadirse cualquier elemento que haya en la lista. Por otro lado, el signo + informa de que el elemento debe aparecer una o más veces.

```

<?xml version="1.0" encoding="iso-8859-1"?>
<!DOCTYPE BDcomponentes [
<!ELEMENT BDcomponentes (componente)+>
<!ELEMENT componente (nombre,(precio|precioiva),tipo+,medida)>
<!ELEMENT nombre (#PCDATA)>
<!ELEMENT precio (#PCDATA)>
<!ELEMENT precioiva (#PCDATA)>
<!ELEMENT tipo (#PCDATA)>
<!ELEMENT medida (#PCDATA)>
]>
<BDcomponentes>
  <componente>
    <nombre>resistencia</nombre>
    <precio>0.25</precio>
    <tipo>pasivo</tipo>
    <tipo>bobinadas</tipo>
    <medida>ohmios</medida>
  </componente>
  <componente>
    <nombre>condensador</nombre>
    <precioiva>0.75</precioiva>
    <tipo>pasivo</tipo>
    <medida>Faradios</medida>
  </componente>
</BDcomponentes>

```

Si ahora se modifica el ejemplo anterior y no se añade al componente el elemento *tipo*, no será válido e informará del error.

```

<?xml version="1.0" encoding="iso-8859-1"?>
<!DOCTYPE BDcomponentes [
<!ELEMENT BDcomponentes (componente)+>
<!ELEMENT componente (nombre,(precio|precioiva),tipo+,medida)>
<!ELEMENT nombre (#PCDATA)>
<!ELEMENT precio (#PCDATA)>
<!ELEMENT precioiva (#PCDATA)>
<!ELEMENT tipo (#PCDATA)>
<!ELEMENT medida (#PCDATA)>
]>
<BDcomponentes>
  <componente>
    <nombre>resistencia</nombre>
    <precio>0.25</precio>
    <tipo>pasivo</tipo>
    <tipo>bobinadas</tipo>
    [.../...]
  </componente>

```

```

<medida>ohmios</medida>
</componente>
<componente>
  <nombre>condensador</nombre>
  <precioiva>0.75</precioiva>
  <medida>Faradios</medida>
</componente>
</BDcomponentes>

```

Para solucionar el problema, podría cambiarse la siguiente línea, modificando el signo más (+) por un asterisco (\*):

```
<!ELEMENT componente (nombre,(precio|precioiva),tipo*,medida)>
```

## 2. Atributos

Los atributos en una DTD se usan para enriquecer o añadir información a un elemento, pero su uso no es muy recomendable, ya que no pueden contener varios valores, no son extensibles, etc. Si un elemento contiene más de un atributo, se definen uno detrás de otro, separándose mediante espacios, tabuladores o nuevas líneas. Se declaran usando la marca `<!ATTLIST` seguida del elemento que contiene el atributo, su nombre, tipo y valor, tal y como se muestra a continuación.

```
<!ATTLIST Elemento Atributo Tipo Valor>
```

Donde:

- *Elemento*: es el nombre del elemento al que quiere añadirse un atributo.
- *Atributo*: nombre del atributo por añadir.
- *Tipo*: se selecciona un tipo del cuadro 4.3.
- *Valor*: comportamiento del valor del atributo.

### CUADRO 4.3

#### Tabla de tipos de atributos

Tipo atributo	Explicación
CDATA	El valor es alfanumérico (cadena de texto).
Enumeraciones	Permite especificar varios valores para un atributo: <code>&lt;!ATTLIST numero octal(0 1 2 3 4 5 6 7) CDATA #REQUIRED&gt;</code>
ENTITY	El atributo es una entidad que se ha definido previamente.
IDREF	Hace referencia al identificador de otro elemento.
IDREFS	Hace referencia a un conjunto de identificadores (ej1   ej2   ej3).
ID	Su contenido es único, identificando de manera única a cada elemento.

[.../...]

## CUADRO 4.3 (CONT.)

NOTATION	Se especifican datos que no son XML.
ENTITIES	Conjunto o lista de entidades.
NMTOKENS	Permite especificar el valor de un atributo mediante caracteres permitidos.

Para especificar si los valores de los atributos son obligatorios u opcionales, se usan los modificadores del cuadro 4.4.

CUADRO 4.4  
Tabla de valores atributos

Valor	Significado
#FIXED	Debe especificarse un valor que es fijo y no podrá cambiarse.
#IMPLIED	El atributo es opcional.
#DEFAULT	Debe especificarse un valor por defecto que sí podrá cambiarse.
#REQUIRED	El atributo es obligatorio.

En el ejemplo siguiente, se aprecia un documento XML con su DTD asociada, donde aparecen dos atributos obligatorios definidos mediante <!ATTLIST>.

```
<?xml version="1.0" encoding="iso-8859-1"?>
<!DOCTYPE bdusuarios [
    <!ELEMENT bdusuarios (usuario)+>
    <!ELEMENT usuario (nombre,email,fechaNac)>
    <!ATTLIST usuario
        clave CDATA #REQUIRED
        dni CDATA #REQUIRED>

    <!ELEMENT nombre (#PCDATA)>
    <!ELEMENT email (#PCDATA)>
    <!ELEMENT fechaNac EMPTY>
]>
<bdusuarios>
    <usuario clave="**" dni="*****">
        <nombre>Ataulfo G. Pascual</nombre>
        <email>user1@librolm.com</email>
        <fechaNac></fechaNac>
    </usuario>
    <usuario clave="***" dni="*****">
        <nombre>Francisco J. García</nombre>
        <email>user2@librolm.com</email>
        <fechaNac></fechaNac>
    </usuario>
</bdusuarios>
```

### 3. Entidades

Las entidades se consideran constantes y pueden usarse para abreviar texto o utilizar algunos caracteres especiales como el signo de menor que (<) cuya entidad es (&lt;).

Pueden clasificarse en internas y externas (especificando la URL del documento que contiene la entidad).

Para crearlas dentro de una DTD, se usa la marca <!ENTITY> seguida del nombre de la entidad y su valor asociado, tal y como se muestra a continuación.

```
<!ENTITY nombre valor>
```

La manera de referenciar a una entidad es dentro del documento XML usando el carácter *ampersand* (&) que precede al nombre de la entidad terminada en el carácter punto y coma (;).

Para comprender mejor este concepto, va a crearse una serie de entidades propias en el documento DTD, usándolas en el XML, como se muestra en el ejemplo siguiente.

Las dos entidades son:

- Web con valor “<http://www.librolm.com>”.
- Cliente cuyo valor es “Se informa que el usuario ha accedido al”.

Las referencias se hacen mediante:

- &cliente;
- &web;

```
<?xml version="1.0" encoding="iso-8859-1"?>
<!DOCTYPE bdusuarios[
    <!ELEMENT bdusuarios (usuario)+>
    <!ELEMENT usuario (email,nombre,suceso)>
    <!ELEMENT email (#PCDATA)>
    <!ELEMENT nombre (#PCDATA)>
    <!ELEMENT suceso (#PCDATA)>
    <!ENTITY web "http://www.librolm.com">
    <!ENTITY cliente "Se informa que el usuario ha accedido al">
]>
<bdusuarios>
    <usuario>
        <email>user1@librolm.com</email>
        <nombre>Ataulfo G. Pascual</nombre>
        <suceso> &cliente; registro en la página &web; </suceso>
    </usuario>
    <usuario>
        <email>user2@librolm.com</email>
        <nombre>Francisco J. García</nombre>
        <suceso> &cliente; contenido de &web; para consultar el libro X</
            suceso>
    </usuario>
</bdusuarios>
```

Al visualizarlo en un navegador, la referencia a la entidad del documento XML se transforma en el valor asociado (figura 4.6).



```

<?xml version="1.0" encoding="iso-8859-1" ?>
<!DOCTYPE bdusuarios (View Source for full doctype...)>
- <bdusuarios>
  - <usuario>
    <email>user1@librolm.com</email>
    <nombre>Ataulfo G. Pascual</nombre>
    <suceso>Se informa que el usuario ha accedido al registro en la
      página http://www.librolm.com</suceso>
  </usuario>
  - <usuario>
    <email>user2@librolm.com</email>
    <nombre>Francisco J. García</nombre>
    <suceso>Se informa que el usuario ha accedido al contenido de
      http://www.librolm.com para consultar el libro X</suceso>
  </usuario>
</bdusuarios>

```

**Figura 4.6**

Visualización de entidades XML en el navegador.



### Actividades propuestas

Responde si las siguientes afirmaciones son verdaderas o falsas y razona tu respuesta:

- V F 4.5.** Los componentes principales de una DTD son (elementos, atributos y entidades).
- V F 4.6.** Los atributos en una DTD se declaran con la marca <!ENTITY.

**Verificar**



## B) Asociación DTD en XML

Existen dos maneras de vincular un fichero DTD a un XML. La primera de ellas es ubicarlo en el mismo documento XML y la segunda usando un fichero externo que puede estar en el mismo servidor o uno externo. Aunque las dos posibilidades son aceptadas, es recomendable usar la segunda opción por temas de claridad, compartición y manipulación, ya que, si una DTD se usa para distintos XML y hubiera que hacer modificaciones, habría que cambiar varios documentos en vez de uno.

### RECUERDA

- ✓ Puede declararse una DTD en el mismo XML, así como en un fichero externo.

Para declarar una DTD en el mismo documento, la segunda línea del XML será `<!DOCTYPE` seguida del elemento raíz y, entre corchetes, los distintos elementos, tal y como se muestra en el ejemplo del ejemplo:

```

<?xml version="1.0" encoding="iso-8859-1"?>
<!DOCTYPE bdcorreo [
    <!ELEMENT bdcorreo (email)+>
    <!ELEMENT email (receptor,asunto,cuerpo)>
    <!ELEMENT receptor (#PCDATA)>
    <!ELEMENT asunto (#PCDATA)>
    <!ELEMENT cuerpo (#PCDATA)>
]>
<bdcorreo>
    <email>
        <receptor>r1@librolm.com</receptor>
        <asunto>Cumpleaños</asunto>
        <cuerpo>Desearte un feliz..</cuerpo>
    </email>
    <email>
        <receptor>r2@librolm.com</receptor>
        <asunto>Día libre</asunto>
        <cuerpo>Estimado señor X,...</cuerpo>
    </email>
</bdcorreo>

```

Si, por el contrario, el usuario desea declarar la DTD en un documento externo, simplemente tendrá que crear dos ficheros independientes. El primero para la DTD siguiendo las reglas que van a detallarse a continuación y el segundo para el XML.

Para comenzar a declarar una *DTD en un fichero independiente*, simplemente hay que introducir los distintos elementos que lo forman. En el documento XML, la llamada al DTD se realiza mediante la instrucción o marca `<!DOCTYPE` seguida del elemento raíz y el nombre del fichero DTD entre comillas.

Entre el nombre del documento y el elemento raíz, suele ponerse si la definición es privada (SYSTEM), siendo esta la manera más usada o pública (PUBLIC) definida por organismos, lo que obliga a poner un campo más llamado *identificador FPI (formal public identifier)*.

Para ver una definición de DTD independiente, se aprecia la asociación del DTD en el XML en la segunda línea:

```

<?xml version="1.0" encoding="iso-8859-1"?>
<!DOCTYPE bdcorreo SYSTEM "midtd.dtd">
<bdcorreo>
    <email>
        <receptor>r1@librolm.com</receptor>
        <asunto>Cumpleaños</asunto>
        <cuerpo>Desearte un feliz..</cuerpo>
    </email>
    <email>
        <receptor>r2@librolm.com</receptor>
        <asunto>Día libre</asunto>
        <cuerpo>Estimado señor X,...</cuerpo>
    </email>
</bdcorreo>

```

El fichero DTD creado se llamará *midtd.dtd* y será el del ejemplo siguiente.

```
<!ELEMENT bdcorreo (email)+>
<!ELEMENT email (receptor,asunto,cuerpo)>
<!ELEMENT receptor (#PCDATA)>
<!ELEMENT asunto (#PCDATA)>
<!ELEMENT cuerpo (#PCDATA)>
```

#### RECUERDA

La primera línea de un documento XML posee un atributo llamado *standalone* con los siguientes valores:

- Si se usa DTD externa, el valor es "no".
- Si no se usa DTD externa, el valor es "yes".

#### Ejercicio resuelto 4.4

Dado el DTD para crear distintas especies de árboles, obtén un XML asociado:

```
<!ELEMENT árbol (especie)+>
<!ELEMENT especie (nombre,variedad,(origen)+,color_fruto,
    (maduración)+,(riego)?,(precio)?)>
<!ELEMENT nombre (#PCDATA)>
<!ELEMENT variedad (#PCDATA)>
<!ELEMENT origen (#PCDATA)>
<!ELEMENT color_fruto (#PCDATA)>
<!ELEMENT maduración (#PCDATA)>
<!ELEMENT riego (#PCDATA)>
<!ELEMENT precio (#PCDATA)>
```

**Solución** 

### 4.3.2. XML Schema XSD

El lenguaje XML Schema o XSD (*XML schema definition*) surgido en 1998 y se usa como alternativa a documentos DTD, al no cubrir este las expectativas que se necesitan, ya que no permite especificar tipos de elementos o atributos más complejos, así como añadir otras restricciones más complejas.

Entre las ventajas que presenta, destacan:

- Más definición de tipos básicos.
- Mayor control sobre el número de ocurrencias de los elementos.
- Sintaxis similar a XML.

Los XML Schema son ficheros planos con extensión.xsd que permiten definir los elementos y atributos que componen un XML, así como sus tipos, el orden de aparición, el número de veces que puede repetirse cada elemento, valores por defecto, etc.

### A) Creación y elementos de un XSD

La estructura de todo documento XSD es similar a la de un XML, pero presentando el elemento raíz `<xs:schema>`. Dicho elemento contiene una serie de atributos, entre los que destacan los siguientes:

- `xmlns:alias`. Se especifica el espacio de nombres de donde provienen los elementos y tipos usados, cuyo valor tiene que ser “<http://www.w3.org/2001/XMLSchema>”. El alias suele ser `xs` o `xsd`, aunque, mientras se use la misma en todo el documento, no importa el nombre que se le asigne. Hacen referencia a las etiquetas del espacio de nombres.
- `elementFormDefault`. A la hora de declarar los elementos, hay que indicar si debe añadirse el espacio de nombres delante, cuyos posibles valores son “qualified” y “unqualified”.
- `attributeFormDefault`. A la hora de declarar los atributos, hay que indicar si debe añadirse el espacio de nombres delante, cuyos posibles valores son “qualified” y “unqualified”, que indica que el primero de ellos debe ser obligatorio. Por defecto, se usa la segunda opción.
- `targetNamespace`. Se especifica el espacio de nombres de los elementos definidos.
- `versión`. Se especifica la versión del documento de esquema.

Aunque no es obligatorio, la primera línea puede ser la declaración del XML. En la segunda, tal y como se muestra a continuación, se define el esquema, con la especificación de los distintos elementos. El alias en este caso es `xs`.

```

<?xml version= "1.0" encoding= "utf-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
             elementFormDefault="qualified">
    ...
    Elementos
    ...
</xs:schema>

```

#### 1. Elementos

Puede realizarse una distinción entre los tipos de elementos de un XSD. Están los elementos simples y complejos. Elementos *sin hijos ni atributos* o *simples* se definen con la marca `<xs:element>`, conteniendo solamente valores de un determinado tipo.

```
<xs:element Nombre Tipo Valor/>
```

Donde:

- *Nombre*: se representa mediante el atributo *name*, asignándole el valor que se desee.
- *Tipo*: se usa el atributo *type*, que permite especificar el tipo de elemento (ver cuadro 4.6).
- *Valor*: puede ser un valor por defecto (*default*) que permite ser modificado o un valor fijo obligatorio (*fixed*) que no podrá cambiarse. Cada uno de ellos irá acompañado de su correspondiente asignación.

A continuación, se aprecia un elemento XSD llamado *color* de tipo cadena de caracteres o texto y valor por defecto “azul”.

```
<xs:element name="color" type="xs:string" default="azul"/>
```

## ■ Indicadores de ocurrencias

Indica las veces que un elemento debe aparecer o repetirse mediante dos atributos que informan del número máximo y mínimo de veces que puede aparecer.

**CUADRO 4.5**  
Indicadores de ocurrencias de elementos

Atributo	Significado
<b>maxOccurs</b>	Se especifica un valor con el número máximo de veces que puede aparecer el elemento, cuyo valor por defecto es 1. Para especificar un número de veces ilimitado, el valor es “unbounded”.
<b>minOccurs</b>	Se especifica un valor con el número mínimo de veces que puede aparecer el elemento, cuyo valor por defecto 1. Para especificar un número de veces ilimitado, el valor es “unbounded”.

## ■ Elementos complejos

Contienen otros elementos y atributos. Dependiendo del contenido, pueden darse ejemplos de elementos con hijos, con contenido vacío y atributos, con contenido no vacío y atributos, entre otros.

Se definen con el tipo de datos `<xs:complexType>`, seguido de un indicador de orden y los elementos simples, que puede variar dependiendo del contenido que tenga.

```

<xs:element name="nombreElementoCompuesto">
  <xs:complexType>
    <Indicador_Orden>
      Elementos / atributos
    </ Indicador_Orden>
  </xs:complexType>
</xs:element>

```

Donde:

- `<xs:complexType>`: se usa para definir elementos de tipo complejo.
- `Indicador_Orden`: en este apartado, hay que especificar un indicador de orden de aparición de los distintos elementos, que puede ser tres distintos:
  1. `<xs:sequence>`: se especifican los hijos del elemento principal, siguiendo la secuencia u orden de aparición indicada.
  2. `<xs:choice>`: se especifica el elemento hijo que puede aparecer de entre todos. Solo uno de ellos.
  3. `<xs:all>`: los hijos pueden aparecer en cualquier orden una sola vez, la secuencia aparecerá en el orden que se desee.

Para ver con detalle este tipo de elementos, va a realizarse una serie de ejemplos donde irán definiéndose los XDS de los XSM.

En el siguiente ejemplo, se declara un XSD con elementos complejos mediante referencias. La secuencia de elementos que debe aparecer es *nombre*, *email* y *fechaNac*, en ese orden. La referencia se realiza con el atributo *ref* y cuyo valor es el nombre del elemento situado en la parte inferior, donde se especifica el tipo.

```

<xs:schema elementFormDefault="qualified" xmlns:xs="http://www.w3.org/2001/XMLSchema">
<xs:element name="usuario">
  <xs:complexType>
    <xs:sequence>
      <xs:element ref="nombre"/>
      <xs:element ref="email"/>
      <xs:element ref="fechaNac"/>
    </xs:sequence>
  </xs:complexType>
</xs:element>
<xs:element name="nombre" type="xs:string"/>
<xs:element name="email" type="xs:string"/>
<xs:element name="fechaNac" type="xs:date"/>

```

El ejemplo anterior puede realizarse sin referencias, como se muestra a continuación, donde se observa un elemento compuesto (*usuario*) y tres elementos simples (*nombre*, *email* y *fechaNac*).

```

<xs:schema elementFormDefault="qualified" xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <xs:element name="usuario">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="nombre" type="xs:string"/>
        <xs:element name="email" type="xs:string"/>
        <xs:element name="fechaNac" type="xs:date"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
</xs:schema>

```

Un XML correcto a la hora de validarla podría ser el siguiente.

```

<?xml version="1.0" encoding="iso-8859-1"?>
<usuario>
  <nombre>Francisco J. García</nombre>
  <email>user2@librolm.com</email>
  <fechaNac>2018-10-25</fechaNac>
</usuario>

```

Si, por el contrario, el usuario se equivoca a la hora de construir el XML siguiendo el esquema, que dará error al validar. Este aspecto se aprecia a continuación, donde se ha cambiado de orden el segundo y el tercer elemento.

```

<?xml version="1.0" encoding="iso-8859-1"?>
<usuario>
  <nombre>Francisco J. García</nombre>
  <fechaNac>2018-10-25</fechaNac>
  <email>user2@librolm.com</email>
</usuario>

```

El indicador de orden `<xs:choice>` permite especificar uno de los elementos hijos que aparecen, pero solo uno de ellos. En el ejemplo siguiente, se pretende que el usuario se identifique con su nombre de usuario o *email*, con uno de los dos.

```

<xs:element name="usuario">
  <xs:complexType>
    [.../...]

```

```

<xs:choice>
    <xs:element name="nombreU" type="xs:string"/>
    <xs:element name="email" type="xs:string"/>
</xs:choice>
</xs:complexType>
</xs:element>
</xs:schema>

```

Un posible XML válido sería.

```

<?xml version="1.0" encoding="iso-8859-1"?>
<usuario>
    <email>user2@librolm.com</email>
</usuario>

```

Un XML que no cumple a la hora de validar porque se han añadido dos elementos en vez de seleccionar uno de la lista sería el que se presenta en el ejemplo siguiente.

```

<?xml version="1.0" encoding="iso-8859-1"?>
<usuario>
    <nombreU>Sergio</nombreU>
    <email>user2@librolm.com</email>
</usuario>

```

El indicador de orden <xs:all>: permite especificar la secuencia de elementos en cualquier orden, aunque sin poder añadir dentro de él <xs:choice> o <xs:sequence>. La especificación se muestra en el ejemplo siguiente.

```

<xs:schema elementFormDefault="qualified" xmlns:xs="http://www.w3.org/2001/
  XMLSchema">
    <xs:element name="usuario">
        <xs:complexType>
            <xs:all>
                <xs:element name="nombre" type="xs:string"/>
                <xs:element name="email" type="xs:string"/>
                <xs:element name="fechaNac" type="xs:date"/>
            </xs:all>
        </xs:complexType>
    </xs:element>
</xs:schema>

```

Algunos de los XML válidos al tener en cuenta el XSD del ejemplo anterior son los que se muestran a continuación, donde aparecen distintas alternativas dependiendo del número de elementos que haya dentro de <xs:all>.

```
?xml version="1.0" encoding="iso-8859-1"?>
<usuario>
    <nombre>Francisco J. García</nombre>
    <fechaNac>2018-10-25</fechaNac>
    <email>user2@librolm.com</email>
</usuario>

<?xml version="1.0" encoding="iso-8859-1"?>
<usuario>
    <fechaNac>2018-10-25</fechaNac>
    <nombre>Francisco J. García</nombre>
    <email>user2@librolm.com</email>
</usuario>

<?xml version="1.0" encoding="iso-8859-1"?>
<usuario>
    <fechaNac>2018-10-25</fechaNac>
    <email>user2@librolm.com</email>
    <nombre>Francisco J. García</nombre>
</usuario>
```

Si pretende crearse un XML que no cumpla con las especificaciones del XSD, bastaría con eliminar uno de los elementos de la lista que tiene que aparecer:

```
<?xml version="1.0" encoding="iso-8859-1"?>
<usuario>
    <fechaNac>2018-10-25</fechaNac>
    <email>user2@librolm.com</email>
</usuario>
```

Tal y como se ve en el XML del ejemplo siguiente, donde se muestra un dato complejo con texto y atributos, existe un elemento *nombre* con el atributo *DNI* y contenido Eduardo.

```
<?xml version="1.0" encoding="iso-8859-1"?>
<nombre dni="11111111K">Eduardo</nombre>
```

Un esquema asociado al ejemplo anterior sería el que se presenta a continuación.

```

<?xml version="1.0" encoding="iso-8859-1"?>
<xss:schema elementFormDefault="qualified" xmlns:xss="http://www.w3.org/2001/
  XMLSchema">
  <xss:element name="nombre">
    <xss:complexType>
      <xss:simpleContent>
        <xss:extension base="xss:string">
          <xss:attribute name="dni" type="xss:string" />
        </xss:extension>
      </xss:simpleContent>
    </xss:complexType>
  </xss:element>
</xss:schema>

```

### Ejercicio resuelto 4.5



Crea un XSD para un XML de un elemento vacío con atributo como el siguiente:

```

<?xml version="1.0" encoding="iso-8859-1"?>
<nombre dni="11111111K"></nombre>

```

**Solución**

## 2. Tipos de datos XSD

XML Schema dispone de numerosos tipos de datos simples predefinidos incorporados para los elementos o atributos, siendo los más comunes los que se observan en el cuadro 4.6

**CUADRO 4.6**  
Tipos de datos predefinidos XSD

Tipo	Descripción
<b>decimal</b>	Número decimal.
<b>boolean</b>	Un único valor lógico (1, 0).
<b>dateTime</b>	Fecha y hora en formato (AAAA-MM-DD T HH:MM:SS).
<b>string</b>	Texto o cadena de caracteres.
<b>date</b>	Fecha: año-mes-día (aaaa-mm-dd).
<b>time</b>	Hora: hh:mm:ss.
<b>integer</b>	Número entero positivo o negativo.
<b>positiveInteger</b>	Número entero positivo.
<b>long</b>	Entero de 64 bits.
<b>short</b>	Entero de 16 bits.

En el cuadro 4.7, se aprecia un elemento nombre de tipo cadena de caracteres (xs:string).

