



1. Reconocimiento de las características de lenguajes de marcas

Autor	ⓧ Xerach Casanova
Clase	Lenguajes de Marcas
Fecha	@Dec 7, 2020 3:09 PM

1. Lenguajes de marcas.
2. Evolución de los lenguajes de marcas
 - 2.1 GML (Generalized Markup Language).
 - 2.2 SGML (Standard Generalized Markup Language)
 - 2.3 HTML (HyperText Markup Language)
 - 2.4 XML (eXtensible Markup Language)
 - 2.5 Diferencias entre XML y HTML
 - 2.7. Etiquetas.
 - 2.8. Herramientas de edición.
 - Editores XML
 - Procesadores XML
3. XML, estructura y sintaxis.
 - 3.1. El prólogo
 - Declaración XML
 - Declaración del tipo de documento
 - 3.1 El ejemplar. Los elementos
 - 3.1.2. Atributos
4. Documentos XML bien formados
5. Utilización de espacios de nombres en XML
6. Mapa conceptual

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. Todo lo cual es relativo a la estructura del texto o su formato de presentación.

Los lenguajes de marcas van definidos en un documento DTD (Document Type Definition). En el cual se establecen las marcas, los elementos utilizados por dicho lenguaje, sus correspondientes etiquetas, atributos, sintaxis y normas de uso.

Se pueden combinar varios tipos diferentes de lenguajes de marcas en un solo documento.

- **De presentación.** Define el formato del texto
- **De procedimientos.** Igual que el de presentación pero interpretado en el orden en que aparece.
- **Descriptivo o semántico.** Se describen las diferentes partes de su estructura sin especificar como se deben representar.

Según el ámbito en el que se empleen se pueden utilizar distintos lenguajes:

- **Lenguajes de marcas de documentación electrónica:** RTF, TeX, Wikibox, Docbook...
- **Lenguajes de marcas de tecnología de internet:** HTML, XHTML, RSS...
- **Otros lenguajes especializados:** MathML, VoiceML, MusicXML...

2. Evolución de los lenguajes de marcas

Paralelamente a los lenguajes de programación para manejar grandes cantidades de datos, los lenguajes de marcas surgen por la necesidad de describir y estructurar la información en los primeros procesadores de textos.

Inicialmente surgen como conjunto de código que los procesadores de textos introducen en los documentos para dirigir el proceso de presentación mediante la impresora, pero estos lenguajes estaban ligados a las características de la máquina o programa, de manera que el "formateador de textos" no podía abstraerse de las características del procesador de textos y trabajar el lenguaje de marcas del documento de forma independiente.

```
1 <times 14><color verde><centrado> Este texto es un ejemplo para mostrar l
2 <color granate><times 10><cursiva>Para realiza este ejemplo se utilizan e
3 Las partes importantes del texto pueden resaltarse usando la
4 <negrita>negrita</negrita>, o el <subrayar>subrayado</subrayar></times 14>
```

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 añade como medio de presentación la pantalla, de manera que el código desaparecía aunque seguía existiendo para uso interno de las aplicaciones (abstracción). También se incluían otros medios de marcado distintos a la inclusión a mano (combinaciones de teclas, botón...).

Al estar exclusivamente orientado a la presentación de información, pronto surgieron nuevos usos que resolvían gran variedad de necesidades, apareciendo el formato generalizado.

2.1 GML (Generalized Markup Language).

El objetivo de GML es describir documentos de tal modo que el resultado sea independiente de la plataforma y la aplicación utilizada.

Para resolver la falta de formatos de información en los distintos programas, IBM encargó la construcción de un sistema de edición, almacenamiento y búsqueda de documentos.

Para ello crearon un formato estándar que fuera válido para los distintos tipos de documentos legales de la empresa y poder gestionarlos independientemente del departamento o aplicación donde se generó.

2.2 SGML (Standard Generalized Markup Language)

Es la evolución del formato GML que en 1986 dio lugar al estándar ISO 8879.

Lenguaje complejo que requería herramientas y software caro, quedando relegado a grandes aplicaciones industriales.

```
1  <email>
2      <remitente>
3          <persona>
4              <nombre> Pepito </nombre>
5              <apellido> Grillo </apellido>
6          </persona>
7      </remitente>
8      <destinatario>
9          <direccion> pinocho@hotmail.com </direccion>
10     </destinatario>
11     <asunto>¿quedamos?</asunto>
12     <mensaje> Hola, he visto que ponen esta noche la película que quería:
13 </email>
```

2.3 HTML (HyperText Markup Language)

Cuando se creó el World Wide Web (www) se encontró la necesidad de organizar, enlazar y compatibilizar gran cantidad de información procedente de diversos sistemas, para ello surge HTML, que era una combinación de:

- ASCII: es el formato que cualquier procesador de textos sencillo puede almacenar permitiendo transferencia de datos entre ordenadores distintos.
- SGML: lenguaje que permite dar estructura al texto aplicando diversos formatos al texto.

HTML es una versión simplificada de SGML. Debido a su comprensión tuvo rápida aceptación, la cual no tuvo SGML, convirtiéndose así en el estándar general para la creación de páginas web.

Desde entonces, las herramientas de software y navegadores que visualizan HTML no han parado de mejorar.

Sus desventajas son:

- No soporta tareas de impresión y diseño.
- No es flexible (etiquetas limitadas)
- No muestra contenido dinámico.
- La estructura y el diseño están mezclados en el mismo documento.

```

1 <html>
2   <head>
3     <title> Ejemplo de código HTML</title>
4   </head>
5   <body>
6     <p></p>
7     <p>
8       <b>23 de julio de 2020</b>
9     </p>
10    <p><b> Bienvenido al modulo de “Lenguajes de Marcas y Sistemas de
11    <p> En este curso aprender&aacute;s, entre otras cosas:<br/>
12    <ul>
13      <li>Las ventajas que ofrece XML </li>
16    </ul>
17    </p>
18  </body>
19 </html>

```

23 de julio de 2020

Bienvenido al modulo de “Lenguajes de Marcas y Sistemas de Gestión de Información”

En este curso aprenderás, entre otras cosas:

- Las ventajas que ofrece XML
- La creación de documentos bien formados
- La creación de DTD

2.4 XML (eXtensible Markup Language)

Para resolver los problemas de HTML en 1998 se establece el estándar XML, un lenguaje de marcas puramente estructural que no incluye ninguna información relativa al diseño.

Las etiquetas indican el significado de los datos y no su formato.

XML es un conjunto de los siguiente estándares:

- XSL eXtensible Style Language, permite definir hojas de estilo y capacidad de transformación de documentos.
- XML Linking Language, incluye Xpath, Xlink y Xpointer y determinan el aspecto sobre los enlaces entre documentos XML.
- XML Namespaces, proveen un contexto al que se aplican las marcas de un XML.

- XML Schemas, permiten definir restricciones a un XML (los más usados son los DTD).

```

1  <?xml version="1.0" encoding="iso-8859-1"?>
2  <!DOCTYPE biblioteca">
3  <biblioteca>
4      <ejemplar tipo Ejem="libro" titulo="XML practico" editorial="Edi
5      <tipo> <libro isbn="978-2-7460-4958-1" edicion="1" paginas="347";
6      <autor nombre="Sebastien Lecomte"></autor>
7      <autor nombre="Thierry Boulanger"></autor>
8      <autor nombre="Ángel Belinchon Calleja" funcion="traductor"></aut
9      <prestado lector="Pepito Grillo">
10         <fecha_pres dia="13" mes="mar" año="2009"></fecha_pres>
11         <fecha_devol dia="21" mes="jun" año="2009"></fecha_devol>
12     </prestado>
13 </ejemplar>
14 <ejemplar tipo Ejem="revista" titulo="Todo Linux 101. Virtualización
15     <tipo>
16         <revista>
17             <fecha_publicacion mes="abr" año="2009"></fecha_publicaci
18         </revista>
19     </tipo>
20     <autor nombre="Varios"></autor>
21     <prestado lector="Pedro Picapiedra">
22         <fecha_pres dia="12" mes="ene" año="2010"></fecha_pres>
23     </prestado>
24 </ejemplar>
25 </biblioteca>

```

2.5 Diferencias entre XML y HTML

XML:

- Es un perfil de GSML.
- Especifica cómo deben definirse conjuntos de etiquetas de un mismo documentos.
- Modelo de hiperenlaces complejo.
- El navegador es una plataforma para desarrollo de aplicaciones.
- Fin de la guerra de los navegadores y etiquetas propietarias.

HTML:

- 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 alcanza un punto de solución difícil.

Ejemplo fichero con código XML

```

1 <?xml version="1.0" encoding="iso-8859-1"?>
2 <!DOCTYPE libro>
3 <libro>
4   <titulo>XML practico </titulo>
5   <autor>SebastienLecomte</autor>
6   <autor>Thierry Boulanger</autor>
7   <editorial>Ediciones Eni</editorial>
8   <isbn>978-2-7460-4958-1</isbn>
9   <edicion>1</edicion>
10  <paginas>347</paginas>
11 </libro>

```

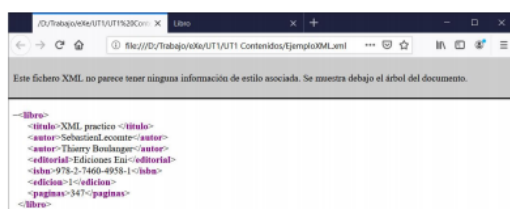
Ejemplo de fichero con código HTML

```

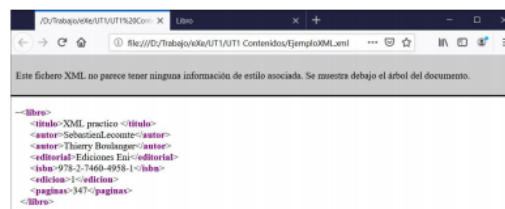
1 <html>
2   <head>
3     <title>Libro</title>
4   </head>
5   <body>
6     <h3>XML practico</h3><br>
7     <p>autores: Sebastien Lecomte,
8       Thierry Boulanger</p>
9     <ul>
10      <li>editorial: Ediciones Eni</li>
11      <li>isbn:978-2-7460-4958-1</li>
12      <li>edicion: 1 </li>
13      <li>paginas: 347</li>
14    </ul>
15  </body>
16 </html>

```

Visualización en Navegador fichero código XML



Visualización en Navegador fichero código HTML



2.7. Etiquetas.

Los lenguajes de marcas utilizan etiquetas especiales intercaladas con texto sin formato, las cuales son interpretadas por intérpretes del lenguaje para procesar el documentos.

Estas se escriben entre ángulos utilizándose dos tipos: etiqueta de inicio < > y etiqueta de fin </ >

Las últimas especificaciones emitidas por W3C indican que siempre vayan en minúsculas.

2.8. Herramientas de edición.

Para trabajar con XML es necesario editar documentos y luego procesarlos.

Editores XML

Basta con utilizar un procesador de textos normal, pero para crear documentos XML complejos se utilizan editores que nos ayuden a crear estructuras y etiquetas.

Algunos incluyen ayuda para la creación de DTD, hojas de estilo CSS o XSL. Un editor gratuito es Amaya.

Procesadores XML

Permiten leer documentos XML y acceder a su contenido y estructura.

Es un conjunto de módulos de software en el que se encuentra el parser, el cual comprueba que el documento cumple las normas establecidas para abrirse.

Pueden ser:

- Validadores (solo trabajan con documentos de tipo válido)
- No validadores (pueden exigir solamente que el documento esté bien formado).

3. XML, estructura y sintaxis.

XML (lenguaje de etiquetas extendido) es un lenguaje de etiquetas creadas por el programador para estructurar y dar forma ordenada a la información. No representa datos por sí mismo.

Ahorra tiempo de desarrollo, permite guardar información de una forma potente y es asimilado por todo tipo de S.O. y dispositivos móviles.

Componentes de documento XML

- Su estructura es de pareja de etiquetas y en árbol.
- El marcado de etiquetas se sitúa entre "<" — "&" — ">";
- Los comentarios se añaden entre las cadenas "<!-- comentario -->" pero no pueden estar antes del prólogo ni dentro de una etiqueta.
- Se componen de una parte opcional llamada prólogo y otra obligatoria llamada ejemplar.

Sus características son:

- Compatible con protocolos HTTP y URL.
- Todo documento que verifique las reglas XML está conforme con SGML.
- No se requieren conocimientos de programación.
- Fáciles de crear.
- Difusión de documentos asegurada.
- Legible para los humanos.
- Diseño formal y conciso.
- Es extendible y adaptable.
- Orientado a objetos.
- Se compone exclusivamente de datos de marcados y datos carácter entremezclados.

El proceso de creación de documentos XML pasa por varias etapas en las que su éxito depende de la calidad de la anterior:

1. Especificación de requisitos.
2. Diseño de etiquetas.
3. Marcado de documentos.

Se deben crear documentos con estructuras fácilmente actualizables para necesidades futuras.

3.1. El prólogo

El prólogo precede al ejemplar del documento. Facilita el procesado de la información y está dividido en

Declaración XML

Ha de ser la primera línea o da error. Es opcional y eso permite que HTML y SGML sea procesado como XML.

Puede tener tres funciones:

1. Declaración de versión XML:
2. Declaración de codificación empleada para representar caracteres.

3. Declaración de la autonomía del documento. Informa si el documento necesita de otro para su interpretación. (sí /no - yes/no)

<?xml versión="1.0" encoding="iso-8859-1" standalone="yes" ?>

1 2 3

Los estándar ISO 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 Shift_JIS	Japonés

Declaración del tipo de documento

Define que tipo de documento vamos a crear para que se procesado correctamente.

```
<!DOCTYPE nombre_tipo ....>
```

3.1 El ejemplar. Los elementos

Contiene los datos reales del documento y está formado por elementos anidados.

Los elementos son los distintos bloques de info que permiten definir la estructura del documento. Los elementos pueden estar formados por otros elementos o atributos.

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

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

El ejemplar es el elemento raíz (root) y todos los datos de un documento XML han de pertenecer a un elemento del mismo.

Los nombres de etiquetas deben ser autodescriptivos.

Las reglas que han de cumplir los documentos XML son:

- Solo un elemento root.
- Todos los elementos tienen una etiqueta de inicio y otra de cierre. En caso de que existan elementos vacíos se empleará `<elemento/>`.
- Los elementos no pueden tener anidados otros elementos sin cerrar.
- Los nombres de las etiquetas de inicio y cierre deben ser idénticos.
 - No contener espacios.

- Respetar mayús/minús.
- No puede comenzar por el carácter dos puntos ":".
- No puede comenzar por xml, XML, Xml, etc...
- El contenido de los elementos no puede contener la cadena]] por compatibilidad con SGML. Tampoco se puede utilizar los caracteres < > & " ' sustituibles por:

Carácter	Cadena
>	>
<	<
&	&
"	"
'	'

- Los caracteres especiales £, ©, ®... se debe utilizar &#D (código decimal) &#H (código hexadecimal) correspondiente al carácter en UNICODE. Ejemplo: € es € o €

3.1.2. Atributos

Permiten añadir propiedades a los elementos. No se pueden organizar en jerarquía y no pueden contener otro elemento o atributo. No reflejan estructura lógica. Tampoco se debe utilizar para contener información susceptible de ser dividida.

<pre> 1 <?xml version="1.0" encoding="UTF-8" standalone="yes" ?> 2 <!DOCTYPE biblioteca > 3 <biblioteca> 4 <ejemplar tipo_ejem="libro" titulo="XML práctico" editorial="Ediciones Eni"> 5 <tipo> <libro isbn="978-2-7460-4958-1" edicion="1" paginas="347"/> </tipo> 6 <autor nombre="Sebastien Lecomte"/></autor> 7 <autor nombre="Thierry Boulanger"/></autor> 8 <autor nombre="Angel Belinchon Calleja" funcion="traductor"/></autor> 9 <prestado lector="Pepito Grillo"> 10 <fecha_pres dia="13" mes="mar" año="2009"/></fecha_pres> 11 <fecha_devol dia="21" mes="jun" año="2009"/></fecha_devol> 12 </prestado> 13 </ejemplar> 14 </biblioteca> </pre>	<pre> <--biblioteca> <--ejemplar tipo_ejem="libro" titulo="XML práctico" editorial="Ediciones Eni"> <--tipo> <libro isbn="978-2-7460-4958-1" edicion="1" paginas="347"/> </tipo> <autor nombre="Sebastien Lecomte"/> <autor nombre="Thierry Boulanger"/> <autor nombre="Angel Belinchon Calleja" funcion="traductor"/> <--prestado lector="Pepito Grillo"> <fecha_pres dia="13" mes="mar" año="2009"/> <fecha_devol dia="21" mes="jun" año="2009"/> </prestado> </ejemplar> </biblioteca> </pre>
--	--

Los atributos se definen y dan valor dentro de la etiqueta de inicio o en un elemento vacío, a continuación del nombre del elemento o de la definición de

otro atributo, se separan por espacios. Se definen con comillas simples o dobles y siguen las mismas reglas que los elementos.

4. Documentos XML bien formados

Los documentos XML deben verificar reglas sintácticas que define la recomendación del W3C para el estándar XML, que son:

- Ha de tener definido un prólogo con la declaración XML completa.
- Un único elemento raíz para cada documento, el resto de elementos y contenidos están anidados.
- Hay que cumplir las reglas sintácticas de XML para elementos y atributos.

5. Utilización de espacios de nombres en XML

Permiten utilizar etiquetas idénticas para distintos tipos de información de texto. Dan un nombre único a cada elemento indexándolos según el nombre del vocabulario adecuado asociados a un URI (identificador global único).

De esta forma se evita ambigüedad al juntar dos o más documentos en los que se han definido etiquetas con el mismo nombre. A las etiquetas ambiguas se les precede con un prefijo que determina el contexto al que pertenece la etiqueta seguido de dos puntos:

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

A esta etiqueta se le denomina "nombre cualificado". El prefijo no puede utilizar espacios ni caracteres especiales y tampoco puede comenzar por un dígito.

