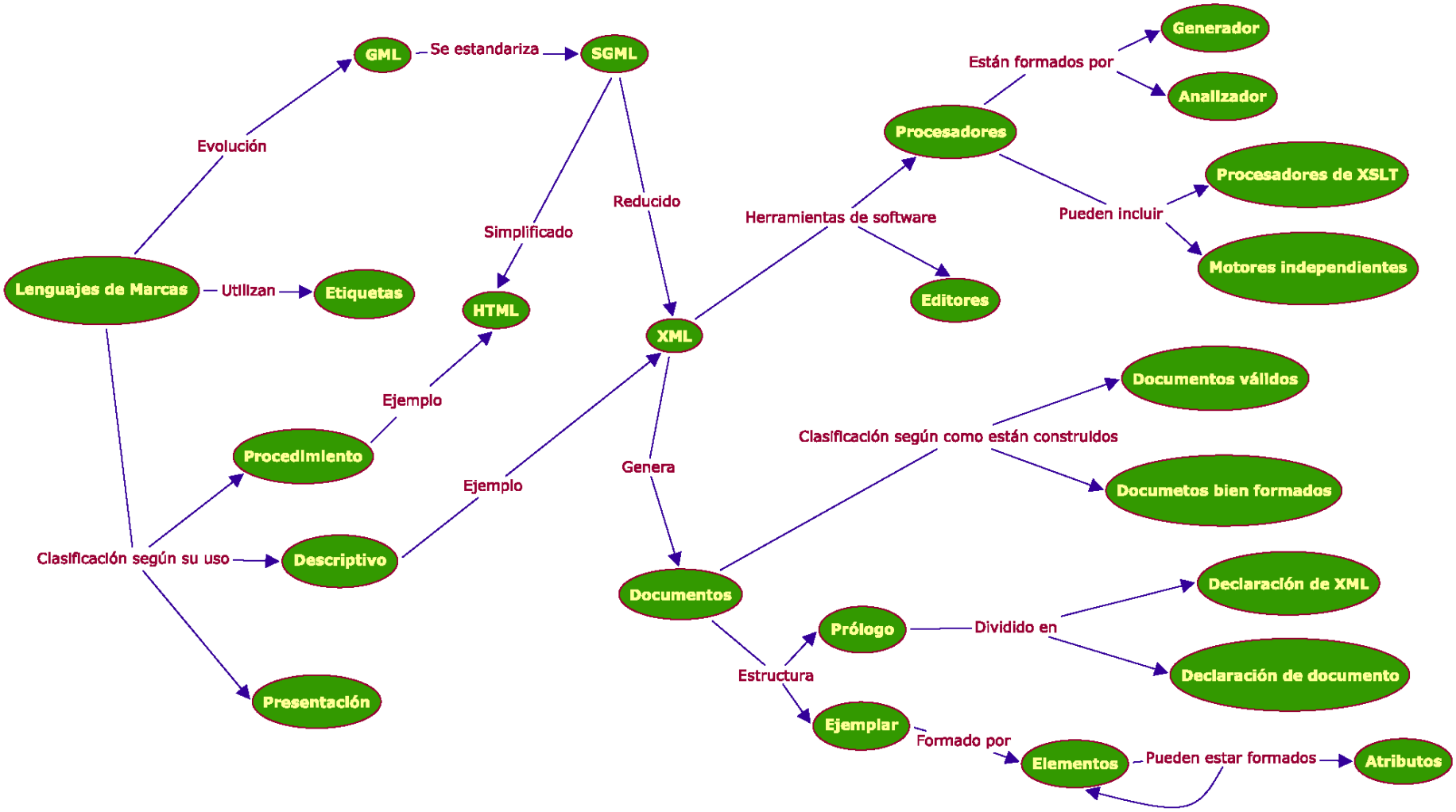
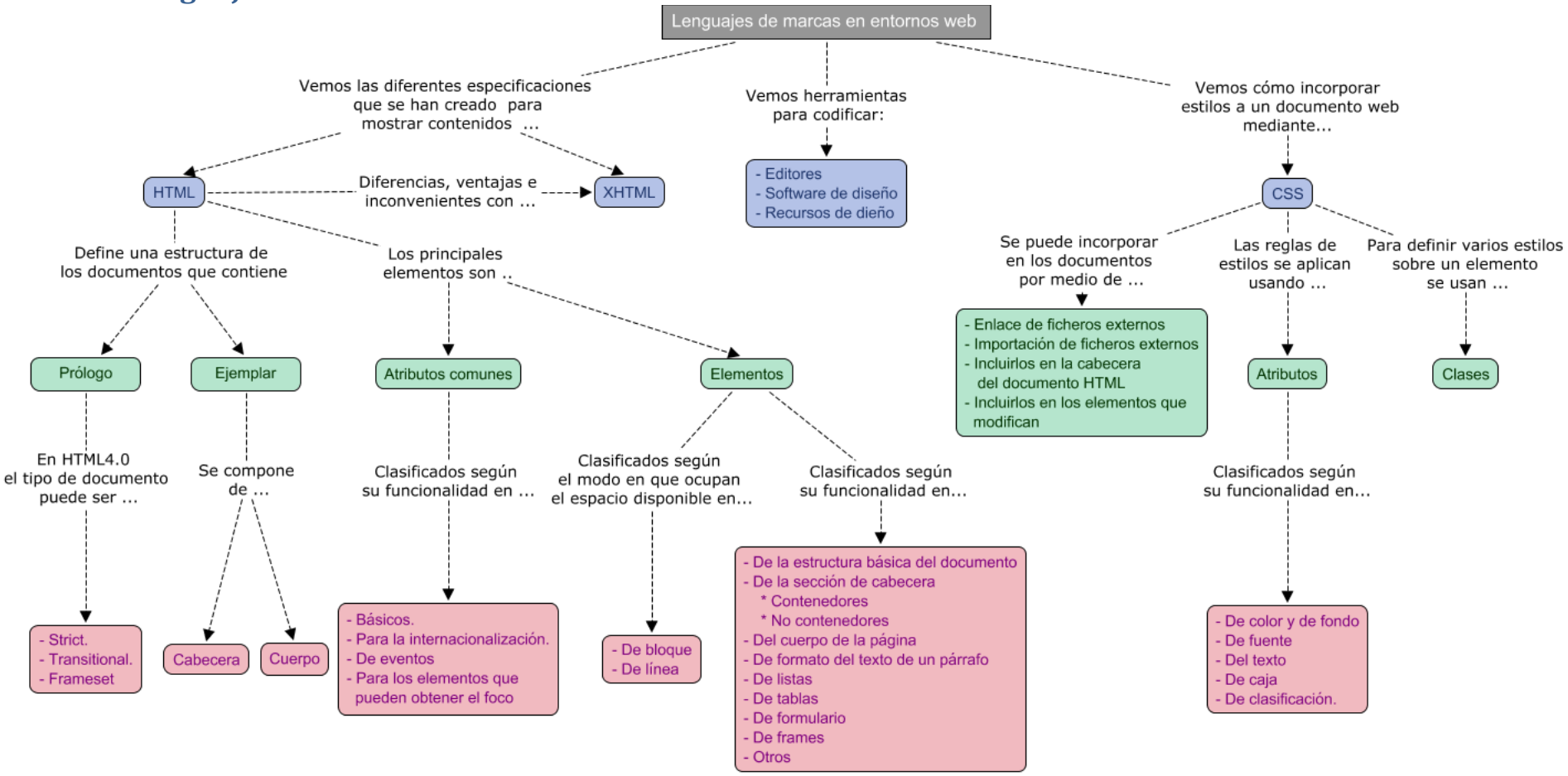
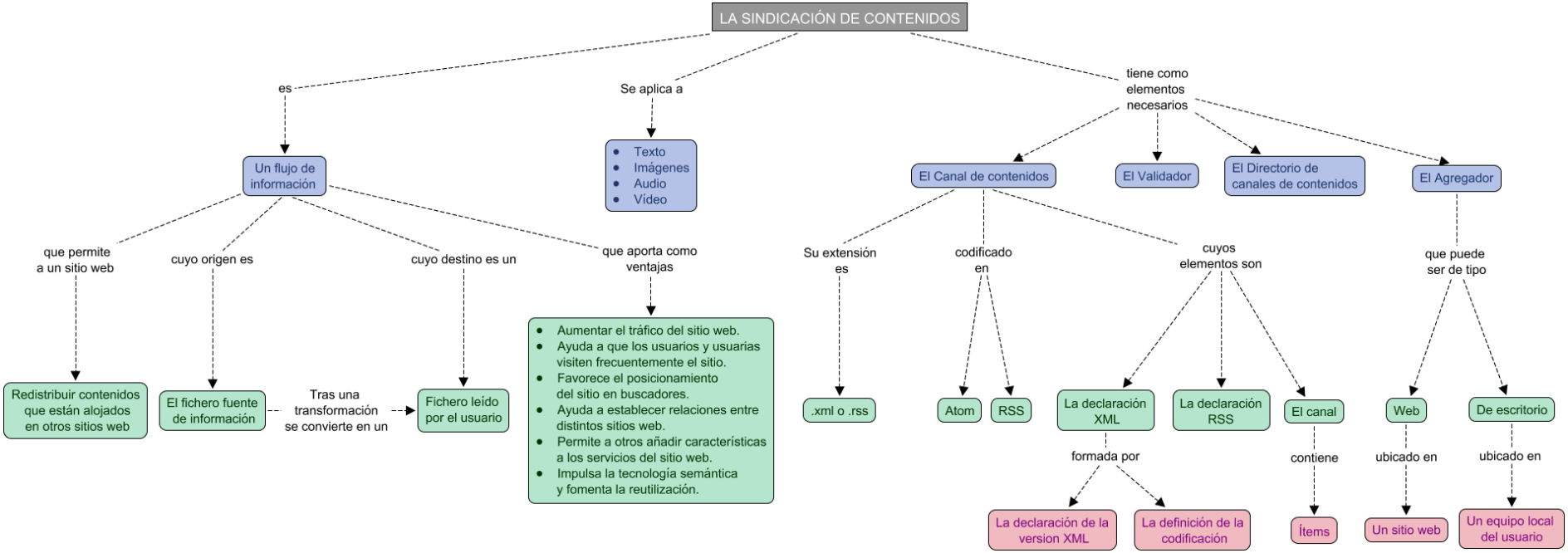
Tema 1 - Lenguajes de Marcas



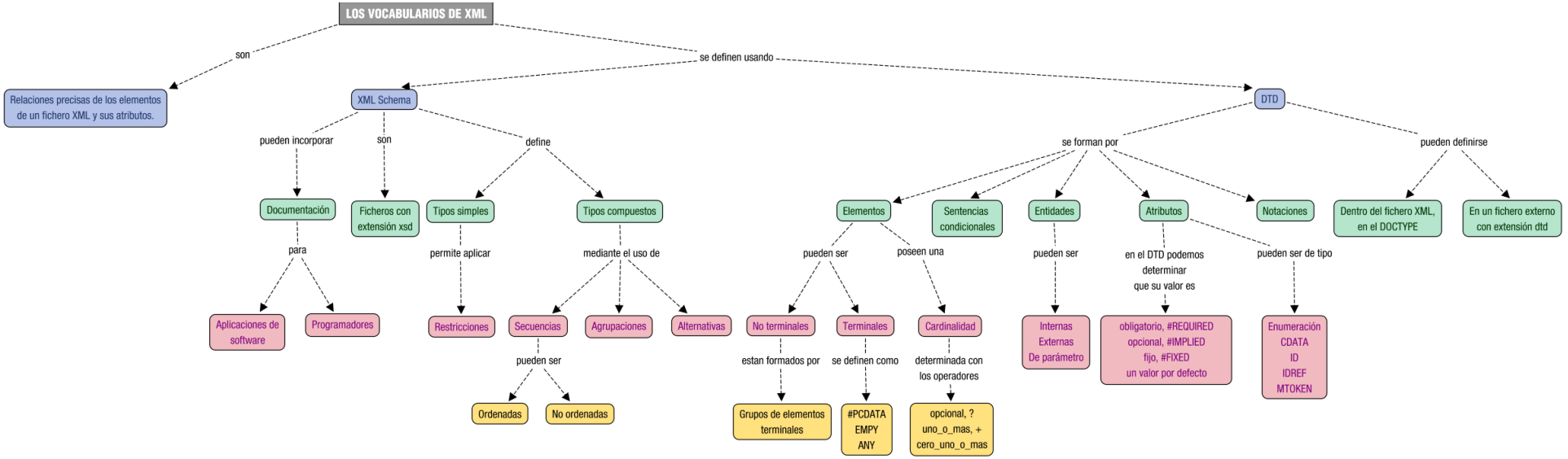
Tema 2 - Lenguajes de marcas en entornos web



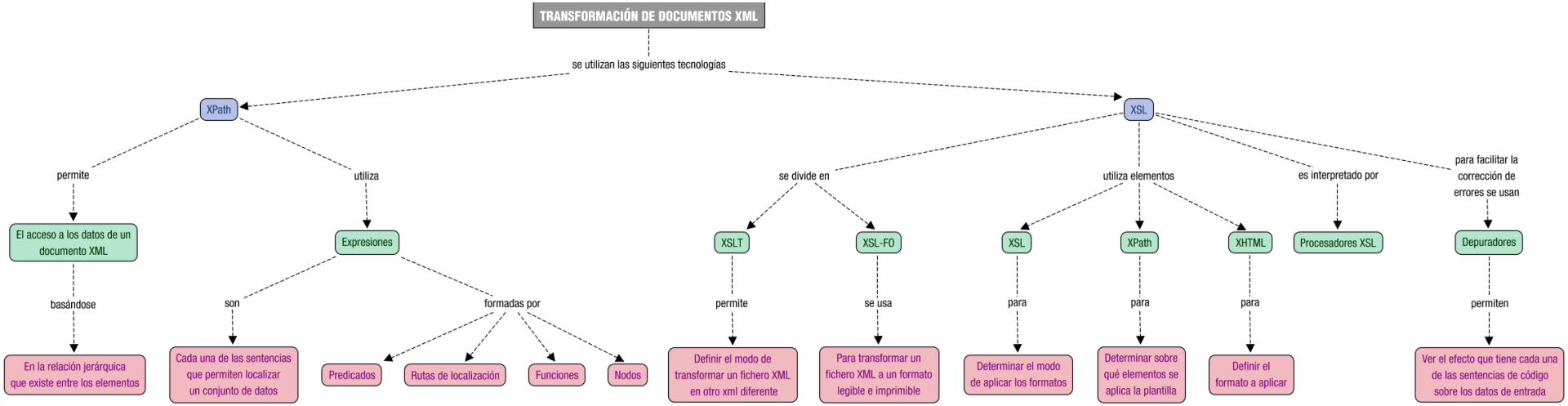
Tema 3 - La sindicación de contenidos



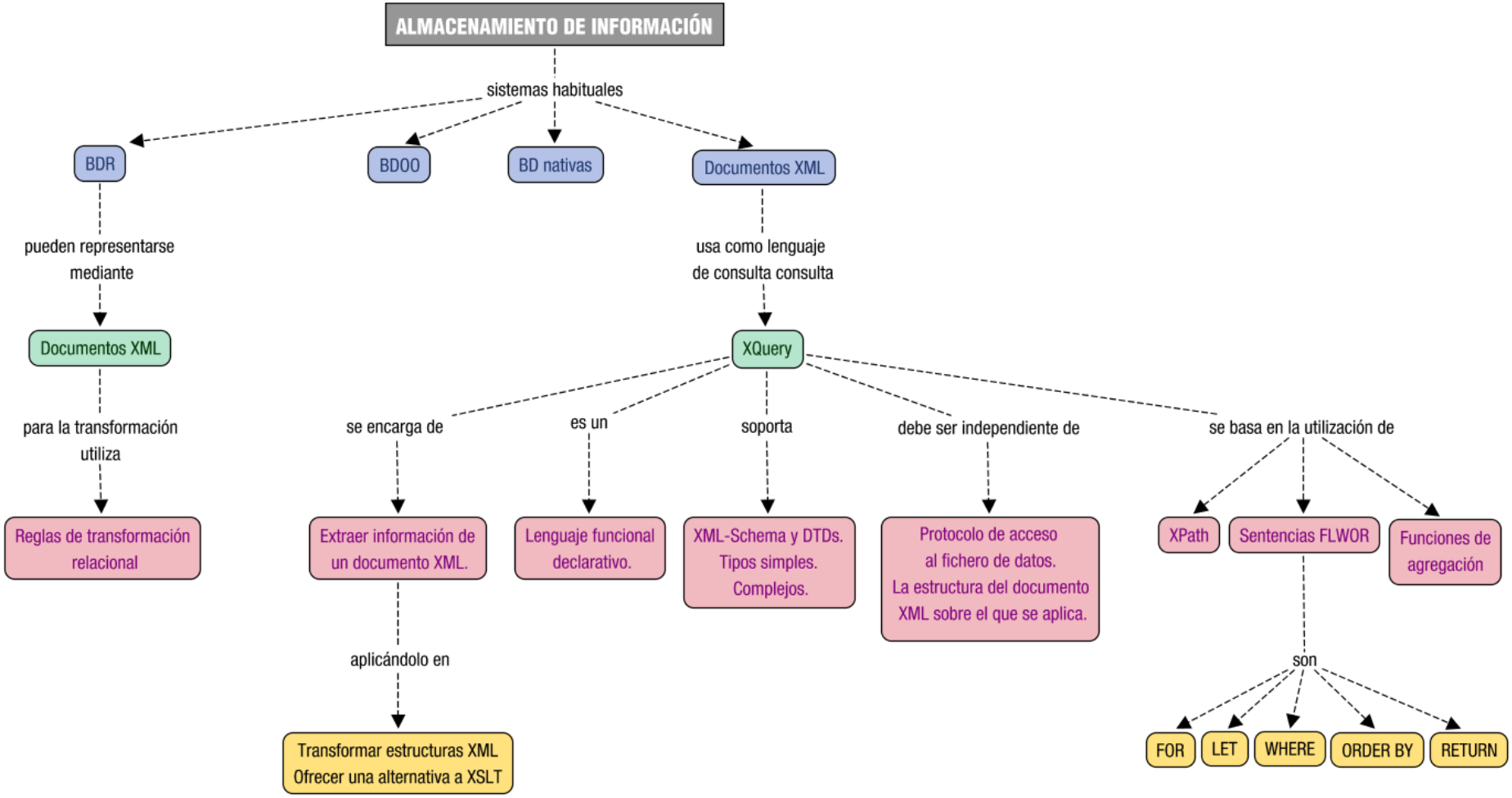
Tema 4 - Los vocabularios de XML



Tema 5 - Transformación de documentos XML



Tema 6 - Almacenamiento de información

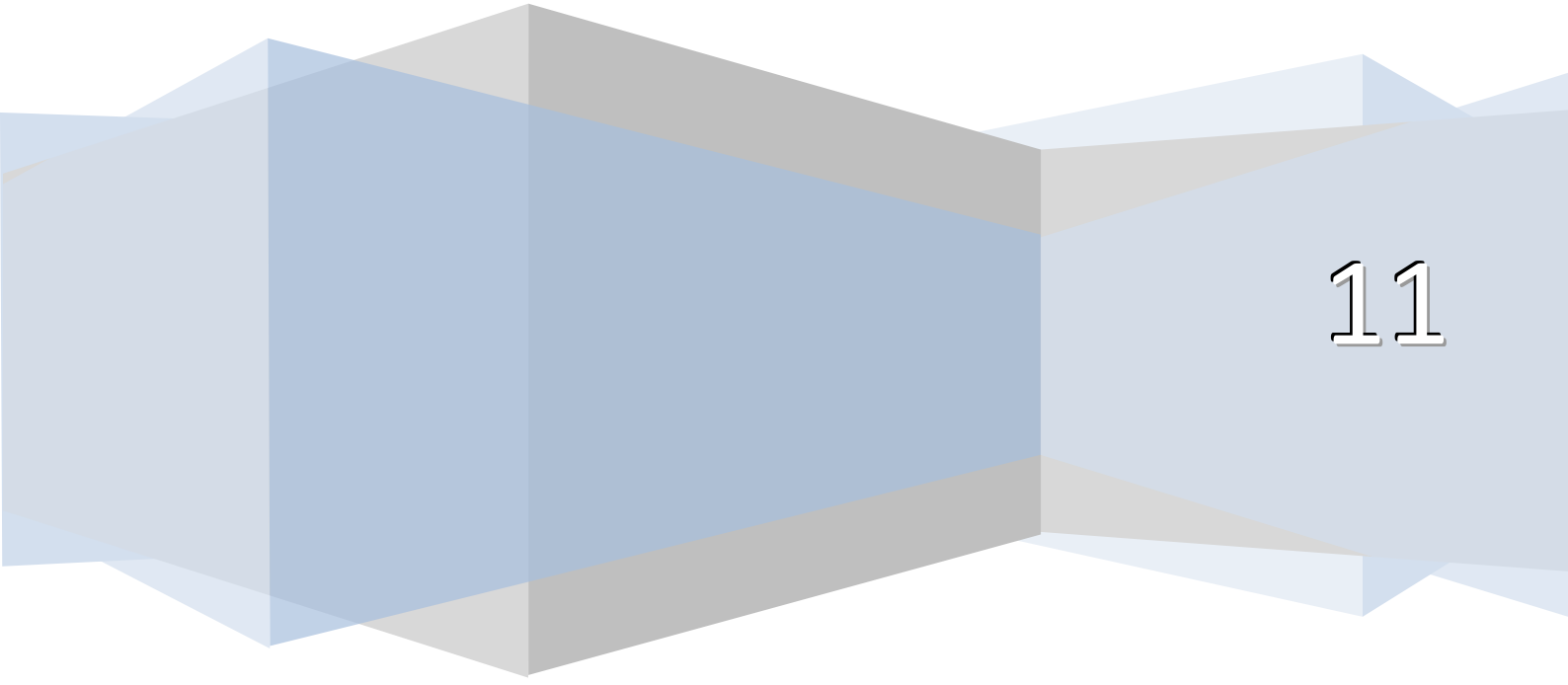


Tema 7 - Sistemas de gestión empresarial

LENGUAJES DE MARCAS

Desarrollo de Aplicaciones Web

José Luis Comesaña



11

RECONOCIMIENTO DE LAS CARACTERÍSTICAS DE LENGUAJES DE MARCAS.

Caso práctico

María y Félix son los fundadores y propietarios de una asesoría legal y empresarial, que tiene su sede en Cantabria, con oficinas en los municipios más importantes de la región.

María, licenciada en Derecho, ejercía como abogada especializada en derecho laboral y representaba a alguna empresa, además de particulares en su propio despacho situado en Torrelavega. Tenía una red informática cliente-servidor sobre un sistema operativo Windows y trabajaba con una base de datos de documentos jurídicos.

Félix, diplomado en Ciencias Empresariales, había creado una asesoría empresarial, ubicada en Santander, que básicamente, se encargaba de la contabilidad de varias PYMES. También tenía una red cliente-servidor, pero ésta bajo un sistema Linux con software libre de contabilidad.

Ambos eran amigos y un día en que habían estado hablando de sus respectivos trabajos, decidieron que sus ingresos podían aumentar sustancialmente si, además de mantener sus respectivas carteras de clientes, se unían y formaban una sociedad que ofreciese a las empresas asesoría legal y empresarial de forma conjunta.

Desde el principio, la idea de asociarse fue un éxito. Al cabo de dos años el volumen del negocio se había extendido y se hizo imprescindible el intercambio de comunicación entre ambos.

Dado que trabajaban con sistemas informáticos diferentes se plateaba el problema de cómo podían compartir información sobre los clientes comunes manteniendo la infraestructura informática con la que trabaja cada uno.

Consultaron el problema a Juan, un técnico en administración de sistemas informáticos en red, y éste les dijo que no había ningún problema de interconexión si los ficheros que manejaban se ajustaban a un formato estándar conocido como XML. Según lo que Juan les dijo, generar documentos con dicho estándar apenas requiere conocimientos previos de informática, por tanto era una solución que parecía perfecta para su problema.

1. Lenguajes de marcas.

Un "lenguaje de marcas" **es un modo de codificar un documento** donde, junto con el texto, se **incorporan etiquetas**, marcas o anotaciones **con información adicional** relativa a la estructura del texto o su formato de presentación. Permiten hacer explícita la estructura de un documento, su contenido semántico o cualquier otra información lingüística o extralingüística que se quiera hacer patente.

Todo lenguaje de marcas está definido en un documento denominado DTD (Document Type Definition). En él se establecen las marcas, los elementos utilizados por dicho lenguaje y sus correspondientes etiquetas y atributos, su sintaxis y normas de uso.

Ejemplo

Aspecto de un documento realizado con un lenguaje de marcas:

```
<carta>
<fecha>22/11/2006</fecha>
<presentacion>Estimado cliente:</presentacion>
<contenido>bla bla bla bla ...</contenido>
<firma>Don Jose Gutiérrez González</firma>
</carta>
```

Aunque en la práctica, en un mismo documento pueden combinarse varios tipos diferentes de lenguajes de marca los lenguajes de marcas, éstos se pueden clasificar como sigue:

- De presentación: Define el formato del texto.
- De procedimientos: Orientado también a la presentación pero, en este caso, el programa que representa el documento debe interpretar el código en el mismo orden en que aparece.
- Descriptivo o semántico: Describen las diferentes partes en las que se estructura el documento pero sin especificar cómo deben representarse.

Algunos ejemplos de lenguajes de marcado agrupados por su ámbito de utilización son:

- Documentación electrónica
  - **RTF** (Rich Text Format): Formato de Texto Enriquecido, fue desarrollado por Microsoft en 1987. Permite el intercambio de documentos de texto ente distintos procesadores de texto.
  - **TeX**: Su objetivo es la creación de ecuaciones matemáticas complejas.
  - **Wikitexto**: Permite la creación de páginas wiki en servidores preparados para soportar este lenguaje.
  - **DocBook**: Permite generar documentos separando la estructura lógica del documento de su formato. De este modo, dichos documentos, pueden publicarse en diferentes formatos sin necesidad de realizar modificaciones en el documento original.
- Tecnologías de internet
  - **HTML**, **XHTML**: (Hypertext Markup Language, eXtensible Hypertext Markup Language): Su objetivo es la creación de páginas web.
  - **RSS**: Permite la difusión de contenidos web
- Otros lenguajes especializados
  - **MathML** (Mathematical Markup Language): Su objetivo es expresar el formalismo matemático de tal modo que pueda ser entendido por distintos sistemas y aplicaciones.
  - **VoiceXML** (Voice Extended Markup Language) tiene como objetivo el intercambio de información entre un usuario y una aplicación con capacidad de reconocimiento de habla.
  - **MusicXML**: Permite el intercambio de partituras entre distintos editores de partituras.

Autoevaluación

Los lenguajes de marcas se utilizan para:

- ☐ Dar formato a los documentos de texto.
- ☐ Definir la estructura de los datos de un documento.
- ☐ Permitir el intercambio de ficheros entre diferentes aplicaciones y plataformas.
- ☉ Todas las anteriores.

2. Evolución de los lenguajes de marcas.

En los años 70 continúa surgen unos lenguajes informáticos, distintos de los lenguajes de programación, orientados a la gestión de información. Con el desarrollo de los editores y procesadores de texto surgen los primeros lenguajes informáticos especializados en tareas de descripción y estructuración de información: los lenguajes de marcas. Paralelamente, también, surgen otros lenguajes informáticos orientados a la representación, almacenamiento y consulta eficiente de grandes cantidades de datos: lenguajes y sistemas de bases de datos.

Los lenguajes de marcas surgieron, inicialmente, como lenguajes formados por el conjunto de códigos de formato que los procesadores de texto introducen en los documentos para dirigir el proceso de presentación (impresión) mediante una impresora. Como en el caso de los lenguajes de programación, inicialmente estos códigos de formato estaban ligados a las características de una máquina, programa o procesador de textos concreto y, en ellos, inicialmente no había nada que permitiese al programador (formateador de documentos en este caso) abstraerse de las características del procesador de textos y expresar de forma independiente a éste la estructura y la lógica interna del documento.

Ejemplo

Código de marcas anterior a GML. Las etiquetas son de invención propia.

Dado el siguiente documento:

<times 14><color verde><centrado> Este texto es un ejemplo para mostrar la utilización primitiva de las marcas</centrado></color></times 14>

<color granate><times 10><cursiva>Para realiza este ejemplo se utilizan etiquetas de nuestra invención. </cursiva> Las partes importantes del texto pueden resaltarse usando la <negrita>negrita</negrita>, o el <subrayar>subrayado</subrayar></times 10></color>

Al imprimirlo se obtendría:

Este texto es un ejemplo para mostrar la utilización primitiva de las marcas

Para realiza este ejemplo se utilizan etiquetas de nuestra invención. Las partes importantes del texto pueden resaltarse usando la **negrita**, o el subrayado

Posteriormente, se añadieron como medio de presentación a la pantalla. Los códigos de estilo de visualización anteriores ya no aparecen, y se emplean otros medios para marcados, distintos de la inclusión a mano de cadenas formateadoras, ahora ese proceso se automatiza y basta pulsar una combinación de teclas, o pulsar un botón, para lograr los resultados requeridos. Aunque esto es sólo una abstracción, para su uso interno las aplicaciones siguen utilizando marcas para delimitar aquellas partes del texto que tienen un formato especial.

Este marcado estaba exclusivamente orientado a la presentación de la información, aunque pronto se percataron de las posibilidades del marcado y le dieron nuevos usos que resolvían una gran variedad de necesidades, apareció el formato generalizado.

2.1 GML (Generalizaed Markup Language).

Uno de los problemas que se conocen desde hace décadas en la informática es la falta de estandarización en los formatos de información usados por los distintos programas.

Para resolver este problema, en los años sesenta IBM encargó a Charles F. Goldfab la construcción de un sistema de edición, almacenamiento y búsqueda de documentos legales. Tras analizar el funcionamiento de la empresa llegaron a la conclusión de que para realizar un buen procesado informático de los documentos había que establecer un formato estándar para todos los documentos que se manejaban en la empresa. Con ello se lograba gestionar cualquier documento en cualquier departamento y con cualquier aplicación, sin tener en cuenta dónde ni con qué se generó el documento. Dicho formato tenía que ser válido para los distintos tipos de documentos legales que utilizaba la empresa, por tanto, debía ser flexible para que se pudiera ajustar a las distintas situaciones.

El formato de documentos que se creó como resultado de este trabajo fue GML, cuyo objetivo era describir los documentos de tal modo que el resultado fuese independiente de la plataforma y la aplicación utilizada.

2.2 SGML (Standard Generalized Markup Language).

El formato GML evolucionó hasta que en 1986 dio lugar al estándar ISO 8879 que se denominó SGML. Éste era un lenguaje muy complejo y requería de unas herramientas de software caras. Por ello su uso ha quedado relegado a grandes aplicaciones industriales.

Ejemplo

Documento SGML sencillo:

```
<email>
   <remitente>
      <persona>
         <nombre> Pepito </nombre>
         <apellido> Grillo </apellido>
      </persona>
   </remitente>
   <destinatario>
      <direccion> pinocho@hotmail.com </direccion>
   </destinatario>
   <asunto>¿quedamos?</asunto>
   <mensaje> Hola, he visto que ponen esta noche la película que querías ver. ¿Te apetece ir?</mensaje>
</email>
```

2.3 HTML (HyperText Markup Language).

En 1989/90 Tim Berners-Lee creó el World Wide Web y se encontró con la necesidad de organizar, enlazar y compatibilizar gran cantidad de información procedente de diversos sistemas. Para resolverlo creó un lenguaje de descripción de documentos llamado HTML, que, en realidad, era una combinación de dos estándares ya existentes:

- **ASCII**: Es el formato que cualquier procesador de textos sencillo puede reconocer y almacenar. Por tanto es un formato que permite la trasferencia de datos entre diferentes ordenadores.

- **SGML**: Lenguaje que permite dar estructura al texto, resaltando los títulos o aplicando diversos formatos al texto.

HTML es una versión simplificada de SGML, ya que sólo se utilizaban las instrucciones absolutamente imprescindibles. Era tan fácil de comprender que rápidamente tuvo gran aceptación logrando lo que no

pudo SGML, HTML se convirtió en un estándar general para la creación de páginas web. Además, tanto las herramientas de software como los navegadores que permiten visualizar páginas HTML son cada vez mejores.

A pesar de todas estas ventajas HTML no es un lenguaje perfecto, sus principales desventajas son:

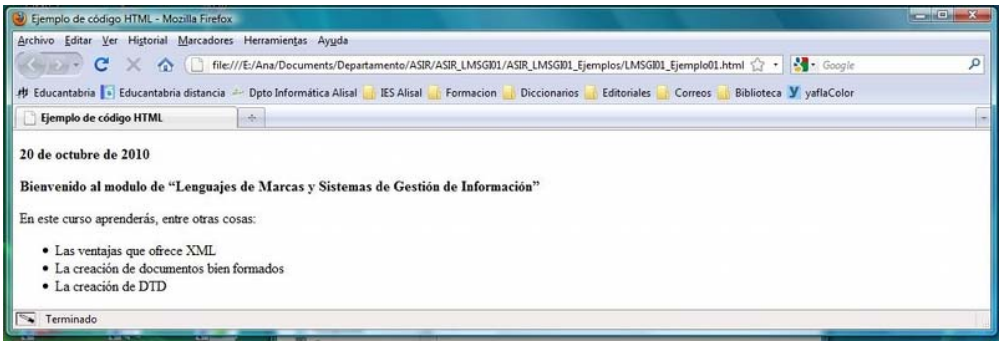
- No soporta tareas de impresión y diseño.
- El lenguaje no es flexible, ya que las etiquetas son limitadas.
- No permite mostrar contenido dinámico.
- La estructura y el diseño están mezclados en el documento.

Ejemplo

Documento HTML

```
<html>
   <head>
      <title> Ejemplo de código HTML</title>
   </head>
   <body bgcolor="#ffffff">
      <p></p>
      <p>
         <b>20 de octubre de 2010</b>
      </p>
      <p><b> Bienvenido al modulo de “Lenguajes de Marcas y Sistemas de Gestión de Información” </b></p>
      <p> En este curso aprender&aacute;s, entre otras cosas:<br/>
         <ul>
            <li>Las ventajas que ofrece XML </li>
            <li>La creaci&oacute;n de documentos bien formados </li>
            <li>La creaci&oacute;n de DTD</li>
         </ul>
      </p>
   </body>
</html>
```

Al publicarlo en un navegador, por ejemplo en el Firefox, tendríamos:



2.4 XML (eXtensible Markup Language).

Para resolver estos problemas de HTML el W3C establece, en 1998, el estándar internacional XML, un lenguaje de marcas puramente estructural que **no incluye ninguna información relativa al diseño**. Está

convirtiéndose con rapidez en estándar para el intercambio de datos en la Web. A diferencia de HTML las etiquetas indican el significado de los datos en lugar del formato con el que se van a visualizar los datos.

XML es un metalenguaje caracterizado por:

- Permitir definir etiquetas propias.
- Permitir asignar atributos a las etiquetas.
- Utilizar un esquema para definir de forma exacta las etiquetas y los atributos.
- La estructura y el diseño son independientes.

En realidad XML es un conjunto de estándares relacionados entre sí y que son:

- **XSL**, eXtensible Style Language. Permite definir hojas de estilo para los documentos XML e incluye capacidad para la transformación de documentos.
- **XML Linking Language**, incluye Xpath, Xlink y Xpointer. Determinan aspectos sobre los enlaces entre documentos XML.
- **XML Namespaces**. Proveen un contexto al que se aplican las marcas de un documento de XML y que sirve para diferenciarlas de otras con idéntico nombre válidas en otros contextos.
- **XML Schemas**. Permiten definir restricciones que se aplicarán a un documento XML. Actualmente los más usados son las DTD.

En realidad XML es un conjunto de estándares relacionados entre sí y que son:

Ejercicio resuelto

Documento XML

```
<?xml version="1.0" encoding="iso-8859-1"?>
<!DOCTYPE biblioteca">
<biblioteca>
   <ejemplar tipo_ejem="libro" titulo="XML practico" editorial="Ediciones Eni">
      <tipo> <libro isbn="978-2-7460-4958-1" edicion="1" paginas="347"></libro> </tipo>
      <autor nombre="Sebastien Lecomte"></autor>
      <autor nombre="Thierry Boulanger"></autor>
      <autor nombre="Ángel Belinchon Calleja" funcion="traductor"></autor>
      <prestado lector="Pepito Grillo">
         <fecha_pres dia="13" mes="mar" año="2009"></fecha_pres>
         <fecha_devol dia="21" mes="jun" año="2009"></fecha_devol>
      </prestado>
   </ejemplar>
   <ejemplar tipo_ejem="revista" titulo="Todo Linux 101. Virtualización en GNU/Linux" editorial="Studio Press">
      <tipo>
         <revista>
            <fecha_publicacion mes="abr" año="2009"></fecha_publicacion>
         </revista>
      </tipo>
      <autor nombre="Varios"></autor>
      <prestado lector="Pedro Picapiedra">
         <fecha_pres dia="12" mes="ene" año="2010"></fecha_pres>
      </prestado>
```

```
   </ejemplar>
</biblioteca>
```

2.5 Comparación de XML con HTML.

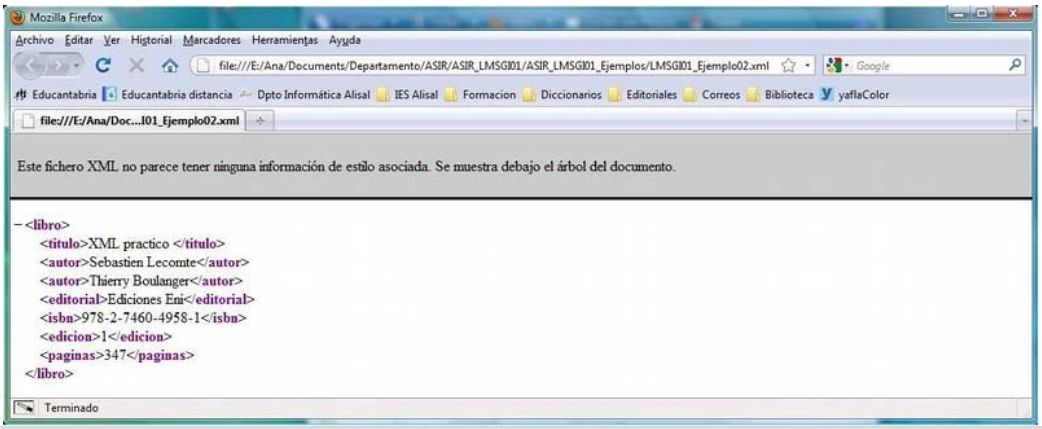
XML	HTML
• Es un perfil de SGML. • Especifica cómo deben definirse conjuntos de etiquetas aplicables a un tipo de documento. • Modelo de hiperenlaces complejo. • El navegador es una plataforma para el desarrollo de aplicaciones. • Fin de la guerra de los navegadores y etiquetas propietarias.	• Es una aplicación de SGML. • Aplica un conjunto limitado de etiquetas sobre un único tipo de documento. • Modelo de hiperenlaces simple. • El navegador es un visor de páginas. • El problema de la "no compatibilidad" y las diferencias entre navegadores ha alcanzado un punto en el que la solución es difícil

Ejemplo

Fichero XML

```
<?xml version="1.0" encoding="iso-8859-1"?>
<!DOCTYPE libro>
<libro>
   <titulo>XML practico </titulo>
   <autor>SebastienLecomte</autor>
   <autor>Thierry Boulanger</autor>
   <editorial>Ediciones Eni</editorial>
   <isbn>978-2-7460-4958-1</isbn>
   <edicion>1</edicion>
   <paginas>347</paginas>
<libro>
```

Al interpretar este fichero con un navegador, por ejemplo Mozilla, se obtiene:

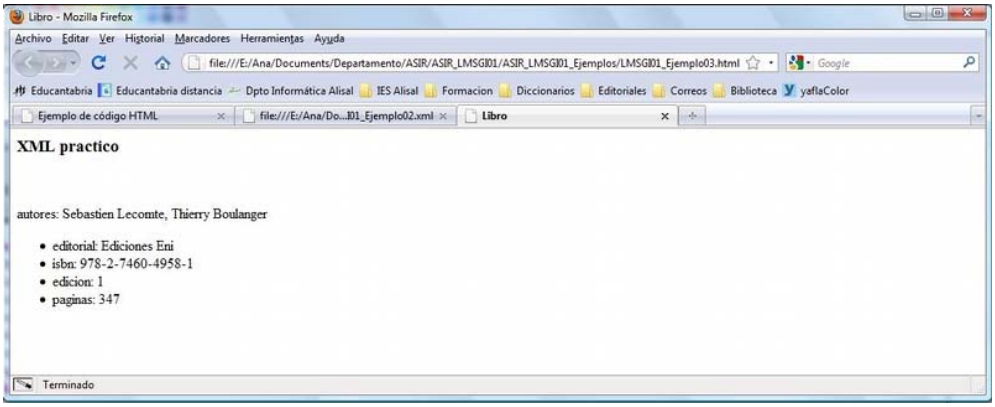


Fichero HTML

```
<html>
   <head>
      <title>Libro</title>
   </head>
   <body>
      <h3>XML practico</h3><br>
      <p>autores: Sebastien Lecomte,
```

```
      Thierry Boulanger</p>
      <ul>
         <li>editorial: Ediciones Eni</li>
         <li>isbn:978-2-7460-4958-1</li>
         <li>edicion: 1 </li>
         <li>paginas: 347</li>
      </ul>
   </body>
</html>
```

Al interpretarlo con el navegador Mozilla Firefox tendremos:



2.6 Comparación de XML con SGML.

XML	SGML
• **Su uso es sencillo.** • **Trabaja con documentos bien formados, no exige que estén validados.** • **Facilita el desarrollo de aplicaciones de bajo coste.** • **Es muy utilizado en informática y en más áreas de aplicación.** • **Compatibilidad e integración con HTML.** • **Formateo y estilos fáciles de aplicar.** • **No usa etiquetas opcionales.**	• Su uso es muy complejo. • Sólo trabaja con documentos válidos. • Su complejidad hace que las aplicaciones informáticas para procesar SGML sean muy costosas. • Sólo se utiliza en sectores muy específicos. • No hay una compatibilidad con HTML definida. • Formateo y estilos relativamente complejos.

