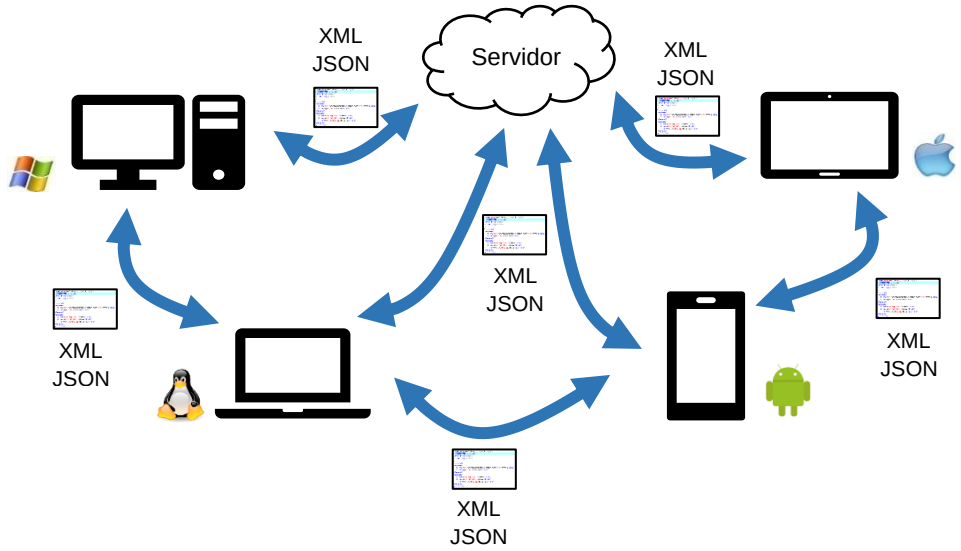
ÍNDICE

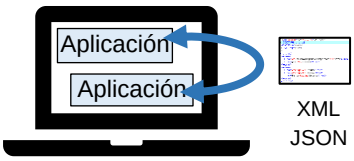
1. Introducción

1.1 Lenguajes de almacenamiento y transmisión de información

Existen distintos tipos de lenguajes que permiten el almacenamiento y la transmisión de información.

Mediante estos lenguajes es posible estructurar datos para compartirlos entre distintos sistemas o aplicaciones.





1.2 Tipos de lenguajes

Estos lenguajes se clasifican en dos tipos:

- De listas: forma de codificar un documento que, junto con el texto, incorpora símbolos y metadatos para estructurar los datos. En esta categoría se encuadra el lenguaje **JSON** (JavaScript Object Notation).

- De marcas: forma de codificar un documento que, junto con el texto, incorpora etiquetas para estructurar los datos. En esta categoría se encuadra el lenguaje **XML** ((eXtended Markup Language). Se profundizará en este lenguaje más adelante.

- **Lenguaje de listas: JSON (JavaScript Object Notation)**

JSON es un formato de texto sencillo para **el intercambio de datos**.

Se trata de una notación de datos procedente del lenguaje JavaScript: utiliza la sintaxis de objetos de ese lenguaje para representar datos estructurados. Debido a su amplio uso, actualmente se considera un formato independiente del lenguaje JavaScript.

No se considera lenguaje de marcas, ya que no hay diferencia en el texto a través de etiquetas.

Hay dos elementos básicos:

OBJETO	Se confina entre { }
	No se da nombre al objeto.
	
	Es un conjunto de datos separados por comas:
	- Cada dato es una pareja "nombre":valor (o un array)
	- El nombre del dato se especifica entre comillas dobles.
	- El valor se indica con comillas dobles si se trata de un texto.
		`{`
		`"nombre": "Ana",`
		`"edad": 45,`
		`"profesión": "barrendera"`
		`}`

ARRAY	Se confina entre [ ]
	Sí se da nombre al array
	
	Es una colección de elementos separados por comas, que pueden ser valores simples u objetos.
		`"aficiones": [`
		`"tocar la guitarra",`
		`"pintura",`
		`"leer"`
		`]`

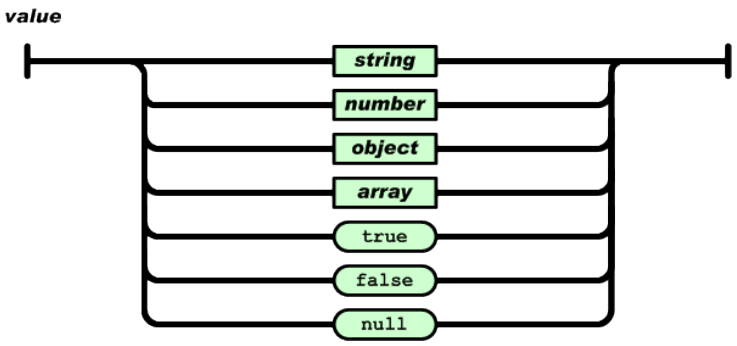
Dentro de un objeto pueden haber arrays:

```
{
   "nombre": "Ana",
   "edad": 45,
   "profesión": "barrendera",
   "aficiones": [
                 "tocar la guitarra",
                 "pintura",
                 "leer"
                ]
}
```