Para utilizar un prefijo de un espacio de nombres es necesario declarar el espacio de nombres y asociar un índice con el URI asignado al espacio de nombres con el atributo especial xmlns. Esto se hace entre el prólogo y el ejemplar.

XML de alumnos:

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

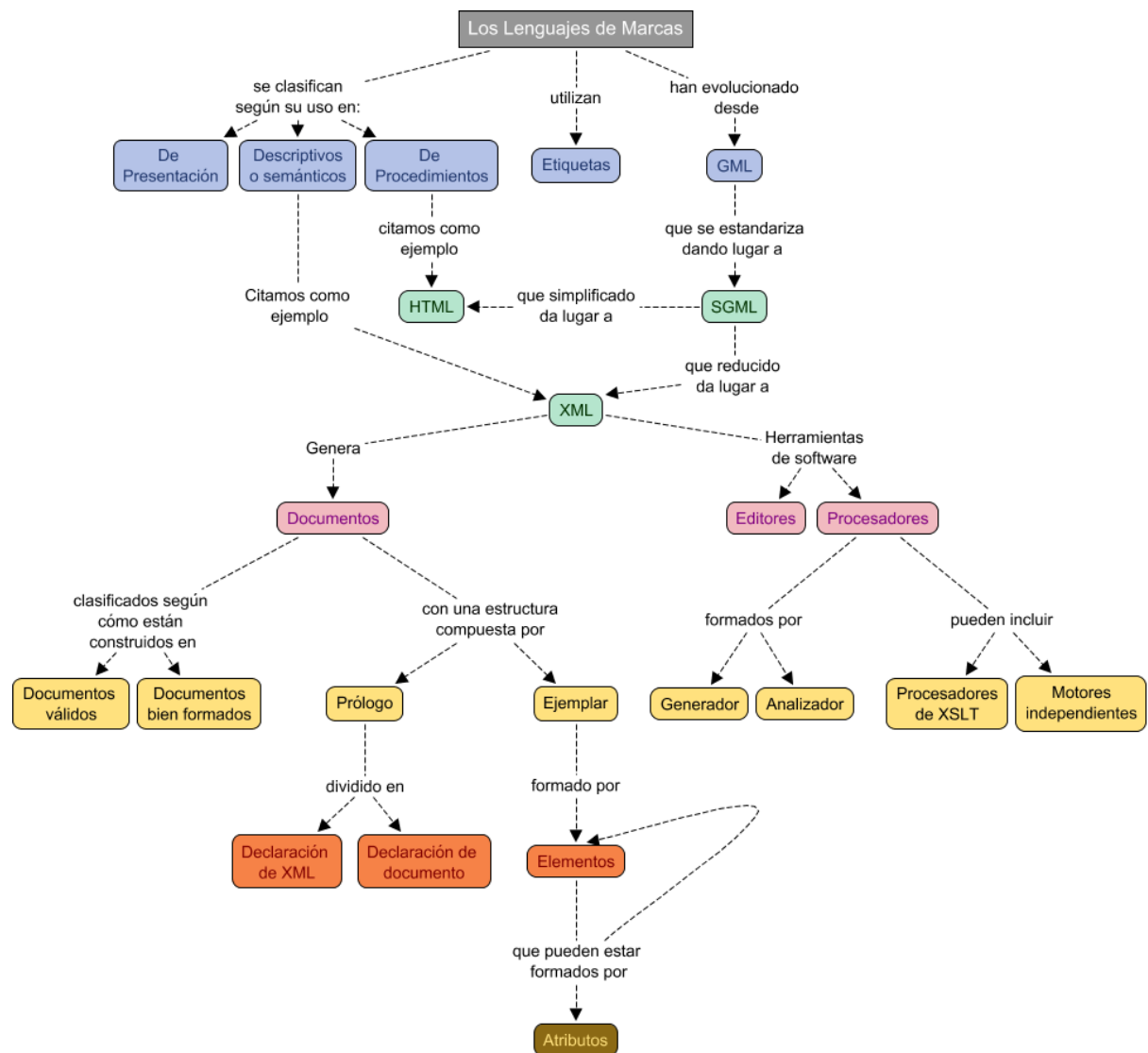
XML de profesores

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

Uniendo estos dos XML utilizando espacios de nombres quedaría:

```
1 <?xml version="1.0" encoding="iso-8859-1" standalone="yes" ?>
2 <!DOCTYPE miembros>
3 <alumnos xmlns:alumnos="http://DAM/alumnos">
4 <profesores xmlns:profesores="http://DAM/profesores">
5 <asistentes>
6   <alumnos:nombre>Fernando Fernández González</alumnos:nombre>
7   <alumnos:nombre>Isabel González Fernández</alumnos:nombre>
8   <alumnos:nombre>Ricardo Martínez López</alumnos:nombre>
9   <profesores:nombre>Pilar Ruiz Pérez</profesores:nombre>
10  <profesores:nombre>Tomás Rodríguez Hernández</profesores:nombre>
11 </asistentes>
```

6. Mapa conceptual





2. Utilización de lenguajes de marcas en entornos web.

Autor	Xerach Casanova
Clase	Lenguajes de Marcas
Fecha	@Mar 29, 2021 9:18 PM

1. Introducción a la utilización de lenguajes de marcas en entornos web

1.1. HTML: evolución y versiones

1.2. HTML y XHTML

Diferencias sintácticas y estructurales con HTML

1.3. Estructura de un documento HTML

Prólogo

Ejemplar

1.4. Identificación de etiquetas y atributos HTML

1.5. Clasificación de los atributos comunes según su funcionalidad

Atributos básicos o globales.

Atributos para internacionalización.

Atributos de eventos y atributos para elementos que pueden obtener foco

2. Elementos HTML

2.1. Elementos de estructura básica de un documento

2.2. Elementos de la sección cabecera

Elementos contenedores

Elementos no contenedores

2.3. Encabezados y párrafos

2.3.1. Salto de línea y espacios en blanco

2.3.2. Comentarios

2.4. Semántica a nivel texto

2.5. Elementos de listas

2.6. Elementos de tablas

2.7. Elementos de formularios

2.7.1. Declaración de formulario

2.7.2. Campos de formulario - input

2.7.3. Campos de formulario - Área de texto

2.7.4. Campos de formularios - Lista desplegable

2.7.5. Campos de formularios - checkbox

2.7.6. Botones de formulario

2.7.7.- Otros campos de formulario

2.7.8. Campos de formulario con formato de fecha

2.7.9. Campos de formulario - Rangos

2.7.10. Organización de formularios

2.8. Multimedia

2.9. Secciones y etiquetas semánticas

2.10. Elemento iframe

3. Hojas de estilo o CSS

3.1. Cómo incluir CSS en un documento HTML o XHTML

3.1.1. Elemento span

3.2. Sintaxis de las reglas de estilo

3.3. Cascada y herencia de estilos

3.4. Selectores

3.4.1. Selectores de clase

3.4.2. Selectores de ID

3.4.3. Selectores descendientes

3.4.4. Selector hijo

3.5. Propiedades principales

3.5.1. Propiedades de color y fondo

3.5.2. Propiedades de fuente
3.5.2. Propiedades de texto
3.5.3. Propiedades de lista
3.5.5. Propiedad display
3.6. Avanzado: Modelo de cajas
3.6.1. Propiedades de caja
3.6.2. Unidades de tamaño
3.6.3. Ejemplos del modelo cajas
3.6.4. Posicionamiento
Mapa Conceptual

1. Introducción a la utilización de lenguajes de marcas en entornos web

1.1. HTML: evolución y versiones

HTML es un estándar reconocido por todos los navegadores, los cuales visualizan una página HTML de forma muy similar, independientemente del SO.

El origen fue un sistema de hipertexto para compartir documentos electrónicos en 1990. Las dos primeras propuestas de estándar HTML llegaron a ser oficiales (HTML y HTML+).

- HTML 2.0 fue la primera versión oficial de HTML. IETF (Internet Engineering Task Force) publicó el estándar en 09/1995.
- HTML 3.2 el 14/01/1997 W3C se incorporan los applets de java y texto alrededor de imágenes.
- HTML 4.0 se publica el 24/04/1998 se incorporan las hojas de estilo CSS y posibilidad de incluir pequeños programas.
- HTML 4.01 el 24/12/1999 se actualiza la versión anterior y W3C no actualiza hasta marzo de 2007 debido a la presión de WHATWG (Web Hypertext Application Technology Working Group).
- HTML 5 es el estándar actual y la versión más avanzada, la cual ha ido evolucionando (5.1, 5.2 y 5.3).

El cambio más significativo de HTML 5 es el paso a CSS de todo lo relativo a la presentación del documento. HTML solo se encarga de la estructura, información y semántica.

1.2. HTML y XHTML

XHTML (Extensible HyperText Markup Language) es similar a HTML, siendo una adaptación a XML.

HTML 4.01 es incluido en XHTML 1.0 debido a que solo añade unas pocas modificaciones.

XHTML soluciona los problemas del desorden y permisividad de HTML añadiendo normas en la forma de escribir etiquetas y atributos. Permite:

- Sencillez a la hora de editar y mantener el código.
- Al ser regular, es más fácil escribir código que lo procese.
- Permite utilizar herramientas creadas para trabajar XML genéricos, editores XSLT, etc...

Diferencias sintácticas y estructurales con HTML

El HTML se considera XML bien formado cuando:

- Toda la página está contenida en el elemento raíz <html>
- Los nombres de etiquetas y atributos están siempre en minúsculas.
- El valor de los atributos entre comillas (incluso numéricos).
- En atributos que el nombre coincide su valor, no se da el valor por entendido (no se pueden comprimir).
- Las etiquetas se cierran siempre. En etiquetas vacías se utiliza la etiqueta que se abre y se cierra de una sola vez (ejemplo
).

Los diseñadores web suelen trabajar con HTML y los desarrolladores con XHTML, pero los tres primeros requisitos son buena práctica y se deben cumplir.

En errores de sintaxis, HTML muestra la mayor parte del contenido posible, pero XHTML puede dar errores de sintaxis que no muestren documento.

1.3. Estructura de un documento HTML

Igual que en XML, tiene prólogo y ejemplar.

Prólogo

Le indica al navegador el documento que se va a iniciar y la versión de HTML. Para la versión 4 hay tres tipos:

- HTML 4.0 Strict. Es la DTD utilizada por defecto. No permite uso de elementos declarados "deprecated" en otras versiones o recomendaciones HTML.

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

- HTML 4.0 Transitional. Permite el uso de los elementos deprecated.

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

- HTML 4.0 Frameset. Una variante para documentos que usan frames, en los cuales se reemplaza la etiqueta body por frameset.

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

- HTML5 solo utiliza una declaración

```
<!DOCTYPE html>
```

Ejemplar

Un documento HTML está dentro de las etiquetas <html></html>, que a su vez se divide en:

- Cabecera: <head></head> contiene info sobre el título de la página, autor, palabras clave, etc... Es obligatorio definir el título del documento con <title></title> (el título aparece en la barra del título de la parte superior, el resto de información no se muestra).
- Cuerpo: contiene la info que se presenta en pantalla y se delimita entre las etiquetas <body></body> o <frameset></frameset>

1.4. Identificación de etiquetas y atributos HTML

Una de las diferencias entre HTML y XML, es que HTML tiene etiquetas limitadas por las definidas en el propio lenguaje.

HTML tiene una gran cantidad de etiquetas, pero no son suficientes para crear páginas complejas, por ello se añade información adicional a las etiquetas mediante atributos (dando lugar a elementos).

Estos atributos tienen un conjunto de valores que se les puede asignar y si alguno no es válido, el navegador los ignora.

1.5. Clasificación de los atributos comunes según su funcionalidad

Atributos básicos o globales.

Se usan en casi todas las etiquetas.

- **name="texto"**. Permite asignar el nombre texto a un objeto HTML
- **title="texto"**. Asigna un título a un elemento HTML mejorando su accesibilidad. Este título es mostrado en navegadores cuando se pasa el ratón por encima, útil en elementos: a, link, img, object, abbr y acronym

- **id="texto"**. Permite identificar el elemento HTML sobre el que se aplica de forma individual. Útil trabajando con CSS y Javascript. No puede empezar por números y solo puede contener letras, números, guiones medios o bajos.
- **style="texto"**. Aplica al elemento el estilo llamado texto directamente.
- **class="texto"**. Aplica al elemento el estilo "texto" definido en las CSS. No puede empezar por números y solo puede contener letras, números, guiones medios o bajos.

Atributos para internacionalización.

Lo utilizan páginas en varios idiomas para indicar de forma explícita el idioma de sus contenidos.

- **dir**. Indica la dirección del texto, solo puede tomar dos valores: **ltr** de izda. a dcha o **rtl** de derecha a izquierda.
- **lang="código"**. Especifica el idioma del elemento mediante un código predefinido en el documento RFC1766 (en, en-US, ja, es, fr, fr-CA...)
- **xml:lang="código"**. Especifica el idioma del elemento mediante un código definido según RFC 1766.

en XML xml:lang tiene más prioridad que lang y es obligatorio si se incluye ese atributo.

Atributos de eventos y atributos para elementos que pueden obtener foco

Utilizados en webs dinámicas con JavaScript.

2. Elementos HTML

Un elemento HTML está formado por

- Etiqueta de apertura.
- Cero o más atributos.
- Opcionalmente un texto encerrado por la etiqueta (no todas las etiquetas encierran texto).
- Etiqueta de cierre.

Pueden ser de dos tipos

- **Elementos en línea**: ocupan el espacio necesario para mostrar sus contenidos
- **Elementos de bloque**. Siempre empiezan en una nueva línea y ocupan todo el espacio disponible hasta el final de la línea aunque el contenido no llegue hasta allí. Puede ser texto, elementos en línea u otros elementos de bloque.

```
<!DOCTYPE html>

<html>
  <head>
    <title>Ejemplo de la diferencia entre los elementos en línea
      y los elementos de bloque
    </title>
  </head>
  <body>
    <h1>Los encabezados son elementos de bloque.</h1>
    <p>Y los párrafos también.</p>
    <a href="#">Los enlaces son elementos de línea</a>
    <p>Incluso si esta definido dentro de un párrafo,
      <strong>un texto resaltado</strong>
      sigue siendo un elemento en línea.</p>
  </body>
</html>
```

Los encabezados son elementos de bloque.

Y los párrafos también.

[Los enlaces son elementos de línea](#)

Incluso si esta definido dentro de un párrafo, **un texto resaltado** sigue siendo un elemento en línea.

2.1. Elementos de estructura básica de un documento

<html></html> define el inicio y el final del documento HTML. Contiene

<head></head> define inicio y final de cabecera del documento

<body></body> define inicio y final del cuerpo del documento.

```
<!DOCTYPE html>

<html>
  <head>
    <title>Ejemplo de la estructura de un documento HTML</title>
  </head>
  <body>
    Aquí; es donde se coloca la información que se quiere visualizar en el navegador.
  </body>
</html>
```

2.2. Elementos de la sección cabecera

Elementos contenedores

Elemento	Descripción
title	Título del documento. Elemento obligatorio.
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. (Eliminado de los estándares web aunque todavía algún navegador lo soporta)
link	Enlaces a documentos externos de librerías
meta	Metadatos sobre la página, como la codificación de caracteres, descripción o autores.

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="UTF-8">
    <title>El título es obligatorio</title>
  </head>
  <body>
    ...
  </body>
</html>
```

UTF-8 permite que los acentos se vean bien.

2.3. Encabezados y párrafos

Para agrupar párrafos se usa el elemento <p></p>

Para encabezados se utilizan 6 elementos

<h1>, <h2>, <h3>, <h4>, <h5>, <h6>

A menor número mayor importancia de encabezado. Se utilizan para organizar jerárquicamente el contenido.

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="UTF-8">
```

```
<title>Párrafos y encabezados</title>
</head>
<body>
  <h1>Equipos</h1>
  <p>Primer párrafo</p>
  <p>Segundo párrafo</p>
  <h2>Recién ascendidos</h2>
  <p>...</p>
  <h1>Jugadores</h1>
  <p>...</p>
</body>
</html>
```

Equipos

Primer párrafo

Segundo párrafo

Recién ascendidos

Jugadores

...

2.3.1. Saltos de línea y espacios en blanco

Los documentos HTML ignoran los saltos de línea y para varios espacios consecutivos solo muestran uno.

 introduce saltos de línea y añade espacios en blanco.

[illegible]

Esto se ve en una línea sin espacios de más

Aquí hay espacios

Esto
introduce un salto de línea

2.3.2. Comentarios

se pueden introducir comentarios de una o varias líneas, no son procesados por los navegadores.

```
<!-- comentario
de varias
```

2.4. Semántica a nivel texto

- **<a>** para definir hipervínculos.
 - Su atributo más importante es **href="URL"**, a veces se utiliza **href="#"** para referirse a la misma página del vínculo.
 - **target** permite elegir donde se abrirá el vínculo.
 - **target="_blank"** en ventana nueva.
 - **target="_self"** para que se abra en la misma (valor por defecto).
- **** para representar texto importante.
- **** para indicar énfasis.
- **
** salto de línea.
- **<small>** para comentarios accesorios (letra pequeña).