**CUADRO 4.7**  
Aplicando tipo cadena de caracteres a un elemento

Elemento XSD	<code>&lt;xs:element name="nombre" type="xs:string" /&gt;</code>
Ej. XML	<code>&lt;nombre&gt;Francisco J. García&lt;/nombre&gt;</code>

### ■ Creación de tipos de datos simples personalizados

En ocasiones, es interesante crear tipos de datos nuevos personalizados, dotándolos de mayor precisión y eficiencia. Estos pueden ser asignados a elementos o atributos, siendo reutilizados en el esquema XDS.

Para definir un tipo simple nuevo, se usa la siguiente estructura `<xs:simpleType>`, cuyo atributo *name* servirá para asignárselo al elemento e indicar que es de un tipo simple nuevo. Dentro de un tipo simple, pueden añadirse restricciones o facetas, listas, uniones, etc.

La sintaxis puede apreciarse en el ejemplo siguiente.

```
<xs:simpleType name="Tcat1">
  ...
</xs:simpleType>
```

### ■ Tipos de datos complejos

Ya vistos con los elementos complejos, se muestran en el primer ejemplo el XML de usuarios y en el segundo la DTD asociada.

```
<?xml version="1.0" encoding="iso-8859-1"?>
<usuario>
  <nombre>Francisco J. García</nombre>
  <email>user2@librolm.com</email>
  <fechaNac>2018-10-25</fechaNac>
</usuario>
```

```
<?xml version="1.0" encoding="iso-8859-1"?>
<xs:schema attributeFormDefault="unqualified" elementFormDefault="qualified"
  xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <xs:element name="usuario" type="Tuser" />
  <xs:complexType name="Tuser" >
    <xs:sequence>
      <xs:element name="nombre" type="xs:string" />
      <xs:element name="email" type="xs:string" />
      <xs:element name="fechaNac" type="xs:date" />
    </xs:sequence>
  </xs:complexType>
</xs:schema>
```

### 3. Atributos

Poseen la misma funcionalidad que los atributos de un elemento DTD. Los elementos con atributos se consideran elementos complejos, declarándose como tipos simples. Intentan dotarlos de más información usando la siguiente sintaxis.

Existe la posibilidad, al igual que en los elementos, de añadir restricciones a sus tipos.

```
<xs:attribute NombreA TipoA USO ValorA/>
```

Donde:

- *NombreA*: se representa mediante el atributo *name*, asignándole el valor que se desee.
- *TipoA*: se usa el atributo *type*, que permite especificar el tipo de elemento.
- *USO*: se especifica la opcionalidad o no del atributo mediante *use*, que puede tener los valores: “prohibited”, que indica que es prohibido; “required” para indicar que es obligatorio, y “optional”, que informan de que es opcional y es el valor por defecto.
- *ValorA*: puede ser un valor por defecto (*default*) o un valor fijo u obligatorio (*fixed*) con su correspondiente asignación.

```
<xs:attribute name="dni" type="xs:integer" use="required"/>
```

### 4. Restricciones o facetas

Si quisiera ponerse una restricción (también llamadas *facetas*) a un atributo o elemento, hasta ahora, con DTD no podía especificarse. XML Schema permite especificar esta característica controlando en mejor medida los datos introducidos en el XML. En el cuadro 4.8, se aprecian diversas facetas junto con su descripción.

**CUADRO 4.8**

Restricciones o facetas

Faceta	Descripción
<b>length</b>	Contiene un atributo <i>value</i> donde se especifica un valor de longitud fija.
<b>minLength</b>	Contiene un atributo <i>value</i> donde se especifica el valor de longitud mínima.
<b>maxLength</b>	Contiene un atributo <i>value</i> donde se especifica el valor de longitud máxima.
<b>whiteSpace</b>	Contiene un atributo <i>value</i> donde se especifica un valor que puede ser (“preserve”, “replace” y “collapse”). Trata el uso de los espacios en blanco, retornos de carro, etc., que puedan aparecer.
<b>maxExclusive</b>	Contiene un atributo <i>value</i> donde se especifica el valor, que es menor al especificado.
<b>minExclusive</b>	Contiene un atributo <i>value</i> donde se especifica el valor, que es mayor al especificado.

[.../...]

CUADRO 4.8 (CONT.)

<b>minInclusive</b>	Contiene un atributo <i>value</i> donde se especifica el valor, que es mayor o igual al especificado.
<b>maxInclusive</b>	Contiene un atributo <i>value</i> donde se especifica el valor, que es menor o igual al especificado.
<b>totalDigits</b>	Contiene un atributo <i>value</i> donde se especifica el valor, que es el número máximo de dígitos de un número teniendo en cuenta los decimales.
<b>fractionDigits</b>	Contiene un atributo <i>value</i> donde se especifica el valor, que es el número máximo de dígitos de decimales de un número.
<b>enumeration</b>	Contiene un atributo <i>value</i> donde se especifica uno de los valores admitidos de la lista.
<b>pattern</b>	Contiene un atributo <i>value</i> donde se especifica un rango de caracteres admitidos o expresión regular.

Para comprender mejor este apartado, se presenta el XML siguiente, que almacena datos relativos a varios productos.

```
<?xml version="1.0" encoding="iso-8859-1"?>
<bdproductos xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:no-
NamespaceSchemaLocation="zax.xsd">
  <producto>
    <nombre>teclado</nombre>
    <código>C001</código>
    < categoría>A</ categoría>
    <iva>21</iva>
    <precio>15</precio>
    <descuento>10</descuento>
  </producto>
  <producto>
    <nombre>monitor</nombre>
    <código>C002</código>
    < categoría>B</ categoría>
    <iva>21</iva>
    <precio>147.2</precio>
    <descuento>9</descuento>
  </producto>
</bdproductos>
```

El esquema asociado es el que aparece a continuación, donde algunos de los tipos de los elementos son tipos simples nuevos personalizados, con una serie de restricciones cada uno de ellos.

Para definir una restricción sobre un elemento, puede hacerse de varias maneras.

```

<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema" elementFormDefault="qualified">
  <xs:element name="bdproductos">
    <xs:complexType>
      <xs:sequence>
        <xs:element maxOccurs="unbounded" ref="producto" />
      </xs:sequence>
    </xs:complexType>
  </xs:element>
  <xs:element name="producto">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="nombre" type="xs:string"/>
        <xs:element name="código" type="Tlongitudentre"/>
        <xs:element name="categoría" type="Toatl1"/>
        <xs:element name="iva" type="xs:integer"/>
        <xs:element name="precio" type="Tdecimal"/>
        <xs:element name="descuento" type="Tdescuento"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
</xs:schema>
Tipos simples
...
  
```

## ■ Restricción numérica

Si desea especificarse que el descuento puede tener el valor entre 1 y 10, ambos incluidos, va a crearse un tipo simple llamado *Tdescuento*, que permite introducir descuentos con el rango de valores [1..10]. Si, en el XML, se introduce en ese elemento un contenido no perteneciente al rango, estará bien formado, pero no será válido.

Lo primero es crear un tipo simple con una restricción. Para añadir una restricción, se usa el elemento *xs:restriction*, que define las restricciones del tipo. Dispone de un atributo *base*, que indica el tipo de datos.

```

<xs:simpleType name="Tdescuento">
  <xs:restriction base="xs:integer">
    <xs:minInclusive value="1"/>
    <xs:maxInclusive value="10"/>
  </xs:restriction>
</xs:simpleType>
  
```

El código tendrá una longitud entre 1 y 4.

```

<xs:simpleType name="Tlongitudentre">
  <xs:restriction base="xs:string">
    <xs:minLength value="1"/>
    <xs:maxLength value="4"/>
  </xs:restriction>
</xs:simpleType>
  
```

Después, se crea la restricción para que el precio tenga un total de 4 dígitos y hasta 2 decimales.

```
<xs:simpleType name="Tdecimal">
  <xs:restriction base="xs:decimal">
    <xs:totalDigits value="4"/>
    <xs:fractionDigits value="2"/>
  </xs:restriction>
</xs:simpleType>
```

### ■ Restricción longitud del contenido

Es preciso especificar que el código tendrá 4 caracteres.

```
<xs:simpleType name="Tlongitud">
  <xs:restriction base="xs:string">
    <xs:length value="4"/>
  </xs:restriction>
</xs:simpleType>
```

### ■ Restricción de enumeración

Hay que limitar el contenido de un elemento o atributo a un conjunto de valores posibles. Por ejemplo, la categoría puede valer A, B, C, D y E:

```
<xs:simpleType name="Tcat">
  <xs:restriction base="xs:string">
    <xs:enumeration value="A"/>
    <xs:enumeration value="B"/>
    <xs:enumeration value="C"/>
    <xs:enumeration value="D"/>
    <xs:enumeration value="E"/>
  </xs:restriction>
</xs:simpleType>
```

### ■ Restricción mediante expresiones regulares o patrones

Otra manera de presentar el ejemplo anterior es mediante el uso de patrones, donde el valor permitido es uno de los del rango de valores permitidos. Algunas de las expresiones regulares o patrones que pueden incluirse son:

- *[0-9]*: número del 0 al 9.
- *[a-z]*: letra minúscula perteneciente a ese intervalo.

- $[A-Z]$ : letra mayúscula perteneciente a ese intervalo.
- $[aeiou]$ : un carácter de los que aparecen entre los corchetes.
- $[^aeiou]$ : un carácter que no aparezca entre corchetes.
- $\{X\}$ : las llaves indican que tiene que repetirse X veces el contenido que haya delante.

```
<xs:simpleType name="Tcat1">
  <xs:restriction base="xs:string">
    <xs:pattern value="[A-E]" />
  </xs:restriction>
</xs:simpleType>
```

El XSD con todas las restricciones es el siguiente:

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
  elementFormDefault="qualified">
  <xs:element name="bdproductos">
    <xs:complexType>
      <xs:sequence>
        <xs:element maxOccurs="unbounded" ref="producto"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
  <xs:element name="producto">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="nombre" type="xs:string"/>
        <xs:element name="código" type="Tlongitudentre"/>
        <xs:element name="categoría" type="Tcat1"/>
        <xs:element name="iva" type="xs:integer"/>
        <xs:element name="precio" type="Tdecimal"/>
        <xs:element name="descuento" type="Tdescuento"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
  <xs:simpleType name="Tdescuento">
    <xs:restriction base="xs:integer">
      <xs:minInclusive value="1"/>
      <xs:maxInclusive value="10"/>
    </xs:restriction>
  </xs:simpleType>
  <xs:simpleType name="Tdecimal">
    <xs:restriction base="xs:decimal">
      <xs:totalDigits value="4"/>
      <xs:fractionDigits value="2"/>
    </xs:restriction>
  </xs:simpleType>
  <xs:simpleType name="Tcat">
    [.../...]
```

```

<xs:restriction base="xs:string">
    <xs:enumeration value="A"/>
    <xs:enumeration value="B"/>
    <xs:enumeration value="C"/>
    <xs:enumeration value="D"/>
    <xs:enumeration value="E"/>
</xs:restriction>
</xs:simpleType>
<xs:simpleType name="Tcat1">
    <xs:restriction base="xs:string">
        <xs:pattern value="[A-E]"/>
    </xs:restriction>
</xs:simpleType>
<xs:simpleType name="Tlongitud">
    <xs:restriction base="xs:string">
        <xs:length value="4"/>
    </xs:restriction>
</xs:simpleType>
<xs:simpleType name="Tlongitudentre">
    <xs:restriction base="xs:string">
        <xs:minLength value="1"/>
        <xs:maxLength value="4"/>
    </xs:restriction>
</xs:simpleType>
</xs:schema>

```

## Actividad resuelta 4.6

Elabora un patrón para DNI.

**Solución** 



## 5. Documentación de esquemas

La creación de un esquema puede ser algo complejo, por lo que introducir datos relacionados con la autoría, utilidad, derechos de autor, etc., es un aspecto que ha de tenerse en cuenta. Podría pensarse que, con poner comentarios, valdría, pero existen elementos destinados a tal fin.

Si necesita dotarse de información a los esquemas para que pueda ser consultada por otros usuarios, se usan las siguientes instrucciones:

- *xs:annotation*: especifica los comentarios dentro de un esquema que actúan como documentación de este. Tiene un atributo opcional (*id*) y dos elementos hijos (*appinfo* o *documentation*).
- *xs:appinfo*: se usa para especificar la información de la aplicación. Tiene un atributo opcional (*source*).

- xs:documentation: *añade comentarios en un esquema. Debe ir dentro de xs:annotation y tiene dos atributos opcionales (source y xml:lang).*

```
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
<xs:annotation>
  <xs:appinfo>Lenguaje de marcas</xs:appinfo>
  <xs:documentation xml:lang="es">
    comentarios sobre
  </xs:documentation>
</xs:annotation>
...

```



### Actividades propuestas

Responde si las siguientes afirmaciones son verdaderas o falsas y razona tu respuesta:

- V F 4.7.** La faceta *pattern* se usa para añadir restricciones mediante expresiones regulares.
- V F 4.8.** En los elementos complejos, existe la marca *<xs:choice>*, que indica que los hijos pueden aparecer en cualquier orden una sola vez, la secuencia aparecerá en el orden que se desee.

**Verificar**



## B) Asociación XSD en XML

La asociación de un documento XSD a uno XML se realiza mediante un espacio de nombres con una serie de atributos:

- *xmlns*: definir el espacio de nombres, indicando que van a usarse los elementos definidos en la URL especificada. Los elementos del esquema llevarán el prefijo que se defina, en este caso, será *xs*.

```
xmlns:xs="http://www.w3.org/2001/XMLSchema-instance"
```

Las referencias a los esquemas pueden realizarse mediante los siguientes atributos:

- *noNamespaceSchemaLocation*: se emplea cuando no se utilizarán espacios de nombres en el documento. Se usará un fichero con el esquema:

```
xs:noNamespaceSchemaLocation = "longan.xsd"
```

- *schemaLocation*: se emplea cuando se utilizan explícitamente los nombres de los espacios de nombres en las etiquetas.

Tal y como se aprecia en el ejemplo siguiente, el documento XML de bases de datos de alumnos hace referencia a un fichero de espacio de nombres previamente creado, xsdalumnos.xsd, el cual servirá para que sea validado el XML.

```
<?xml version="1.0" encoding="iso-8859-1"?>
<bdalumnos xmlns:xs="http://www.w3.org/2001/XMLSchema-instance" xs:noNa-
    mespaceSchemaLocation="xsdalumnos.xsd">
    <alumno>
        <nombre>José Antonio</nombre>
        <padre>José Luis</padre>
        <madre>Carmen</madre>
        <edad>18</edad>
        <ciudad>Málaga</ciudad>
    </alumno>
    <alumno>
        <nombre>Francisco Jesús</nombre>
        <padre>José</padre>
        <madre>María</madre>
        <edad>25</edad>
        <ciudad>15</ciudad>
    </alumno>
</bdalumnos>
```

El esquema xsdalumnos.xsd es:

```
<?xml version="1.0" encoding="iso-8859-1"?>
<xs:schema attributeFormDefault="unqualified" elementFormDefault="qualified"
    xmlns:xs="http://www.w3.org/2001/XMLSchema">
    <xs:element name="bdalumnos">
        <xs:complexType>
            <xs:sequence>
                <xs:element maxOccurs="unbounded" name="alumno">
                    <xs:complexType>
                        <xs:sequence>
                            <xs:element name="nombre" type="xs:string" />
                            <xs:element name="padre" type="xs:string" />
                            <xs:element name="madre" type="xs:string" />
                            <xs:element name="edad" type="xs:unsignedByte" />
                            <xs:element name="ciudad" type="xs:string" />
                        </xs:sequence>
                    </xs:complexType>
                </xs:element>
            </xs:sequence>
        </xs:complexType>
    </xs:element>
</xs:schema>
```

#### 4.4. Herramientas de creación y edición XML

Existen multitud de herramientas que permiten crear y editar de manera fácil y amena documentos XML. Aunque las gratuitas permiten crear, editar y validar estos documentos y crear

esquemas, así como comprobar si están bien formados, las de licencia propietaria permiten la descarga de una versión de evaluación con un periodo determinado de tiempo y presentan muchas más prestaciones.

Algunas de ellas son Stylus Studio, Oxygen XML, Exchanger XML Editor, EditiX XML Editor, Liquid Studio y XMLBlueprint, entre otras.

#### 4.4.1. XML Copy Editor

*Software* con licencia GNU disponible para plataformas Windows y Linux, permite la validación DTD. Resalta la sintaxis y permite plegado y la terminación de etiquetas.

#### 4.4.2. Altova XMLSpy

Es uno de los editores de XML más vendidos, con avanzadas funciones de modelado, edición, transformación y depuración. Se permite su compra o realizar una evaluación de 30 días. Entre sus características, se destaca: Editor XML, cliente y depurador SOAP, editor gráfico de esquemas XML y DTD, editor WSDL gráfico, vista texto XML, vista cuadrícula XML, editor JSON y JSON Schema, analizador y generador de expresiones XPath, integración con Visual Studio y Eclipse, edición, depuración y generación de perfiles XQuery, etc.

#### 4.4.3. XMLSpear

Editor XML libre que permite la validación en tiempo real. Construido en Java y disponible para todas las plataformas. Puede crearse un documento XML desde cero o abrir uno y añadir o eliminar nodos. Algunas características que han de destacarse: panel de búsqueda XPath; soporte de catálogo XML; traducciones XSLT; validación completa del esquema; editor de árbol para insertar, repetir y eliminar nodos; validación en tiempo real contra esquema o DTD durante la edición, etc.

La pantalla principal, en la parte inferior, permite ver el código en forma de elemento o tabla, lo que resulta de gran utilidad para tener una visión general del XML.

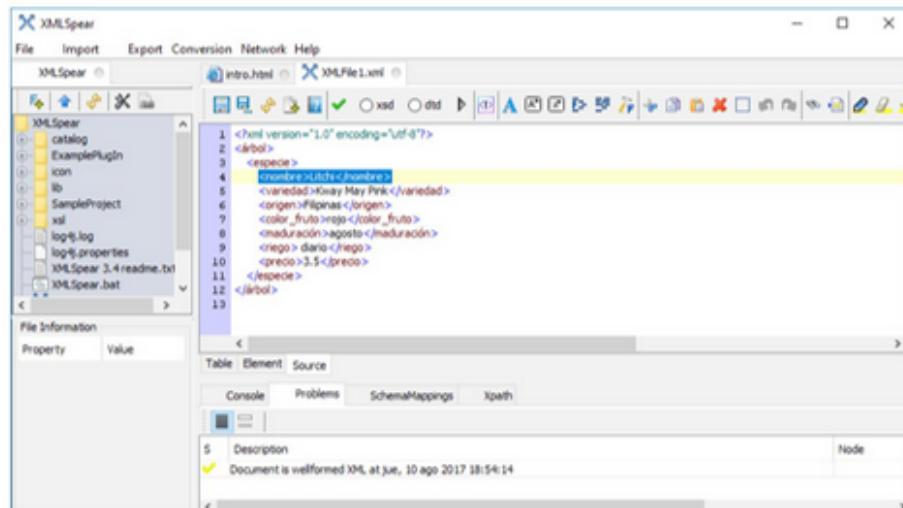


Figura 4.7. Interfaz de XMLSpear.

#### 4.4.4. XML Notepad

Editor de XML con dos partes diferenciadas (código y estructura de árbol) que permite editar y visualizar documentos XML de manera fácil y amena.

#### 4.4.5. Online XML Editor

Editor de XML que muestra la información en forma de árbol. El usuario va escribiendo las etiquetas en el panel izquierdo y, de manera automática, va apareciendo el árbol a mano derecha.

**Recursos web**

www

A través de estos enlaces podrás acceder a XML Copy Editor, Altova XMLSpy, XMLSpear, XML Notepad y Online XML Editor.



### 4.5. Validación

El método de validación es un proceso que permite comprobar que el contenido de un fichero XML presenta una estructura adecuada y cumple una serie de normas (tipo de elementos y atributos que pueden añadirse, orden de aparición, etc.).

Para construir un documento bien formado o sintácticamente correcto, hay que seguir las normas vistas en el capítulo 1. Un documento válido tiene que estar bien formado y cumplir una serie de requisitos especificados en una estructura DTD, XML Schema o mediante Relax NG.

**RECUERDA**

- ✓ Un documento puede estar bien formado, pero no ser válido. En cambio, si es válido, seguro que está bien formado.

Las herramientas que se encargan de realizar este proceso se denominan *validadores* o *parsers*. Puede validarse de varias maneras, como se detalla a continuación.

### A) Usando línea de comando

XMLStarlet es una herramienta de línea de comandos que permite validar, transformar, editar, consultar, ficheros XML para sistemas operativos (Linux, Solaris, Windows, Mac OS X, etc.).

www

### Recursos web

A través de los siguientes enlaces podrás acceder a dos herramientas *online* de validación.

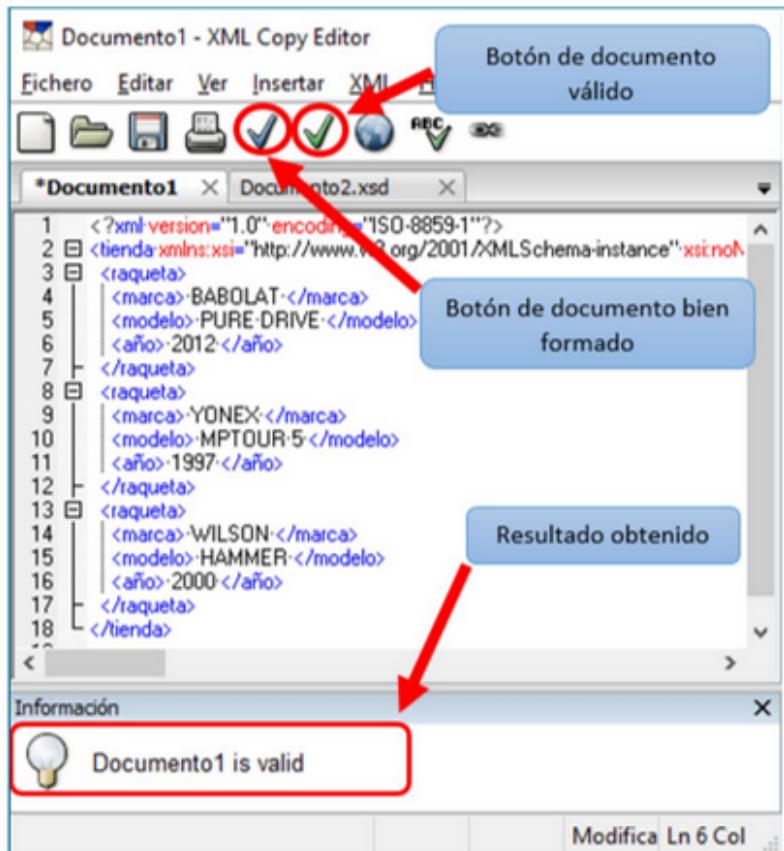



### B) Aplicaciones de escritorio para validar

Existen numerosas aplicaciones de escritorio del ámbito de XML que ofrecen la opción de comprobar si un documento está bien formado y es válido. Para ello, simplemente, hay que introducir el documento XML y acceder a las opciones que se deseen. Algunos editores XML que ofrecen esta posibilidad son:

- XML Copy Editor.
- Stylus Studio.
- Liquid Studio.
- Oxygen XML Editor.
- EditiX XML Editor.

En la figura 4.8, se observan las opciones de validar y comprobar si un documento XML está bien formado usando la herramienta XML Copy Editor.



**Figura 4.8**  
Validar un documento empleando XML Copy Editor.

## Ejercicio resuelto 4.7



Valida el siguiente XML usando Liquid Studio.

XML:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<tienda>
<raqueta>
    <marca>BABOLAT</marca>
    <modelo>PURE DRIVE</modelo>
    <año>2012</año>
</raqueta>
<raqueta>
    <marca>YONEX</marca>
    <modelo>MPTOUR 5</modelo>
    <año>1997</año>
</raqueta>
<raqueta>
    <marca>WILSON</marca>
    <modelo>HAMMER</modelo>
    <año>2000</año>
</raqueta>
</tienda>
```

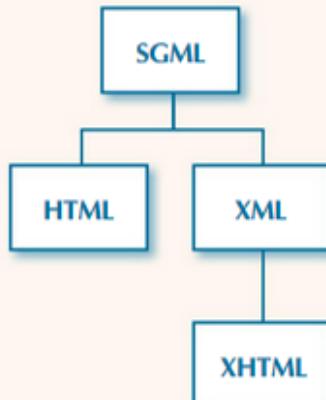
XSD:

```
<?xml version="1.0" encoding="iso-8859-1"?>
<xs:schema attributeFormDefault="unqualified" elementFormDefault="qualified" xmlns:xs="http://www.w3.org/2001/XMLSchema">
    <xs:element name="tienda">
        <xs:complexType>
            <xs:sequence>
                <xs:element maxOccurs="unbounded" name="raqueta">
                    <xs:complexType>
                        <xs:sequence>
                            <xs:element name="marca"
                                type="xs:string" />
                            <xs:element name="modelo"
                                type="xs:string" />
                            <xs:element name="año"
                                type="xs:unsignedShort" />
                        </xs:sequence>
                    </xs:complexType>
                </xs:element>
            </xs:sequence>
        </xs:complexType>
    </xs:element>
</xs:schema>
```

Solución

## Resumen

■ Origen XML:



■ Estructura XML y sintaxis:

- El prólogo:
  - Versión.
  - Encoding.
- Elemento raíz o cuerpo:

```

<?xml version="1.0" encoding="utf-8"?>
<padre>
  <hijo>
    <subhijos> ... </subhijos>
  </hijo>
</padre>
  
```

■ En las marcas o etiquetas, se usan los signos menor que (<) y mayor que (>) para definirlas. Existen tres tipos de etiquetas o marcas:

1. Las de apertura.
2. Las de cierre.
3. Etiquetas vacías.

- Los atributos se especifican con el par NombreAtributo="valor".
- Los comentarios se expresan comenzando por <!-- y terminando con -->.
- Sección CDATA:

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

- Creación, asociación y elementos de DTD y XSD.
- DTD (definición de tipo de documento):

- Elementos:

```
<!ELEMENT nombreElemento (elementos_hijo1, elemento_hijo2,...)>
```

- Atributos:

```
<!ATTLIST Elemento Atributo Tipo Valor>
```

- Entidades:

```
<!ENTITY nombre valor>
```

- Asociación DTD en XML:

- Interna:

```
<!DOCTYPE raíz [
    elementos
]>
```

- Externa:

```
<!DOCTYPE bdcorreo SYSTEM "midtd.dtd">
```

- XML Schema XSD:

- Estructura:

```
<?xml version = "1.0" encoding = "utf-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema" elementForm-
Default="qualified">
    ...
    Elementos
    ...
</xs:schema>
```

- Elementos:

```
<xs:element Nombre Tipo Valor"/>
```

- Tipos de datos XSD:

Tipo	Descripción
<i>decimal</i>	Número decimal.
<i>boolean</i>	Un único valor lógico (1, 0).
<i>dateTime</i>	Fecha y hora en formato (AAAA-MM-DD T HH:MM:SS).
<i>string</i>	Texto o cadena de caracteres.
<i>date</i>	Fecha: año-mes-día (aaaa-mm-dd).
<i>time</i>	Hora: hh:mm:ss.
<i>integer</i>	Número entero positivo o negativo.
<i>positiveInteger</i>	Número entero positivo.
<i>long</i>	Entero de 64 bits.
<i>short</i>	Entero de 16 bits.