Dentro de un array pueden haber objetos:

```
"teléfonos": [
   {
      "tipo": "fijo",
      "número": "555 9666 777"
   },
   {
      "tipo": "móvil",
      "number": "444 555 666"
   }
]
```

Un valor puede ser:



- Mediante estas estructuras se pueden crear jerarquías de datos complejas
- Un archivo JSON debe ser necesariamente codificado como UTF8. Todo el contenido debe estar comprendido entre los símbolos { y }.
- No existen esquemas de validación
- No se pueden incluir comentarios

Actividad 1: JSON

1.1 Crear un documento JSON
1.2 Validación de la sintaxis del documento

Aunque en la actividad crearemos un documento JSON manualmente, lo normal es que este tipo de archivos se generen automáticamente a través de alguna función o aplicación, a partir de una base de datos u otra fuente de datos.

- **Lenguaje de marcas: XML (Extensible Markup Language, Lenguaje de Marcado de Extensible)**

Analiza la definición de XML en la WiKipedia. Extrae al menos tres ideas del siguiente texto:

Extensible Markup Language

Artículo principal: Lenguaje de marcado

XML, siglas en inglés de *eXtensible Markup Language*, traducido como "Lenguaje de Marcado Extensible" o "Lenguaje de Marcas Extensible", es un meta-lenguaje que permite definir lenguajes de marcas desarrollado por el *World Wide Web Consortium*(W3C) utilizado para almacenar datos en forma legible. Proviene del lenguaje SGML y permite definir la gramática de lenguajes específicos (de la misma manera que HTML es a su vez un lenguaje definido por SGML) para estructurar documentos grandes. A diferencia de otros lenguajes, XML da soporte a bases de datos, siendo útil cuando varias aplicaciones deben comunicarse entre sí o integrar información.[1]

XML no ha nacido únicamente para su aplicación en Internet, sino que se propone como un estándar para el intercambio de información estructurada entre diferentes plataformas. Se puede usar en bases de datos, editores de texto, hojas de cálculo y casi cualquier cosa imaginable.

XML es una tecnología sencilla que tiene a su alrededor otras que la complementan y la hacen mucho más grande, con unas posibilidades mucho mayores. Tiene un papel muy importante en la actualidad ya que permite la compatibilidad entre sistemas para compartir la información de una manera segura, fiable y fácil.

https://es.wikipedia.org/wiki/Extensible_Markup_Language

- **Enlaces de interés**

JSON
Objetos JSON
W3C JSON
W3C XML

2. Introducción XML

2.1 Definición

XML (Extensible Markup Language, Lenguaje de Marcado de Extensible) es un **lenguaje de marcas** que surge de SGML y permite definir una gramática para lenguajes específicos. Nace como estándar para el **intercambio de información estructurada** entre distintas plataformas y es muy utilizado para el intercambio de datos entre programas y en la web.

La forma del XML es muy parecida a la de un documento HTML. Los datos contenidos en las páginas XML están estructurados mediante etiquetas del tipo `<etiqueta>....</etiqueta>.`

La diferencia es que XML puede definir una gramática y etiquetas propias.

Ejemplo:

```
<?xml version="1.0" encoding="UTF-8"?>
<cartelera>
    <película>
       <título>La Quimera del Oro</título>
       <año>1925</año>
       <proyección hora="22:30">lunes</proyección>
       <proyección hora="23:00">jueves</proyección>
    </película>
    <película>
       <título>El Maquinista de la General</título>
       <año>1926</año>
       <proyección hora="21:00">martes</proyección>
  </película>
</cartelera>
```

2.2 Ejemplos de uso XML

El uso de XML está muy extendido, algunos ejemplos:

- Transferencia y organización de información

  Permite separar los datos de la presentación. XML no indica cómo presentar la información, un mismo documento XML puede presentarse de muchas formas. Es muy utilizado junto a aplicaciones HTML, donde XML guarda o transporta los datos, y HTML les da formato y los muestra.

  Cuando los datos cambian, no es necesario tener que editar el archivo HTML (XML es un fichero independiente).

  Con los lenguajes de programación web como PHP o Javascript se pueden leer archivos XML y actualizar el contenido de cualquier página HTML.

- Lenguajes derivados de XML

  Muchos lenguajes de propósito específico se han creado a partir de XML: SVG (Scalable Vector Graphics), MusicXML, DocBook, MathML…

- XML en Ofimática

OpenDocument es un formato de archivo, abierto y estándar, para almacenamiento de documentos ofimáticos como documentos de texto, hojas de cálculo, presentaciones y gráficos.

Actualmente es el formato de almacenamiento nativo de algunas aplicaciones de ofimática como OpenOffice y LibreOffice.

¿Cómo se almacena un documento ofimático mediante XML?

