

Lenguajes de marcas y sistemas de gestión de la información

Tema 1: Lenguajes de marcas

1. Introducción

Definición:

Es la forma de **codificar un documento** de manera que, **junto con el texto, se incorporan etiquetas, marcas con información adicional** sobre la estructura del texto o su formato de presentación. Permiten hacer explícita la estructura de un documento, su contenido semántico o cualquier otra información.

La estructura y la sintaxis de un lenguaje de marcas quedan definidas en el DTD (Document Type Definition). En él se establecen las marcas, los elementos utilizados por dicho lenguaje y sus correspondientes etiquetas y atributos, su sintaxis y normas de uso.

Ejemplo:

```
<contacto>
  <nombre>Esther</nombre>
  <apellidos>Lastra</apellidos>
  <telefono>111222333</telefono>
  <mail>esther@correo.es</mail>
</contacto>
```

El lenguaje de marcas es el que especifica cuáles serán las etiquetas posibles, dónde deben colocarse y el significado que tendrá cada una de ellas. Por tanto, aplica un determinado formato a la información que transmitimos. En el caso de páginas web, por ejemplo HTML es el navegador o agente de usuario (user –agentswitcher), quien interpreta las marcas del formato.

Ejemplo:

```
<html>
  <head>
    <title>Ejemplo 1 HTML</title>
  </head>
```

```

<body bgcolor='green'>
  <br/>
  <h1>Mi primer documento</h1>
  <p>Aquí escribo párrafos</p>
  <br/>
  <p><b>Vamos a ver</b></p>
    <ul>
      <li>XML </li>
      <li>HTML</li>
      <li>CSS</li>
    </ul>
  </body>
</html>

```

Etiquetas, elementos y atributos

- Una **etiqueta** (tag) es un texto que va entre el símbolo menor que (<) y el símbolo mayor que (>). Normalmente, se utilizan dos etiquetas: una de inicio y otra de fin para indicar que ha terminado el **efecto que queremos presentar**. La única diferencia entre ambas es que la de cierre lleva una barra inclinada "/" antes del código.

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

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

- Los **elementos** representan estructuras mediante las que se **organizará el contenido** del documento o acciones que se desencadenan cuando el programa navegador interpreta el documento. Consta de la etiqueta de inicio, la etiqueta de fin y de todo aquello que se encuentran entre ambas.

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

- Un **atributo** es un **par nombre-valor** que se encuentra **dentro de la etiqueta de inicio** de un elemento e indican las **propiedades que pueden llevar asociadas los elementos**. Los atributos no pueden organizarse en ninguna jerarquía, no pueden contener ningún otro elemento o atributo y no reflejan ninguna estructura lógica. No se debe utilizar un atributo para contener información susceptible de ser dividido.

Ejemplo

`<direccion>`

```
<destinatario>
  <nombre>Esther</nombre>
  <apellidos>Lastra</apellidos>
</destinatario>
<calle>Santa Lucía</calle>
<numero>36</numero>
<localidad ccaa="Cantabria">Santander</localidad>
</direccion>
```

En este ejemplo, el elemento <destinatario> contiene dos elementos hijos: <nombre> y <apellidos> y ccaa es un atributo del elemento <localidad>

Diferencias entre lenguajes de marcas y lenguajes de programación.

Los lenguajes de marcas **no disponen** de los elementos típicos de programación como **variables, arrays, sentencias de control, funciones, etc...** Por lo tanto, los lenguajes de marca no se programan.

Sin embargo, los lenguajes de marca se pueden **combinar** dentro del mismo documento, con otros lenguajes como javascript o PHP, que si son lenguajes de programación, con el objetivo de aportar funcionalidad o dinamismo a la página web.

Independencia del destinatario de la información.

El lenguaje de marcas debe ser independiente del destinatario final, es el **intérprete del lenguaje** quien se encarga de representar las marcas de forma adecuada.

Para independizar aún más la representación de la página web de su contenido, se creó **CSS, que es un lenguaje de estilos.**

El CSS permite especificar con mayor precisión y eficacia **la representación de la información**, para cada intérprete y para diferentes soportes, como monitores, dispositivos móviles, papel, voz.