— Tipos de datos XSD:

Tipo	Descripción
<i>decimal</i>	Número decimal.
<i>boolean</i>	Un único valor lógico (1, 0).
<i>dateTime</i>	Fecha y hora en formato (AAAA-MM-DD T HH:MM:SS).
<i>string</i>	Texto o cadena de caracteres.
<i>date</i>	Fecha: año-mes-día (aaaa-mm-dd).
<i>time</i>	Hora: hh:mm:ss.
<i>integer</i>	Número entero positivo o negativo.
<i>positiveInteger</i>	Número entero positivo.
<i>long</i>	Entero de 64 bits.
<i>short</i>	Entero de 16 bits.

— Atributos:

```
<xs:attribute NombreA TipoA USO ValorA"/>
```

— Restricciones o facetas:

Faceta	Descripción
<i>length</i>	Contiene un atributo <i>value</i> donde se especifica un valor de longitud fija.
<i>minLength</i>	Contiene un atributo <i>value</i> donde se especifica el valor de longitud mínima.
<i>maxLength</i>	Contiene un atributo <i>value</i> donde se especifica el valor de longitud máxima.
<i>whiteSpace</i>	Contiene un atributo <i>value</i> donde se especifica un valor que puede ser ("preserve", "replace" y "collapse"). Trata el uso de los espacios en blanco, retornos de carro, etc., que puedan aparecer.
<i>maxExclusive</i>	Contiene un atributo <i>value</i> donde se especifica el valor, que es menor al especificado.
<i>minExclusive</i>	Contiene un atributo <i>value</i> donde se especifica el valor, que es mayor al especificado.
<i>minInclusive</i>	Contiene un atributo <i>value</i> donde se especifica el valor, que es mayor o igual al especificado.
<i>maxInclusive</i>	Contiene un atributo <i>value</i> donde se especifica el valor, que es menor o igual al especificado.
<i>totalDigits</i>	Contiene un atributo <i>value</i> donde se especifica el valor, que es el número máximo de dígitos de un número teniendo en cuenta los decimales.
<i>fractionDigits</i>	Contiene un atributo <i>value</i> donde se especifica el valor, que es el número máximo de dígitos de decimales de un número.
<i>enumeration</i>	Contiene un atributo <i>value</i> donde se especifica uno de los valores admitidos de la lista.
<i>pattern</i>	Contiene un atributo <i>value</i> donde se especifica un rango de caracteres admitidos o expresión regular

- Asociación XSD en XML:

```
<?xml version="1.0" encoding="iso-8859-1"?>
<bdalumnos xmlns:xs="http://www.w3.org/2001/XMLSchema-instance"
  xs:noNamespaceSchemaLocation="xsdalumnos.xsd">
```

- Herramientas de creación y edición XML:

- XML Copy Editor.
- Altova XMLSpy.
- XMLSpear.
- XML Notepad.
- Online XML Editor.

- Validación:

- Usando línea de comando.
- Herramientas *online*.
- Aplicaciones de escritorio.

## Ejercicios prácticos resueltos

1. Cuenta el número de elementos del siguiente código XML:

```
<item>
  <title>Agregadores</title>
  <link>http://www.librolm.com/html</link>
  <description>Apuntes agregadores</description>
  <guid>B4BA124D-D535-4830-B28C-E0F0BD49D80E</guid>
  <pubDate>Tue, 26 Dec 2017 17:02:39 GMT</pubDate>
</item>
```

Solución 

2. Contesta a las siguientes preguntas:

- a) ¿Está bien formado el siguiente XML teniendo en cuenta su esquema?
- b) ¿Es válido el siguiente XML teniendo en cuenta su esquema?

XML:

```
<?xml version="1.0" encoding="iso-8859-1"?>
<bdproductos xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:noNamespaceSchemaLocation="zax.xsd">
  <producto>
    <nombre>teclado</nombre> [...]
```

```

<código>C001</código>
<iva>21</iva>
< categoría>A</ categoría>
<precio>15</precio>
<descuento>10</descuento>
</ producto>
< producto>
    < nombre>monitor</ nombre>
    < código>C002</ código>
    < categoría>B</ categoría>
    <iva>21</iva>
    <precio>147.2</precio>
    <descuento>9</descuento>
</ producto>
</ bdproductos>

```

Esquema:

```

<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema" elementFormDefault="qualified">
    <xs:element name="bdproductos">
        <xs:complexType>
            <xs:sequence>
                <xs:element maxOccurs="unbounded" ref="producto"/>
            </xs:sequence>
        </xs:complexType>
    </xs:element>
    <xs:element name="producto">
        <xs:complexType>
            <xs:sequence>
                <xs:element ref="nombre"/>
                <xs:element ref="código"/>
                <xs:element ref=" categoría"/>
                <xs:element ref="iva"/>
                <xs:element ref="precio"/>
                <xs:element ref="descuento"/>
            </xs:sequence>
        </xs:complexType>
    </xs:element>
    <xs:element name="nombre" type="xs:NCName"/>
    <xs:element name="código" type="Tlongitudentre"/>
    <xs:element name=" categoría" type="Toatl1"/>
    <xs:element name="iva" type="xs:integer"/>
    <xs:element name="precio" type="Tdecimal"/>
    <xs:element name="descuento" type="Tdescuento"/>
    <xs:simpleType name="Tdescuento">
        <xs:restriction base="xs:integer">
            <xs:minInclusive value="1"/>
            <xs:maxInclusive value="10"/>
        </xs:restriction>
    </xs:simpleType>
    <xs:simpleType name="Tdecimal">

```

[.../...]

```

<xs:restriction base="xs:decimal">
    <xs:totalDigits value="4"/>
    <xs:fractionDigits value="2"/>
</xs:restriction>
</xs:simpleType>
<xs:simpleType name="Tcat">
    <xs:restriction base="xs:string">
        <xs:enumeration value="A"/>
        <xs:enumeration value="B"/>
        <xs:enumeration value="C"/>
        <xs:enumeration value="D"/>
        <xs:enumeration value="E"/>
    </xs:restriction>
</xs:simpleType>
<xs:simpleType name="Tcat1">
    <xs:restriction base="xs:string">
        <xs:pattern value="[A-E]"/>
    </xs:restriction>
</xs:simpleType>
<xs:simpleType name="Tlongitud">
    <xs:restriction base="xs:string">
        <xs:length value="4"/>
    </xs:restriction>
</xs:simpleType>
<xs:simpleType name="Tlongitudentre">
    <xs:restriction base="xs:string">
        <xs:minLength value="1"/>
        <xs:maxLength value="4"/>
    </xs:restriction>
</xs:simpleType>
</xs:schema>

```

Solución 

3. Dado el siguiente documento XML, donde se almacenan datos relacionados de una librería con libros de informática, crea el documento DTD asociado.

XML:

```

<?xml version="1.0" encoding="iso-8859-1"?>
<librería>
    <libro>
        <título>Lenguaje de marcas</título>
        <autor>Autor2</autor>
        <autor>Autor1</autor>
        <ISBN></ISBN>
        <páginas>158</páginas>
        <precio>15</precio>
    </libro>
    <libro>
        <título>XML</título>
        <autor>Autor1</autor>
        <ISBN></ISBN>
        <páginas>278</páginas>
        <precio>30</precio>
    </libro>
</librería>

```

¿Qué modificación harías en el documento DTD para que puedan añadirse varios precios, teniendo en cuenta que puede haber libros en papel y *online*?

**Solución** 

4. Pon un ejemplo de los siguientes elementos:

- Elementos sin atributos con datos.
- Elementos con atributos y datos.
- Elementos con atributos sin datos.

**Solución** 

## Ejercicios prácticos

1. Realiza una DTD para describir un conjunto de artículos, sabiendo que hay que almacenar el nombre, precio, código y fabricante.
2. Obtén el DTD y XSD del siguiente documento XML:

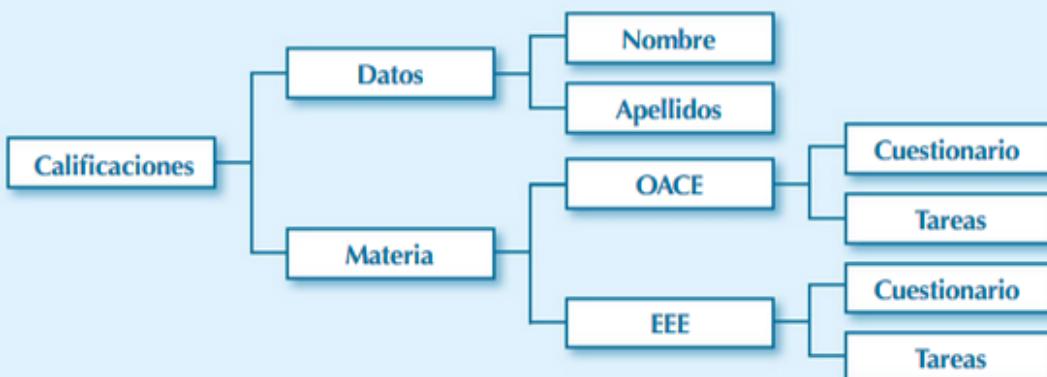
```
<?xml version="1.0" encoding="iso-8859-1"?>
<vivero>
    <especie siembra="2018">
        <nombre>Litchi</nombre>
        <precio moneda="euro">25</precio>
        <variedad>Kway May Pink</variedad>
        <origen>Filipinas</origen>
        <color_fruto>rojo</color_fruto>
        <color_fruto>rosa</color_fruto>
        <otros_datos>
            <maduración>agosto</maduración>
            <riego> diario</riego>
        </otros_datos>
    </especie>
    <especie siembra="2017">
        <nombre>Longan</nombre>
        <precio moneda="euro">10</precio>
        <variedad>Champoo</variedad>
        <origen> China</origen>
        <color_fruto>marrón</color_fruto>
        <otros_datos>
            <maduración>octubre</maduración>
            <riego> diario</riego>
        </otros_datos>
    </especie>
    <especie siembra="2016">
        <nombre>Litchi</nombre>
        <precio moneda="euro">21</precio>
        <variedad> mauritius</variedad>
    [.../...]
```

```

<origen>Florida</origen>
<color_fruto>rojo</color_fruto>
<otros_datos>
    <maduración>agosto</maduración>
    <riego> diario</riego>
</otros_datos>
</especie>
</vivero>

```

3. Quiere implementarse en un documento XML la estructura de las calificaciones de los alumnos de un centro de Málaga, entre los que hay que guardar: datos personales (nombre y apellidos), materia (OACE y EEE) con sus respectivas tareas y cuestionarios, tal y como se muestra en la siguiente figura:



Los datos que han de introducirse son:

ALUMNOS					
Datos		Materia			
		OACE		EEE	
Nombre	Apellidos	Cuestionario	Tareas	Cuestionario	Tareas
Felipe	Pascual	4	8	7	6
María	García	9	2	9	6

Una vez terminado, añade otra firma, completando con los valores que estimes oportuno.

4. Dado el siguiente documento XML, determina sus etiquetas, elementos y atributos:

```

<?xml version="1.0" encoding="utf-8" ?>
<agenda>
    <contacto nombre="alumno LM">
        <ciudad>Málaga</ciudad>
        <teléfono n="666777888" />
        <email>666777888@prueba.com</email>
    </contacto>
</agenda>

```

5. Dado el siguiente fichero DTD, construye un documento XML con algunos datos:

```
<!ELEMENT Móviles (Movil+)>
<!ELEMENT Movil (Modelo, precio, Fabricante, color)>
<!ELEMENT Modelo (#PCDATA)>
<!ELEMENT precio (#PCDATA)>
<!ELEMENT Fabricante (#PCDATA)>
<!ELEMENT color (#PCDATA)>
```

6. Quiere almacenarse la siguiente información en un XML, que contiene datos relativos a socios de un gimnasio, sabiendo que *género* e *id* son atributos.
- Crea el XML donde se almacena dicha información.

Persona		Nombre y apellidos	Teléfono	Email	Socio
Género	ID				
Femenino	001	Segundo Pascual	6XXXXXXXXX 95XXXXXXXX	User1@gmail.com User2@gmail.com User3@gmail.com	Sí
Masculino	002	Obdulia Toledo		Carmen@uma.es	No

- Elabora un fichero DTD asociado, para lo cual has de tener en cuenta que:
  - Un usuario puede tener más de un *email* (+).
  - Puede haber más de un teléfono por usuario, o ninguno (*teléfono*)\*.
  - Puede introducirse más de un usuario (*persona*)+.

7. Dado el siguiente XML:

```
<agenda>
  <contacto>
    <nombre_apellidos>Francisco Ruiz González</nombre_apellidos>
    <telefono>600222xxx</telefono>
    <edad>39</edad>
    <email>curro@gmail.com</email>
    <direccion>
      <calle>cómpeta 16</calle>
      <cod_post>12345</cod_post>
      <provincia>Málaga</provincia>
    </direccion>
  </contacto>
<agenda>
  <contacto>
    <nombre_apellidos>Francisco Ruiz González</nombre_apellidos>
    <telefono>600222xxx</telefono>
    <edad>39</edad>
    <email>curro@gmail.com</email>
    [.../...]
```

```

<direccion>
    <calle>cómpeta 16</calle>
    <cod_post>12345</cod_post>
    <provincia>Málaga</provincia>
</direccion>
</contacto>
<contacto>
    <nombre_apellidos>Pepe Escudero</nombre_apellidos>
    <telefono>954122xxx</telefono>
    <telefono>602222xxx</telefono>
    <edad>25</edad>
    <direccion>
        <calle>cristo</calle>
        <cod_post>29785</cod_post>
        <provincia>Málaga</provincia>
    </direccion>
</contacto>
<contacto>
    <nombre_apellidos>Carmen Ruiz García</nombre_apellidos>
    <edad>18</edad>
    <email>carmen@gmail.com</email>
    <email>carmen1@gmail.com</email>
    <email>carmen2@gmail.com</email>
    <direccion>
        <calle>canalejas</calle>
        <cod_post>25985</cod_post>
        <provincia>Cádiz</provincia>
    </direccion>
</contacto>
</agenda>

```

Completa el siguiente fichero DTD:

```

<!ELEMENT agenda (contacto____)>
<!ELEMENT contacto (nombre_apellidos, telefono_, edad, email_, di-
rección)>
<!ELEMENT nombre_apellidos (#PCDATA)>
<!ELEMENT telefono (____)>
<!ELEMENT edad (#PCDATA)>
<!ELEMENT email (#PCDATA)>
<!ELEMENT dirección (calle, cod_post, _____)>
<!ELEMENT calle (#PCDATA)>
<!ELEMENT cod_post (____)>
<!ELEMENT provincia (#PCDATA)>

```

8. Dado el siguiente esquema XSD:

```

<?xml version="1.0" encoding="utf-8"?>
<xs:element name="agenda">
    <xs:complexType>

```

[.../...]

```
<xs:sequence>
  <xs:element maxOccurs="unbounded" name="contacto">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="nombre_apellidos" type="xs:string"/>
        <xs:element minOccurs="0" maxOccurs="2" name="telefono" type="xs:unsignedInt" />
        <xs:element name="edad" type="xs:unsignedByte" />
        <xs:element minOccurs="1" maxOccurs="3" name="email" type="xs:string" />
      </xs:sequence>
    </xs:complexType>
  </xs:element>
</xs:sequence>
</xs:complexType>
</xs:element>
</xs:schema>
```

Responde a las siguientes preguntas, argumentando el resultado:

- a) ¿Podrías añadir 7 contactos más a la agenda?
  - b) Si un usuario deja el campo teléfono sin rellenar, ¿estaría bien?
  - c) El usuario Ataulfo dispone de dos teléfonos móviles y uno fijo, ¿podrían añadirse todos en el XML?
  - d) ¿Cuántos correos electrónicos, como máximo y mínimo, hay que introducir?
  - e) ¿De qué tipo es el campo *edad*?
9. Enumera las principales diferencias entre DTD y XML.

AUTOEVALUACIÓN

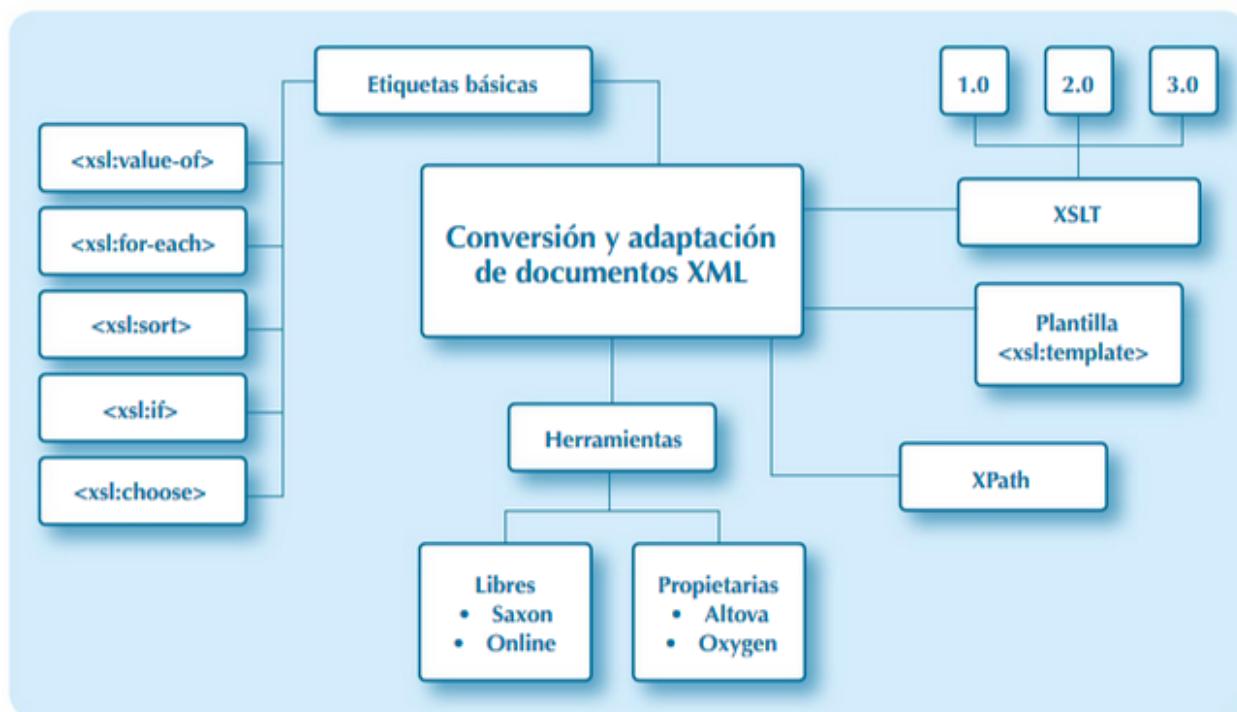


# Conversión y adaptación de documentos XML

## Objetivos

- ✓ El objetivo de este capítulo es que aprendas a transformar documentos XML. XML es un estándar que permite definir el formato y los datos en un mismo fichero. Con XSLT combinado con otras herramientas como XPath, podrás transformar unos ficheros XML en otros o en HTML. Este tipo de herramientas a nivel profesional son muy importantes y necesarias en ciertos proyectos o trabajos.
- ✓ Es importante que conozcas las diferencias existentes entre las versiones 1, 2 y 3 de XSLT.
- ✓ También deberás aprender la sintaxis de un archivo XSL y practicar las conversiones tanto a XML como a HTML. Dentro de dichas conversiones, deberás conocer el funcionamiento y la aplicación de las etiquetas básicas XSL como `<xsl:value-of>`, `<xsl:for-each>` o `<xsl:sort>`, entre otras.

## Mapa conceptual



## Glosario

**Expresión regular.** Patrón para encontrar una combinación de caracteres dentro de un texto o una cadena.

**Refactorización.** Acción de reestructurar el código para una mejor comprensión, mantenibilidad, eficiencia, etc., pero sin modificar su funcionalidad.

**Template.** Plantilla.

**W3C.** Acrónimo de World Wide Web Consortium. Organismo que se encarga de regular los estándares web.

**XPath.** Es un lenguaje de consulta para un documento XML. Permite seleccionar nodos dentro del mismo.

**XSL.** Sigla de *extensible stylesheet language* (lenguaje de hojas de estilo extensibles).

**XSL-FO.** Sigla de *extensible stylesheet language formatting objects*.

**XSLT.** Sigla de *XML stylesheets language for transformation* (lenguaje de transformación basado en hojas de estilo).

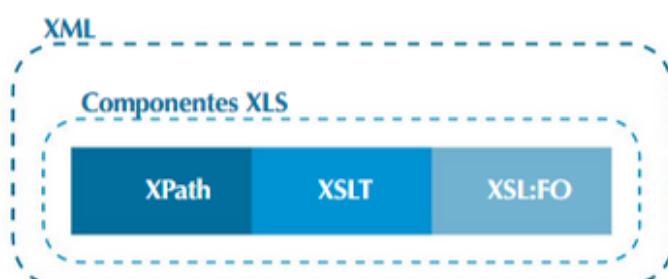
## 5.1. Introducción

XSL es la sigla de *extensible stylesheet language* (lenguaje de hojas de estilo extensibles). Básicamente, es un lenguaje para la transformación de documentación. Se basa en los siguientes pilares:

- *Transformación.* Convirtiendo los ficheros XML en otro formato.
- *Formateo de objetos.* A través del diseño de un documento XML.
- *Búsqueda de contenido en documentos.* A través del árbol de nodos de un documento XML.

El XSL es parte del XML y basa su versatilidad en tres componentes fundamentales:

1. **XPath.** Es un lenguaje para seleccionar o navegar entre los nodos en lenguaje XML dentro de un DOM.
2. **XSL-FO.** FO de formateo de objetos (*formatting objects*). Transformaciones para convertir el documento XML en otro tipo de información como HTML o PDF (el famoso formato de Adobe).
3. **XSLT.** Sigla de *XML stylesheets language for transformation* (lenguaje de transformación basado en hojas de estilo).



**Figura 5.1**  
Componentes de XML.



SABÍAS QUE...

Actualmente, existen tres versiones de XSLT: la 1.0, 2.0 y la 3.0.

XSLT se utiliza para las transformaciones y es la parte más importante de XSL, por lo que se considera una recomendación del W3C (World Wide Web Consortium).

El objetivo de XSLT es convertir los documentos XML en:

- Otros documentos XML.
- Otros formatos reconocibles por un navegador como HTML o XHTML, por ejemplo.

Generalmente, la transformación que hace XSLT es de uno a uno. Cada elemento XML se transformará en otro HTML o XHTML.

RECUERDA

- ✓ XSL es más que un lenguaje de hojas de estilo. CSS hace un documento más legible, pero que sigue siendo el mismo, mientras que XSL transforma un documento en otro.



PARA SABER MÁS

### XSL-FO

Es la sigla de *extensible stylesheet language formatting objects*. Básicamente, es un lenguaje para formatear un documento XML, pero, como es muy complejo, no cuenta con mucho soporte.