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="UTF-8">
    <title>Semántica a nivel de texto</title>
  </head>
  <body>
    Texto marcado como <strong>importante</strong>.
    <br>
    Texto con <em>énfasis</em>
    <br>
    Texto marcado <small> con el elemento small </small>
    <br>
    Pulsa <a href="https://www.w3.org/">aquí</a> para ir a la página del W3C.
  </body>
</html>
```

Texto marcado como **importante**.
 Texto con *énfasis*
 Texto marcado con el elemento small
 Pulsa [aquí](https://www.w3.org/) para ir a la página del W3C.

2.5. Elementos de listas

Hay tres tipos: Ordenadas, desordenadas y de definición.

Elemento	Descripción
<u>ul</u>	Delimita los elementos que forman una lista desordenada
<u>ol</u>	Delimita los elementos que forman una lista ordenada. Tiene varios atributos.
<u>dl</u>	Delimita los elementos que forman una lista de definición
<u>li</u>	Cada uno de los elementos de una lista ordenada o desordenada.
<u>dt</u>	Cada uno de los términos que se definen de una lista de definición.
<u>dd</u>	Cada una de las definiciones de una lista de definición.

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="UTF-8">
    <title>Listas</title>
  </head>
  <body>
    <h1>Ejemplo de lista desordenada: Módulos de 1º de ASIR</h1>
```

```

<ul>
  <li>Fundamentos de Hardware</li>
  <li>Gestión de Bases de Datos</li>
</ul>
</h1>Ejemplos de listas ordenadas: Módulos de 1º de ASIR</h1>
Comenzando en 1
<ol>
  <li>Fundamentos de Hardware</li>
  <li>Gestión de Bases de Datos</li>
</ol>
Comenzando en 4
<ol start = "4">
  <li>Fundamentos de Hardware</li>
  <li>Gestión de Bases de Datos</li>
</ol>
<h1>Ejemplo de lista de definición: Módulos de 1º de ASIR</h1>
<dl>
  <dt>Fundamentos de Hardware</dt>
  <dd>Componentes físicos de un ordenador</dd>
  <dt>Gestión de Bases de Datos</dt>
  <dd>Diseño y uso de bases de datos relacionales</dd>
</dl>
</body>
</html>

```

Ejemplo de lista desordenada: Módulos de 1º de ASIR

- Fundamentos de Hardware
- Gestión de Bases de Datos

Ejemplos de listas ordenadas: Módulos de 1º de ASIR

Comenzando en 1

1. Fundamentos de Hardware
2. Gestión de Bases de Datos

Comenzando en 4

4. Fundamentos de Hardware
5. Gestión de Bases de Datos

Ejemplo de lista de definición: Módulos de 1º de ASIR

Fundamentos de Hardware
 Componentes físicos de un ordenador
 Gestión de Bases de Datos
 Diseño y uso de bases de datos relacionales

2.6. Elementos de tablas

Los elementos de las tablas son los siguientes (sin necesidad de usarlos todos)

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.
caption	Para añadir una leyenda a la tabla

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="UTF-8">
    <title>Tablas</title>
  </head>
  <body>
    <table>
      <caption>Tabla de socios</caption>
      <tr>
        <th>Nombre</th>
        <th>Apellido</th>
        <th>Edad</th>
      </tr>
      <tr>
        <td>Juan</td>
        <td>Puertas</td>
        <td>54</td>
      </tr>
      <tr>
        <td>Eva</td>
        <td>Montes</td>
        <td>44</td>
      </tr>
    </table>
  </body>
</html>
```

Tabla de socios		
Nombre	Apellido	Edad
Juan	Puertas	54
Eva	Montes	44

2.7. Elementos de formularios

Recogen información que el usuario introduce en el navegador. Se deben validar los datos para detectar errores en local en caso de haberlos y así evitar sobrecargar el servidor con tareas innecesarias.

2.7.1. Declaración de formulario

<form></form> se utiliza para abrir y cerrar formulario y permite especificar los siguientes atributos:

- **name** - Nombre de formulario
- **action** - Acción que se ejecuta cuando se envía el formulario.
- **enctype** - formato en el que se envían los valores del formulario.
- **method** - método de transmisión de datos (get o post).
 - **GET**
 - Permite pasar valores en ASCII hasta 100 caracteres.

- Las variables que se transmiten se ven en la URL y van concatenadas con (&)
- **POST**
 - Permite pasar valores de variables y archivos.
 - Carece de restricciones de longitud.
 - Las variables no son visibles en URL.

2.7.2. Campos de formulario - input

el elemento **<input>**, sin etiqueta de cierre, se utiliza para varios tipos de control según se declare el atributo type. Además otros atributos son:

- **name** - nombre del campo.
- **size** - caracteres visibles en el campo (por defecto 20).
- **maxlength** - máximo de caracteres permitidos a introducir.
- **value** - valor por defecto del campo.
- **placeholder** - valor sugerido, se representa en gris y desaparece al hacer foco en él.
- **readonly** - no puede ser modificado.
- **autofocus** - se sitúa el foco en él en cada recarga de página.
- **required** - campo obligatorio.

```
<input type="text" name="usuario" size="30" maxlength="20" placeholder="Escriba aquí el nombre de usuario" required>
```

Usuario

2.7.3. Campos de formulario - Área de texto

Son campos de texto que permiten un mayor número de caracteres y se especifican con la etiqueta **<textarea>** **</textarea>**. Sus atributos principales son:

- **name**: nombre del campo.
- **rows**: número de filas (reemplazable por la propiedad CSS height)-
- **cols**: número de columnas (reemplazable por CSS width).

También puede utilizar los siguientes atributos input: **placeholder, readonly, autofocus, maxlength, required.**

```
Describe a continuación sus intereses: <br> <textarea name="area"></textarea>
```

Describe a continuación sus intereses:

2.7.4. Campos de formularios - Lista desplegable

Permite al usuario seleccionar un valor entre diferentes opciones. Se usan las siguientes etiquetas.

<select></select> para definir la lista. Sus atributos son:

- **name** - Nombre del elemento.
- **size** - número de elementos de la lista que se mostrarán.
- **multiple** - permite seleccionar varias opciones de la lista.

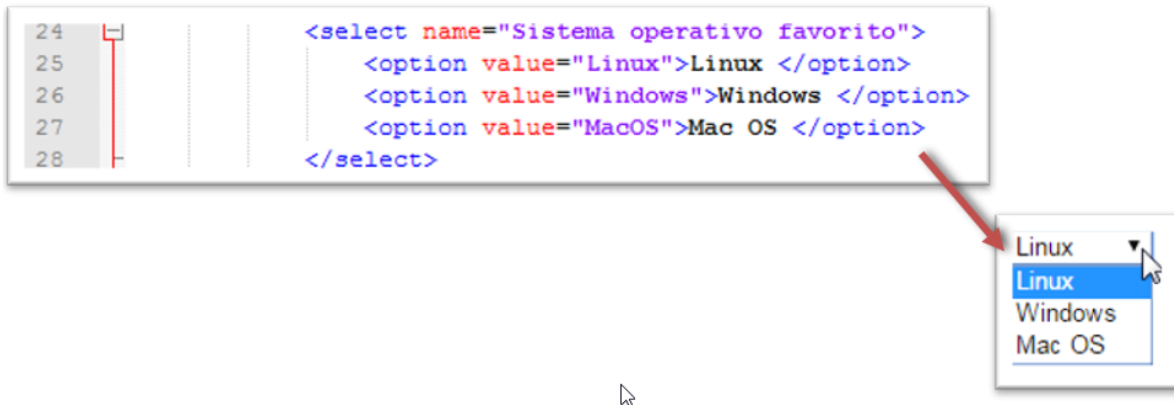
<option></option> para definir cada elemento de la lista. Sus atributos son:

- **selected** - seleccionado por defecto.

- **valor** - valor que se transmite cuando se envía el formulario.

Por último seleccione el sistema operativo de su ordenador:

```
<select name="Sistema operativo favorito">
  <option value="Linux">Linux </option>
  <option value="Windows">Windows </option>
  <option value="MacOS">Mac OS </option>
</select>
```



2.7.5. Campos de formularios - checkbox

Para que el usuario indique una o varias opciones: `<input type="checkbox">`

- **name** - obligatorio.
- **checked** - marcado por defecto.
- **value** - valor a enviar en formulario.

Estoy conforme con la política de privacidad de la página

```
<input type="checkbox" name="conforme" checked>
```

Estoy conforme con la política de privacidad de la página ☒

2.7.6. Botones de formulario

Botón de envío

`<input type="submit">`

Envía el formulario a un destinatario (servidor web, email...), dependiendo del valor del atributo action de la etiqueta `<form>`. Su atributo es:

- **value** - indica el texto del botón.

Botón de anulación

`<input type="reset">`

Borra el contenido de los campos cumplimentados del formulario. También utiliza el atributo value.

2.7.7.- Otros campos de formulario

Campo	Etiqueta	Notas
Oculto	<code><input type="hidden"></code>	
De contraseña	<code><input type="password"></code>	
Para envío de ficheros	<code><input type="file"></code>	Requiere formulario definido con post y enctype="multipart/form-data"
Para recoger correo electrónico	<code><input type="email"></code>	Valida formato automáticamente
Para recoger una URL	<code><input type="url"></code>	Valida formato automáticamente
Para recoger números enteros en un rango	<code><input type="number"></code>	Atributos: max – valor máximo min – valor mínimo step – incremento del contador

2.7.8. Campos de formulario con formato de fecha

Se pueden utilizar diferentes variaciones dependiendo de lo que queramos recoger:

`<input type="datetime-local">` Día, mes, año y hora

`<input type="date">` Día, mes, año

`<input type="month">` Mes y año

`<input type="time">` Hora. Sus atributos son:

- **min** - menor hora seleccionable
- **max** - mayor hora seleccionable

2.7.9. Campos de formulario - Rangos

Permite seleccionar un valor dentro de un rango.

`<input type="range">`

Atributos:

- **min** - valor mínimo del rango.
- **max** - valor máximo del rango.
- **step** - valor del incremento del contador.
- **value** - valor inicial.

2.7.10. Organización de formularios

Etiqueta <label>

Permite asociar una etiqueta con su nombre a cada campo, además constituye una ayuda de usabilidad para invidentes.

```
<label for="conforme">Acepto el acuerdo de licencia</label>
<input type="checkbox" name="licencia" id="conforme" value="ok">
```

En este ejemplo label se asocia al input mediante el atributo for y el valor del identificador (id) de la etiqueta input.

Acepto el acuerdo de licencia ☐

2.8. Multimedia

Elemento	Descripción
img	Permite insertar una imagen en una página web.
audio	Para insertar audio
video	Para insertar vídeo
object	Para incrustar contenido multimedia (audio, vídeo, ficheros PDF, applets...)

Elemento

No tiene etiqueta de cierre y sus atributos son:

- **src** - ruta de la imagen, local o URL, en general se utiliza jpg, png y gif.
- **alt** - Texto alternativo. Es obligatorio.
- **height** - Altura, si no es específica se escoge el original y si se especifica, se escala. HTML5 lo soporta.
- **width** - Anchura de la imagen.

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset='UTF-8'>
    <title>Imágenes</title>
  </head>
  <body>
    <p> Imagen en internet </p>
    <img src='https://i.blogs.es/aa1b9a/luna-100mpx/450_1000.jpg' alt='la luna'>
    <p>Imagen no encontrada</p>
    <img src='http://rutaincorrecta.jpg' alt='la luna'>
    <p>Imagen local</p>
    <img src='index.jpeg' alt='logo'>
  </body>
</html>
```

Imagen en internet



Imagen no encontrada
la luna

Imagen local



En la segunda imagen al no encontrarla muestra texto alternativo.

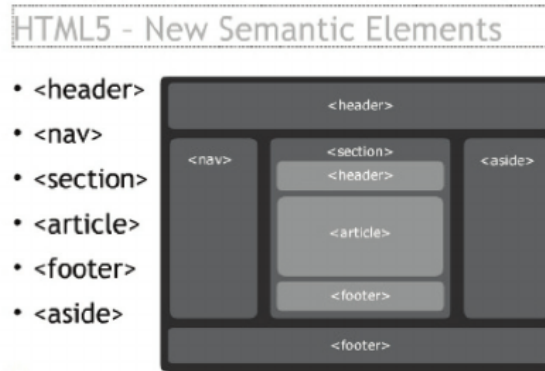
2.9. Secciones y etiquetas semánticas

Se usa el elemento <div> para agrupar otros elementos o secciones, tanto para organizar contenido como para posicionarlo con hojas de estilo.

En HTML5 aparecen varias etiquetas para estructurar contenido de página. Solo se debe usar <div> cuando no haya una etiqueta más apropiada.

- **<header>** Contenido introductorio. Suele contener elementos encabezado: h1, h2...
- **<aside>** contenido parcialmente relacionado con el principal. No tiene por qué ser en el lateral.
- **<footer>** Contenido sobre la sección correspondiente, como el autor. No tiene por qué ser la parte inferior.

- **<section>** Una sección genérica dentro del documento.
- **<article>** Representa un elemento que se puede distribuir de manera independiente o reutilizable.
- **<nav>** Contiene vínculos, internos o externos, se suele usar para barra de navegación.



2.10. Elemento iframe

Permite integrar una web dentro de otra, sustituye a frameset de HTML4, ya obsoleta. Atributos:

height - altura, por defecto en píxeles, se puede fijar con CSS y se usa barra de desplazamiento si no cabe todo el contenido.

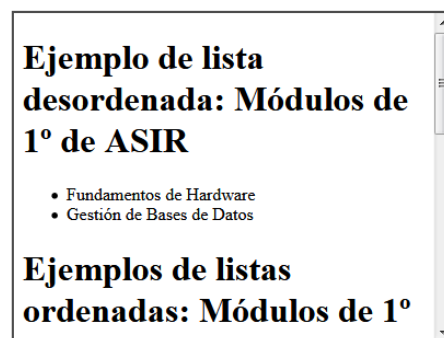
width - como el anterior.

src - ruta del contenido inicial del iframe

name - nombre, para referirse al iframe desde un vínculo.

[Ejemplo de listas](#) [Ejemplo de tablas](#)

Aquí está el iframe



El contenido del iframe cambia al seguir los vínculos, para eso se establece en el atributo target del vínculo el nombre del iframe.

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="UTF-8">
    <title>iframe</title>
  </head>
  <body>
    <a href = "listas.html" target = "contenido">Ejemplo de listas</a>
    <a href = "tablas.html" target = "contenido">Ejemplo de tablas</a>
    <p>Aquí está el iframe</p>
    <iframe height = "300" width = "400" name = "contenido" src = "listas.html">
  </body>
</html>
```

3. Hojas de estilo o CSS

CSS (Cascading Style Sheets) permite controlar el estilo y formato de múltiples web al mismo tiempo. Hasta su aparición, las páginas web definían su aspecto en cada elemento de las etiquetas HTML.

CSS separa los contenidos de la página de su aspecto. Cualquier cambio en el estilo CSS afecta a todas las páginas vinculadas a ella en las que aparezca ese elemento.

Están compuestas por una o más reglas de estilo aplicadas a HTML o XML. Una vez creados los contenidos de estos se utiliza CSS para definir el formato de sus elementos.

CSS aparece poco después del lenguaje de etiquetas SGML. En 1995 W3C añade el desarrollo y estandarización CSS a su grupo de trabajo.

CSS1 (1996).

CSS2 (1998)

CSS3 (Junio de 1999). Se divide en varios documentos separados llamados módulos. Un módulo primero debe pasar por la fase de candidato antes de ser aprobado.

CSS4. Última versión. Algunos módulos están especificación nivel 3 mientras otros (como selectores) están en nivel 4.

3.1. Cómo incluir CSS en un documento HTML o XHTML

Se hace de 3 formas distintas.

- **Declaración en línea:** desaconsejada.

```
<p style="color: red;"> ... </p>
```

- **Declaración interna:** se declaran en el encabezado de la página mediante <style>

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset='UTF-8'>
    <title>CSS interna</title>
    <style>
      p {color:green}
    </style>
  </head>
  <body>
    <p>Hola</p>
  </body>
</html>
```

- **Declaración externa:** se declara la hoja de estilo en otro documento (con extensión .css) y se llama a ella desde las páginas con la etiqueta link

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

El código del .css no tendrá etiqueta de declaración de estilo.

También se pueden llamar a hojas de estilo externas con la etiqueta @import

```
<style type="text/css">
  @import url("formato1.css");
</style>
```

3.1.1. Elemento span

Para dar estilo a un texto no marcado.

```
<p> Parte de este párrafo <span style="color:red">está en rojo</span>
```

3.2. Sintaxis de las reglas de estilo

Cada uno de los estilos de un CSS se denomina regla, formada por: selector, llave de apertura, declaración y llave de cierre.

```
p { color: blue;}
```

3.3. Cascada y herencia de estilos

Ante estilos contradictorios, el navegador aplica la siguiente precedencia:

1. Declaración en línea.
2. Declaración interna.
3. Declaración externa.
4. Propiedades por defecto del navegador.

Las hojas de estilo tienen herencia en sus propiedades. Si tenemos varios elementos HTML anidados, estos heredan estilos de los externos, siempre y cuando no tengan los suyos propios definidos.

3.4. Selectores

Los selectores permiten identificar a qué elementos de nuestro código HTML vamos a aplicar el estilo definido:

Selector universal (*)

Se aplica a todos los elementos de la página.

```
* {  
  margin: 10px;  
  padding: 5px;  
};
```

Selectores de etiqueta

Se aplica el estilo a la etiqueta.

```
p { text-align: center;}
```

Selector múltiple

Se aplica el estilo a todas las etiquetas.

```
p, h1, h2 { text-align: center;}
```

3.4.1. Selectores de clase

Se utilizan para asignar estilos a etiquetas de una clase determinada, sin que afecten a todas las etiquetas del mismo tipo.