Características

- **Uso del texto plano.** Los archivos están compuestos únicamente de caracteres de texto. Estos caracteres se pueden codificar con distintos códigos dependiendo del idioma o alfabeto. Ejemplo: ASCII, UTF-8.

- **Es un lenguaje compacto.** Las instrucciones de marcado se mezclan con el propio contenido.
- **Independencia del dispositivo final.** El mismo documento puede ser interpretado de diferentes formas dependiendo del dispositivo final. Así tendremos diferentes resultados si se usa un dispositivo móvil, un pc o una impresora.
- **Especialización del lenguaje.** Inicialmente los lenguajes de marcas se idearon para visualizar documentos de texto, en la actualidad se han especializado según el tipo de documento que necesitemos procesar. (XML, HTML, RSS...)

2. HISTORIA. INTRODUCCIÓN A LOS PRINCIPALES LENGUAJES DE MARCAS

En los años 60, manejar documentos electrónicos tenía el problema de **falta de compatibilidad entre aplicaciones**, IBM intentó resolver estos problemas a través de un lenguaje de marcas denominado

GML (Generalized Markup Language)

IBM encargó a **Charles F. Goldfab** la construcción de un **sistema de edición, almacenamiento y búsqueda de documentos legales**. Tras analizar el funcionamiento de la empresa llegaron a la conclusión de que:

- Para realizar un buen procesado informático de los documentos había que **establecer un formato estándar para todos los documentos** que se manejaban en la empresa. Con ello se lograba **gestionar cualquier documento en cualquier departamento y con cualquier aplicación**, sin tener en cuenta dónde ni con qué se generó el documento.
- Dicho **formato tenía que ser válido para los distintos tipos de documentos** legales que utilizaba la empresa. Debía ser **flexible** para que se pudiera ajustar a las distintas situaciones.

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

SGML(Standard Generalized Markup Language)

En 1986 GML pasó a manos de **ISO (International Organization for Standardization)** y se convirtió en SGML.

Lenguaje de **software libre y de código abierto** (ISO 8879). SGML es un **metalenguaje**, es decir, un **conjunto de normas que permiten crear otros lenguajes de marcas**. Esto se hace definiendo un vocabulario o conjunto de elementos a utilizar y una gramática o conjunto de reglas que rigen el uso de los elementos y atributos. Por ejemplo HTML, es uno de los lenguajes creados a partir de SGML.

SGML permite que **la estructura** de un documento **pueda ser definida** en base a la relación lógica de sus partes. Esta estructura puede **ser validada** por la **definición de tipo de datos (DTD)**. La norma ISO8879 define la sintaxis del documento y la sintaxis y semántica de DTD.

Éste era un lenguaje muy complejo y requería de unas herramientas de software caras. Por ello su uso ha quedado relegado a grandes aplicaciones industriales.

Sample from SGML		
element	tag	attributes
starter document	<sd>	sec=and status=
title	<td>	
heading level 1	<h1>	id=and st=
paragraph	<p>	
heading reference	<hr>	pn and rid=

```
<!DOCTYPE motd [ <!--
<motd>
<!-- created: 2003-12-12-->
<sentence>Do not throw
out the <keep>baby</>
with the
<refuse>dirty</>,
<refuse>stinky</>,
<refuse>bathwater</>.
</>
<!-- finish this later-->
</motd>
```



HTML (Hyper Text Markup Language)

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

- ASCII.
- SGML.

HTML es una versión simplificada de SGML, de manera que rápidamente se convirtió en un **estándar para la creación de páginas web**. Pero, a pesar de todas estas ventajas HTML presenta las siguientes **desventajas**:

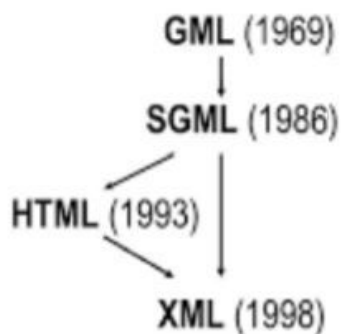
- **No soporta tareas** de impresión y diseño.
- El lenguaje **no es flexible**, las etiquetas son limitadas.
- No permite mostrar contenido dinámico.
- **La estructura y el diseño están mezclados** en el documento.