El objetivo de este lenguaje es parecido al CSS, definir la forma en la que deberá mostrarse un documento XML por pantalla (o papel).

**Actividad propuesta 5.1**

Marca cuáles de estos lenguajes no permiten transformar un documento:

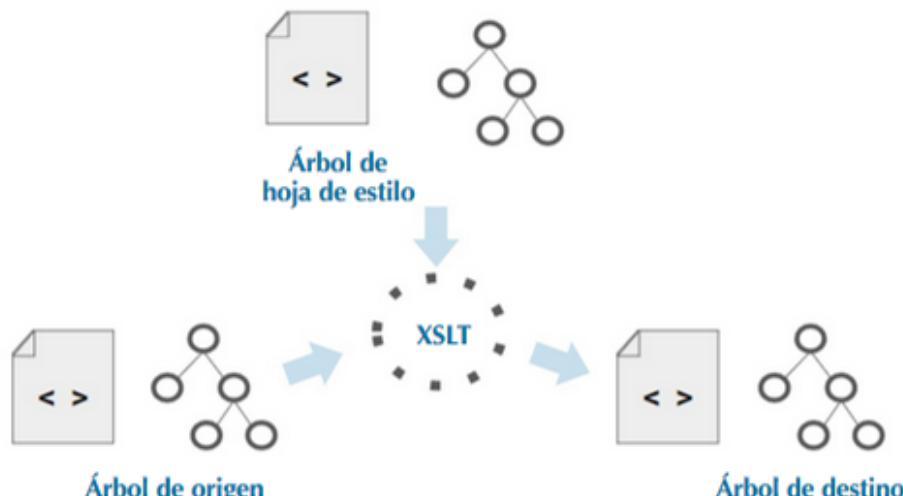
- a) HTML.
- b) XSL.
- c) XPath.
- d) XHTML.

**Verificar**

## 5.2. Técnicas de transformación de documentos XML

Una de las ventajas de las transformaciones es que pueden añadirse o quitarse elementos y atributos del fichero de salida. Como puede deducirse, XSLT ofrece mucho juego, puesto que los elementos de un fichero pueden reestructurarse, ordenarse, etc.

Además, pueden tomarse decisiones como qué elementos van a mostrarse u ocultarse, entre otras.



**Figura 5.2**  
Esquema de una transformación XSLT.

Básicamente, una transformación XSLT consiste en modificar un árbol de entrada en uno de salida.

### ¿Qué navegadores soportan XSLT?



Una transformación XSLT se realiza usando un procesador XSLT, *software* que toma como entrada dos archivos (XML y XSL) y genera como documento de salida otro aplicando en la transformación la hoja de estilos XSL al fichero origen XML.

Los procesadores XSLT pueden estar integrados en un editor XML o en un explorador web o bien pueden invocarse desde la línea de comandos (como Xalan o Saxon).

Muchas veces, los procesadores XML son librerías que pueden estar integradas en cualquier tipo de programa.

### 5.2.1. Diferencias entre las versiones de XSLT 1.0, 2.0 y 3.0

XSLT 1.0 fue creada el 16 de noviembre de 1999 y, ocho años después (en el 2007), nació XSLT 2.0. Por lo tanto, aunque son retrocompatibles, la versión 2.0 tiene bastantes mejoras con respecto a la versión 1.0. Saxonica (Saxon) recomendó encarecidamente utilizar este estándar, que es compatible hacia delante (un programa que funcionaba con la versión 1.0 debería funcionar con procesadores con versiones superiores).

Actualmente, existen buenos procesadores XSLT como son Xalan, Saxon y MSXML.

Entre las novedades de la versión 2.0, están las siguientes:

- *Mejora del sistema de tipos.* La versión 1.0 era muy restrictiva, puesto que solamente existían cinco: *string*, *number*, *boolean*, *node-set* y *result tree fragment*.
- *Mejora de las funciones incorporadas y los operadores.*
- *Mejora del modelo de datos.*

#### Recurso web

*www*

Te recomendamos acceder a la web donde está publicada la recomendación del W3C:



- *Soporte para expresiones regulares.*
- *Múltiples árboles como resultado.* En la versión 1.0, solamente podía crearse un árbol de salida.
- *La instrucción <xsl:function>.* Sirve para poder definir funciones que puedan ser utilizadas en expresiones XPath.
- *Mejora del sistema de errores.*
- *Mejora de la funcionalidad de agrupación.* Por ejemplo, se añade la función *<xsl:for-each-group>*.

#### Recurso web

*www*

El W3C tiene toda la documentación sobre XSLT 3.0, a la que podrás acceder a través de este enlace.



Una de las mejoras que se han incorporado es la posibilidad de *streaming* de los documentos fuente. Muchas veces, los documentos son muy grandes y se necesita producir resultados antes de que todo el documento esté disponible. Para ello, se crearon nuevas instrucciones para poder utilizar esta nueva funcionalidad como *<xsl:source-document>*, *<xsl:iterate>*, *<xsl:merge>* o *<xsl:fork>*.

Asimismo se añadió modularidad a los programas creando el concepto de *paquete* como en Java. Un paquete define un interfaz que regula funciones, variables, plantillas y otros compo-

nentes que, pese a estar visibles fuera del paquete, pueden ser sobreescritas. El objetivo último es la reutilización y mantenibilidad del código.

Otras instrucciones nuevas son:

- *xsl:evaluate*. Para evaluar expresiones XPath que se construyen dinámicamente con *strings* (o que se leen de un documento).
- *xsl:try*. Para recuperarse de errores en tiempo de ejecución.
- *xsl:global-context-item*. Se utiliza cuando se necesita un elemento cuyo contexto sea global (también se declara el tipo del elemento).
- *xsl:assert*. Para ayudar a los desarrolladores a crear código robusto.

### 5.3. Formatos de salida y ámbitos de aplicación

Como se ha mencionado anteriormente, XSLT puede transformar un documento XML en otro XML o en otro formato, como puede ser XHTML.

A continuación, se mostrará un ejercicio guiado para ver el formato de salida de una transformación XSLT, con los siguientes pasos

1. Crear un fichero XML.
2. Crear un fichero XSL.
3. Abrir el fichero XML con un navegador.

#### Ejercicio práctico guiado 5.1

### 5.4. Descripción de la estructura y la sintaxis

Lo primero que va a hacerse en este apartado es explicar en profundidad el ejercicio práctico guiado 5.1.

#### 5.4.1. Fichero XML

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<?xml-stylesheet type="text/xsl" href="RAQUETAS.XSL"?>
<TIENDA>
  <RAQUETA>
    <MARCA> BABOLAT </MARCA>
    <MODELO> PURE DRIVE </MODELO>
    <ANIO> 2012 </ANIO>
  </RAQUETA>
  <RAQUETA>
    <MARCA> YONEX </MARCA>
    <MODELO> MPTOUR 5 </MODELO>
    <ANIO> 1997 </ANIO>
  </RAQUETA>
  <RAQUETA>
    <MARCA> WILSON </MARCA>
    <MODELO> HAMMER </MODELO>
    <ANIO> 2000 </ANIO>
  </RAQUETA>
</TIENDA>
```

Enlace con la hoja de estilo XSL

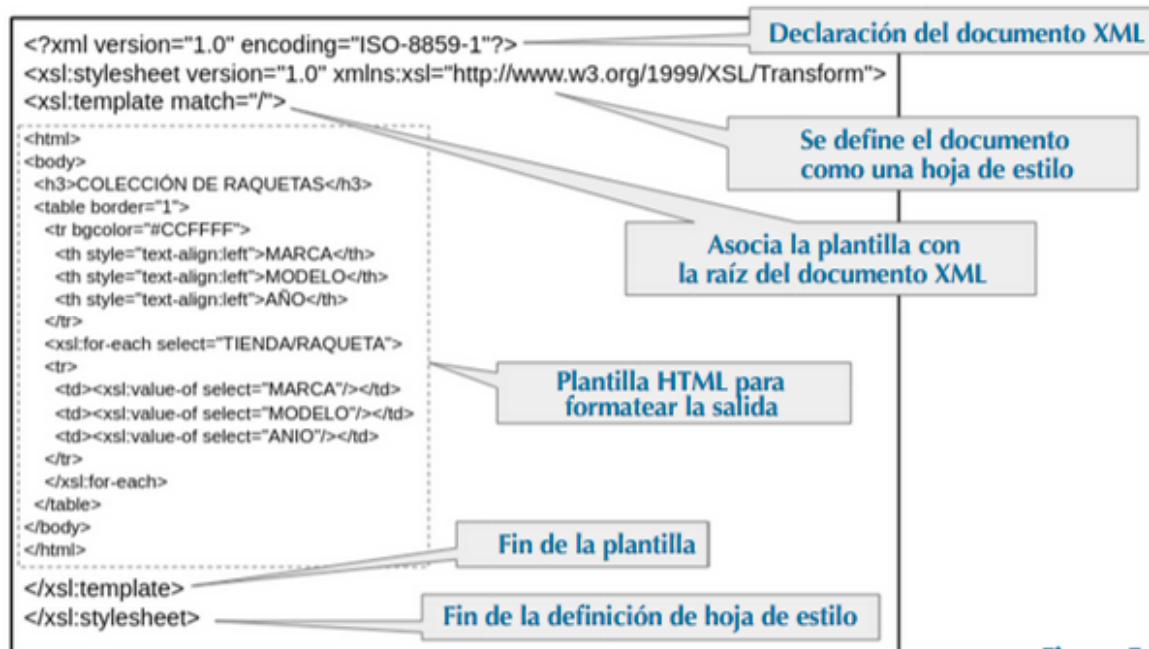
**Figura 5.3**  
Estructura de un fichero XML.

La estructura de este fichero es similar a la del documento XML visto anteriormente, pero la diferencia radica en la utilización de una plantilla XSL, como puede apreciarse mediante la siguiente línea:

```
<?xml-stylesheet type="text/xsl" href="RAQUETAS.XSL"?>
```

Como puede observarse, la etiqueta indica que es una plantilla XSL y el atributo *href* con el valor “RAQUETAS.XSL” indica la ubicación de la propia plantilla RAQUETAS.XSL. En este caso, se encuentra en la misma ubicación que el fichero XML.

#### 5.4.2. Hoja de estilo XSL



**Figura 5.4**  
Estructura de un fichero XSL.

#### RECUERDA

- ✓ Una hoja de estilo XSL es una serie de reglas que se llaman *plantillas*.
- ✓ Una plantilla contendrá las acciones o reglas necesarias, que se aplicarán cuando coincida dicha regla con un nodo concreto.

En la orden siguiente del fichero XSL, pueden distinguirse dos partes:

```
<xsl:template match="/">
```

a) La propia orden *<xsl:template>*.

- Se utiliza para crear plantillas

b) El atributo *match* (*match*=""/").

- Asocia una plantilla a un elemento XML.
- Normalmente, suele utilizarse para definir una plantilla para todo el documento XML. Para ello, se utiliza el atributo *match* con la expresión XPath “/”, la cual indica que va a utilizarse todo el documento. En vez de la raíz, puede especificarse otra ruta en XPath.

El contenido dentro de la etiqueta <xsl:template> tendrá el HTML necesario para formatear la salida según la necesidad del programador.

Como se ha mencionado con anterioridad, el atributo *match* utiliza una expresión XPath, que se estudiará en profundidad en el capítulo siguiente, pero, en el cuadro 5.1, se detallan algunas expresiones comunes de este lenguaje.

#### CUADRO 5.1

#### Expresiones comunes de XPath

Expresión	Resultado
/	Todo el documento.
/TIENDA	El elemento <i>TIENDA</i> del documento.
//RAQUETAS	Todos los elementos <i>RAQUETAS</i> del documento.
/TIENDA/*/*/MARCA	Los elementos <i>MARCA</i> que son <i>nietos</i> del elemento <i>TIENDA</i> del documento.

Generalmente, se utiliza el primero ejemplo visto (“/”) para seleccionar todo el documento, pero pueden elegirse nodos concretos. Las posibilidades de XPath son bastante amplias.

#### Actividad propuesta 5.2



¿Qué permite el lenguaje XPath?

- a) Modificar el formato de los datos de un fichero XML.
- b) Transformar los datos de un fichero XML.
- c) Acceder a los datos de un fichero XML.
- d) Crear una jerarquía de datos dentro de un fichero XML.

**Verificar**

## 5.5. Utilización de plantillas

La operativa básica para transformar archivos XML no es muy complicada, pero la creación de la plantilla correcta para realizar una transformación concreta puede ser más o menos difi-

cultosa, para lo cual, XSLT tiene una serie de etiquetas. A continuación, se describirán las más utilizadas.

### 5.5.1. <xsl:value-of>

Dentro de la plantilla XSL del ejercicio práctico guiado 5.1, pueden observarse órdenes del tipo:

```
<td><xsl:value-of select="MARCA" /></td>
<td><xsl:value-of select="MODELO" /></td>
<td><xsl:value-of select="ANIO" /></td>
```

La etiqueta `<td>` ya se ha tratado, dado que es una etiqueta HTML, pero aparece una nueva etiqueta: `<xsl:value-of>`.

Esta etiqueta extrae el valor de un elemento XML y lo añade al flujo de salida de la transformación. En el ejemplo trabajado, a la salida HTML.

Con el atributo `select`, el cual utiliza una expresión XPath, se seleccionará el elemento concreto.

### 5.5.2. <xsl:for-each>

Otra etiqueta cuyo significado es intuitivo, pero que se estudiará a continuación, es la siguiente:

```
<xsl:for-each select="TIENDA/RAQUETA">
```

En esta etiqueta `<xsl:for-each>`, se seleccionarán todos los elementos uno a uno de una lista determinada. Los elementos que van a utilizarse estarán filtrados por el atributo `select`, el cual utilizará una expresión XPath.

#### Ejercicio resuelto 5.1

Imagina que, en el ejercicio práctico guiado 5.1, quieren mostrarse solamente las raquetas de la marca BABOLAT. Reescribe la orden `<xsl:for-each>` del archivo XSL.

**Solución** 

#### RECUERDA

- ✓ La etiqueta `</xsl:for-each>` siempre tendrá su etiqueta de cierre `</xsl:for-each>`, por lo que no hay que olvidarse de escribirla en la plantilla XSL.

### Ejercicio propuesto 5.1



Se ha modificado el fichero del ejercicio resuelto 5.1 con varios modelos más de raqueta:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<?xml-stylesheet type="text/xsl" href="RAQUETAS.XSL"?>
<TIENDA>
    <RAQUETA>
        <MARCA> BABOLAT </MARCA>
        <MODELO> PURE DRIVE </MODELO>
        <ANIO> 2012 </ANIO>
        <PRECIO>170</PRECIO>
    </RAQUETA>
    <RAQUETA>
        <MARCA> BABOLAT </MARCA>
        <MODELO> PURE STORM</MODELO>
        <ANIO> 2013 </ANIO>
        <PRECIO>190</PRECIO>
    </RAQUETA>
    <RAQUETA>
        <MARCA> YONEX </MARCA>
        <MODELO> MPTOUR 5 </MODELO>
        <ANIO> 1997 </ANIO>
        <PRECIO>150</PRECIO>
    </RAQUETA>
    <RAQUETA>
        <MARCA> WILSON </MARCA>
        <MODELO> HAMMER </MODELO>
        <ANIO> 2000 </ANIO>
        <PRECIO>95</PRECIO>
    </RAQUETA>
    <RAQUETA>
        <MARCA> WILSON </MARCA>
        <MODELO> PRO STAFF </MODELO>
        <ANIO> 1997 </ANIO>
        <PRECIO>215</PRECIO>
    </RAQUETA>
</TIENDA>
```

Con los siguientes predicados XPath:

```
/TIENDA/RAQUETAS[1]
/TIENDA/RAQUETAS[last()]
/TIENDA/RAQUETAS[last()-1]
/TIENDA/RAQUETAS[position()<3]
/TIENDA/RAQUETAS[PRECIO>100]
/TIENDA/RAQUETAS[PRECIO>100]/MARCA
```

¿Cuál será el resultado de aplicarlos con una plantilla XSL? Comprueba el resultado creando una plantilla XSL al respecto.

Selecciona aquellas raquetas con un precio superior a 150 euros.

## RECUERDA

El operador OR (|) en XPath permite seleccionar dos conjuntos de nodos. Por ejemplo, si desea seleccionarse la marca y el modelo de las raquetas de la tienda, podría utilizarse la expresión:

```
//MARCA | //MODELO
```

### 5.5.3. Filtrando contenido

Cuando un fichero es muy grande, lo más normal es filtrar el contenido y no trabajar con todo el árbol de nodos. Por ejemplo, trabajar solamente con las raquetas de la marca BABOLAT.

La etiqueta `<xsl:for-each>` podría modificarse de la siguiente forma para que filtre solamente las raquetas de dicha marca:

```
<xsl:for-each select="TIENDA/RAQUETA[MARCA = 'BABOLAT']">
  <tr>
    <td><xsl:value-of select="MARCA" /></td>
    <td><xsl:value-of select="MODELO" /></td>
    <td><xsl:value-of select="ANIO" /></td>
    <td><xsl:value-of select="PRECIO" /></td>
  </tr>
</xsl:for-each>
```

### 5.5.4. Ordenando el contenido

En este momento, se necesita que, además de la marca BABOLAT, las raquetas queden ordenadas por el precio.

Es preciso utilizar la etiqueta `<xsl:sort>` añadiendo la propiedad `select="PRECIO"`. El resultado sería añadir la siguiente línea al fichero:

```
<xsl:sort select="PRECIO" />
```

Véase cómo quedaría el filtro anterior incluyendo la ordenación:

```
<xsl:for-each select="TIENDA/RAQUETA[MARCA = 'BABOLAT']">
  <xsl:sort select="PRECIO" />
  <tr>
    <td><xsl:value-of select="MARCA" /></td>
    <td><xsl:value-of select="MODELO" /></td>
    <td><xsl:value-of select="ANIO" /></td>
    <td><xsl:value-of select="PRECIO" /></td>
  </tr>
</xsl:for-each>
```

**RECUERDA**

Se sobreentiende que los campos que se ordenan tendrán que ir en la salida. No tiene mucho sentido ordenar por un campo que, posteriormente, no va a aparecer en la salida.

### 5.5.5. Selección avanzada IF

Si quieren mostrarse todas las raquetas con un precio superior a 100 euros. Para esta operación, puede utilizarse la etiqueta `<xsl:if>` de la siguiente forma:

```
<xsl:if test="PRECIO>100">
```

El lugar donde hay que colocar esta sentencia es siempre dentro de la etiqueta `<xsl:for-each>`.

Véase el resultado completo:

```
<xsl:for-each select="TIENDA/RAQUETA[MARCA = 'BABOLAT']">
  <xsl:if test="PRECIO>100">
    <tr>
      <td><xsl:value-of select="MARCA"/></td>
      <td><xsl:value-of select="MODELO"/></td>
      <td><xsl:value-of select="ANIO"/></td>
      <td><xsl:value-of select="PRECIO"/></td>
    </tr>
  </xsl:if>
</xsl:for-each>
```

**RECUERDA**

El contenido de `test` siempre será una expresión XPath.

### 5.5.6. Selección avanzada CHOOSE

Imagínese que quiere conseguirse un resultado como el de la figura 5.5

**Figura 5.5**

Resultado del ejercicio que quiere conseguirse.

#### COLECCIÓN DE RAQUETAS

MARCA	MODELO	AÑO	PRECIO
BABOLAT	PURE DRIVE	2012	170
BABOLAT	PURE STORM	2013	190
YONEX	MPTOUR 5	1997	150
WILSON	HAMMER	2000	95
WILSON	PRO STAFF	1997	215

Dependiendo del valor de la raqueta, el precio se sombreará de amarillo si supera el precio de 150 euros y de verde en caso contrario.

La plantilla que habrá que utilizar para realizar esta transformación es la siguiente:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
<xsl:template match="/">
<html>
<body>
    <h3>COLECCION DE RAQUETAS</h3>
    <table border="1">
        <tr bgcolor="#CCFFFF">
            <th style="text-align:left">MARCA</th>
            <th style="text-align:left">MODELO</th>
            <th style="text-align:left">AÑO</th>
            <th style="text-align:left">PRECIO</th>
        </tr>
        <xsl:for-each select="TIENDA/RAQUETA">
            <tr>
                <td><xsl:value-of select="MARCA"/></td>
                <td><xsl:value-of select="MODELO"/></td>
                <td><xsl:value-of select="ANIO"/></td>
                <xsl:choose>
                    <xsl:when test="PRECIO>150">
                        <td bgcolor="#ffff66"><xsl:value-of select="PRECIO"/></td>
                    </xsl:when>
                    <xsl:otherwise>
                        <td bgcolor="#99ff99"><xsl:value-of select="PRECIO"/></td>
                    </xsl:otherwise>
                </xsl:choose>
            </tr>
        </xsl:for-each>
    </table>
</body>
</html>
</xsl:template>
</xsl:stylesheet>
```

Se ha utilizado la etiqueta `<xsl:choose>` de tal manera que, al seleccionar las raquetas con precio mayor a 150 euros (`<xsl:when test="PRECIO>150">`), la salida tendrá un sombreado y, en caso contrario (`<xsl:otherwise>`), el sombreado será diferente.

## 5.6. Utilización de herramientas de procesamiento: verificación, depuración y generación de documentación

Existen muchas herramientas para la edición de documentos XML. Las mejores herramientas son de pago, mientras que las menos potentes suelen ser *open source*.

**www****Recurso web**

A través de este enlace podrás acceder a Free Online XSLT Tool, que es una herramienta *online* para hacer transformaciones XSLT.

**¿Cómo usar Free Online XSLT Tool?**

A continuación, se hablará de las dos herramientas de *software* comercial más utilizadas en la actualidad.

**Actividad propuesta 5.3**

Responde si la siguiente afirmación es verdadera o falsa:

**V F**

Para transformar un documento mediante XSLT, es necesario utilizar un editor XML, ya sea propietario u *open source*, con un procesador XSLT incorporado.

**Verificar**



PARA SABER MÁS

**Los depuradores**

Los depuradores son herramientas que permiten seguir paso a paso la ejecución de la transformación de un archivo XML. Permiten localizar y corregir errores de una manera más rápida y cómoda que si no se dispusiese de depurador.

Solo están disponibles en herramientas potentes que, normalmente, son de pago como Oxygen, Altova o EditiX.

Con estas herramientas, es posible ver la plantilla que está ejecutándose, los elementos XML implicados en la transformación y qué salida está produciendo dicha transformación.

**5.6.1. Oxygen XML Editor**

El editor XSLT de Oxygen es uno de los más utilizados por los desarrolladores XSLT. Pueden realizarse todas las acciones necesarias con los documentos XSLT, como puede ser su creación, edición, testeo o validación.

**Funciones del editor XSLT de Oxygen XML Editor**

### 5.6.2. Altova Editor XSLT

El editor XSLT de Altova tiene todos los aspectos de un editor avanzado como el coloreado de sintaxis, plegamiento de código, marcadores, finalización de código, etc.

Asimismo tiene funciones de herramientas avanzadas como la generación y evaluación de XPath, transformación de alta velocidad y explorador web integrado.

#### Funciones de Altova Editor XSLT



#### Recursos web

www

A través de los siguientes enlaces podrás acceder a las páginas de Oxygen XML Editor, Altova Editor XSLT y XPontus XML Editor.



PARA SABER MÁS

#### Aceleradores de la velocidad de transformación

En muchas herramientas de nivel profesional, el *kernel* del sistema de transformación tiene un acelerador u optimizador de velocidad. Eso quiere decir que, cuando los ficheros son muy grandes, hay ciertos cuellos de botella que hacen que las transformaciones sean lentas.

Estos optimizadores de velocidad realizan cambios en los programas (refactorización) sin que se vea afectada la funcionalidad ni el formato de salida con unos patrones predefinidos. De tal manera que, si el optimizador constata que dichos cambios mejoran la velocidad de transformación, los realiza y, si ve que no suponen ninguna mejora, los obvia.

### 5.6.3. Saxon

Saxon es un procesador XSLT desarrollado por Saxonica Limited. Tiene varias versiones (*home edition*, *professional edition* y *enterprise edition*). De estas versiones, solamente la *home-edition* es *open source* bajo licencia Mozilla Public License versión 2.0. La versión más completa es la *enterprise edition*.

**Funciones de Saxon****Recursos web**

A través de estos enlaces podrás acceder a la página web de Saxon, desde la que puedes descargar cualquier versión de esta herramienta, y a Saxonica, donde podrás encontrar más información sobre cómo utilizar esta herramienta.

**Ejercicio resuelto 5.2**

Transforma el fichero RAQUETAS.XML del ejercicio práctico guiado 5.1 con la herramienta Saxon.

**Solución****5.6.4. XPontus XML Editor**

XPontus es un editor XML *open source* no muy sofisticado, pero que tiene muchas funciones que pueden ayudar a un programador a realizar un pequeño proyecto con XML.

**Funciones de XPontus****Ejercicio práctico guiado 5.2. XPontus****5.6.5. Herramientas online**

Se recomienda utilizar XSL Test Online, que puede servir para hacer transformaciones en línea desde cualquier dispositivo.

Estas herramientas tienen la ventaja de poder ejecutarse desde prácticamente cualquier entorno, pero no ofrecen las mismas funcionalidades que una herramienta de escritorio como Altova u Oxygen, por ejemplo.

Para una programación intensiva o a nivel profesional, este tipo de herramientas a veces se quedan cortas. Sobre todo a la hora de depurar errores. Es más difícil depurar cuando la herramienta ayuda poco.

**Recurso web****www**

Con este enlace podrás acceder a la página web de FreeFormatter XSL, herramienta que, al igual que XSL Test Online, es muy intuitiva y prácticamente ofrece un servicio de procesado XSLT sin ninguna funcionalidad añadida.

**Ejercicio práctico guiado 5.3. XSLT 2.0 con FreeFormatter XSL****Resumen**

- Los ficheros XML permiten organizar la información de forma jerárquica. Estos documentos son legibles, pero hay que procesar la información para darle un valor añadido o hacerla comprensible para cualquier persona.
- Se necesita alguna herramienta para transformar esa información XML en otro documento XML o en un documento que pueda leerse o imprimirse. Ahí entra en juego XSLT, que es un estándar para transformar esos documentos XML.
- XSLT va por la versión 3.0, lo que implica la importancia de este estándar.
- Con XSL, puede transformarse, formatearse y buscarse contenido en ficheros XML.
- XPath se utiliza para navegar entre nodos (seleccionar nodos).
- XSL-FO permite formatear objetos XML.
- XSLT permite transformar un fichero XML en otro documento como HTML o XML.
- XSL es más que un lenguaje de hojas de estilo. CSS hace un documento más legible, pero sigue siendo el mismo), mientras que XSL transforma un documento en otro.
- Una de las ventajas de la versión 3.0 de XSLT es el *streaming* de los documentos fuente.
- Etiqueta del fichero XML que hace referencia a la hoja de estilo XSL:

```
<?xml-stylesheet type="text/xsl" href="RAQUETAS.XSL"?>
```

- Orden template:

```
<xsl:template match="/">
```

- Se distingue la orden y el atributo *match* que asocia en XPath la plantilla a un nodo XML.
- Etiquetas más usuales:
  - <xsl:value-of>. Extrae el valor de un elemento XML y lo añade al flujo de salida de la transformación.
  - <xsl:for-each>. Selecciona todos los elementos uno a uno de una lista determinada.
  - <xsl:sort>. Ordena alfabéticamente un grupo de nodos.

- `<xsl:if>`. Estructura condicional. Se ejecutará su contenido siempre y cuando se cumpla la condición del IF.
- `<xsl:choose>`. Estructura condicional parecida a la anterior, pero permite optar entre múltiples condiciones.
- Herramientas de procesamiento:
  - *De pago*. Tienen depurador, se utilizan a nivel profesional y poseen muchas funciones.
    - Oxygen.
    - Altova.
    - Editix.
  - *Open source*. Más modestas, tienen buenos procesadores, pero un IDE no tan logrado.
    - Saxon.
    - Xalan.
    - Xpontus.
    - *Online*. Disponibles en cualquier dispositivo y sistema, aunque tienen pocas funciones.
      - Xsl Test Online.
      - FreeFormater XSL.

## Ejercicios prácticos resueltos

### 1. Teniendo el siguiente fichero XML:

```
<?xml version="1.0" encoding="UTF-8"?>
<ciclos>
    <ciclo codigo="ASIR">
        <nombre>Administración de Sistemas Informáticos en Red</nombre>
        <grado>Superior</grado>
    </ciclo>
    <ciclo codigo="DAM">
        <nombre>Desarrollo de Aplicaciones Multiplataforma</nombre>
        <grado>Superior</grado>
    </ciclo>
    <ciclo codigo="DAW">
        <nombre>Desarrollo de Aplicaciones Web</nombre>
        <grado>Superior</grado>
    </ciclo>
    <ciclo codigo="SMR">
        <nombre>Sistemas Microinformáticos y Redes</nombre>
        <grado>Medio</grado>
    </ciclo>
</ciclos>
```

Realiza un fichero XSL que produzca la siguiente salida:

```
<html>
  <body>
    <p>
      <strong>Administración de Sistemas Informáticos en Red</
      strong>
    </p>
    <p>
      <strong>Desarrollo de Aplicaciones Multiplataforma</
      strong></p>
    <p><strong>Desarrollo de Aplicaciones Web</strong></p>
    <p><strong>Sistemas Microinformáticos y Redes</strong></p>
  </body>
</html>
```

Solución 

2. Modifica el fichero XSL para que ahora la salida sea la siguiente:

```
<html>
  <body>
    <ul>
      <li>
        <strong>
          Administración de Sistemas Informáticos en Red
        </strong>
      </li>
      <li>
        <strong>Desarrollo de Aplicaciones Multiplataforma</
        strong>
      </li>
      <li>
        <strong>Desarrollo de Aplicaciones Web</strong>
      </li>
      <li>
        <strong>Sistemas Microinformáticos y Redes</strong>
      </li>
    </ul>
  </body>
</html>
```

Solución 

3. Cambia el fichero XSL para que ahora la salida sea la siguiente:

```
<html>
  <body>
    <ul>
      <li>
        <strong>
          ASIR:Administración de Sistemas Informáticos en Red
        </strong>
      </li>
      <li>
        <strong>
          DAM:Desarrollo de Aplicaciones Multiplataforma
        </strong>
      </li>
      <li>
        <strong>DAW:Desarrollo de Aplicaciones Web</strong>
      </li>
      <li>
        <strong>SMR:Sistemas Microinformáticos y Redes</strong>
      </li>
    </ul>
  </body>
</html>
```

**Solución** 

4. Transforma el fichero XSL para que ahora la salida sea la siguiente:

```
<html>
  <body>
    <h1>GRADO SUPERIOR</h1>
    <ul>
      <li>
        <strong>
          ASIR:Administración de Sistemas Informáticos en Red
        </strong>
      </li>
      <li>
        <strong>
          DAM:Desarrollo de Aplicaciones Multiplataforma
        </strong>
      </li>
      <li>
        <strong>DAW:Desarrollo de Aplicaciones Web</strong>
      </li>
    </ul>
  </body>
</html>
```

**Solución** 

## Ejercicios prácticos



1. Teniendo el siguiente fichero XML:

```
<?xml version="1.0"?>
<?xmlstylesheet type="text/xsl"?>
<hello-world>
  <yo>Juan Carlos Moreno</yo>
  <saludo>¡Hola Mundo!</saludo>
</hello-world>
```

Y el siguiente fichero XSL:

```
<?xml version="1.0"?>
<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform" version="1.0">
  <xsl:template match="/hello-world">
    <HTML>
      <HEAD>
        <TITLE></TITLE>
      </HEAD>
      <BODY>
        <H1> <xsl:value-of select="saludo" /> </H1>
        <xsl:apply-templates select="yo" />
      </BODY>
    </HTML>
  </xsl:template>
  <xsl:template match="yo">
    <DIV>
      De: <strong><xsl:value-of select=". " /> </strong>
    </DIV>
  </xsl:template>
</xsl:stylesheet>
```

Crea la salida mediante más de alguna herramienta citada en el capítulo (se recomienda utilizar XPontus y alguna herramienta *online* como FreeFormatter XSL).

2. Disponiendo del siguiente fichero XML:

```
<?xml version="1.0" encoding="utf-8"?>
<?xmlstylesheet type="text/xsl" href="discos.XSL"?>
<CATALOGO>
  <DISCO>
    <TITULO>ME VA LA VIDA</TITULO>
    <ARTISTA>JULIO IGLESIAS</ARTISTA>
    <PAIS>ESPAÑA</PAIS>
    <COMPANIA>Kamaleon</COMPANIA>
    <PRECIO>9.80</PRECIO>
    <ANIO>1985</ANIO>
  </DISCO>
```

[.../...]

```

<DISCO>
    <TITULO>GRANDES EXITOS</TITULO>
    <ARTISTA>UN PINGUINO EN MI ASCENSOR</ARTISTA>
    <PAIS>PORTUGAL</PAIS>
    <COMPANIA>Pingu-records</COMPANIA>
    <PRECIO>14.80</PRECIO>
    <ANIO>1992</ANIO>
</DISCO>
<DISCO>
    <TITULO>UNCHAIN MY HEART</TITULO>
    <ARTISTA>JOE COCKER</ARTISTA>
    <PAIS>USA</PAIS>
    <COMPANIA>EMI</COMPANIA>
    <PRECIO>9.20</PRECIO>
    <ANIO>1987</ANIO>
</DISCO>
</CATALOGO>

```

Desea conseguirse el siguiente formato HTML de salida:

```

<html>
    <body>
        <h1>CATALOGO DE DISCOS</h1>
        <table>
            <tr>
                <th>TITULO</th>
                <th>ARTISTA</th>
            </tr>
            <tr>
                <td>ME VA LA VIDA</td>
                <td>JULIO IGLESIAS</td>
            </tr>
            <tr>
                <td>GRANDES EXITOS</td>
                <td>UN PINGUINO EN MI ASCENSOR</td>
            </tr>
            <tr>
                <td>UNCHAIN MY HEART</td>
                <td>JOE COCKER</td>
            </tr>
        </table>
    </body>
</html>

```

Crea el fichero XSL de transformación.

3. Modifica el ejercicio práctico 2 para que solamente aparezcan los discos de Portugal.
4. Transforma el ejercicio práctico 2 para conocer el dinero que vale la colección completa.
5. Añade varios discos al catálogo y modifica el ejercicio práctico 2 para que aparezcan solamente los discos del año 1990 en adelante.
6. Muestra todos los discos en una tabla, pero el fondo será verde si el precio es mayor a 10 euros y amarillo en caso contrario.

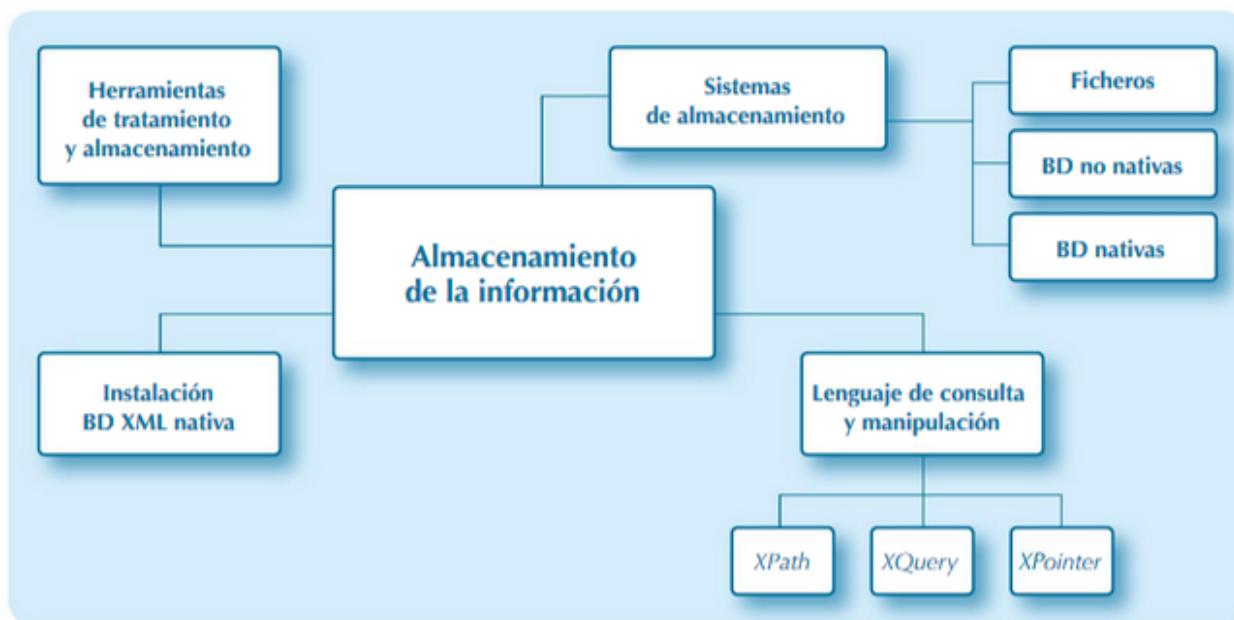


# Almacenamiento de la información

## Objetivos

- ✓ Con este capítulo, se pretende que trabajes y profundices más en las posibilidades que ofrece el lenguaje XML.
- ✓ El auge de las bases de datos No-SQL es un hecho y cada vez están utilizándose más en la industria del software. Las bases de datos XML ofrecen una serie de características que no tienen las bases de datos relacionales y, a la hora de procesar la información, este tipo de bases de datos son más eficientes.
- ✓ Es importante que conozcas y manejes con soltura los lenguajes XPath y XQuery, pues son la herramienta indispensable para poder gestionar la información de una base de datos XML.
- ✓ Finalmente, se desarrolla la instalación de una base de datos XML nativa como es eXist-db y la descripción de herramientas para que puedas ejecutar lenguajes de consulta sobre ella.

## Mapa conceptual



## Glosario

**CSV.** Sigla de *comma-separated values*. Formato tabular para almacenar datos, es muy utilizado por su sencillez.

**Feeds XML RSS y Atom.** Tanto RSS como Atom son dos tipos de *feed* que utilizan XML. Un *feed* es una forma de difundir contenido web para mantener a los suscriptores actualizados en todo momento.

**JSON.** Sigla de *JavaScript object notation*. Formato para intercambiar datos.

**Middleware.** Software que se interpone entre otros dos *software* para mediar o agilizar entre ambos. Las funciones que puede hacer un *middleware* son muy diversas: homogeneizar y racionalizar el acceso a los recursos, agilizar un sistema, servir de árbitro, traducir las peticiones, etc.

**No-SQL.** Bases de datos que, a diferencia de las bases de datos relacionales tradicionales que utilizan SQL, emplean otra estructura de organización y otro tipo de lenguajes para gestionar la información en ella.

**Parser.** Analizador sintáctico que lee un fichero de origen y crea un árbol u otras estructuras que van a ser necesarias en fases posteriores.

**TSV.** Sigla de *tab-separated values*. Formato muy parecido al CSV en el que los valores se almacenan de forma tabular.

### 6.1. Introducción

Una base de datos XML permite almacenar datos en formato XML. Como cualquier base de datos, permite consultar dichos datos y transformarlos en el formato deseado. Generalmente, se asocian a base de datos documentales.

El uso del XML como formato para el intercambio de información hace necesario la utilización de este tipo de bases de datos. De esa manera, el almacenamiento, consulta y procesamiento se hacen más eficientes dado que los gestores no tienen que realizar conversiones, sino que pueden interpretar los datos XML en sí mismos.

Puede considerarse un documento XML como una base de datos, puesto que almacena información y tiene una estructura determinada. Además de eso, puede ser consultado mediante lenguajes como XQuery y XPath.

Las bases de datos XML son una buena solución cuando la información por almacenar tiene estructura jerárquica. Existen muchas áreas donde se utilizan actualmente bases de datos XML en lugar de otro tipo de productos:

- *Bases de datos de fabricación.* Un producto puede estar formado por X piezas y esas piezas, a su vez, estar formadas por otras subpiezas. Imagínese un coche, estará formado por un chasis, un embrague, una caja de cambios, volante, ruedas, etc. La caja de cambios, a su vez, estará formada por otras piezas que, a su vez, están formadas por otras, etc. Como puede intuirse, un documento XML podría ajustarse mucho a la estructura jerárquica que se ha mencionado.
- *Datos médicos.* Igual que en las bases de datos de fabricación, los datos de pacientes estarán formados por los datos del propio paciente (nombre, fecha de nacimiento, sexo, etc.), diagnósticos asociados, medicación, etc. La medicación estará asociada a un diagnóstico que, a su vez, pertenece a un paciente. Otro tipo de información jerárquica que se representaría de una manera perfecta con un esquema XML.
- *Otro tipo de sistemas.* Como portales de información corporativa, información de catálogos, bases de datos de personas, tiendas *online*, etc.

### Investiga



¿Qué es el Health Level Seven (HL7) y qué relación tiene con el XML?

## 6.2. Sistemas de almacenamiento de información

Existen tres formas de almacenar información XML:

1. En ficheros.
2. En bases de datos XML no nativas llamadas también *XML enabled* (habilitadas).
3. En bases de datos nativas XML.

### 6.2.1. Ficheros

No es la mejor opción, puesto que no puede garantizarse la concurrencia, integridad de atomidad, el nivel de seguridad que ofrecen otros sistemas como las bases de datos, etc.

### 6.2.2. Bases de datos XML habilitadas o *enabled*

Almacenamiento en bases de datos relacionales. Son las llamadas bases de datos XML *enabled* o *habilitadas*. En realidad, es una base de datos relacional que convierte los documentos XML en un esquema relacional y, una vez hecho esto, vuelca los datos en dicho esquema.

Estas bases de datos suelen ser bases de datos tradicionales como las bases de datos relacionales y permiten XML como entrada y pueden generar XML como salida.

La base de datos se encarga de procesar el XML sin necesidad de utilizar ningún *middleware*. Estas bases de datos XML *enabled* presentan los siguientes problemas:

- No puede restaurarse el documento original.
- Diferencia de filosofía. XML es jerárquico, mientras que las bases de datos XML *enabled* son relacionales.

### 6.2.3. Bases de datos XML nativas

Su modelo interno se basa en XML y su unidad de almacenamiento es el documento XML (para las bases de datos relacionales es la fila).

Son las que mejores prestaciones ofrecen con este tipo de documentos. A diferencia de las bases de datos XML *enabled*, puede recuperarse el documento original, puesto que lo almacenan sin alterarlo.

Cuando los documentos son grandes o complejos, las bases de datos XML *enabled* suelen generar problemas y la mejor opción es utilizar una base de datos nativa.

Además, estas bases de datos utilizan el modelo jerárquico del documento XML mediante nodos y cada nodo tendrá sus comentarios, atributos, instrucciones, etc.

En estos entornos nativos XML, ya no se utilizan lenguajes como SQL, sino otros adaptados a este modelo como pueden ser XQuery y XPath.

Algunas bases de datos XML son:

- *MarkLogic*. Con licencia propietaria.
- *BaseX*. Con licencia BSD.
- *eXist-db*. Con licencia LGPL.
- *sedna*. Con licencia Apache.



**Figura 6.1**  
Algunas bases de datos XML.

#### Actividades propuestas



Responde si las siguientes afirmaciones son verdaderas o falsas y razona tu respuesta:



**6.1.** Las bases de datos No-SQL son bases de datos siempre más eficientes que las relacionales.



**6.2.** JSON es la sigla de *Java standard object notation*. Un estándar de Java para el intercambio de datos.

**Verificar**



SABÍAS QUE...

- ✓ eXist-db fue creada en el año 2000 por Wolfgang Meier y, en el 2006, en InfoWorld, fue considerada la mejor base de datos XML. Se utiliza de forma masiva en la arquitectura de aplicaciones web XRX.
- ✓ eXist-db es una base de datos XML de código abierto que está catalogada como una base de datos no SQL. Es una base de datos nativa XML y proporciona soporte para JSON, HTML y documentos binarios. Puede trabajar con XQuery y con XSLT, además de con sus herramientas de consulta y lenguajes de programación.

**Investiga**

¿Qué es JSON y para qué se utiliza actualmente en los lenguajes de programación?

### 6.3. Inserción y extracción de información en XML

Muchas herramientas son capaces de guardar y leer documentos en XML. Esto es así porque XML es un formato universal y sencillo de generar y *parsear* (leer). No obstante, en este apartado, va a presentarse una herramienta muy interesante para cualquier programador web que se llama *ImportXML* y está en la *suite* de Google Docs, específicamente en la hoja de cálculo.

¿Por qué es diferente esta herramienta? Porque permite importar datos de formatos estructurados de datos como puede ser XML, HTML, CSV, TSV, Feeds XML RSS y Atom.

**Ejercicio resuelto 6.1**

Quieren importarse todos los enlaces que tenga la página web de la Wikipedia que hablan de HTML: <https://es.wikipedia.org/wiki/HTML>  
Utiliza la función ImportXML de las hojas de cálculo de Google.

**Solución** **Ejercicio propuesto 6.1**

ImportXML también es capaz de leer datos de una celda e incorporarlos a la fórmula.  
Coloca en la celda A1 el siguiente texto: "<https://es.wikipedia.org/wiki/HTML>" y en la celda A2 el siguiente texto: "//a/@href".  
Ahora intenta importar los datos utilizando la fórmula: IMPORTXML(A1;A2).

Como puede observarse, la sintaxis de esta fórmula es muy sencilla:

```
IMPORTXML(url; consulta_xpath)
```

- *url*: URL de la página que desea examinarse (hay que incluir el protocolo como, por ejemplo, `http://`). La URL tiene que ir entre comillas (también puede ser una referencia a una celda).
- *consulta\_xpath*: es la consulta en lenguaje XPath que va a ejecutarse sobre la URL anterior. XPath se estudiará en un apartado posterior de este capítulo.

*¿Para qué puede servir esta herramienta?* Para extraer cualquier información de una página web o cualquier otro formato estructurado de datos. Imagine que quieren extraerse los correos electrónicos de los integrantes de un foro, los *nicks*, los hiperenlaces, etc. Todas esas tareas que pueden llevar mucho tiempo pueden hacerse de forma rápida con esta herramienta.

Para muchos *webmasters*, es una herramienta muy útil. Por ejemplo, las empresas de *marketing online* podrían utilizar estas herramientas para obtener los correos de contacto de páginas web que le interesan. Crear una base de datos de contactos con este tipo de herramientas puede ser relativamente rápido.

### Actividad propuesta 6.3



Marca las características de una base de datos *enabled*:

- a) No permiten restaurar el documento original.
- b) Suelen ser relacionales.
- c) Suelen almacenar el fichero XML.

Verificar



## 6.4. Búsqueda de información en documentos XML: lenguajes de consulta y manipulación

Cuando se habla de lenguajes de consulta, generalmente se refiere a XPath y a XQuery. La diferencia entre ambos es que XPath es un lenguaje para seleccionar nodos en lenguaje XML dentro de un DOM, mientras que XQuery, además de ser un superconjunto de XPath, ofrece sintaxis FLWOR. Esto quiere decir que tiene cláusulas For, Let, Where, Order by y Return.

Cuando se está ante una consulta XQuery, la sensación es que es muy parecido al SQL.

Además de estos dos, también existe el XPointer, que incluye XPath, pero que es mucho más sofisticado, puesto que permite seleccionar nodos dentro de fragmentos XML o incluso partes de nodos (no el nodo completo).

### 6.4.1. XPath

XPath es un lenguaje de rutas para XML (*path language*) y utiliza una sintaxis *path like* para identificar y navegar entre los nodos de un documento XML.



SABÍAS QUE...

XPath es una recomendación del W3C desde el 16 de noviembre de 1999 en el que apareció la versión 1.0 de XPath. Actualmente, la versión es la 3.0 y data del 2014.

Una de las razones que hacen a XPath tan potente es por el número de funciones predefinidas de las que dispone (más de 200). Existen funciones para tratar *strings*, booleanos, valores numéricos, comparar fechas y horas, manipulación de secuencias y nodos, etc.

Las expresiones XPath pueden utilizarse en Java, JavaScript, PHP, Python, C, C++, etc.

XPath se utiliza en XSLT. Se recuerda que XSLT (*extensible stylesheet language for transformations*) es un lenguaje con el que pueden transformarse documentos XML en otros documentos XML, HTML o texto plano (*plain text*).

XPath utiliza un analizador cuya función es crear un árbol de nodos a partir de un documento XML. Los nodos creados en estos árboles son de siete tipos: nodo raíz, elemento, texto, atributo, *namespace* (espacio de nombres), instrucción procesable y comentario.

A continuación se muestra un documento XML y el árbol generado por el analizador de XPath.