Pregunta de Elección Múltiple

¿Cuáles son las características comunes de XML y SGML?

- ☐ Guardan el formato de un documento.
- ◉ Guardan la estructura lógica de los documentos.
- ☐ Guardan los documentos en el formato universal txt.
- ☐ Guardan el formato de los documentos independientemente de la plataforma.

Para saber más

La recomendación de XML publicada por el W3C es pública y accesible en:

http://www.w3.org/TR/2000/REC-xml-20001006

3. Etiquetas.

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

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

<etiqueta>texto que sufrirá las consecuencias de la etiqueta</etiqueta>

Ejemplo

Por ejemplo, en HTML

<u>Esto está subrayado</u>

Al interpretarlo en un navegador se verá así:

Esto está subrayado

Las últimas especificaciones emitidas por el W3C indican la necesidad de que vayan escritas **siempre en minúsculas** para considerar que el documento está correctamente creado.

Autoevaluación

¿Cuál de las siguientes líneas es correcta?

- Texto en cursiva
- Texto en cursiva
- (•) Texto en cursiva
- <I>Texto en cursiva<I>

4. Herramientas básicas de XML.

Para trabajar en XML es necesario editar los documentos y luego procesarlos, por tanto tenemos dos tipos de herramientas:

- **Editores XML**

  Una característica de los lenguajes de marcas es que se basan en la utilización de ficheros de texto plano por lo que basta utilizar un procesador de texto normal y corriente para construir un documento XML.

  Para crear documentos XML complejos e ir añadiendo datos es conveniente usar algún editor XML. Estos nos ayudan a crear estructuras y etiquetas de los elementos usados en los documentos, además algunos incluyen ayuda para la creación de otros elementos como DTD, hojas de estilo CSS o XSL, ... El W3C ha desarrollado un editor de HTML, XHTML, CSS y XML gratuito cuyo nombre es Amaya.

- **Procesadores XML**

  Para interpretar el código XML se puede utilizar cualquier navegador. Los procesadores de XML permiten leer los documentos XML y acceder a su contenido y estructura. Un procesador es un conjunto de módulos de software entre los que se encuentra un parser o analizador de XML que comprueba que el documento cumple las normas establecidas para que pueda abrirse. Estas normas pueden corresponderse con las necesarias para trabajar sólo con documentos de tipo válido o sólo exigir que el documento esté bien formado, primeros se conocen como validadores y los segundos como no validadores. El modo en que los procesadores deben leer los datos XML está descrito en la recomendación de XML establecida por W3C.

  Para publicar un documento XML en Internet se utilizan los procesadores XSLT, que permiten generar archivos HTML a partir de documentos XML.

  Puesto que XML se puede utilizar para el intercambio de datos entre aplicaciones, hay que recurrir a motores independientes que se ejecutan sin que nos demos cuenta. Entre estos destacan "XML para Java" de IBM, JAXP de Sun, etc

Autoevaluación

Para crear documentos XML es necesario:

- ☐ Software especializado para la tecnología XML
- ☐ Herramientas de validación de XML.
- ◉ Un block de notas y un navegador.
- ☐ Al menos, un editor XML.

Para saber más

Información sobre analizadores XML:

http://www.oasis-open.org/cover

Algunos de los analizadores disponibles están en los enlaces siguientes:

http://www.jclark.com/xml/expat.html
http://www.mozilla.org/rdf/doc/xml.html
http://www.microstar.com/XMLindex.htm

5. XML: estructura y sintaxis.

El XML, o Lenguaje de Etiquetas Extendido, es lenguaje de etiquetas, creadas por el programador, que estructuran y guardan de forma ordenada la información. No representa datos por sí mismo, solamente organiza la estructura.

El XML ahorra tiempos de desarrollo y proporciona ventajas, dotando a webs y a aplicaciones de una forma realmente potente de guardar la información. Además, se ha convertido en un formato universal que ha sido asimilado por todo tipo de sistemas operativos y dispositivos móviles.

Al igual que en HTML un documento XML es un documento de texto, en este caso con extensión ".xml", compuesto de parejas de etiquetas, estructuradas en árbol, que describen una función en la organización del documento, que puede editarse con cualquier editor de texto y que es interpretado por los navegadores Web.

Las características básicas de XML son:

- Dado que XML se concibió para trabajar en la Web, es directamente compatible con protocolos que ya funcionan, como HTTP y los URL.
- Todo documento que verifique las reglas de XML está conforme con SGML.
- No se requieren conocimientos de programación para realizar tareas sencillas en XML.
- Los documentos XML son fáciles de crear.
- La difusión de los documentos XML está asegurada ya que cualquier procesador de XML puede leer un documento de XML.
- El marcado de XML es legible para los humanos.
- El diseño XML es formal y conciso.
- XML es extensible, adaptable y aplicable a una gran variedad de situaciones.
- XML es orientado a objetos.
- Todo documento XML se compone exclusivamente de datos de marcado y datos carácter entremezclados.

El proceso de creación de un documento XML pasa por varias etapas en las que el éxito de cada una de ellas se basa en la calidad de la anterior. Estas etapas son:

- Especificación de requisitos.
- Diseño de etiquetas.
- Marcado de los documentos.

El marcado en XML son etiquetas que se añaden a un texto para estructurar el contenido del documento. Esta información extra permite a los ordenadores "interpretar" los textos. El marcado es todo lo que se sitúa entre los caracteres "<" y ">" o "&" y ";"

Los datos carácter son los que forman la verdadera información del documento XML.

El marcado puede ser tan rico como se quiera. Puede ser interesante detectar necesidades futuras y crear documentos con una estructura fácilmente actualizables.

Los documentos XML pueden tener comentarios, que no son interpretados por el interprete XML. Estos se incluyen entre las cadenas "<!-" y "-->", pueden estar en cualquier posición en el documento salvo:

- Antes del prólogo.
- Dentro de una etiqueta.

Los documentos XML pueden estar formados por una parte opcional llamada prólogo y otra parte obligatoria llamada ejemplar.

5.1 El prólogo.

Si se incluye, el prólogo **debe preceder al ejemplar del documento**. Su inclusión facilita el procesado de la información del ejemplar. El prólogo está dividido en dos partes:

- **La declaración XML**: En el caso de incluirse ha de ser la primera línea del documento, de no ser así se genera un error que impide que el documento sea procesado.

  El hecho de que sea opcional permite el procesamiento de documentos HTML y SGML como si fueran XML, si fuera obligatoria éstos deberían incluir una declaración de versión XML que no tienen.

  El prólogo puede tener tres funciones:

  - *Declaración la versión de XML usada para elaborar el documento.*

  Para ello se utiliza la etiqueta:

  ```
  <?xml versión= “1.0” ?>
  ```

  En este caso indica que el documento fue creado para la versión 1.0 de XML.

  - *Declaración de la codificación empleada para representar los caracteres.*

  Determina el conjunto de caracteres que se utiliza en el documento. Para ello se escribe:

  ```
  <?xml versión= “1.0” encoding=”iso-8859-1” ?>
  ```

  En este caso se usa el código iso-8859-1 (Latin-1) que permite el uso de acentos o caracteres como la ñ.

  Los códigos más importantes son:

Estándar ISO	Código de país
UTF-8 (Unicode)	Conjunto de caracteres universal
ISO -8859-1 (Latin-1)	Europa occidental, Latinoamérica
ISO -8859-2 (Latin-2)	Europa central y oriental
ISO -8859-3 (Latin-3)	Sudoeste de Europa
ISO -8859-4 (Latin-4)	Países Escandinavos, Bálticos
ISO -8859-5	Cirílico
ISO -8859-6	Árabe
ISO -8859-7	Griego
ISO -8859-8	Hebreo
ISO -8859-9	Turco
ISO-8859-10	Lapón. Nórdico, esquimal
EUC-JP oder Shitf_JIS	Japonés

- *Declaración de la autonomía del documento.*

  Informa de si el documento necesita de otro para su interpretación. Para declararlo hay que definir el prólogo completo:

  <?xml versión= “1.0” encoding=”iso-8859-1” standalone=”no” ?>

  En este caso, el documento es independiente, de no ser así el atributo *standalone* hubiese tomado el valor "*yes*".

- **La declaración del tipo de documento**, define qué tipo de documento estamos creando para ser procesado correctamente. Toda declaración de tipo de documento comienza por la cadena:

  <!DOCTYPE Nombre_tipo ...>

5.2 El ejemplar. Los elementos.

Es la parte más importante de un documento XML, ya que **contiene los datos reales del documento**. Está formado por elementos anidados.

Los elementos son los distintos bloques de información que permiten definir la estructura de un documento XML. Está, delimitados por una etiqueta de apertura y una etiqueta de cierre. A su vez los elementos pueden estar formados por otros elementos y/o por atributos.

Ejemplo

Sea el siguiente código XML

```
<?xml version="1.0" encoding="iso-8859-1"?>
<!DOCTYPE libro>
<libro>
   <titulo>XML practico </titulo>
   <autor>Sebastien Lecomte</autor>
   <autor>Thierry Boulanger</autor>
   <editorial>Ediciones Eni</editorial>
   <isbn>978-2-7460-4958-1</isbn>
   <edicion>1</edicion>
   <paginas>347</paginas>
<libro>
```

El ejemplar es el elemento <libro>, que a su vez está compuesto de los elementos <autor>, <editorial>, <isbn>, <edicion> y <paginas>.

En realidad, el ejemplar es el elemento raíz de un documento XML. Todos los datos de un documento XML han de pertenecer a un elemento del mismo.

Los nombres de las etiquetas han de ser autodescriptivos, lo que facilita el trabajo que se hace con ellas.

La formación de elementos ha de cumplir ciertas normas para que queden perfectamente definidos y que el documento XML al que pertenecen pueda ser interpretado por los procesadores XML sin generar ningún error fatal. Dichas reglas son:

- En todo documento XML debe existir un elemento raíz, y sólo uno.
- Todos los elementos tienen una etiqueta de inicio y otra de cierre. En el caso de que en el documento existan elementos vacíos, se pueden sustituir las etiquetas de inicio y cierre por una de elemento vacío. Ésta se construye como la etiqueta de inicio, pero sustituyendo el carácter ">" por "/>. Es decir, <elemento></elemento> puede sustituirse por: <elemento/>
- Al anidar elementos hay que tener en cuenta que no puede cerrarse un elemento que contenga algún otro elemento que aún no se haya cerrado.
- Los nombres de las etiquetas de inicio y de cierre de un mismo elemento han de ser idénticos, respetando las mayúsculas y minúsculas. Pueden ser cualquier cadena alfanumérica que no contenga espacios y no comience ni por el carácter dos puntos, ":", ni por la cadena "xml" ni ninguna de sus versiones en que se cambien mayúsculas y minúsculas ("XML", "XmL", "xML",...).
- El contenido de los elementos no puede contener la cadena "]]>" por compatibilidad con SGML. Además no se pueden utilizar directamente los caracteres mayor que, >, menor que, <, ampersand, &, dobles comillas, ", y apostrofe, ‘. En el caso de tener que utilizar estos caracteres se sustituyen por las siguientes cadenas:

Carácter	Cadena
>	&gt;
<	&lt;

Carácter	Cadena
&	&amp;
"	&quot;

Carácter	Cadena
‘	'

- Para utilizar caracteres especiales, como £, ©, ®,... hay que usar las expresiones &#D; o &#H; donde D y H se corresponden respectivamente con el número decimal o hexadecimal correspondiente al carácter que se quiere representar en el código UNICODE. Por ejemplo, para incluir el carácter de Euro, €, se usarían las cadenas € o €

Debes conocer

En el siguiente enlace encontrarás una tabla con los caracteres ASCII, el nombre HTML, y el número HTML de cada uno de ellos que te será imprescindible a la hora de realizar documentos en HTML y XML.

http://ascii.cl/es/codigos-html.htm

5.2.1 Atributos.

Permiten añadir propiedades a los elementos de un documento. Los atributos no pueden organizarse en ninguna jerarquía, no pueden contener ningún otro elemento o atributo y no reflejan ninguna estructura lógica.

No se debe utilizar un atributo para contener información susceptible de ser dividido.

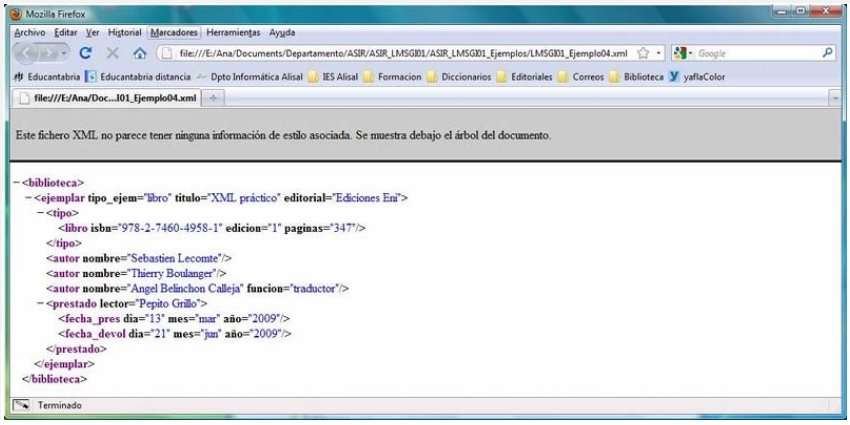
Ejemplo

Dado el siguiente código XML:

```
<?xml version="1.0" encoding="iso-8859-1" standalone="yes" ?
<!DOCTYPE biblioteca >
<biblioteca>
   <ejemplar tipo_ejem="libro" titulo="XML práctico" editorial="Ediciones Eni">
      <tipo> <libro isbn="978-2-7460-4958-1" edicion="1" paginas="347"></libro> </tipo>
```

```
        <autor nombre="Sebastien Lecomte"></autor>
        <autor nombre="Thierry Boulanger"></autor>
        <autor nombre="Angel Belinchon Calleja" funcion="traductor"></autor>
        <prestado lector="Pepito Grillo">
            <fecha_pres dia="13" mes="mar" año="2009"></fecha_pres>
            <fecha_devol dia="21" mes="jun" año="2009"></fecha_devol>
        </prestado>
    </ejemplar>
</biblioteca>
```

Al abrir el documento anterior con el navegador Firefox obtenemos:



Vemos que los elementos aparecen coloreados en ciruela, los nombres de los atributos en negro y sus valores en azul.

Como se observa en el ejemplo, los atributos se definen y dan valor dentro de una etiqueta de inicio o de elemento vacio, a continuación del nombre del elemento o de la definición de otro atributo siempre separado de ellos por un espacio. Los valores del atributo van precedidos de un igual que sigue al nombre del mismo y tienen que definirse entre comillas simples o dobles.

Los nombres de los atributos han de cumplir las mismas reglas que los de los elementos, y no pueden contener el carácter menor que, <.

Autoevaluación

¿Cuáles son los errores del siguiente documento XML?

```
<?XML version="1.0" encoding="UTF-8" standalone="yes" ?>
<!DOCTYPE biblioteca >
<biblioteca>
<ejemplar tipo_ejem='libro' titulo='XML práctico' editorial='Ediciones Eni'>
<tipo> <libro isbn='978-2-7460-4958-1' edicion= paginas='347'></libro> </tipo>
<autor nombre='Sebastien Lecomte'></autor>
<autor nombre='Thierry Boulanger'></autor>
<autor nombre='Angel Belinchon Calleja' funcion='traductor'></autor>
<prestado lector='Pepito Grillo'>
<fecha_pres dia='13' mes='mar' año='2009'></fecha_pres>
<fecha_devol/>
</prestado>
```

```
</ejemplar>
</biblioteca>
```

- [x] Utiliza mayúsculas en la definición de la versión XML.
- [x] No utiliza el código de caracteres adecuado.
- [ ] Los valores de los atributos no están entre comillas dobles.
- [ ] Hay algún atributo vacío.
- [x] La etiqueta <fecha_devol/> no se cierra.
- [ ] Se usan mayúsculas en los datos del documento.

6. Documentos XML bien formados.

Todos los documentos XML deben verificar las reglas sintácticas que define la recomendación del W3C para el estándar XML. Esas normas básicas son:

- El documento ha de tener definido un prólogo con la declaración xml completa.
- Existe un único elemento raíz para cada documento: es un solo elemento en el que todos los demás elementos y contenidos se encuentran anidados.
- Hay que cumplir las reglas sintácticas del lenguaje XML para definir los distintos elementos y atributos del documento

Autoevaluación
¿Está "bien formado" el siguiente documento XML?

```
<?xml version="1.0"?>
<mensaje>
<destinatario>Tomas</ destinatario>
<remitente>Juan</ remitente>
<asunto>
<contenido> No olvides ir a recogerme al aeropuerto mañana por la mañana!</contenido>
</mensaje>
```

Verdadero.  Falso. 

7. Utilización de espacios de nombres en XML.

Permiten definir la pertenencia de los elementos y los atributos de un documento XML al contexto de un vocabulario XML. De este modo se resuelven las ambigüedades que se pueden producir al juntar dos documentos distintos, de dos autores diferentes, que han utilizado el mismo nombre de etiqueta para representar cosas distintas.

Los espacios de nombres también conocidos como name spaces, permiten dar un nombre único a cada elemento, indexándolos según el nombre del vocabulario adecuado además están asociados a un URI que los identifica de forma única.

En el documento, las etiquetas ambiguas se sustituyen por otras en las que el nombre del elemento está precedido de un prefijo, que determina el contexto al que pertenece la etiqueta, seguido de dos puntos, :. Esto es:

```
<prefijo:nombre_etiqueta></prefijo:nombre_etiqueta>
```

Esta etiqueta se denomina "nombre cualificado". Al definir el prefijo hay que tener en cuenta que no se pueden utilizar espacios ni caracteres espaciales y que no puede comenzar por un digito.

Antes de poder utilizar un prefijo de un espacio de nombres, para resolver la ambigüedad de dos o más etiquetas, es necesario declarar el espacio de nombres, es decir, asociar un índice con el URI asignado al espacio de nombres, mediante un atributo especial xmlns. Esto se hace entre el prólogo y el ejemplar de un documento XML y su sintaxis es la siguiente:

```
<conexion>://<direccionservidor>/<apartado1>/<apartado2>/...
```

Ejemplo

Sean los documentos XML que organizan la información sobre los profesores y los alumnos del ASIR respectivamente:

```
<?xml version="1.0" encoding="iso-8859-1" standalone="yes"
<!DOCTYPE alumnos>
<alumnos>
   <nombre>Fernando Fernández González</nombre>
   <nombre>Isabel González Fernández</nombre>
   <nombre>Ricardo Martínez López</nombre>
</alumnos>
```

```
<?xml version="1.0" encoding="iso-8859-1" standalone="yes" ?>
<!DOCTYPE profesores>
<profesores>
   <nombre>Pilar Ruiz Pérez</nombre>
   <nombre>Tomás Rodríguez Hernández</nombre>
</profesores>
```

Al hacer un documento sobre los miembros del curso ASIR no se distinguirían los profesores de los alumnos, para resolverlo definiremos un espacio de nombres para cada contexto:

```
<?xml version="1.0" encoding="iso-8859-1" standalone="yes" ?>
<!DOCTYPE miembros>
<alumnos xlmns:alumnos="http://ASIR/alumnos">
<profesores xlmns:profesores="http://ASIR/profesores">
<asistentes>
   <alumnos:nombre>Fernando Fernández González</alumnos:nombre>
   <alumnos:nombre>Isabel González Fernández</alumnos:nombre>
   <alumnos:nombre>Ricardo Martínez López</alumnos:nombre>
   <profesores:nombre>Pilar Ruiz Pérez</profesores:nombre>
   <profesores:nombre>Tomás Rodríguez Hernández</profesores:nombre>
</asistentes>
```

Autoevaluación

Los espacios de nombres permiten

- (•) Utilizar etiquetas idénticas para estructurar distintos tipos de información de texto.
- ( ) Estructurar la información de un documento XML cuando proviene de varios documentos.
- ( ) Asignar varias etiquetas a una misma información.
- ( ) Definir etiquetas en otros documentos.

Para saber más

Los espacios de nombres tienen una recomendación en XML:

http://www.w3.org/TR/REC-xml-names/

INDICE

Utilización de lenguajes de marcas en entornos web.

Caso práctico
*Después de asociarse, **María** y **Félix** esperan aumentar el número de clientes de su negocio teniendo presencia en Internet, para ello hablan con **Juan**, técnico superior en Administración de Sistemas Informáticos.*
***Juan** les aconseja desarrollar una sencilla página web para la empresa, cuya funcionalidad puede ir aumentando a medida que aumenten sus necesidades. Por lo que el primer paso, será encargárselo a alguien que conozca el lenguaje HTML.*
***María** le pregunta a **Juan** por su disponibilidad y capacidad para realizar dicha tarea. **Juan** le contesta que aprendió HTML en 1998. El boom de Internet le animó a interesarse por este campo y en este lenguaje realizó su primera página web. Desde entonces el lenguaje ha sufrido cambios en los que se ha ido actualizando.*

1.- Del HTML al XHTML: evolución y versiones.

HTML es el lenguaje utilizado para crear la mayor parte de las páginas web. Es un estándar reconocido en todos los navegadores, por lo tanto, todos ellos visualizan una página HTML de forma muy similar independientemente del sistema operativo sobre el que se ejecutan.

La evolución de sus versiones, desde su creación hasta la creación del XHTML podemos verla a continuación:

El origen de HTML fue un sistema de hipertexto (*Permite crear documentos interactivos que proporcionan información adicional cuando se solicita mediante "hiperenlaces", también llamados "enlace", que relacionan dos recursos. Dichos recursos pueden ser otras páginas web, imágenes, documentos o archivos*) para compartir documentos electrónicos en 1980. La primera propuesta oficial para convertir HTML en un estándar se realizó en 1993. Aunque ninguna de las dos propuestas de estándar que se hicieron (HTML y HTML+) consiguieron convertirse en estándar oficial.

HTML 2.0 fue la primera versión oficial de HTML el IETF (*Internet Engineering Task Force o Grupo de Trabajo en Ingeniería de Internet es una organización internacional abierta de normalización, que tiene como objetivos el contribuir a la ingeniería de Internet, actuando en diversas áreas, como transporte, encaminamiento, seguridad. Fue creada en EE. UU. en 1986 y regula las propuestas y los estándares de Internet, conocidos como RFC. Es una institución sin fines de lucro y abierta a la participación de cualquier persona cuyo objetivo es velar porque la arquitectura de Internet y los protocolos que la conforman funcionen correctamente. Se la considera como la organización con más autoridad para establecer modificaciones de los parámetros técnicos bajo los que funciona la red*) publicó el estándar en septiembre de 1995.

HTML 3.2 se publicó el 14 de Enero de 1997 por el W3C. Incorpora los applets de Java (*Son pequeños programas escritos en lenguaje Java que puede incrustarse en un documento HTML permitiendo obtener una gran variedad de efectos en las páginas web. Entre sus características es digno de mencionar un esquema de seguridad que impide que los applets que se ejecutan en el equipo tengan acceso a partes sensibles del sistema, a menos que uno mismo le dé los permisos necesarios*) y texto alrededor de las imágenes.

HTML 4.0 se publicó el 24 de Abril de 1998. Entre las novedades que presenta se encuentran las hojas de estilos CSS y la posibilidad de incluir pequeños programas en las páginas web.

HTML 4.01 es la última especificación oficial de HTML se publicó el 24 de diciembre de 1999. Es una actualización de la versión anterior. En ese momento el W3C detuvo la actividad de estandarización de HTML hasta marzo de 2007, momento en que se retoma debido a la fuerza de las empresas que forman el grupo WHATWG (*Web Hypertext Application Technology Working Group. Organización fundada por trabajadores de las empresas Apple, Mozilla y Opera en 2004, su interés es ayudar a la organización W3C para regular todos*

los estándares que forman parte de la Web) y a la publicación de los borradores de **HTML 5.0**, que será la siguiente versión de este lenguaje.

Tras la publicación del estándar HTML 4.01 se detecta su incompatibilidad con herramientas basadas en XML. Para evitar estos problemas se crea lenguaje XHTML que combina la sintaxis de HTML 4.0 con la de XML.

XHTML 1.0 fue la primera versión, se publicó el 26 de Enero de 2000. Es una adaptación de HTML 4.01 al lenguaje XML, por lo que mantiene sus características, y añade algunas restricciones y elementos de XML.

La versión **XHTML 1.1** ya ha sido publicada en forma de borrador y pretende modularizar XHTML.

El borrador de **XHTML 2.0 ya ha sido publicado**, que presenta grandes novedades respecto de las anteriores versiones.

Autoevaluación
La aparición del lenguaje XHTML hace que deje de evolucionar el lenguaje HTML:

- ☐ Si.
- ◉ **No.**

2.- Estructura de un documento HTML.

Caso práctico

__Félix__ pregunta a __Juan__ si existen grandes diferencias entre el XML y el HTML. __Juan__ le explica que ambos lenguajes tienen origen en otro lenguaje que es el SGML y que sus diferencias son, principalmente, funcionales ya que la estructura del documento es semejante.

La estructura de una página HTML ha de ser coherente con la que hemos visto en el tema anterior para cualquier documento XML. Por ello tendrá un prólogo y un ejemplar.

Prólogo: Todo documento HTML ha de tener una declaración del tipo de documento donde se le indica al navegador el tipo de documento que se va a iniciar y la versión de HTML utilizada para la codificación del mismo y, además, le permite interpretarlo correctamente.

Para la versión HTML 4.0, hay tres prólogos distintos que definen tres tipos de documentos HTML:

HTML 4.0 Strict. Es la DTD utilizada por defecto con HTML 4.0. En estos documentos no se permite el uso de los elementos declarados deprecated en otras versiones o Recomendaciones HTML. La declaración del tipo de documento correspondiente es:

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 //EN" "http://www.w3.org/TR/REC-html40/strict.dtd">
```

HTML 4.0 Transitional. Permite el uso de todos los elementos que permite el HTML 4.0 `Strict`, además de los elementos `deprecated`. La declaración del tipo de documento correspondiente es:

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 transitional//EN" "http://www.w3.org/TR/REC-html40/loose.dtd">
```

HTML 4.0 Frameset. Es una variante de HTML 4.0 Transitional para documentos que usan `frames` (*Son marcos que permiten visualizar simultáneamente, en una misma página web, varios apartados diferentes*). En estos documentos el elemento `body` hay que reemplazarlo por un elemento `frameset`. La declaración del tipo de documento correspondiente es:

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Frameset//EN" "http://www.w3.org/TR/REC-html40/frameset.dtd">
```

Ejemplar: En un documento HTML está delimitado por las etiquetas <`html`> y </`html`>. El ejemplar puede, a su vez dividirse en dos partes:

La cabecera, delimitada por las etiquetas <head> y </head>. Contiene la información sobre el título de la página, el autor, palabras clave, etc. Dentro de esta sección es obligatorio definir el título del documento, para ello se usan las etiquetas <title> </title>. Está información no se presentará en la ventana del navegador, salvo el título que aparecerá en la barra de título de la parte superior.

El cuerpo, contiene la información que se va a presentar en la pantalla. Está limitado por las etiquetas <`body`> y </`body`>, salvo en los documentos de tipo HTML 4.0 Frameset donde éstas se sustituyen por <`frameset`> y </`frameset`>.

Autoevaluación

¿Cuál de las siguientes afirmaciones es verdadera?

- ☐ Un documento HTML ha de tener título aunque no tenga cabecera.
- ☐ Un documento HTML ha de tener cabecera aunque no tenga título.
- ◉ **Un documento HTML ha de tener título y cabecera.**
- ☐ Un documento HTML ha de tener ejemplar aunque no tenga título.

3.- Identificación de etiquetas y atributos de HTML.

Caso práctico

***María**, tras escuchar que XML y HTML tienen el mismo origen, se interesa por si utilizan las mismas etiquetas.*

***Juan** le contesta que en HTML las etiquetas y los atributos están definidos previamente, mientras que en XML los define el programador.*

Un documento HTML está formado por etiquetas y atributos.

Al igual que en XML las etiquetas pueden ser de apertura, <etiqueta>, o de cierre, </etiqueta>. Una de las diferencias con XML es que la cantidad de etiquetas de HTML está limitada a aquellas que están definidas por el lenguaje.

Aunque HTML define una gran cantidad de etiquetas, estas no son suficientes para crear páginas complejas ya que la definición completa de ciertos elementos, como las imágenes y los enlaces, requiere información adicional. Como no es posible crear una etiqueta por cada elemento diferente, se añade la información adicional a las etiquetas mediante los atributos dando lugar a los elementos.

Para cada uno de los atributos hay definido un conjunto de valores que se le puede asignar, si el valor de un atributo no es válido, el navegador le ignora.

Cada una de las etiquetas HTML define los atributos que puede utilizar, aunque algunos de ellos son comunes a muchas etiquetas.

Autoevaluación

Las etiquetas de HTML, al igual que las de XML, son ilimitadas:

- ○ Verdadero.
- ◉ **Falso.**

3.1.- Clasificación de los atributos comunes según su funcionalidad.

Atributos básicos: Se pueden usar en casi todas las etiquetas HTML.

Atributo	Descripción
name = "texto"	Permite asignar el nombre "texto" a un objeto HTML
title = "texto"	Asigna un título a un elemento HTML, mejorando así la accesibilidad. Dicho título es mostrado por los navegadores cuando el usuario pasa el ratón por encima del elemento. Es especialmente útil con los elementos: a, link, img, object, abbr y acronym
id = "texto"	Permite identificar al elemento HTML sobre el que se aplica de forma única mediante el identificador "texto". Sólo es útil cuando se trabaja con CSS y con Javascript. No pueden empezar por números y sólo puede contener letras, números, guiones medios y/o guiones bajos.
style = "texto"	Permite aplicar al elemento HTML el estilo "texto" directamente.
class = "texto"	Permite aplicar al elemento HTML el estilo "texto" definido en las CSS. No pueden empezar por números y sólo puede contener letras, números, guiones medios y/o guiones bajos.

Atributos para internacionalización: Los utilizan las páginas que muestran sus contenidos en varios idiomas o aquellas que quieren indicar de forma explícita el idioma de sus contenidos

<table>
<tr><th>Atributo</th><th>Descripción</th></tr>
<tr><td>dir</td><td>Indica la dirección del texto por lo que sólo puede tomar dos valores:
ltr (left to right) de izquierda a derecha. Es el valor por defecto.
rtl (right to left) de derecha a izquierda.</td></tr>
<tr><td>lang = "codigo"</td><td>Especifica el idioma del elemento mediante un código predefinido. Los posibles valores de este atributo se encuentran en el documento RFC 1766, algunos de los valores posibles son:
<table>
<tr><th>Código</th><th>Idioma</th><th>Código</th><th>Idioma</th></tr>
<tr><td>en</td><td>Inglés (Gran Bretaña)</td><td>es</td><td>Español</td></tr>
<tr><td>en-US</td><td>Inglés americano</td><td>fr</td><td>Francés</td></tr>
<tr><td>ja</td><td>Japones</td><td>fr-CA</td><td>Francés de Canada</td></tr>
</table></td></tr>
<tr><td>xml:lang = "codigo"</td><td>Especifica el idioma del elemento mediante un código definido según la recomendación RFC 1766.</td></tr>
</table>

En las páginas XHTML, el atributo xml:lang tiene más prioridad que lang y es obligatorio incluirlo siempre que se incluye el atributo lang.

Atributos de eventos y atributos para los elementos que pueden obtener el foco: Sólo se utilizan en las páginas web dinámicas creadas con JavaScript. Como no es nuestro objetivo no lo vamos a contemplar.

3.2.- Elementos HTML.

Un elemento HTML está formado por:

Una etiqueta de apertura.

Cero o más atributos.

Texto encerrado por la etiqueta. Es opcional, no todas las etiquetas pueden encerrar texto.

Una etiqueta de cierre.

Según el modo en que ocupan el espacio disponible en la página los elementos pueden ser de dos tipos:

Elementos en línea. Sólo ocupan el espacio necesario para mostrar sus contenidos. Su contenido puede ser texto u otros elementos en línea.

Elementos de bloque. Los elementos de bloque siempre empiezan en una nueva línea y ocupan todo el espacio disponible hasta el final de la línea, aunque sus contenidos no lleguen hasta allí. Su contenido puede ser texto, elementos en línea u otros elementos de bloque.

Existen elementos cuyo comportamiento puede ser en línea o de bloque según las circunstancias.

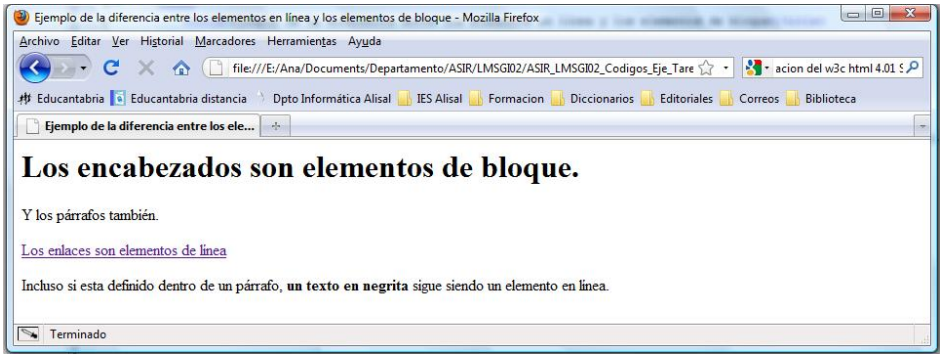
El siguiente ejemplo muestra la diferencia entre ambos comportamientos:

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 transitional//EN" "http://www.w3.org/TR/REC-html40/loose.dtd">

<html>
    <head>
        <title>Ejemplo de la diferencia entre los elementos en l&#xed;nea y los elementos de bloque</title>
    </head>

    <body>
        <h1>Los encabezados son elementos de bloque.</h1>
        <p>Y los p&#xe1;rrafos tambi&#xe9;n.</p>
        <a href="http://www.infoalisal.com">Los enlaces son elementos de l&#xc3;&#xad;nea</a>
        <p>Incluso si esta definido dentro de un p&#xc3;&#xa1;rrafo, <b>un texto en negrita</b>
        sigue siendo un elemento en l&#xc3;&#xad;nea. </p>
    </body>
</html>
```

Al publicarlo en un navegador, por ejemplo en el Firefox, tendríamos:



Autoevaluación
Los elementos de línea y de bloque se diferencian en:

- ☐ Los de línea sólo actúan sobre una línea de texto y los de bloque actúan sobre más de una línea.
- ☐ Los elementos de bloque pueden actuar como elementos de línea pero los de línea no pueden actuar como elementos de bloque.
- ◉ **Los de línea ocupan el espacio imprescindible mientras que los de bloque no.**
- ☐ Tanto los de línea como los de bloque ocupan el espacio imprescindible.

3.2.1.- Elementos de la estructura básica del documento.

La estructura básica de un documento viene determinada por las siguientes etiquetas:

Elemento	Descripción
html	Documento HTML.
head	Cabecera del documento.
body	Cuerpo del documento. Permite definir formatos que se aplican a los elementos de la página de manera global, como son el color del fondo del texto, los márgenes, el color de los enlaces, ...

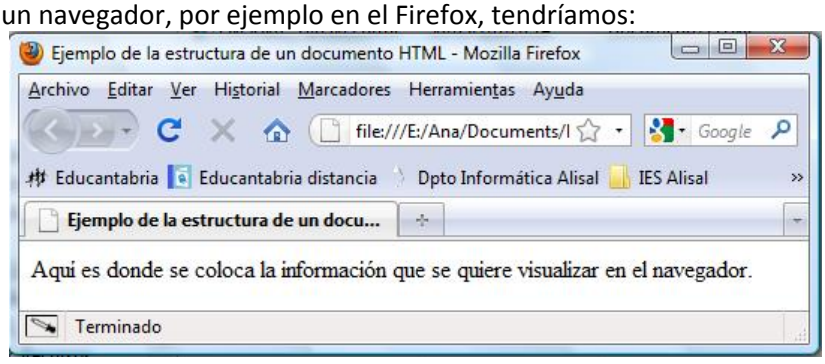
Un ejemplo de un documento HTML básico que utiliza estos elementos es:

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 transitional//EN" "http://www.w3.org/TR/REC-html40/loose.dtd">

<html>
    <head>
        <title>Ejemplo de la estructura de un documento HTML</title>
    </head>

    <body>
        Aqu&iacute; es donde se coloca la información que se quiere visualizar en el navegador.
    </body>
</html>
```

Al publicarlo en un navegador, por ejemplo en el Firefox, tendríamos:



Autoevaluación
La etiqueta body encierra los elementos que van a determinar el formato del documento:

- ◉ **Verdadero.**
- ○ Falso.

3.2.2.- Elementos de la sección de cabecera.

Elementos contenedores:

Elemento	Descripción
title	Título del documento.
script	Script incrustado. Su contenido ha de ir situado entre las marcas de comentarios ya que no ha de ser interpretado.
style	Estilo aplicado al documento utilizando CSS. Su contenido ha de ir situado entre las marcas de comentarios ya que no ha de ser interpretado.

Elementos no contenedores:

Elemento	Descripción
base	URI base del documento
isindex	Prompt de entrada de datos.
link	Enlaces a documentos externos de librerías
meta	Información que agiliza la búsqueda del documento en buscadores.

3.2.3.- Elementos que dan formato al texto de un párrafo.

Los distintos elementos que podemos utilizar para dar formato a nuestro texto son:

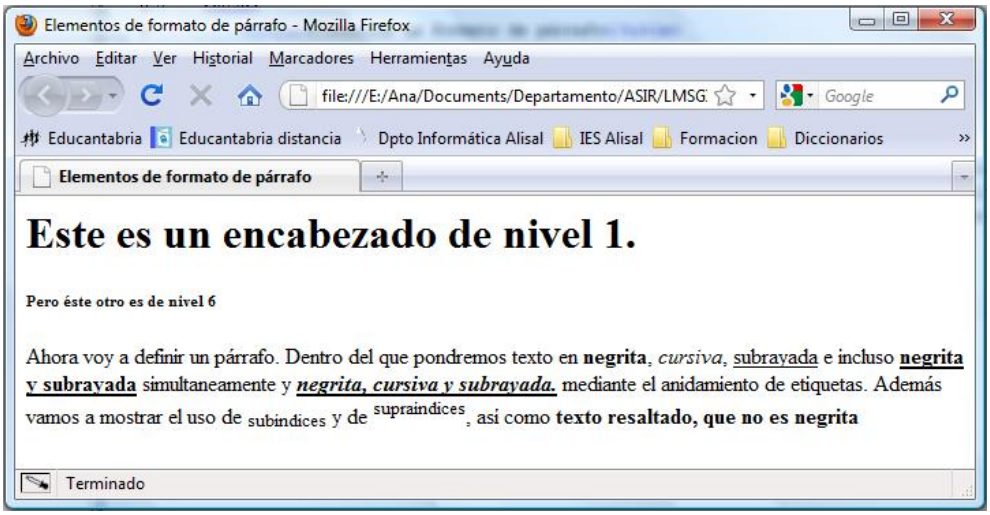
Elemento	Descripción
p	Delimita los párrafos
hi	Encabezado de nivel i, donde i es un número entero entre 1 y 6, ambos inclusive. El tamaño de la letra del encabezado es mayor cuanto menor sea el valor de i. No deben de usarse estas etiquetas para formatear texto. Sólo estarán bien usadas para designar títulos de párrafos.
b	Indica que el texto que está en ese elemento se le pondrá en negrita.
i	Indica que el texto que está en ese elemento se le pondrá en itálica ó cursiva.
u	Indica que el texto que está en ese elemento se le pondrá subrayado.
sup	Indica que el texto que está en ese elemento es un supraíndice.
sub	Indica que el texto que está en ese elemento es un subíndice.
strong	Indica que el texto que está en ese elemento estará resaltado. Habitualmente los navegadores resaltan el texto poniéndolo en negrita aunque podría haber algún navegador que resaltase el texto poniéndolo en cursiva y en naranja.

Un ejemplo de un documento HTML que utiliza estos elementos es:

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 transitional//EN" "http://www.w3.org/TR/REC-html40/loose.dtd">

<html>
    <head>
        <title>Elementos de formato de p&aacute;rrafo</title>
    </head>

    <body>
        <h1>Este es un encabezado de nivel 1.</h1>
        <h6>Pero &eacute;ste otro es de nivel 6</h6>
        <p>Ahora voy a definir un p&aacute;rrafo. Dentro del que pondremos texto en <b>negrita</b>,
        <i>cursiva</i>, <u>subrayada</u> e incluso <b><u>negrita y subrayada</u></b> simultaneamente y
        <b><i><u>negrita, cursiva y subrayada.</u></i></b> mediante el anidamiento de etiquetas.
        Además vamos a mostrar el uso de <sub>sub&iacute;ndices</sub> y de <sup>supra&iacute;ndices</sup>,
         as&iacute; como <strong>texto resaltado, que no es negrita</strong>
        </p>
    </body>
</html>
```



Autoevaluación
Para poner un párrafo en negrita utilizaremos la etiqueta:

- <b>
-
- Ambas.
- <n>

3.2.4.- Elementos de listas.