XML (eXtended Markup Language)

Finalmente, en el **1998, W3C (Word Wide Web Consortium)** hizo público un nuevo estándar que denominaron: XML (eXtended Markup Language)

Más sencillo que SGML y más potente que HTML. **Es una simplificación y adaptación de SGML, que permite definir lenguajes específicos**. Por lo tanto XML no es un lenguaje en particular, sino un **metalenguaje**. Utiliza reglas más estrictas que SGML (más formal). Se usa mucho para **base de datos y para exportación e importación de la información**. Algunos lenguajes basados en XML son (RSS, XHTML...).

XML no se utiliza sólo en internet, sino que se está convirtiendo en un estándar para el intercambio de información estructurada entre diferentes plataformas.



XHTML (eXtensible HyperText Markup Language)

XHTML es una versión más estricta y limpia de XML. Surge con el objetivo de **sustituir a HTML 4.0 debido a las limitaciones que este tiene en cuanto a la mezcla de la estructura del documento con la presentación.**

XHTML extiende HTML 4.0 combinando **HTML (diseñado para mostrar datos)**, con **XML (diseñado para describir los datos)**.

Ejercicio 1

- Busca información en Internet sobre XML y HTML y haz una tabla comparativa de ambos.
- Busca información en Internet sobre XML y SGML y haz una tabla comparativa de ambos.
- Construye un ejemplo de XML y HTML con ellos en el aula. (XML-> catálogo de libros, HTML-> aparezca por pantalla Mi primera clase de LMSG).
 - ¿Qué lenguaje presenta la información de forma legible para cualquiera?
 - ¿Qué lenguaje muestra la estructura u organización jerárquica de la información?
 - ¿Qué lenguaje estará más orientado a la presentación?
 - ¿Qué lenguaje muestra mejor la estructura del documento?
- Busca información en Internet sobre cómo la clasificación de los Lenguajes de Marcas.

3. TIPOS DE LENGUAJES DE MARCAS

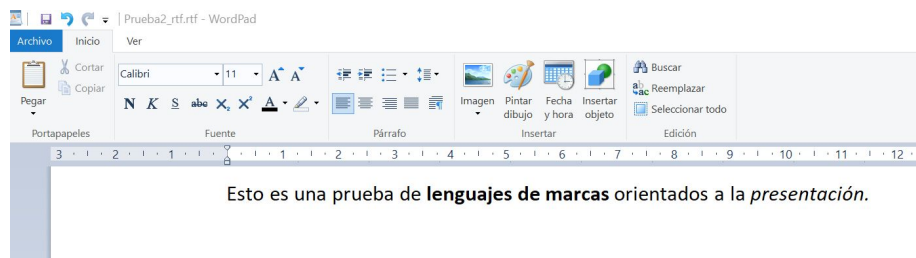
En un documento pueden combinarse varios tipos diferentes de lenguajes de marca, pero normalmente éstos se pueden clasificar como sigue:

- **De presentación:** Este tipo de lenguajes son los **usados tradicionalmente por los procesadores de texto** como puede ser Microsoft **WordPad** RTF y codifican cómo ha de presentarse el documento. Indica el formato del texto (información para el maquetado). Por ejemplo, indicando que una determinada palabra debe presentarse en fuente itálica y en negrita.

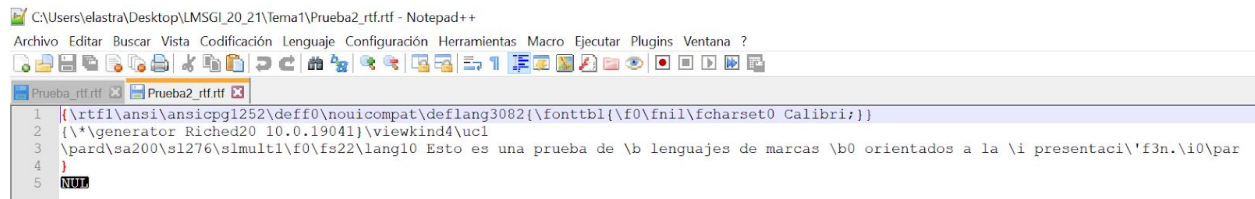
Ejemplo de RTF

RTF (Rich Text Format - Formato de Texto Enriquecido): Desarrollado por Microsoft en 1987. Es un lenguaje de marcas orientado a la presentación.