En HTML:

```
<p class="parrafoCentrado"> .... </p>
```

en CSS:

```
p.parrafoCentrado {text-align: center; }
```

Utilizando la misma definición omitiendo el identificador de etiqueta se aplicaría el estilo a todas las etiquetas que pertenezcan a la clase "parrafoCentrado":

```
.parrafoCentrado { text-align: center; }
```

Ejemplo:

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset='UTF-8'>
    <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:bold; 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 tendrá el texto azul.</p>
    <p class="clase_azul"> Incluso el párrafo.</p>
    <p class="clase_roja">Sobre el párrafo podemos aplicar la clase clase_roja y el texto será rojo, en negrita cursiva y la familia del texto courier.</p>
    <h3 class="clase_roja"> Pero este texto no aparecerá formateado ya que regla de la clase clase_roja solo actúa sobre párrafos.
  </body>
</html>
```

Ejemplo del uso de clases en hojas de estilo

Cualquier elemento sobre el que apliquemos la clase clase_azul tendrá el texto azul.

Incluso el párrafo.

Sobre el párrafo podemos aplicar la clase clase_roja y el texto será rojo, en negrita cursiva y la familia del texto courier.

Pero este texto no aparecerá formateado ya que regla de la clase clase_roja solo actúa sobre párrafos.

3.4.2. Selectores de ID

Permite seleccionar un elemento de la página por medio de su atributo ID. Se asocia a elementos de estilo que se van a aplicar de manera excepcional una única vez. El valor del atributo id no se debe repetir en dos elementos diferentes de la página.

3.4.3. Selectores descendientes

Asocia elementos que están dentro de otros elementos. Por ejemplo:

```
p h1 {color: red; }:
```

Se aplicará el estilo a todas las etiquetas h1 que estén dentro de bloques p. Se debe tener en cuenta que

- No tiene por qué ser descendiente directo.
- El nivel de anidación puede tener varios niveles:

```
p a b i {text-decoration: underline; }
```

En este ejemplo se aplica a elementos en cursiva que están dentro de etiquetas de negrita, anidados dentro de enlaces que se encuentren en párrafos.

Hay que tener en cuenta que en el ejemplo anterior no sería lo mismo si estuvieran los selectores separados por coma.

También podemos combinar el selector universal con selectores descendientes:

```
p * b {color: #0000FF;}
```

Se aplica a todas las etiquetas que estén anidadas en cualquier otra etiqueta que a su vez esté dentro de una etiqueta tipo <p>, pero no a las que están dentro de <p> directamente.

3.4.4. Selector hijo

Es parecido al anterior, pero solo afecta al primer nivel de anidamiento.

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset='UTF-8'>
    <title>Selector hijo</title>
    <style>
      section > p {color:red}
    </style>
  </head>
  <body>
    <p>Párrafo inicial</p>
    <section>
      <p>Párrafo hijo de sección</p>
      <article>
        <p>Párrafo nieto de sección</p>
      </article>
    </section>
  </body>
</html>
```

Párrafo inicial


Párrafo hijo de sección

Párrafo nieto de sección

3.5. Propiedades principales

3.5.1. Propiedades de color y fondo

No todos los nombres de colores son admitidos en el estándar, así que se aconseja utilizar RGB.

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: <code>repeat-x</code> , <code>repeat-y</code> o <code>no-repeat</code> .
background-attachment	Especifica si la imagen ha de permanecer fija o realizar un scroll. Los valores que pueden tomar son: <code>scroll</code> o <code>fixed</code> .
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 <code>[top, center, bottom]</code> <code>[left, center, right]</code>
background	Establece en un solo paso cualquiera de las propiedades de <code>background</code> anteriores. Los valores que puede tomar son: <code>background-color</code> , <code>background-image</code> , <code>background-repeat</code> , <code>background-attachment</code> , <code>background-position</code> .

```
<!DOCTYPE html>
<html>
  <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 párrafo que es blanco, es amarillo y el fondo negro.</p>
  </body>
</html>
```

Ejemplo del uso de atributos de color y fondo

El texto de cualquier elemento, salvo el del párrafo que es blanco, es amarillo y el fondo negro.

3.5.2. Propiedades de fuente

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: <code>normal</code> , <code>bold</code> , <code>bolder</code> , <code>lighter</code> , 100, 200, 300, 400, 500, 600, 700, 800 o 900
font-style	Determina si la fuente es normal o cursiva. El estilo <code>oblique</code> es similar al cursiva. Los valores posibles son: <code>normal</code> , <code>italic</code> , <code>oblique</code> .
font-variant	Determina si la fuente es normal o mayúsculas pequeñas. Los valores que puede tomar son: <code>normal</code> , <code>small-caps</code>
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: <code>font-style</code> , <code>font-variant</code> , <code>font-weight</code> , <code>font-size</code> [<code>line-height</code>], <code>font-family</code> . Los valores han de estar separados por espacios. No es obligatorio el uso de todos los valores.

```
<!DOCTYPE html>
<html>
  <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 amarillo, salvo el del párrafo que es Verdana, blanco y de tamaño 12
  </body>
</html>
```

Ejemplo del uso de atributos de fuente

El texto de cualquier elemento es de la familia Courier y amarillo, salvo el del párrafo que es Verdana, blanco y de tamaño 12 px.

3.5.2. Propiedades de texto

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

```
<!DOCTYPE html>
<html>
  <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 está subrayado, centrado y la primera letra de cada palabra es mayúscula.
    <p>El párrafo está tabulado</p>
  </body>
</html>
```



3.5.3. Propiedades de lista

Elemento	Descripción
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.

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset='UTF-8'>
    <title>Estilo para listas</title>
    <style>
      #flecha{ list-style-image: url("flecha.png") }
      .circ{ list-style-type: circle }
      #armenio{ list-style-type: armenian }
    </style>
  </head>
  <body>
    <p>Lista con imagen</p>
```

```

<ul id="flecha">
  <li>Patatas</li>
  <li>Peras</li>
</ul>
<p>Lista con círculo</p>
<ul class="circ">
  <li>Patatas</li>
  <li>Peras</li>
</ul>
<p>Alfabeto armenio</p>
<ol id="armenio" reversed>
  <li>Peras</li>
  <li>Manzanas</li>
</ol>
</body>
</html>

```

Lista con imagen



Lista con círculo

- Patatas
- Peras

Alfabeto armenio

Ք. Peras
Ա. Manzanas

3.5.5. Propiedad display

Se usa para:

- Hacer que un elemento sea de bloque o de línea.
- Ocultarlo o hacerlo visible (a través de javascript).

En el siguiente ejemplo vemos varios vínculos (elementos en línea). Los que tienen la clase "especial" están definidos como elementos de bloque:

```

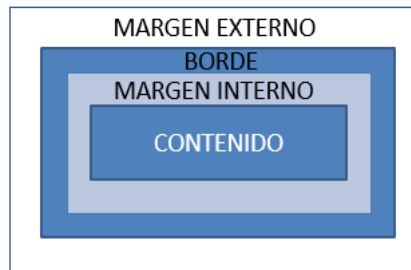
<!DOCTYPE html>
<html>
  <head>
    <title>Ejemplo de atributos CSS de fuente</title>
    <style>
      .especial { display: block}
    </style>
  </head>
  <body>
    <a href="#">Primer vínculo (normal)</a>
    <a href="#">Segundo vínculo (normal)</a>
    <a href="#" class="especial">Tercer vínculo (especial)</a>
    <a href="#" class="especial">Cuarto vínculo (especial)</a>
  </body>
</html>

```

[Primer vínculo \(normal\)](#) [Segundo vínculo \(normal\)](#)
[Tercer vínculo \(especial\)](#)
[Cuarto vínculo \(especial\)](#)

3.6. Avanzado: Modejlo de cajas

La W3C define lo que llaman "modelo de caja", que es una zona rectangular que rodea a cada uno de los elementos de la web:



- **Contenido:** contenido de la etiqueta.
- **Margen interior.** Distancia desde el contenido hasta el borde del elemento (**padding**).
- **Borde.** El borde del elemento (**border**).
- **Margen exterior.** Distancia desde el borde del elemento a los elementos adyacentes (**margin**).

3.6.1. Propiedades de caja

Contenido

No se tiene en cuenta el borde y los márgenes.

FORMATO	PROPIEDAD	VALORES
anchura	width	auto Longitud en px o equivalente
altura	height	auto Longitud en px o equivalente

Margen interno

- Utilizando padding solamente se aplica a los cuatro lados.
- Con dos valores, el primero aplica al superior y al inferior y el segundo a los laterales.
- Con tres valores se aplica el primero al superior, el segundo a los laterales y el tercero al inferior.
- Con 4 valores se aplica en orden: superior, derecho, inferior, izquierdo.

FORMATO	PROPIEDAD	VALORES
anchura	width	auto Longitud en px o equivalente
altura	height	auto Longitud en px o equivalente

Borde

- Si utilizamos border-color, border-width o border-style se aplicará a los cuatro lados.
- Con dos valores se aplica el primero al superior y al inferior y el segundo a los laterales.
- Con tres valores se aplica el primero al superior, el segundo a los laterales y el tercero al inferior.
- Con cuatro valores se aplica en orden: superior, derecha, inferior, izquierda.

FORMATO	PROPIEDAD	VALORES
Color del borde	border-color border-top-color border-bottom-color border-right-color border-left-color	Color en alguna de las notaciones permitidas transparent
Grueso del borde	border-width border-top-width border-bottom-width border-right-width border-left-width	Valor de longitud thin medium thick
Estilo del borde	border-style border-top-style border-bottom-style border-right-style border-left-style	Solid dashed dotted double ridge Groove inset outset hidden none

Margen externo

Mismo funcionamiento que el padding para declarar valores.

FORMATO	PROPIEDAD	VALORES
Ancho del margen externo	margin margin-top margin-bottom margin-right margin-left	auto valor de longitud valor de porcentaje

3.6.2. Unidades de tamaño

CSS permite utilizar diferentes tipos de unidades, que pueden ser:

- **Absolutas:** cualquier longitud expresada en estas unidades muestra el mismo tamaño.

Unidades absolutas	
cm	Centímetros
mm	Milímetros
in	Pulgadas
px *	Píxeles
pt	Puntos
pc	Picas

- **Relativas:** dependen del tamaño de otro elemento.

UNIDADES RELATIVAS	
em	Relativa al tamaño del tipo de letra por defecto
porcentajes (%)	Relativos a las dimensiones del elemento contenedor
ex	Relativa al valor de x-height de la fuente actual
ch	Relativa al ancho del cero "0"
rem	Relativa al tamaño de letra del elemento raíz

Para imágenes y vídeos se suele utilizar píxeles. Es la única unidad absoluta usada habitualmente.

Ejemplo de porcentajes 1:

La primera sección tiene fondo verde y su anchura es la mitad del elemento contenedor <body> la cual ocupa toda la anchura disponible, la segunda sección ocupa el 30%

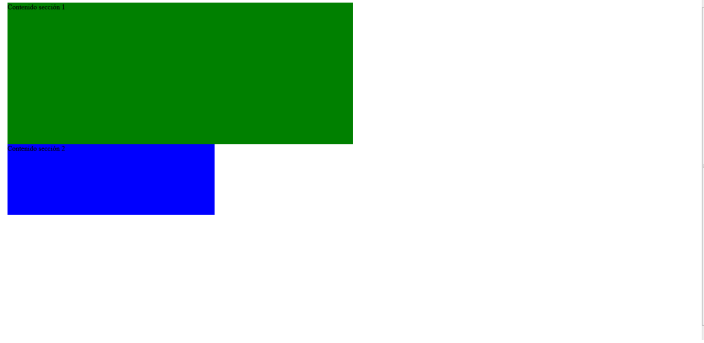
```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="UTF-8">
    <title>Porcentajes</title>
    <style>
      #s1{
        background-color:green;
        width:50%;
      }
      #s2{
        background-color:blue;
        width:30%;
      }
    </style>
  </head>
  <body>
    <section id = "s1">Contenido sección 1</section>
    <section id = "s2">Contenido sección 2</section>
  </body>
</html>
```



Ejemplo de porcentajes 2:

Con el alto ocurre lo mismo, pero depende del contenido de la página, a no ser que se fije un valor para el elemento como en este ejemplo. <body> tiene una altura fijada en píxeles y el tamaño de las secciones tienen porcentajes relativos al <body>

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="UTF-8">
    <title>Porcentajes</title>
    <style>
      body{
        height:800px;
      }
      #s1{
        background-color:green;
        width:50%;
        height:40%;
      }
      #s2{
        background-color:blue;
        width:30%;
        height:20%;
      }
    </style>
  </head>
  <body>
    <section id = "s1">Contenido sección 1</section>
    <section id = "s2">Contenido sección 2</section>
  </body>
</html>
```



Ejemplo en unidades em

Para las fuentes es habitual utilizar la unidad em, que hace referencia al tamaño de la fuente actual.

```

<!DOCTYPE html>
<html>
  <head>
    <meta charset="UTF-8">
    <title>Unidades em</title>
    <style>
      .grande {font-size: 2em;}
      .muyGrande {font-size: 4em}
    </style>
  </head>
  <body>
    <p>Normal</p>
    <p class = "grande">Grande</p>
    <p class = "muyGrande">Muy grande</p>
  </body>
</html>

```

Normal

Grande

Muy grande

3.6.3. Ejemplos del modelo cajas

Ejemplo 1:

```

<!DOCTYPE html>
<html>
  <head>
    <meta charset="UTF-8">
    <title>Relleno y margen</title>
    <style>
      #relleno{
        background-color:yellow;
        padding: 2em; /*se aplica a izquierda, derecha arriba y abajo*/
      }
      #rellenoIzq{
        background-color:green;
        padding-left: 2em; /*se aplica a izquierda, derecha arriba y abajo*/
      }
      #rellenoMargen{
        background-color:blue;
        margin-bottom: 2em; /*se aplica a izquierda, derecha arriba y abajo*/
      }
    </style>
  </head>
  <body>
    <section id = "relleno">Sección con relleno (margen interior)</section>
    <section>Sección sin relleno</section>
    <section id = "rellenoMargen">Sección sin relleno, pero con margen inferior</section>
  </body>
</html>

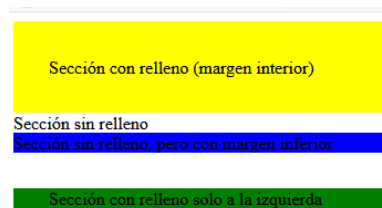
```



```

<section id = "rellenoIzq">Sección con relleno solo a la izquierda</section>
</body>
</html>

```



Ejemplo 2:

```

<html>
  <head>
    <meta charset="UTF-8">
    <title>Relleno y margen</title>
    <style>
      #borde1{
        background-color:yellow;
        border-color: green;
        border-style: solid;
      }
      #borde2{
        background-color:pink;
        border-width: 16px;
        border-left-color: green;
        border-top-color: green;
        border-left-style: dashed;
        border-top-style: dashed;
      }
    </style>
  </head>
  <body>
    <section id = "borde1">Sección con borde1</section>
    <br>
    <section id = "borde2">Sección con borde2</section>
  </body>
</html>

```



3.6.4. Posicionamiento

Con CSS es posible modificar el posicionamiento por defecto de los elementos, con las propiedades `position` y `float`.

Con la propiedad `float`, los elementos flotan hacia la izquierda o la derecha, todos los elementos flotados se van situando uno junto a otro y si no hay espacio pasan a una nueva línea.

En este ejemplo se utiliza la propiedad `float` para maqueta una página sencilla sobre los elementos `<nav>` y `<section>`.

```

<!DOCTYPE html>
<html>
  <head>
    <meta charset="UTF-8">
    <title>Etiquetas semánticas y float</title>
    <style>
      body{
        background-color:pink;
      }
      header{ background-color:blue;}
      nav {
        background-color:red;
        width:10%;
      }
      section {

```

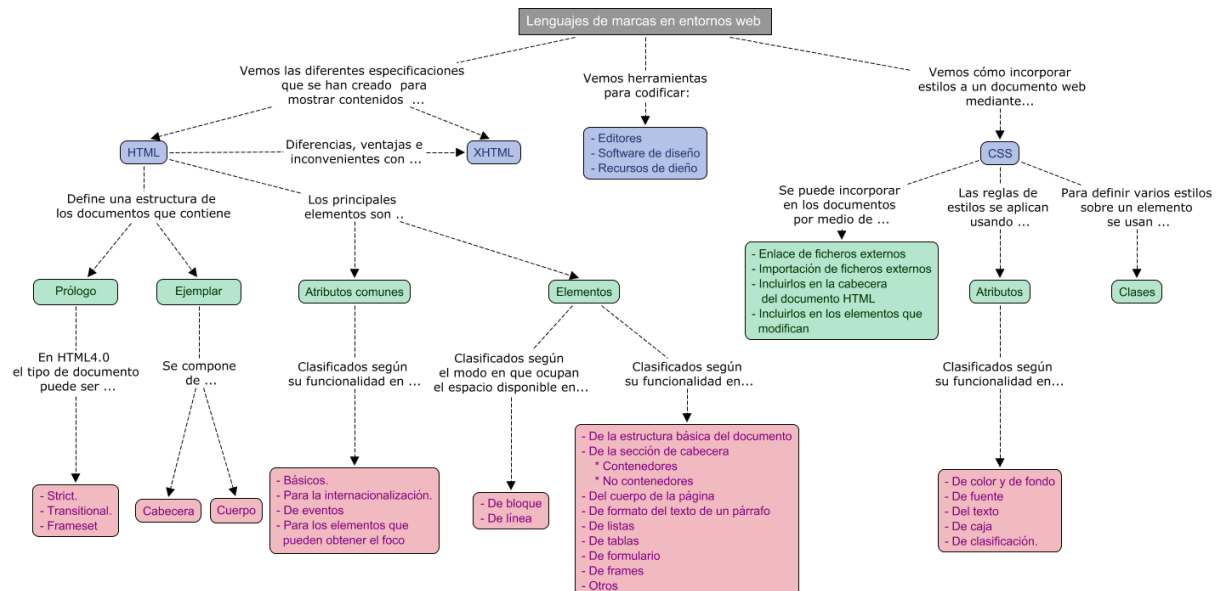
```

        background-color:green;
        width:90%;
    }
    footer {
        background-color:yellow;
    }
    nav, section {
        height:500px;
        float: left;
    }
}
</style>
</head>
<body>
    <header>Encabezado</header>
    <nav>Vinculos</nav>
    <section>Contenido principal del página</section>
    <footer>Página creada por...</footer>
</body>
</html>

```



Mapa Conceptual





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

Autor	Xerach Casanova
Clase	Lenguajes de Marcas
Fecha	@Mar 29, 2021 9:11 PM

1. Sindicación de contenidos

1.1. Características

Publicación en la web

Sindicación

1.1. Sistemas de gestión de contenidos (CMS)

1.2. Ventajas de la sindicación de contenidos

2. Ámbito de aplicación

3. Tecnologías de creación de canales de contenidos

4. Estructura de los canales de contenidos

4.1. RSS

5. Validación

6. Agregadores

Mapa conceptual

1. Sindicación de contenidos

La redifusión o sindicación de contenidos permite a un sitio utilizar los servicios o contenidos ofertados por otro sitio distinto.