Hay tres tipos de listas: ordenadas, desordenadas y listas de definición.

Elemento	Descripción
ul	Delimita los elementos que forman una lista desordenada
ol	Delimita los elementos que forman una lista ordenada
li	Indica cada uno de los elementos de una lista
dl	Delimita los elementos que forman una lista de definición
dt	Cada uno de los términos que se definen de una lista de definición.
dd	Cada una de las definiciones de una lista de definición.

Un ejemplo de un documento HTML que muestra la forma de utilizar estos elementos es:

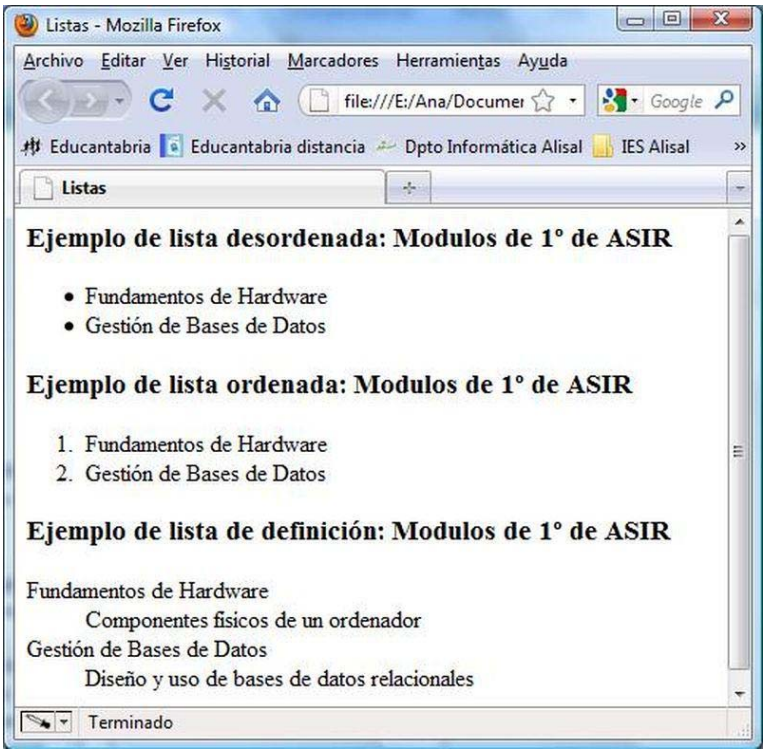
```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 transitional//EN" "http://www.w3.org/TR/REC-html40/loose.dtd">

<html>
    <head>
        <title>Listas</title>
    </head>

    <body>
        <h3>Ejemplo de lista desordenada: Modulos de 1º de ASIR</h3>
        <ul>  <li>Fundamentos de Hardware</li>
                <li>Gesti&oacute;n de Bases de Datos</li>
        </ul>

        <h3>Ejemplo de lista ordenada: Modulos de 1º de ASIR</h3>
        <ol> <li>Fundamentos de Hardware</li>
            <li>Gesti&oacute;n de Bases de Datos</li>
        </ol>

        <h3>Ejemplo de lista de definici&oacute;n: Modulos de 1º de ASIR</h3>
        <dl> <dt>Fundamentos de Hardware</dt>
            <dd>Componentes f&iacute;sicos de un ordenador</dd>
            <dt>Gesti&oacute;n de Bases de Datos</dt>
            <dd>Diseño y uso de bases de datos relacionales</dd>
        </dl>
    </body>
</html>
```



3.2.5.- Elementos de tablas.

Los elementos para definir una tabla son los siguientes:

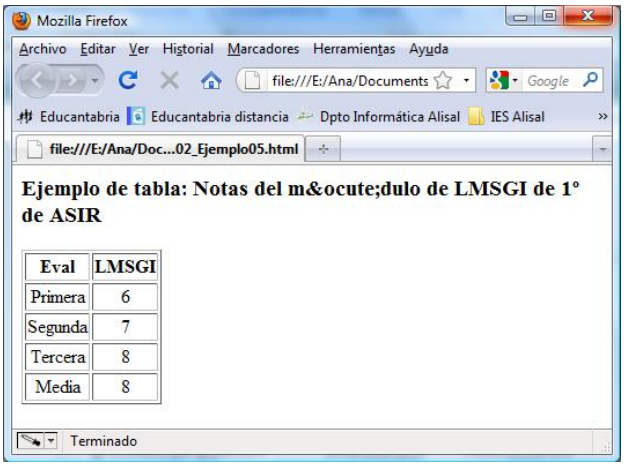
Elemento	Descripción
table	Delimita el contenido de una tabla.
tr	Delimita cada una de las líneas de la tabla.
td	Delimita el contenido de cada celda de la tabla.
colgroup	Permite agrupar columnas.
tbody	Permite agrupar líneas de la tabla.
thead	Define la línea cabecera de la tabla.
th	Delimita cada una de las celdas de la cabecera
tfoot	Define la fila pie de la tabla.

Un ejemplo de un documento HTML que muestra el modo de utilizar algunos de estos elementos es:

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 transitional//EN" "http://www.w3.org/TR/REC-html40/loose.dtd">

<html>
    <head>
        <title>Tablas</title>
    </head>

    <body>
        <h3>Ejemplo de tabla: Notas del m&ocute;dulo de LMSGI de 1º de ASIR</h3>
        <table border = "1">
            <thead> <tr> <th>Eval</th> <th>LMSGI</th> </tr>  </thead>
            <tfoot align="center"> <tr> <td>Media</td> <td>8</td> </tr> </tfoot>
            <tbody align="center">
                <tr> <td>Primera</td> <td>6</td> </tr>
                <tr> <td>Segunda</td> <td>7</td> </tr>
                <tr>  <td>Tercera</td> <td>8</td> </tr>
            </tbody>
        </table>
    </body>
</html>
```

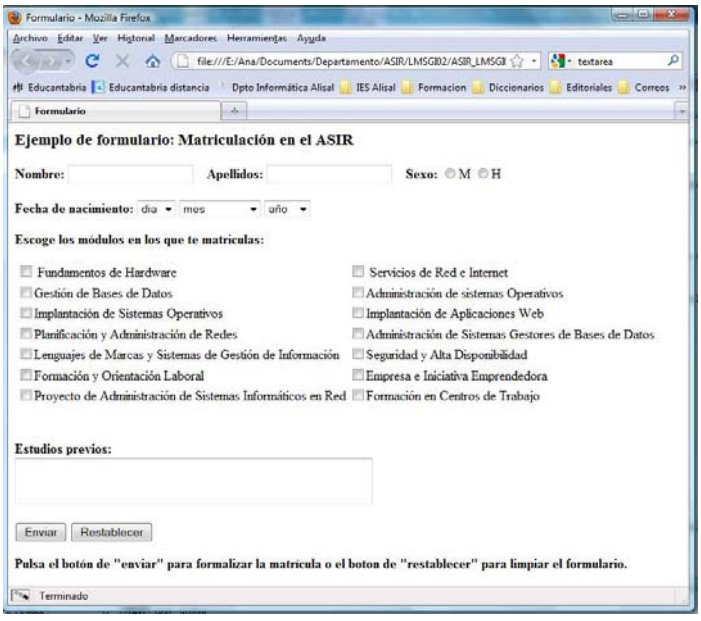


3.2.6.- Elementos de formularios.

Lo elementos que puede contener un formulario son los siguientes:

Elemento	Descripción
form	Delimita el contenido del formulario
input	Caja de texto para texto corto. Dependiendo del valor que tome el atributo type de este elemento podemos estar ante un texto sin más, un campo de texto donde al escribir no se visualice el contenido si no que escriba asteriscos, un botón de radio que el usuario podrá elegir, una opción que el usuario podrá activar, botones, ...
textarea	Caja de texto para texto largo.
select	Crea un menú desplegable que permite elegir de una lista de opciones que contiene el elemento.
option	Delimita cada una de las opciones de un menú desplegable que le contiene.
button	Permite definir un botón. Su principal ventaja frente a los botones hechos con input es que este elemento permite introducir en el botón cualquier otro elemento de HTML, como por ejemplo imágenes.
fieldset	Permite agrupar elementos de un un formulario.
legend	Permite poner un título al fieldset.
label	Etiqueta de un campo del formulario.

¿Cuál podría ser el código HTML asociado al documento que se muestra en la siguiente imagen?



Teclearíamos algo similar a esto:

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 transitional//EN" "http://www.w3.org/TR/REC-
html40/loose.dtd">

<html>
    <head>
        <title>Formulario</title>
    </head>

    <body>
        <h3>Ejemplo de formulario: Matriculación en el ASIR</h3>
        <form
action="http://www.juntadeandalucia.es/educacion/adistancia/cursos/file.php/433/LMSGI02/LMSGIO
2_Contenidos_WEB/matricula.php" method="get">
            <b>Nombre:</b> <input type="text" name="Nombre"></input>
               <b>Apellidos:</b> <input type="text" name="Apellidos"></input>
               <b>Sexo:</b> <input type="radio" name="Sexo"/>M
            <input type="radio" name="Sexo"/>H

            <br/><br/><b>Fecha de nacimiento:</b>
            <select>
                <option>dia</option>
                <option>1</option>
                <option>2</option>
                <option>3</option>
                <option>4</option>
                <option>5</option>
                <option>6</option>
                <option>7</option>
                <option>8</option>
                <option>9</option>
                <option>10</option>
                <option>11</option>
                <option>12</option>
                <option>13</option>
                <option>14</option>
                <option>15</option>
                <option>16</option>
                <option>17</option>
                <option>18</option>
                <option>19</option>
                <option>20</option>
                <option>21</option>
                <option>22</option>
                <option>23</option>
                <option>24</option>
                <option>25</option>
                <option>26</option>
                <option>27</option>
                <option>28</option>
                <option>29</option>
                <option>30</option>
                <option>31</option>
            </select>
            <select>
                <option>mes</option>
                <option>Enero</option>
                <option>Febrero</option>
                <option>Marzo</option>
                <option>Abril</option>
                <option>Mayo</option>
                <option>Junio</option>
                <option>Julio</option>
                <option>Agosto</option>
                <option>Septiembre</option>
                <option>Octubre</option>
                <option>Noviembre</option>
                <option>Diciembre</option>
            </select>
            <select>
```

```
                <option>año</option>
                <option>1992</option>
                <option>1991</option>
                <option>1990</option>
                <option>1989</option>
                <option>1988</option>
                <option>1987</option>
                <option>1986</option>
            </select>
            <br/>
            <p><b>Escoge los módulos en los que te matriculas: </b></p>
            <table>
                <tr>
                    <td>
                        <input type="checkbox" name="Modulos"/> Fundamentos de Hardware
                    </td>
                    <td>
                        <input type="checkbox" name="Modulos"/> Servicios de Red e Internet
                    </td>
                </tr>
                <tr>
                    <td>
                        <input type="checkbox" name="Modulos"/>Gestión de Bases de Datos
                    </td>
                    <td>
                        <input  type="checkbox"  name="Modulos"/>Administración  de  sistemas
Operativos
                    </td>
                </tr>
                <tr>
                    <td>
                       <input   type="checkbox"   name="Modulos"/>Implantación   de   Sistemas
Operativos
                    </td>
                    <td>
                        <input  type="checkbox"  name="Modulos"/>Implantación  de  Aplicaciones
Web
                    </td>
                </tr>
                <tr>
                    <td>
                        <input type="checkbox" name="Modulos"/>Planificación y Administración
de Redes
                    </td>
                    <td>
                        <input  type="checkbox"  name="Modulos"/>Administración  de  Sistemas
Gestores de Bases de Datos</input>
                    </td>
                </tr>
                <tr>
                    <td>
                        <input type="checkbox" name="Modulos"/>Lenguajes de Marcas y Sistemas
de Gestión de Información</input>
                    </td>
                    <td>
                        <input type="checkbox" name="Modulos"/>Seguridad y Alta Disponibilidad
                    </td>
                </tr>
                <tr>
                    <td>
                        <input type="checkbox" name="Modulos"/>Formación y Orientación Laboral
                    </td>
                    <td>
                        <input    type="checkbox"    name="Modulos"/>Empresa    e    Iniciativa
Emprendedora
                    </td>
                </tr>
                <tr>
                    <td>
                        <input  type="checkbox"  name="Modulos"/>Proyecto  de  Administración  de
Sistemas Informáticos en Red
```

```
                    </td>
                    <td>
                        <input type="checkbox" name="Modulos"/>Formación en Centros de Trabajo
                    </td>
                </tr>
            </table>
            <br/><br/>
            <b>Estudios previos:</b>
            <br/>
            <textarea rows="2" cols="50"></textarea>
            <br/><br/>
            <button type="button">Enviar</button>
            <input type="reset" name="Reset"></input>
        </form>
        <p><b>Pulsa el botón de "enviar" para formalizar la matrícula o el boton de "restablecer" para limpiar el formulario.</b></p>
    </body>
</html>
```

Autoevaluación

Para crear un botón utilizaremos la etiqueta:

- ☐ <button>
- ☐ <input>
- ◉ **Ambas.**
- ☐ Ninguna.

3.2.7.- Elementos de frames.

Los elementos utilizados para trabajar con frames son:

Elemento	Descripción
frameset	Define la partición de la ventana del navegador en marcos. Sólo puede partirse en filas o en columnas. Para partir la ventana del navegador en filas y columnas hay que anidar frames.
Frame	Define un marco que contiene información

Un ejemplo de un documento HTML que utiliza estos elementos es:

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Frameset//EN" "http://www.w3.org/TR/REC-html40/frameset.dtd">

<html>
    <head> <title>Frames</title> </head>

    <frameset rows="20%,30%, *">
            <frame src="LMSGI02_Ejemplo07_1.html" />  <frame src="LMSGI02_Ejemplo03.html" />
            <frame src="LMSGI02_Ejemplo05.html" />
    </frameset>
</html>
```

El contenido del fichero LMSGI02_Ejemplo07_1.html es:

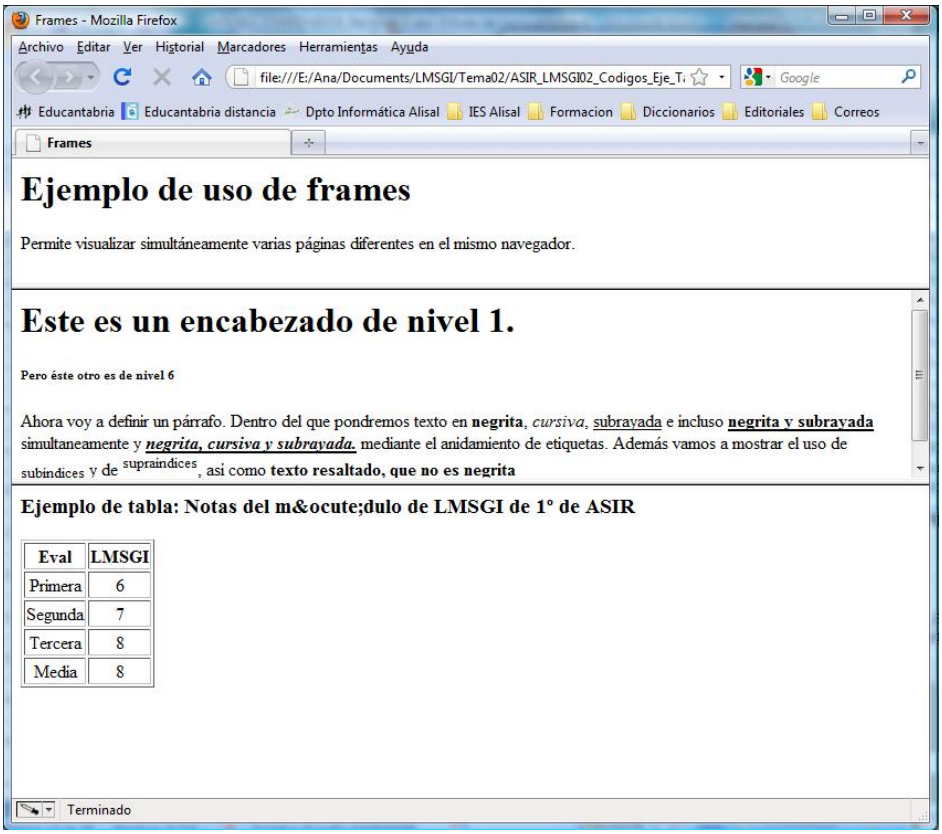
```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 transitional//EN" "http://www.w3.org/TR/REC-html40/loose.dtd">

<html>
    <head> <title>Frames_2</title> </head>

    <body>
        <h1>Ejemplo de uso de frames</h1>
        <p>Permite visualizar simult&aacute;neamente varias p&aacute;ginas diferentes en el mismo navegador.</p>
    </body>
</html>
```

Los ficheros LMSGI02_Ejemplo03.html y LMSGI02_Ejemplo05.html se corresponden con los códigos de ejemplos que hemos visto anteriormente para mostrar el uso de los formatos de texto y las tablas.

Al publicarlo en un navegador, por ejemplo en el Firefox, tendríamos:

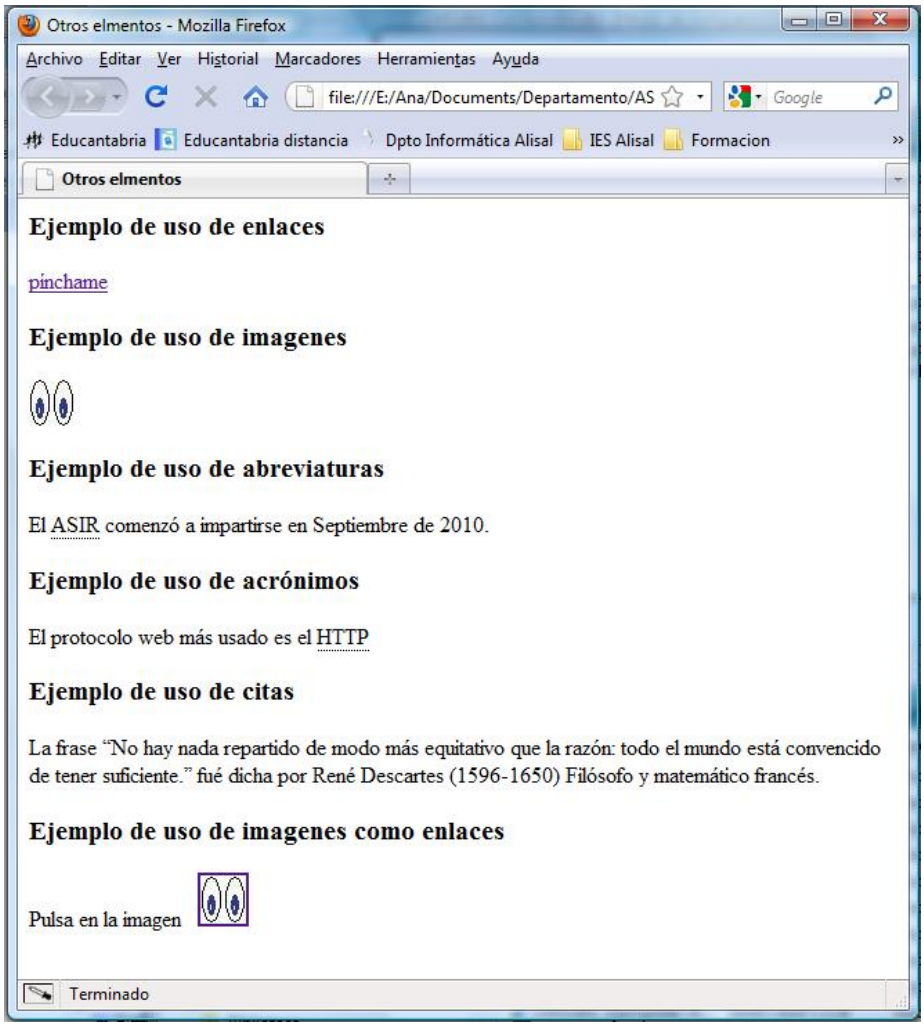


3.2.8.- Otros elementos.

Otros elementos que podemos utilizar son:

Elemento	Descripción
a	Permite definir un enlace a una página web, un archivo o una dirección de correo.
img	Permite insertar una imagen en una página web. Es obligatorio utilizar el atributo src para determinar el path del fichero de imagen que queremos insertar.
abbr	Indica una forma abreviada.
acronym	Indica un acrónimo.
blockquote	Contiene un bloque de texto con sangría.
q	Contiene una cita por lo que el navegador le añade las marcas de citación
br	Inserta una línea en blanco. No tiene etiqueta de cierre, sólo se abre.

¿Cuál podría ser el código HTML asociado al documento que se muestra en la imagen?



El código HTML asociado a ese documento podría ser el siguiente:

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 transitional//EN" "http://www.w3.org/TR/REC-html40/loose.dtd">

<html>
    <head>
        <title>Otros elmentos</title>
    </head>

    <body>
        <h3>Ejemplo de uso de enlaces</h3>
        <a href="http://www.educantabria.es/">p&iacute;nchame</a>

        <h3>Ejemplo de uso de imagenes</h3>
        <img src="ojos.jpg" ></img>

        <h3>Ejemplo de uso de abreviaturas</h3>
        El <abbr title="Administraci&oacute;n de Sistemas Inform&aacute;ticos y en Red">ASIR</abbr>
        comenz&oacute; a impartirse en Septiembre de 2010.

        <h3>Ejemplo de uso de acr&oacute;nimos</h3>
        El protocolo web m&aacute;s usado es el <acronym title="Hypertext Transfer Protocol">HTTP</acronym>

        <h3>Ejemplo de uso de citas</h3>
        La frase <q>No hay nada repartido de modo m&aacute;s equitativo que la raz&oacute;n: todo el mundo
        est&aacute; convencido de tener suficiente.</q> fu&eacute; dicha por Ren&eacute; Descartes (1596-1650)
        Fil&oacute;sofo y matem&aacute;tico franc&eacute;s.

        <h3>Ejemplo de uso de im&aacute;genes como enlaces</h3>
        Pulsa en la imagen   <a href="http://www.educantabria.es/"><img src="ojos.jpg" ></a>
    </body>
</html>
```

Autoevaluación
Todos los elementos de HTML están formados por:

- ☐ Etiquetas de apertura y cierre.
- ☐ Etiquetas de apertura y cierre, un solo atributo y su valor correspondiente.
- ☐ Etiquetas de apertura, cierre, varios atributos y sus valores.
- ☉ **Etiquetas de apertura, cierre, ninguno o varios atributos y sus valores.**

Debes conocer

Deberías conocer la especificación de HTML 4.01 que hace el W3C, así que te invitamos a que las conozcas:

Recomendación del W3C "HTML 4.01 Specification"	http://www.w3.org/TR/html401/struct/global.html
Traducción de esta recomendación al castellano	http://www.lawebera.es/manuales/html/

4.- XHTML frente a HTML.

Caso práctico

*Puesto que **Juan** ha aceptado realizar la página web se plantea el hacerla en HTML o en XHTML consulta con **Marina**, trabajadora de su empresa informática.*

***Marina** opina que, desde un punto de vista formal, no hay diferencias sustanciales entre utilizar uno u otro lenguaje, y siguiendo la evolución lógica le parece que sería más apropiado utilizar XHTML y añade que este lenguaje tiene la ventaja de ser compatible con navegadores antiguos.*

***Juan** opina que es una buena opción, pero que algunos navegadores, a pesar de ser compatibles con el lenguaje no interpretan los formatos.*

El lenguaje XHTML es muy similar al lenguaje HTML. De hecho, no es más que una adaptación de HTML al lenguaje XML, el estándar XHTML 1.0 sólo añade pequeñas mejoras y modificaciones menores al estándar HTML 4.01, por lo que este último está prácticamente incluido en el primero, lo que hace que pasar del HTML 4.01 Strict a XHTML no requiere casi ningún cambio.

El lenguaje HTML tiene una sintaxis muy permisiva, por lo que es posible escribir sus etiquetas y atributos de muchas formas diferentes. Las etiquetas, por ejemplo, podían escribirse en mayúsculas, en minúsculas e incluso combinando mayúsculas y minúsculas. El valor de los atributos de las etiquetas se pueden indicar con o sin comillas. Además, el orden en el que se abrían y cerraban las etiquetas no era importante.

La flexibilidad de HTML da lugar a páginas con un código desordenado, difícil de mantener y muy poco profesional.

XHTML soluciona estos problemas añadiendo ciertas normas en la forma de escribir las etiquetas y atributos.

Autoevaluación

La principal diferencia entre HTML y XHTML es:

- ☐ HTML permite usar las etiquetas en mayúsculas y XHTML no.
- ☐ XHTML da lugar a códigos ordenados.
- ◉ **XHTML es compatible con XML.**
- ☐ HTML es permisivo.

4.1.- XHTML: diferencias sintácticas y estructurales con HTML.

El esquema básico del documento, para considerarse conforme a la especificación deberá cumplir las siguientes condiciones:

El elemento raíz del documento debe ser <html>.

El elemento raíz del documento debe indicar el espacio nominal XHTML usando el atributo xmlns. El espacio nominal para XHTML es http://www.w3.org/1999/xhtml

Debe haber una declaración DOCTYPE en el prólogo del documento. El identificador público incluido en la declaración DOCTYPE debe hacer referencia a alguna de las tres DTD definidas por el W3C usando el Identificador Formal Público correspondiente:

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

Restricciones básicas que introduce XHTML respecto a HTML en la sintaxis de sus etiquetas:

Las etiquetas se tienen que cerrar en orden inverso al que se abren, nunca pueden solaparse.

Los nombres de las etiquetas y atributos siempre se escriben en minúsculas.

El valor de los atributos, incluso los numéricos, siempre se encierra entre comillas.

Los atributos en los que el nombre coincide con su valor, no puede darse el valor por entendido, es decir, no se pueden comprimir. Este tipo de atributos no son muy habituales.

Todas las etiquetas deben cerrarse siempre. XHTML permite que en lugar de abrir y cerrar de forma consecutiva la etiqueta (

) se puede utilizar la sintaxis <br/> para indicar que es una etiqueta vacía que se abre y se cierra en ese mismo punto.

Otras restricciones:

Además de las cinco restricciones básicas, XHTML incluye otros cambios más avanzados respecto a HTML, entre ellas:

Antes de acceder al valor de un atributo, se eliminan todos los espacios en blanco que se encuentran antes y después del valor. Además, se eliminan todos los espacios en blanco sobrantes dentro del valor de un atributo.

El código JavaScript debe encerrarse entre unas etiquetas especiales (<![CDATA[y]]>) para evitar que el navegador interprete de forma errónea caracteres como & y <.

Las páginas XHTML deben prescindir del atributo name en su lugar, siempre debe utilizarse el atributo id.

XHTML es necesario separar el formato del contenido. Los párrafos deben separarse consistentemente y las cabeceras h1-h6 sólo deben usarse para destacar los diferentes apartados. Es recomendable dar el formato a los datos por medio del uso de las CSS.

4.2.- Ventajas e inconvenientes de XHTML sobre HTML.

Ventajas:

Compatibilidad parcial con navegadores antiguos: la información se visualiza, aunque sin formato.

Un mismo documento puede adoptar diseños radicalmente distintos en diferentes apartados.

Sencillez a la hora de editar y mantener el código.

Es compatible con los estándares que está desarrollando el W3C como recomendación para futuros agentes de usuario o navegadores.

Los documentos escritos conforme a XHTML 1.0 presentan mejor rendimiento en las actuales herramientas web que aquellos escritos conforme a HTML.

La separación de los contenidos y su presentación hace que los documentos XHTML se adapten mejor a las diferentes plataformas: pantallas de ordenador, pantallas de dispositivos móviles, ...

Como es XML se pueden utilizar fácilmente herramientas creadas para procesar documentos XML genéricos (editores, XSLT, etc.).

Inconvenientes:

Algunos navegadores antiguos no son totalmente compatibles con los estándares, lo que hace que las páginas no siempre se muestren correctamente. Esto cada vez es menos problemático ya que estos navegadores van cayendo en desuso.

Muchas herramientas de diseño web aún no generan código XHTML correcto.

Autoevaluación
El lenguaje XHTML permite:

- [x] **La información contenida en el documento se visualiza aunque el navegador no interprete el código de formato.**
- [x] **Su funcionalidad presenta un rendimiento más alto que el de HTML.**
- [ ] Al ser un lenguaje más novedoso que HTML es compatible con todas las herramientas de diseño que permiten trabajar con éste último.
- [ ] Ninguna respuesta es válida.

5.- Herramientas de diseño web.

Caso práctico
Juan *muestra a* ***María*** *y* ***Félix*** *una primera versión de la página web.*
Tras verla ***Félix*** *tiene la curiosidad de saber si para codificarla hay que utilizar algún software característico o basta con usar un editor de texto plano, como en el caso de XML.*
Juan *le cuenta que puede bastar el bloc de notas, pero que existen varios editores que facilitan la tarea.*

Actualmente hay herramientas que permiten diseñar un sitio web sin necesidad de saber programar HTML. Las más populares:

Macromedia Dreamweaver.
Microsoft Front Page.
Adobe Go live.
NetObjects Fusion.

Amaya es una herramienta de libre distribución creada por el W3C que permite visualizar y editar páginas HTML y XHTML con hojas de estilo CSS, expresiones MathML (*Mathematical Markup Language es un lenguaje de marcas basado en XML, cuyo objetivo es expresar el formalismo matemático de tal modo que pueda ser entendido por distintos sistemas y aplicaciones*) y dibujos SVG (*Scalable Vector Graphics Es un lenguaje basado en XML que permite describir gráficos vectoriales bidimensionales*); además de documentos XML. Está disponible para plataformas Windows, GNU/Linux, Mac OS X, entre otras. La última versión soporta HTML 4.01, XHTML 1.0, XHTML Basic, XHTML 1.1, HTTP 1.1, MathML 2.0, muchas características CSS 2, e incluye soporte para gráficos SVG.

Además de las anteriores, para generar páginas web, es conveniente tener algunas de las siguientes herramientas:

Software de diseño

Macromedia Flash para hacer animaciones, banners o sitios enteros con esta tecnología.

Macromedia Fireworks o *Adobe Illustrator* para diseñar botones, logos, imágenes, etc.

Adobe Photoshop o *Gimp*, para retocar fotografías y trabajar con imágenes.

Recursos: diseño web

My Fonts, es un sitio web que nos vende fuentes que pueden ser utilizadas para la web.

Color Voodoo, tiene interesante información sobre el uso de los colores y su influencia en la web.

moreCrayons, una paleta de colores web seguros (Pues no todos los colores se ven iguales en distintos sistemas operativos).

Yafla Color, permite crear la gama a partir del color que se elija como principal además funciona como conversor entre los modelos de color RGB y HSV.

ColorJack, permite crear una gama a partir de un color y además calcula equivalencias de colores entre RGB, HSV y CMYK.

Kuler, es una aplicación online de Adobe Labs en la que se puede elegir una combinación de colores y compartirla con otros usuarios, que pueden votar sus favoritas. Requiere registro.

Autoevaluación
Todas las herramientas web permiten hacer animaciones y retocar fotografías:

- ☐ Verdadero.
- ☑ **Falso.**

Debes conocer

El siguiente enlace te permite descargar la versión del editor Amaya adecuada para tu sistema operativo

Amaya	http://www.w3.org/Amaya/User/BinDist.html

El siguiente enlace te permite descargar la versión del editor html-kit adecuada para tu sistema operativo

html-kit	http://www.htmlkit.com/

Para saber más

Estos sitios, ofrecen gratuitamente plantillas (templates) de webs que solamente tenemos que adaptar a nuestras necesidades para realizar nuestra página.

Free web templates	http://www.freewebtemplates.com/
GUI stuff	http://www.guistuff.com/

6.- Hojas de estilo o CSS.

Caso práctico

María *muestra su conformidad con la estructura y contenidos de la futura web corporativa que* ***Juan*** *ha realizado, aunque le gustaría probar otras gamas de colores y tipos de fuente.*

Félix *plantea que, quizás, resulte un trabajo demasiado laborioso ya que supondría modificar todos los ficheros de la web.*

Juan*, sonriendo, comenta que de hecho no es exactamente así. En realidad los datos están separados de sus formatos.*

CSS (Cascading Style Sheets) permite a los desarrolladores Web controlar el estilo y el formato de múltiples páginas Web al mismo tiempo.

Antes del uso de CSS, los diseñadores de páginas web debían definir el aspecto de cada elemento dentro de las etiquetas HTML de la página. El principal problema de esta forma de definir el aspecto de los elementos es que habría que definir el formato de cada uno de los elementos que formen la página, lo cual hace que sea muy difícil de actualizar.

CSS permite separar los contenidos de la página y su aspecto. Para ello se define en una zona reservada el formato de cada uno de los elementos de la web. Cualquier cambio en el estilo marcado para un elemento en la CSS afectará a todas las páginas vinculadas a ella en las que aparezca ese elemento. Las hojas de estilo están compuestas por una o más reglas de estilo aplicadas a un documento HTML o XML.

Al crear una página web, se utiliza en primer lugar el lenguaje HTML/XHTML para marcar los contenidos, es decir, para designar la función de cada elemento dentro de la página: párrafo, cabecera, texto destacado, etc. Una vez creados los contenidos, se utiliza el lenguaje CSS para definir el formato de cada elemento.

CSS obliga a crear documentos semánticos HTML/XHTML, mejora la accesibilidad del documento, reduce la complejidad de su mantenimiento y permite visualizar el mismo documento en infinidad de dispositivos diferentes.

Las hojas de estilos aparecieron poco después que el lenguaje de etiquetas SGML, alrededor del año 1970. Desde la creación de SGML, se observó la necesidad de definir un mecanismo que permitiera aplicar estilos a los documentos electrónicos. La guerra de navegadores y la falta de un estándar para la definición de los estilos dificultaban la creación de documentos que tuvieran igual apariencia en distintos navegadores.

El organismo W3C propuso la creación de un lenguaje de hojas de estilos específico para el lenguaje HTML.

En 1995, el W3C añadió a su grupo de trabajo de HTML el desarrollo y estandarización de CSS.

CSS 1, se publicó en 1996, es la primera recomendación oficial.

CSS 2, publicada en 1998, es la segunda recomendación oficial.

CSS 3, continúa en desarrollo desde 1998.

Actualmente se utiliza la versión CSS 2.1, actualizada por última vez el 19 de julio de 2007.

El diseño web siempre está limitado por las posibilidades de los navegadores que utilizan los usuarios para acceder a sus páginas. Por este motivo es imprescindible conocer el soporte de CSS en cada uno de los navegadores más utilizados del mercado.

Autoevaluación
Las hojas de estilo en casacada permiten:

- [x] **Definir formatos que se aplican sobre varias páginas web de un sitio.**
- [ ] Separar el formato de la estructura de una página web.
- [ ] Estructurar el contenido de la página web.
- [ ] Ninguna respuesta es válida.

6.1.- Soporte de CSS en los navegadores.

El soporte de CSS de un navegador viene determinado por el motor del mismo ya que es éste el encargado de interpretar el CSS.

La siguiente tabla muestra el soporte de **CSS 1**, **CSS 2.1** y **CSS 3** de los cinco navegadores más utilizados en la actualidad:

Navegador	Motor	CSS 1	CSS 2.1	CSS 3
Safari	WebKit	Completo	Casi completo	Parcial
Opera	Presto	Completo	Casi completo	Parcial
Firefox	Gecko	Completo	Casi completo	Parcial
Google Chrome	WebKit	Completo	Casi completo	Parcial
Internet Explorer	Trident	Completo, desde la versión 6.0	Completo, desde la versión 8.0	Prácticamente nulo

6.2.- Cómo incluir CSS en un documento HTML o XHTML.

Existen tres opciones para incluir CSS en un documento HTML o XHTML:

Definir CSS en un archivo externo.

En este caso, todos los estilos CSS se incluyen en uno, o varios, archivos de texto plano, cuya extensión es .css, que las páginas HTML enlazan mediante el elemento <link> de la cabecera del fichero HTML.

Puesto que una página web puede tener asociados varios ficheros CSS es recomendable agrupar estos últimos en un directorio.

El navegador descarga los archivos CSS externos, además de la página web asociada a ellos, y aplica los estilos a los contenidos de la página antes de mostrar sus contenidos.

Esta es la forma de incluir CSS en las páginas HTML más utilizada. La principal ventaja es que se puede incluir un mismo archivo CSS en multitud de páginas HTML, por lo que se garantiza la aplicación homogénea de los mismos estilos a todas las páginas que forman un sitio web.

Además, el mantenimiento del sitio web se simplifica al máximo, ya que el cambio en un solo archivo CSS permite variar de forma instantánea los estilos de todas las páginas HTML asociadas.

Puede hacerse de dos modos diferentes:

Mediante enlaces.

Importando el fichero CSS.

Incluir CSS en el documento HTML.

Este método se emplea cuando se definen pocos estilos o cuando se quieren incluir estilos específicos en una determinada página HTML que completen los estilos globales de todas las páginas del sitio web.

Tiene el inconveniente de que para modificar los estilos definidos, es necesario modificar todas las páginas que incluyen el estilo que se va a cambiar.

Incluir CSS en los elementos HTML.

El último método para incluir estilos CSS en documentos HTML es el peor y el menos utilizado, ya que para modificar un formato hay que cambiar todos los elementos que estén asociados a él.

Solamente se utiliza en determinadas situaciones en las que se debe incluir un estilo muy específico para un solo elemento concreto.

Autoevaluación
El mejor modo de aplicar formatos a una página web es:

- ☐ Definiendo los formatos directamente a través de los atributos de los elementos HTML.
- ☐ Incluyendo el formato CSS en los elementos de HTML.
- ☐ Definiendo los estilos en la cabecera del documento HTML.
- ☑ **Definiendo un fichero CSS externo.**

6.2.1.- Definir CSS en un archivo externo enlazado.

Para realizar una página web usando un archivo CSS externo, se deben seguir los tres pasos siguientes:

Se crea un archivo de texto plano con las definiciones de los formatos.

Dicho archivo de texto se guarda con extensión .css

Se enlaza el archivo CSS externo mediante la etiqueta <link> en la cabecera de la página web.

El elemento <link> puede tener definidos cuatro atributos cuando se enlaza un archivo CSS:

rel, indica el tipo de relación que tiene el archivo enlazado y la página HTML. Para los archivos CSS, siempre se utiliza el valor stylesheet

type, indica el tipo de recurso enlazado. Para los archivos CSS su valor siempre es text/css

href, indica la URL del archivo CSS que contiene los estilos. Puede ser relativa o absoluta y puede referenciar a un recurso interno o externo al sitio web.

media, indica el medio en el que se van a aplicar los estilos del archivo CSS.

Un ejemplo del uso de archivos externos CSS enlazados para la construcción de páginas web es:

El archivo formatos.css contiene:

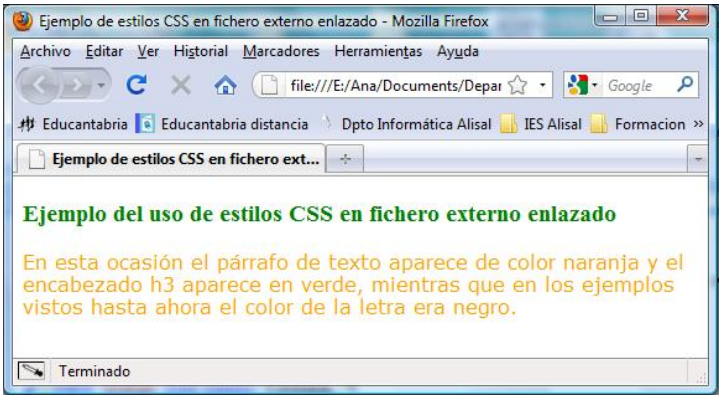
```
h3 { color: green;}
p { color: orange; font-family: Verdana; }
```

```
<!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>Ejemplo de estilos CSS en fichero externo enlazado</title>
        <link rel="stylesheet" type="text/css" href="formatos.css" />
    </head>

    <body>
        <h3>Ejemplo del uso de estilos CSS en fichero externo enlazado</h3>
        <p>En esta ocasi&oacute;n el p&aacute;rrafo de texto aparece de color naranja y el encabezado h3
aparece en verde, mientras que en los ejemplos vistos hasta ahora el color de la letra era negro.</p>
    </body>
</html>
```

Al publicarlo en un navegador, por ejemplo en el Firefox, tendríamos:



6.2.2.- Definir CSS en un archivo externo importado.

Se puede obtener el mismo resultado anterior utilizando el elemento <`style`> en lugar de <`link`>.