Escribe un pequeño texto en WordPad y guardalo como archivo .rtf.



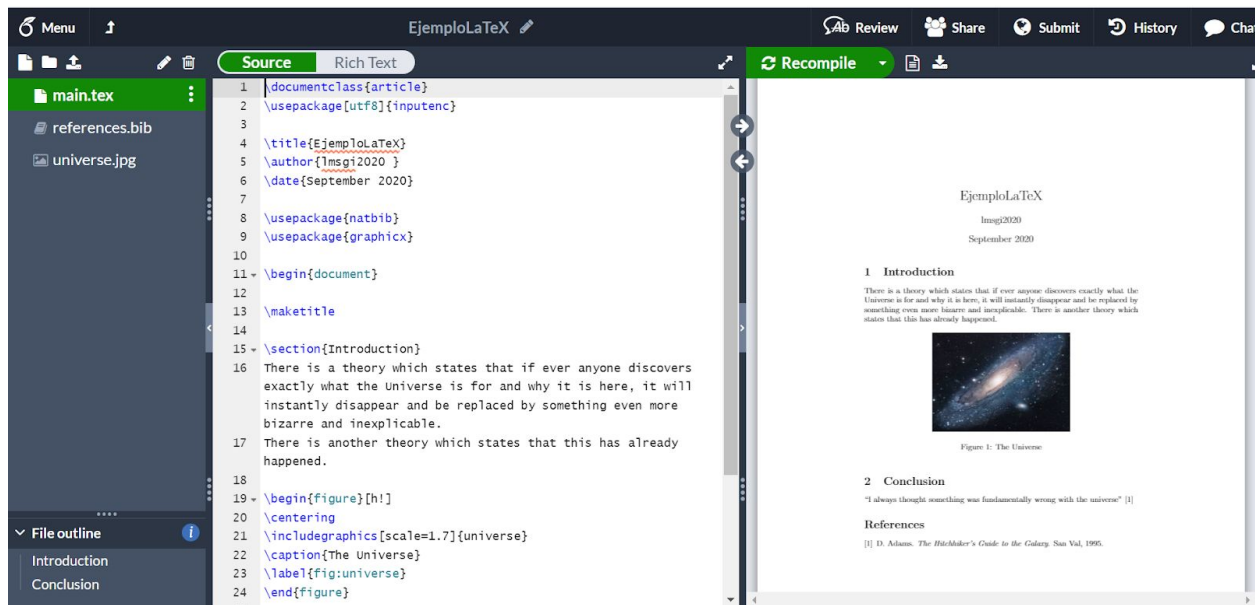
Si lo quisieras editar utilizando por ejemplo notepad, se mostraría del siguiente modo:



- **De procedimientos:** En este tipo de lenguajes las **etiquetas son también orientadas a la presentación** pero se integran dentro de un marco procedural que permite definir **macros (secuencias de acciones) y subrutinas**. Entre los ejemplos más comunes de lenguajes procedurales podemos encontrar TeX, LaTeX y postcript.

Ejemplo 1 de LaTeX:

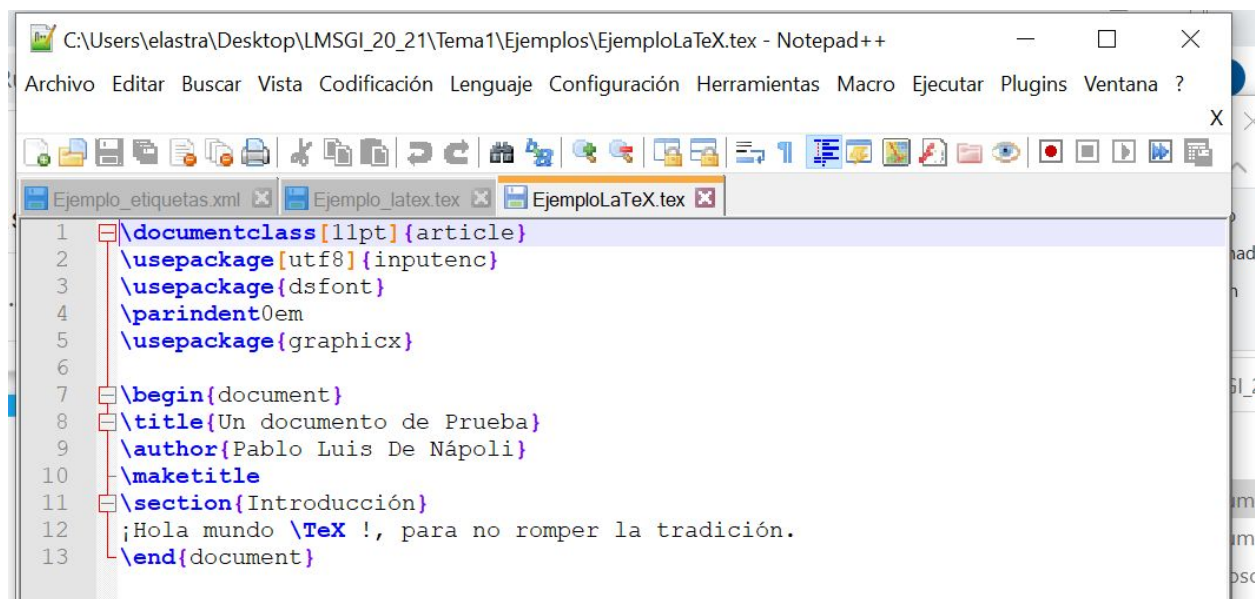
Overleaf es un programa online que nos permite editar y compilar desde nuestro navegador un documento de LaTeX.



A la izquierda escribimos el código en LaTeX, y a la derecha podemos comprobar el resultado del código.

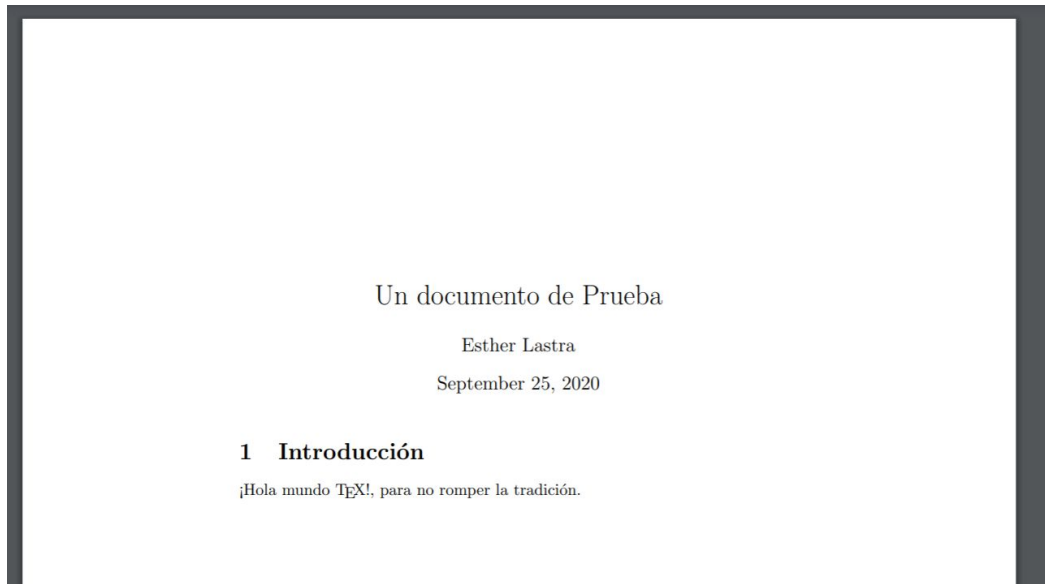
Ejemplo 2 de LaTeX:

Copia en un documento notepad el siguiente código LaTeX:



Podemos comprobar el resultado en la página web

<http://latex.informatik.uni-halle.de/latex-online/latex.php>



El programa que representa el documento debe **interpretar el código en el mismo orden en que aparece**.