La redifusión web consiste en ofrecer contenido desde una fuente web, cuyo origen esté en otra, proporcionando actualización del mismo.

Los servicios que ofrece la web original junto con los metadatos que tiene asociado, forman los feed o canales de contenidos.

Para leer una fuente o canal hay que suscribirse a ella utilizando un agregador. Esta redifusión se suele realizar bajo licencia de normas de uso o contrato que regule derechos de los contenidos.

Estas fuentes se suelen codificar en XML, pero se puede realizar en cualquier lenguaje que pueda transportar mediante el protocolo HTTP (HyperText Transfer Protocol).

1.1. Características

Publicación en la web

Puede ser visto como un flujo de información que va desde el origen hasta los usuarios que lo leen a través de su navegador, accediendo a una web.

Sindicación

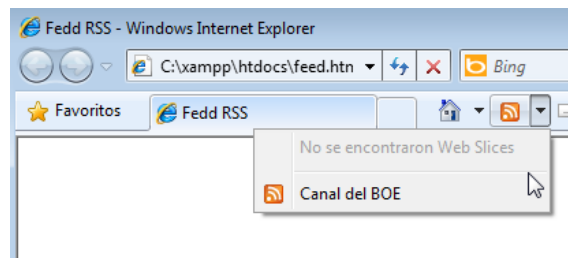
Una web puede ser suministradora de un canal de información (información sindicada), para lograrlo en la cabecera se debe incluir debajo del elemento <title> un enlace al canal de contenidos. Dependiendo de si el canal está hecho con un estándar RSS o con un Atom se utilizan las siguientes líneas:

```
<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" />
```

Al vincular un canal, el resultado puede ser poco claro y cambia entre navegadores, por ejemplo:

```
<!DOCTYPE html>
<html>
  <head>
    <title>Fedd RSS</title>
    <link rel="alternate" type="application/rss+xml" title="Canal del BOE" href="https://www.boe.es/rss/boe.php" />
    <meta charset="UTF-8">
  </head>
  <body>
  </body>
</html>
```

El resultado en explorer es:



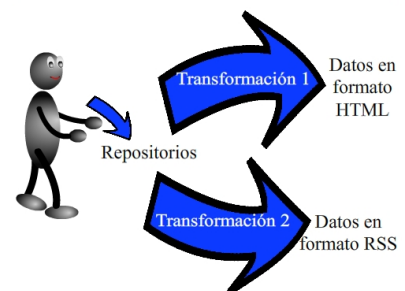
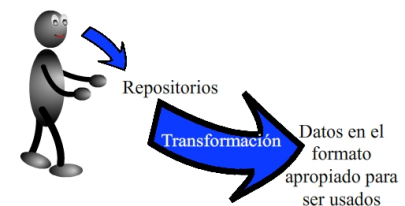
Pero para que el resultado sea más claro, se puede utilizar un vínculo normal, para el que se utiliza normalmente el símbolo de RSS.

1.1. Sistemas de gestión de contenidos (CMS)

Un sistema gestor de contenidos es un programa que permite crear estructuras de soporte para la creación y administración de contenidos en web por parte de los participantes. Disponen de una interfaz que controla una o varias bases de datos donde se aloja el sitio.

El origen de los contenidos es un repositorio y antes de ser servidos al cliente en formato adecuado, se transforman:

- Documento XML → transformación XSLT (lenguaje extensible de transformaciones de hojas de estilo) → documento XHTML.
- Bases de datos → script en Perl → documento HRML
- Texto plano → Página de servidor activo - documento HTML



- Mente del autor → Bloc de notas → documento HTML

Con un CMS de cualquier tipo la transformación puede replicarse. Podemos tener más de una entrada de información y varias salidas (podemos tener tantos ficheros HTML como canales RSS).

1.2. Ventajas de la sindicación de contenidos

- Aumentar el tráfico de la web
- Ayudar a los usuarios a visitarlo frecuentemente
- Favorecer posicionamiento en buscadores
- Ayudar a establecer relaciones entre distintas web dentro de la comunidad

- Permite a otras personas añadir características a los servicios del sitio web (notificaciones de actualizaciones mediante mensajes...)
- Enriquecer internet impulsando la tecnología semántica y fomentar la reutilización.

2. Ámbito de aplicación

La redifusión de contenidos web se puede aplicar a todo tipo de contenidos: texto, audio, videos e imágenes. Se pueden ofrecer contenidos propios en otras webs de forma integrada.

Desde el punto de vista del subscritor, la redifusión permite la actualización profesional. Los usuarios pueden estar al día en temas relacionados con su profesión de sus sitios relevantes.

3. Tecnologías de creación de canales de contenidos

Los estándares más utilizados son:

RSS (Really Simple Syndication), parte de la familia de formatos XML. Se utiliza para:

- Conectar con sistemas de mensajería instantánea.
- Conversión RSS en mensajes de correo electrónico.
- Transformar los enlaces favoritos del navegador en RSS.

Tiene 7 formatos diferentes entre sí:

- RSS 0.90: lo creó Netscape en 1999, se basa en especificación RDF de metadatos, con la intención de que su proyecto My Netscape estuviese formado por titulares de otras webs
- RSS 0.91: versión simplificada lanzada posteriormente. Se detuvo por falta de éxito, pero UserLand Software usó esta versión para desarrollar blogs.
- RSS 1.0: creado a partir del estándar 0.90, más estable y permite definir mayor cantidad de datos que el resto de versiones RSS.
- RSS 2.0: UserLand Software rechazó 1.0 por su complejidad y desarrolló 0.92, 0.93 y 0.94. Todas ellas incompletas y no cumplen todas las normas XML. 2.0 subsana esos errores.

Atom: Fue publicado como un estándar por el grupo de trabajo Atom Publishing Format and Protocol de la IETF en el RFC4287. Se desarrolla como alternativa RSS para evitar confusión por todos los estándares similares, los cuales tenían cierta incompatibilidad entre ellos. Este estándar se desarrolla para convivir con ellos. Es flexible y permite tener mayor control sobre la cantidad de información a representar en los agregadores.

4. Estructura de los canales de contenidos

Para construir un canal de contenido es necesario crear un fichero con extensión rss o atom basado en XML, este se publicará en uno de los directorios de la web en la que se oferta. El fichero consta de:

- Declaración del documento XML y definición de la codificación empleada en el documento, preferiblemente UTF-8.
- Un canal en el que se determina el sitio web asociado a la fuente web a la que se hace referencia en el fichero. Además de su definición está formado por:
 - Secciones: cada una de las cuales es referencia a la web que contiene uno de los servicios que se ofrece. Se pueden incluir tantas como se quieran.

No existen restricciones respecto a la cantidad de canales de contenidos a ofrecer desde una web.

4.1. RSS

El documento RSS incluye como primera línea la declaración del documento XML:

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

A continuación aparece la etiqueta <rss> que indica que es un documento rss y la versión empleada. Dentro de ella aparece la etiqueta <channel> encargada de describir el feed RSS, con tres elementos hijos obligatorios:

- <title> título del canal
- <link> hiperenlace al canal
- <description> descripción del canal

Y elementos opcionales, tales como:

- <language>
- <category>
- <copyright>

Los canales tienen uno o más artículos (etiqueta <item>). Cada uno cuenta una historia del canal. Tiene 3 hijos obligatorios:

- <title> título del canal
- <link> hiperenlace al canal
- <description> descripción del canal

y elementos opcionales, tales como:

- <author>
- <category>
- <guid> Define un identificador único para el elemento

```
<?xml version="1.0" encoding="UTF-8"?>
<rss>
  <channel>
    <title></title>
    <link></link>
    <description></description>
    <item>
      <title></title>
      <link></link>
      <description></description>
    </item>
  </channel>
</rss>
```

5. Validación

Para validar un documento RSS se le da la dirección del fichero donde se encuentra alojado y comprueba que lo pueden encontrar (URL válida) y que no contiene errores. Una vez validado se obtiene una imagen del tipo XML o RSS de color naranja que se puede incluir en la página principal para enlazar a la dirección del fichero.

Algunos de estos validadores son:

- [FeedValidator](#).
- [W3C Feed Validation Service mediante URL](#).
- [W3C Feed Validation Service mediante código](#).
- [RSS Advisory Board](#).
- [Googletransitdatafeed](#).

6. Agregadores

Agregador o lector de fuentes es una aplicación de software para suscribirse a fuentes en formatos RSS y Atom. Estos agregadores avisan de qué páginas incorporan nuevo contenido desde la última lectura.

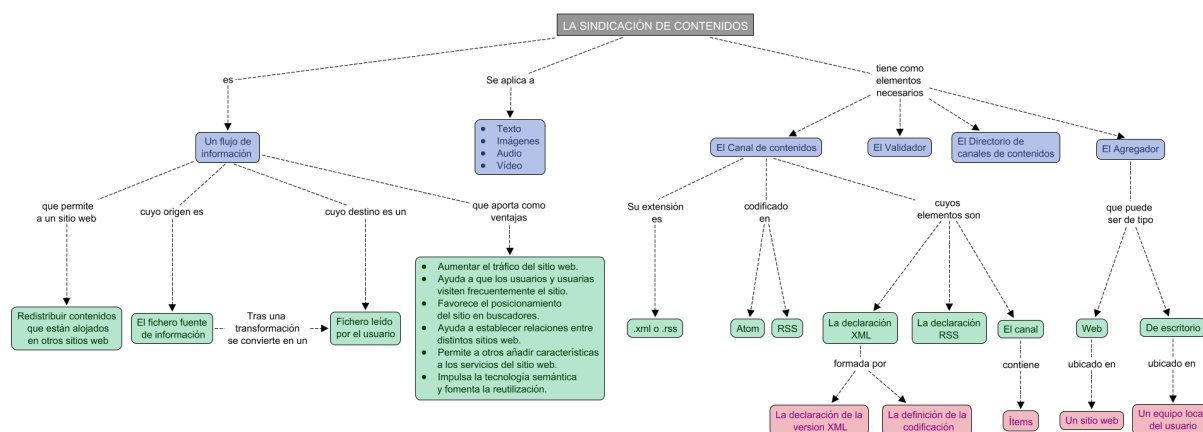


En el agregador se debe indicar la dirección a cada archivo fuente RSS o Atom.

Existen varios tipos:

- **Agregadores Web o agregadores en línea.** Aplicaciones que residen en determinados sitios web y que se ejecutan a través de la propia web. Recomendables cuando el usuario no accede desde el mismo ordenador: **Feedly, Inoreader o NewsBlur**
- **Agregadores de escritorio.** Aplicaciones instaladas en el ordenador. Interfaz gráfica parecida a clientes de correo electrónico, con panel donde se agrupan suscripciones y otro donde se accede a las entradas individuales para su lectura: **RSSOwl, QuiteRSS.**
- **Complementos de navegador,** disponibles como complementos de navegador web: **Awesome RSS o LiveMarks**

Mapa conceptual





4. Definición de esquemas y vocabularios en xml.

Autor	Xerach Casanova
Clase	Lenguajes de Marcas
Fecha	@Feb 3, 2021 7:48 PM

1. Documento XML. Estructura y sintaxis
 2. Definición de tipo de documento, DTD.
 - 2.1. Declaraciones de tipos de elementos terminales
 - 2.2. Declaraciones de tipos de elementos no terminales
 - 2.3. Declaraciones de listas de atributos para los tipos de elementos
 - 2.4. Declaraciones de entidades
 - 2.5. Declaraciones de notación
 - 2.6. Secciones condicionales
 3. XML Schema
 - 3.1. Tipos de datos
 - 3.1.1. Facetas de los tipos de datos
 - 3.1.2. Facetas: ejercicios.
 - 3.2. Elementos del lenguaje.
 - 3.3. Definición de tipos de datos XML Schema
 - 3.3.1. Definición de tipos de datos: ejercicios
 - 3.2. Asociación con documentos XML
 - 3.5. Documentación del esquema
 4. Herramientas de creación y validación
- Mapa conceptual

1. Documento XML. Estructura y sintaxis

Los documentos XML están formado por:

- **Prólogo:** informa al intérprete encargado de procesar el documento de todos aquellos datos que necesita para realizar el trabajo.
 - **Definición de XML.** se indica la versión de XML, el código de los datos a procesar y la autonomía del documento.
 - **Declaración del tipo de documento.** Hasta ahora solo hemos usado `<!DOCTYPE` y separado por al menos un espacio.
- **Ejemplar:** Contiene los datos del documento a procesar. Es el elemento raíz del documento y ha de ser único. Contiene elementos estructurados en árbol. La raíz es el ejemplar y las hojas los elementos terminales, que pueden a su vez estar formados por atributos.

Pero además de lo anterior, se deben definir las cualidades que tiene ese ejemplar. Para ello utilizaremos DTD's o XML Schemas.

2. Definición de tipo de documento, DTD.

Están formadas por una relación precisa de qué elementos pueden aparecer en un documento y dónde, así como el contenido y atributos del mismo. Garantizan que los datos del XML cumplen las restricciones impuestas en la DTD:

- Especificar la estructura del documento.

- Reflejar una restricción de integridad referencial mínima utilizando (ED e IDREF).
- Utilizar pequeños mecanismos de abstracción comparables a las macros, que son las entidades.
- Incluir documentos externos.

Los principales inconvenientes de los DTD 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 se permiten secuencias ordenadas.
- No es posible formar claves a partir de varios atributos o elementos.
- Una vez definido el DTD no se puede añadir nuevo vocabulario.

Cuando se definen dentro del documento XML se ubican entre corchetes después del nombre del ejemplar en el elemento `<!DOCTYPE>`, pero cuando se define en fichero externo, se hace en un fichero de texto plano con extensión `dtd`.

2.1. Declaraciones de tipos de elementos terminales

Los tipos terminales son elementos que se corresponden con hojas de la estructura de árbol formado por los datos del documento XML asociado al DTD.

La declaración de tipos de elementos se hace con la cadena `<!ELEMENT>` separada por al menos un espacio del nombre del elemento XML que se declara, seguidamente irá la declaración del contenido que puede tener el elemento.

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

- `EMPTY`. El elemento no es contenedor.
- `ANY`: Permite que el contenido del elemento sea cualquier cosa.
- `(#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: `< &]] >`,

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

Ejemplo.

En la siguiente estructura

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

un DTD puede ajustarse a:

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

2.2. Declaraciones de tipos de elementos no terminales

Definen las ramas de un documento, es decir, elementos que están formados por otros elementos.

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

El elemento A está formado por el elemento B, seguido de un elemento C.

Los siguientes operadores nos permiten definir la cardinalidad de un elemento.

- **Operador ?**: indica que el elemento no es obligatorio.

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

- **Operador uno-o-más, +**. El componente estará presente al menos una vez.

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

- **Operador cero-o-más, ***. Define un componente presente, cero, una o más veces.

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

- **Operador de elección |**. Si se utiliza sustituyendo las comas en la declaración de grupos indica que para formar el documento XML debes elegir entre elementos separados por ese operador.

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

Para la siguiente estructura:

```
<alumno>
<nombre>Olga</nombre>
<dirección>El Percebe 13</dirección>
</alumno>
```

Un DTD puede ser:

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

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

Para declarar los atributos asociados a un elemento se utiliza la cadena `<!ATTLIST` seguida del nombre del elemento asociado al atributo y luego el nombre del atributo, seguido del tipo 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 asociarlo a dicho elemento de manera individual.

No todos los atributos son del mismo tipo, los más destacados son:

- **Enumeración**. El atributo solo toma uno de los valores determinados dentro de un paréntesis separados por |

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

- **CDATA** se utiliza cuando un atributo es una cadena de texto.
- **ID** permite declarar un atributo identificador de un elemento. El valor ha de ser único y además se deben tener en cuenta que los números no son nombres válidos en XML.
- **IDREF** permite hacer referencia a identificadores. En este caso el valor del atributo ha de corresponder con el identificador de un elemento existente.
- **NMTOKEN** permite declarar que el valor de un atributo ha de ser una sola palabra compuesta por los caracteres permitidos en XML.

También debemos indicar si un atributo es obligatorio o no con los siguientes modificadores:

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

Ejemplo. para la siguiente estructura XML

```
<alumno edad=15>
  <nombre>Olga</nombre>
  <apellidos>Velarde Cobo</apellidos>
  <dirección>El Percebe 13</dirección>
</alumno>
```

Un DTD válido podría ser:

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

2.4. Declaraciones de entidades

Si queremos declarar valores constantes dentro de documentos debemos usar entidades. Se limitan por & y ;, por ejemplo &entidad;

El intérprete sustituye la entidad por el valor asociado en el DTD. No se admite recursividad, por tanto una entidad no puede hacer referencia a ella misma. Se definen con el elemento `<!ENTITY>` y pueden ser de cuatro tipos:

- **Internas.** Existen cinco entidades predefinidas en el lenguaje que son:
 - **<**: corresponde al signo menor que, <
 - **>**: corresponde al signo mayor que, >
 - **"**: corresponde a las comillas rectas dobles, ".
 - **'**: corresponde a comilla simple, '.
 - **&**: corresponde al ampersand, &.