En este caso, se usa una regla de tipo `@import` seguida de una cadena de texto encerrada con comillas simples o dobles que se corresponde con la URL del archivo CSS, o de url() conteniendo dicha cadena entre los paréntesis. Las siguientes reglas `@import` son equivalentes para un fichero formatos.css que está en el directorio css.:

```
@import '/css/formatos.css';
@import "/css/formatos.css"
@import url('/css/formatos.css');
@import url("/css/formatos.css");
```

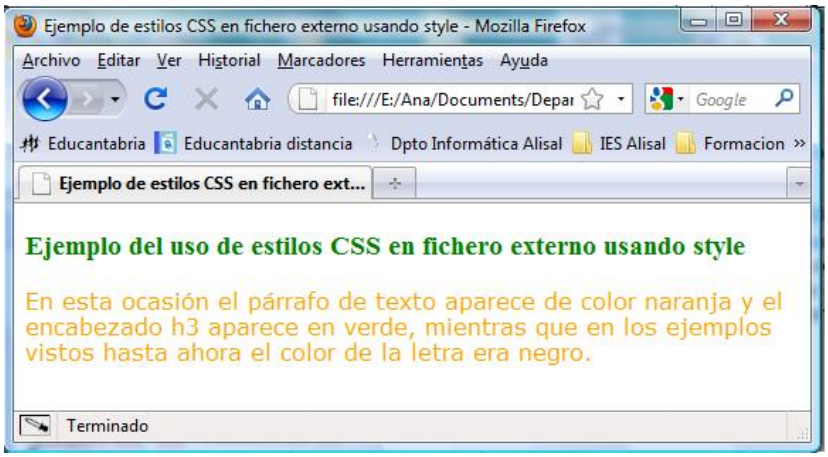
El ejemplo anterior quedaría:

```
<!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>Ejemplo de estilos CSS en fichero externo usando style</title>
        <style type="text/css">
            @import 'formatos.css';
        </style>
    </head>

    <body>
        <h3>Ejemplo del uso de estilos CSS en fichero externo usando style</h3>
        <p>En esta ocasi&oacuten el p&aacute;rrafo de texto aparece de color naranja y el encabezado h3 aparece
en verde, mientras que en los ejemplos vistos hasta ahora el color de la letra era negro.</p>
    </body>
</html>
```

Al publicarlo en un navegador, por ejemplo en el Firefox, tendríamos:



6.2.3.- Definir CSS en el documento HTML.

En este caso los formatos de los elementos se definen en la cabecera del documento HTML, dentro del elemento <style>.

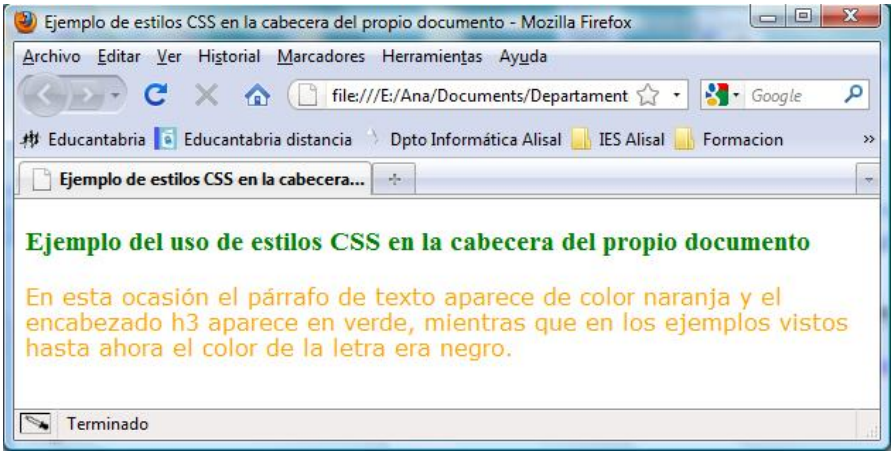
Un ejemplo de un documento XHTML en el que se utiliza este método para incluir formatos es:

```
<!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>Ejemplo de estilos CSS en la cabecera del propio documento</title>
        <style type="text/css">
            h3 { color: green; font-family: Times; }
            p { color: orange; font-family: Verdana; }
        </style>
    </head>

    <body>
        <h3>Ejemplo del uso de estilos CSS en la cabecera del propio documento</h3>
        <p>En esta ocasi&oacute;n el p&aacute;rrafo de texto aparece de color naranja y el encabezado h3
aparece en verde, mientras que en los ejemplos vistos hasta ahora el color de la letra era negro.</p>
    </body>
</html>
```

Al publicarlo en un navegador, por ejemplo en el Firefox, tendríamos:



6.2.4.- Incluir CSS en los elementos HTML.

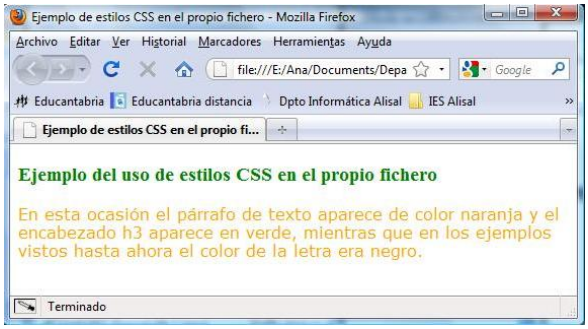
Un ejemplo de un documento XHTML en el que se utiliza este método para incluir formatos es:

```
<!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>Ejemplo de estilos CSS en el propio fichero</title>
    </head>

    <body>
        <h3 style="color: green;">Ejemplo del uso de estilos CSS en el propio fichero</h3>
        <p style="color: orange; font-family: Verdana;">En esta ocasi&oacute;n el p&aacute;rrafo de texto
aparece de color naranja y el encabezado h3 aparece en verde, mientras que en los ejemplos vistos hasta ahora
el color de la letra era negro.</p>
    </body>
</html>
```

Al publicarlo en un navegador, por ejemplo en el Firefox, tendríamos:



6.3.- Sintaxis de las reglas de estilo.

Cada uno de los estilos que componen una hoja de estilos CSS se denomina regla. Cada regla se forma por:

Selector: indica el elemento o elementos HTML a los que se aplica la regla CSS

Llave de apertura, `{`

Declaración: especifica los estilos que se aplican a los elementos.

Propiedad: permite modificar el aspecto de un atributo del elemento.

Valor: indica el nuevo valor del atributo modificado en el elemento.

Llave de cierre, `}`.

Ejemplo: `p{ color : blue; }`

En este caso el selector es "`p`", la declaración es: "`color : blue`" y, dentro de ésta, podemos diferenciar la propiedad "`color`" y el valor "`blue`".

Un archivo CSS puede contener infinitas reglas CSS, cada regla puede contener varios selectores y cada declaración puede estar formada por diferentes declaraciones.

Autoevaluación
El elemento HTML sobre el que se aplica un estilo se especifica:

- ☐ En etiquetas, es decir, entre < y >.
- ☐ Entre paréntesis.
- ☐ Entre llaves.
- ◉ **No va encerrado entre signos.**

6.4.- Atributos principales.

En los siguientes subapartados vamos a ver los atributos principales que se usan en CSS como son:

- ✓ **Atributos de color y fondo.**
- ✓ **Atributos de fuente.**
- ✓ **Atributos de texto.**
- ✓ **Atributos de caja.**
- ✓ **Atributos de clasificación.**
- ✓ Pasemos a verlos detenidamente.

6.4.1.- Atributos de color y fondo.

Los atributos de color y fondo son los que enumeramos a continuación:

Elemento	Descripción
color	Indica el color del texto. Lo admiten casi todas las etiquetas de HTML. El valor de este atributo es un color, con su nombre o su valor RGB.
background-color	Indica el color de fondo del elemento. El valor de este atributo es un color, con su nombre o su valor RGB.
background-image	Permite colocar una imagen de fondo del elemento. El valor que toma es el nombre de la imagen con su camino relativo o absoluto
background-repeat	Indica si ha de repetirse la imagen de fondo y, en ese caso, si debe ser horizontal o verticalmente. Los valores que puede tomar son: repeat-x, repeat-y o no-repeat.
background-attachment	Especifica si la imagen ha de permanecer fija o realizar un scroll. Los valores que pueden tomar son: scroll o fixed.
background-position	Es una medida, porcentaje o el posicionamiento vertical u horizontal con los valores establecidos que sirve para posicionar una imagen. Los valores que puede tomar son: porcentaje, tamaño, o [top, center, bottom] [left, center, rigth]
background	Establece en un solo paso cualquiera de las propiedades de background anteriores. Los valores que puede tomar son: background-color, background-image, background-repeat, background-attachment, background-position.

Dado que no todos los nombres de colores son admitidos en el estándar, es aconsejable utilizar el valor RGB.

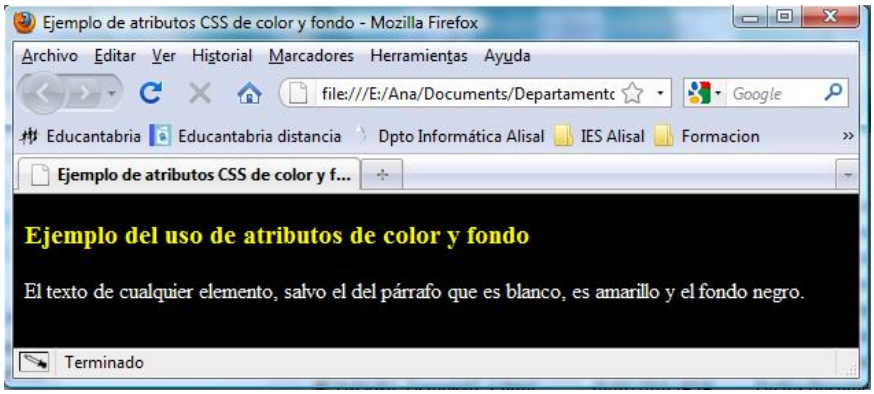
Un ejemplo de un documento XHTML en el que se utiliza este método para incluir formatos es:

```
<!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>Ejemplo de atributos CSS de color y fondo</title>
        <style type="text/css">
            body { background-color: black; color:yellow; }
            p { color: #ffffff;}
        </style>
    </head>

    <body>
        <h3>Ejemplo del uso de atributos de color y fondo</h3>
        <p>El texto de cualquier elemento, salvo el del p&aacute;rrafo que es blanco, es amarillo y el fondo
negro.</p>
    </body>
</html>
```

Al publicarlo en un navegador, por ejemplo en el Firefox, tendríamos:



6.4.2.- Atributos de fuente.

En este apartado vamos a ver los distintos atributos que podemos utilizar referentes a las fuentes de nuestro docuemtno y que son:

Elemento	Descripción
font-size	Indica el tamaño de la fuente. Puede ser un tamaño absoluto, relativo o en porcentaje. Toma valores de unidades de CSS
font-family	Establece la familia a la que pertenece la fuente. Si el nombre de una fuente tiene espacios se utilizan comillas para que se entienda bien. El valor es el nombre de la familia fuente.
font-weight	Define el grosor de los caracteres. Los valores que puede tomar son: normal, bold, bolder, lighter, 100, 200, 300, 400, 500, 600, 700, 800 o 900
font-style	Determina si la fuente es normal o cursiva. El estilo oblique es similar al cursiva. Los valores posibles son: normal, italic, oblique.
font-variant	Determina si la fuente es normal o mayúsculas pequeñas. Los valores que puede tomar son: normal o small-caps
line-height	El alto de una línea y por tanto, el espaciado entre líneas. Es una de esas características que no se podían modificar utilizando HTML.
font	Permite establecer todas las propiedades anteriores en el orden que se indica a continuación: font-style, font-variant, font-weight, font-size[line-height], font family. Los valores han de estar separados por espacios. No es obligatorio el uso de todos los valores.

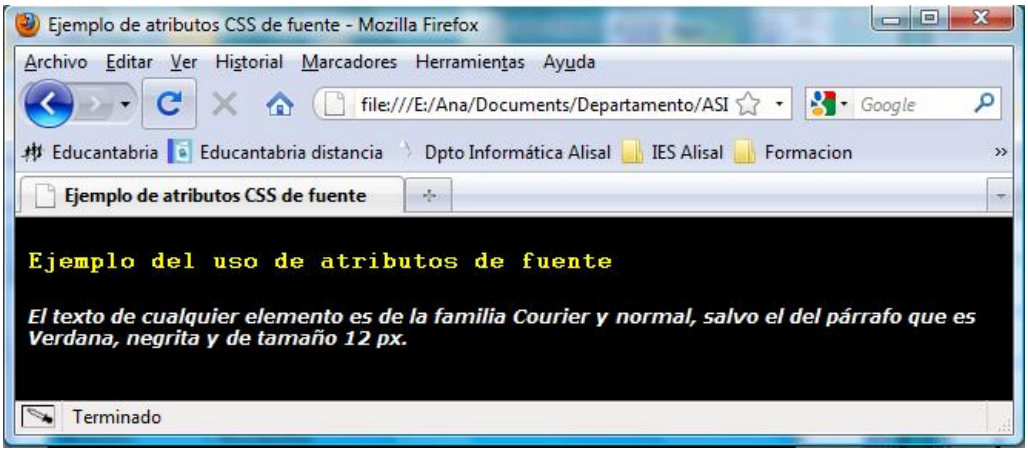
Un ejemplo de un documento XHTML en el que se utiliza este método para incluir formatos es:

```
<!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>Ejemplo de atributos CSS de fuente</title>
        <style type="text/css">
            body { background-color: black; color:yellow; font-family: courier }
            p { color: #ffffff; font:italic 900 12px Verdana;  }
        </style>
    </head>

    <body>
        <h3>Ejemplo del uso de atributos de fuente</h3>
        <p>El texto de cualquier elemento es de la familia Courier y normal, salvo el del p&aacute;rrafo que es
Verdana, negrita y de tamaño 12 px.</p>
    </body>
</html>
```

Al publicarlo en un navegador, por ejemplo en el Firefox, tendríamos:



6.4.3.- Atributos de texto.

En el apartado anterior vimos los atributos relacionados con las fuentes y en este vamos a ver los relacionados con el texto en sí y son los siguientes:

Elemento	Descripción
text-decoration	Establece si el texto está subrayado, sobrerayado o tachado. los valores que puede tomar son: none, underline, overline, line-through o blink
text-align	Indica la alineación del texto. Aunque las hojas de estilo permiten el justificado de texto no funciona en todos los sistemas. Los valores que puede tomar son: left, right, center o justify
text-indent	Determina la tabulación del texto. Los valores que toma son una longitud, en unidades CSS, o un porcentaje de la establecida.
text-transform	Nos permite transformar el texto, haciendo que tenga la primera letra en mayúsculas de todas las palabras, todo en mayúsculas o minúsculas. Los valores que puede tomar son: capitalize, uppercase, lowercase o none
word-spacing	Determina el espaciado entre las palabras. Los valores que puede tomar es un tamaño.
letter-spacing	Determina el espaciado entre letras. Los valores que puede tomar es un tamaño.
vertical-align	Establece la alineación vertical del texto. Sus valores posibles son: baseline, sub, super, top, text-top, middle, bottom, text-bottom o un porcentaje.
line-height	Altura de la línea. Puede establecerse mediante un tamaño o un porcentaje

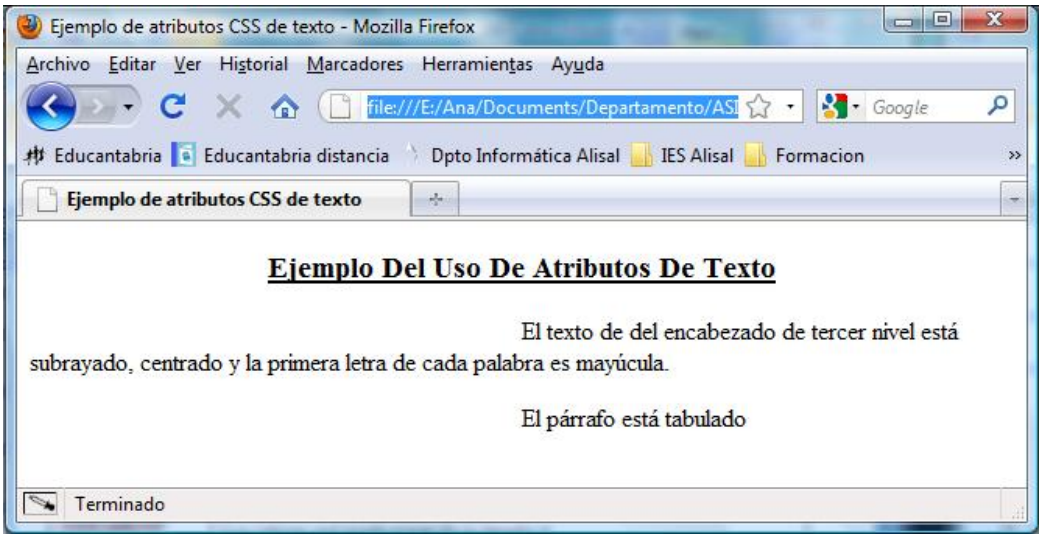
Un ejemplo de un documento XHTML en el que se utiliza este método para incluir formatos es:

```
<!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>Ejemplo de atributos CSS de texto</title>
        <style type="text/css">
            h3 { text-decoration:underline; text-align: center; text-transform: capitalize }
            p { text-indent: 50%;  }
        </style>
    </head>

    <body>
        <h3>Ejemplo del uso de atributos de texto</h3>
        <p>El texto de del encabezado de tercer nivel est&aacute; subrayado, centrado y la primera letra de
cada palabra es may&uacute;cula.</p>
        <p>El p&aacute;rrafo está tabulado</p>
    </body>
</html>
```

Al publicarlo en un navegador, por ejemplo en el Firefox, tendríamos:



6.4.4.- Atributos de caja.

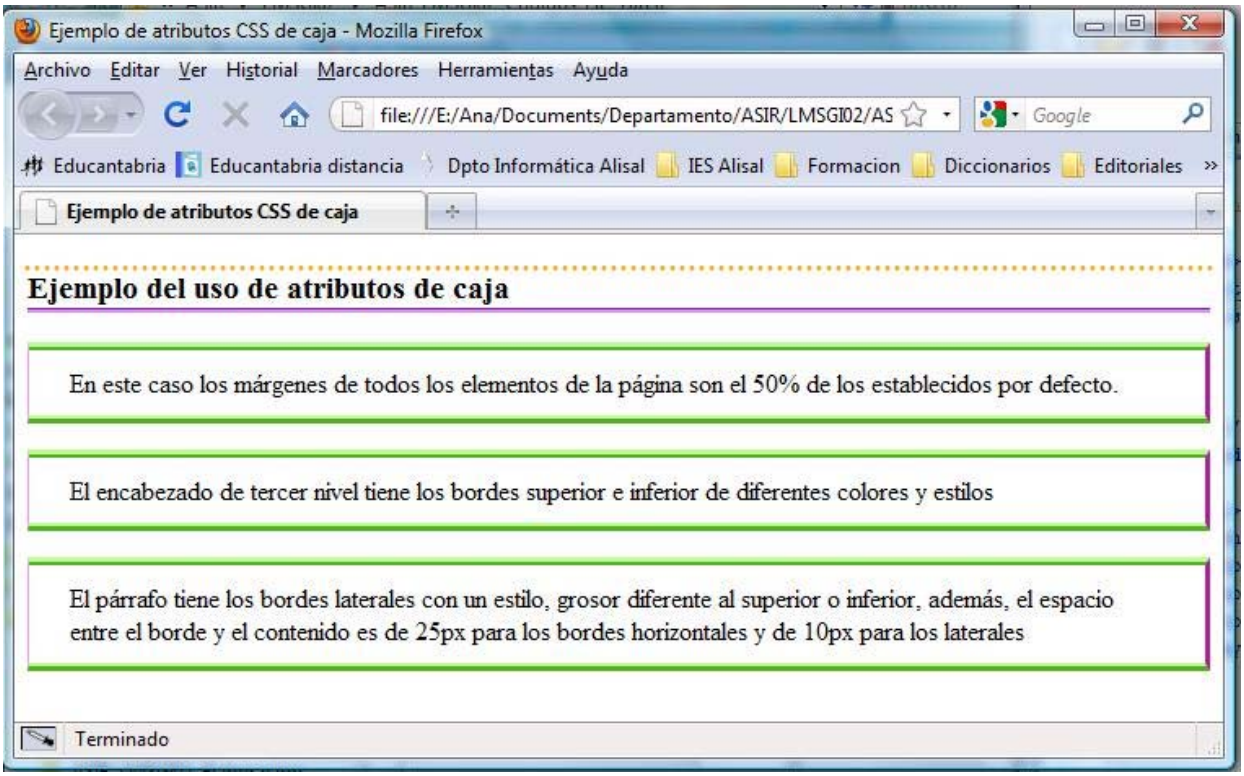
Ahora vamos a ver otros atributos muy importantes y que utilizaremos muy a menudo y que no son ni más ni menos que los atributos de caja:

Elemento	Descripción
margin-left	Indica el tamaño del margen izquierdo. Puede usarse una longitud, en unidades CSS, o un porcentaje.
margin-rigth	Indica el tamaño del margen derecho. Puede usarse una longitud, en unidades CSS, o un porcentaje.
margin-top	Indica el tamaño del margen superior. Puede usarse una longitud, en unidades CSS, o un porcentaje.
margin-bottom	Indica el tamaño del margen inferior. Puede usarse una longitud, en unidades CSS, o un porcentaje.
margin	Permite establecer los márgenes de una vez. Hay que seguir el orden: superior, derecho, inferior e izquierdo.
padding-left	Indica el espacio izquierdo entre el borde y el contenido. Puede usarse una longitud, en unidades CSS, o un porcentaje.
padding -rigth	Indica el espacio derecho entre el borde y el contenido. Puede usarse una longitud, en unidades CSS, o un porcentaje.
padding -top	Indica el espacio superior entre el borde y el contenido. Puede usarse una longitud, en unidades CSS, o un porcentaje.
padding -bottom	Indica el espacio inferior entre el borde y el contenido. Puede usarse una longitud, en unidades CSS, o un porcentaje.
padding	Establece el espacio entre los bordes y el contenido de una sola vez. Hay que respetar el orden superior, derecho, inferior e izquierdo.
border-left-color	Establece el color del borde izquierdo del elemento. Su valor es un color RGB o el nombre del color.
border-rigth-color	Establece el color del borde derecho del elemento. Su valor es un color RGB o el nombre del color.
border-top-color	Establece el color del borde superior del elemento. Su valor es un color RGB o el nombre del color.
border-bottom-color	Establece el color del borde inferior del elemento. Su valor es un color RGB o el nombre del color.
border-color	Establece el color de los bordes del elemento de una sola vez. Hay que seguir el orden superior, derecho, inferior e izquierdo. Su valor es un color RGB o el nombre del color.

border-style	Establece el estilo del borde, los valores significan: none=ningun borde, dotted=punteado (no funciona siempre), solid=solido, double=doble borde, los valores groove, ridge, inset y outset son bordes con varios efectos 3D.
border-left-width	Grosor del borde izquierdo. Sus valores posibles son: thin, médium, thick o un tamaño.
border-rigth-width	Grosor del borde derecho. Sus valores posibles son: thin, médium, thick o un tamaño.
border-top-width	Grosor del borde superior. Sus valores posibles son: thin, médium, thick o un tamaño.
border-bottom-width	Grosor del borde inferior. Sus valores posibles son: thin, médium, thick o un tamaño.
border-width	Establece el tamaño de los bordes del elemento al que lo aplicamos. Hay que seguir el orden superior, derecho, inferior, izquierdo.
width	Establece el ancho del contenido del elemento. El valor es un porcentaje o un tamaño.
height	Establece la altura del contenido del elemento. El valor es un porcentaje o un tamaño.
float	Sirve para alinear un elemento a la izquierda o la derecha haciendo que el texto se agrupe alrededor de dicho elemento. Toma los valores none, left o right
clear	Establece si un elemento tiene a su altura imágenes u otros elementos alineados a la derecha o la izquierda. Sus valores posibles son: none, left, right o both.

Para que practiques todo lo aprendido te recomiendo que intentes hacer el ejercicio que se propone a continuación, antes de ver su solución. ¡Ánimo!

¿Cuál sería un posible código XHTML para el documento de la imagen que se muestra a continuación?



```
<!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>Ejemplo de atributos CSS de caja</title>
        <style type="text/css">
            body{ margin: 50%, 50%, 50%, 50%; }
            h3 { border-top-color: orange; border-bottom-color: #b428ff; border-top-style: dotted; border-
bottom-style: groove;   }
            p { border-color: #73ff28 #ff28df #73ff28 #ff28df;  border-style: ridge outset ridge outset; border-
width: thick medium thick thin;  padding: 10px 25px 10px 25px;}
        </style>
    </head>

    <body>
        <h3>Ejemplo del uso de atributos de caja</h3>
        <p>En este caso los m&aacute;rgenes de todos los elementos de la p&aacute;gina son el 50% de los
establecidos por defecto.</p>
        <p>El encabezado de tercer nivel tiene los bordes superior e inferior de diferentes colores y estilos </
p>
        <p>El p&aacute;rrafo tiene los bordes laterales con un estilo, grosor diferente al superior o inferior,
adem&aacute;s, el espacio entre el borde y el contenido es de 25px para los bordes horizontales y de 10px para
los laterales </p>
    </body>
</html>
```

Autoevaluación
Para modificar el tamaño del borde izquierdo de una caja o utilizar el atributo:

- ☐ border-left-width
- ☐ border-width
- ◉ **Ambas.**
- ☐ Ninguna.

6.4.5.- Atributos de clasificación.

En este apartado vamos a ver otros atributos que hemos etiquetado como atributos de clasificación y que son los siguientes:

Elemento	Descripción
display	Determina si el elemento es de bloque, línea, lista o ninguno de ellos. Los valores que puede tomar son: block, inline, list-item o none.
white-space	Indica el modo en que se ha de gestionar los espacios en blanco que hay en el elemento, es decir, si se mantienen todos los existentes tal y como estén en el documento o si se anulan a uno las secuencias de blancos, es el valor por defecto y el de la opción normal. Valores que puede tomar son: pre, nowrap, normal.
list-style-type	Indica cual es el símbolo que se utiliza como marcador en las listas. Valores que puede tomar son: disc, circle, square, decimal, lower-roman, upper-roman, lower-alpha, upper-alpha, none.
list-style-image	Permite utilizar el uso de una imagen como marcador en una lista. El valor que toma es la ruta del fichero imagen
list-style-position	Determinan la posición del marcador en una lista. Puede tomar los valores: outside o inside.
list-style	Permite establecer de una única vez todas las características de una lista. Hay que seguir el orden siguiente: list-style-type, list-style-position y list-style-image.

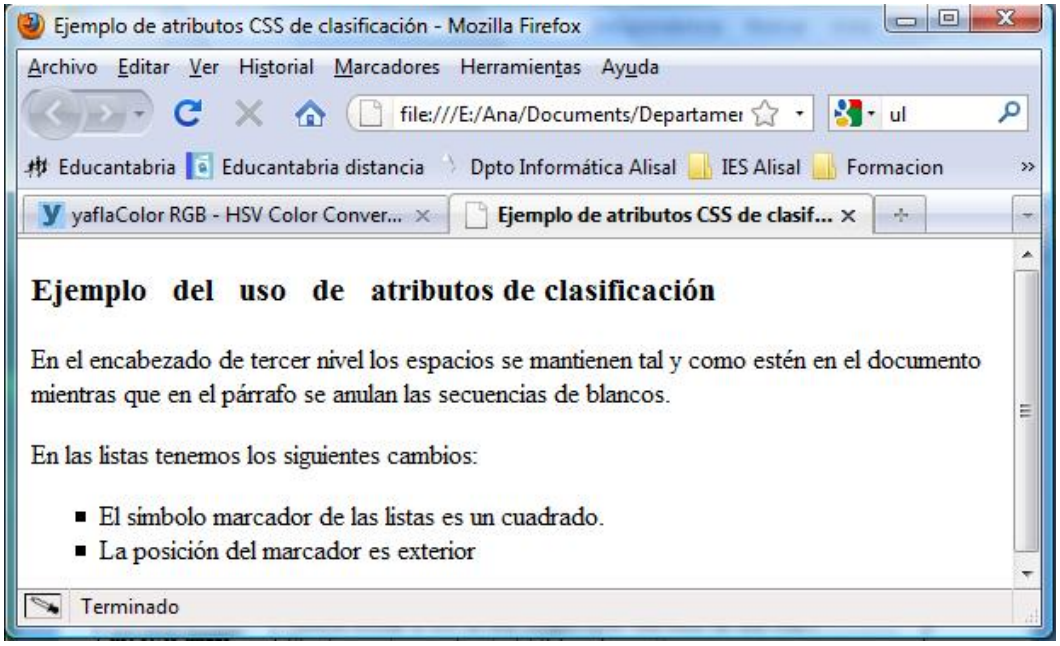
Un ejemplo de un documento XHTML en el que se utiliza este método para incluir formatos es:

```
<!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>Ejemplo de atributos CSS de clasificaci&oacute;n</title>
        <style type="text/css">
            h3 { white-space:pre;    }
            p { white-space:normal;    }
            ul { list-style:square; list-style-position: outside;}
        </style>
    </head>

    <body>
        <h3>Ejemplo   del   uso   de   atributos de clasificaci&oacute;n</h3>
        <p>En          el       encabezado de tercer nivel los espacios se mantienen tal y como est&eacute;n
en el documento mientras que en el p&aacute;rrafo se anulan las secuencias de blancos.</p>
         <p>   En las listas tenemos los siguientes cambios:</p>
        <ul>
            <li>El s&iacute;mbolo marcador de las listas es un cuadrado.</li>
            <li>La posici&oacute;n del marcador es exterior</li>
        </ul>
    </body>
</html>
```

Al publicarlo en un navegador, por ejemplo en el Firefox, tendríamos:



6.5.- CSS de posicionamiento.

Es un añadido a CSS que permite determinar el modo en que se ha de colocar un determinado elemento. Las propiedades definidas en CSS-P son las siguientes:

Elemento	Descripción
clip	Permite seleccionar una zona. Los valores que puede tomar son: shape o auto.
height	Permite establecer la altura de un elemento. Los valores que puede tomar son: auto o un tamaño.
width	Permite establecer la anchura de un elemento. Los valores que puede tomar son: auto o un tamaño o porcentaje.
visibility	Indica si el elemento sobre el que actúa será visible o no. Los valores que puede tomar son:
left	Indica la posición del lado izquierdo del elemento. Los valores que puede tomar son: auto o un tamaño o porcentaje.
top	Indica la posición del lado superior del elemento. Los valores que puede tomar son: auto

	o un tamaño o porcentaje.
overflow	Indica si el elemento será visible o no en caso de superar los límites del contenedor. Los valores que pueden tomar son: visible, hidden, scroll o auto.
position	Determinan si el posicionamiento de un elemento es absoluto, relativo o estático. Los valores que puede tomar son: absolute, relative o static.
z-index	Define la posición del elemento en el tercer eje de coordenadas, permitiendo superponer unos elementos sobre otros como si fueran capas.

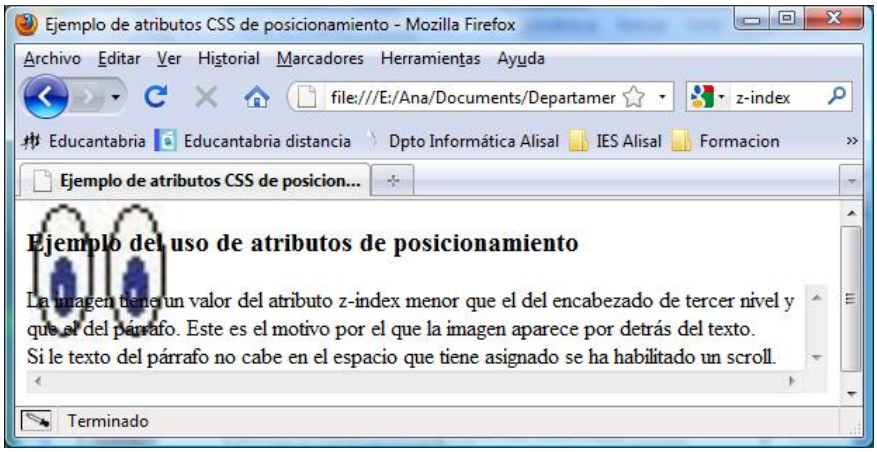
Un ejemplo de un documento XHTML en el que se utiliza este método para incluir formatos es:

```
<!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>Ejemplo de atributos CSS de posicionamiento</title>
        <style type="text/css">
            img{ position:absolute; left: 10px; top: 0px; z-index:-1;}
            p{overflow:scroll;}
        </style>
    </head>

    <body>
        <h3>Ejemplo del uso de atributos de posicionamiento</h3>
        <img alt="Esto es una imagen" src="ojos.jpg" width="100" height="100"></img>
        <p> La imagen tiene un valor del atributo z-index menor que el del encabezado de tercer nivel y que el
del p&aacute;rrafo. Este es el motivo por el que la imagen aparece por detr&aacute;s del texto. <br><br/>
            Si le texto del p&aacute;rrafo no cabe en el espacio que tiene asignado se ha habilitado un
scroll.</p>
    </body>
</html>
```

Al publicarlo en un navegador, por ejemplo en el Firefox, tendríamos:



6.6.- Unidades de tamaño.

Las distintas unidades que podemos utilizar para indicar tamaños son las siguientes:

Relativas

Element (em): Expresa el tamaño relativo al tamaño de la fuente utilizada.
X-height (ex): Expresa el tamaño relativo al de la letra "x".
Pixel (px): Expresa el tamaño relativo a la resolución del monitor.

Absolutas

Milímetros (mm).
Centímetros (cm): Cada centímetro son 10 mm.
Pulgadas (in): Cada pulgada equivale a 2,54 cm.
Puntos (pt): Cada punto son 1/72 in.
Picas (pc): Cada pica son 12 pt.

Autoevaluación
Las unidades relativas especifican el tamaño en relación al tamaño de una letra determinada que escoge el programador:

○ Verdadero. ◉ **Falso.**

6.7.- Definición y uso de clases.

Cuando las reglas de estilos se asocian a un documento HTML utilizando un fichero externo o incluyéndolas en el contenido de la etiqueta STYLE en la cabecera del documento, pueden definirse estilos y asociarlos a determinados elementos del documento.

Para definir una clase hay que usar la sintaxis siguiente:

```
.clase_azul{color:blue}
```

Para asociar un elemento HTML a una clase habrá que usar el atributo CLASS al usar dicho elemento en el documento HTML del siguiente modo:

```
<h3 class="clase_azul">El encabezado de tercer nivel es ahora azul</h3>
```

Para restringir la clase a un determinado elemento basta poner el elemento delante del punto al definir la regla. Por ejemplo, para restringir el uso de la clase a párrafos tendremos:

```
p.clase_azul{color:blue}
```

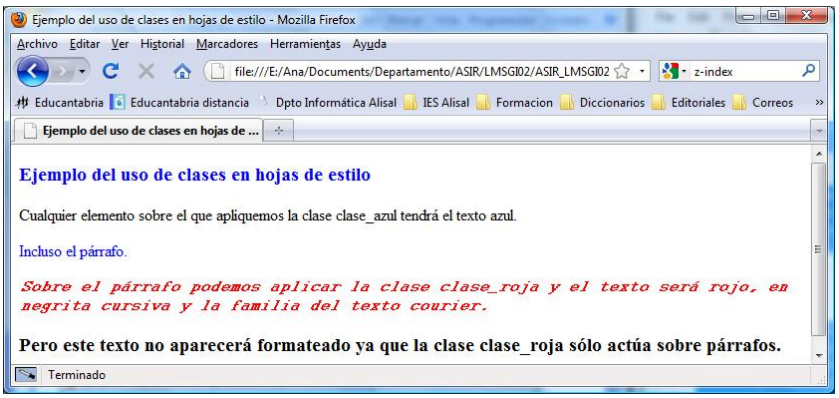
Un ejemplo de un documento XHTML en el que se utiliza este método para incluir formatos es:

```
<!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>Ejemplo del uso de clases en hojas de estilo</title>
        <style type="text/css"> .clase_azul{color:blue} p.clase_roja{color:#ff0000; font-style:italic; font-
weight:bolder; font-family:courier;} </style>
    </head>

    <body>
        <h3 class="clase_azul">Ejemplo del uso de clases en hojas de estilo</h3>
        <p>Cualquier elemento sobre el que apliquemos la clase clase_azul tendr&aacute; el texto azul.</p>
        <p class="clase_azul"> Incluso el p&aacute;rrafo.</p>
        <p class="clase_roja">Sobre el p&aacute;rrafo podemos aplicar la clase clase_roja y el texto ser&aacute;
rojo, en negrita cursiva y la familia del texto courier.</p>
        <h3 class="clase_roja"> Pero este texto no aparecer&aacute formateado ya que la clase clase_roja
s&oacute;lo act&uacute;a sobre p&aacute;rrafos.</h3>
    </body>
</html>
```

Al publicarlo en un navegador, por ejemplo en el Firefox, tendríamos:



Autoevaluación
Cuál de los siguientes valores del atributo color no es válido:

○ rgb(33,33,0). ○ red. ○ #f06. ◉ **#ff06.**

ÍNDICE

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

Caso práctico
Desde que **María** y **Félix** se asociaron para formar la asesoría legal y empresarial, la empresa ha ido creciendo. Después de lograr compartir la información de sus negocios individuales con la ayuda de **Juan**, su amigo Técnico Superior en Administración de Aplicaciones Informáticas, dueño de una empresa de consultoría informática y **Marina**, la trabajadora de esta empresa, y publicar la página web de la empresa, el número de clientes de su negocio aumentó considerablemente, tal y como esperaban.
Con motivo de la cercanía de las fiestas navideñas, **María** y **Félix** invitan a cenar a todos los colaboradores y trabajadores de la empresa, entre ellos se encuentran **Juan** y **Marina**, cuya empresa se ha convertido en la encargada de los asuntos informáticos de la asesoría. Haciendo un repaso de los logros conseguidos con las mejoras informáticas, bromean preguntando a **Juan** cuál es la siguiente tecnología a utilizar para lograr mejorar los servicios que ofrecen a sus clientes. **Juan,** entre risas, les sorprende con su respuesta, comentando la posibilidad de utilizar la web de la empresa para que los clientes puedan informarse de noticias de actualidad que afecten a sus negocios. Esta propuesta les resulta muy atractiva a **María** y **Félix**, que se interesan por el modo de llevarla a cabo.

Sindicación de contenidos

Caso práctico
Juan *les dice que basta con aplicar los lenguajes de marcas a la redifusión de contenidos web para lograr el nuevo servicio.* ***María*** *quiere que le explique con un poco más de detalle en qué consiste la redifusión de contenidos web.*
Juan *comenta que esta tecnología permite utilizar contenidos que ya existen en otras web y ofrecerlos como servicios a través de la propia Web. Siempre, cumpliendo las licencias de las normas de uso de esos contenidos o, si es el caso, respetando las condiciones del contrato que regula los derechos de ese contenido.*

En el mercado televisivo, existen muchas series emitidas por varias cadenas de televisión que, cuando fueron creadas, fueron compradas con exclusividad por una cadena concreta, pero con el paso del tiempo han sido vendidas y distribuidas a otras cadenas para su redifusión. Es un claro ejemplo de la redifusión o sindicación de contenidos, en este caso televisivo.

Desde el punto de vista Web, la sindicación de contenidos permite a un sitio utilizar los servicios o contenidos ofertados por otra web diferente. Esos servicios junto con los metadatos que tiene asociados en el sitio original, forman los feed o canales de contenidos.

La redifusión de contenidos web suele realizarse bajo una licencia de normas de uso en lugar de mediar un contrato para regular los derechos de los contenidos.

En la actualidad la redifusión web consiste en ofrecer un contenido desde una fuente web (*También llamados canales de contenidos o feeds. Son ficheros consistentes que el ordenador puede leer automáticamente y que permiten a los sitios web compartir su contenido de forma estándar con otras aplicaciones. Es decir, es un medio de difusión de contenido web que se encarga de suministrar con frecuencia información actualizada a sus suscriptores*), cuyo origen está en una página web, para proporcionar a los usuarios la actualización del mismo.

Las fuentes suelen codificarse en lenguaje XML (*eXtensibleMarkupLanguage, significa Lenguaje de Marcas Extensible*), aunque es válido hacerlo en cualquier lenguaje que se pueda transportar mediante el protocolo (*conjunto de normas utilizado por un conjunto de ordenadores conectados en red para comunicarse*) http (*HiperText Transfer Protocol o Protocolo de Transferencia de Hipertexto*)

Para leer una fuente, o canal, hay que suscribirse a ella utilizando un agregador (*Es un software que permite suscribirse a fuentes web. Muestra al suscriptor las modificaciones que han tenido lugar en los contenidos publicados por el proveedor en los canales de contenidos elegidos*).

Autoevaluación

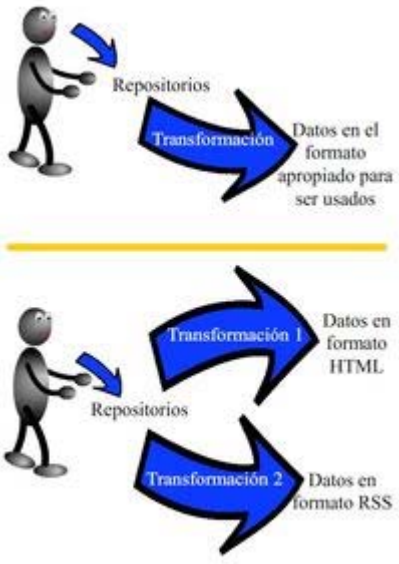
La sindicación de contenidos permite...

- Que el usuario o la usuaria de un sitio web reciba en su correo información sobre cuándo se actualiza una web a la que está suscrito o suscrita.
- Que el usuario o la usuaria de un sitio web puedan acceder a una información o servicio que se encuentra en un sitio web diferente.
- A las usuarias y a los usuarios de varios sitios web poder acceder a varios sitios diferentes desde una misma web.
- A los agregadores leer un canal rss.