```
<?xml version="1.0" encoding="UTF-8"?>
<concesionario>
    <coche>
        <marca>Volkswagen</marca>
        <modelo>Passat</modelo>
        <color>Azul</color>
        <año>2005</año>
        <precio>7900.00</precio>
    </coche>
</concesionario>
```

A partir de este documento XML, XPath creará un árbol de nodos (figura 6.2).

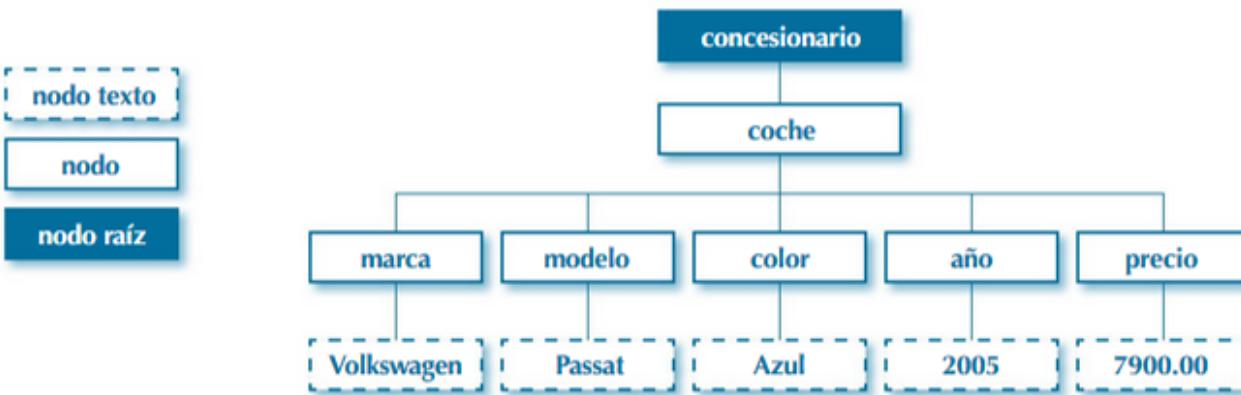


Figura 6.2  
Los nodos en XPath. Un ejemplo.

### A) Los nodos en XPath

Como puede verse en la figura 6.2, los nodos formarán parte de la estructura arborescente creada por el *parser* de XPath. En él, pueden encontrarse:

- *Nodo*. Un ejemplo de nodo sería coche, modelo, color, etc.
- *Nodo raíz*. Es el que se sitúa en la parte superior del árbol (concesionario).
- *Valor atómico*. En el ejemplo anterior, podría ser “Passat” o “2005”.
- *Ítem o elemento*. Un ítem o elemento puede ser un nodo o un valor atómico.

Los nodos en las estructuras arborescentes de XML están relacionados unos con otros. Pueden encontrarse nodos:

- *Padre (parent)*. Un nodo es padre de otro cuando este último desciende de él. Por ejemplo, coche desciende de concesionario.
- *Hijo (children)*. El hijo es el nodo que desciende del nodo padre. Por ejemplo, coche es nodo hijo de concesionario.
- *Hermanos (siblings)*. Son aquellos nodos hijo del mismo padre (marca, modelo, color, etc.).
- *Antecesores (ancestors)*. Los nodos coche y concesionario serán antecesores de color, puesto que jerárquicamente están en un orden superior directo.
- *Descendientes (descendant)*. Son los hijos, hijos de los hijos, etc., de un nodo.

## B) Cómo seleccionar los nodos de un documento XML

Para seleccionar los nodos de un documento XML, XPath utiliza expresiones y, de esa manera, un nodo puede ser seleccionado siguiendo una serie de pasos o ruta.

Las expresiones más utilizadas se presentan en el cuadro 6.1.

**CUADRO 6.1**

Expresiones utilizadas por XPath en la selección de nodos

Expresión	Descripción
nombre_nodo	Selecciona todos los nodos con el nombre <i>nombre_nodo</i> .
/	Selecciona desde el nodo raíz.
//	Selecciona los nodos del documento que cumplan los criterios de selección desde el nodo actual sin importar dónde se encuentren.
.	Selecciona el nodo actual.
..	Selecciona el nodo padre del nodo actual.
@	Selecciona atributos.

### Ejemplo

#### Cómo funciona XPath

En este ejemplo, va a desarrollarse de una forma sencilla cómo funciona XPath embebido en un lenguaje de programación como puede ser JavaScript.

Paso 1. En primer lugar, hay que realizar este ejercicio en un servidor web. Si no se dispone de un servidor web en internet, puede instalarse uno de forma local (XAMPP podría ser una buena y rápida opción).

Paso 2. En segundo lugar, debería tenerse un documento XML al cual realizar consultas. Partiendo del ejemplo, se ha completado el fichero con más elementos y el resultado sería el siguiente:

```
<?xml version="1.0" encoding="UTF-8"?>
<concesionario>
    <coche combustible="D">
        <marca>Volkswagen</marca>
        <modelo>Passat</modelo>
        <color>Azul</color>
        <año>2005</año>
        <precio>7900.00</precio>
    </coche>
    <coche combustible="G">
        <marca>Volkswagen</marca>
        <modelo>Touran</modelo>
        <color>Azul</color>
        <año>2007</año>
        <precio>8200.00</precio>
    </coche>
    <coche combustible="D">
        <marca>Opel</marca>
        <modelo>Astra</modelo>
        <color>Negro</color>
        <año>2013</año>
        <precio>8490.00</precio>
    </coche>
</concesionario>
```

Este fichero se denominará *coches.xml* y será el fichero que servirá para realizar la consulta.

Paso 3. El siguiente paso es crear un archivo *coches.html* que contendrá un *script* que hará una consulta en XPath sobre los datos del fichero *coches.xml*.

```
<!DOCTYPE html>
<html>
<body>

<p id="listado"></p>

<script>
var xhttp = new XMLHttpRequest();
xhttp.onreadystatechange = function() {
    if (this.readyState == 4 && this.status == 200) {
        recuperaDatos(xhttp.responseText);
    }
};
xhttp.open("GET", "coches2.xml", true);
xhttp.send();
```

[.../...]

```

function recuperaDatos(xml) {
    var datos = "";
    path = "/concesionario/coche/marca"
    if (xml.evaluate) {
        var nodos = xml.evaluate(path, xml, null, XPathResult.ANY_TYPE,
            null);
        var resultado = nodos.iterateNext();
        while (resultado) {
            datos = datos + "marca: " + resultado.childNodes[0].nodeValue +
                "<br>";
            resultado = nodos.iterateNext();
        }
    }
    document.getElementById("listado").innerHTML = datos;
}
</script>
</body>
</html>

```

Nótese que la variable *path* (en negrita) contiene los nodos que van a ser seleccionados en el procedimiento *recuperaDatos()*.

*Paso 4.* Es el momento de probar el código anteriormente escrito. Bastaría con escribir en un navegador la siguiente dirección:

<http://localhost/coches.html>

Como puede observarse, la prueba se ha hecho en un servidor local. El código anterior está probado con Google Chrome y muestra el siguiente resultado:

marca: Volkswagen  
 marca: Volkswagen  
 marca: Opel

Nota: Es posible que haya que cambiar el código si el navegador desde el cual va a ejecutarse no es Google Chrome.

*Paso 5.* Cambiando los criterios de selección. Imagínese que se sabe qué precio es un nodo que desciende de concesionario, pero no en qué nivel está ni cuáles son sus antecesores. Para obtener el precio de los coches del concesionario, se asignaría el siguiente valor a la variable *path*:

path = "concesionario//precio"

De esa manera, se obtendrá el contenido del nodo precio sin importar la ruta siempre y cuando sea descendiente del nodo concesionario.

También podría haberse utilizado la siguiente expresión para seleccionar el precio de todos los coches del concesionario:

path = "//precio"

El resultado sería el mismo:

precio: 7900.00  
 precio: 8200.00  
 precio: 8490.00

**Actividades propuestas**

Responde si las siguientes afirmaciones son verdaderas o falsas y razona tu respuesta:

- V F 6.4.** Las bases de datos XML nativas también se llaman *XML enabled* (habilitadas).
- V F 6.5.** En las bases de datos *enabled*, no puede restaurarse el documento original.

**Verificar****C) Utilizando predicados para mejorar las búsquedas**

Los predicados son modificadores que se colocan entre corchetes y sirven para extraer un nodo concreto o un nodo con un determinado valor.

Teniendo en cuenta el ejemplo anterior, se ofrece una serie de predicados con el resultado correspondiente en el cuadro 6.2.

**CUADRO 6.2****Lista de predicados con su resultado**

Expresión	Resultado
/concesionario/coche[1]/marca	Seleccionaría la marca del primer coche del concesionario. En algunos navegadores, los nodos empiezan a contar desde el 0.
/concesionario/coche[last()]/marca	Seleccionaría la marca del último coche del concesionario.
/concesionario/coche[last()]/marca	Seleccionaría la marca del penúltimo coche del concesionario.
/concesionario/coche[position()<4]/marca	Seleccionaría la marca de los tres primeros coches del concesionario.
/concesionario/coche[precio>8000.00]/modelo	Selecciona el modelo de aquellos coches cuyo precio sea superior a 8000 euros. El resultado es el siguiente: modelo: Touran modelo: Astra

**D) Combinar varias rutas**

En ocasiones, no solamente quiere seleccionarse un nodo, sino varios nodos con distinto contenido. Imagínese que quiere conocerse el modelo y el precio de los coches del concesionario. En este caso, se utilizará el operador | que funciona como un AND lógico.

La ruta utilizada es la siguiente:

```
path = "concesionario/coche/modelo | concesionario/coche/precio"
```

Y el resultado de aplicar el filtro sobre el fichero XML del ejemplo será:

```

Passat
Touran
Astra
7900.00
8200.00
8490.00

```

La pregunta que puede hacerse es ¿por qué aparecen primero los modelos y luego el precio?

Muy sencillo, porque XPath selecciona nodos y el orden es el dado en la variable *path* de la ruta utilizada.

## E) Comodines

XPath permite el uso de comodines (cuadro 6.3).

**CUADRO 6.3**

Comodines cuyo uso permite XPath

Comodín	Descripción
*	Selecciona cualquier nodo elemento.
@*	Selecciona cualquier nodo atributo.
node()	Selecciona cualquier nodo de cualquier clase.

Por lo tanto, si se utilizase la siguiente ruta:

```
path = "concesionario/coche/modelo | concesionario/coche/precio"
```

El resultado de aplicarla a los datos del fichero del ejemplo sería el siguiente:

```

Volkswagen
Passat
Azul
2005
7900.00
Volkswagen
Touran
Azul
2007
8200.00
Opel
Astra
Negro
2013
8490.00

```

**Ejercicio resuelto 6.2**

Prueba si funciona la siguiente ruta y, en caso afirmativo, indica qué resultado producirá:

```
path = "/concesionario/coche[marca='Volkswagen']/modelo"
```

**Solución** 

### 6.4.2. XQuery

XQuery es para XML lo que SQL es para una base de datos tradicional. Un ejemplo de consulta en XQuery sería el siguiente:

```
for $x in doc("coches.xml")/concesionario/coches  
where $x/precio>25000  
order by $x/modelo  
return <li>{data($x/modelo)}</li>
```

Como puede apreciarse, es muy similar a SQL en su semántica, pero pueden verse características de los lenguajes de programación como cláusulas *for*, *return*, *let*, etc. Las expresiones que se utilizan en XQuery se denominan *sentencias FLWOR* (For, Let, Where, Order by y Return).

**SABÍAS QUE...**

- ✓ XQuery 1.0 es una recomendación del W3C desde el 23 de enero del 2007.
- ✓ XQuery es compatible con otros estándares del W3C como XML, XSLT, XPath o Namespaces.

Como puede observarse, XQuery es un lenguaje de consulta y extracción de información y atributos de archivos XML. El ejemplo de consulta en XQuery sería el siguiente en lenguaje normal:

Selecciona todos aquellos modelos de coches que estén en el concesionario cuyo precio sea mayor a 25 000 euros y que estén registrados en el archivo coches.xml

XQuery 1.0 y XPath 2.0 comparten las mismas funciones, operadores y el mismo modelo de datos.

**PARA SABER MÁS**

- ✓ XPath se utiliza para navegar a través de elementos y atributos en un documento XML.
- ✓ XPath es un estándar del W3C y XQuery y XPointer se basan en las expresiones XPath.

*¿Para qué utilizar XQuery?*

- Para transformar datos de XML a XHTML.
- Para realizar búsquedas en documentos XML a través de la web.
- Para realizar informes con datos de estadísticas, etc.
- Para ser utilizado por un servicio web para mostrar información.
- Procesar ficheros XML.

Para entender mejor el concepto, véase cómo se integra XQuery en HTML para realizar consultas en documentos XML y cuáles son los resultados obtenidos. En primer lugar, se dispone del siguiente documento HTML el cual, que tiene código XQuery:

```
<html>
<body>
  <h1>
    Concesionarios del Sur: Modelos de coches</h1>
  <ul>
    { for $x in doc("coches.xml")/concesionario/coches
      where $x/precio>25000
      order by $x/modelo
      return <li>{data($x/modelo)}</li>
    }
  </ul>
</body>
</html>
```

Al procesar el intérprete de XQuery la expresión anterior, generará un resultado HTML que se fusiona con el ya existente y el resultado será el siguiente:

```
<html>
<body>
  <h1>
    Concesionarios del Sur: Modelos de coches</h1>
  <ul>
    <li>Ferrari Testa Rosa</li>
    <li>Lamborghini Diablo</li>
    <li>Mercedes SLK</li>
    <li>Range Rover Vogue</li>
  </ul>
</body>
</html>
```

## 6.5. Almacenamiento XML nativo

Ya se ha hablado al comienzo de este capítulo de que existen muchas bases de datos XML nativas, así que es el momento de realizar la instalación de una de ellas.

### 6.5.1. Instalación y ejecución de una base de datos XML nativa: eXist-db

Se ha elegido eXist-db por ser una base de datos de código abierto y multiplataforma. Estas características hacen que sea una de las bases de datos No-SQL más difundidas.

**Instalación de eXist-db****Ejecutar eXist-db**

### 6.5.2. BaseX

Ya se ha visto en el apartado anterior una base de datos nativa como es eXist-db. En este apartado, se estudiará una herramienta análoga como es BaseX, que es una base de datos XML bastante fiable, junto con un procesador XQuery 3.1. Suele utilizarse para crear aplicaciones web y una clave de su éxito es que es *open source*.

**WWW**

### Recursos web

A través de estos enlaces podrás acceder a las páginas web de eXist-db y BaseX.



A continuación, se verán varios ejemplos tanto en XPath como XQuery que toman como datos de referencia el siguiente fichero XML:

```
<?xml version="1.0" encoding="iso-8859-1"?>
<vivero>
    <especie siembra="2018">
        <nombre>Litchi</nombre>
        <precio moneda="euro">25</precio>
        <variedad>Kway May Pink</variedad>
        <origen>Filipinas</origen>
        <color_fruto>rojo</color_fruto>
        <color_fruto>rosa</color_fruto>
        <otros_datos>
            <maduración>agosto</maduración>
            <riego> diario</riego>
        </otros_datos>
    </especie>
    <especie siembra="2017">
        <nombre>Longan</nombre>
        <precio moneda="euro">10</precio>
        <variedad>Champoo</variedad>
        <origen> China</origen>
        <color_fruto>marrón</color_fruto>
        <otros_datos>
            <maduración>octubre</maduración>
            <riego> diario</riego>
        </otros_datos>
    </especie>
</vivero>
```

[.../...]

```

<especie siembra="2016">
    <nombre>Litchi</nombre>
    <precio moneda="euro">21</precio>
    <variedad> mauritius</variedad>
    <origen>Florida</origen>
    <color_fruto>rojo</color_fruto>
    <otros_datos>
        <maduración>agosto</maduración>
        <riego> diario</riego>
    </otros_datos>
</especie>
</vivero>

```

### Ejemplos en XPath y en XQuery ➔

## 6.6. Herramientas de tratamiento y almacenamiento de información en formato XML

Actualmente, existen infinidad de herramientas con las cuales poder editar un documento XML (y también HTML). A diferencia de un editor normal, como puede ser el Notepad o Gedit, los editores XML ayudan al usuario porque se *chivan* de los errores que pueda contener el documento y formatean el documento indentándolo, cierran de forma automática las etiquetas, etc. A continuación, se hablará de forma breve de algunas de estas herramientas XML.

### Recurso web ➔

www

Aunque Altova XMLSpy es de pago, puede descargarse una versión de evaluación de su página web.



A continuación, va a explicarse en profundidad una herramienta para el tratamiento de información dentro de un documento como es eXide (herramienta incluida en eXist-db).

### Funciones de eXide (eXist-db) ➔

### Ejercicio práctico guiado 6.1 ➔

### Ejercicio práctico guiado 6.2 ➔

www

## Recurso web

Oxygen XML Editor es una herramienta de pago, sus precios comienzan por los 99 dólares.

Ofrece una *suite* de herramientas de desarrollo y creación de ficheros XML. Disponible prácticamente en todas las plataformas, puede utilizarse de forma *stand alone* o como *plugin* de Eclipse.

El editor es uno de los mejores del mercado y permite salvar documentos no solo en XML, sino también en formato PDF, ePUB o HTML, entre otros.

Entre otras opciones, permite trabajar de forma colaborativa con esta herramienta manteniendo un repositorio con Subversion (sistema de control de versiones de Apache).



## Resumen

- XML es una tecnología que engloba muchos conceptos como el XML, XML namespaces, XML Schema, XSLT, etc.
- Actualmente, XML está omnipresente en el mundo de la informática y se prevé que siga siendo así porque es un estándar del W3C y su uso está promocionándose cada vez más.
- XML está usándose actualmente en muchas aplicaciones, por ejemplo, en dispositivos con bajo ancho de banda o cuyo rendimiento es crítico. XML ha demostrado que es efectivo y muy eficiente.
- Las bases de datos XML cada vez van a hacerse más populares, puesto que pueden ser igual o más potentes que las relacionales según el caso y los lenguajes de consulta y manipulación son igual de completos como puede ser el SQL.
- XML es una tecnología de presente y futuro, por lo tanto, cualquier profesional que quiera progresar en el ámbito web lo debe conocer y manejar con soltura.
- Una base de datos XML permite almacenar datos en formato XML.
- Puede considerarse un documento XML como una base de datos, puesto que almacena información y tiene una estructura determinada.
- La información XML puede ser consultada mediante lenguajes como XQuery y XPath.
- Existen tres formas de almacenar información XML:
  1. En ficheros.
  2. En bases de datos XML no nativas llamadas también *XML enabled* (habilitadas).
  3. En bases de datos nativas XML.
- Las bases de datos *enabled* suelen ser bases de datos relacionales.
- En las bases de datos *enabled*, no puede recuperarse el fichero XML original.
- Las bases de datos nativas XML siguen el modelo jerárquico del documento XML.
- ImportXML es una herramienta de la suite de Google Docs que permite parsear información XML en páginas web.
- XPath es un lenguaje de rutas para XML (*path language*) y utiliza una sintaxis *path like* para identificar y navegar entre los nodos de un documento XML.

- XPath utiliza un analizador cuya función es crear un árbol de nodos a partir de un documento XML.
- Para seleccionar los nodos de un documento XML, XPath utiliza expresiones y, de esa manera, un nodo puede ser seleccionado siguiendo una serie de pasos o ruta.
- XQuery es para XML lo que SQL es para una base de datos tradicional. Se parece a SQL porque pueden utilizarse sentencias FLWOR.

## Ejercicios prácticos resueltos

1. Escribe el siguiente código en un nuevo archivo en eXide:

```
xquery version "3.1";
for $x in doc("/db/coches.xml")/concesionario/coches
where $x/precio>5000
order by $x/modelo
return <li>{data($x/modelo)}</li>
```

Siendo el contenido del fichero coches el siguiente:

```
<concesionario>
  <coche combustible="D">
    <marca>Volkswagen</marca>
    <modelo>Passat</modelo>
    <color>Azul</color>
    <año>2005</año>
    <precio>7900.00</precio>
  </coche>
  <coche combustible="G">
    <marca>Volkswagen</marca>
    <modelo>Touran</modelo>
    <color>Azul</color>
    <año>2007</año>
    <precio>8200.00</precio>
  </coche>
  <coche combustible="D">
    <marca>Opel</marca>
    <modelo>Astra</modelo>
    <color>Negro</color>
    <año>2013</año>
    <precio>8490.00</precio>
  </coche>
</concesionario>
```

Verifica si tiene algún error y, si existe, corríjalo. Una vez corregido, ejecútelo.

**Solución** 

2. Teniendo en cuenta el archivo de datos XML del apartado 6.5.2:
  - a) Obtén el precio de las especies de litchis utilizando XPath.
  - b) Obtén el precio de las especies de litchis, pero utilizando XQuery (FLWOR).
  - c) Obtén el precio de las especies de litchis, pero mostrando solamente el precio sin las etiquetas (utilice *data* o *string*).

**Solución**

3. Utilizando los datos del ejercicio práctico resuelto 2, obtén la maduración del campo "otros\_datos" de las especies que cuesten más de 9 euros.
  - a) Realiza el ejercicio pedido utilizando la orden LET.
  - b) Realiza el ejercicio pedido, pero sin utilizar la orden LET.

**Solución**

4. Utilizando los datos del ejercicio práctico resuelto 3, halla el campo "otros\_datos" de las especies de Litchis cuyo precio sea superior a 24 euros.

**Solución**

5. Determina el campo origen de las especies cuya siembra fue en el 2018.

**Solución**

6. Suma el precio de las especies que se sembraron a partir del 2010.

**Solución**

7. Establece el precio de las especies cuyo nombre empieza por Lo.

**Solución**

## Ejercicios prácticos



1. Investiga qué es XPointer y cuáles son las características de este lenguaje.
2. Con los datos del ejercicio práctico resuelto 1, ejecuta una consulta con XQuery o XPath para conocer los coches cuyo precio esté entre 7900 y 9300 euros.
3. Utiliza la función ImportXML para hacer un listado de los personajes jedi de la siguiente página web:
4. Con los datos del ejercicio práctico resuelto 1, ejecuta una consulta con XPath para conocer los coches cuyo año de matriculación no sea el 2007.

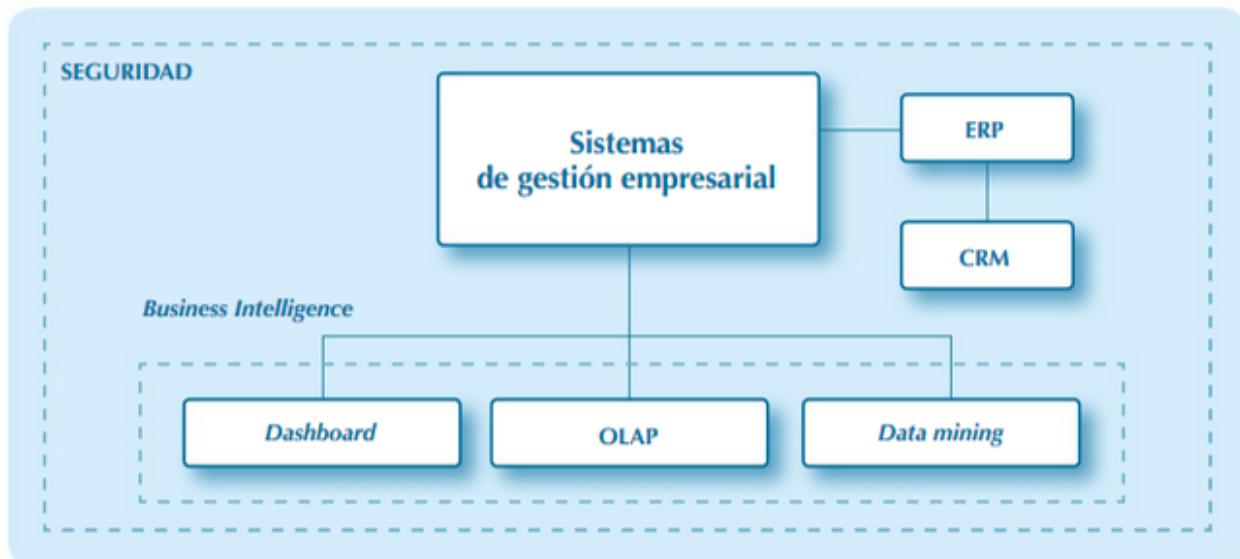


# Sistemas de gestión de la información

## Objetivos

- ✓ Es preciso que conozcas en profundidad los sistemas de gestión que se utilizan actualmente en las empresas, que son los ERP.
- ✓ También debes saber que se utilizan otros como los CRM para fidelizar a los clientes o herramientas y técnicas como la minería de datos o *data mining*, las herramientas OLAP y los *dashboard* o *scorecard* (cuadros de mandos).
- ✓ Igualmente, en el mundo empresarial, se utilizan conceptos como *business intelligence* que tienes que manejar y comprender.
- ✓ Por último y no menos importante, debes comprender cómo funcionan los sistemas de seguridad de todas estas herramientas empresariales.

## Mapa conceptual



## Glosario

**Big data.** Análisis y procesamiento de volúmenes muy grandes de información que no pueden ser tratados de forma normal o convencional. Estos datos pueden ser de todo tipo, estructurados, no estructurados, etc.

**CRM (customer relationship management).** Módulo de un sistema de gestión empresarial cuyo objetivo es la gestión de relaciones con los clientes.

**Data mining.** Disciplina que intenta encontrar patrones dentro de grandes bases de datos o set de datos. Un ejemplo ficticio de esto puede ser que, a los directivos de las empresas, les gustan los coches de color azul oscuro. Sabiendo esto, un hábil comercial intentará ofrecer coches con color azul oscuro a todos sus clientes que sean directivos. De esa manera, gana clientes y ventas.

**Data warehouse.** Almacén de datos especializado para el *data mining*.

**ERP (enterprise resource planning).** Sistema de gestión integral de una empresa. Aunque se asocia a empresas grandes, actualmente está siendo muy demandado y utilizado en las pymes.

**Escalabilidad.** Capacidad de poder adaptarse y crecer un sistema con respecto a la demanda. Cuanta más demanda exista, más potencia podrá tener el sistema.

**Internet de las cosas.** Interconexión de cualquier objeto cotidiano a internet (una lavadora, un frigorífico, la cafetera, etc.).

**JIT (just in time).** Sistema de almacenamiento que tiene como objetivo reducir el stock ofreciendo el mismo servicio. En este sistema, se ahorran muchos costes, pero la organización y el sistema informático tienen que estar perfectamente engrasados para que funcione todo correctamente.

**Licencia GNU.** Licencia de derechos de autor mediante la cual cualquier persona podría usar, compartir y modificar un *software* a su conveniencia. La licencia protege el *software* de usos no otorgados como el comercial, por ejemplo.

**SGE.** Sigla de sistema de gestión empresarial.

**XAMPP.** Megaservidor que aglutina otros servidores como el servidor web Apache, servidor de base de datos MySQL, servidor de aplicaciones PHP y servidor FTP ProFTPD, entre otros.

## 7.1. Introducción

Los sistemas de gestión empresarial se usan en las empresas para prácticamente todas las tareas. Es imposible concebir una empresa que no tenga informatizado todo o parte de su negocio.

Los sistemas de gestión empresarial o *management information systems* se encargarán del almacén o aprovisionamiento, de la gestión de la empresa, de la producción, del control de las ventas, de la logística, etc.

Hace tiempo, cada uno de los departamentos de una empresa utilizaba un *software* distinto de gestión, pero actualmente suele utilizarse *software* que integren toda la gestión.

Este *software* se divide en módulos, los cuales están pensados para gestionar su parcela de la forma más eficiente. El objetivo es maximizar la eficiencia y efectividad del personal de la empresa. En otras palabras, mejorar la productividad.

El punto fuerte de las empresas es la información y su gestión, por lo que, en este capítulo, se ha hecho hincapié en estudiar los sistemas de gestión de la información.

Cualquier empresa deberá gestionar la información de manera:

- *Rápida.* Toda la información tiene que ser recogida y puesta a disposición de los miembros o departamentos de la empresa que la necesiten. Mejor en tiempo real.
- *Global.* Hay que recoger toda la información posible. Aunque no se procese en ese mismo momento, en un futuro podrá hacerse.
- *Puesta a disposición de manera selectiva o piramidal.* Los datos más importantes deberán estar a disposición de los mandos o gestores de la empresa. Estos datos servirán para poder planificar y tomar decisiones, así como para poder definir políticas de empresa.

El objetivo de un sistema de gestión empresarial es el desarrollo de la compañía a todos los niveles mejorando no solo la cadena de producción, sino la toma de decisiones. En futuros apartados, se estudiará el concepto *business intelligence* o inteligencia de negocios, el cual es una pieza fundamental de un SGE.

### 7.1.1. Los CRM

Uno de los sistemas de gestión empresarial que más impacto está teniendo últimamente en las empresas son los CRM, los cuales tienen objetivos ambiciosos, puesto que, además de fidelizar los clientes que ya pueda tener una empresa, intenta ganar y atraer nuevos clientes.

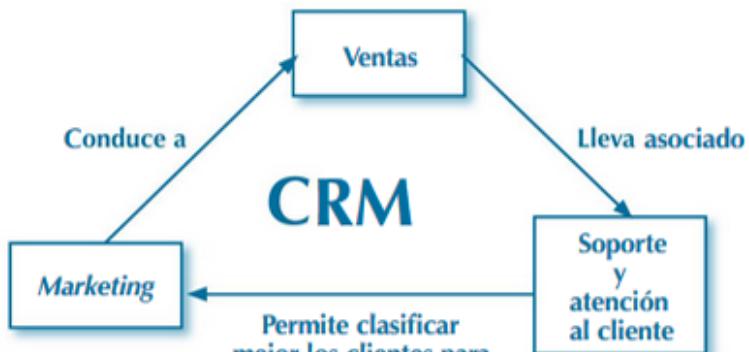
La estrategia de un CRM está probado que reduce el coste de los presupuestos de *marketing* a la vez que mejora el servicio al cliente.

Generalmente, los CRM suelen tener como mínimo tres módulos: los módulos de ventas para gestionar el proceso de posventa, el módulo de *marketing* para mejorar la comercialización y planificación de las actividades de *marketing* dentro de la empresa y algún módulo que gestione las contrataciones, servicios, etc.

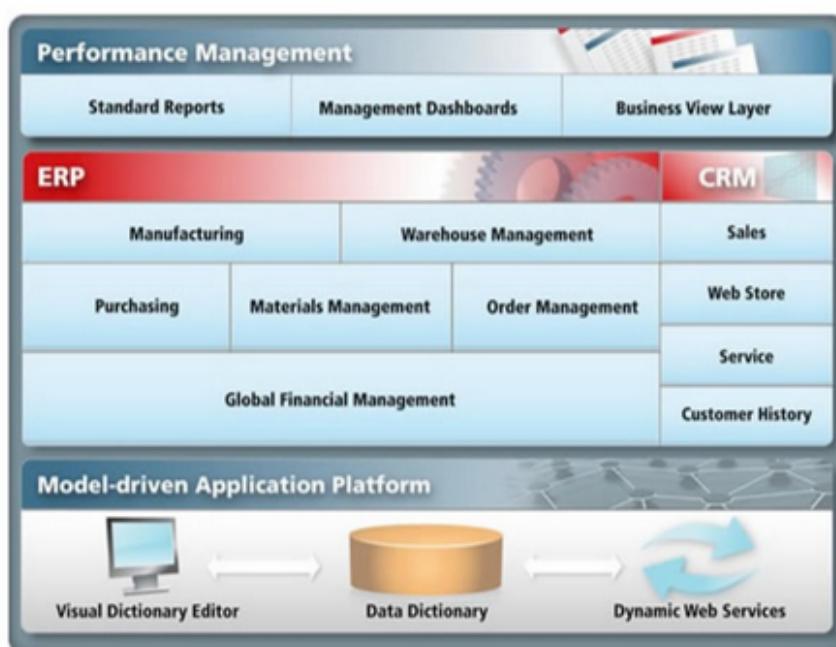
Los sistemas CRM han ido evolucionando y ahora, en vez de utilizar el *email* o el teléfono, su manera de contactar con el cliente es a través de alertas, SMS, redes sociales, etc. El trato con el cliente no se pierde con la venta, sino que es un proceso de larga duración, por lo tanto, la gestión correcta de la información será crucial en este ámbito.

Algunos de los sistemas CRM más utilizados son:

- SugarCRM.
- ZOHO.
- B-Kin.
- Salesforce.
- SalesBoom.
- Vtiger.
- Microsoft Dynamics CRM.



**Figura 7.1**  
Estructura de un CRM.



**Figura 7.2**  
Vista integrada de un sistema de información.

**FUNDAMENTAL** ***Los sistemas de análisis de la información***

Los sistemas de análisis de la información son muy utilizados en la actualidad por muchas empresas. La información es poder y, en este caso, para muchas empresas, gestionarla de la mejor forma es necesario no solo para obtener ventajas competitivas, sino para su propia supervivencia, dado que las demás empresas del sector ya lo hacen. Por lo tanto, ha pasado de ser de un añadido a una necesidad. De la mejor o peor gestión de la información, muchas veces depende el éxito de un negocio.

## 7.2. ERP

Los ERP son sistemas de gestión empresarial que cubren las necesidades de prácticamente todas las áreas de una empresa (finanzas, contabilidad, ventas, compras, recursos humanos, almacén, fabricación, etc.).

**Investiga**

¿Qué es *big data* y cómo está resolviendo problemas que hasta hace poco parecían imposibles de resolver?

Una de los principales requisitos que se pide a un ERP actual es que sea multiplataforma y que pueda integrarse con otros productos como *business intelligence*, *dashboards* o *scorecards*, etc.

Existen ERP libres y propietarios. Aunque un sistema sea libre u *open source*, no quiere decir que salga totalmente gratis a la empresa. Algunos módulos, *plugin* o mejoras tendrán que ser adquiridas en los *markets* de las compañías u organizaciones que soportan el producto. No obstante, la ventaja es la posibilidad de modificar el código, un mejor soporte y compatibilidad y la inexistencia de ataduras o gastos en licencias.

**RECUERDA**

- ✓ Prácticamente, la totalidad de los ERP más usados tienen un módulo de CRM.

Hay muchas empresas que utilizan diversas aplicaciones informáticas para su gestión. En ocasiones, una diferente por departamento.

Generalmente, al no trabajar de forma coordinada, se pierde mucho tiempo y se limita el crecimiento y, a veces, se frena la productividad.

Si desean controlarse los procesos de negocio de una empresa, es necesario tener una herramienta unificada como un ERP, la cual puede permitir a la gerencia la toma de decisiones, reducir costes y centrarse en la gestión a alto nivel.

A continuación, se citan algunos de los ERP más utilizados:

- *Dolibarr.* Software completamente modular en el cual solamente se activan los módulos que se necesiten para la gestión empresarial de pequeñas y medianas empresas, autónomos o asociaciones. Combina un ERP y un CRM. Es *open source* y hay que ejecutarlo en un servidor web como XAMPP (WAMP, MAMP o LAMP) que tenga servidor web Apache, base de datos MySQL y servidor de aplicaciones PHP.
- *Odoo.* Open-source comercial con una versión *community* gratuita y otra de pago llamada *enterprise*. Dependiendo del volumen de la empresa y las necesidades, puede optarse por una solución más económica u otra más completa y potente, aunque no sea gratuita.
- *SAP Business ByDesign.* Una de las herramientas más implantadas y sofisticadas del mercado. Actualmente, el producto ERP SAP S/4HANA Cloud se presenta como un ERP en la nube con todo tipo de funcionalidades y posibilidades como la conexión con redes de negocio, internet de las cosas o *big data*, entre otros. Las ventajas que ofrecen son similares a las encontradas en productos anteriores, destacando la escalabilidad y un sistema robusto de seguridad.
- *Tryton.* Al igual que Odoo, es un ERP *open source* con licencia GNU. Tryton es un ERP escrito en Python y PostgreSQL como base de datos, la cual ofrece una solución escalable, modular y segura.
- *WorkPLAN.* ERP que permite gestionar de forma centralizada los procesos de negocio de una empresa. Es un *software* propietario que está especializado en talleres, desarrollo de prototipos y fabricación a medida.

www

### Recursos web

A través de estos enlaces podrás acceder a las páginas web de Dolibarr, Odoo, SAP Business ByDesign, Tryton y WorkPLAN.



Otros ERP también muy implantados son: Abanq, Abas Business Software, Adempiere, Compiere, Delfos ERP, Eneboo, FacturaScripts, IDempiere, JFire, Microsoft Dynamics, Openbravo o WebERP.

#### Actividad propuesta 7.1



De los ERP citados anteriormente, investiga cuáles de ellos son los más implantados y realiza una comparativa entre los mismos. Para la comparativa, utiliza los tres que creas que son más importantes.

 PARA SABER MÁS

### ¿Por qué utilizar un ERP en la nube?

Una de las razones por las cuales utilizar una herramienta en la nube es la flexibilidad y escalabilidad que presentan la mayoría de productos contrastados. Es más, hay una gran variedad, con lo cual el usuario tiene la libertad de elegir el que más le conviene según el tipo de negocio.

Un ERP en la nube, generalmente, es más barato que un ERP físico en un servidor de la empresa, puesto que no hay que hacer una inversión inicial grande para implantarlo.

Además, el ROI (*return on investment*) se reduce dado que, al ser baja la inversión inicial, la amortización es más rápida.

La seguridad aumenta, aunque parezca lo contrario. Generalmente, los sistemas en la nube suelen implementar las últimas novedades en seguridad de forma inmediata, mientras que, en los sistemas implantados físicamente en la empresa, normalmente, la seguridad se relaja pasado un tiempo.

En los sistemas en la nube, la seguridad se aplica por capas, con lo cual es difícil presentar vulnerabilidades.

Otra ventaja fundamental son las actualizaciones y mejoras. En estos sistemas, prácticamente siempre está trabajándose con la última versión, con lo cual se mitigan rápidamente los posibles agujeros de seguridad o errores de software.

También se ven beneficiados los usuarios que acceden mediante dispositivos móviles, puesto que, al ser un sistema en la nube, este acceso es más transparente.

El presupuesto en hardware se reduce. Al solo requerir un navegador y residir la funcionalidad en un servidor externo, cualquier equipo con un navegador actualizado a las últimas versiones y un sistema operativo relativamente nuevo podría servir.

Es multiplataforma y pueden reducirse drásticamente los costes utilizando software libre. Como se ha dicho antes, el único requisito es tener un navegador actualizado.



### Actividades propuestas

Responde si las siguientes afirmaciones son verdaderas o falsas y razona tu respuesta:

**V F 7.2.** Con *data mining*, pueden encontrarse patrones dentro de grandes bases de datos o conjuntos de datos.

**V F 7.3.** Un *data warehouse* puede servir para llevar la gestión de una empresa de forma integral.

**Verificar**

#### 7.2.1. Integración de módulos

La funcionalidad de los módulos de un ERP es muy amplia, pero, en esta sección, intentará resumirse la de los principales módulos.

Como se ha explicado anteriormente, una de las ventajas de un ERP es su total integración de los módulos que lo componen en una gran base de datos. La ventaja de esto es su total compatibilidad y disponibilidad inmediata de la información para la empresa.

La información no se duplica ni se crean versiones independientes de esta, sino que residirá de forma centralizada en el sistema ERP.

A continuación, se describen algunos de los módulos más utilizados e imprescindibles de un ERP.

### A) Módulo de ventas

Una de las funcionalidades principales de este módulo es poder gestionar los inventarios de una manera eficiente. Muchas empresas utilizan sistemas JIT (*just in time*) para reducir inventarios, así que disponer de una herramienta de este tipo es fundamental.



#### Investiga

¿Qué es JIT (*just in time*)? ¿Dónde nació y cómo revolucionó los sistemas de inventario?

Además, el módulo tendrá que registrar los contratos de los clientes, órdenes, listados de precios, condiciones de pago, gestión de facturas, recepción de entregas, precios al público, direcciones de entrega, bonificaciones y descuentos, seguimiento y control, etc.

#### Actividades propuestas



Responde si las siguientes afirmaciones son verdaderas o falsas y razona tu respuesta:

- V F 7.4.** Con un sistema JIT, una empresa puede tener un inventario más reducido, ahorrando así espacio y coste de almacenamiento.
- V F 7.5.** El objetivo de un sistema *just in time* es fidelizar los clientes de una empresa para que compren sus productos o servicios.

Verificar



Muchos de estos módulos suelen tener asociado un *e-commerce* para poder realizar tanto venta *online* como venta física.

#### RECUERDA

Los posibles informes que se generen pueden descargarse con formato Microsoft Office (MS Excel o MS Word), OpenOffice o HTML.

Los informes deberán ser flexibles y permitir ventas independientes del periodo o ventas desagregadas.

## B) Módulo de compras

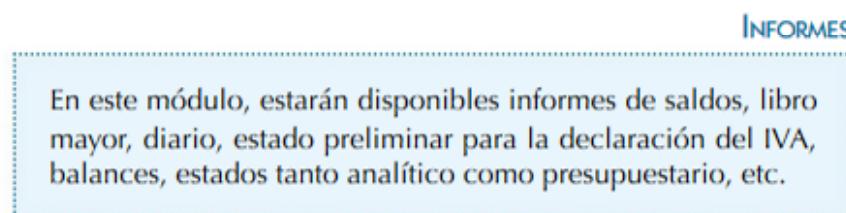
Entre las funcionalidades básicas que se exige a un módulo de compras está la del seguimiento de los pedidos, el control de entrega al almacén (o en su caso al consumidor final, dado que hay negocios que no tienen almacén físico), reagrupar órdenes de compra, control de facturas, gestión de inventarios, listados de precios, contratos con proveedores, control de precios límite (no comprar productos por encima de un precio tasado), planificación de entregas, soporte para código de barras de 12 dígitos, clasificación jerárquica de proveedores, etc.

## C) Módulo de gestión financiera y contabilidad

Otro módulo imprescindible en un ERP es el de gestión financiera y contabilidad. Es básico que este módulo siga el plan contable general y puedan controlarse las cuentas pendientes de cobro, así como las pendientes de pago.

Los contables, en ocasiones, suelen trabajar en varios ejercicios al mismo tiempo. Sería de gran ayuda para el trabajador que el módulo pudiese trabajar de forma simultánea en dos ejercicios contables para poder hacer comparativas.

Además, podrán hacerse cierres parciales, planes multinivel, conciliación, controles y generar todo tipo de informes.



## D) Módulo CRM

Como ya se ha dicho anteriormente, el objetivo de un CRM es fidelizar a los clientes haciendo que las campañas de *marketing* sean más eficientes y menos costosas.

El módulo de CRM deberá crear las oportunidades comerciales con los posibles clientes y ofrecer al usuario una manera clara de hacer un seguimiento de estas. Deberá poder gestionar zonas de venta y los presupuestos.

La estrategia enfocada al cliente de un CRM se focaliza en conseguir la mayor información sobre este creando una relación a largo plazo. Hoy en día, es prácticamente imposible encontrar una empresa de éxito sin que tenga en cuenta la orientación al cliente y la gestión comercial.



**INFORMES**

Deberán poder generarse informes por cuenta, zona, comercial, línea de productos, etc.

### E) Módulo de recursos humanos

Cuando una empresa es mediana o grande, se hace imprescindible un *software* especializado en la gestión de RR. HH. Este módulo del ERP podrá hacer una gestión de las vacaciones, las enfermedades, las peticiones de los empleados, las licencias, la asistencia y los horarios de los empleados.

Muchas veces, la asistencia u horas realizadas no tiene como finalidad controlar al cliente, sino controlar el gasto que conlleva que un empleado trabaje un determinado tiempo en un proyecto.

También puede gestionarse con este *software* el gasto de los empleados para que, posteriormente, pueda ser reembolsado por la empresa. Es más, muchas veces, estos gastos podrán o deberán ser repercutidos al cliente.

Además de las tareas anteriores, el ERP deberá gestionar el proceso de contratación y evaluación de candidatos. De esa forma, puede gestionarse de una manera más eficiente y transparente dicho proceso.

**INFORMES**

El módulo de RR. HH. podrá hacer informes sobre el seguimiento de los empleados, control de asistencia, gestión del rendimiento del personal, gastos y costes del departamento, etc.

### F) Módulo de gestión de almacenes

Una buena gestión del almacén es fundamental en cualquier empresa, puesto que el dinero que puede ahorrarse y los beneficios que aporta son claves.

El módulo de almacén de un ERP deberá no solamente gestionar el inventario o la trazabilidad de productos, sino que también deberá establecer una valoración de estos. Muchos de estos módulos utilizan una gestión de inventarios por partida doble como en el *software* de contabilidad.

**INFORMES**

El módulo de almacén podrá hacer informes sobre el inventario y el costo de los productos almacenados, ya sea por su coste de producción o cualquier coste que se crea oportuno.

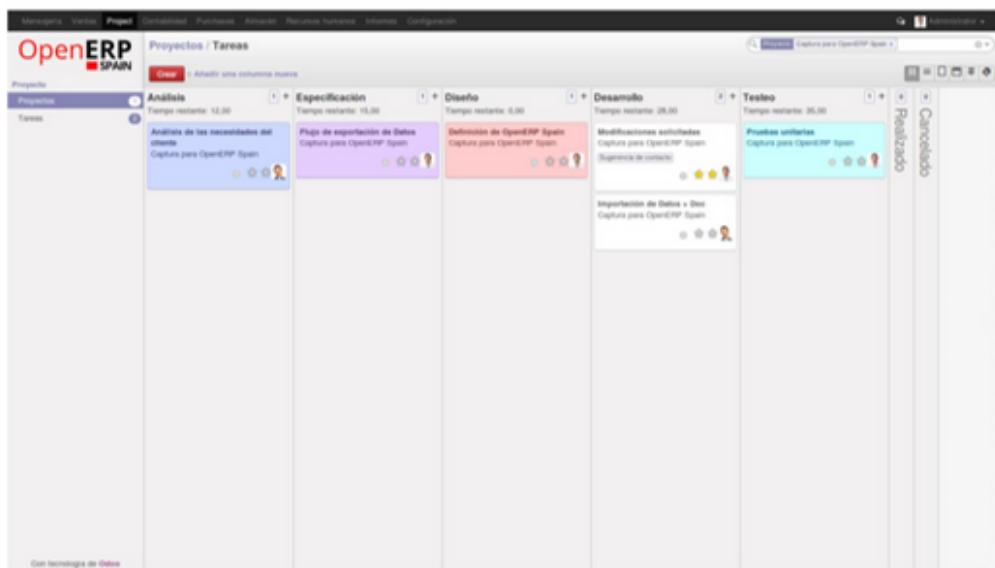
## G) Módulo de proyectos

Los proyectos son algo básico en una empresa que tiene que seguirse con una herramienta corporativa. Es cierto que existen muchas herramientas buenas para hacer un seguimiento como Microsoft Project o Trello, por citar una herramienta de escritorio propietaria y otra *open source*. No obstante, el módulo de proyectos es uno de los más demandados al implantar un ERP, dado que, muchas veces, toda la información de la que se vertebra el proyecto reside en el propio ERP.

Además, estas herramientas, al ser corporativas, tienen servicio de mensajería, con lo cual puede centralizarse la comunicación y guardarla para analizarla posteriormente.

Este tipo de módulos permitirá realizar la planificación de las fases de cualquier proyecto y las tareas de este asignando recursos y mostrándolo visualmente mediante una vista Gantt o un diagrama parecido.

Es importante que los proyectos puedan tener sus hojas de asistencia de los empleados y que dichos empleados puedan registrar las horas trabajadas a cada proyecto, puesto que, a la hora de elaborar costes, es muy importante.



**Figura 7.4**  
Módulo de proyectos en OpenERP, tecnología Odoo.

## H) Módulo de marketing

La automatización del *marketing* puede ser muy ventajosa para cualquier empresa. Muchas de las campañas pueden dejar de hacerse por falta de herramientas o recursos. Con un buen *software* de *marketing*, puede captarse un cliente, enviarle productos, realizar seguimiento de revisión, conocer el interés de los clientes, saber quién es más activo o quién no responde.

Puede llegar a tenerse una retroalimentación que puede hacer que la empresa tome las decisiones oportunas al respecto. Permite medir la efectividad de una campaña. Este hecho es difícil de controlar con estrategias de *marketing* obsoletas.

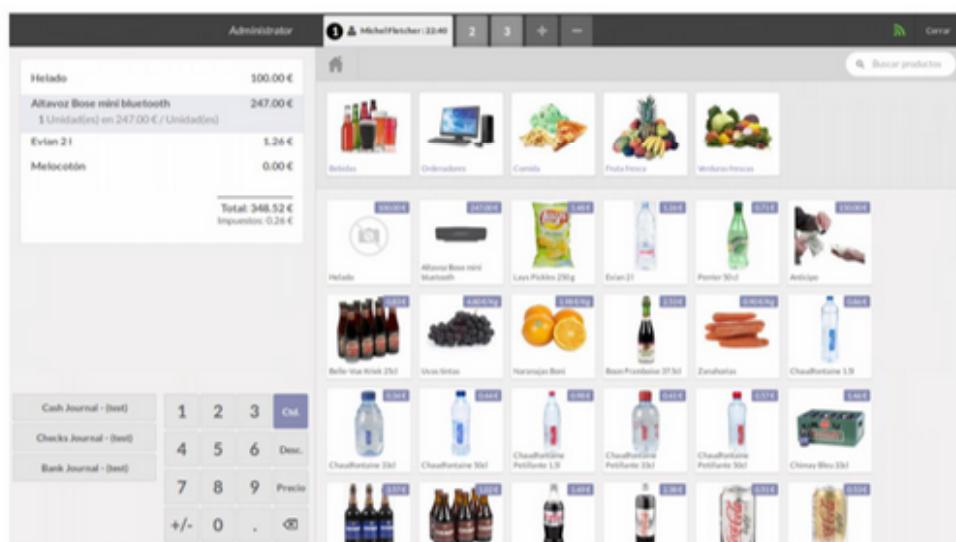
Además, este *software* permite la segmentación de los clientes para adecuar las campañas con las actividades de la empresa. Las campañas tienen que estar sincronizadas con la segmentación realizada en el sistema.

### RECUERDA

- ✓ El módulo de *marketing* estará fuertemente ligado al módulo de CRM.

## I) Módulo TPV

Un módulo TPV para algunos negocios es una parte fundamental. Si, además, está conectado con todo el sistema corporativo de la empresa, mejor que mejor.



**Figura 7.5**  
Módulo de TPV en OpenERP.

En ocasiones, es necesario realizar una venta desde un terminal, una tableta, un *smartphone*, etc. Para que sea un ERP potente y flexible, bastará con tener un navegador conectado al servidor ERP de la empresa y estará disponible un punto de venta itinerante en cualquier punto del planeta.

Será necesario poder buscar los productos por código de barras, descripción, categoría, etc.

Es preciso también que dichos módulos estén conectados con los módulos de ventas, almacén, facturación, etc. De esa manera, puede tenerse en tiempo real el inventario y es posible realizar hasta el asiento contable de la operación. También podrán generarse facturas o tiques al instante.

Del mismo modo, el departamento de contabilidad puede tener al instante las entradas de dinero y gestionar esa información en tiempo real.

Este módulo es básico cuando el ERP se implanta en una empresa con una o varias tiendas.

Importante también poder personalizar o adaptar este módulo, puesto que no todas las tiendas son iguales.