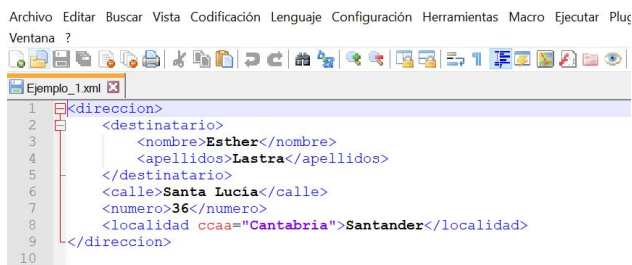
- Enfocado hacia la presentación del texto, sin embargo, también es **visible para el usuario que edita el texto**.
- Por ejemplo, para formatear un párrafo, debe haber una **serie de directivas inmediatamente antes del texto en cuestión, indicando al software instrucciones** tales como aumentar el tamaño de la fuente, o poner negrita. Inmediatamente después del título deberá haber etiquetas inversas de cierre de la directiva.
- Lenguaje de marcado utilizado en aplicaciones de edición profesional, manipulados por tipógrafos cualificados, ya que es complejo.
- **Descriptivo o semántico:**

Este tipo de lenguajes no definen qué debe hacer con un trozo o sección del documento sino que por lo contrario las marcas sirven para indicar qué es esa información, esto es, **describen qué es lo que está representando pero no indican cómo se debe presentar en pantalla** el

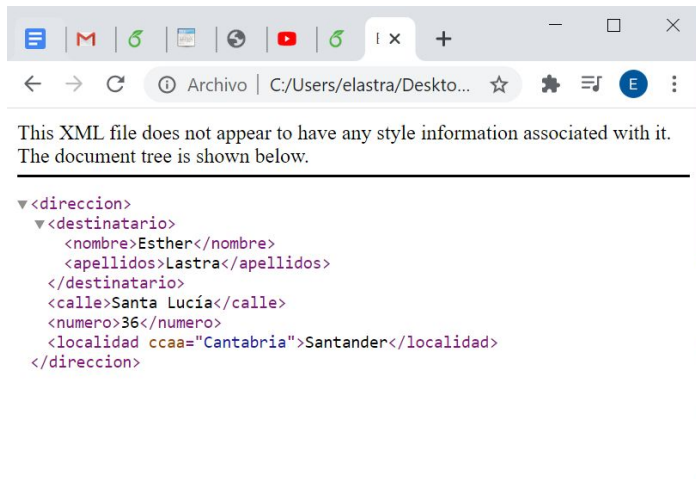
mismo. La mayoría de los lenguajes de marcas que se hoy en día se encuentran dentro de este grupo. Como por ejemplo de XML (de SGML) y JSON en el que la presentación nunca se indica en el documento; simplemente se indica una semántica de contenido que lo hace ideal para almacenar datos.

- Utilizan las **etiquetas para describir los fragmentos de texto**, pero sin especificar cómo deben ser representados, o en qué orden.
- Los lenguajes expresamente diseñados para generar marcado descriptivo es **XML** (SGML).
- Una de las virtudes del marcado descriptivo es su **flexibilidad**: los fragmentos de texto se etiquetan tal como son, y no tal como deben aparecer. Estos fragmentos pueden utilizarse para más usos de los previstos inicialmente. Por ejemplo, los hiperenlaces fueron diseñados en un principio para que un usuario que lee el texto los pulse. Sin embargo, los buscadores los emplean para localizar nuevas páginas con información relacionada, o para evaluar la popularidad de determinado sitio web.
- El marcado descriptivo **está evolucionando** hacia el marcado genérico. **Los nuevos sistemas de marcado descriptivo estructuran los documentos en árbol, con la posibilidad de añadir referencias cruzadas. Esto permite tratarlos como bases de datos, en las que el propio almacenamiento tiene en cuenta la estructura.** Estos sistemas no tienen un esquema estricto como las bases relacionales, por lo que a menudo se las considera bases semi-estructuradas.