Características

Publicar en la web puede ser visto como un flujo de información. Para que una web sea suministradora de un canal en su cabecera hay que incluir, por debajo del elemento **<title>**, un enlace al canal de contenidos. Para hacer esto hay que usar una de las dos líneas siguientes, dependiendo de que el canal esté hecho con un estándar **RSS** o con uno **Atom**, respectivamente:

```
<link rel="alternate" type="application/rss+xml" title="titulo_que_tendrá_el_enlace" href="http://www.misitio.com/fichero.rss" />
<link rel="alternate" type="application/atom+xml" title="titulo_que_tendrá_el_enlace" href="http://www.misitio.com/fichero.atom" />
```



Supongamos que el flujo de información de la publicación tiene su origen en unos ficheros localizados en un ordenador local. Por tanto el trabajo es hacer que dicha información llegue a los usuarios y usuarias que leen. Cuando la información se ha codificado en un documento HTML (*HyperTextMarkupLanguage, significa Lenguaje de Marcas de Hipertexto*), esto se logra actualizando dicho documento en el directorio adecuado del servidor web que contiene la página.

Actualmente es habitual el uso de algún CMS (*Content Management System o Sistema Gestor de Contenidos. Es un programa que permite crear una estructura de soporte para la creación y administración de contenidos, principalmente en páginas Web, por parte de los participantes. Para ello dispone de una interfaz que controla una o varias bases de datos donde se aloja el contenido del sitio*). En este caso los contenidos se encuentran en un repositorio y, antes de ser servidos al cliente en el formato adecuado, sufren algún tipo de transformación. La parte superior de la figura muestra la estructura del flujo de la información en este caso. Incluso puede haber más de un repositorio.

Esta transformación puede corresponder a uno de los siguientes casos:

- ✓ Documento XML -> Transformación XSLT (*eXtensibleStylesheetLanguageTransformations, que significa Lenguaje Extensible de Transformaciones de Hojas de Estilo*) -> Documento XHTML (*eXtensibleHypertextMarkupLanguage, significa Lenguaje Extensible de Marcado de Hipertexto*).
- ✓ Base de datos -> script en Perl -> Documento HTML.
- ✓ Texto plano -> ASP (*Active Server Pages, o servidor de páginas activo*)-> Documento HTML.
- ✓ Mente del autor -> Bloc de notas -> Documento HTML.

Al utilizar un CMS de cualquier tipo la transformación puede replicarse. Además de tener más de una entrada de información podríamos tener varias salidas. Por ejemplo, podemos generar tanto ficheros HTML como canales RSS (*ReallySimplySindication, significa Redifusión Realmente Simple. Estándar de la familia XML que permite compartir contenidos entre sitios Web*) tal y como se muestra en la parte inferior de la figura.

Autoevaluación
La única manera de publicar un canal en una web es a través de un fichero HTML.

- ○ Verdadero.
- ◉ Falso.

Ventajas de la redifusión de contenidos.

¿Cuáles serán las ventajas de utilizar los canales de contenidos de otros propietarios?

- ✓ Aumentar el tráfico de nuestro sitio web.
- ✓ Ayuda a que los usuarios y usuarias visiten frecuentemente el sitio web.
- ✓ Favorece el posicionamiento del sitio en buscadores.
- ✓ Ayuda a establecer relaciones entre distintos sitios web dentro de la comunidad.
- ✓ Permite a otras personas añadir características a los servicios del sitio web (por ejemplo, notificaciones de actualizaciones mediante mensajes instantáneos), aunque se requiera de tecnologías adicionales.
- ✓ Enriquece Internet impulsando la tecnología semántica y fomentando la reutilización.

Autoevaluación
Marca las ventajas de la sindicación de contenidos.

- ☑ Ayuda al sitio web a aparecer en los primeros lugares de los buscadores.
- ☐ Favorece la distribución de contenidos no aclarando la identidad del sitio.
- ☑ Favorece el trabajo con el significado de los datos en lugar de ocuparse sólo de los datos.
- ☐ Permite la generación de páginas web independientes.

Ámbitos de aplicación.

Caso práctico
Tras conocer las ventajas de la redifusión web, María y Félix se interesan por el tipo de datos que se pueden sindicar, ya que esto puede ser determinante para saber con qué formatos de ficheros se pueden trabajar.
Juan les contesta que, aunque lo más habitual es el texto, debido a que es el formato de datos más habitual de los blogs, en realidad se puede sindicar cualquier tipo de información y les pone como ejemplo de redifusión de videos a Youtube.

La redifusión web no es sólo un fenómeno vinculado a los weblogs, aunque ha ayudado mucho a su popularización. Siempre se han sindicado contenidos y se ha compartido todo tipo de información en formato XML.

De esta forma podemos ofrecer contenidos propios para que sean mostrados en otras páginas web de forma integrada, lo que aumenta el valor de la página que muestra el contenido y también nos genera más valor, ya que normalmente la redifusión web siempre enlaza con los contenidos originales.

La redifusión de contenidos web puede aplicarse a todo tipo de contenidos, es decir, texto, audio, vídeos e imágenes.

Desde el punto de vista de los suscriptores, la redifusión de contenidos permite, entre otras cosas, la actualización profesional. Mediante la suscripción a sitios relevantes, el usuario o la usuaria puede estar al día en temas relacionados con su profesión, recibiendo las noticias e informaciones en su blog o en su programa agregador de noticias.

Autoevaluación
Marcar cuáles de las siguientes aplicaciones son ejemplos de redifusión de contenidos.

- [x] Blog.
- [ ] Facebook.
- [ ] Google.
- [x] Youtube.

Tecnologías de creación de canales de contenidos.

Caso práctico
*Viendo que el formato de la información a distribuir no plantea ningún problema, **María** se cuestiona si para ofrecer estos servicios de información a sus clientes basta con hacer un enlace a los ficheros que contienen la información utilizando HTML o XHTML.*
***Juan** le informa que para sindicar contenidos hay que utilizar alguno de los estándares de sindicación, los cuales están basados en XML y se agrupan en dos estándares: RSS y Atom.*
Los estándares más utilizados se clasifican en dos grupos:

- **RSS**: (Really Simple Syndication) es parte de la familia de los formatos XML, desarrollado para compartir la información que se actualiza con frecuencia entre sitios web. Además se utiliza en la conexión con sistemas de mensajería instantánea, la conversión de RSS en mensajes de correo electrónico, o la capacidad de transformar los enlaces favoritos del navegador en RSS. Ha sido desarrollado por tres organizaciones diferentes, lo que ha dado lugar a siete formatos diferentes entre sí:

  - **RSS 0.90**, es el estándar que creó la empresa Netscape en el año 1999. Se basa en la especificación RDF de metadatos, con la intención de que su proyecto My Netscape estuviese formado por titulares de otras webs.
  - **RSS 0.91**, es la versión simplificada de RSS 0.90 que Nestscape lanzó posteriormente. El desarrollo de este formato se detuvo por falta de éxito, aunque la empresa UserLand Software decidió usar esta versión para desarrollar blogs.
  - **RSS 1.0**, fue creado a partir del estándar el RSS 0.90. Es más estable y permite definir una cantidad mayor de datos que el resto de versiones de RSS.
  - **RSS 2.0**, UserLand Software rechazó el estándar RSS 1.0 por considerarlo complejo y continuó el desarrollo del formato RSS 0.91, publicando las versiones 0.92, 0.93 y 0.94. Su sintaxis está incompleta y no cumplen todas las normas de XML. El estándar RSS 2.0 se publicó para subsanar esos problemas.
- **Atom:** fue publicado como un estándar propuesto por el grupo de trabajo *Atom Publishing Format and Protocol* (Formato y protocolo de publicación Atom) de la IETF en el RFC4287. Se desarrolló como una alternativa a RSS, con el fin de evitar la confusión creada por la existencia de estándares similares para la sindicación de contenidos, entre los que existía cierta incompatibilidad. En lugar de sustituir a los estándares existentes, se creó un nuevo estándar que convive con ellos. Se caracteriza por su flexibilidad. Atom permite tener un mayor control sobre la cantidad de información a representar en los agregadores.

Debes conocer
Las especificaciones de estos estándares se encuentran en los siguientes enlaces:

RSS 1.0	http://web.resource.org/rss/1.0
RSS 2.0	http://backend.userland.com/rss
Atom 0.3	http://tools.ietf.org/html/rfc4287

Estructura de los canales de contenidos.

Caso práctico
*Después de saber con qué lenguajes de marcas hay que trabajar, **Félix** pregunta qué es lo necesario para poder establecer un canal de contenidos.*
***Juan** le informa de que el primer paso es generar un fichero en RSS que ha de verificar las normas XML. Además de definir en él cuál es el estándar con el que se va a trabajar, hay que definir un canal donde se establecerá el sitio web asociado al mismo. Además en ese canal se definirán tantos ítems como sitios web se pretendan sindicar.*

Para construir un canal de contenido, es necesario crear un fichero, con extensión **rss** o**atom**, basado en XML. Este fichero se publicará en uno de los directorios del sitio web desde el que se oferta.

Estará formado por los siguientes elementos básicos:

- ✓ **Declaración del documento xml y la definición de la codificación empleada** en el documento. Ésta última será, preferentemente, **UTF-8** (*Unicode TransformationFormat 8 bits, significa Formato de transformación Unicode de 8 bits*), que es la codificación que se está imponiendo.
- ✓ **Un canal** en el que se determina el sitio web asociado a la fuente web a la que hace referencia el fichero. Éste, además de su propia definición, estará formado por:
  - → **Secciones**, cada una de las cuales es una referencia a la web que contiene uno de los servicios que se van a ofrecer. En un canal pueden incluirse tantas secciones como se quiera, lo que hace que un canal de contenido pueda tener un tamaño enorme si contiene un gran número de enlaces independientes.

No existe ninguna restricción respecto a la cantidad de canales de contenidos que se pueden ofrecer desde un sitio web.

RSS

¿Qué vamos a hacer ahora?

Vamos a construir un canal de contenidos utilizando el lenguaje RSS.

Para ello comenzaremos creando, con un editor de texto plano, un fichero con extensión rss.

La primera línea del mismo será, como hemos dicho en el apartado anterior, la **declaración de xml**y la **definición de la codificación** utilizada en el documento.

Después se determinará la **versión de RSS** utilizada mediante el elemento ***rss***, que será el ejemplar del documento xml. Un ejemplo de esta línea es:

```
<rssversion="2.0">
```

Tras ello definiremos el canal. Para ello utilizamos el elemento ***channel***, para su definición se han de incluir, como mínimo, los siguientes elementos:

Elementos imprescindibles para crear un canal en RSS	
Elemento	**Definición**
`<title>`	Es el título que se va a dar al sitio web. Se puede poner cualquier cosa.
`<link>`	Dirección web de la página asociada al fichero rss.
`<description>`	Breve comentario que defina la finalidad del sitio.
`<language>`	Determina el idioma utilizado en el sitio, en el caso del español su valor será **es**.

<item>	Definirá cada una de las secciones del canal.

En el fondo estos ítems del canal RSS son links a otros recursos, cada uno de los cuales tiene una descripción diferente. Los canales RSS son siempre usados con sistemas en los que el contenido puede estar segmentado en partes independientes que pueden ser enlazadas.

Los elementos incluidos obligatoriamente en cada ítem son:

Elementos imprescindibles para definir un ítem en RSS	
Elemento	**Definición**
<title>	Es el título del enlace al que se referencia (no tiene por qué coincidir con el del canal).
<guid>	URL (*UniformResourceLocator, significa Localizador Uniforme de Recursos*) de la página enlazada, que ha de pertenecer al dominio establecido en el canal.
<description>	Comentario que defina el contenido que ofrece este enlace..

Ejercicio resuelto

Ejemplo simple de un canal de contenidos con dos enlaces codificado en RSS.

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

<rss version="2.0">

  <!--Se define el canal-->

  <channel>
      <title>Ejemplo de RSS</title>
      <link>http://www.infoalisal.com</link>
      <description>Este es un ejemplo de web basada en la sindicacion de contenidos</description>
      <language>es</language>

      <!--Se definen los distintos items del canal-->

      <item>
          <title>Item01</title>
          <link>http://www.infoalisal.com/seccion01</link>
          <description>Este es un ejemplo01 de web basada en la  sindicacion de contenidos movido</description>
      </item>

      <item>
          <title>Item02</title>
          <link>http://www.infoalisal.com/seccion02</link>
          <description>Este es un ejemplo02 de web basada en la  sindicacion de contenidos</description>
      </item>
  </channel>
</rss>

```

Autoevaluación

Para generar un canal de contenidos en RSS es necesario...

- (•) Un fichero XML que además de tener una declaración XML ha de tener otra de RSS.
- ( ) Un fichero HTML en el que se establecen los enlaces de los diferentes servicios que se sindican.
- ( ) Dos ficheros XML diferentes, uno para definir el canal y otro para los ítems.
- ( ) Un fichero RSS en el que hay definido al menos un ítem en el interior del elemento *rss*.

Atom

En este caso construiremos el mismo canal de contenidos utilizando el estándar Atom. Empezamos generando, con un editor de texto plano, un fichero con extensión atom.

Al igual que en el caso anterior la primera línea del mismo será la declaración de xml y la definición de la codificación utilizada en el documento.

Tras ello definimos el canal, el estándar de atom y el lenguaje utilizado en el fichero. En este caso usamos el elemento feed.

```
<feedxmlns="http://www.w3.org/2005/Atom" xml:lang="es-es">
```

Para definir el canal, el elemento feed ha de incluir, como mínimo, los siguientes elementos:

Elementos imprescindibles para definir un canal en Atom.	
Elemento	**Definición**
`<title>`	Es el título que se va a dar al sitio web. Se puede poner cualquier cosa.
`<id>`	Identificador del canal. Habitualmente es su URL.
`<link>`	Enlaces que definen el canal. Son necesarios dos: ✓ Uno al propio fichero .atom, cuyo valor del atributo ***rel***del elemento ***link***será "***self***". ✓ Otro al fichero web que oferta ese canal, en este caso ***rel="alternate"***.
`<updated>`	Fecha y hora de actualización, utilizando el formato **CCYY-MM-DDTHH:MM:SSZ,** donde **T** es el separador entre la fecha y la hora y **Z** indica que la hora hace referencia al sistema de tiempo universal, esto es la hora zulú, o la hora del meridiano de Greenwich. Entonces para indicar que el canal se ha actualizado el 6 de febrero de 2010 a la 17:15 hora española tenemos que poner: 2010-02-06T16:15:00Z
`<author>`	Determina el autor del enlace. Puede contener otros elementos como name o email.
`<entry>`	Definirá cada una de las secciones del canal.

Los elementos incluidos obligatoriamente en cada elemento entry son:

Elementos imprescindibles para definir un *entry* en Atom.	
Elemento	**Definición**
`<title>`	Es el título del enlace al que se referencia.
`<id>`	Identificador de la sección, ha de ser única en el fichero.
`<link>`	Enlace a la fuente de la sección, en este caso ***rel="alternate".***
`<updated>`	Fecha y hora de actualización, sólo se modifica en casos significativos. Sigue el formato explicado anteriormente.
`<author>`	Determina el autor del enlace.
`Summary`	Breve resumen del contenido del enlace.

Ejercicio resuelto

Ejemplo simple de un canal de contenidos con dos enlaces codificado en Atom.

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

<!--Se crea el canal, se determina el estándar a utilizar y el idioma del documento -->

<feed xmlns="http://www.w3.org/2005/Atom"  xml:lang="es-es">
   <title type="text">Ejemplo de RSS</title>
  <updated>2011-01-14T19:17:46Z</updated>
 <link rel="self" type="application/atom+xml" href="http://www.infoalisal.com/feed/atom" />
  <link rel="alternate" type="text/html" href="http://www.infoalisal.com"/>
  <id>http://www.infoalisal.com/feed/atom</id>

  <!--Se definen las secciones que forman el canal -->

  <entry>
      <title>Ejemplo de Atom</title>
     <link rel="alternate" type="text/html" href="http://www.infoalisal.com/seccion01" />
     <updated>2011-01-14T19:19:46Z</updated>
     <id>http://www.infoalisal.com/seccion01</id>
     <author>
         <name>Marina Toro</name>
         <email>marina.toro@alisal.com</email>
     </author>
     <summary>Ejemplo01 de seccion para un canal de contenidos</summary>
   </entry>

  <entry>
      <title>Ejemplo de Atom</title>
     <link rel="alternate" type="text/html" href="http://www.infoalisal.com/seccion01" />
     <updated>2010-12-14T19:17:55Z</updated>
     <id>http://www.infoalisal.com/seccion02</id>
     <author>
         <name>Juan Flores</name>
     </author>
     <summary>Ejemplo02 de seccion para un canal de contenidos</summary>
   </entry>
</feed>

```

Autoevaluación

Para generar un canal de contenidos es necesario...

- [x] Un fichero XML que además de tener una declaración XML ha de tener otra de RSS.
- [ ] Un fichero HTML en el que se establecen los enlaces de los diferentes servicios que se sindican.
- [ ] Dos ficheros XML diferentes, uno para definir el canal y otro para los ítems.
- [ ] Un fichero RSS en el que hay definido al menos un ítem en el interior del elemento rss.

Validación.

Caso práctico
Juan *les explica que una vez generado el fichero con el canal hay que verificar que su codificación es correcta. Para ello no es necesario tener en el equipo local ningún elemento especial, ya que basta con tener una conexión a Internet y entrar en uno de los validadores de fuentes de contenidos.*
María *se interesa por los datos que hay que darle al validador para que realice el trabajo.* ***Juan*** *le contesta que hay dos posibilidades. La más habitual es darle al validador la URL del canal que se quiere validar, pero existen validadores que permiten que se les proporcione el código fuente del fichero.*
Una vez que se ha creado un fichero fuente, ¿qué debemos hacer?

¡Hay que comprobar que es válido!

En internet hay múltiples lugares que dan este servicio.

Para validar un documento RSS con uno de estos validadores, se le da la dirección del fichero donde se encuentra alojado y comprueba que lo pueden encontrar, es decir que la URI es válida, y que no contiene errores.

Una vez validado, suelen ofrecer una imagen del tipo "XML" o "RSS", de color naranja por lo general, que se puede incluir en la página principal, para enlazar a la dirección del fichero alojado en su dominio. Así, cuando un visitante pulse sobre este pequeño icono, accederá directamente al contenido actual de la fuente y podrá navegar a través de él a las páginas que más le interesen.

Algunos de estos servicios de validación también ofrecen imágenes que se pueden incluir en la página para que cualquier visitante compruebe que el canal es válido.

Debes conocer

Algunos de los validadores que podemos encontrar en Internet son:

- ✓ FeedValidator. http://feedvalidator.org/
- ✓ W3C Feed Validation Service mediante URI. http://validator.w3.org/feed/#validate_by_uri
- ✓ W3C Feed Validation Service mediante código. http://validator.w3.org/feed/#validate_by_input
- ✓ RSS Advisory Board. http://www.rssboard.org/rss-validator/
- ✓ googletransitdatafeed. http://code.google.com/p/googletransitdatafeed/wiki/FeedValidator

Utilización de herramientas.

Caso práctico
María *preguntó a* ***Juan****, si al trabajar con estas tecnologías de sindicación hay que escribir los ficheros en el bloc de notas o, al igual que al trabajar con HTML, XHTML y XML también existen editores que faciliten la creación de estos ficheros.*
Éste respondió que el trabajo puede hacerse con el bloc de notas, pero que, como en el resto de los casos, existen editores que permiten a cualquier persona realizar esa tarea sin dificultad alguna. Además de permitir modificar y crear el documento, estas herramientas tienen más funcionalidades.

Existen herramientas que permiten crear y editar fuentes web sin necesidad de ser un experto en tecnologías de Internet. Poseen una interface que simplifica al máximo el trabajo con canales de contenidos.

Un ejemplo de este tipo de herramientas es el PSPad editor.

Algunos de los trabajos que pueden realizar son:

- ✓ Trabajar con distintos estándares.
- ✓ Importar documentos de texto CSV (*Comma-SeparatedValues, significa Valores Separados por Comas. Un documento CSV es un fichero de texto en el que los datos se representan en forma de tabla del siguiente modo: las columnas se separan por punto y coma y las filas por saltos de línea*) y HTML.
- ✓ Editar HTML con el editor WYSIWYG (*WhatYouSeeIsWhatYouGet, significa lo que ves es lo que obtienes. Se aplica a los editores de texto con formato que permiten ver el resultado del texto que se escribe*).
- ✓ Editar documentos XML e imágenes.
- ✓ Actualizar las fuentes por vía FTP (*File Transfer Protocol, significa Protocolo de Transferencia de Ficheros. Conjunto de reglas que se han de cumplir en el intercambio de archivos entre un cliente y un servidor a través de una red TCP/IP*).
- ✓ Tienen capacidad para exportar documentos RSS a HTML, CSV y JavaScript (*Aquel programa escrito en el lenguaje de programación interpretado y orientado a objetos de este nombre*)

Para saber más
Algunas de estas herramientas pueden descargarse desde el siguiente enlace:
Descargar herramientas. http://www.extralabs.net/downloads.htm

Autoevaluación
Trabajar con algún editor de fuentes web nos permite, entre otras cosas, generar fuentes con cualquier tecnología de sindicación.

- ◉ Verdadero.
- ○ Falso.

Directorios de canales de contenidos.

Caso práctico

__Juan__ continúa explicándoles el último paso para llevar a cabo el proceso de sindicación. Después de tener el fichero fuente validado, es necesario sindicarle.
__María__ pregunta qué pasos implica hacerlo.
__Juan__ le explica que no es más que registrar el fichero en un directorio de canales web.
__Félix__ pregunta en qué consiste y __Juan__ le dice que es un sitio web al que basta darle la URL de la fuente que se quiere sindicar. También les explica que en la mayor parte de los casos se requiere estar registrado en el sitio.

¿Para qué sirven los directorios de canales de contenidos?

Permiten que el fichero RSS esté disponible para cualquiera, además de facilitar a los usuarios y usuarias finales la búsqueda de información, ya que los directorios de canales de contenidos clasifican los ficheros RSS.

Para ello es necesario registrar el fichero RSS en un directorio RSS. El proceso es similar a registrar un sitio en un motor de búsqueda.

Para comenzar podemos utilizar:

- ✓ Syndic8: http://www.syndic8.com/Está en inglés y requiere registro a través de una cuenta de Yahoo!
- ✓ Uatsap: http://www.uatsap.com/Está en castellano.

Es posible que a la hora de sindicar un canal puedan surgir problemas debido al uso de tildes o ñ en los contenidos del canal. Es recomendable usar la codificación UTF-8 y las entidades XML que les sustituyen, es decir, ***á*** en lugar de á, ***é*** en lugar de é, ...,***<***en lugar de <, ***>*** en lugar de >, ...

Además de sindicar un canal, también podemos promocionarle en los buscadores y directorios más populares.

En Yahoo este servicio está reservado a los usuarios y usuarias registrados que ya tienen una página denominada MiYahoo.

En Google también puede hacerse, pero sólo para aquellos usuarios y usuarias que posean una cuenta gmail.

Para saber más

Puedes ver cómo promocionar un sitio web desde el siguiente enlace:
Promocionar. http://www.google.com/addurl

Agregación.

Caso práctico
Juan *les aclara que aquellos clientes que quieran utilizar este nuevo servicio tendrán que utilizar un agregador de contenidos para poder leer estos canales de información.*
Félix *se interesa mucho por este tema, ya que habrá que informar a los clientes de cómo instalarlo y utilizarlo.*
Juan *le tranquiliza diciéndole que es tan sencillo de usar como un gestor de correo y que no es imprescindible instalar la aplicación, ya que se puede utilizar un agregador web, en lugar de uno de escritorio.*

¿Qué es un agregador o lector de fuentes?

Es una aplicación de software para suscribirse a fuentes en formatos RSS y Atom. El agregador avisa al usuario o usuaria de qué webs han incorporado contenido nuevo desde nuestra última lectura y cuál es ese contenido.

En el agregador hay que indicar la dirección web de cada archivo fuente, ya sea en formato RSS o Atom, para que pueda acceder a sus contenidos, los interprete y los muestre.

Existen dos tipos de agregadores:

- **Los agregadores web (o agregadores en línea)**, son aplicaciones que residen en determinados sitios web y que se ejecutan a través de la propia web. Son recomendables cuando el usuario o la usuaria no accede siempre a Internet desde el mismo ordenador. Es el caso de Google Reader, Bloglines o Netvibes.
- **Los agregadores de escritorio**, son aplicaciones que se instalan en el ordenador del usuario o usuaria. Su uso es aconsejable para quienes accedan a Internet siempre desde el mismo ordenador. Su interfaz gráfica suele ser parecida a la de los programas de cliente de correo electrónico, con un panel donde se agrupan las suscripciones, y otro donde se accede a las entradas individuales para su lectura. Ejemplos de ellos son: Newsgator y FeedDemon.

Autoevaluación
Las siguientes afirmaciones son diferencias entre los directorios RSS y los agregadores de contenidos:

☐ El uso de directorios de contenidos es obligatorio mientras que el uso de los agregadores es opcional.

◉ Los primeros son usados por los creadores de canales de contenidos y los agregadores por los usuarios de los mismos.

☐ Los directorios de contenidos están en un sitio web mientras que los agregadores de contenidos siempre están en un equipo local.

☐ Los directorios de contenidos sólo son utilizados por los creadores de un canal mientras que los agregadores se usan tanto en el proceso de creación como en el de uso.

TEMA 4

Contenido

[DEFINICIÓN DE ESQUEMAS Y VOCABULARIOS EN XML]

José Luis Comesaña Cabeza --- 2011/2012

Programación del curso de "*Desarrollo de Aplicaciones Web*"

Definición de esquemas y vocabularios en xml.

Caso práctico
María había salido a dar un paseo por el puerto, como todos los domingos por la mañana iba acompañada de su marido José Ramón y su hijo.
Mientras les veía jugar, estaba pensando que al día siguiente tenía que hablar con Juan, el técnico que se encargaba de resolver los problemas de informática de la empresa de la cual es socia. Había estado pensando si existiría algún modo de poder garantizar que la estructura de datos de cada uno de los documentos XML que comparte con Félix, su socio, es la que tiene que ser y no otra, además de asegurar que todos los documentos del mismo tipo mantienen la misma estructura.

1 - Documento XML. Estructura y sintaxis.

Caso práctico
*Al día siguiente, **cuando habla con Juan**, sus dudas quedan disipadas. Resulta que hay **varias posibilidades para asegurar una normalización en el formato de los documentos** XML. Juan comienza por describir la estructura de un documento XML y Félix y María descubren que puede ser un poco más compleja que la que habían estado usando hasta entonces para generar sus documentos.*

Hasta ahora hemos trabajado con documentos básicos de XML. Esto significa que dichos documentos están incompletos ya que solo hemos declarado el tipo de documento que va a ser, es decir que ejemplar vamos a definir, pero no hemos definido qué cualidades tiene ese tipo.

En la primera unidad vimos que un documento XML básico estaba formado por un prólogo y un ejemplar.

Recordamos que cada una de esas partes tiene el siguiente cometido:

- **Prólogo:** Informa al intérprete encargado de procesar el documento de todos aquellos datos que necesita para realizar su trabajo. Consta de dos partes:
  - **Definición de XML:** Donde se indica la versión de XML que se utiliza, el código de los datos a procesar y la autonomía del documento. Este último dato hasta ahora siempre ha sido "yes" ya que los documentos generados eran independientes.
  - **Declaración del tipo de documento:** Hasta el momento solo hemos dicho que es el nombre del ejemplar precedido de la cadena <!DOCTYPE y separado de ésta por, al menos un espacio. En el apartado siguiente nos encargamos de sus características.
- **Ejemplar:** Contiene los datos del documento que se quiere procesar. **Es el elemento raíz del documento y ha de ser único**. Está compuesto de elementos estructurados según una estructura de árbol en la que el elemento raíz es el ejemplar y las hojas los elementos terminales, es decir, aquellos que no contienen elementos. Los elementos pueden estar a su vez formados por atributos.

Autoevaluación
Marcar los componentes de un documento XML:

- [x] Prólogo.
- [x] Ejemplar.
- [x] Definición de codificación del documento.
- [ ] Cabecera.

El prólogo, el ejemplar y la definición de la codificación del documento son partes de un fichero XML.

1.1 - Declaración de tipo de documento.

Ya habíamos visto que permite **al autor definir restricciones y características en el documento**, aunque no habíamos profundizado en las partes que la forman:

- ✓ **La declaración del tipo de documento propiamente dicha**. Comienza con el texto que indica el nombre del tipo, precedido por la cadena **" "** separado del nombre del tipo por, al menos, un espacio. El nombre del tipo ha de ser idéntico al del ejemplar del documento XML en el que se está trabajando.
- ✓ **La definición del tipo de documento**. Permite asociar al documento una definición de tipo DTD , la cual se encarga de definir las cualidades del tipo. Es decir, define los tipos de los elementos, atributos y notaciones que se pueden utilizar en el documento así como las restricciones del documento, valores por defecto, etc. Para formalizar todo esto, XML está provisto de ciertas estructuras llamadas **declaraciones de marcado**, las cuales pueden ser internas o externas. Normalmente un documento XML se compone de una mezcla de declaraciones de marcado internas y externas. En este último caso debe expresarse en el documento dónde encontrar las declaraciones, así como indicar en la declaración de XML que el documento no es autónomo. Las diferencias entre estos tipos de declaraciones de marcado dan lugar a dos subconjuntos el interno y el externo, conviene saber que primero se procesa el subconjunto interno y después el externo, lo que permite sobrescribir declaraciones externas compartidas entre varios documentos y ajustar el DTD a un documento específico.
  - → **Subconjunto interno:** Contiene las **declaraciones que pertenecen exclusivamente a un documento** y no es posible compartirlas. Se localizan dentro de unos corchetes que siguen a la declaración de tipo del documento.
  - → **Subconjunto externo:** Están localizadas en un documento con extensión **dtd** que puede situarse en el mismo directorio que el documento XML. Habitualmente son declaraciones que pueden ser compartidas entre múltiples documentos XML que pertenecen al mismo tipo. En este caso la declaración de documento autónomo ha de ser negativa, ya que es necesario el fichero del subconjunto externo para la correcta interpretación del documento. Con ello el procesado del documento será más lento, ya que antes de procesar el documento el procesador ha de obtener todas las entidades.

```
<!DOCTYPE nombre_ejemplar SYSTEM "URI"
```

En este caso, se especifica un URI donde pueden localizarse las declaraciones.

```
<!DOCTYPE nombre_ejemplar PUBLIC "id_publico" "URI"
```

En este caso también se especifica un identificador, que puede ser utilizado por el procesador XML para intentar generar un URI alternativo, posiblemente basado en alguna tabla. **Como se puede observar también es necesario incluir algún URI**.

  - → Ahora los corchetes pierden sentido, para localizar las declaraciones del tipo de documento externo mediante una declaración explícita de subconjunto externo se utiliza:

Autoevaluación
La definición de tipo de documento ha de ser:

- (•) Interna o externa al documento XML al que está referida.
- ( ) Interna al documento XML que le refiere.
- ( ) Externa al documento XML que le refiere.
- ( ) Única.

Muy bien. Has captado la idea. Correcta.

1.2 - Definición de la sintaxis de documentos XML.

Recordamos que en estos documentos las etiquetas de marcado describen la estructura del documento.

Un elemento es un grupo formado por una etiqueta de apertura, otra de cierre y el contenido que hay entre ambas.

En los documentos de lenguajes de marcas, la distribución de los elementos está jerarquizada según una estructura de árbol, lo que implica que es posible anidarlos pero no entrelazarlos.

Hemos visto que en los elementos el orden es importante, ¿lo es también para los atributos? En este caso el orden no es significativo. Lo que hay que tener presente es que no puede haber dos atributos con el mismo nombre.

Sabemos que los atributos no pueden tener nodos que dependan de ellos, por tanto solo pueden corresponder con hojas de la estructura de árbol que jerarquiza los datos. ¿Significa esto que todas las hojas van a ser atributos? Pues no, es cierto que los atributos son hojas, pero las hojas pueden ser atributos o elementos.

En ese caso, ¿qué criterios podemos utilizar para decidir si un dato del documento que se pretende estructurar ha de representarse mediante un elemento o un atributo? Aunque no siempre se respetan, podemos usar los siguientes criterios:

- ✓ El dato será un elemento si cumple alguna de las siguientes condiciones:
  - → Contiene subestructuras.
  - → Es de un tamaño considerable.
  - → Su valor cambia frecuentemente.
  - → Su valor va a ser mostrado a un usuario o aplicación.
- ✓ Los casos en los que el dato será un atributo son:
  - → El dato es de pequeño tamaño y su valor raramente cambia, aunque hay situaciones en las que este caso puede ser un elemento.
  - → El dato solo puede tener unos cuantos valores fijos.
  - → El dato guía el procesamiento XML pero no se va a mostrar.

Los espacios de nombres, o namespaces, ¿qué nos permiten?

- ✓ Diferenciar entre los elementos y atributos de distintos vocabularios con diferentes significados que comparten nombre.
- ✓ Agrupar todos los elementos y atributos relacionados de una aplicación XML para que el software pueda reconocerlos con facilidad.

¿Cómo se declaran?

```
xmnls:"URI_namespace"
```

¿Y si se usa un prefijo que nos informe sobre cuál es el vocabulario al que está asociada esa definición?

```
xmnls:prefijo="URI_namespace"
```

En ambos casos URI_namespace es la localización del conjunto del vocabulario del espacio de nombres al que se hace referencia.

Autoevaluación
Marcar las afirmaciones válidas que hacen referencia a un espacio de nombres.

- ☑ Facilitan que el software localice el vocabulario de la aplicación.
- ☐ Define los atributos que forman parte de un documento.
- ☑ Permiten tener varios elementos homónimos con diferentes significados.

- Determinan los elementos que forman parte de un documento XML.

Los espacios de nombre facilitan la diferenciación entre elementos de distintos vocabularios que se llaman igual, además simplifican la localización del vocabulario de la aplicación XML.

2 - Definiciones de tipo de documento, DTD.

Caso práctico

Según Juan el método más sencillo para intentar normalizar los documentos *con los que trabajan, consiste en* ***definir unos vocabularios*** *que han de cumplir los documentos que generan, estos se llaman Definición de Tipo de Documento. Además, aunque no es un lenguaje XML, tiene una sintaxis sencilla y fácil para que ella y Félix puedan comprenderla y utilizarla.*

Están formadas por una relación precisa de qué elementos pueden aparecer en un documento y dónde, así como el contenido y los atributos del mismo. Garantizan que los datos del documento XML cumplen las restricciones que se les haya impuesto en el DTD, ya que estas últimas permiten:

- ✓ Especificar la estructura del documento.
- ✓ Reflejar una restricción de integridad referencial (*Conjunto de normas que utilizan las bases de datos relacionales para garantizar que los registros de tablas relacionadas son válidos*) mínima utilizando (ID e IDREF).
- ✓ Utilizar unos pequeños mecanismos de abstracción comparables a las macros (*Instrucción que se almacena en la aplicación que la utiliza y se ejecuta de forma secuencial*), que son las entidades.
- ✓ Incluir documentos externos.

¿Cuáles son los inconvenientes de los DTD?

Los principales son:

- ✓ Su sintaxis no es XML.
- ✓ No soportan espacios de nombres.
- ✓ No definen tipos para los datos. Solo hay un tipo de elementos terminales, que son los datos textuales.
- ✓ No permite las secuencias no ordenadas.
- ✓ No es posible formar claves a partir de varios atributos o elementos.
- ✓ Una vez que se define un DTD no es posible añadir nuevos vocabularios.

Cuando están definidas dentro del documento XML se ubican entre corchetes después del nombre del ejemplar en el elemento <!DOCTYPE> pero, cuando está definido en un fichero externo ¿a qué tipo de fichero corresponde?

Definimos el DTD externo en un fichero de texto plano con extensión **dtd**.

Autoevaluación

Marcar las afirmaciones referidas a un DTD:

- ☐ Permite la formación de identificadores compuestos por varios atributos.
- ☑ Para construirlas no se usa un lenguaje basado en XML.
- ☑ Permite incluir ficheros binarios en el documento.
- ☐ Definen distintos tipos para los datos.

Un DTD permite incluir en el fichero XML ficheros binarios, además no se construyen en XML.

2.1 - Declaraciones de tipos de elementos terminales.

Los **tipos terminales** son aquellos elementos que se corresponden con hojas de la estructura de árbol formada por los datos del documento XML asociado al DTD. La declaración de tipos de elementos está formada por la cadena **"<!ELEMENT"** separada por, al menos un espacio del nombre del elemento XML que se declara, y seguido de la declaración del contenido que puede tener dicho elemento.

En el caso de elementos terminales, es decir, aquellos que no contienen más elementos, esta declaración de contenido es dada por uno de los siguientes valores:

- ✓ **EMPTY:** Indica que el elemento no es contenedor. Por ejemplo, la siguiente definición muestra un elemento A que no contiene nada:

```
<!ELEMENT A EMPTY>
```

- ✓ **ANY:** Permite que el contenido del elemento sea cualquier cosa. Un ejemplo de definición de un elemento de este tipo es:

```
<!ELEMENT A ANY>
```

- ✓ **(#PCDATA):** Indica que los datos son analizados en busca de etiquetas, resultando que el elemento no puede contener elementos, es decir solo puede contener datos de tipo carácter exceptuando los siguientes: <, &,]], >. Si es de este tipo, el elemento A tendrá una definición como:

```
<!ELEMENT A (#PCDATA)>
```

Ejercicio resuelto

Creación de un DTD correspondiente a la siguiente estructura de datos de un documento XML:

```
<alumno>Olga Velarde Cobo</alumno>
```

Resultado

```
<!ELEMENT alumno (#PCDATA)>
```

Autoevaluación

Los elementos terminales de tipo ANY son aquellos que están:

- ○ Formados por cadenas de texto exclusivamente.
- ○ Vacíos.
- ◉ Formados por cualquier cosa.
- ○ Estará formada por otros elementos.

2.2 - Declaraciones de tipos de elementos no terminales.

Una vez que sabemos el modo de definir las hojas de un árbol de datos **veamos cómo definir sus ramas**, es decir los elementos que están formados por otros elementos.

Para definirlos utilizamos referencias a los grupos que los componen tal y como muestra el ejemplo:

```
<!ELEMENT A (B, C)>
```

En este caso se ha definido un elemento A que está formado por un elemento B seguido de un elemento C.

¿Y qué sucede cuando un elemento puede aparecer en el documento varias veces, hay que indicarlo de algún modo? Pues sí, también hay que indicar cuando un elemento puede no aparecer. Para ello usamos los siguientes operadores, que nos permiten definir la cardinalidad (*Cantidad de veces que un elemento puede aparecer en un documento*) de un elemento:

- ✓ **Operador opción, ?.** Indica que el elemento no es obligatorio. En el siguiente ejemplo el subelemento trabajo es opcional.

```
<!ELEMENT telefono (trabajo?, casa ) >
```

- ✓ **Operador uno-o-más, +.** Define un componente presente al menos una vez. En el ejemplo definimos un elemento formado por el nombre de una provincia y otro grupo, que puede aparecer una o varias veces.

```
<!ELEMENT provincia (nombre, (cp, ciudad)+ ) >
```

- ✓ **Operador cero-o-mas, *.** Define un componente presente cero, una o varias veces. En el ejemplo el grupo (cp, ciudad) puede no aparecer o hacerlo varias veces.

```
<!ELEMENT provincia (nombre, (cp, ciudad)* ) >
```

- ✓ **Operador de elección, |.** Cuando se utiliza sustituyendo las comas en la declaración de grupos indica que para formar el documento XML hay que elegir entre los elementos separados por este operador. En el ejemplo siguiente, el documento XML tendrá elementos provincia que estarán formados por el elemento nombre y el cp (código postal), o por el elemento nombre y la ciudad.

```
<!ELEMENT provincia (nombre, (cp | ciudad) ) >
```

Ejercicio resuelto

```
Creación de un DTD correspondiente a la siguiente estructura de datos de un documento XML:
<alumno>
   <nombre>Olga</nombre>
   <dirección>El Percebe 13</dirección>
</alumno>
```

Resultado

```
<!ELEMENT alumno (nombre, apellidos, direccion)>
<!ELEMENT nombre  (#PCDATA)>
<!ELEMENT dirección (#PCDATA))>
```

Autoevaluación

La diferencia entre el operador ? y el + es que el primero permite que el elemento sobre el que se aplica esté presente una vez, como máximo, mientras que el operador + no limita el número máximo de veces que está presente el elemento en el documento XML.

- ◉ Verdadero.
- ○ Falso.

2.3 - Declaraciones de listas de atributos para los tipos de elementos.

Ya sabemos cómo declarar elementos, ahora veamos el modo de **declarar los atributos asociados a un elemento**. Para ello utilizamos la cadena **<!ATTLIST** seguida del nombre del elemento asociado al atributo que se declara, luego el nombre de éste último seguido del tipo de atributo y del modificador. Este elemento puede usarse para declarar una lista de atributos asociada a un elemento, o repetirse el número de veces necesario para asociar a dicho elemento esa lista de atributos, pero individualmente.

Al igual que los elementos no todos los atributos son del mismo tipo, los más destacados son:

- ✓ Enumeración, es decir, el atributo solo puede tomar uno de los valores determinados dentro de un paréntesis y separados por el operador |.

```
<!ATTLIST fecha dia_semana (lunes|martes|miércoles|jueves|viernes|sábado|domingo) #REQUIRED>
```

- ✓ **CDATA**, se utiliza cuando el atributo es una cadena de texto.
- ✓ **ID**, permite declarar un atributo identificador en un elemento. Hay que recordar que este valor ha de ser único en el documento. Además hay que tener en cuenta que los números no son nombres válidos en XML, por tanto no son un identificador legal de XML. Para resolverlo suele incluirse un prefijo en los valores y separarlo con un guión o una letra.
- ✓ **IDREF**, permite hacer referencias a identificadores. En este caso el valor del atributo ha de corresponder con el de un identificador de un elemento existente en el documento.
- ✓ **NMTOKEN**, permite determinar que el valor de un atributo ha de ser una sola palabra compuesta por los caracteres permitidos por XML.