Para definir una entidad diferente podemos usar la siguiente sintaxis:

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

```
<!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.

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

El contenido de los ficheros es analizado y debe seguir sintaxis XML. Cuando se incluyen ficheros de formato binario, se utiliza la palabra reservada `NDATA` en la definición de la entidad y se asocia a dicha entidad una declaración de notación.

- **De parámetro:** da nombres a partes de un DTD y hace referencias a ellas a lo largo del mismo. 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.

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

2.5. Declaraciones de notación

Cuando se incluyen ficheros binarios en un fichero debemos usar la siguiente sintaxis:

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

Por ejemplo, una notación llamada gif donde se indica que se hace referencia a un editor de formatos gif:

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

Para asociar una entidad externa no analizada a esa notación:

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

2.6. Secciones condicionales

Permiten incluir o ignorar partes de la declaración de un DTD para ello se usa:

- INCLUDE, permite que se vea parte de la declaración del DTD

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

Ejemplo:

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

- IGNORE, permite ocultar esa sección de declaraciones del DTD

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

Por ejemplo

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

3. XML Schema

Para hacer que los elementos y atributos de los ficheros XML correspondan a datos de un tipo determinado y que cumplan ciertas restricciones se usan los XML Schemas.

Se definen en ficheros planos y son documentos XML con extensión 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 pertenecer a un documento XML asociado válido.

Los elementos hijos inmediatos de este ejemplar son `<xs:element>`, que nos permiten crear globalmente un documento. Esto significa que el elemento creado puede ser el ejemplar del documento XML asociado.

Ejemplo de esquema correspondiente a un documento XML

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

Un XML Schema sería:

```
< ?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>
```

3.1. Tipos de datos

Son los distintos valores que puede tomar el atributo `type` cuando se declara un elemento o atributo.

Algunos de estos valores son:

- **string** caracteres UNICODE
- **boolean**
- **integer**
- **positiveinteger**
- **negativeinteger**
- **decimal** números decimales
- **datetime** fecha y hora absolutas.
- **duration** duración de tiempo expresado en años, meses, días horas, minutos y segundos. El formato utilizado es: PnYnMnDTnHnMnS. Para representar una duración de 2 años, 4 meses, 3 días 5 horas, 6 minutos y 10 segundos se pondría P2Y4M3DT5H6M7S. Se pueden omitir valores nulos, para indicar duración negativa se pone - delante de la P.
- **time** hora en formato hh:mm:ss
- **date** fecha en formato CCYY-MM-DD
- **gYearMonth** mes de un año determinado en formato CCYY-MM
- **gYear** año en formato CCYY
- **gMonthDay** día de un mes en formato -MM-DD.
- **gDay** ordinal del día del mes en formato —DD, el 4º día será -04
- **gMonth** mes en formato -MM
- **anyURI** representa una URI
- **language** representa identificadores de lenguaje, valores definidos en RFC 1766
- **ID, IDREF, ENTITY, NOTATION, MTOKEN** representan lo mismo que en los DTD.

3.1.1. Facetas de los tipos de datos

Las restricciones que se pueden aplicar sobre los valores de los datos de elementos o atributos se definen por las facetas y se aplican sobre tipos simples, utilizando el elemento `xs:restriction`.

Se expresan como un elemento dentro de una restricción y se combinan para lograr restringir más el valor del elemento:

- **length, minlength, maxlength:** longitud del tipo de dato.
- **enumeration:** restringe a un determinado conjunto de valores
- **whitespace:** define el tratamiento de espacios (preserve/replace, collapse).
- **(max/min) (Inclusive/Exclusive):** límites superiores e inferiores de tipos de datos. Cuando son Inclusive, el valor que se determine es parte del conjunto de valores válidos.
- **totalDigits, fractionDigits:** número de dígitos totales y decimales de un número decimal.
- **pattern:** permite construir máscaras que han de cumplir los datos de un elemento:

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

3.1.2. Facetas: ejercicios.

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>
```

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>
```

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>
```

Creación de la máscara de un DNI mediante pattern.

```
<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>
```

3.2. Elementos del lenguaje.

- **Esquema:** `xs:schema` contiene la definición del esquema.
- **Tipos completos:** `xs:complexType` define tipos complejos.
- **Tipos simples:** `xs:simpleType` define un tipo simple restringiendo sus valores.
- **Restricciones:** `xs:restriction` establece restricciones sobre un elemento de tipo base
- **Agrupaciones:** `xs:group` nombra agrupaciones de elementos y atributos para hacer referencia a ellas.
- **Secuencias:** `xs:sequence` construye elementos complejos mediante la enumeración de los que les forman.
- **Alternativa:** `xs:choice` representa alternativas, teniendo 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.

```
<?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"/>
  <!-- Definición del tipo datosAlum -->
  <xs:complexType name="datosAlum">
    <xs:sequence>
      <xs:element name="alumno" type="datos" minOccurs="1" maxOccurs="unbounded"/>
    </xs:sequence>
  </xs:complexType>

  <!-- Definición 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:sequence>
  </xs:complexType>
</xs:schema>
```

```

        <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>

```

3.3. Definición de tipos de datos XML Schema

En los XML Schema se diferencian también los elementos terminales y los elementos no terminales:

- **Tipos de datos simples.** Se suelen definir para hacer una restricción sobre un tipo de datos XSD ya definido y establece el rango de valores que se pueden tomar. Se pueden crear también tipos de datos simples basados en listas de valores utilizando derivedBy de simpleType.
- **Tipos de datos compuestos.** El elemento xsd:complexType permite definir estructuras complejas de datos. Su contenido son las declaraciones de elementos y a tributos, o referencias a elementos y atributos declarados de forma global.

3.3.1. Definición de tipos de datos: ejercicios

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:element name="edad">
  <xs:simpleType>
    <xs:restriction base="xs:positiveInteger">
      <xs:minInclusive value="12"/>
      <xs:maxInclusive value="18"/>
    </xs:restriction>
  </xs:simpleType>
</xs:element>

```

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>

```

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

```

<xs:complexType name="alumno">
  <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="web" type="xs:string" minOccurs="0" maxOccurs="5">
      <xs:complexType>
        <xs:attribute name="href" type="xs:string"/>
      </xs:complexType>
    </xs:element>
  </xs:sequence>
</xs:complexType>

```

3.2. Asociación con documentos XML

Una vez creado el fichero xsd a un documento XML es un espacio de nombres al ejemplar del documentos, donde se indica la localización de los ficheros esquema mediante su URI, precedida del prefijo xsi

```
<?xml version="1.0" encoding="ISO-8859-1"? >
<alumnos xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:SchemaLocation="file:/D:/ubicación/del archivo/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>
```

3.5. Documentación del esquema

Para incorporar el autor, limitación de derechos de autor, utilidad del esquema etc, se puede realizar con el elemento xs:annotation, y permite guardar información adicional, a su vez se puede utilizar:

- xs:documentation, puede contener elementos XML bien estructurados.
- xs:appinfo, se diferencia poco de la anterior, este guarda información para los programas de software. También se usa para genera ayuda contextual de cada elemento del esquema.

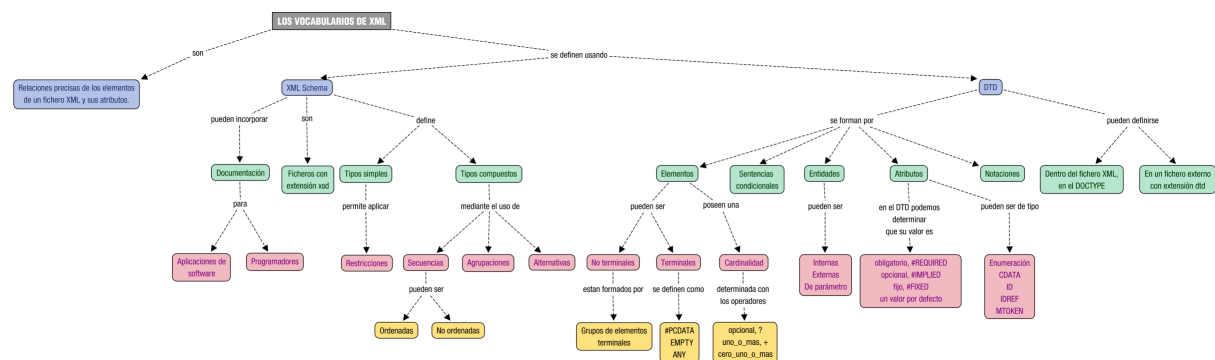
```
<xs:schema xmlns:xsi="http://www.w3.org/2001/XMLSchema">
  <xs:annotation>
    <xs:documentation xml:lang="es-es">
      Materiales para formación e-Learning
      <modulo>Lenguajes de marcas y sistemas de gestión de información.</modulo>
      <fecha_creación> 2011</fecha_creacion>
      <autor> 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>
```

4. Herramientas de creación y validación

- [Editix XML Editor](#) (Versiones open source y comercial).

- Microsoft Core XML Services (MSXML) (Gratuito).
- XMLFox (freeware).
- Altova XML Spy Edición Estándar (comercial).
- Editor XML xmlBlueprint. (comercial)
- Stylus Studio 2001 (comercial).
- Oxygen XML Editor (comercial).
- Exchanger XML Editor (comercial)
- XML copy editor (open source).

Mapa conceptual





5. Conversión y adaptación de documentos XML

Autor	ⓧ Xerach Casanova
Clase	Lenguajes de Marcas
Fecha	@Feb 23, 2021 7:02 AM

- 1. [Introducción](#)
- 2. [Estructura básica de una hoja XSLT.](#)
- 3. [Elementos XSLT](#)
- 4. [XPath](#)
 - 4.1. [Términos básicos](#)
 - 4.2. [Expresiones](#)
 - 4.3. [Procesando expresiones](#)
 - 4.4. [Ruta de localización](#)
 - 4.5. [Funciones del XPath](#)
 - 4.6. [Predicados](#)
 - 4.7. [Ejercicio resuelto](#)
 - 4.8. [Acceso a datos desde otro documento XML](#)
- 5. [Utilización de plantillas](#)
 - 5.1. [Ejercicio resuelto de plantillas](#)
- 6. [Procesadores XSLT](#)
- 7. [Depuradores XSLT](#)
- 8. [Para saber más](#)
- [Mapa conceptual](#)

1. Introducción

Si queremos usar directamente los datos de un documento XML es necesario transformarlo primero.

Los navegadores interpretan las etiquetas del documento XML aplicando un formato especificado en las hojas CSS. Es posible transformar un documento

XML en otro tipo de documento. Algunas de las tecnologías que lo permiten son:

- **XSLT**: permite definir el modo de transformar un documento XML en otro.
- **XSL-FO**: permite transformar un documento XML en otro documento de formato legible e imprimible, por ejemplo PDF.
- **Xpath**: permite acceder a diversos componentes de un XML.

A día de hoy se usa XSLT, estándar aprobado por W3C. y dichos documentos se llaman hojas XSLT.

XSLT deriva de XML, por tanto también son documentos XML, al igual que los documentos XSD, RSS o atom.

A partir de usar XSLT podemos generar otro documento XML, HTML o un documento de texto.

2. Estructura básica de una hoja XSLT.

- Elementos XSLT. Precedidos del prefijo `xsl:` pertenecen al espacio de nombres `xsl`, definidos en el estándar del lenguaje e interpretados por cualquier procesador XSLT.
- Elementos LRE, no pertenecen a XSLT, sino que se repiten en la salida sin más.
- Elementos de extensión, no pertenecen al espacio de nombres `xsl`, son manejados por implementaciones concretas del procesador. Normalmente no se usan.

Para asociar un XML a una hoja XSLT se usa la siguiente línea:

```
<?xml-stylesheet type="text/xsl" href="ruta_del_fichero_xsl.xsl"?>
```

3. Elementos XSLT

El elemento raíz es `xsl:stylesheet` o `xsl:transform`. Sus atributos principales son:

- **version**, 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 se les conoce como elementos de nivel superior, estructuras contenedoras de instrucciones. Si son hijos directos no se pueden anidar, excepto xsl: variable y xsl:param.

- **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 los resultados
- **xsl:decimal-format**, define un patrón que permite convertir en cadenas de texto números en coma flotante.
- **xsl:for-each**, se aplica 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 XSL en función de una condición.
- **xsl:import**, importa una hoja de estilos XSLT utilizada en una URI dada.
- **xsl:key**, define una o varias claves para ser referenciadas desde cualquier lugar del documento.
- **xsl:output**, define el tipo de salida que se genera 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: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.
- **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 más cómodamente.

4. XPath

Es un estándar, diferente de XML aprobado por el W3C, que permite navegar entre los elementos y atributos de un XML.

Para hacerlo se basa en relaciones de parentesco entre los nodos del documento.

En la actualidad se utiliza con XLT, XML Schema, Xquery, Xlink, Xpointer, Xforms, etc.

Expresiones de camino

Xpath se usa definiendo expresiones de camino para seleccionar nodos o conjuntos de no de un documento XML. las cuales se parecen mucho a las path de los sistemas de fichero.

Estas expresiones se aplican a un XML, asumiendo que su estructura interna es en árbol. Ejemplo:

```
<?xml version="1.0" encoding="UTF-8"?>
<matriculas>
  <alumno>
    <nombre>Pedro</nombre>
    <apellido1>López</apellido1>
    <apellido2>Ortega</apellido2>
    <DNI pais="es">López</DNI>
  </alumno>
</matriculas>
```

Al evaluar la expresión /matriculas/alumno/nombre podemos obtener los nombres de alumnos matriculados.

4.1. Términos básicos

Nodos

Un documento XML es tratado en Xpath como un árbol de nodos. Hay 7 tipos: elemento, atributo, texto, espacio de nombres, instrucción de proceso, comentario y documento. El elemento más alto en el árbol es el nodo raíz.

- **Nodo raíz.** Contiene el ejemplar del fichero XML, no hay que confundir con el elemento raíz del documento, el cual está debajo de él.
- **Nodo elemento,** cada uno de los elementos del documento XML.
 - Tienen un elemento padre
 - El padre del elemento raíz es el nodo raíz del documento.
 - Pueden tener identificadores únicos para referenciarlos de forma más directa. Es necesario que entonces el atributo esté definido en un DTD

o un XSD asociado.

- **Nodos texto.** Aquellos caracteres que no están marcados con ninguna etiqueta. No tienen hijos.
- **Nodos atributos,** son los atributos de un elemento.
 - Se consideran etiquetas añadidas al nodo elemento.
 - No se consideran hijos de ese elemento
 - Aquellos atributos con valor asignado en el esquema asociado se tratarán como si ese valor se le hubiese dado al escribir el XML.
 - Para las definiciones de espacios de nombre y para aquellos atributos definidos con la propiedad #IMPLIED en su DTD no se crean nodos.

```
<?xml version="1.0" encoding="UTF-8"?>
<matriculas>
  <alumno>
    <nombre>Pedro</nombre>
    <apellido1>López</apellido1>
    <apellido2>Ortega</apellido2>
    <DNI pais="es">López</DNI>
  </alumno>
</matriculas>
```

<matrículas> es el nodo elemento raíz.

<nombre>Pedro</nombre> es un nodo elemento.

pais="es" es un nodo atributo.

Ítems

Los ítems pueden ser nodos o valores atómicos

En el caso ejemplo, valores atómicos son "Pedro" o "es"

Relaciones entre nodos

Según como se sitúen los nodos dentro del árbol se habla de relación entre ellos:

- <nombre> es hijo de <alumno>
- <nombre> y <DNI> son hermanos
- <matricula> es un ascendente de <nombre>
- <apellido1> es descendiente de <matriculas>

4.2. Expresiones

Los resultados que da la evaluación de una expresión son:

- Un conjunto de nodos (node-set)
 - No está ordenado.
 - Se considera que todos los elementos de un conjunto de nodos son hermanos, independientemente de lo que fueran originalmente.
 - Aunque los hijos de los nodos formar un conjunto de nodos son accesibles, los subárboles de un nodo no se consideran elementos del conjunto.
- Un valor booleano.
- Un número.
- Una cadena.

Elementos a utilizar en una expresión Xpath:

- Agrupaciones: (), {}, [].
- Elemento actual, elemento padre.
- Atributos: @.
- Elementos "*"
- Separadores "::-"
- Comas ","
- 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-attribute, 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, se utiliza; \$nombreVariable.

4.3. Procesando expresiones

- Nodo actual, es aquél en el que se encuentra el procesador.
- Nodo contexto, cada expresión está formada por subexpresiones que se van evaluando antes de resolver la siguiente.
- Tamaño del contexto, es el número de nodos que ese están evaluando en un momento dado en una expresión Xpath.

4.4. Ruta de localización

La ruta de localización de un nodo es la ruta que hay que seguir en un árbol para localizarlo. Se evalúan y esa evaluación devuelve un conjunto de nodos, o vacío.

Una ruta de localización se realiza mediante varios pasos de localización:

- **Ruta de localización del nodo raíz del documento:** Es la barra diagonal del documento, se trata de una ruta absoluta.
- **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 (ruta relativa). Puede devolver un elemento vacío.
- **Localización de atributos,** para referirnos a ellos se utiliza el símbolo @ seguido de su nombre.
- **Localización de espacios de nombres,** no se tratan explícitamente.
- **Localización de comentarios**
- **Localización de nodos de texto.**
- **Localización de instrucciones de procesamiento.**

Para comparar distintos elementos y tipos de nodo simultáneamente se utilizan los siguientes comodines:

- **Asterisco**, compara cualquier nodo de elemento, independientemente de su nombre. No compara atributos, 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, nodos de espacio de nombre, los atributos y comentarios.
- **@** compara los nodos de atributo.

4.5. Funciones del XPath

Podemos emplearlas unidas a las rutas de localización para hacer cosas sobre conjuntos de elementos que devuelven predicados.

Las más importantes son:

- **boolean()**, al aplicarla sobre un conjunto devuelve true si no es vacío.
- **not()**, al aplicarla sobre un predicado devuelve true si 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".
- **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.

4.6. Predicados

Es una expresión booleana que añade un nivel de verificación al paso de localización, en las que podemos incorporar funciones XPath.

La ventaja que ofrece es permitir seleccionar un nodo que cumple ciertas características.

Los predicados se incluyen dentro de una ruta de localización utilizando corchetes. Ejemplo:

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

En un predicado se pueden incluir:

- **Ejes**. Permiten seleccionar el subárbol dentro del nodo contexto que cumple un patrón. Puede ser o no de contenido:
 - **child**, es el eje 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 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, se pueden agrupar en función de los ejes a los que se puede aplicar.

Aplicable a cualquier eje:

- ***** solo devuelve elementos, atributos o espacios de nombres.
- **nod()** devuelve todos los nodos de todo tipo.

Aplicable a ejes de contenido:

- **text()**, devuelve cualquier nodo de tipo texto.
- **comment()** devuelve cualquier nodo de tipo contenido.
- **processing-instruction()** devuelve cualquier tipo de instrucción de proceso.