Ejemplo XML



```
Archivo  Editar  Buscar  Vista  Codificación  Lenguaje  Configuración  Herramientas  Macro  Ejecutar  Plug
Ventana ?
Ejemplo_1.xml
1 <direccion>
2   <destinatario>
3     <nombre>Esther</nombre>
4     <apellidos>Lastra</apellidos>
5   </destinatario>
6   <calle>Santa Lucia</calle>
7   <numero>36</numero>
8   <localidad ccaa="Cantabria">Santander</localidad>
9 </direccion>
10
```



Ejemplo JSON

```
1  {
2    "marcadores": [
3      {
4        "latitude": 40.416875,
5        "longitude": -3.703308,
6        "city": "Madrid",
7        "description": "Puerta del Sol"
8      },
9      {
10       "latitude": 40.417438,
11       "longitude": -3.693363,
12       "city": "Madrid",
13       "description": "Paseo del Prado"
14     },
15     {
16       "latitude": 40.407015,
17       "longitude": -3.691163,
18       "city": "Madrid",
19       "description": "Estación de Atocha"
20     }
21   ]
22 }
```

<pre> <contacto> <nombre>Esther</nombre> <telefono>66666666</telefono> <email>esther@mail.com</email> </contacto> </pre>	<pre> { "contacto": { "nombre": "Esther", "telefono": "66666666", "email": "esther@mail.com" } } </pre>
<pre> <contacto> <nombre>Esther</nombre> <apellidos>Lastra</apellidos> <direccion ccaa="Cantabria"> <calle>Avenida Valdecilla</calle> <numero>16</numero> <piso>2A</piso> </direccion> </contacto> </pre>	<pre> { "contacto": { "nombre": "Esther", "apellidos": "Lastra", "direccion": { "-ccaa": "Cantabria", "calle": "Avenida Valdecilla", "numero": "16", "piso": "2A" } } } </pre>
<pre> <agenda> <contacto> <nombre>Esther</nombre> <email>esther@mail.com</email> </contacto> <contacto> <nombre>Ana</nombre> <email>Ana@mail.com</email> </contacto> <contacto> <nombre>Pepe</nombre> <email>pepe@mail.com</email> </contacto> </agenda> </pre>	<pre> { "agenda": { "contacto": [{ "nombre": "Esther", "email": "esther@mail.com" }, { "nombre": "Ana", "email": "Ana@mail.com" }, { "nombre": "Pepe", "email": "pepe@mail.com" }] } } </pre>

Podemos editar y validar un json en línea en las páginas web:

<https://jsoneditoronline.org/>

<https://jsonformatter.org/json-editor>

4. XML

El organismo W3C en 1998, trata de resolver los problemas del HTML y crea el estándar internacional XML, se trata de no incluir información sobre el diseño sino sobre la estructura, de forma que con las etiquetas se muestra el significado en vez del formato de los datos que se van a visualizar.

<https://www.w3.org/standards/xml/>

Características

- **Extensible.** Permitir definir etiquetas propias y permitir asignar atributos a las etiquetas.
- **Estructurado.** Se pueden modelar datos a cualquier nivel de complejidad.
- **Versátil.** La estructura y el diseño son independientes.
- **Validable:** cada documento se puede validar frente a un DTDSchema. Utilizar un esquema para definir de forma exacta las etiquetas y los atributos.
- **Abierto.** Independientemente de empresas, sistemas operativos, lenguajes de programación o entornos de desarrollo. La difusión de los documentos XML está asegurada ya que cualquier procesador de XML puede leer un documento de XML.
- **Sencillo.** Fácil de aprender y de usar. No se requieren conocimientos de programación para realizar tareas sencillas en XML.
- Compatible con protocolos que ya funcionan, como HTTP y los URL.
- El marcado de XML es legible para los humanos.
- El diseño XML es formal y conciso.
- XML es extensible, adaptable y aplicable a una gran variedad de situaciones.
- XML es orientado a objetos.
- Todo documento XML se compone exclusivamente de datos de marcado y datos carácter entremezclados.