¿También hemos de declarar si el valor de un atributo es obligatorio o no? Si, para ello se usan los siguientes modificadores:

- ✓ **#IMPLIED**, determina que el atributo sobre el que se aplica es opcional.
- ✓ **#REQUIRED**, determina que el atributo tiene carácter obligatorio.
- ✓ **#FIXED**, permite definir un valor fijo para un atributo independientemente de que ese atributo se defina explícitamente en una instancia del elemento en el documento XML.
- ✓ Literal, asigna a un atributo el valor dado por una cadena entre comillas.

Ejercicio resuelto

```
Creación de un DTD correspondiente a la siguiente estructura de datos de un documento XML:
<alumno edad=15>
   <nombre>Olga</nombre>
   <apellidos>Velarde Cobo</apellidos>
   <dirección>El Percebe 13</dirección>
</alumno>
```

Resultado

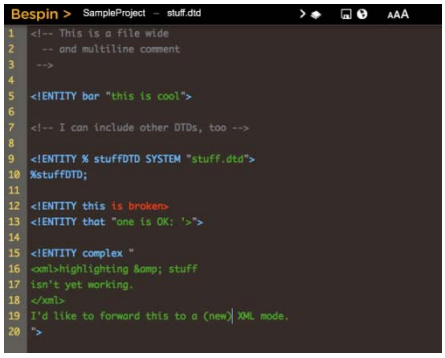
```
<!ELEMENT alumno (nombre, apellidos, direccion)>
<!ATTLIST alumno edad CDATA #REQUIRED>
<!ELEMENT nombre (#PCDATA)>
<!ELEMENT apellidos (#PCDATA)>
<!ELEMENT dirección (#PCDATA)>
```

Autoevaluación

¿Cuál de los siguientes atributos permite añadir a los datos una restricción de integridad referencial?

- ☐ MTOKEN.
- ☐ ID.
- ☉ IDREF.
- ☐ CDATA.

2.4 - Declaraciones de entidades.



¿Qué sucede si queremos **declarar valores constantes dentro de los documentos**? ¿Podemos?

Las entidades nos permiten definir constantes en un documento XML. Cuando se usan dentro del documento XML se limitan por "&" y ";", por ejemplo &entidad;

¿Cómo trabaja el intérprete con ellos? Al procesar el documento XML, el intérprete sustituye la entidad por el valor que se le ha asociado en el DTD.

No admiten recursividad, es decir, una entidad no puede hacer referencia a ella misma.

Para definir una entidad en un DTD se usa el elemento <!ENTITY>

Las entidades pueden ser de tres tipos:

- ✓ **Internas:** Existen cinco entidades predefinidas en el lenguaje, son:
  - → **<:** Se corresponde con el signo menor que, <.
  - → **>:** Hace referencia al signo mayor que, >.

- **":** Son las comillas rectas dobles, ''.
- **':** Es el apóstrofe o comilla simple, '.
- **&:** Es el et o ampersand, &.

¿Se puede definir una entidad diferente? ¿Cómo?

Utilizando la siguiente sintaxis:

```
<!ENTITY nombre_entidad "valor de la entidad">
```

Por ejemplo, <!ENTITY dtd "Definiciones de Tipo de Documento">

Externas: Permiten establecer una relación entre el documento XML y otro documento a través de la URL de éste último. Un ejemplo de declaración de una entidad externa es:

```
<!ENTITY nombre_entidad SYSTEM "http://localhost/docsxml/fichero_entidad.xml">
```

En este caso el contenido de los ficheros es analizado, por lo que deben seguir la sintaxis XML.

Cuando es necesario incluir ficheros con formatos binarios, es decir ficheros que no se analicen, se utiliza la palabra reservada NDATA en la definición de la entidad y habrá que asociar a dicha entidad una declaración de notación, tal y como muestra el ejemplo del apartado siguiente.

De parámetro: Permite dar nombres a partes de un DTD y hacer referencia a ellas a lo largo del mismo. Son especialmente útiles cuando varios elementos del DTD comparten listas de atributos o especificaciones de contenidos. Se denotan por %entidad;

```
<!ENTITY %direccion "calle, numero?, ciudad, cp">
<!ENTITY alumno (dni, %direccion;)>
<!ENTITY ies (nombre, %direccion;)>
```

De parámetro externas: Permite incluir en un DTD elementos externos, lo que se aplica en dividir la definición DTD en varios documentos.

```
<!ENTITY persona SYSTEM "persona.dtd">
```

Autoevaluación

Las entidades permiten definir elementos cuyo valor es constante dentro de un DTD.

- ○ *Verdadero.*
- ◉ *Falso.*

2.5 - Declaraciones de notación.

Cuando se incluyen ficheros binarios en un fichero, ¿cómo le decimos qué aplicación ha de hacerse cargo de ellos? La respuesta es utilizando notaciones. La sintaxis para declarar **notaciones** es:

```
<!NOTATION nombre SYSTEM aplicacion>
```

Por ejemplo, una notación llamada **gif** donde se indica que se hace referencia a un editor de formatos gif para visualizar imágenes será:

```
<!NOTATION gif SYSTEM "gifEditor.exe">
```

Para asociar una entidad externa no analizada, a esta notación basta declarar dicha entidad del siguiente modo:

```
<!ENTITY dibujo SYSTEM "imagen.gif" NDATA gif>
```

Autoevaluación

Las notaciones permiten:

- ☐ *Establecer el modo en el que se ha de estructurar un fichero XML.*
- ◉ *Determinar cuál es la aplicación que ha de procesar un fichero binario asociado a un fichero XML.*
- ☐ *Asociar un fichero binario a un fichero XML.*
- ☐ *Definir las etiquetas válidas en un fichero XML.*

2.6 - Secciones condicionales.

Permiten incluir o ignorar partes de la declaración de un DTD. Para ello se usan dos tokens (*Cadena de caracteres que forman parte del vocabulario de un lenguaje de programación*):

✓ **INCLUDE**, permite que se vea esa parte de la declaración del DTD. Su sintaxis es:

```
<![INCLUDE [Declaraciones visibles] ] >
```

Por ejemplo:

```
<![INCLUDE [ <!ELEMENT nombre (#PCDATA)>] ] >
```

✓ **IGNORE**, permite ocultar esa sección de declaraciones dentro del DTD. La forma de uso es:

```
<![IGNORE [Declaraciones ocultas] ] >
```

Por ejemplo:

```
<![IGNORE [<!ELEMENT clave (#PCDATA)>] ] >
```

Autoevaluación

Las sentencias condicionales permiten definir unos elementos u otros dentro del fichero XML en función de una determinada condición.

- ☐ Verdadero.
- ◉ Falso.

3 - XML Schema.

Caso práctico

Félix, quien considera que la normalización de los documentos XML que manejan en la empresa va a ser un duro trabajo para María, él y otros trabajadores inexpertos, plantea la posibilidad de que se encargue de ello algún trabajador de la consultoría informática que dirige Juan.
Al final se va a encargar de ello Marina. Les explica que, en lugar de trabajar con DTD's le parece mejor hacerlo con un lenguaje XML llamado XML Schema, el cual tiene, entre otras, la ventaja de permitir definir el tipo de datos de cada uno de los componentes de cada documento.

Los **DTD** permiten diseñar un **vocabulario para ficheros XML**, pero, ¿qué sucede cuando los valores de los elementos y atributos de esos ficheros han de corresponder a datos de un tipo determinado, o cumplir determinadas restricciones que no pueden reflejarse en los DTD? Para ello se definen **XML Schemas**.

¿También **se definen en ficheros planos**? Si, ya que son documentos XML, pero en este caso la extensión de los archivos es xsd, motivo por el cual también se les denomina documentos XSD.

Los elementos XML que se utilizan para generar un esquema han de pertenecer al espacio de nombre XML Schema, que es: http://www.w3.org/2001/XMLSchema.

El ejemplar de estos ficheros es **<xs:schema>**, contiene declaraciones para todos los elementos y atributos que puedan aparecer en un documento XML asociado válido. Los elementos hijos inmediatos de este ejemplar son **<xs:element>** que nos permiten crear globalmente un elemento. Esto significa que el elemento creado puede ser el ejemplar del documento XML asociado.

Ejercicio resuelto

Creación de un esquema correspondiente al siguiente documento XML:

```
<?xml version="1.0" encoding="UTF-8" standalone="yes?>
<!DOCTYPE alumno>
<alumno edad="22">Olga Velarde Cobo</alumno>
```

Resultado

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

<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
   <xs:element name="alumno" type="xs:string"/>
</xs:schema>
```

Debes conocer

En este primer enlace encontrarás los fundamentos del estándar XML Schema.
XML Schema Fundamentos http://www.w3.org/TR/2001/REC-xmlschema-0-20010502

3.1 - Tipos de datos.

Son los distintos valores que puede tomar el **atributo type** cuando se declara un elemento o un atributo y representan el tipo de dato que tendrá el elemento o atributo asociado a ese **type** en el documento XML.

Algunos de estos valores predefinidos son:

- ✓ **string**, se corresponde con una cadena de caracteres UNICODE.
- ✓ **boolean**, representa valores lógicos, es decir que solo pueden tomar dos valores, true o false.
- ✓ **integer**, número entero positivo o negativo.
- ✓ **positiveInteger**, número entero positivo.
- ✓ **negativeInteger**, número entero negativo.
- ✓ **decimal**, número decimal, por ejemplo, 8,97.

- ✓ **dateTime**, representa una fecha y hora absolutas.
- ✓ **duration**, representa una duración de tiempo expresado en años, meses, días, horas, minutos segundos. El formato utilizado es: PnYnMnDTnHnMnS. Por ejemplo para representar una duración de 2 años, 4 meses, 3 días, 5 horas, 6 minutos y 10 segundos habría que poner: P2Y4M3DT5H6M7S. Se pueden omitir los valores nulos, luego una duración de 2 años será P2Y. Para indicar una duración negativa se pone un signo – precediendo a la P.
- ✓ **time**, hora en el formato hh:mm:ss.
- ✓ **date**, fecha en formato CCYY-MM-DD.
- ✓ **gYearMonth**, representa un mes de un año determinado mediante el formato CCYY-MM.
- ✓ **gYear**, indica un año gregoriano, el formato usado es CCYY.
- ✓ **gMothDay**, representa un día de un mes mediante el formato –MM-DD.
- ✓ **gDay**, indica el ordinal del día del mes mediante el formato –DD, es decir el 4º día del mes será –04.
- ✓ **gMonth**, representa el mes mediante el formato –MM. Por ejemplo, febrero es –02.
- ✓ **anyURI**, representa una URI.
- ✓ **language**, representa los identificadores de lenguaje, sus valores están definidos en RFC 1766.
- ✓ **ID, IDREF, ENTITY, NOTATION, MTOKEN**. Representan lo mismo que en los DTD's (ver apartado 2.3).

Autoevaluación
Cuál de los siguientes tipos no hace referencia a un dato de tiempo:

- ☐ dateTime.
- ☐ duration.
- ◉ gDayMonth.
- ☐ gMonthDay.

Debes conocer
En este enlace encontrarás los tipos de datos admitidos por el estándar.
XML Schema Tipos de datos http://www.w3.org/TR/2001/REC-xmlschema-2-20010502

3.2 - Facetas de los tipos de datos.

¿Cuáles son las restricciones que podemos aplicar sobre los valores de los datos de un elemento o atributo?

Están definidos por las **facetas**, que solo pueden aplicarse sobre tipos simples utilizando el elemento **xs:restriction**. Se expresan como un elemento dentro de una restricción y se pueden combinar para lograr restringir más el valor del elemento. Son, entre otros:

- ✓ **length, minlength, maxlentgh**: Longitud del tipo de datos.
- ✓ **enumeration**: Restringe a un determinado conjunto de valores.

Ejercicio resuelto

Creación de una cadena de texto con una longitud máxima de 9 caracteres y dos valores posibles.

```
<xs:simpleType name="estado">
   <xs:restriction base="xs:string">
      <xs:maxLength value="9"/>
      <xs:enumeration value="conectado"/>
      <xs:enumeration value="ocupado"/>
   </xs:restriction>
<xs:simpleType>
```

- ✓ **whitespace:** Define el tratamiento de espacios (preserve/replace, collapse).

Ejercicio resuelto

Creación de un elemento en el que se respetan los espacios tal y como se han introducido.

```
<xs:simpleType name="nombre">
   <xs:restriction base="xs:string">
      <xs:whitespace value="preserve"/>
   </xs:restriction>
<xs:simpleType>
```

- ✓ **(max/min)(In/Ex)clusive:** Límites superiores/inferiores del tipo de datos. Cuando son Inclusive el valor que se determine es parte del conjunto de valores válidos para el dato, mientras que cuando se utiliza Exclusive, el valor dado no pertenece al conjunto de valores válidos.
- ✓ **totalDigits, fractionDigits:** número de dígitos totales y decimales de un número decimal.

Ejercicio resuelto

Creación de un elemento calificaciones de dos dígitos cuyo valor es un número entero comprendido entre 1 y 10, ambos inclusive.

```
<xs:simpleType name="calificaciones">
   <xs:restriction base="xs:integer">
      <xs:totalDigits value="2"/>
      <xs:minExclusive value="0"/>
      <xs:maxInclusive value="10"/>
   </xs:restriction>
</xs:simpleType>
```

- ✓ **pattern:** Permite construir máscaras que han de cumplir los datos de un elemento. La siguiente tabla muestra algunos de los caracteres que tienen un significado especial para la generación de las máscaras.

Elementos para hacer patrones.			
Patron	**Significado**	**Patron**	**Significado**
[A-Z a-z]	Letra.	**AB**	Cadena que es la concatenación de las cadenas A y B.
[A-Z]	Letra mayúscula.	**A?**	Cero o una vez la cadena A.
[a-z]	Letra minúscula.	**A+**	Una o más veces la cadena A.
[0-9]	Dígitos decimales.	**A***	Cero o más veces la cadena A.
\D	Cualquier carácter excepto un dígito decimal.	**[abcd]**	Alguno de los caracteres que están entre corchetes.
(A)	Cadena que coincide con A.	**[^abcd]**	Cualquier carácter que no esté entre corchetes.
A | B	Cadena que es igual a la cadena A o a la B.	**\t**	Tabulación.

Ejercicio resuelto

El ejemplo siguiente muestra la utilización de pattern para crear la máscara de un DNI.

```
<xs:simpleType name="dni">
   <xs:restriction base="xs:string">
      <xs:pattern value="[0-9] [0-9] [0-9] [0-9] [0-9] [0-9] [0-9] [0-9] [A-Z]"/>
   </xs:restriction>
</xs:simpleType>
```

3.3 - Elementos del lenguaje.

Algunos de los más usados son:

- ✓ Esquema, **xs:schema**, contiene la definición del esquema.
- ✓ Tipos complejos, **xs:complexType**, define tipos complejos.
- ✓ Tipos simples, **xs:simpleType**, permite definir un tipo simple restringiendo sus valores.
- ✓ Restricciones, **xs:restriction**, permite establecer una restricción sobre un elemento de tipo base.

- ✓ Agrupaciones, **xs:group**, permite nombrar agrupaciones de elementos y de atributos para hacer referencia a ellas.
- ✓ Secuencias, **xs:secuence**, permite construir elementos complejos mediante la enumeración de los que les forman.
- ✓ Alternativa, **xs:choice**, representa alternativas, hay que tener en cuenta que es una o-exclusiva.
- ✓ Contenido mixto, definido dando valor true al atributo mixed del elemento **xs:complexType**, permite mezclar texto con elementos.
- ✓ Secuencias no ordenadas, **xs:all**, representa a todos los elementos en cualquier orden.

Ejercicio resuelto

Ejemplo de esquema correspondiente a un documento XML para estructurar la información personal sobre los alumnos de un centro educativo.

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<xs:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema">
   <!-- elemento raíz -->
   <xs:element name="alumnos" type="datosAlum"/>
   <!-- Definicion del tipo datosAlum -->
   <xs:complexType name="datosAlum">
      <xs:sequence>
         <xs:element name="alumno" type="datos" minOccurs="1" maxOccurs="unbounded"/>
      </xs:sequence>
   </xs:complexType>
   <!-- Definicion del tipo datos -->
   <xs:complexType name="datos">
      <xs:sequence>
         <xs:element name="nombre" type="xs:string" minOccurs="1"  maxOccurs="1"/>
         <xs:element name="apellidos" type="xs:string" minOccurs="1" maxOccurs="1"/>
         <xs:element name="direccion" type="datosDireccion" minOccurs="1" maxOccurs="1"/>
         <xs:element name="contactar" type="datosContactar" minOccurs="1" maxOccurs="1"/>
      </xs:sequence>
      <!-- Atributos  del elemento usuario -->
      <xs:attribute name="id" type="xs:string"/>
   </xs:complexType>
   <xs:complexType name="datosDireccion">
      <xs:sequence>
         <xs:element name="domicilio" type="xs:string" minOccurs="0" maxOccurs="1"/>
         <xs:element name="codigo_postal" minOccurs="0" maxOccurs="1" >
            <xs:complexType>
               <xs:attribute name="cp" type="xsd:string"/>
            </xs:complexType>
         </xs:element>
         <xs:element name="localidad" type="xs:string" minOccurs="0" maxOccurs="1"/>
         <xs:element name="provincia" type="xs:string" minOccurs="0" maxOccurs="1"/>
      </xs:sequence>
   </xs:complexType>
   <xs:complexType name="datosContactar">
      <xs:sequence>
         <xs:element name="telf_casa" type="xs:string" minOccurs="0" maxOccurs="1"/>
         <xs:element name="telf_movil" type="xs:string" minOccurs="0" maxOccurs="1"/>
         <xs:element name="telf_trabajo" type="xs:string" minOccurs="0" maxOccurs="1"/>
         <xs:element name="email" minOccurs="0" maxOccurs="unbounded" >
            <xs:complexType>
               <xs:attribute name="href" type="xs:string"/>
            </xs:complexType>
         </xs:element>
      </xs:sequence>
   </xs:complexType>
</xs:schema>
```

Autoevaluación

Para hacer un elemento complejo formado por un listado de elementos en los que importa el orden hay que usar el elemento:

- ☐ <xs:choice>.
- ☐ <xs:all>.
- ☐ <xs:group>.
- ◉ <xs:secuence>.

Debes conocer

Este enlace te permitirá consultar las estructuras del estándar XML Schema.

XML Schema Estructuras http://www.w3.org/TR/2001/REC-xmlschema-1-20010502

3.4 - Definición de tipos de datos XML Schema.

En los DTD se diferencia entre los elementos terminales y los no terminales ¿en este caso también? Si, este lenguaje **permite trabajar tanto con datos simples como con estructuras de datos complejos**, es decir, compuestos por el anidamiento de otros datos simples o compuestos.

- ✓ **Tipos de datos simples.**

  Estos datos se suelen definir para hacer una restricción sobre un tipo de datos XDS ya definido y establece el rango de valores que puede tomar.

Ejercicio resuelto

Creación de un elemento simple de nombre edad que representa la edad de un alumno de la ESO, por tanto su rango está entre los 12 y los 18 años.

```
<xs:simpleType name="edad">
   <xs:restriction base="xsd:positiveInteger">
      <xs:maxExclusive value="10"/>
   </xs:restriction>
</xs:simpleType >
```

También se pueden crear tipos de datos simples basados en listas de valores utilizando el atributo derivedBy de simpleType.

Ejercicio resuelto

Creación de una lista con los días de la semana en letras.

```
<xs:simpleType name="dia_semana" base="xs:string" derivedBy="list"/>
<dia_semana>Lunes Martes Miercoles Jueves Viernes Sabado Domingo<dia_semana>
</xs:simpleType>
```

- ✓ **Tipos de datos compuestos.**

  El elemento xsd:complexType permite definir estructuras complejas de datos. Su contenido son las declaraciones de elementos y atributos, o referencias a elementos y atributos declarados de forma global. Para determinar el orden en que estos elementos aparecen en el documento XML se utiliza el elemento .

Ejercicio resuelto

Creación de un elemento compuesto de nombre alumno, formado por los elementos nombre, apellidos, web personal.

```
<xs:complexType name="alumno">
   <xs:secuence>
      <xs:element name="nombre" type="xs:string" minOccurs="1" maxOccurs="1"/>
      <xs:element name="apellidos" type="xs:string" minOccurs="1" maxOccurs="1"/>
      <xs:element name="web" type="xs:string" minOccurs="0" maxOccurs="5">
         <xs:complexType>
            <xs:attribute name="href" type="xs:string"/>
         </xs:complexType>
      </xs:element>
   <xs:secuence>
</xs:complexType>
```

3.5 - Asociación con documentos XML.

Una vez que tenemos creado el **fichero XSD** ¿cómo lo **asociamos a un fichero XML**?

El modo de asociar un esquema a un documento XML es un espacio de nombres al ejemplar del documento, donde se indica la ruta de localización de los ficheros esquema mediante su URI, precedida del prefijo " **xsi:**".

Ejercicio resuelto

Un documento XML asociado al esquema que se ha realizado anteriormente para estructurar la información personal sobre los alumnos de un centro educativo (apartado 3.3) puede ser:

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

<alumnos                                        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:SchemaLocation="file:/C:/Users/Ana_Desktop/alumnos.xsd">

   <alumno>
      <nombre>Jose Ramón</nombre>
      <apellidos>García González</apellidos>
      <direccion>
         <domicilio>El Pez, 12</domicilio>
         <codigo_postal>85620</código_postal>
         <localidad>Suances</localidad>
         <provincia>Cantabria</provincia>
      </direccion>
      <contactar>
         <telf._casa>985623165</telf._casa>
         <telf._movil>611233544</telf._movil>
         <telf._trabajo>965847536</telf._trabajo>
         <email>pepito@educadistancia.com</email>
      </contactar>
   </alumno>

   <alumno>
      <nombre>Carlos</nombre>
      <apellidos>López Pérez</apellidos>
      <direccion>
         <domicilio>El Cangrejo, 25</domicilio>
         <codigo_postal>86290</código_postal>
         <localidad>Santillana</localidad>
         <provincia>Cantabria</provincia>
      </direccion>
      <contactar>
         <telf._casa>931132565</telf._casa>
         <telf._movil>623863544</telf._movil>
         <telf._trabajo>984657536</telf._trabajo>
         <email>carlos@educadistancia.com</email>
      </contactar>
   </alumno>

</alumnos>
```

Autoevaluación

La asociación de un documento XML a un esquema se hace en:

- El prólogo del documento XML.
- La definición de tipo de datos.
- El ejemplar.
- En una sección llamada declaración del esquema que se sitúa entre el prólogo y el ejemplar.

3.6 - Documentación del esquema.

Una vez que hemos visto como crear un esquema vamos a ver el modo de incorporar cierta documentación (quién es el autor, limitaciones de derechos de autor, utilidad del esquema, etc.) al mismo.

Podemos pensar que un método para añadir esta información es utilizar comentarios. El problema es que los analizadores no garantizan que los comentarios no se modifiquen al procesar los documentos y por tanto, que los datos añadidos no se pierdan en algún proceso de transformación del documento.

En lugar de usar los comentarios, XML Schema tiene definido un elemento **xs:annotation** que permite **guardar información adicional**. Este elemento a su vez puede contener una combinación de otros dos que son:

- ✓ **xs:documentation**, además de contener elementos de esquema puede contener elementos XML bien estructurados.

  También permite determinar el idioma del documento mediante el atributo xml:lang.

- ✓ **xs:appinfo**, se diferencia muy poco del elemento anterior, aunque lo que se pretendió inicialmente era que xs:documentation fuese legible para los usuarios y que xs:appinfo guardase información para los programas de software.

  También es usado para generar una ayuda contextual para cada elemento declarado en el esquema.

Ejercicio resuelto

Ejemplo de documentación de un esquema.

```
<xs:schema xmlns:xsi=http://www.w3.org/2001/XMLSchema>

   <xs:annotation>
      <xs:documentation xml:lang ="es-es">
         Materiales para formación e-Learning
         <modulo>Lenguajes de marcas y sistemas de gestión de información.<modulo>
         <fecha_creación> 2011<fecha_creacion>
         <autor> Nuky La Bruji</autor>
      </xs:documentation>
   </xs:annotation>

   <xs:element name="lmsgi"  type=xs:string>
      <xs:annotation>
         <xs:appinfo>
            <texto_de_ayuda>Se debe de introducir el nombre completo del tema</texto_de_ayuda>
         <xs:appinfo>
      </xs:annotation>
   </xs:element>
</xs:schema>
```

Autoevaluación

La mejor solución para documentar un esquema es usar los comentarios:

- ○ Verdadero.
- ◉ Falso.

4 - Herramientas de creación y validación.

Caso práctico

*Antes de comenzar a trabajar con la **normalización de los documentos** que utiliza **la empresa de María y Félix. Marina** le presenta a **Juan** un informe sobre las diferentes herramientas que pueden facilitarles el trabajo de edición y validación de los documentos **XSD** y **XML**.*
***Juan** hará un estudio de costes y escogerá alguna de ellas para realizar el trabajo.*

Igual que hasta ahora, para crear y validar los documentos XML y los esquemas basta con un editor de texto plano y un navegador. ¿Pero no hay ninguna herramienta que nos facilite el trabajo? Pues sí, existen aplicaciones que permiten al usuario visualizar, validar y editar documentos en el lenguaje XML. Algunos de estos productos son:

- ✓ Editix XML Editor (Gratuito).
- ✓ Microsoft Core XML Services (MSXML) (Gratuito).
- ✓ XMLFox Advance.
- ✓ Altova XML Spy Edición Estándar.
- ✓ Editor XML xmlBlueprint.
- ✓ Editor Gráfico XSD y XML (XML Studio) (Gratuito).
- ✓ Estudio XML Líquido (Gratuito).
- ✓ Stylus Studio 2001 (Gratuito).
- ✓ Oxygen XML Editor.
- ✓ Exchanger XML Editor.

TEMA 5

Contenido

[CONVERSIÓN Y ADAPTACIÓN DE DOCUMENTOS XML]

José Luis Comesaña --- 2011/2012

Lenguajes de marca del curso de "*Desarrollo de Aplicaciones Web*"

Conversión y adaptación de documentos XML.

Caso práctico
La empresa de Félix y María continúa creciendo.
Dado que los documentos de los clientes de la empresa están en formato XML, Juan les plantea una nueva posibilidad para su página web. Dar la posibilidad a los clientes de consultar sus datos en la empresa a través de su página web.
Aunque a Félix y a María no les parece una idea muy seductora inicialmente, debido a que no consideran que el formato de los documentos XML sea el más atractivo para los clientes de la misma, deciden pedir consejo.
Una vez más, Juan les explica que existen unos lenguajes de marcas, en este caso Xpath y XSL, que les permitirán dar a la información de esos documentos el formato que quieran antes de publicarlo en una página segura de Internet.

Técnicas de transformación de documentos XML.

Caso práctico
Félix escucha atentamente las explicaciones de Juan sobre las posibilidades que da la transformación de documentos.
Gracias a ellas acaba de pensar que, además de permitir a los clientes consultar los datos a través de la web de la empresa, también puede servirles para generar informes en formato PDF, que pueden utilizar para informar a los clientes de sus negocios periódicamente, bien a través del correo ordinario o del electrónico.

Los documentos XML son documentos de texto con etiquetas que contienen exclusivamente información sin entrar en detalles de formato. Esto implica la necesidad de algún medio para expresar la transformación de un documento XML, para que una persona pueda utilizar directamente los datos para leer, imprimir, etc.

Las tecnologías que entran en juego en la transformación de documentos son:

- ✓ **XSLT**: (e***X***tensible ***S***tylesheet ***L***anguage ***T***ransformation)permite definir el modo de transformar un documento XML en otro.
- ✓ **XSL-FO**: (e***X***tensible ***S***tylesheet ***L***anguage - ***F***ormatting ***O***bject) Se utiliza para transformar XML en un formato legible e imprimible por una persona, por ejemplo en un documento PDF.
- ✓ **Xpath**: permite el acceso a los diversos componentes de un documento XML

La parte de transformaciones ganó en importancia, y se llega a la terminología actual que es XSL y comprende las anteriores.

Los lenguajes de marcas que no permiten la transformación de documentos son:

- [x] HTML.
- [x] XHTML.
- [x] XPath.
- [ ] XSL.

XPath.

Caso práctico

María dice que empieza a creer que todo es posible mediante el uso de los lenguajes de marcas.
Juan le dice que en lo que a intercambio de información se refiere si es así.
Su pregunta ahora es, ¿de qué modo van a lograr separar la información de los documentos XML de las etiquetas de los mismos?
Juan les explica que para eso existe un lenguaje llamado Xpath, cuya sintaxis es muy semejante a la que se usa para desplazarse a través de un árbol de directorios en modo comando.

Una expresión XPath es un predicado que se aplica sobre una estructura de árbol correspondiente a la jerarquía de los datos de un documento XML, y devuelve todo lo que encaja con ese predicado.

Es un estándar diferente de XML, aprobado por el W3C, que nos permite acceder a partes de un documento XML basándonos en las relaciones de parentesco entre los nodos del documento.

Su notación es similar a las de las rutas de los ficheros.

Inicialmente se creo para utilizarlo con XLST, pero en la actualidad se utiliza también con XML Schema, Xquery, Xlink, Xpointer, Xforms, etc.

XPath es un lenguaje XML que permite:

- Transformar el formato de los datos de un fichero XML.
- Definir un vocabulario que ha de cumplir un documento XML.
- Obtener los datos del fichero XML de una base de datos.
- Acceder a los datos de un fichero XML.

En el siguiente enlace puedes encontrar el estándar de Xpath que aprobó el W3C el 19 de Noviembre de 1999:
Recomendación del W3C sobre XPath. http://www.w3.org/TR/xpath

Términos básicos.

- **Nodo raíz**, es el nodo que contiene al ejemplar del fichero XML. No debe confundirse con el elemento raíz del documento, ya que éste último está por debajo de él.
  Se identifica por "/".
- **Nodos elemento**, son cada uno de los elementos del documento XML. Todos ellos tienen un elemento padre, el padre del elemento raíz, es decir del ejemplar, es el nodo raíz del documento.
  Pueden tener identificadores únicos, para ello es necesario que un atributo esté definido de ese modo en un DTD o un fichero XSD asociado, esto nos permite referenciar dicho elemento de forma mucho más directa.
- **Nodos texto**, son aquellos caracteres del documento que no están marcados con ninguna etiqueta. Este tipo de nodos no tienen hijos.
- **Nodos atributo**, no se consideran hijos del elemento al que están asociados sino etiquetas añadidas al nodo elemento.
  Aquellos atributos que tengan un valor asignado en el esquema asociado, se tratarán como si ese valor se le hubiese dado al escribir el documento XML.
  Para las definiciones de los espacios de nombre y para aquellos atributos que se han definido con la propiedad **#IMPLIED** en su **DTD** no se crean nodos.
- **Nodos de comentario y de instrucciones de proceso**, son los nodos que se generan para los elementos con comentarios e instrucciones de proceso.
- **Nodo actual**, es aquél al que nos referimos cuando se evalúa una expresión en Xpath.

- ✓ **Nodo contexto**, cada expresión está formada por subexpresiones que se van evaluando antes de resolver la siguiente. El conjunto de nodos obtenido tras evaluar una expresión y que se utiliza para evaluar la siguiente es el nuevo contexto.
- ✓ **Tamaño del contexto**, es el número de nodos que se están evaluando en un momento dado en una expresión Xpath.

El nodo raíz de un documento XML coincide con el ejemplar del mismo:

- ☐ Si.
- ☉ No.

Expresiones.

¿Cuáles son los resultados que da la evaluación de una expresión Xpath?

Podemos obtener cuatro tipos de resultados diferentes:

- ✓ Un conjunto de nodos que no está ordenado, **node-set.**. Se considera que todos los elementos de un conjunto de nodos son hermanos, independientemente de lo que fuesen originalmente. Aunque los hijos de los nodos que forman un conjunto de nodos son accesibles, los subárboles de un nodo no se consideran elementos del conjunto.
  Los nodos pueden ser de 7 tipos diferentes:
  - ➔ Elemento.
  - ➔ Atributo.
  - ➔ Texto.
  - ➔ Espacio de nombres.
  - ➔ Instrucción de procesamiento.
  - ➔ Comentario.
  - ➔ Raíz.
- ✓ **Booleano.**
- ✓ **Número.**
- ✓ **Cadena.**

¿Cuáles son los tokens que podemos utilizar en una expresión XPath?

- ✓ Paréntesis, “()”; llaves , “{}” y corchetes, “[ ]”.
- ✓ Elemento actual, elemento padre.
- ✓ Atributo, “@”.
- ✓ Elemento, “*”.
- ✓ Separador, “::”.
- ✓ Coma, “,”.
- ✓ El nombre de un elemento.
- ✓ Tipo de nodo, que puede ser:
  - ➔ comment.
  - ➔ text.
  - ➔ procesing instruction.
  - ➔ node.
- ✓ Operadores: and, or, mod, div, *, /, //, |, +, -, =, !=, <, >, <=, >=.
- ✓ Nombres de función.
- ✓ Denominación de ejes: ancestor, ancestor-or-self-atribute, child, descendant, descendant-or-self, following, following-sibling, namespace, parent, preceding, preceding-sibling, self.
- ✓ Literales, se ponen entre comillas dobles o simples. Pueden anidarse alternando el tipo de comillas.
- ✓ Números.
- ✓ Referencias a variables, para lo que se utiliza la sintaxis: **$nombreVariable**

Indica cuáles de los siguientes elementos de un documento XML pueden ser nodos del mismo:

- [x] Atributos.
- [x] Comentarios.
- [ ] Etiquetas.
- [x] Texto.

Ruta de localización.

Se corresponde con la ruta que hay que seguir en un árbol de datos para localizar un nodo.

La evaluación de una ruta de localización siempre devuelve un conjunto de nodos, aunque puede estar vacío.

¿Cómo podemos crear una ruta de localización? Mediante la unión de varios pasos de localización.

Las rutas de localización básicas son:

- ✓ **La ruta de localización del nodo raíz del documento**, es la barra diagonal "/". Se trata de una ruta absoluta, porque siempre significa lo mismo independientemente de la posición del procesador en el documento de entrada al aplicar la regla.
- ✓ **Localización de un elemento**, selecciona todos los hijos del nodo de contexto con el nombre especificado. Los elementos a los que se refiera dependerán de la localización del nodo de contexto, por lo que es una ruta relativa. En el caso de que el nodo contexto no tenga ningún nodo hijo con esa denominación, el valor de la ruta de localización será un elemento vacío.
- ✓ **Localización de atributos**, para referirnos a ellos en XPath, se utiliza el símbolo "@" seguido del nombre del atributo deseado.
- ✓ **Localización de espacios de nombres**, no se tratan explícitamente.
- ✓ **Localización de comentarios**.
- ✓ **Localización de nodos de texto.**
- ✓ **Localización de instrucciones de procesamiento.**

¿Cómo podemos comparar distintos elementos y tipos de nodo simultáneamente? Utilizando alguno de los tres comodines mostrados a continuación:

- ✓ **Asterisco "*"**: compara cualquier nodo de elemento, independientemente de su nombre. No compara ni atributos ni comentarios, nodos de texto o instrucciones de procesamiento.
- ✓ **Node()**: compara, además de los tipos de elementos, el nodo raíz, los nodos de texto, los de instrucción de procesamiento, los nodos de espacio de nombre, los de atributos y los de comentarios.
- ✓ **"@*"**: compara los nodos de atributo.

Las rutas de localización:

- [ ] Devuelven todos los nodos de un documento XML que verifican una condición dada.
- [x] Devuelven todos los nodos de un árbol XML que verifican una condición dada.
- [ ] El primer nodo que se encuentra y cumple una condición dada.
- [ ] No pueden devolver valores de atributos.

Predicado.

Es una expresión booleana que añade un nivel de verificación al paso de localización.

En estas expresiones podemos incorporar funciones XPath

¿Qué ventajas nos da el uso de predicados?

Mediante las rutas de localización se pueden seleccionar varios nodos a la vez, pero el uso de predicados permite seleccionar un nodo que cumple ciertas características.

Los predicados se incluyen dentro de una ruta de localización utilizando los corchetes, por ejemplo:

/receta/ingredientes/ingrediente[@codigo="1"]/nombre

En este caso se está indicando al intérprete que escoja, dentro de un fichero XML de recetas, el nombre del ingrediente cuyo código tiene el valor "1".

Veamos qué es lo que podemos incluir en un predicado.

- ✓ **Ejes**, permiten seleccionar el subárbol dentro del nodo contexto que cumple un patrón. Pueden ser o no de contenido.
  - ➔ **child**, es el eje utilizado por defecto. Su forma habitual es la barra, "/", aunque también puede ponerse: /child::
  - ➔ **attribute**, permite seleccionar los atributos que deseemos. Es el único eje que veremos que no es de contenido.
  - ➔ **descendant**, permite seleccionar todos los nodos que descienden del conjunto de nodos contextos. Se corresponde con la doble barra, //, aunque se puede usar: descendant::
  - ➔ **self**, se refiere al nodo contexto y se corresponde con el punto ".".
  - ➔ **parent**, selecciona los nodos padre, para referirnos a él usamos los dos puntos, "..".
  - ➔ **ancestor**, devuelve todos los nodos de los que el nodo contexto es descendiente.
- ✓ Nodos test, permiten restringir lo que devuelve una expresión Xpath. Podemos agruparlos en función de los ejes a los que se puede aplicar.
  - ➔ Aplicable a cualquier eje:
    - → **"*"**, solo devuelve elementos, atributos o espacios de nombres pero no permite obtener nodos de texto, o comentarios de cualquier tipo.
    - → **nod()**, devuelve los nodos de todos los tipos.
  - ➔ Solo aplicables a ejes de contenido:
    - → **text()**, devuelve cualquier nodo de tipo texto.
    - → **comment()**, devuelve cualquier nodo de tipo comentario.
    - → **processing-instruction()**, devuelven cualquier tipo de instrucción de proceso.

Varios predicados pueden unirse mediante los operadores lógicos **and**, **or** o **not**.

Relaciona los ejes siguientes con sus equivalentes.

Eje.	Relación.	Representación.
child::	2	1. @
descendant::	3	2. /
attribute::	1	3. //
parent::	4	4. ..

Funciones de Xpath.

Entre las funciones más importantes que podemos utilizar en Xpath destacan:

- ✓ **boolean()**, al aplicarla sobre un conjunto de nodos devuelve true si no es vacío.

- **not()**, al aplicarla sobre un predicado devuelve true si el predicado es falso , y falso si el predicado es true.
- **true()**, devuelve el valor true.
- **false()**, devuelve el valor false.
- **count()**, devuelve el número de nodos que forman un conjunto de nodos.
- **name()**, devuelve un nombre de un nodo.
- **local-name()**, devuelve el nombre del nodo actual o del primer nodo de un conjunto de nodos.
- **namespace-uri()**, devuelve el URI del nodo actual o del primer nodo de un conjunto dado.
- **position()**, devuelve la posición de un nodo en su contexto comenzando en 1. Por ejemplo, para seleccionar los dos primeros elementos de tipo elemento de un fichero XML pondremos: **//elemento[position()<=2**]
- **last()**, Devuelve el último elemento del conjunto dado.
- **normalize-space()**, permite normalizar los espacios de una cadena de texto, es decir, si se introduce una cadena donde hay varios espacios consecutivos, esta función lo sustituye por uno solo.
- **string()**, es una función que convierte un objeto en una cadena. Los valores numéricos se convierten en la cadena que los representa teniendo en cuenta que los positivos pierden el signo. Los valores booleanos se convierten en la cadena que representa su valor, esto es “true” o “false”.
- **concat()**, devuelve dos cadenas de texto concatenadas. El ejemplo siguiente devuelve “Xpath permite obtener datos de un fichero XML”.
- concat('XPath', 'permite obtener datos de un fichero XML')
- **string-length()**, devuelve la cantidad de caracteres que forman una cadena de caracteres.
- **sum()**, devuelve la suma de los valores numéricos de cada nodo en un conjunto de nodos determinado.

Para lograr ir directamente al último de los elementos hijos de un elemento en el que nos encontramos habrá que usar la función:

- position(end)
- last()
- first()
- end()

Acceso a atributos.

¿Cómo podemos acceder a los atributos?

Utilizando el eje **attribute**:, que puede sustituirse por la arroba, “**@**”.

Ejercicio resuelto

Dado el siguiente fichero XML

```
<?xml version="1.0" encoding="iso-8859-1" standalone="yes"?>
<!DOCTYPE agenda>
<agenda>
   <propietario>
      <identificadores>
         <nombre>Alma</nombre>
         <apellidos>López Terán</apellidos>
      </identificadores>

      <direccion>
         <calle>El Percebe 13, 6F</calle>
         <localidad>Torrelavega</localidad>
         <cp>39300</cp>
      </direccion>

      <telefonos>
         <movil>970898765</movil>
```

```
         <casa>942124567</casa>
         <trabajo>628983456</trabajo>
      </telefonos>
   </propietario>

   <contactos>
      <persona id="p01">
         <identificadores>
            <nombre>Inés</nombre>
            <apellidos>López Pérez</apellidos>
         </identificadores>

         <direccion>
            <calle>El Ranchito 24, 6B</calle>
            <localidad>Santander</localidad>
            <cp>39006</cp>
         </direccion>

         <telefonos>
            <movil>970123123</movil>
         </telefonos>
      </persona>

      <persona id="p02">
         <identificadores>
            <nombre>Roberto</nombre>
            <apellidos>Gutiérrez Gómez</apellidos>
         </identificadores>

         <direccion>
            <calle>El Marranito 4, 2F</calle>
            <localidad>Santander</localidad>
            <cp>39004</cp>
         </direccion>