Varios predicados se pueden unir mediante operadores lógicos: and, or o not.

4.7. Ejercicio resuelto

```
<?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>
    </persona>
  </contactos>
</agenda>
```

```

        </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:

1. Nombre del propietario de la agenda.
2. Teléfono de casa del propietario.
3. Nombres y apellidos de los contactos de la agenda.
4. Nombre e identificador de cada contacto.
5. Datos del contacto con identificador "p02".
6. Identificadores de los contactos que tienen móvil.

- Nombre del propietario de la agenda.

```
/agenda/propietario/identificadores/nombre
```

- Teléfono de casa del propietario.

```
/agenda/propietario/telefonos/casa
```

- Nombres y apellidos de los contactos de la agenda.

```
//contactos/persona/identificadores/nombre |  
//contactos/persona/identificadores/apellidos
```

- Nombre e identificador de cada contacto.

```
//contactos/persona/identificadores/nombre | //contactos/persona/@id
```

- Datos del contacto con identificador "p02".

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

- Identificadores de los contactos que tienen teléfono en casa.

```
//contactos/persona/telefonos/casa/../../../../@id
```

4.8. Acceso a datos desde otro documento XML

Es útil poder acceder a los datos desde otros ficheros. Para ello se usa la función `document()`, pero no es del lenguaje XPath, sino que pertenece a XSLT.

Esta función admite dos argumentos:

`document(URI)` - devuelve el elemento raíz del documento XML que se localiza en el URI especificado.

`document(nodo)` - devuelve el conjunto de nodos cuya raíz es el nodo dado.

5. Utilización de plantillas

El elemento `xsl:template` controla el formato de salida que se le aplica a ciertos datos de entrada. Para especificar la salida se utilizan sentencias XHTML.

Para especificar donde queremos que se apliquen las plantillas utilizamos `xsl:apply-templates`.

El atributo `match` selecciona los nodos del árbol de entrada conforme a XPath, a los que se le va a aplicar la plantilla.

Con el atributo `select` seleccionamos los hijos del nodo de entrada conforme a XPath a los que se les aplica la plantilla.

Con ese atributo se puede especificar el orden en el que son procesados los nodos hijo, si no se especifica, se aplicará en el orden de arriba a abajo según se lee el documento XML.

Ejemplo

```
<xsl:template match="/">
<html>
  <body>
    <h2>Agenda de </h2>
    <xsl:apply-templates/>
  </body>
</html>
</xsl:template>
<xsl:template match="propietario/identificadores">
<h3>
<xsl:apply-templates select="apellidos"/>
,
<xsl:apply-templates select="nombre"/>
</h3>
</xsl:template>
```

El resultado:

Agenda de

López Terán , Alma

El Percebe 13, 6F Torrelavega 39300 970898765 942124567 628983456 Inés López Pérez El Ranchito 24, 6B Santander 39006 970123123 Roberto Gutiérrez Gómez El Marranito 4, 2F Santander 39004 970987456 942333323 Juan Sánchez Martínez El Cangrejo 10, sn Torrelavega 39300 997564343 942987974 677899234

5.1. Ejercicio resuelto de plantillas

```
<?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>
```

```

<?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>XML

```

Resultado

```

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

Inés,
López Pérez

Roberto,
Gutiérrez Gómez

Juan,
Sánchez Martínez

```


6. Procesadores 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 XML.

Puede estar integrado en el explorador web, servidor web o puede ser un programa desde la línea de comandos.

Existen diferentes modos de realizar la transformación XSLT:

- Mediante el procesador MSXML (servicios principales de Microsoft XML).
- Utilizando un procesador XSLTPROC como por ejemplo sltproc desde la línea de comandos.
- Invocando a la biblioteca de transformación desde un programa.
- Realizando un enlace entre la hoja XSLT y el documento XML, añadiendo en el fichero XML la definición de la versión XML y la definición del documento.

```
<?xml-stylesheet type="text/xsl" href="path_hoja_xsl"?>
```

La mayoría de editores XML permiten escoger el intérprete encargado de procesar un documento XSLT

7. Depuradores XSLT

Son elementos de software que permiten seguir la generación del documento a partir de un XML aplicándole hojas de estilo XSLT.

Nos facilitan la localización y corrección de errores dejando ejecutar código paso a paso.

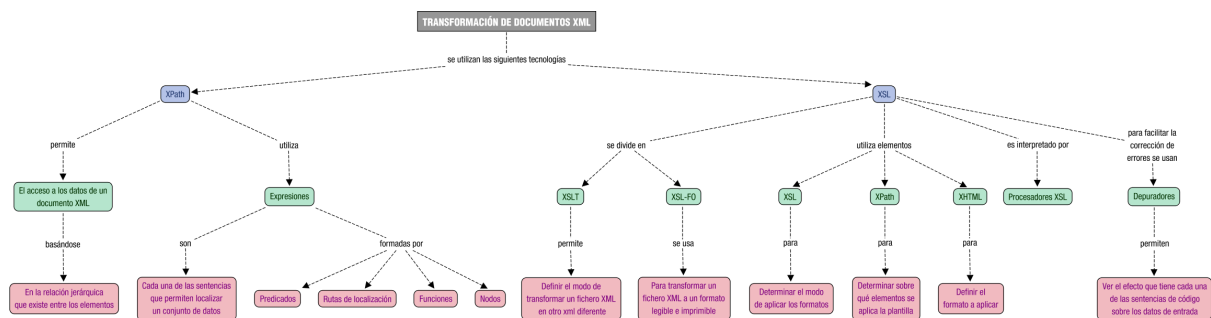
Algunos editores XML incluyen un depurador que permite visualizar la plantilla y los datos.

8. Para saber más

- [Recomendación del W3C sobre XSLT version 1.0.](#)





- Recomendación del W3C sobre XSLT version 1.1.
- Recomendación del W3C sobre XSLT version 2.0.
- Recomendación del W3C sobre XSLT versión 3.0.
- MCLibre - teoría y ejemplos XPath

Mapa conceptual





6. Almacenamiento de la información

 Autor	 Xerach Casanova
 Clase	Lenguajes de Marcas
 Fecha	@Mar 11, 2021 7:16 AM

1. Utilización de XML para almacenamiento de la información

1.1. Ámbitos de aplicación

2. Sistemas de almacenamiento de la información

3. XML y BD relacionales

3.1. De BD relacional a XML.

4. XML y BD orientadas a objetos

5. BD XML nativas

6. XQuery

6.1. Aplicaciones

6.2. Modelo de datos

6.3. Expresiones

6.4. Cláusulas

6.5. Ejemplos XQuery

6.6. Funciones

6.7. Ejemplo: definición y llamada a una función.

6.8. Operadores

Mapa conceptual

1. Utilización de XML para almacenamiento de la información

Podemos pensar en XML como una base de datos, una colección de documentos XML, cada uno de ellos representa un registro de la base de datos.

La estructura de un documento XML suele seguir un mismo esquema XML, aunque no es necesario que sea así. Cada archivo se puede configurar de forma estructurada e independiente, pero fácilmente accesible.

La ventaja principal de bases de datos XML es que proporcionan gran flexibilidad, lo que conlleva a crear aplicaciones que usen bases de datos XML.

1.1. Ámbitos de aplicación

Los documentos y requerimientos de almacenamiento de datos XL se pueden agrupar en:

- **Sistemas centrados en los datos.** Cuando los XML tienen una estructura bien definida y contienen datos que pueden ser actualizados y usados de diversos modos. Es apropiado para ítems con contenidos de periódicos, artículos, publicidad, facturas, órdenes de compra y algunos documentos menos estructurados.
- **Sistemas centrados en los documentos.** Cuando tienden a ser más impredecibles de tamaño y contenido. Presentan más tipos de datos, con reglas flexibles para campos y opcionales para el contenido.

La mayoría de los productos se enfocan en servir uno de esos dos formatos de datos mejor que el otro. Las bases de datos relacionales tradicionales tratan mejor requerimientos centrados en los datos, mientras que los sistemas de administración de contenido y documentos son mejores para almacenar datos centrados en el documento.

Los sistemas de bases de datos deben ser capaces de exponer datos relaciones y documentos XML, así como almacenar un documento XML

2. Sistemas de almacenamiento de la información

XML permite integrar 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 bbdd 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).

Debido a que no es una base de datos, XML carece de almacenamiento y actualización eficientes como índices, seguridad, transacciones, integridad de datos, acceso concurrente, disparadores, etc...

3. XML y BD relacionales

Las BD relacionales se basan en relaciones como medio para representar los datos del mundo real y están asociadas a SQL.

Las bases de datos relacionales suponen una posibilidad para el almacenamiento de datos XML, pero no están bien preparadas para almacenar estructuras de tipo jerárquico como lo son XML. Algunas causas:

- Las bd relacionales tienen estructura regular frente al carácter heterogéneo de los documentos XML.
- Los documentos XML suelen contener muchos niveles de anidamiento y los datos relacionales son planos.
- Los documentos XML tienen un orden intrínseco, los relacionales no son ordenados.
- Los datos relacionales son generalmente densos (cada columna tiene valor). Los datos XML son dispersos, pueden representar carencia de información mediante la ausencia del elemento.

Las razones para usar los tipos de BD relacionales y los productos de bd existentes para almacenar XML, aunque no sea de forma nativa son:

- Las bd relacionadas y orientadas a objetos son conocidas, mientras que las bd XML nativas son nuevas.
- como resultado, La familiaridad con las bd relacionales y orientadas a objetos, los usuarios se inclinan a ellas por el rendimiento.

3.1. De BD relacional a XML.

El proceso de traducción se descompone en los siguientes pasos.

- **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 se marcan como requeridos y las no requeridas se marcan como

opcionales en el esquema XML. Las columnas pueden ser también anidadas como elementos, pero pueden surgir problemas cuando el mismo nombre de columna se usa en más de una tabla.

- **Crear las claves primarias en el esquema XML.** Se puede agregar un atributo para la columna clave, con un ID agregado al nombre de la columna. Este atributo puede necesitar ser definido en el esquema XML como tipo ID. Para resolver el problema de posibles colisiones por coincidencia de nombres en las tablas de la bd relacional, se puede agregar el nombre del elemento (nombre de la tabla), al valor de la clave primaria (valor del atributo).
- **Establecer las relaciones de clave migrada.** Esto se logra mediante 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 una BD relacional.

4. XML y BD orientadas a objetos

Las bases de datos orientadas a objetos, soportan un modelo de objetos puro y no están basados en extensiones de otros modelos como el relacional.

- Están influenciados por lenguajes POO.
- Pueden verse como un intento de añadir funcionalidad SGBD a un lenguaje de programación.
- Son una alternativa para el almacenamiento y gestión de XML.

Componentes estándar de Orientación a Objetos:

- **Modelo de objetos:** Concebido para proporcionar un modelo de objetos estándar para BDOO. Es el modelo en que se basan los demás componentes.
- **Lenguajes de especificación de objetos (ODL)** para definir objetos.
- **Lenguaje de consulta de objetos (OQL)** para realizar consultas a objetos.
- **Bindings para C++, Java y Smalltalk.** Definen un lenguaje de manipulación de objetos OML, que extiende el lenguaje de programación para soportar objetos persistentes. Además incluyen soporte para OQL, navegación y transacciones.

Cuando se transforma el XML en objetos, son gestionados por SGBDOO, y se consulta la información mediante OQL, los mecanismos de indexación, optimización y procesamiento de consultas son del propio SGBDOO, no suelen ser específicos para el modelo XML.

5. BD XML nativas

Las bases de datos XML nativas son bases de datos y soportan transacciones, acceso multi-usuario, lenguajes de consulta, etc... y están diseñadas para almacenar documentos xml.

Se caracterizan por:

- Almacenar documentos en colecciones. Estas colecciones son como las tablas en BD relacionales.
- Validación de los documentos.
- Consultas. la mayoría de las BD XML nativas soportan uno o más lenguajes de consulta. El más popular es XQuery.
- Indexación XML. Se permite la creación de índices que aceleran consultas.
- Creación de identificadores únicos. A cada documento se le asocia una ID única.
- Actualizaciones y borrado.

Según el tipo de almacenamiento se divide en dos grupos.

- **Almacenamiento basado en texto.** Se almacena el documento en texto y proporciona alguna funcionalidad de base de datos para acceder a él. Dos posibilidades.
 - Posibilidad 1. se almacena como un BLOB en una bd relacional, mediante un fichero y proporciona algunos índices sobre el documento que aceleren el acceso a la info.
 - Posibilidad 2. Almacenar un 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, etc.
 - Posibilidad 2: traducir el DOM a objetos en un BDOO.
 - Posibilidad 3: utilizar un almacén creado para esta finalidad.

6. XQuery

XQuery es un lenguaje diseñado para escribir consultas sobre colecciones de datos expresadas en XML. Puede aplicarse a archivos XML y a bases de datos relacionales con funciones de conversión a registros XML. Su principal función es extraer información de un conjunto de datos organizados como un árbol de etiquetas XML. XQuery es independiente del origen de datos.

XQuery a XML es lo mismo que SQL a bases de datos relacionales.

Los requerimientos técnicos más importantes son:

- Debe ser un lenguaje declarativo.
- Debe ser independiente del protocolo de acceso a la colección de datos. XQuery funciona igual al consultar un archivo local, que al consultar una base de datos o un XML en una web.
- Las consultas y resultados deben respetar el modelo de datos XML.
- Las consultas y los resultados deben ofrecer soporte para los namespaces.
- Debe soportar XML-Schemas y DTD's, y también debe ser capaz de trabajar sin ellos.
- Ha de ser independiente de la estructura del documento, funciona sin conocerla.
- Debe soportar tipos simples, como enteros y cadenas, o tipos complejos como un nodo compuesto.
- Las consultas deben soportar cuantificaciones universales (para todo) y existenciales (existe).
- las consultas deben soportar operaciones jerárquicas de nodos y secuencias de nodos.
- Debe ser posible combinar información de múltiples fuentes en una consulta.
- Las consultas deben ser capaces de manipular datos independientemente del origen de estos.
- El lenguaje de consulta debe ser independiente de la sintaxis. Existen varias sintaxis distintas en una misma consulta XQuery.

6.1. Aplicaciones

- Recuperar información a partir de datos XML.
- Transformar una estructura de datos XML en otras estructuras que organizan la información de forma distinta.

- Ofrecer una alternativa a XSLT para realizar transformaciones.

Los motores XQuery de código abierto más relevantes son:

- BaseX. Open Source y disponible para Linux, Windows y Mac.
- Qexo. Escrito en Java y con licencia GPL que se distribuye dentro del paquete Kawa.
- Saxon. Escrito en Java y distribuido en múltiples paquetes, algunos de ellos open-source.

6.2. Modelo de datos

XQuery y SQL se pueden considerar similares, pero el modelo de datos de XQuery es distinto al de SQL. XML incluye conceptos como el de jerarquía y orden de datos.

A diferencia de SQL el orden en el que se encuentran los datos en XQuery es importante. No es lo mismo buscar una etiqueta dentro de <A>, que todas las etiquetas .

La entrada y salida de una consulta se define en términos de un modelo de datos, el cual proporciona una representación abstracta de uno o más XML.

Sus características son:

- Se basa en la definición de secuencia, como colección ordenada de cero o más ítems, pueden ser heterogéneas) contener varios nodos y valores atómicos), pero nunca puede ser un ítem de otra secuencia.
- Orden del documento. Corresponde al orden en que los nodos aparecerían si la jerarquía fuese representada en XML, si el primer carácter de un nodo ocurre antes que el primero de otro, lo precede también en el orden del documento.
- Contempla un valor especial llamado "error value", que es el resultado de evaluar una expresión que da error.

6.3. Expresiones

Una consulta XQuery es una expresión que lee una secuencia de datos XML y devuelve como resultado otra secuencia de datos en XML.

La mayoría de expresiones están compuestas por la combinación de expresiones más simples unidas por operadores y palabras reservadas.

XQuery se ha construido sobre la base de XPath, toda expresión XPath es una consulta XQuery válida.

Los comentarios se delimitan entre (: y :).

En un XQuery los caracteres {} delimitan expresiones que son evaluadas para crear un nuevo documento.

XQuery admite condicionales tipo if-then-else, con la misma semántica de los lenguajes habituales.

Pueden estar formadas por cinco tipos de cláusulas distintas. Siguen la norma FLWOR (se pronuncia flower). Estas cláusulas son los bloques principales de XQuery que equivalen a select, from, where, group by, having, order by y limit de SQL:

En una sentencia FLWOR al menos debe existir un FOR o un LET, el resto si existen deben respetar el orden dado por FLWOR.

Con estas sentencias se consigue buena parte de la funcionalidad que diferencia 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. Donde se declaran nombres, funciones, variables....
- Expresión, consulta propiamente dicha.

6.4. Cláusulas

- **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 ascendente, pero se puede usar el modificado DESCENDING para cambiar el sentido del orden.

- **RETURN:** construye el resultado de la expresión FLWOR para una tupla dada.

6.5. Ejemplos XQuery

Los siguientes ejemplos están realizados a partir del siguiente código.

```
<?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>16</paginas>
      <versionElectronica/>
    </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>
      <versionElectronica/>
    </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>
</biblioteca>

```

- El elemento raíz es **biblioteca**. Contiene un elemento libros.
- Dentro de libros, hay varios elementos libro.
- Los elementos **libro** tienen atributos **publicacion** y **edicion** (opcional). También tienen elementos titulo, autor (puede haber más de uno), **editorial**, **paginas** y un elemento opcional para indicar si hay edición electrónica, **edicionElectronica**.

Ejercicio 1

Título y editorial de todos los libros. Para devolver varios campos, los envolvemos en un elemento.