XML está formado por un conjunto de estándares relacionados entre sí:

- **XSL.**
- **XML Linking Language.**
- **XML Namespaces.**
- **XML schema**

Herramientas básicas de XML

Para trabajar en XML es necesario:

- 1) Editar los documentos.
- 2) Procesarlos.

Se necesitan dos tipos de herramientas: Editores y Procesadores.

Editores XML

Los lenguajes de marcas es que se basan en la utilización de ficheros de texto plano con el bloc de notas podemos crearlos.

Para documentos XML complejos es mejor usar algún editor XML que ayudan a crear estructuras y etiquetas de los elementos usados en los documentos, incluyen ayuda para la creación de otros elementos como DTD, hojas de estilo CSS o XSL, ...


El W3C ha desarrollado un editor de HTML, XHTML, CSS y XML gratuito cuyo nombre es Amaya.

Nosotros utilizaremos: **Notepad++** al principio.

<https://notepad-plus-plus.org/download/v7.5.7.html>

Procesadores XML

El código XML se puede interpretar en cualquier navegador. Los procesadores de XML permiten leer los documentos XML y acceder a su contenido y estructura. El modo en que los



procesadores deben leer los datos XML está descrito en la recomendación de XML establecida por W3C.

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

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

XML: Estructura y sintaxis

El XML, o Lenguaje de Etiquetas Extendido, es lenguaje de etiquetas. No representa datos por sí mismo, solamente organiza la estructura.


El XML ahorra tiempos de desarrollo y se ha convertido en un formato universal que ha sido asimilado por todo tipo de sistemas operativos y dispositivos móviles.

Al igual que en HTML un documento XML es un documento de texto, en este caso con extensión “.xml”, compuesto de:

- parejas de etiquetas,
- estructuradas en árbol, que describen una función en la organización del documento,
- puede editarse con cualquier editor de texto,
- es interpretado por los navegadores Web.

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

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



El marcado en XML son etiquetas que se añaden a un texto para estructurar el contenido del documento. El marcado es todo lo que se sitúa entre los caracteres “<” y “>” o “&” y “;”. El marcado puede ser tan rico como se quiera.

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

- Antes del prólogo.
- Dentro de una etiqueta.

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

El prólogo

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

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

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

El prólogo puede tener tres funciones:

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

Para ello se utiliza la etiqueta:

```
<?xml versión= “1.0”>
```

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

- Declaración de la codificación empleada para representar los caracteres.

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

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

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

Los códigos más importantes son:

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

- Declaración de la autonomía del documento.

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

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

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

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

<!DOCTYPE Nombre_tipo ...>

El ejemplar. Los elementos

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

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

En el ejemplo:

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

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

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

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

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

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

Carácter	Cadena
>	>
<	<

Carácter	Cadena
&	&
“	"

Carácter	Cadena
‘	'

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

Documentos XML bien formados

Documentos XML bien formados

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

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

Utilización de espacios de nombres en XML

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

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

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

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

Esta etiqueta se denomina “nombre cualificado”. Al definir el prefijo hay que tener en cuenta que no se pueden utilizar espacios ni caracteres especiales y que no puede comenzar por un dígito.

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

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

Ejemplo: Sean los documentos XML que organizan la información sobre los profesores y los alumnos del ASIR1 respectivamente:

```
<?xml version="1.0" encoding="iso-8859-1" standalone="yes"?>
<alumnos>
  <nombre>Ignacio·Fernández·González</nombre>
  <nombre>Marina·González·Fernández</nombre>
  <nombre>Javier·Martínez·López</nombre>
</alumnos>
```

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

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

```
<?xml:version="1.0" encoding="iso-8859-1" standalone="yes"?>
<asistentes xmlns:alumnos="http://ASIR1/alumnos" xmlns:profesores="http://ASIR1/profesores">
<alumnos:nombre>Ignacio Fernández González</alumnos:nombre>
<alumnos:nombre>Marina González Fernández</alumnos:nombre>
<alumnos:nombre>Javier Martínez López</alumnos:nombre>
<profesores:nombre>Ricardo Ruiz Pérez</profesores:nombre>
<profesores:nombre>Marta Rodríguez Hernández</profesores:nombre>
</asistentes>
```