         <telefonos>
            <movil>970987456</movil>
            <casa>942333323</casa>
         </telefonos>
      </persona>

      <persona id="p03">
         <identificadores>
            <nombre>Juan</nombre>
            <apellidos>Sánchez Martínez</apellidos>
         </identificadores>

         <direccion>
            <calle>El Cangrejo 10, sn</calle>
            <localidad>Torrelavega</localidad>
            <cp>39300</cp>
         </direccion>

         <telefonos>
            <movil>997564343</movil>
            <casa>942987974</casa>
            <trabajo>677899234</trabajo>
         </telefonos>
      </persona>
   </contactos>
</agenda>
```

Construir las sentencias Xpath que permitan obtener los siguientes datos:

- ✓ Nombre del propietario de la agenda.
- ✓ Teléfono de casa del propietario.
- ✓ Nombres y apellidos de los contactos de la agenda.
- ✓ Nombre e identificador de cada contacto.
- ✓ Datos del contacto con identificador "p02".
- ✓ Identificadores de los contactos que tienen móvil.

Respuesta:

1. Nombre del propietario de la agenda.

→ **/agenda/propietario/identificadores/nombre**

2. Teléfono de casa del propietario.

→ **/agenda/propietario/telefonos/casa**

3. Nombres y apellidos de los contactos de la agenda.

→ **//contactos/persona/identificadores/nombre | //contactos/persona/identificadores/apellidos**

4. Nombre e identificador de cada contacto.

→ **//contactos/persona/identificadores/nombre | //contactos/persona/@id**

5. Datos del contacto con identificador "p02".

→ **//contactos/persona[@id="p02"]/*/***

6. Identificadores de los contactos que tienen teléfono en casa.

→ **//contactos/persona/telefonos/casa/../../@id**

Acceso a elementos de otro documento XML.

Además de acceder a los datos del fichero XML con el que se está trabajando directamente, es útil poder acceder a los datos de otros ficheros.

Para ello utilizaremos la función **document()**, pero dicha función **NO** es del lenguaje XPath, sino que pertenece a XSLT.

Esta función puede admitir dos argumentos diferentes:

document(URI)

En este caso la función devuelve el elemento raíz del documento XML que se localiza en el URI especificado.

document(nodo)

En esta ocasión lo que devuelve la función, es el conjunto de nodos cuya raíz es el nodo dado.

Indica cuál de estas afirmaciones es correcta:

- ☐ Para aplicar una ruta de XPath a un documento diferente de aquel con el que se trabaja en un momento dado el lenguaje nos proporciona la función document().
- ☐ La función document() devuelve únicamente una rama de un nodo dado.
- ☐ No existe ningún modo de acceder a los elementos de varios documentos.
- ☑ XPath no proporciona un modo de acceder a datos de varios documentos simultáneamente.

XSLT.

Caso práctico

Una vez que Juan ha resuelto el modo de separar los datos de las etiquetas. Félix pregunta si van a usar las hojas de estilo para darles formato, del mismo modo que lo hacen para crear la página web.
Juan les explica que es algo semejante, pero que, en este caso, utilizarán otro lenguaje de marcas llamado XSLT.

XSLT es un estándar aprobado por el W3C, ¿pero una hoja XSLT es también un documento XML?

Sí, XSLT es uno de los lenguajes derivados de XML, por tanto las hojas XSLT también son documentos XML (al igual que sucede con los canales RSS, atom o los documentos XSD).

¿Qué transformaciones podemos realizar sobre un documento XML usando XSLT?

- ✓ A otro documento XML.
- ✓ A un documento HTML.
- ✓ A un documento de texto.

En los siguientes enlaces puedes encontrar el estándar de las diferentes versiones que existen de XSLT aprobadas por el W3C

Recomendación del W3C sobre XSLT version 1.0.	http://www.w3.org/TR/xslt
Recomendación del W3C sobre XSLT version 1.1.	http://www.w3.org/TR/xslt11/
Recomendación del W3C sobre XSLT version 2.0.	http://www.w3.org/TR/xslt20/

Estructura básica de una hoja XSLT.

¿Cuáles son los elementos contenidos en una hoja XSLT?

Básicamente hay tres tipos de elementos:

- ✓ **Elementos XSLT**, están precedidos del prefijo xsl:, pertenecen al espacio de nombres xsl, están definidos en el estándar del lenguaje y son interpretados por cualquier procesador XSLT.
- ✓ **Elementos LRE** (***L***iteral ***R***esult ***E***lements), no pertenecen a XSLT, sino que se repiten en la salida sin más.
- ✓ **Elementos de extensión**, al igual que los anteriores, no pertenecen al espacio de nombres xsl. Son manejados por implementaciones concretas del procesador. Estos normalmente no son usados.

Ahora que sabemos cómo es la estructura de un fichero XSLT, nos preguntamos si para asociar un fichero XML a un XSLT hay que incluir en ellos algún elemento, como se hace con los vocabularios.

La respuesta es afirmativa. Para indicar que un fichero XML está asociado a un XSLT hay que añadir, después del prólogo, una línea como la que sigue:

```
<?xml-stylesheet type="text/xsl" href="ruta_del_fichero_xsl.xsl"?>
```

Con ella se indica al procesador el lugar donde se encuentra la hoja de estilos con la que dar formato a los datos del documento.

Cuáles de los siguientes tipos de documentos pueden obtenerse a partir de la transformación XSL de un documento XML:

- ☐ doc.
- ☑ HTML.
- ☐ PDF.
- ☑ Texto.

Elementos XSLT.

El elemento raíz de una hoja XSLT es **xsl:stylesheet** o **xsl:transform**, que son prácticamente equivalentes. Sus atributos principales son:

- ✓ **version**, cuyo valor puede ser 1.0 o 2.0.
- ✓ **xmlns:xsl**, se utiliza para declarar el espacio de nombres xsl. Para XSLT suele ser la dirección:

http://www.w3.org/1999/XSL/Transform

A los elementos hijos de estos se les conoce como elementos de nivel superior, son estructuras contenedoras de instrucciones. Dado que si son hijos directos no pueden anidarse, excepto **xsl:variable** y **xsl:param**. Los más destacados son:

- ✓ **xsl:attribute**, añade un atributo a un elemento en el árbol de resultados.
- ✓ **xsl:choose**, permite decidir que parte de la hoja XSL se va a procesar en función de varios resultados.
- ✓ **xsl:decimal-format**, define un patrón que permite convertir en cadenas de texto números en coma flotante.
- ✓ **xsl:for-each**, aplican sentencias a cada uno de los nodos del árbol que recibe como argumento.
- ✓ **xsl:if**, permite decidir si se va a procesar o no una parte del documento XSL en función de una condición
- ✓ **xsl:import**, importa una hoja de estilos XSLT localizada en una URI dada.
- ✓ **xsl:key**, define una o varias claves que pueden ser referenciadas desde cualquier lugar del documento.
- ✓ **xsl:output**, define el tipo de salida que se generará como resultado.
- ✓ **xsl:preserve-space**, especifica cuales son los elementos del documento XML que no tienen espacios en blanco eliminados antes de la transformación.
- ✓ **xsl:sort** permite aplicar un template a una serie de nodos ordenándolos alfabético numéricamente.
- ✓ **xsl:strip-space**, especifica cuales son los elementos del documento XML que tienen espacios en blanco eliminados antes de la transformación.
- ✓ **xsl:template**, es el bloque fundamental de una hoja XSLT, por lo que veremos su descripción en el apartado siguiente.
- ✓ **xsl:value-of**, calcula el valor de una expresión XPath dada y lo inserta en el árbol de resultados del documento de salida.
- ✓ **xsl:variable**, asigna un valor a una etiqueta para usarlo cómodamente.

Relaciona los siguientes elementos con sus funcionalidades.

Elemento	Relación	Funcionalidad
preserve-space.	4	1. Convierte datos numéricos en cadenas de texto.
value-of	3	2. Marca aquellos elementos que tienen espacios blancos eliminados antes de la transformación.
strip-space	2	3. Resuelve una expresión XPath dada.
decimal-format	1	4. Marca aquellos elementos que no tienen espacios blancos eliminados antes de la transformación.

Utilización de plantillas.

El elemento **xsl:template** permite controlar el formato de salida que se aplica a ciertos datos de entrada. Dicho formato se especifica utilizando sentencias XHTML

Tiene un atributo denominado **match**, que se utiliza para seleccionar los nodos del árbol de entrada.

Un ejemplo simple del uso de este elemento es, siempre utilizando como fuente de datos el fichero XML que se ha descrito en el apartado 2.6 de este tema:

```
<xsl:template match=”propietario”>
   <p>Agenda de </p>
</xsl:template>
```

Donde el elemento < p> es el elemento párrafo de XHTML.

Hay que tener en cuenta que el contenido del elemento template ha de ser XHTML bien estructurado.

En una plantilla podemos tener más de una regla que afecte a distintos elementos. ¿Cuál es el orden de aplicación de estas reglas de formato? Por defecto ese orden es el que el interprete utiliza al leer el documento XML, es decir, de arriba abajo, aunque dicho orden puede ser modificado en la plantilla, para ello utilizaremos el elemento **xsl:apply-templates**. Un ejemplo del uso es:

```
<xsl:template match=”contactos”>
   <xsl:apply-templates select=”identificadores”/>
</xsl:template>
```

Con esto la plantilla solo se aplicará sobre los datos incluidos dentro del elemento **`identificadores`** y no sobre los elementos **`direccion`** o **`telefonos`**.

Ejercicio resuelto

Veamos un ejemplo de una hoja de estilos completa.

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform" version="2.0">

   <xsl:template match="identificadores">
      <xsl:value-of select="nombre"/>,
      <xsl:value-of select="apellidos"/>
   </xsl:template>

   <xsl:template match="persona">
      <xsl:apply-templates select="identificadores"></xsl:apply-templates>
   </xsl:template>
</xsl:stylesheet>
```

Cuando un procesador XSLT aplica esta hoja de estilos al fichero XML siguiente:

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

<!DOCTYPE agenda>

<?xml-stylesheet type="text/xsl" href="./LMSGI_CONT_Ejemplo02.xsl"?>

<agenda>
   <persona id="p01">
      <identificadores>
         <nombre>Inés</nombre>
         <apellidos>López Pérez</apellidos>
      </identificadores>

      <direccion>
         <calle>El Ranchito 24, 6B</calle>
         <localidad>Santander</localidad>
         <cp>39006</cp>
      </direccion>

      <telefonos>
         <movil>970123123</movil>
      </telefonos>
   </persona>

   <persona id="p02">
      <identificadores>
         <nombre>Roberto</nombre>
         <apellidos>Gutiérrez Gómez</apellidos>
      </identificadores>

      <direccion>
         <calle>El Marranito 4, 2F</calle>
```

```
         <localidad>Santander</localidad>
         <cp>39004</cp>
      </direccion>

      <telefonos>
         <movil>970987456</movil>
         <casa>942333323</casa>
      </telefonos>
   </persona>
</agenda>
```

Procesadores XSLT.

Un procesador XSLT es un software que lee un documento XSLT y otro XML, y crea un documento de salida aplicando las instrucciones de la hoja de estilos XSLT a la información del documento XML.

Pueden estar integrados dentro de un explorador Web, como MSXML en IE 6, en un servidor web, como el Cocoon de Apache XML o puede ser un programa que se ejecuta desde la línea de comandos, como Xalan de Apache o SAXON.

Existen diferentes modos de realizar la transformación XSLT

- ✓ Mediante el procesador MSXML, que es un ejecutable que se limita a llamar a la biblioteca de transformación de Internet Explorer.
- ✓ Usando un procesador XSLTPROC..
- ✓ Invocando a la biblioteca de transformación desde un programa.
- ✓ Realizando un enlace entre la hoja XSLT y el documento XML, en este caso hay que añadir, en el fichero XML entre el la definición de la versión XML y la definición del tipo de documento, la línea:

```
<?xml-stylesheet type="text/xsl" href="path_hoja_xsl"?>
```

El fichero puede verse directamente desde cualquier navegador que soporte XSLT, aunque tiene la desventaja de que queda ligado a esa vista.

La mayoría de editores XML, como Oxygen, Editix o Estudio XML Líquido permiten escoger el intérprete que debe encargarse de procesar un documento XSLT.

El único modo de generar un documento a partir de una transformación XSLT de otro es utilizando un editor XML que tenga incorporado un procesador XSLT:

- ◉ No.
- ○ Si.

En los siguientes enlaces puedes encontrar algunos de los procesadores de XSLT:

Xalan	http://xml.apache.org/xalan-j/
Saxon	http://saxon.sourceforge.net/

Depuración.

Los depuradores son elementos de software que permiten seguir la generación de un documento, a partir de los datos de un documento XML aplicándoles una hoja de estilo XSLT.

Los depuradores XSLT nos facilitan la localización y corrección de errores dejándonos ejecutar el código paso a paso, salir, ejecutar el código hasta el cursor, ejecutar hasta el final, pausar y detener.

Los editores de XML, como Oxygen XML o Editix, incluyen un depurador que permite visualizar simultáneamente la plantilla que se está ejecutando y sobre qué datos del documento XML actúan y cuál es la salida que genera dicha orden.

De este modo se facilita averiguar cuáles son las plantillas que no dan lugar al formato que deseamos en la salida del documento.

TEMA 6

Contenido

[ALMACENAMIENTO DE LA INFORMACIÓN]

José Luis Comesaña --- 2011/2012

Lenguajes de marca del curso de "*Desarrollo de Aplicaciones Web*"

Almacenamiento de la información.

Caso práctico
*Antes de irse de vacaciones con su familia, **María y Félix tienen una reunión con el responsable del departamento de informática de la empresa, Juan**, en la que éste último les explica que aún se puede mejorar la gestión de datos de la empresa.*
*Juan opina que ya que en la empresa **se ha impuesto el uso de la tecnología XML**, sería mejor utilizar un sistema de almacenamiento y consulta de datos compatible con dicha tecnología.*
Van a intentar utilizar un sistema de almacenamiento de datos, diferente de la base de datos relacional que han usado hasta ahora, que es compatible con la tecnología XML que están utilizando.
*Además, para acceder a los datos de esta base de datos nativa van a utilizar un lenguaje de consultas llamado **XQuery**, que equivale al SQL usado con las bases de datos relacionales.*

Utilización de XML para almacenamiento de información.

Caso práctico
*Mientras **María** está de vacaciones con su familia le **explica a su marido, José Ramón**, las nuevas **modificaciones** que va a realizar Juan en el sistema informático de la empresa.*
Comienza por explicarle lo que es un sistema de almacenamiento, su relación con la tecnología XML y los distintos ámbitos de aplicación que pueden tener.

Como ya hemos visto, XML, es un estándar potente y de amplia aceptación para guardar y comunicar información acerca de objetos. Permite la codificación de información separada de la forma en la que se debe presentar al usuario. Para encontrar un fragmento específico de información en los contenidos de un nodo o atributo XML, habrá que procesar completamente el archivo XML.

Una base de datos XML se puede ver como una colección de documentos XML, en la que cada documento representa un registro.

Cada documento XML es un archivo en el sistema de archivos y contiene una cadena válida XML.

La estructura de un documento XML suele seguir un mismo esquema XML, aunque no es necesario que sea así. Este es uno de los beneficios de las bases de datos XML, ya que cada archivo se puede configurar de forma estructurada por lo que es independiente, pero fácilmente accesible.

El tener colecciones de documentos con un esquema independiente proporciona a la base de datos flexibilidad, y facilita el desarrollo de la aplicación.

En los últimos años, se ha visto necesaria la existencia de estándares de intercambio de información, con el objetivo de que las organizaciones puedan compartir su información de una manera más cómoda, automática y eficiente.

La tecnología XML:

- [x] **Permite tratar a los ficheros como si fuesen base de datos..**
- [x] **Facilita el desarrollo de las aplicaciones al tener el esquema independiente de los datos..**
- [x] **Hace que el compartir información sea más cómodo.**
- [ ] Permite el acceso directo a los datos buscados, es decir sin recorrer todo el archivo.

Ámbitos de aplicación.

Los documentos y los requerimientos de almacenamiento de datos XML pueden ser agrupados en dos categorías generales:

- ✓ **Centrados en los datos.**

Suelen incluir documentos de facturación típicos, órdenes de compra, y documentos menos estructurados. Es apropiada para ítems como contenidos de periódicos, artículos y publicidad.

Si el documento XML tiene una estructura bien definida y contiene datos que pueden ser actualizados y usados de diversos modos, el documento es habitualmente centrado en los datos.

- ✓ **Centrados en los documentos.**

Tienden a ser más impredecibles en tamaño y contenido que los centrados en los datos, los cuales son altamente estructurados, con tipos de datos de tamaño limitado y reglas menos flexibles para campos opcionales y contenido.

Los sistemas de almacenamiento XML deben acomodarse eficientemente con ambos tipos de requerimientos de datos, dado que XML está siendo usado en sistemas que administran ambos tipos de datos. La mayoría de los productos se enfocan en servir uno de esos formatos de datos mejor que el otro.

Las bases de datos relacionales tradicionales son mejores para tratar con requerimientos centrados en los datos, mientras que los sistemas de administración de contenido y de documentos, suelen ser mejores para almacenar datos centrados en el documento.

Los sistemas de bases de datos deben ser capaces de exponer los datos relacionales como XML, y almacenar el XML recibido como datos relacionales para transferir, obtener y almacenar los datos requeridos por la aplicación.

Elige las características que se ajusten a los sistemas centrados en datos:

- ☑ **Contiene datos con una estructura no muy definida.**
- ☑ **Es apropiada para publicidad.**
- ☑ **Los datos pueden actualizarse.**
- ☐ Contiene datos con una estructura muy definida.

Sistemas de almacenamiento de información.

Caso práctico

Mientras María disfruta de sus vacaciones Félix y Juan estudian si el cambio propuesto por este último es viable.

Para ello lo primero que hacen es informarse sobre los distintos sistemas de almacenamiento posibles y sus ventajas a la hora de utilizarlos con la tecnología XML.

En este proceso ***Félix descubre que, además de las bases de datos relacionales, existen las orientadas a objetos y las nativas.***

Una expresión XPath es un predicado que se aplica sobre una estructura de árbol correspondiente a la jerarquía de los datos de un documento XML y devuelve todo lo que encaja con ese predicado.

En lo que concierne a las bases de datos, XML permite integrar sistemas de información hasta ahora separados:

- ✓ **Sistemas de información basados en documentos** (ficheros), tienen estructura irregular, utilizan tipos de datos relativamente simples y dan gran importancia al orden.

- ✓ **Sistemas de información estructurados** (bases de datos relacionales), son relativamente planos, utilizan tipos de datos relativamente complejos y dan poca importancia al orden.

Podemos establecer las siguientes semejanzas entre una base de datos y un fichero XML con su esquema asociado:

- ✓ La tecnología XML usa uno o más documentos para almacenar la información.
- ✓ Define esquemas sobre la información.
- ✓ Tiene lenguajes de consulta específicos para recuperar la información requerida.
- ✓ Dispone de APIs (SAX, DOM).

Pero aparecen muchas más cosas que lo diferencian. Debido a que no es una base de datos, la tecnología XML carece, entre otras cosas, tanto de almacenamiento y actualización eficientes como de índices, seguridad, transacciones, integridad de datos, acceso concurrente, disparadores, etc.; que son algunas de las características habituales en las bases de datos. Por tanto es imposible pensar que XML se vaya a utilizar para las tareas transaccionales de una organización para las cuales sigue estando sobradamente más justificado utilizar una base de datos.

Marcar las afirmaciones que son correctas sobre bases de datos XML:

- ☑ **Integran sistemas de información basados en documentos.**
- ☑ **Integran sistemas de información estructurados.**
- ☑ **La información se estructura siguiendo esquemas definidos previamente.**
- ☐ Garantiza la actualización eficiente.

XML y BD relacionales.

Las bases de datos relacionales se basan en las relaciones (tablas bidimensionales), como único medio para representar los datos del mundo real. Están asociadas al lenguaje estándar SQL.

Se han creado complejas teorías y patrones para encajar objetos o estructuras jerarquizadas en bases de datos relacionales.

Existen numerosos middleware (*Es un software que permite la comunicación entre dos aplicaciones de software independientes. Por ejemplo, un middleware puede permitir a una base de datos acceder a los datos de otra*) encargados de la transferencia de información entre estructuras XML y bases de datos relacionales.

Las bases de datos relacionales suponen una posibilidad para el almacenamiento de datos XML. Sin embargo, no están bien preparadas para almacenar estructuras de tipo jerárquico como son los documentos XML, algunas de las causas son:

- ✓ Las bases de datos relacionales tienen una estructura regular frente al carácter heterogéneo de los documentos XML.
- ✓ Los documentos XML suelen contener muchos niveles de anidamiento mientras que los datos relacionales son planos.
- ✓ Los documentos XML tienen un orden intrínseco mientras que los datos relacionales son no ordenados.
- ✓ Los datos relacionales son generalmente densos (cada columna tiene un valor), mientras que los datos XML son dispersos, es decir, pueden representar la carencia de información mediante la ausencia del elemento.

Algunas de las razones para usar los tipos de bases de datos relacionales y los productos de bases de datos existentes para almacenar XML, aún cuando no sea de forma nativa son:

- ✓ Las bases de datos relacionales y orientadas a objetos son bien conocidas, mientras que las bases de datos XML nativas son nuevas.
- ✓ Como resultado de la familiaridad con las bases de datos relacionales y orientadas a objetos, los usuarios se inclinan a ellas especialmente por el rendimiento.

El hecho de que el uso de las bases de datos relacionales esté muy extendido es uno de los motivos por los que se utilizan para almacenar datos XML.

- ○ No
- ◉ **Si**

Reglas de transformación relacional. DTD.

La estructura XML debe ser capaz de acomodar algunos conceptos básicos, incluyendo claves primarias, claves secundarias, tablas, y columnas.

El proceso de traducción puede ser descompuesto en los siguientes pasos básicos:

- ✓ **Crear el esquema XML** con un elemento para cada tabla y los atributos correspondientes para cada columna no clave. Las columnas que no permiten valores nulos pueden ser marcadas como requeridas, mientras que aquellas que permiten valores nulos pueden ser marcadas como opcionales en el esquema XML. Las columnas pueden ser también anidadas como elementos, pero pueden surgir problemas cuando el mismo nombre de columna es usado en más de una tabla. Por ello, lo más simple es transformar las columnas como atributos XML, donde las colisiones de nombre en el esquema XML no son un problema.
- ✓ **Crear las claves primarias en el esquema XML**. Una solución podría ser agregar un atributo para la columna clave primaria, con un ID agregado al nombre de la columna. Este atributo podría necesitar ser definido en el esquema XML como de tipo ID. Pueden surgir problemas de colisión al crear claves primarias en el esquema XML, ya que a diferencia de las bases de datos relacionales, donde las claves primarias necesitan ser únicas sólo dentro de una tabla, un atributo ID dentro de un documento XML debe ser único a través de todo el documento.

Para resolverlo se puede agregar el nombre del elemento (nombre de la tabla), al valor de la clave primaria (valor del atributo). Esto asegura que el valor es único a través del documento XML.

- ✓ **Establecer las relaciones de clave migrada**. Esto se puede lograr mediante el anidamiento de elementos bajo el elemento padre, un ID de esquema XML puede ser usado para apuntar a una estructura XML correspondiente conteniendo un IDREF.

Pueden existir muchas variaciones de esquemas XML para representar la misma base de datos relacional.

Relaciona los siguientes pasos con su orden a la hora de transformar una base de datos relacional a almacenamiento XML.

Paso	Relación	Orden
Establecer las relaciones de clave migrada.	3	1.- Primer lugar.
Crear el esquema XML	1	2.- Segundo lugar.
Crear las claves primarias	2	3.- Tercer lugar.

XML y Bases de Datos Orientadas a Objetos.

Las **bases de datos orientadas a objetos** soportan un modelo de objetos puro, en el sentido de que no están basados en extensiones de otros modelos más clásicos como el relacional:

- ✓ Están influenciados por los lenguajes de programación orientados a objetos.
- ✓ Pueden verse como un intento de añadir la funcionalidad de un SGBD a un lenguaje de programación.
- ✓ Son una alternativa para el almacenamiento y gestión de documentos XML.

Componentes del estándar:

- ✓ **Modelo de Objetos.** Está concebido para proporcionar un modelo de objetos estándar para las bases de datos orientadas a objetos. Es el modelo en el que se basan el lenguaje de definición de objetos y el lenguaje de consultas.
- ✓ **Lenguajes de Especificación de Objetos.** ODL
- ✓ **Lenguaje de Consulta.** Conocido como OQL .
- ✓ **'Bindings'** (*En programación se denomina binding a toda adaptación que se hace en un conjunto de programas, utilizados en desarrollo de software, conocido como librería o biblioteca; para poder utilizarles al trabajar en un lenguaje de programación diferente a aquel en el que se codificaron*) **para C++, Java y Smalltalk.** Definen un OML que extiende el lenguaje de programación para soportar objetos persistentes (*En programación es un objeto que tiene la particularidad de que sus propiedades se almacenan en un medio secundario. Esto implica que el tiempo de vida del objeto es independiente del proceso que le creo*). Además incluye soporte para OQL, navegación y transacciones.

Una vez transformado el documento XML en objetos, éstos son gestionados directamente por el SGBDOO. Dicha información se consulta acudiendo al lenguaje de consulta OQL. Los mecanismos de indexación, optimización, procesamiento de consultas, etc. son los del propio SGBDOO, y por lo general, no son específicos para el modelo XML.

Las rutas de localización:

- ☐ SQL
- ☉ **OQL**
- ☐ XQuery
- ☐ XQL

XML y Bases de Datos Nativas.

Son bases de datos, y como tales soportan transacciones, acceso multi-usuario, lenguajes de consulta, etc. Están diseñadas especialmente para almacenar documentos XML.

Las BD nativas se caracterizan principalmente por:

- ✓ **Almacenamiento de documentos en colecciones.** Las colecciones juegan en las bases de datos nativas el papel de las tablas en las BD relacionales.
- ✓ **Validación de los documentos.**
- ✓ **Consultas.** La mayoría de las BD XML soportan uno o más lenguajes de consulta. Uno de los más populares es XQuery.
- ✓ **Indexación XML.** Se ha de permitir la creación de índices que aceleren las consultas realizadas.
- ✓ **Creación de identificadores únicos.** A cada documento XML se le asocia un identificador único.
- ✓ **Actualizaciones y Borrados**.

Según el tipo de almacenamiento utilizado pueden dividirse en dos grupos:

- ✓ **Almacenamiento Basado en Texto.** Almacena el documento XML entero en forma de texto y proporciona alguna funcionalidad de base de datos para acceder a él. Hay dos posibilidades:
  - ➔ **Posibilidad 1**: Almacenar el documento como un BLOB en una base de datos relacional, mediante un fichero, y proporcionar algunos índices sobre el documento que aceleren el acceso a la información.

- ➔ **Posibilidad 2**: Almacenar el documento en un almacén adecuado con índices, soporte para transacciones, etc.

✓ **Almacenamiento Basado en el modelo.** Almacena un modelo binario del documento (por ejemplo, DOM) en un almacén existente o bien específico.

- ➔ **Posibilidad 1**: Traducir el DOM a tablas relacionales como Elementos, Atributos, Entidades, etc.
- ➔ **Posibilidad 2**: Traducir el DOM a objetos en una BDOO.
- ➔ **Posibilidad 3**: Utilizar un almacén creado especialmente para esta finalidad.

Selecciona aquellas características de las bases de datos nativas:

- ☐ Triggers
- ☑ **Consultas**
- ☑ **Creación de índices**
- ☑ **Creación de identificadores únicos**

XQuery

Caso práctico

Una vez que María vuelve de vacaciones Juan y Félix le ponen al día de sus estudios.
La única cuestión que ***queda por decidir es el modo de acceder a los datos guardados en los archivos XML.***
Juan les explica que existe un lenguaje semejante al SQL, con el que están familiarizados, llamado ***XQuery*** *y que a su vez está basado en* ***XPath****.*

XQuery es un lenguaje diseñado para escribir consultas sobre colecciones de datos expresadas en XML. Puede aplicarse tanto a archivos XML, como a bases de datos relacionales con funciones de conversión de registros a XML. Su principal función es extraer información de un conjunto de datos organizados como un árbol de etiquetas XML. En este sentido XQuery es independiente del origen de los datos.
Permite la construcción de expresiones complejas combinando expresiones simples de una manera muy flexible.

De manera general podemos decir que XQuery es a XML lo mismo que SQL es a las bases de datos relacionales. Al igual que éste último, XQuery es un lenguaje funcional (*Lenguaje en el que cada consulta es una expresión que es evaluada y devuelve un resultado*).

Los r**equerimientos técnicos** más importantes **de XQuery** se detallan a continuación:

- ✓ Debe ser un **lenguaje declarativo** (*Lenguaje en el que hay que indicar qué se quiere, no la manera de obtenerlo*).
- ✓ Debe ser i**ndependiente del protocolo de acceso** a la colección de datos. Esto significa que una consulta en XQuery, debe funcionar igual al consultar un archivo local, que al consultar un servidor de bases de datos, o que al consultar un archivo XML en un servidor web.
- ✓ Las consultas y los resultados deben respetar el **modelo de datos XML**.
- ✓ Las consultas y los resultados deben ofrecer **soporte para los namespaces.**
- ✓ Debe soportar **XML-Schemas y DTDs** y también debe ser capaz de trabajar sin ellos.
- ✓ Ha de ser **independiente de la estructura** del documento, esto es, funcionar sin conocerla.
- ✓ Debe soportar **tipos simples**, como enteros y cadenas, y **tipos complejos**, como un nodo compuesto.
- ✓ Las consultas deben soportar **cuantificadores universales** (para todo) y existenciales (existe).
- ✓ Las consultas deben soportar **operaciones sobre jerarquías de nodos** y secuencias de nodos.
- ✓ Debe ser posible **combinar información de múltiples fuentes** en una consulta.
- ✓ Las consultas deben ser capaces de **manipular los datos independientemente del origen** de estos.

- ✓ **El lenguaje de consulta debe ser independiente de la sintaxis**, esto es, pueden existir varias sintaxis distintas para expresar una misma consulta en XQuery.

Aplicaciones

Una vez que hemos visto la definición del lenguaje y sus principales requerimientos, queda pensar, ¿para qué se utiliza?

Sus principales aplicaciones se resumen en tres:

- ✓ **Recuperar información a partir de conjuntos de datos XML**.
- ✓ **Transformar unas estructuras de datos XML** en otras estructuras que organizan la información de forma diferente.
- ✓ **Ofrecer una alternativa a XSLT** para realizar transformaciones de datos en XML a otro tipo de representaciones, como HTML o PDF.

¿Y cuáles son los motores XQuery de código abierto (*También llamado "open source", es la denominación que se le da al software que se desarrolla y distribuye libremente, es decir aquellos programas que podemos utilizar, modificar y redistribuir de forma gratuita*) más relevantes y sus características principales?

- ✓ **Qexo:** escrito en Java y con licencia GPL que se distribuye integrado dentro del paquete Kawa (*Es un IDE (Entorno de Desarrollo Integrado) que permite crear aplicaciones en lenguaje Java*).
- ✓ **Saxon:** escrito en Java y distribuido en dos paquetes:
  - ➔ Saxon-B es open-source bajo licencia GPL y contiene una implementación básica de XSLT 2.0 y XQuery.
  - ➔ Saxon-SA, contiene un procesador completo XSLT y XQuery pero tiene licencia propietaria, aunque está disponible una licencia de evaluación de 30 días.
- ✓ **Qizx/open:** es una implementación Java de todas las características del lenguaje excepto la importación y validación de XML-Schemas. Es uno de los motores con licencia GPL más completo que existe.

¿Qué otras herramientas relacionadas con XQuery existen?

- ✓ **Xquark Bridge:** es una herramienta que permite importar y exportar datos a bases de datos relacionales utilizando XML, ofreciendo, por tanto, la posibilidad de manejar estructuras XML y realizar la transformación a objetos de la base de datos, y viceversa. Además XQuark respeta las restricciones de integridad y transforma las relaciones implícitas en los documentos XML, en relaciones explícitas en la base de datos.

  También soporta la consulta y manipulación de los datos en formato XML utilizando el lenguaje XQuery. XQuark-Bridge soporta MySQL, Oracle, SQLServer y Sybase, tiene una licencia LGPL.

- ✓ **BumbleBee:** es un entorno de prueba automático, creado para evaluar motores de XQuery y validar consultas XQuery. BumbleBee permite entre otras cosas, comprobar si una versión más moderna de nuestro motor permite seguir ejecutando nuestras consultas.

  BumbleBee se distribuye con un conjunto de pruebas ya preparadas y además, ofrece un entorno sencillo para redactar y ejecutar nuestras propias pruebas. Soporta los motores XQuery de código abierto: Qexo, Qizx/open y Saxon y los motores XQuery propietarios: Cerisent, Ipedo, IPSI-XQ y X-Hive. Es software propietario aunque está disponible una versión de demostración completamente funcional durante 30 días.

¿Cuáles de las siguientes herramientas se relacionan directamente con XQuery?

☑	Xquark-Brigde	☑	Saxon
☐	Oracle	☐	MySQL

Modelo de datos

Aunque XQuery y SQL puedan considerarse similares, el modelo de datos sobre el que se sustenta XQuery es muy distinto del modelo de datos relacional sobre el que sustenta SQL, ya que XML incluye conceptos como jerarquía y orden de los datos que no están presentes en el modelo relacional.

Por ejemplo, a diferencia de SQL, en XQuery el orden en que se encuentren los datos es importante, ya que no es lo mismo buscar una etiqueta "<B>" dentro de una etiqueta "<A>" que todas las etiquetas "<B>" del documento (que pueden estar anidadas dentro de una etiqueta "<A>" o no).

La entrada y la salida de una consulta XQuery se define en términos de un modelo de datos. Dicho modelo de datos de la consulta proporciona una representación abstracta de uno o más documentos XML (o fragmentos de documentos).

Las principales características de este modelo de datos son:

- ✓ Se basa en la **definición de secuencia**, como una colección ordenada de cero o más ítems. Éstas pueden ser heterogéneas, es decir pueden contener varios tipos de nodos y valores atómicos. Sin embargo, una secuencia nunca puede ser un ítem de otra secuencia.
- ✓ **Orden del documento**: corresponde al orden en que los nodos aparecerían si la jerarquía de nodos fuese representada en formato XML, (si el primer carácter de un nodo ocurre antes que el primer carácter de otro nodo, lo precederá también en el orden del documento).
- ✓ Contempla un **valor especial llamado "error value"** que es el resultado de evaluar una expresión que contiene un error.

Alguna de las características del modelo de datos en el que se basa XQuery es:

- ☐ El orden de los nodos no importa
- ☐ Se basa en secuencias de ítems, donde las secuencias se anidan dentro de otras secuencias
- ☐ La secuencia es un conjunto de datos de cualquier tipo
- ◉ **Tiene un valor especial que aparece cuando se trata de evaluar una expresión errónea**

En el siguiente enlace puedes encontrar el estándar de XQuery aprobado por el W3C en diciembre de 2010.

http://www.w3.org/TR/xquery/

Expresiones

Una consulta en XQuery es una expresión que lee una secuencia de datos en XML, y devuelve como resultado otra secuencia de datos en XML.

El valor de una expresión es una secuencia heterogénea de nodos y valores atómicos.

La mayoría de las expresiones están compuestas por la combinación de expresiones más simples unidas mediante operadores y palabras reservadas.

Ya hemos visto que Xpath es un lenguaje declarativo para la localización de nodos y fragmentos de información en árboles XML.

Puesto que XQuery ha sido construido sobre la base de Xpath y realiza la selección de información y la iteración a través del conjunto de datos basándose en dicho lenguaje, toda expresión XPath también es una consulta Xquery válida.

Los comentarios en XQuery están limitados entre caras sonrientes, es decir: (: Esto es un comentario XQuery :).

En un documento XQuery los caracteres { } delimitan las expresiones que son evaluadas para crear un documento nuevo.

XQuery admite expresiones condicionales del tipo **if-then-else** con la misma semántica que tienen en los lenguajes de programación habituales.

Las consultas XQuery pueden estar formadas por hasta cinco tipos de cláusulas diferentes, siguen la norma FLWOR (que se pronuncia "flower"). Estas cláusulas son los bloques principales del XQuery, equivalen a las cláusulas **select, from, where, group by, having, order by** y **limit** de **SQL.**

En una sentencia FLWOR al menos ha de existir una cláusula FOR o una LET, el resto, si existen, han de respetar escrupulosamente el orden dado por el nombre, FLWOR.

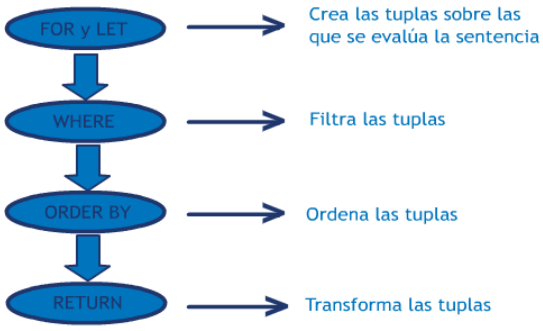
Con estas sentencias se consigue buena parte de la funcionalidad que diferencia a XQuery de XPath. Entre otras cosas permite construir el documento que será la salida de la sentencia.

Una consulta XQuery está formada por dos partes:

- ✓ **Prólogo**: Lugar donde se declaran los espacios de nombres, de funciones, variables, etc.
- ✓ **Expresión**: Consulta propiamente dicha.

Toda sentencia XPath es una sentencia XQuery:

- ◉ **Verdadero**
- ○ Falso



Cláusulas

Hemos visto el modo de crear sentencias FLWOR, vamos ahora a estudiar aisladamente cada una de las cláusulas que pueden formar estas sentencias.

- ✓ **FOR**: asocia una o más variables con cada nodo que encuentre en la colección de datos. Si en la consulta aparece más de una cláusula FOR (o más de una variable en una cláusula FOR), el resultado es el producto cartesiano de dichas variables.
- ✓ **LET**: vincula las variables al resultado de una expresión. Si esta cláusula aparece en una sentencia en la que ya hay al menos una cláusula FOR, los valores de la variable vinculada por la cláusula LET se añaden a cada una de las tuplas generadas por la cláusula FOR.
- ✓ **WHERE**: filtra tuplas producidas por las cláusulas FOR y LET, quedando solo aquellas que cumplen con la condición.

- ✓ **ORDER BY**: ordena las tuplas generadas por FOR y LET después de que han sido filtradas por la cláusula WHERE. Por defecto el orden es ascendiente, pero se puede usar el modificador **descending** para cambiar el sentido del orden.
- ✓ **RETURN:** construye el resultado de la expresión FLWOR para una tupla dada.

Ejercicio resuelto

Veamos unos ejemplos de la forma de uso de estas cláusulas. Para ello partiremos del fichero XML de datos:

```
<?xml version="1.0" encoding="ISO-8859-1" standalone="yes"?>

<!-- Fichero: libros.xml -->

<biblioteca>
<libros>

<libro publicacion="2003" edicion="2">
<titulo>Learning XML</titulo>

<autor>
<apellido>Ray</apellido>
<nombre>Erik T.</nombre>
</autor>

<editorial>O'Reilly</editorial>

<paginas>416</paginas>
</libro>

<libro publicacion="2003" edicion="2">
<titulo>XML Imprescindible</titulo>

<autor>
<apellido>Harold</apellido>
<nombre>Elliot Rusty</nombre>
</autor>

<autor>
<apellido>Means</apellido>
<nombre>W. Scott</nombre>
</autor>

<editorial>O'Reilly</editorial>

<paginas>832</paginas>
</libro>

<libro publicacion="2002">
<titulo>XML Schema</titulo>

<autor>
<apellido>van der Vlist</apellido>
<nombre>Eric</nombre>
</autor>

<editorial>O'Reilly</editorial>

<paginas>400</paginas>
</libro>

<libro publicacion="2002">
<titulo>XPath Essentials</titulo>

<autor>
<apellido>Watt</apellido>
<nombre>Adrew</nombre>
</autor>

<editorial>Wiley</editorial>
```

```
<paginas>516</paginas>
</libro>

<libro publicacion="2005">
<titulo> Beginning XSLT 2.0: Form Novice to Professional</titulo>

<autor>
<apellido>Tennison</apellido>
<nombre>Jeni</nombre>
</autor>

<editorial>Apress</editorial>

<paginas>797</paginas>
</libro>

<libro publicacion="2007">
<titulo> XQuery</titulo>

<autor>
<apellido>Walmsley</apellido>
<nombre>Priscilla</nombre>
</autor>

<editorial>O'Reilly</editorial>

<paginas>491</paginas>
</libro>
</libros>

<!-- Tabla prestamos -->

<prestamos>
<entrada>
<titulo>XML Imprescindible</titulo>

<prestamo>
<inicio>2011-05-02</inicio>

<lector>
<nombre>Pepito</nombre>
<apellidos>Grillo</apellidos>
<direccion>Rue Percebe, 13</direccion>
<telefono>972987654</telefono>
</lector>
</prestamo>
</entrada>

<entrada>
<titulo>XML Imprescindible</titulo>

<prestamo>
<inicio>2011-02-12</inicio>
<devolucion>2011-02-16</devolucion>

<lector>
<nombre>Jose</nombre>
<apellidos>Gutiérrez González</apellidos>
<direccion>Rue Percebe, 13</direccion>
<telefono>919485432</telefono>
</lector>
</prestamo>
</entrada>

<entrada>
<titulo>XPath Essentials</titulo>

<prestamo>
<inicio>2011-02-23</inicio>
<devolucion>2011-03-10</devolucion>

<lector>
<nombre>Pepito</nombre>
<apellidos>Grillo</apellidos>
<direccion>Rue Percebe, 13</direccion>
<telefono>972987654</telefono>
</lector>
```

```
</prestamo>
</entrada>
</prestamos>

</biblioteca>
```

1. Ejemplo de uso de la cláusula FOR. Obtener todos los títulos de los libros del fichero libros.xml.
2. Ejemplo de uso de la cláusula LET. Obtener todos los títulos de los libros del fichero libros.xml.
3. Ejemplo de uso de la cláusula FOR y LET juntas. Obtener todos los títulos de los libros del fichero libros.xml junto con los autores de cada libro.
4. Ejemplo de uso de las cláusulas WHERE y ORDER BY en una consulta con dos ficheros. Obtiene los títulos de los libros prestados con sus autores y la fecha de inicio y devolución del préstamo, ordenados por la fecha de inicio del préstamo.

Resultado:

1. Ejemplo de uso de la cláusula FOR. Obtener todos los títulos de los libros del fichero libros.xml.