### 7.2.2. Implementación, adaptación y configuración

Uno de los inconvenientes o reticencias de una empresa que ha de implantar un ERP es que son difíciles de configurar y utilizar. Muchos empleados de una empresa están habituados a utilizar herramientas más sencillas y se resisten al cambio.

Además, los consultores o personal que implanta un ERP en una empresa son profesionales con altos sueldos, lo que hace que tanto su contratación directa como por medio de una empresa especializada sea costoso.

En ocasiones, al hablar con gestores que han implantado un ERP costoso como SAP, al final de la integración, están contentos. Sin embargo, algunos de los inconvenientes que plantean es que este tipo de productos hace que se tengan menos empleados, pero más especializados con lo cual el nivel salarial de los mismos es más alto.

No obstante, actualmente existen productos *open source*, como puede ser Odoo, que hacen que el precio total de la implantación baje y cualquier problema o nueva funcionalidad sea más fácil de conseguir al tener detrás una comunidad muy grande de gente utilizando el producto.

#### RECUERDA

- ✓ Para tener siempre la instalación actualizada en un sistema Debian, hay que ejecutar `apt-get update`.

#### Ejercicio propuesto 7.1

Instala Odoo versión *community* en tu sistema y comprueba que funciona.



#### Recurso web

¿Quieres ver la aplicación Odoo sin tener que hacer ninguna instalación? Visita su página web.



Como puede observarse, el proceso más largo es la adaptación o configuración de la herramienta para un cliente concreto. Este proceso deberá realizarse por un informático que conozca el ERP en profundidad.

#### Instalación de Odoo



#### Configuración de Odoo



### 7.2.3. Importación y exportación de información: generación de informes

La información de una empresa que utiliza un ERP está dentro del sistema, como no podría ser de otra forma. Al contrario que otras herramientas especializadas no tan completas, el intercambio de información y los cambios de formato se reducen considerablemente.

Muchas veces, la exportación de la información en formato informe puede ser en formato PDF o en formato Microsoft Office. Este último permite la personalización, con lo cual suele ser bastante utilizado.

También se reducen los problemas o errores debido a esa transformación de formato.

Generalmente, de un ERP, puede exportarse o importarse información mediante los siguientes sistemas:

- *Ficheros CSV.* El formato es *comma-separated values* y lleva utilizándose desde hace décadas. Este formato permite separar la información, ya sean números o texto, de forma tabular en texto plano. Suele utilizarse una coma como separador de campos, aunque, en ocasiones, los módulos de exportación o importación permiten utilizar otro tipo separador. Si se utiliza otro carácter como separador en la exportación, en la importación, habrá que tener en cuenta este hecho.

En ocasiones, se utiliza el tabulador que, aunque no es un carácter visible, hace que la importación sea más sencilla a la hora de *parsear* el archivo origen.

En este formato, cada registro ocupa una línea del fichero y habrá que tener un formato de texto plano (normalmente, ASCII o Unicode como UTF-8 o EBCDIC).

Es imprescindible que todos los registros tengan el mismo número de campos y en la misma posición.

- *Ficheros XML.* Este formato tiene numerosas ventajas, dado que es legible tanto por las máquinas como por el programador o usuario.

Con este formato, una de las ventajas que se obtiene es poder importar de una forma limpia los datos al tener estrictas reglas sintácticas de la composición de un documento.

Otra de las ventajas es que no hace falta que el documento tenga forma tabular y puede ser tratado por cualquier sistema, lenguaje y alfabeto.

- *Documentos ofimáticos.* Generalmente, el formato ofimático por excelencia es Microsoft Office en sus distintas versiones, aunque algunos también permiten la generación de informes o datos en PDF (*portable document format*). Hay que decir que este último es solamente de exportación y, generalmente, se importan los datos en tablas de Excel.

Una de las ventajas de este formato es su posterior edición o modificación por parte del usuario (no en el caso del formato PDF, pues, aunque podría editarse con un programa especializado, su objetivo es ser un documento de solo lectura).

- *Programas de conexión a la base de datos del sistema.* Estos programas son más complejos y se utilizan por personal con más conocimiento del ERP (programadores y demás). Generalmente, estos programas están realizados en un lenguaje de programación como Java, Python, C++, etc., y acceden directamente a los campos necesarios de la base de datos.

Entre las ventajas que da este tipo de acceso, destaca la flexibilidad de poder modificar y personalizar cualquier tipo de intercambio de información. Además, es más eficiente realizar la importación o exportación con este sistema cuando el volumen de información con el que está trabajándose es grande.

En las fases de migración, cuando los usuarios están trasladando la información del sistema antiguo al nuevo, lo más normal es utilizar un programa de este tipo porque se acortan mucho los tiempos de traspaso de información, puesto que los volúmenes que se manejan son muy altos.

### 7.3. Minería de datos

No mucha gente conoce el término *data mining* (aunque es perfectamente conocido en círculos informáticos y empresariales), pero la verdad es que es algo inherente a sus vidas.

Cada vez que se realiza una compra en una tienda o gran almacén y se presenta la tarjeta de fidelización (esa con la que hacen descuentos si se compran ciertos productos u obsequian con un porcentaje de las compras) está alimentándose el sistema de *data mining* de la empresa. Lo que se genera son datos asociados a la identidad del cliente de tal forma que esos datos se almacenan en grandes servidores.



SABÍAS QUE...

El término *minería de datos* se ha utilizado en estadística para referirse al uso de datos que derivaban en conclusiones erróneas.

Cada vez que se hace una compra, los datos y los productos comprados, la hora, el día, el lugar, etc., quedan registradas (patrones de compra) y, por lo tanto, los patrones de compra o hábitos de consumo.

Esos datos sirven a los grandes almacenes para interpretar patrones y ayudar a sus responsables a servir mejor a sus clientes y a tomar decisiones con mejor información. Imagínese que se analizan los pedidos a domicilio de un gran almacén y se observa que los viernes se generan muchos más pedidos que el resto de la semana. Una decisión que podría tomarse es reforzar el turno del viernes con más efectivos y ofrecer así un mejor servicio al cliente.

Al mismo tiempo, esa información puede servir a estos grandes almacenes para vender más o vender al cliente productos más caros o que tengan más margen para el comercio. La información siempre es un arma de doble filo y este es el punto en el que muchos de los críticos de estos sistemas hacen hincapié.

En muchos países (sobre todo en los Estados Unidos), existen organizaciones dedicadas a investigar empresas que almacenan grandes cantidades de datos, algunos de ellos sensibles y el uso que estas empresas les dan.

Por lo tanto, podría definirse *data mining* como la disciplina que intenta encontrar patrones (no conocidos) útiles y válidos en grandes bases de datos o conjuntos de datos.

El objetivo es encontrar relaciones entre los datos no sospechadas y presentar los datos de una manera que sean entendibles y útiles a los usuarios. En principio, los datos en bruto no aportan información, las técnicas antiguas se han visto que no funcionan cuando se procesan grandes volúmenes de datos. Estas nuevas técnicas de *data mining* aplican análisis de datos, junto con algoritmos de búsqueda de patrones, y son útiles a la hora de procesar estadísticas, aprendizaje automático, visualización de datos, etc.

### FUNDAMENTAL

#### *¿Por qué esta necesidad de minería de datos?*

Por muchas razones. La presión competitiva de las grandes empresas empuja a mejorar los sistemas, adaptarse a las necesidades de los clientes, adelantarse a sus competidores, etc. La información es un recurso que hay que gestionar de la mejor manera posible y esto requiere procesarla de una manera eficiente y efectiva.

Imagínese los millones de registros que pueden generarse en una cadena de grandes almacenes, una cadena hotelera, de automoción, etc. Si esta cantidad de información no se trata y se gestiona de forma eficiente, ya llegará otro competidor que lo haga y te supere.

Actualmente, este tipo de técnicas se utilizan mucho en el comercio electrónico. Hace años, eran técnicas de laboratorio o experimentales, pero, hoy en día, todas las empresas líderes en sus sectores las implementan en mayor o menor medida.

Hay que tener en cuenta que este tipo de técnicas no sustituyen a las personas, sino que le proporcionan indicios, patrones y relaciones en los datos que hacen que los gestores entiendan mejor sus negocios y puedan tomar decisiones más acertadas y fundamentadas.

#### 7.3.1. El proceso de descubrimiento de conocimiento y la minería de datos

En este apartado, se estudiará el proceso de descubrimiento de información y en qué parte actúan las técnicas de minería de datos.

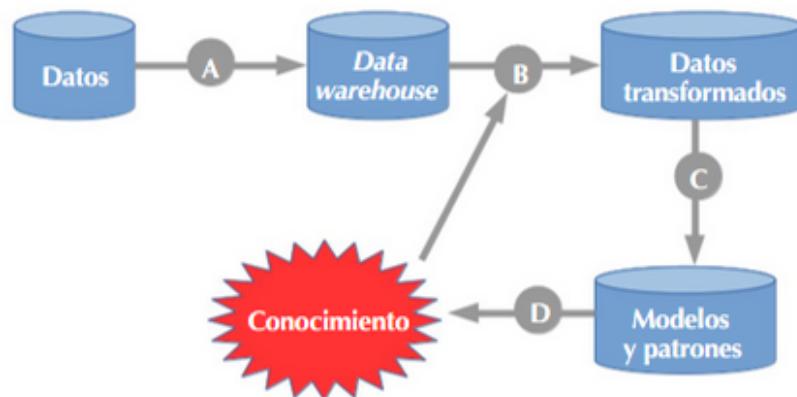
- *Datos.* Corresponden a los datos en bruto de las bases de datos operacionales.
- *Paso A: limpieza, clasificación y agrupación de los datos.* Los datos en bruto son procesados para incorporarlos al *data warehouse*, un almacén de datos especializado en tratamiento de datos con fines de *data mining*. La limpieza de datos implica su filtrado, eliminación de datos inconsistentes o datos inconexos, etc.
- *Paso B: preparación de datos.* Generalmente, se transforman los datos y se reduce su dimensionalidad para poder tratarlos de una manera más eficiente. Además, se seleccionan hipótesis por verificar o el modelo adecuado de análisis. La preparación de datos normalmente se lleva la mayoría del tiempo y esfuerzo a la hora de realizar este proceso.

- Paso C: data mining.**

En este proceso, se aplican varias técnicas y algoritmos (*clustering*, regresión, clasificación, detección de desviaciones, etc.) a los datos ya transformados. A la hora de realizar *data mining*, existen los métodos descriptivos, en los que lo que se busca es generar patrones

interpretables por el usuario que describen los datos en sí mismos, o métodos predictivos, en los que se busca el predecir sucesos o hechos que ocurrirán o podrían ocurrir en un futuro.

- Modelos y patrones.** Hay que hacer distinciones entre los modelos y los patrones. En un símil, si se utiliza un molde para hacer bombones, el molde sería el modelo y los bombones fabricados serían los patrones.
- Paso D: evaluación de los patrones y su verificación.** Hay que evaluar si los resultados obtenidos son correctos y pueden consolidarse y utilizarse en el futuro.



**Figura 7.7**  
Proceso de descubrimiento de conocimiento.



### Actividades propuestas

Responde si las siguientes afirmaciones son verdaderas o falsas y razona tu respuesta:

**V F 7.6.** El término *minería de datos* se ha utilizado en contabilidad para referirse al uso de datos que derivaban en conclusiones veraces.

**V F 7.7.** Podría definirse *ERP* como la disciplina que intenta encontrar patrones (no conocidos) útiles y válidos en grandes bases de datos o conjuntos de datos.

**Verificar**

## 7.4. OLAP

OLAP es el acrónimo de *online analytical processing*, que, traducido al español, sería “procesamiento en línea analítico” o, lo que es lo mismo, una manera de extraer de forma selectiva datos y visualizarlos y analizarlos desde distintos puntos de vista.

Las bases de datos relacionales permiten un cierto tipo de consultas, pero hay otras que resultan extremadamente complicadas de realizar en este tipo de bases de datos. Un ejemplo de ello puede ser mostrar el número de productos vendidos en el mes de agosto por una cadena de perfumerías en la zona de Levante y compararlos con los mismos resultados del año anterior.

El problema de esta consulta radica en que las bases de datos tradicionales son bidimensionales (filas y columnas, las filas tienen los datos y las columnas los campos), mientras que las bases de datos OLAP son multidimensionales. Por lo tanto, conocer datos con varias dimensiones (zona de Levante, productos vendidos, mes de agosto, etc.) son operaciones sencillas para una base de datos OLAP.

Como se ha mencionado, en las bases de datos OLAP, los datos se almacenan de forma multidimensional. Cada atributo puede ser una dimensión diferente (productos, zona geográfica, periodo de tiempo, etc.). En el ejemplo anterior, la solución sería establecer un corte en las diferentes dimensiones que se han citado y, de esa manera, se obtendrá la información.

OLAP puede utilizarse para realizar *data mining* y así poder descubrir patrones y relaciones entre datos.

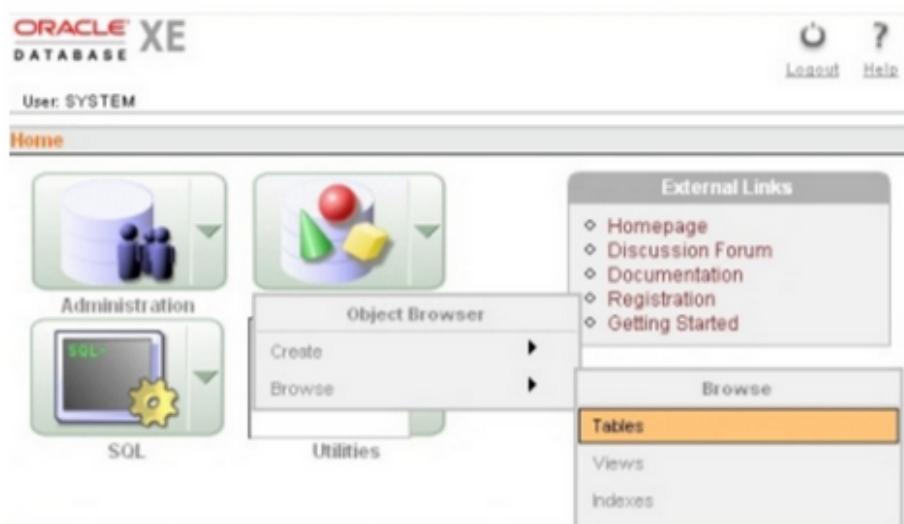


#### SABÍAS QUE...

Cuando empezó a trabajarse en este tipo de tecnologías (OLAP y *data mining*), algunos grandes almacenes descubrieron que, en sus tiendas, había una relación entre pañales de bebé y cerveza. Existía un cierto número de compras en las que ambos productos eran adquiridos por el mismo cliente. Una vez descubierto este hecho, dichos almacenes colocaron la cerveza y los pañales de bebé en zonas cercanas para que el cliente cogiera ambos productos y así maximizar las ventas.

Generalmente, los técnicos que utilizan la tecnología OLAP alimentan dicha base de datos OLAP con las bases de datos transaccionales que residen en servidores relacionales. Esta importación puede hacerse mediante ODBC (Open DataBase Connectivity) u otro sistema similar.

Dos de los productos OLAP más utilizados son Oracle Express Server e Hyperion Solution Essbase. Generalmente, este tipo de sistemas propietarios establecen el coste del producto dependiendo del número de personas que van a hacer uso de él.



**Figura 7.8**  
Interfaz web de Oracle Express.

## 7.5. Dashboard y scorecard: business intelligence e informes

Los *dashboards* y *scorecards*, más conocidos como *cuadros de mando*, son herramientas analíticas que están utilizándose con más asiduidad. Las empresas invierten cada vez más en *business intelligence* o inteligencia de negocio y este tipo de herramientas son muy utilizadas en esos ministerios. Los gestores no solamente necesitan números o informes, sino que requieren algo más. Necesitan saber el porqué y el cómo de esos datos, por lo tanto, estas herramientas los ayudan en su tarea de gestión.

Para mucha gente, un *dashboard* o un *scorecard* es lo mismo, pero realmente presentan algunas diferencias entre ellos.

Un *dashboard* o cuadro de mandos es algo parecido al cuadro de mandos que pueda tener un coche, un avión u otro aparato. Es una herramienta que muestra una serie de métricas que deben ser interpretadas por el gestor.

¿Qué es *business intelligence* o inteligencia de negocio? Los analistas de cualquier negocio necesitan tener referencias o instantáneas de un negocio para ver tendencias, datos agregados, ver las variables que hacen que se rompan esas tendencias, etc.

La inteligencia de negocio es ese proceso. Se extraen los datos de una base de datos OLAP para después analizarlos y así poder tomar decisiones apoyadas por información contrastada.

Esta técnica necesita una evolución de los típicos informes preestablecidos que, hoy en día, quedan obsoletos en muchas organizaciones a algo más potente, elaborado, personalizable y en tiempo real en ocasiones.

### 7.5.1. La evolución de los informes

Como puede observarse en este apartado, los informes han pasado de ser un objeto físico y estático como un papel o papeles a algo dinámico, en tiempo real y *online*.

La potencia de los informes se multiplica, puesto que, con los cuadros de mando, el gerente puede tener una visión global de la empresa o las áreas que le interesen.

Es más, este tipo de herramientas analíticas, además de ofrecer los datos, permiten profundizar en ellos y obtener ayuda a la hora de la gestión de cualquier gerente, por lo tanto, son herramientas muy codiciadas por el alto mando de una empresa.

Es importante a la hora de implementar un *scorecard* o cuadro de mando la correcta elección de indicadores. Si esto no se hace de manera correcta, el sistema no será de utilidad.

Este tipo de informes o cuadros de mando suelen centrarse en aspectos como estrategias, comportamiento o satisfacción de los clientes, distribución de recursos, nivel de cumplimiento de objetivos, gastos, costos, productividad, etc.

## 7.6. Seguridad en sistemas de gestión empresarial

Prácticamente, la mayoría de los ERP más utilizados, si están actualizados, tienen un nivel de seguridad más que aceptable. No obstante, hay que tener en cuenta que un ERP puede estar instalado en un servidor de una organización y, entonces, los aspectos de seguridad no solamente se circunscriben al *software* en sí, sino que tienen que ver también los sistemas operativos, la red, la seguridad física, etc.

Entre los problemas que puede encontrarse un administrador de sistemas con respecto a la seguridad de un ERP, pueden estar los accesos a informes o realización de copias no autorizadas, manipulación de datos de la compañía, etc.

Aunque pueden existir acciones premeditadas para boicotear un *software* o una organización concreta, muchas veces, son errores humanos por accidente o desconocimiento los que provocan situaciones no esperadas. En otras ocasiones, el error es de procedimiento. El usuario no sigue el patrón adecuado y los datos registrados son incorrectos. Es más, podría darse el caso de que el error se debiera al mismo *software*. El administrador o responsable del ERP es la persona responsable de encauzar, minimizar o evitar que ocurran dichas desgracias.

Generalmente, los ERP tienen alguna herramienta de auditoría o control de datos que permiten saber los cambios que ha sufrido un registro. Como mínimo, deberían registrarse en un log los siguientes datos:

- Usuario.
- Acción realizada.
- Fecha y hora.
- Dirección IP (si es posible identificar el dispositivo u ordenador mejor).
- Tabla campo y dato accedido.
- Valor anterior y actual del dato.

Entre las herramientas de las que dispone un ERP para implantar un nivel de seguridad aceptable, están las siguientes:

- *Niveles de acceso.* Generalmente, se crea una seguridad por capas o por niveles y los usuarios con menor jerarquía solamente acceden a unos datos y los que tienen mayor jerarquía pueden acceder a más.
- *Profiles, roles y grupos.* Estas herramientas permiten hacer la seguridad más flexible. Además de la seguridad con niveles de acceso, puede mejorarse añadiendo *profiles, roles* y grupos. De esta manera, al administrador le es más fácil y eficiente gestionar el acceso a la información. Entre otras cosas, podría ocultar o manipular botones dentro de un formulario dependiendo del rol o perfil, realizar restricciones sobre las consultas, etc.
- *Contraseñas de usuario y cifrado.* Se generarán contraseñas por usuario con un nivel de cifrado tal que no puedan ser exportadas ni seleccionadas. También podrán incorporarse bloqueos del sistema tras inactividad, petición de la contraseña para ciertas acciones, bloqueo de una cuenta de usuario si se intenta acceder X veces con una contraseña incorrecta, etc.
- *Personalización de menús.* Esta opción ofrece mucha seguridad para los administradores y para el propio sistema. Un usuario solamente podrá acceder a la información que sea precisa para hacer su trabajo. De esta manera, se evitan fallos no intencionados.

## Resumen

- Actualmente, las empresas han pasado de trabajar con aplicaciones o programas aislados con más o menos compatibilidad unos con otros a utilizar ERP, donde la información de la organización está totalmente integrada y puesta a disposición de los miembros de la misma.
- Todos estos sistemas y técnicas (CRM, ERP, *data mining, dashboard, scorecard, business intelligence*, etc.) se utilizan por la ventaja estratégica que suponen.

- El objetivo de un sistema de gestión empresarial es el desarrollo de la compañía a todos los niveles mejorando no solo la cadena de producción, sino también la toma de decisiones.
- CRM es la sigla de *customer relationship management*. Es un módulo de un sistema de gestión empresarial cuyo objetivo es la gestión de relaciones con los clientes.
- Los ERP son sistemas de gestión empresarial que cubren las necesidades de prácticamente todas las áreas de una empresa (finanzas, contabilidad, ventas, compras, recursos humanos, almacén, fabricación, etc.).
- Prácticamente, la totalidad de los ERP más usados tienen un módulo de CRM.
- Los módulos más utilizados e imprescindibles de un ERP son:
  - Ventas.
  - Compras.
  - Gestión financiera y contabilidad.
  - CRM.
  - Recursos humanos.
  - Gestión de almacenes.
  - Proyectos.
  - Marketing.
  - TPV.
- Los consultores o personal que implanta un ERP en una empresa son profesionales con altos sueldos, lo que hace que tanto su contratación directa como por medio de una empresa especializada sea costoso.
- En un ERP, puede exportarse o importarse información mediante los siguientes sistemas:
  - Ficheros CSV.
  - Ficheros XML.
  - Documentos ofimáticos.
  - Programas de conexión a la base de datos del sistema.
- *Data mining* es una disciplina que intenta encontrar patrones dentro de grandes bases de datos o set de datos. Su objetivo es encontrar relaciones entre los datos no sospechadas y presentar los datos de una manera que sean entendibles y útiles a los usuarios.
- OLAP es el acrónimo de *online analytical processing*, que, traducido al español, sería “procesamiento en línea analítico” o, lo que es lo mismo, una manera de extraer de forma selectiva datos y visualizarlos y analizarlos desde distintos puntos de vista.
- Los *dashboards* y *scorecards*, más conocidos como *cuadros de mando*, son herramientas analíticas que están utilizándose con más asiduidad en disciplinas como *business intelligence* o inteligencia de negocio.
- La inteligencia de negocio es un proceso en el cual se extraen los datos de una base de datos OLAP para después analizarlos y así poder tomar decisiones apoyadas por información contrastada.
- Un ERP actualizado suele tener un nivel de seguridad aceptable. Además, habrá que cuidar otros aspectos de seguridad como los accesos, sistemas operativos, etcétera.



## Ejercicios prácticos

- Realiza un pequeño informe sobre cómo un supermercado o cadena de supermercados puede sacar provecho de la *data mining* y aporta ideas que la cadena de supermercados puede implementar o algunas acciones que ya están utilizando.
- PrestaShop es una de las tiendas *online* más implementadas por su potencia, bajo coste y multitud de *plugins* que hacen que se multipliquen sus posibilidades.

Investiga si existe alguna solución informática (ERP) que permita conectar la tienda física a la tienda *online* creada con PrestaShop. El objetivo es que puedan sincronizarse los productos, los artículos, pedidos, fabricantes y proveedores.

Además, el ERP debería ofrecer las ventajas de estecomo generar modelos de IVA, contabilidad, etc.

- Quartup ERP Cloud (<http://www.quartup.com/>) es un ERP en la nube cuya ventaja es que funcionan a un precio reducido y son totalmente escalables. Se paga por lo que se utiliza. Investiga un producto *cloud* parecido y crea una tabla comparativa como la siguiente:

	Quartup	ERP cloud elegido
Licencias y precio.		
Módulos disponibles.		
Integrado con la tienda física y <i>online</i> .		
Personalización de informes.		
CRM.		
Multitienda.		
Multimoneda.		
Importación de información y sincronización en tiempo real.		

- Enumera las ventajas e inconvenientes que pueda tener un ERP en la nube frente a un ERP *offline* instalado en la sede de una empresa.
- Una vez claras cuáles son las ventajas e inconvenientes de los ERP *online* y *offline*, elabora un informe para una pequeña empresa local justificando las razones de usar uno u otro. En el informe, se añadirán datos como el coste de implantación y el tiempo que llevaría la migración del sistema actual al ERP.