```

for $x in doc("libros.xml")/biblioteca/libros/libro
return <libro>{$x/titulo, $x/editorial}</libro>

```

```

<libro>
  <titulo>Learning XML</titulo>
  <editorial>O'Reilly</editorial>
</libro>
<libro>
  <titulo>XML Imprescindible</titulo>
  <editorial>O'Reilly</editorial>
</libro>
<libro>
  <titulo>XML Schema</titulo>
  <editorial>O'Reilly</editorial>
</libro>
<libro>
  <titulo>XPath Essentials</titulo>
  <editorial>Wiley</editorial>
</libro>
<libro>
  <titulo>Beginning XSLT 2.0: From Novice to Professional</titulo>
  <editorial>Apress</editorial>
</libro>

```

```
<libro>
  <titulo>XQuery</titulo>
  <editorial>O'Reilly</editorial>
</libro>
```

Ejercicio 2.

El título (sin etiquetas) de todos los libros de menos de 100 páginas. Para hacer comparaciones con números, lo mejor es convertir los datos con la función `number` para evitar problemas de tipo de dato o que los compare como cadenas.

```
for $x in doc("libros.xml")/biblioteca/libros/libro
where number($x/paginas) < 100
return data($x/titulo)
```

Learning XML

Ejercicio 3.

El número de libros de menos de 100 páginas. Utilizamos la función `count()`.

```
for $x in doc("libros.xml")/biblioteca/libros
let $y := $x/libro[number(paginas) < 100]
return count($y)
```

1

Ejercicio 4.

Una lista HTML con el título de los libros de la editorial "O'Reilly" ordenados por título. Podemos mezclar etiquetas HTML y XQuery y obtener HTML como resultado de una consulta.

```
<ul>
{
  for $x in doc("libros.xml")/biblioteca/libros/libro
  where $x/editorial = "O'Reilly"
  order by $x/titulo
  return <li>{data($x/titulo)}</li>
}
</ul>
```

```
<ul>
  <li>Learning XML</li>
  <li>XML Imprescindible</li>
  <li>XML Schema</li>
  <li>XQuery</li>
</ul>
```

Ejercicio 5

Título y editorial de los libros de 2002.

```
for $x in doc("libros.xml")/biblioteca/libros/libro
where $x[@publicacion=2002]
return <libro>{$x/titulo, $x/editorial}</libro>
```

```
<libro>
  <titulo>XML Schema</titulo>
  <editorial>O'Reilly</editorial>
</libro>
<libro>
  <titulo>XPath Essentials</titulo>
  <editorial>Wiley</editorial>
</libro>
```

Ejercicio 6

Título y editorial de los libros con más de un autor.

```
for $x in doc("libros.xml")/biblioteca/libros/libro
where count($x/autor)>1
return <libro>{$x/titulo, $x/editorial}</libro>
```

```
<libro>
  <titulo>XML Imprescindible</titulo>
  <editorial>O'Reilly</editorial>
</libro>
```

Ejercicio 7

Título y editorial de los libros que tienen versión electrónica.

```
for $x in doc("libros.xml")/biblioteca/libros/libro
where $x/versionElectronica
return <libro>{$x/titulo, $x/editorial}</libro>
```

```

<libro>
  <titulo>Learning XML</titulo>
  <editorial>O'Reilly</editorial>
</libro>
<libro>
  <titulo>XPath Essentials</titulo>
  <editorial>Wiley</editorial>
</libro>

```

Ejercicio 8

Título de los libros que no tienen versión electrónica.

```

for $x in doc("libros.xml")/biblioteca/libros/libro
where not($x/versionElectronica)
return$x/titulo

```

```

<titulo>XML Imprescindible</titulo>
<titulo>XML Schema</titulo>
<titulo>Beginning XSLT 2.0: Form Novice to Professional</titulo>
<titulo>XQuery</titulo>

```

6.6. Funciones

Las funciones que soporta XQuery son matemáticas, de cadenas, tratamiento de expresiones regulares, comparaciones de fechas y horas, manipulación de nodos, manipulación de secuencias, comprobación y conversión de tipos y lógica booleana. Algunas de ellas son:

• 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.
 - `exists()`, 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.
 - `data()`, devuelve el valor de los elementos que recibe como argumentos, es decir, sin etiquetas.
- Cuantificadores existenciales:
 - `some`, `every`, permiten definir consultas que devuelven algún, o todos los elementos, que verifiquen la condición dada.

Ejemplo de la función data:

Esta consulta devuelve todos los títulos, incluyendo las etiquetas:

```
for $x in doc("libros.xml")/biblioteca/libros/libro/titulo
return $x
```

Resultado:

```
<titulo>Learning XML</titulo>
<titulo>XML Imprescindible</titulo>
<titulo>XML Schema</titulo>
<titulo>XPath Essentials</titulo>
<titulo>Beginning XSLT 2.0: From Novice to Professional</titulo>
<titulo>XQuery</titulo>
```

Utilizando data:


```
for $x in doc("libros.xml")/biblioteca/libros/libro/titulo
return data($x)
```

se obtiene:

```
Learning XML
XML Imprescindible
XML Schema
XPath Essentials
Beginning XSLT 2.0: From Novice to Professional
XQuery
```

También se pueden crear funciones propias:

```
declare nombre_funcion($param1 as tipo_dato1, $param2 as tipo_dato2,... $paramN as tipo_datoN)
as tipo_dato_devuelto
{
  ...CÓDIGO DE LA FUNCIÓN...
}
```

6.7. Ejemplo: definición y llamada a una función.

En este ejemplo puedes ver la definición y llamada a una función escrita por el usuario que nos calcula el precio de un libro una vez que se le ha aplicado el descuento.

```
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>
```

6.8. Operadores

Agrupados por funcionalidad.

- **Comprobación de valores:** Comparan dos valores escalares y produce un error si alguno de los operandos es una secuencia de longitud mayor de 1.

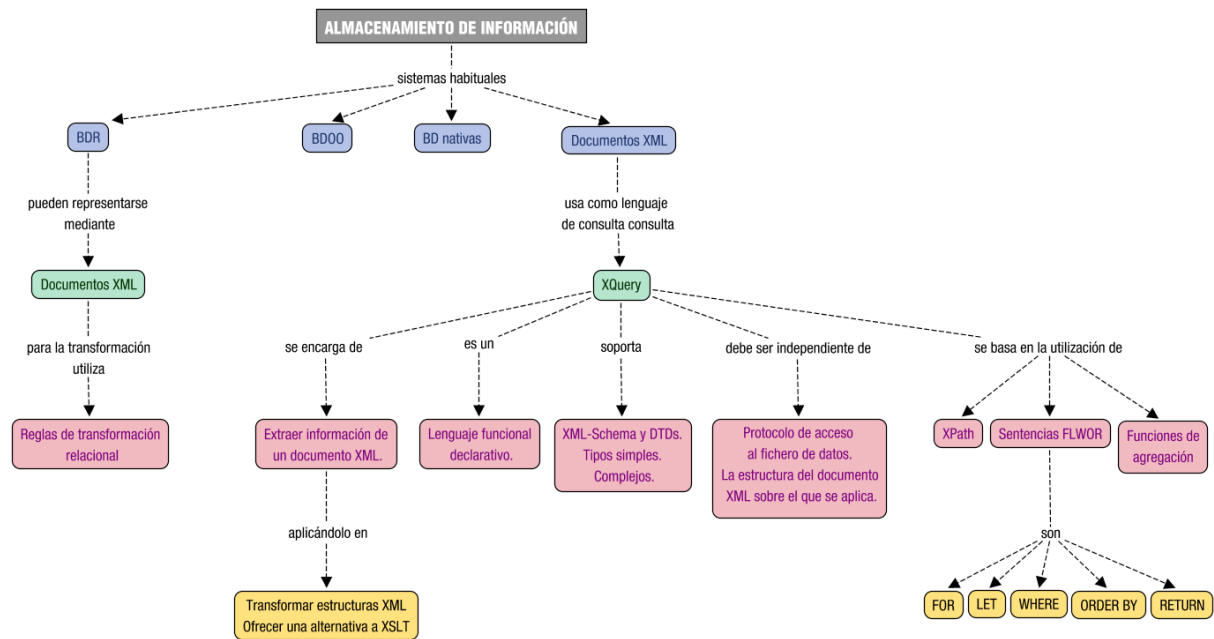
Estos operadores son:

- **eq:** igual.
- **nee:** no igual
- **lt:** menor qué
- **le:** menor o igual qué
- **gt:** mayor qué
- **ge:** mayor o igual qué
- **Comparación generales:** permiten comparar operandos que sean secuencias.
 - **=** igual
 - **!=** distinto
 - **>** mayor qué
 - **>=**, 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.
 - **Unión**, 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.

Mapa conceptual





7. Sistemas de gestión empresarial

Autor	ⓧ Xerach Casanova
Clase	Lenguajes de Marcas
Fecha	@Mar 14, 2021 3:54 PM

1. ERP y CRM
 2. Ventajas y desventajas
 3. Descarga e instalación
 - 3.1. Odoo
 4. Adaptación y configuración. Integración de módulos.
 5. Planificación de la seguridad
 6. Usuarios y roles
 7. Elaboración de informes
 8. Integración con aplicaciones ofimáticas
 9. Exportación de información
- Mapa conceptual

1. ERP y CRM

ERP (Sistemas de gestión empresarial)

Los ERP son sistemas de gestión de información que se están compuestos por varios módulos, cada uno de ellos proporciona una unidad de gestión y funcionalidad específica: producción, ventas, compras, pedidos, nóminas...

Los objetivos son:

- Optimizar procesos empresariales.
- Acceder a info confiable y precisa.
- permitir compartir información entre los componentes de la organización.
- Eliminar los datos y operaciones innecesarias.

Las características que diferencian a un ERP de otros software empresariales es que son sistemas integrales, modulares y adaptables. Se caracterizan por:

- Ser un programa con acceso a una bd.
- Sus componentes interactúan entre sí.
- Los datos deben ser consistentes, completos.

A veces son sistemas complejos y difíciles de implantar, ya que necesitan un desarrollo personalizado para cada empresa a partir del paquete inicial. Estas adaptaciones suelen encargarse a consultorías.

La consultoría en materia ERP puede ser:

- **Consultoría de negocios.** Estudia procesos de negocio de la compañía y evalúa su correspondencia con procesos del ERP para personalizarlo y ajustarlo a sus necesidades.
- **Consultoría técnica.** Conlleva el estudio de recursos tecnológicos existentes, en ocasiones implica la programación del sistema.

Gracias a la intranet, la mayoría de las empresas utilizan sistemas ERP con interfaz web.

CRM (Gestión de la relación con el cliente)

Un CRM es un sistema de gestión de información que se centra en gestionar toda la información referida a los clientes de la empresa. Los ERP suelen tener incluido un módulo CRM.

2. Ventajas y desventajas

Ventajas

- Integra diferentes unidades que facilitan la gestión de la información.
- Si incluye un CRM aporta beneficios de gestión de info y comunicación con clientes.
 - Aumenta la info que la empresa posee de sus clientes actuales o de los potenciales.
 - Permite dirigir la oferta hacia sus deseos y necesidades.
 - Optimiza el ciclo de vida de los productos y ciclo de venta.

Inconvenientes del uso de ERP

La mayoría de problemas con ERP son debidos a una inversión inadecuada de la formación del persona y falta de políticas corporativas.

Además existen las siguientes limitaciones y obstaculos.

- Ha de ser utilizado y realizado por personal capacitado.
- Son vistos como sistemas rígidos, difíciles de adaptar al modo de trabajo de las empresas.
- Sufren problemas de cuello de botella (los usuarios del sistemas se pueden ver afectados por la ineficiencia de uno de los departamentos participantes.
- Es muy caro modificar un ERP ya implantado.

3. Descarga e instalación

Para realizar la instalación primero debemos definir las necesidades a cubrir por el software y buscar el que mejor se ajuste.

Después es importante leer los requisitos mínimos para no tener problemas de funcionamiento.

Debemos tener claro que nuestro sistema se basa en una bd que previamente habremos creado en nuestro sistema.

Actualmente es habitual incorporar el ERP en la intranet. Para ello necesitamos un servidor web activo con soporte para bd y el lenguaje de script en el que se haya codificado la aplicación. La instalación se realiza a través de un navegador web.

Existen ERP en la nube, como www.salesforce.com, no tienen costes de instalación.

3.1. Odoo

Es un complemento sistema de gestión empresarial de código abierto y sin coste de licencias, que cubre necesidades de:

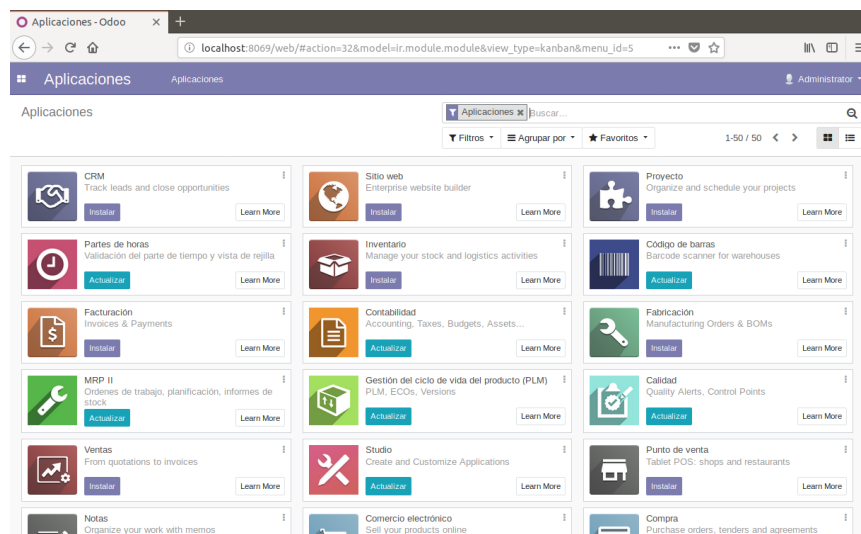
- Contabilidad y finanzas.
- Ventas.
- Recursos humanos.
- Compras.

- Proyectos.
- Almacenes.
- Relaciones con el cliente.
- Fabricación.

4. Adaptación y configuración. Integración de módulos.

Tras el proceso de instalación del paquete básico, viene la personalización para adecuarlo a la empresa que lo va a utilizar. Se puede incorporar el logo de la compañía, dar de alta a usuarios con permisos, configurar el sistema de avisos que proporciona la aplicación, configurar la compatibilidad con herramientas de correo...

Los ERP permiten incorporación de distintos módulos predefinidos que facilitan la personalización del paquete.



La integración de los módulos que complementan la aplicación base se puede realizar en el inicio o posteriormente.

Los recursos proporcionados son variados: creación de informes avanzados, servicios de comunicación para plataformas móviles, interconexión con el paquete ofimático de la empresa, CRM...

Los módulos incorporados después pueden estar prediseñados por el fabricante de la aplicación base, ser módulos programados por terceros para

ese software o ser programas solicitados a medida por la empresa.

Para realizar la instalación de módulo se entra en el paquete con un usuario con permisos de administrador y usa el cargador de módulos en la sección de administración.

5. Planificación de la seguridad

Las medidas de seguridad se basan en:

- **Niveles de acceso configurables para usuarios según su rol.**

En función de las tareas a realizar, el usuario cuenta con políticas de permisos para acceder a determinados datos.

- **Auditoría de cada transacción.**

Se controla cada envío de datos, que garantiza las operaciones realizadas.

- **Soporte de conexión segura mediante HTTPs.**

Para garantizar la seguridad de comunicación entre cliente y servidor, se realiza mediante un protocolo de comunicación seguro, que también se emplea para el proceso de autenticación de usuarios.

6. Usuarios y roles

Parte de la configuración de la seguridad es consecuencia de una buena asignación de roles a los usuarios del sistema, permitiéndoles solo el acceso a la información para realizar su trabajo.

Los paquetes básicos de ERP y CRM tienen varios tipos de usuarios posibles, en algunos ERP existe un paquete dedicado a la gestión de roles de usuario que añade funcionalidades extra.

Las características de los roles son:

- Un rol es un grupo particular de privilegios.
- Solo tiene validez cuando está asignado a algún usuario.
- Un usuario puede tener varios roles, prevalece el más restrictivo.
- Los cambios realizados en los roles no son efectivos hasta que se vuelve a iniciar sesión.

- Un rol niega el acceso a un módulo cuando se pierde la posibilidad de ver cualquier subpanel del mismo.

La asignación de roles las lleva un usuario administrador. Hay aplicaciones en las que los usuarios, no se pueden eliminar (pero sí desactivarlos) y se debe hacer desde la bd.

7. Elaboración de informes

Los sistemas ERP cuentan con herramientas para elaborar informes y ayudan a tomar decisiones de planificación, gestión y control.

Estas herramientas permiten a cualquier usuario sin conocimientos técnicos realizar informes dinámicos, flexibles e interactivos.

También realizan integración entre todos los sistemas o departamentos de la compañía.

La posibilidad de generar informes y tableros con gráficos permiten tomar decisiones estratégicas. Los equipos de Business Intelligence se encargan de la funcionalidad.

El módulo no está disponible en todas las ediciones de sistemas ERP.

8. Integración con aplicaciones ofimáticas

Los ERP y CRM permiten interacción con suites ofimáticas para procesar datos y realizar informes y escritos.

Se consigue así potenciar el uso de la herramienta sin salir de ella y se logra un flujo de información automática facilitando lectura de cartas u otros documentos. Además, se almacenan los documentos generados dentro de la ERP y facilita así el compartirlos. La integración se realiza a través de:

- Instalación de módulo en la aplicación ERP.
- Instalación de un complemento en la suite ofimática incorporando una nueva barra de herramientas.

Muchos ERP soportan integración en herramientas ofimáticas Cloud.

9. Exportación de información

La exportación de la información se facilita al integrar aplicaciones ofimáticas dentro de los ERP.

Con ello, podemos exportar hojas de cálculo, realizar gráficos e incorporar presentación de diapositivas, incorporar una oferta para enviar por correo electrónico, proporcionar rapidez en la ejecución de la tarea deseada al permitirse flujo de datos entre ERP y suite ofimática...

Por otro lado, tiene el inconveniente de que se puede filtrar información delicada de nuestra empresa al exterior.

En el caso de Odoo la mayoría de informes se generan solamente en PDF, aunque existen módulos de terceros que facilitan realizar otras exportaciones.

Mapa conceptual