```
for $d in doc("libros.xml")/biblioteca/libros/libro/titulo
return
<titulos>{ $d }</titulos>
```

2. Ejemplo de uso de la cláusula LET. Obtener todos los títulos de los libros del fichero libros.xml.

```
let $d := doc("libros.xml")/biblioteca/libros/libro/titulo
return
<titulos>{ $d }</titulos>
```

3. Ejemplo de uso de la cláusula FOR y LET juntas. Obtener todos los títulos de los libros del fichero libros.xml junto con los autores de cada libro.

```
for $b in doc("libros.xml")//libro
let $c := $b/autor
return
<libro>{ $b/titulo, <autores>{ $c }</autores>}</libro>
```

Otra solución posible a este ejercicio, en este caso utilizando únicamente la cláusula FOR

```
for $b in doc("libros.xml")//libro
return
<libro>{ $b/titulo, <autores>{$b/autor}</autores>}</libro>
```

4. Ejemplo de uso de las cláusulas WHERE y ORDER BY en una consulta con dos ficheros. Obtiene los títulos de los libros prestados con sus autores y la fecha de inicio y devolución del préstamo, ordenados por la fecha de inicio del préstamo.

```
for $t in doc("libros.xml")//libro,
$e in doc("prestamos.xml")//entrada
where $t/titulo = $e/titulo
order by $e/prestamo/inicio
return <prestamo>{ $t/titulo, $t/autor/*, $e/prestamo/inicio, $e/prestamo/devolucion }</prestamo>
```

En el siguiente enlace puedes ver una presentación que muestra el modo en el que se ejecuta una sentencia FLWOR.

http://www.youtube.com/watch?v=0fR4Twl9Ias&feature=player_embedded#!

Funciones

Ahora que conocemos las cláusulas que sustentan el lenguaje **XQuery**, vamos a ocuparnos de las **funciones que soporta.**

Estas son funciones matemáticas, de cadenas, para el tratamiento de expresiones regulares, comparaciones de fechas y horas, manipulación de nodos XML, manipulación de secuencias, comprobación y conversión de tipos y lógica booleana. Además permite definir funciones propias y funciones

dependientes del entorno de ejecución del motor XQuery. Las funciones más importantes se muestran a continuación:

- ✓ **Funciones numéricas**
  - ➔ **floor(),** que devuelve el valor numérico inferior más próximo al dado**.**
  - ➔ **ceiling(),** que devuelve el valor numérico superior más próximo al dado**.**
  - ➔ **round(),** que redondea el valor dado al más próximo.
  - ➔ **count(),** determina el número de ítems en una colección.
  - ➔ **min()** o **max(),** devuelven respectivamente el mínimo y el máximo de los valores de los nodos dados.
  - ➔ **avg(),** calcula el valor medio de los valores dados.
  - ➔ **sum(),** calcula la suma total de una cantidad de ítems dados.
- ✓ **Funciones de cadenas de texto**
  - ➔ **concat(),** devuelve una cadena construida por la unión de dos cadenas dadas.
  - ➔ **string-length(),** devuelve la cantidad de caracteres que forman una cadena.
  - ➔ **startswith(), ends-with(),** determinan si una cadena dada comienza o termina, respectivamente, con otra cadena dada.
  - ➔ **upper-case(), lower-case(),** devuelve la cadena dada en mayúsculas o minúsculas respectivamente.
- ✓ **Funciones de uso general**
  - ➔ **empty(),** devuelve "**true**" cuando la secuencia dada no contiene ningún elemento.
  - ➔ **exits(),** devuelve "**true**" cuando una secuencia contiene, al menos, un elemento.
  - ➔ **distinct-values(),** extrae los valores de una secuencia de nodos y crea una nueva secuencia con valores únicos, eliminando los nodos duplicados.
- ✓ **Cuantificadores existenciales:**
  - ➔ **some, every**, permiten definir consultas que devuelven algún, o todos los elementos, que verifiquen la condición dada.

Ejercicio resuelto

Dados los ficheros XML del ejemplo anterior, hacer una consulta que devuelva los títulos de los libros almacenados en el fichero libros.xml y su primer autor. En caso de que haya más de un autor para un libro se añade un segundo autor "cia".

```
for $b in doc("libros.xml") //libro
return
<libro>
{ $b/titulo }
{
for $a at $i in $b/autor
where $i <= 1
return <autor>{string($a/nombre), ", ",
string($a/apellido)}</autor>
}
{
if (count($b/autor) > 1)
then <autor>cia.</autor>
else ()
}
</libro>
```

Además de estas funciones que están definidas en el lenguaje XQuery permite al desarrollador construir sus propias funciones:

```
declare nombre_funcion($param1 as tipo_dato1, $param2 as tipo_dato2,… $paramN as tipo_datoN)
as tipo_dato_devuelto
{
...CÓDIGO DE LA FUNCIÓN...
}
```

Ejercicio resuelto

```
declare function minPrice($p as xs:decimal?,$d as xs:decimal?)
as xs:decimal?
{
let $disc := ($p * $d) div 100
return ($p - $disc)
}
```

Un ejemplo de cómo invocar a la función desde la consulta es:

```
<minPrice>{minPrice($libros/precio,$libros/descuento)}</minPrice>
```

Operadores

Veamos ahora algunos de los operadores más importantes agrupados según su funcionalidad:

- ✓ **Comparación de valores:** Comparan dos valores escalares y produce un error si alguno de los operandos es una secuencia de longitud mayor de 1. Estos operadores son:
  - ➔ **eq**, igual.
  - ➔ **ne**, no igual.
  - ➔ **lt**, menor que.
  - ➔ **le**, menor o igual que.
  - ➔ **gt**, mayor que.
  - ➔ **ge**, mayor o igual que.
- ✓ **Comparación generales:** Permiten comparar operandos que sean secuencias.
  - ➔ **=**, igual.
  - ➔ **!=**, distinto.
  - ➔ **>**, mayor que.
  - ➔ **>=**, mayor o igual que.
  - ➔ **<,** menor que.
  - ➔ **<=**, menor o igual que.
- ✓ **Comparación de nodos:** Comparan la identidad de dos nodos.
  - ➔ **is**, devuelve true si las dos variables que actúan de operandos están ligadas al mismo nodo.
  - ➔ **is not**, devuelve true si las dos variables no están ligadas al mismo nodo.
- ✓ **Comparación de órdenes de los nodos: <<**, compara la posición de dos nodos. Devuelve "**true**" si el nodo ligado al primer operando ocurre primero en el orden del documento que el nodo ligado al segundo.
- ✓ **Lógicos: and** y **or** Se emplean para combinar condiciones lógicas dentro de un predicado.
- ✓ **Secuencias de nodos:** Devuelven secuencias de nodos en el orden del documento y eliminan duplicados de las secuencias resultado.
  - ➔ **Union**, devuelve una secuencia que contiene todos los nodos que aparecen en alguno de los dos operandos que recibe.
  - ➔ **Intersect**, devuelve una secuencia que contiene todos los nodos que aparecen en los dos operandos que recibe.
  - ➔ **Except**, devuelve una secuencia que contiene todos los nodos que aparecen en el primer operando que recibe y que no aparecen en el segundo.
- ✓ **Aritméticos**: **+, -, *, div y mod**, devuelven respectivamente la suma, diferencia, producto, cociente y resto de operar dos números dados.

Cuáles de los siguientes operadores pertenecen a SQL y a XQuery:

☑ is	☐ lt
☑ union	☑ >=

TEMA 7

Contenido

Sistema de Gestión Empresarial.

Caso práctico
*Tras las vacaciones de **María** se implantaron las tecnologías XSLT y **XQuery en la empresa**, logrando mejorar los sistemas de almacenamiento de información.*
*Ahora es **Félix** quien disfruta de un merecido descanso en un pueblecito costero.*
Mientras tanto Juan se está informando sobre la utilidad de los ERP, ha leído que son sistemas informáticos que pueden mejorar notablemente la actividad comercial de una empresa. Además, se pueden usar desde los dispositivos móviles, con lo que no es imprescindible estar en un despacho con un ordenador conectado a una red local determinada para poder consultar la información alojada en ellos.
El mayor problema que ve Juan, es la posible necesidad de formar a los trabajadores de la empresa para que puedan utilizar esta herramienta, así como la integración de la misma con el resto de programas que se usan, los que casi pueden reducirse a una suite ofimática.
Según se va informando del tema, cada vez encuentra menos inconvenientes ya que no parece que la utilización de estos recursos requieran de una gran formación por parte de los usuarios y, por otro lado, la mayoría de los ERP del mercado, si no todos, permiten la integración las suites informáticas. Esto significa que los usuarios podrán seguir usando los mismos programas que hasta ahora.
Juan decide no contarle nada a María hasta que Félix regrese y seguir informándose sobre este tema, que le parece muy interesante desde el punto de vista de la gestión de la información en la empresa.

Identificación de flujos de información.

Caso práctico
Félix ya ha vuelto a su puesto de trabajo y Juan ha solicitado una reunión para plantear su nueva idea.

Está preparando esta reunión del siguiente modo:
- ✓ *Comenzará por exponer cuales son los distintos flujos de información que hay en la empresa, haciendo especial hincapié en aquellos que se usan en los procesos más importantes.*
- ✓ *Después pasará a analizar los distintos subsistemas de información que existen y su relación con los flujos de información. Incidiendo en que si se optimizan los flujos de información entre los subsistemas, se logrará mejorar la gestión y el control de la información.*

Para estudiar este tema lo primero que hemos de hacer es pararnos a pensar cómo fluye la información dentro de la empresa. Si lo meditamos con calma veremos que lo hace de varias formas posibles:

- ✓ Entre los empleados de la empresa.
- ✓ Entre los empleados y la empresa.
- ✓ Entre la empresa con sus clientes y proveedores.

Estos flujos de información pueden ser de dos tipos:

- ✓ Informales y no estructurados.
- ✓ Formales y estructurados, que se centran en información acerca de procesos críticos de la empresa.

Para facilitar este flujo de información es conveniente tener instalado un Sistema de Información. Pero, ¿qué es un Sistema de Información?

Son subsistemas dentro de la empresa que facilitan la transferencia de información. Los elementos de los que constan se clasifican en: Físicos, Humanos y Normas y protocolos.

Denominamos Sistema de Información a un conjunto organizado de elementos relacionados orientados al tratamiento y administración de información.

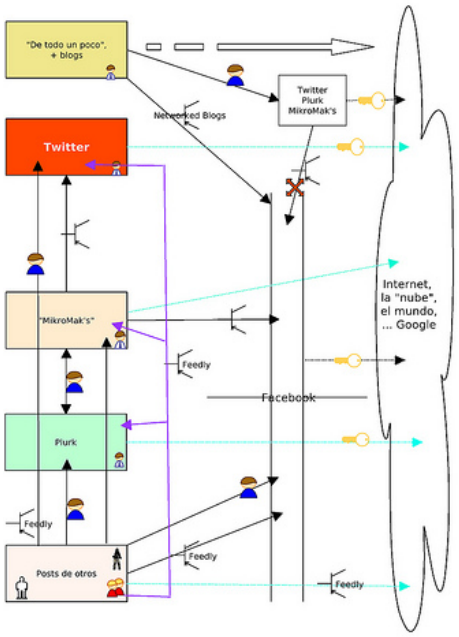
Un SI facilita el conocimiento propio de la empresa para mejorar la planificación, la gestión y el control.

¿Qué es lo que la empresa obtiene del conocimiento y la gestión de los flujos de información?

Principalmente ventajas competitivas, que mejoran su eficiencia, la calidad del producto, el servicio ofrecido a los clientes y facilitan la captación de clientes.

La automatización de los flujos de información asociándolos a los flujos físicos de producción de materiales, cambió sustancialmente cuando aparecieron los sistemas de información ERP y CRM, en los que se integran las aplicaciones que soportan los diferentes procesos de la empresa.

La rápida expansión de Internet ha incrementado las interacciones entre empresas y entre éstas y sus clientes, provocando la modificación de los modelos de negocio tradicionales. La proliferación del uso de dispositivos portátiles ha facilitado la existencia de empleados no ubicados en una oficina fija, lo que ha obligado a buscar soluciones para integrar a los empleados móviles en los flujos de información de la empresa.



ERP y CRM.

Caso práctico

*Durante la reunión **Félix** y **María** escuchan atentamente las explicaciones de Juan, pero no logran ver el beneficio que les aportará el uso de un ERP.*

***Juan** pasa a explicarles que **un ERP está formado por diversas partes integradas cuya funcionalidad es totalmente independiente de los demás**. Les expone que en realidad la información estaría guardada en una única base de datos, a la que accedería el ERP a medida que los usuarios lo demandasen. La gran ventaja que Juan ve en todo esto, es que todos los usuarios comparten la misma información garantizando que los datos con los que trabajan siempre estén actualizados.*

Los ERP, son sistemas de gestión de información que se caracterizan por ser una aplicación en la que hay varias partes integradas. Estas partes son de diferente uso, por ejemplo: producción, ventas, compras, pedidos, nóminas, etc.

Un sistema de CRM es un sistema de gestión de información se centra en mantener información de contacto orientada a ventas.

Los objetivos principales de los sistemas ERP son:

- ✓ **Optimizar** los procesos empresariales.
- ✓ **Acceder** a la información confiable y precisa.
- ✓ **Permitir** compartir información entre los componentes de la organización.
- ✓ **Eliminar** los datos y operaciones innecesarias.

Las características que distinguen a un ERP de cualquier otro software empresarial, es que deben de ser sistemas integrales, modulares y adaptables. Además, un ERP se caracteriza por:

- ✓ Ser un programa con **acceso a una base de datos**.
- ✓ Sus **componentes interactúan** entre sí.
- ✓ Los **datos** deben ser **consistentes**, **completos**.

En ocasiones son sistemas complejos y difíciles de implantar, debido a que necesitan un desarrollo personalizado para cada empresa a partir del paquete inicial. Estas adaptaciones suelen encargarse a las consultorías.

La consultoría en materia de ERP puede ser de dos tipos:

- ✓ **Consultoría de negocios**. Estudia los procesos de negocio de la compañía, y evalúa su correspondencia con los procesos del sistema ERP para poder personalizarlo, y de este modo ajustarlo a las necesidades de la organización.
- ✓ **Consultoría técnica**. Conlleva el estudio de los recursos tecnológicos existentes, en ocasiones implica la programación del sistema, obtener determinados informes.

En la actualidad, y debido a la amplia implantación de las intranets en las empresas, la mayoría de los sistemas ERP tienen interfaz web, lo que aporta la ventaja de permitir el acceso al ERP a través del navegador web.

Marca aquellas afirmaciones que sean ciertas para un ERP:

- ☐ Es un sistema formado por varias aplicaciones que interactúan entre sí.
- ☐ La empresa ha de escoger entre los diversos ERP que existen, el que se adapte a su modo de trabajo.
- ☑ **Para implementar un ERP es necesario tener asociada una base de datos centralizada.**
- ☑ **Un ERP está formado por diversos módulos.**

Ventajas y desventajas.

Contar con **un sistema ERP personalizado, permite a la empresa tener integradas diferentes utilidades que le facilitan la gestión de la información**. Permitiendo entre otras cosas, administrar interdependencias de los recibos de materiales, de los productos estructurados en el mundo real, de los cambios de la ingeniería y de la revisión y la mejora, y la necesidad de elaborar materiales sustitutos, etc. La seguridad de los ordenadores está incluida dentro del ERP, para proteger a la organización contra el espionaje industrial y la malversación.

Los ERP que incluyen CRM aportan beneficios relacionados con la administración de los clientes de la empresa, y pueden incluir un control de calidad en los productos finales, que garantiza que no hay problemas no resueltos. La principal ventaja de la implementación CRM en una empresa, es el aumento de la información que ésta posee sobre sus actuales y potenciales clientes, lo que le permite direccionar la oferta hacia sus deseos y necesidades, aumentando así el grado de satisfacción y optimizando su ciclo de vida. Otras ventajas de la implementación del CRM son el aumento de las ventas y la reducción del ciclo de venta.

Muchos de los problemas que tienen las compañías con el ERP, son debidos a la inversión inadecuada para la formación del personal, y una falta de políticas corporativas que afectan al modo en que se obtienen y actualizan los datos del ERP.

Limitaciones y obstáculos del ERP:

- ✓ El ERP ha de ser utilizado y realizado por personal capacitado.
- ✓ La instalación del ERP es muy costosa.
- ✓ Los ERP son vistos como sistemas muy rígidos, y difíciles de adaptar al modo de trabajo de las empresas.
- ✓ Son sistemas que sufren problemas de "cuello de botella", es decir, todos los usuarios del sistema se pueden ver afectados por la ineficiencia en uno de los departamentos participantes.
- ✓ Altos costes de las modificaciones del ERP una vez implantado.

¿Te has parado a pensar alguna vez en los efectos negativos que tiene la falta de normalización en los procedimientos que se realizan en las pequeñas empresas? Y ¿la falta de interés que suelen tener los empleados de las mismas en establecer un protocolo de comportamiento que los formalice?

Marca en las siguientes afirmaciones cuál es una ventaja para un ERP:

- ☐ Solo personal capacitado debe utilizarlo.
- ☐ En la actualidad la instalación es asequible a cualquier empresa.
- ☐ Son sistemas sencillos de manejar en el que el error de un usuario no afecta al resto.
- ☑ **Es un sistema en el que están integradas varias de las utilidades que usa la empresa.**

Descarga e instalación.

Para realizar la **instalación de un paquete de gestión empresarial**, primero tenemos que definir cuáles son las necesidades que debe cubrir el software y buscar aquel que mejor se ajuste a nuestras necesidades.

Después es importante leer los requisitos mínimos que especifica el fabricante antes de proceder a la instalación, para no tener problemas de funcionamiento con posterioridad.

Independientemente del paquete que vayamos a instalar, debemos tener claro que nuestro sistema se basa en una base de datos en la que se irán almacenando los datos, y desde la que se irán generando los informes que requerimos al sistema, por lo que previamente la habremos creado en nuestro sistema.

Hoy en día, lo más habitual es incorporar el ERP dentro de la intranet de la empresa. En este caso, necesitamos tener activo un servidor web con soporte para bases de datos, y el lenguaje de script de servidor en el que se haya codificado la aplicación que vayamos a instalar. La instalación de este último tipo de sistemas se realiza también a través de un navegador web siguiendo los pasos que indica el sistema.

Para realizar la instalación de un ERP en nuestro sistema necesitamos tener una base de datos creada para esta función así como un servidor web con soporte para bases de datos.

- ☐ Verdadero
- ☑ **Falso**

Desde el siguiente enlace puedes descargar el paquete de instalación de SugarCMR, un CRM gratuito y de código abierto en un servidor web local.

http://www.sugarforge.org/frs/?group_id=6

Desde el siguiente enlace puedes descargar el paquete de instalación del servidor web local Wamp Server.

http://www.wampserver.com/en/#download-wrapper

Adaptación y configuración. Integración de módulos.

Tras el proceso de instalación del paquete básico de software ERP viene la personalización, para adecuarlo al entorno de la empresa que va a utilizarlo. Este proceso incluye aspectos como incorporar el logo de la compañía, o dar de alta a los usuarios con los permisos adecuados, además de configurar el sistema de avisos que proporciona la aplicación, o la compatibilidad con herramientas de correo.

Los sistemas ERP que existen hoy en el mercado permiten la incorporación de distintos tipos de módulos predefinidos que facilitan la personalización del paquete.

La integración de estos módulos, que complementan la aplicación base, puede realizarse tanto en el momento inicial de la instalación de la aplicación, como en un proceso posterior de ampliación de la misma.

Los recursos que proporcionan estos módulos son muy variados, desde la creación de informes avanzados a servicios de comunicación para plataformas móviles (tablet-pc o smartphones), pasando por la interconexión con el paquete ofimático utilizado en la empresa. Hoy en día, la mayoría de los sistemas ERP cuentan con un módulo CRM integrado, o con la posibilidad de incorporarlo como un módulo añadido.

Los módulos que se incorporan pueden estar prediseñados por el fabricante de la aplicación base, ser módulos programados por terceros para ese software, o incluso ser un programa solicitado a medida por la empresa en la que se instala la aplicación.

Para realizar la instalación de módulos hay que entrar en el paquete con un usuario que tenga permisos de administración, además hay que usar el cargador de módulos que suelen tener estos paquetes en la sección de administración.

Escoge las afirmaciones correctas:

- [ ] Al instalar un ERP siempre se instalan todos los módulos que hay disponibles para él
- [ ] En el proceso de instalación de un ERP tenemos que decidir todos los módulos que queremos instalar ya que luego no podremos modificarlo.
- [x] **Una empresa puede instalar en su ERP tanto módulos preprogramados como módulos hechos de encargo para ajustarse a sus necesidades.**
- [x] **La integración de módulos puede hacerse en cualquier momento.**

Desde el siguiente enlace puedes ver y descargar distintos módulos para SugarCMR.
http://www.sugarforge.org/

Planificación de la seguridad.

Caso práctico

*Una vez que les convence de la utilidad de los ERP, **María y Félix le preguntan por la seguridad de los datos**, ya que en su empresa es imprescindible garantizar la privacidad de la información que manejan, y el hecho de que todos compartan la misma información, induce a creer que todos los usuarios del sistema van a poder acceder a la información.*

***Juan** les cuenta que ese era uno de los puntos débiles que podría tener su propuesta, pero que después de haberlo estudiado y meditado no hay ningún problema.*

*Los ERP tienen integradas una serie de medidas de seguridad, que se orientan a que haya **distintos perfiles de usuario** con diferentes niveles de acceso a la información, lo que hace que cada usuario sólo acceda a la información que necesita para realizar su trabajo. Además el modo de este acceso es también el imprescindible para cada usuario. Es decir, un usuario que sólo necesita consultar cierta información, podrá consultarla, pero no modificarla.*

La conexión entre distintos equipos se realiza utilizando protocolos de seguros.

Se garantizan las operaciones que se realizan sobre los datos.

Las medidas de seguridad que se implementan en estos sistemas se basan, principalmente, en los siguientes aspectos:

- ✓ **Niveles de acceso configurables para los usuarios según su rol.**

  En función de las tareas que deba realizar, el usuario debe contar con una serie de políticas que le permitan acceder a determinados datos, quedando algunos de ellos reservados para usuarios con un nivel de toma de decisión más elevado.

- ✓ **Auditoría de cada transacción.**

  Se controla cada envío de datos, lo que garantiza las operaciones realizadas.

- ✓ **Soporte para la conexión segura mediante https**.

  Para garantizar la seguridad de la comunicación entre los equipos cliente y el servidor en el que está instalada la aplicación, se realiza con un protocolo de comunicación seguro que no permita espiar el canal de comunicación. Se emplea este tipo de comunicación segura en el proceso de autentificación de usuarios.

La seguridad de los ERP y CRM se basa en los siguientes aspectos:

- ☑ **Controlar cada una de las transacciones de datos.**
- ☑ **Utilizar el protocolo http seguro.**
- ☑ **Establecer distintos perfiles de usuario.**
- ☐ El administrador del sistema ha de confirmar las operaciones que realizan los usuarios antes de que estas sean válidas.

Usuarios y roles.

Parte de la configuración de la seguridad del sistema es consecuencia de una buena asignación del rol de **cada uno de los usuarios del sistema, ya que esto garantiza que cada uno de los usuarios sólo tiene acceso a la información que necesita para realizar su trabajo**. Este trabajo hay que llevarlo a cabo con un usuario administrador.

Los paquetes básicos de los ERP y CRM suelen tener varios tipos de usuarios posibles, por ejemplo, el entorno SugarCRM:

- ✓ **Administrador**, es un tipo de usuario que tiene todos los privilegios, tanto en lo referente al acceso a la información como a las tareas de gestión del sistema, (creación de usuarios, integración de módulos, modificación del aspecto del sistema, etc.).

- ✓ **Usuario normal**, es un usuario que no tiene privilegios de administración, pero tiene todos los privilegios respecto a la información almacenada.
- ✓ **Usuario de grupo**, se crean para recibir el correo entrante para distribución.
- ✓ **Usuario de portal**, permite al usuario acceder a los portales creados en el entorno pero no a la aplicación.

Hay aplicaciones en las que los usuarios no pueden eliminarse directamente, hay que hacerlo desde la base de datos, aunque siempre permiten desactivarlos.

Además de los perfiles de usuarios es interesante **conocer los diferentes roles** que hay definidos para los usuarios de una aplicación.

Un rol define ciertos privilegios a la hora de realizar tareas específicas.

Las características de los roles son:

- ✓ Podemos considerar que un rol es un grupo particular de privilegios.
- ✓ Un rol solo tiene validez cuando está asignado a algún usuario.
- ✓ Un usuario puede tener asignados varios roles, en ese caso prevalece el rol más restrictivo.
- ✓ Los cambios realizados en los roles no son efectivos hasta que no se inicia una nueva sesión.
- ✓ Cuando un rol niega el acceso a un módulo se pierde la posibilidad de ver cualquier subpanel del mismo.

Combinando los distintos tipos de usuarios con los diferentes roles podemos crear usuarios con diferentes niveles de seguridad.

○ Falso ◉ **Verdadero**

Elaboración de informes.

Caso práctico

***María**, se muestra un poco reticente a llevar a cabo este nuevo proyecto, ya que opina que cada vez que hay que realizar un informe de cualquier tipo será necesario contar con una persona del departamento informático, para que extraiga de la base de datos la información necesaria.*

Juan le replica que eso no es cierto, que el propio ERP tiene un módulo que le permite realizar cualquier tipo de informes directamente, y sin necesidad de tener que recopilar la información de diversas fuentes.

Los sistemas ERP cuentan con herramientas que facilitan la elaboración de informes que ayudan en la toma de decisiones estratégicas para mejorar:

A. La planificación.
B. La gestión.
C. El control.

Estas herramientas permiten, a cualquier usuario sin conocimientos técnicos, hacer informes dinámicos, flexibles e interactivos, para que el usuario no esté limitado por listados predefinidos.

También realizan una integración entre todos los sistemas/departamentos de la compañía, garantizando la calidad y la integración de los datos dentro de la empresa.

La versión de código abierto de SugarCRM no incluye el módulo para generar informes. Para poder usar dicho módulo habría que usar una versión comercial. Pero, se pueden usar otras herramientas en su lugar. Por ejemplo, JasperReports or Crystal Reports. Una lista de otras opciones se puede obtener de la página de descarga de módulos para SugarCRM, cuyo enlace te facilitamos a continuación.

Todos los CRM, incluido SugarCRM traen incorporado, en todas sus versiones, un módulo que permite generar informes.

☐ Verdadero ☒ **Falso**

Desde este enlace puedes descargar módulos de SugarCMR que permiten realizar informes.
http://www.sugarforge.org/softwaremap/trove_list.php?form_cat=361

Ahora veamos cómo instalar JasperReports para SugarCRM y usarlo para crear informes:

Descarga del módulo de JasperReports

JasperReports se distribuye en dos paquetes diferentes. Por un lado jasperreportssugarcrmsrc. zip que contiene todo el código fuente y una copia del servidor Apache Tomcat que se configurará una vez que la aplicación se haya instalado. Por otro lado jasperreportssugarcrm.zip, que contiene una copia preconfigurada de Tomcat, sin el código fuente. Ambas opciones nos llevarán al mismo lugar y puede escogerse cualquiera de ellas, pero en este manual tomaremos la segunda, con el añadido de que realizaremos la instalación en un Tomcat previamente instalado en la máquina (sin utilizar el Tomcat preconfigurado que viene con el módulo). Ambos paquetes puedes descargarse de la siguiente dirección:

http://www.sugarforge.org/frs/?group_id=193

Instalación de JasperReports en Apache Tomcat 5.5

- ✓ **PASO 1**: parar Tomcat.
- ✓ **PASO 2**: añadir lassiguientes líneas a /conf/server.xml modificando los datos referentes a la configuración de la máquina:

. . .

```
<GlobalNamingResources>
   <!-- Test entry for demonstration purposes -->
   <Environment name="simpleValue" type="java.lang.Integer" value="30"/>
   <!-- Editable user database that can also be used by
      UserDatabaseRealm to authenticate users -->
   <Resource name="UserDatabase" auth="Container"
         type="org.apache.catalina.UserDatabase"
      description="User database that can be updated and saved"
         factory="org.apache.catalina.users.MemoryUserDatabaseFactory"
         pathname="conf/tomcat-users.xml" />
   <Resource name="jdbc/jasperreports-sugarcrm" auth="Container"
type="javax.sql.DataSource"
         maxActive="100" maxIdle="30" maxWait="10000"
         username="sugarcrm" password="sugarcrm"
driverClassName="com.mysql.jdbc.Driver"
         url="jdbc:mysql://localhost:3306/sugarcrm?autoReconnect=true"/>
</GlobalNamingResources>
...
```

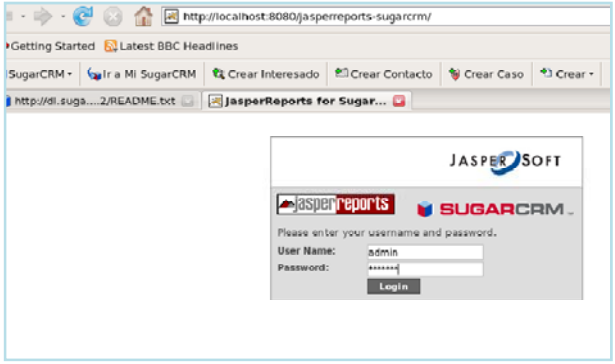
- ✓ **PASO 3**: añadir un contexto para jasperreportssugarcrm en la definición de host adecuada (probablemente localhost):

```
...
<Host name="localhost" appBase="webapps"
      unpackWARs="true" autoDeploy="true"
      xmlValidation="false" xmlNamespaceAware="false">
   <!-- JasperReports for SugarCRM Context -->
   <Context path="/jasperreports-sugarcrm" docBase="jasperreports-sugarcrm" debug="99"
         reloadable="true" crossContext="true" useNaming="true">
      <Logger className="org.apache.catalina.logger.FileLogger"
         prefix="localhost_jasperreports_sugarcrm_log." suffix=".txt"
         timestamp="true"/>
      <Realm className="net.sf.jasperreports.sugarcrm.realm.SugarDataSourceRealm"
         dataSourceName="jdbc/jasperreports-sugarcrm"
         digest="md5" userNameCol="user_name" userRoleTable="roles"
         userTable="users" debug="99" />
      <ResourceLink global="jdbc/jasperreports-sugarcrm"
         name="jdbc/jasperreports-sugarcrm"
         type="javax.sql.DataSource"/>
   </Context>
...
```

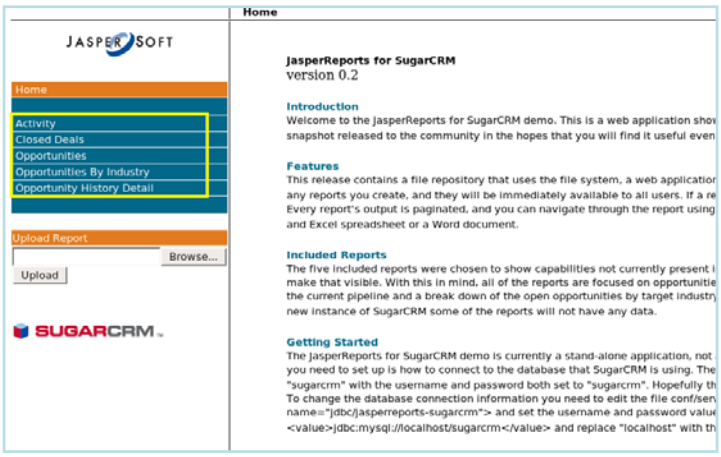
- ✓ **PASO 4**: Copiar el directorio jasperreportssugarcrm y todo su contenido (del archivo .zip descargado) al directorio raiz del Tomcat del servidor (en este ejemplo /usr/share/tomcat5.5/webapps/)
- ✓ **PASO 5**: Del directorio catalina/common/lib/de la distribución de jasperreportssugarcrm habrá que copiar los siguientes jars al directorio WEBINF/lib/ de la carpeta jasperreportsugarcrm en nuestro servidor:
  - ➔ ● commonsdbcp.jar
  - ➔ ● commonspool1.1.jar
- ✓ **PASO 6**: Añadir sugarrealmfortomcat5_5.jar a la carpeta server/lib en nuestro servidor
- ✓ **PASO 7**: Arrancar Tomcat.

Acceso a la interfaz web de JasperReportssugarcrm

Para acceder a la interfaz web de JasperReports hay que introducir la siguiente dirección en la barra del navegador: http://ipdelservidor:8080/jasperreportssugarcrm con lo que debería acceder a la página de acceso que se muestra a la derecha:

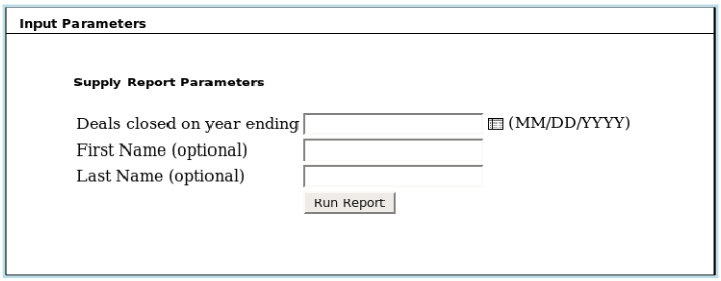


Tanto el nombre de usuario como la contraseña deben pertenecer ya a SugarCRM, pues es el acceso a la información de dicha aplicación lo que hay que autenticar. Una vez loggeado satisfactoriamente, se accede a la página principal de JasperReports, en la cual se muestran algunos criterios para realizar informes en función a la información almacenada en SugarCRM:

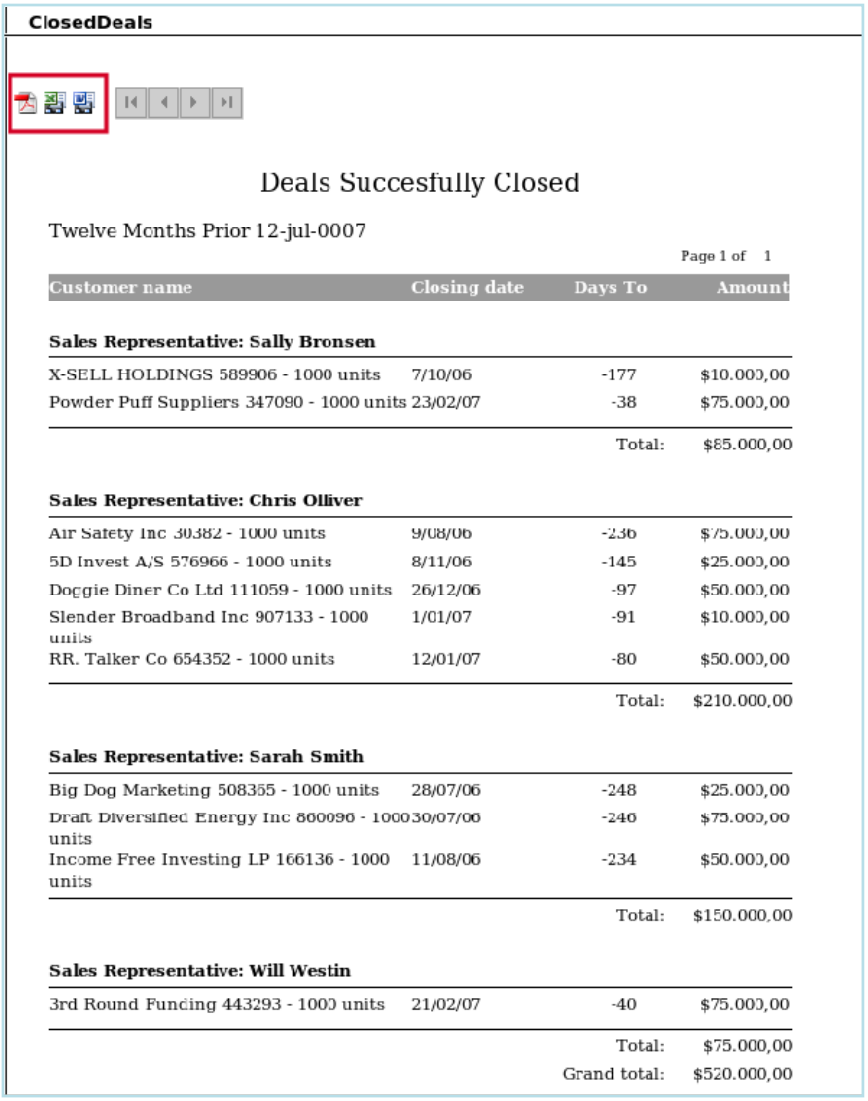


Ejemplo: Informes

Veamos, como ejemplo, el caso de *Closed Deals (Tratos cerrados)*:



Una vez introducida la fecha hasta la cual se quiere la información, y los datos opcionales, se obtendrá el informe correspondiente:



Además del informe correspondiente, también puede apeciarse que dicho informe puede exportarse mediante un simple click tanto a PDF como a documento de Word o de Excel (en el recuadro rojo).

Basta hacer clic sobre el icono correspondiente para que el informe sea exportado de forma sencilla a dicho formato.

Pese a que esta versión inicial consta solo de 5 tipos de informe diferentes, pronto habrá más disponibles, y si no, siempre queda la opción de desarrollar nuestros propios informes en base a criterios de búsqueda y comparación personalizados. Para ello podemos valernos de la herramienta JasperAssistant, que facilita en gran medida dicha tarea.

Integración con aplicaciones ofimáticas.

Caso práctico

__María__, viendo que __Juan__ está empeñado en llevar a cabo el nuevo proyecto, le pide que les cuente el resto de las características de los ERP.
Entonces Juan les habla de la posibilidad de __integrar las suites ofimáticas__ con las que trabajan, haciendo de este modo, que éstas formen parte del propio entorno del ERP y permitan trabajar directamente con los datos del ERP.

Los sistemas ERP y CRM suelen permitir la interacción con suites ofimáticas para facilitar el procesamiento de datos y la realización de informes y escritos. De esta manera, se consigue potenciar el uso de la herramienta ERP, al no tener que salir de ella para continuar realizando las tareas correspondientes. Y además el flujo de información entre la aplicación ERP y la suite es automático, lo que facilita la escritura de cartas y otro tipo de documentos. Otra ventaja a tener en cuenta es que el almacenamiento de los documentos generados se produce dentro de la misma aplicación ERP, lo que facilita compartirlos.

La integración con las aplicaciones ofimáticas puede ser a través de:

- ✓ La instalación de un módulo en la aplicación ERP.
- ✓ La instalación de un complemento en la suite ofimática, lo que suele incorporar una nueva barra de herramientas.

En Microsoft Office basta con bajarse el paquete de integración correspondiente al editor Word, o al gestor de correo Outlook. Una vez que se descomprime el paquete correspondiente ejecutamos el instalador que encontramos dentro de la carpeta descomprimida.

Después abrimos el programa correspondiente, Word u Outlook, vamos a la pestaña del CRM que se habrá creado en la barra del menú y configuramos el programa dándole la URL del CRM y el usuario y la contraseña con el que se va a conectar al CRM.

En el caso del editor Word, antes de realizar la configuración tenemos que registrar la instalación.

En el siguiente enlace puedes descargar el módulo de SugarCRM que permite su integración con Microsoft Office.

http://www.sugarforge.org/frs/?group_id=939

En el siguiente enlace puedes descargar la extensión de OpenOffice, que permite la integración de SugarCRM con el editor Writer y la hoja de cálculo Calc de OpenOffice.

http://www.sugarforge.org/frs/?group_id=986

En el siguiente vídeo se muestra la instalación la extensión de OpenOffice que permite su integración con SugarCRM.

http://openatrix.kinbentechnologies.com/update/crm4office-wcal.htm

Exportación de información.

La exportación de información también se ve facilitada al integrar las aplicaciones ofimáticas dentro de los ERP. La facilidad para exportar datos a una hoja de cálculo, la realización de gráficos y su incorporación

a una presentación de diapositivas, o poder incorporar una oferta para enviarla a través de un correo electrónico a un cliente con unos pocos clics de ratón, proporcionan rapidez en la ejecución de la tarea deseada al permitirse el flujo de datos entre el ERP y la suite ofimática.

Por otro lado, estás facilidades tienen como inconveniente que se puede filtrar información delicada de nuestra empresa al exterior.

La posibilidad de exportar información utilizando las aplicaciones ofimáticas como parte del ERP facilitan la realización de todo tipo de documentos, como presentaciones o correos electrónicos utilizando los datos del ERP.

- (•) Verdadera
- ( ) Falsa