Abre el procesador de texto Writer, crea un documento nuevo.
Escribe la palabra “Hola”
Guarda el documento como prueba.odt.
En el directorio donde esté el documento, renombra manualmente el fichero a prueba.zip.
Descomprímelo.
Observa que hay varios ficheros XML.
Abre content.XML. Observa:

- Contiene etiquetas
- Existen declaraciones de tipo de letra
- Existen referencias a estándares
- Está el contenido del documento (“Hola”)

2.3 Características

→ XML es un metalenguaje

No es un lenguaje en sí, sino una manera de definir muchos lenguajes con diferentes necesidades. Permite definir etiquetas y atributos propios.

→ No incluye ninguna información relativa al formato (es un lenguaje de marcas descriptivo)

Las etiquetas indican el significado de los datos en lugar del formato con el que se van a visualizar. De esta forma **se independiza el contenido y formato de presentación**. Si quiere dar formato al documento se puede asociar una hoja de estilo.

→ XML es un estándar estructurado para escribir datos en un fichero de texto

En la práctica, la mayoría de la información útil no aparece aislada, sino que lo hace de forma estructurada y organizada.

Los diccionarios, guías, enciclopedias…son colecciones de datos que serían inútiles si no estuvieran organizadas de acuerdo con unas determinadas reglas.

XML es un conjunto de reglas, normas y convenciones para diseñar formatos de texto de forma que produzca ficheros fáciles de generar y de leer.

→ XML no requiere licencias, es independiente de la plataforma y tiene un amplio soporte

→ XML integra una familia de tecnologías

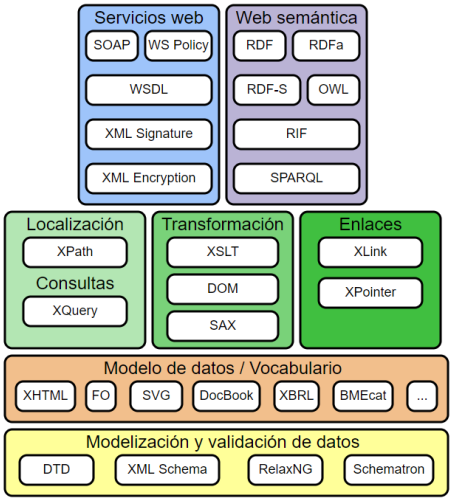
Las tecnologías XML básicas desarrolladas por el W3C son las siguientes:

XML Namespaces (Espacios de nombres XML): Define los mecanismos para permitir que en un mismo documento se puedan utilizar elementos y atributos de diferentes lenguajes de marcas (resolviendo, por ejemplo, el problemas de que algunos nombres de elementos coincidan).

XML Infoset: Describe un modelo de datos abstracto para documentos XML a partir de elementos de información. Se utiliza en las especificaciones de lenguajes XML, para describir restricciones en el lenguaje.

Xpath Permite seleccionar los componentes de un documento XML y facilitar su acceso a los programas que procesan documentos XML.

XSLT Lenguaje de transformación de documentos XML a otros formatos (XML o no XML)



XSL Formatting Objects (XSL-FO): Lenguaje de marcas para formatear documentos XML que se usa, por ejemplo, para generar PDFs.

Xquery Lenguaje de consulta orientado a XML, que permite acceder, manipular y devolver fragmentos de documentos XML.

XML Signature: Define la sintaxis y las reglas de procesamiento para crear firmas digitales en documentos XML.

XML Encryption: Define la sintaxis y las reglas de procesamiento para encriptar documentos XML.

→ XML tiene sintaxis estricta

En HTML si hay algún error en la forma de usar las etiquetas, el navegador intenta solucionarlo. Por ejemplo si falta una etiqueta de cierre, el navegador la asume (aunque puede que no en el sitio correcto), o si un atributo o etiqueta son incorrectos simplemente los ignora, sin afectar al resto del documento.

XML Errors Will Stop You

Errors in XML documents will stop your XML applications.

The W3C XML specification states that a program should stop processing an XML document if it finds an error. The reason is that XML software should be small, fast, and compatible.

HTML browsers are allowed to display HTML documents with errors (like missing end tags).

With XML, errors are not allowed.

https://www.w3schools.com/xml/xml_validator.asp

```
<!DOCTYPE html>
<html>
<head>
<meta charset="utf-8">
</head>
<body>
<div>
<p>Texto</p>

</body>
</html>
```

Ejemplo de documento HTML con error

(falta).

La visualización en pantalla es correcta

```
Texto
```

En cambio, **si hay algún error en XML, el documento es inválido en su totalidad:**

```
<?xml version="1.0" encoding="iso-
  8859-1"?>
<curso>
  <alumno>
      <nombre>Juan Antonio</nombre>
      <apellido>López</apellido>
      <dni>123456789</dni>
  </alumno>
      <nombre>Julia</nombre>
      <apellido>Martínez</apellido>
      <dni>987654321</dni>
  </alumno>
```

Ejemplo de documento XML con error (falta </curso>).

Aparece un error

Error de lectura XML: no se encuentra elemento

Actividad 2: Primer XML

2.1 Definir documento XML con un editor de texto sencillo
2.2 Visualizar en un navegador

2.4 Herramientas básicas de XML

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

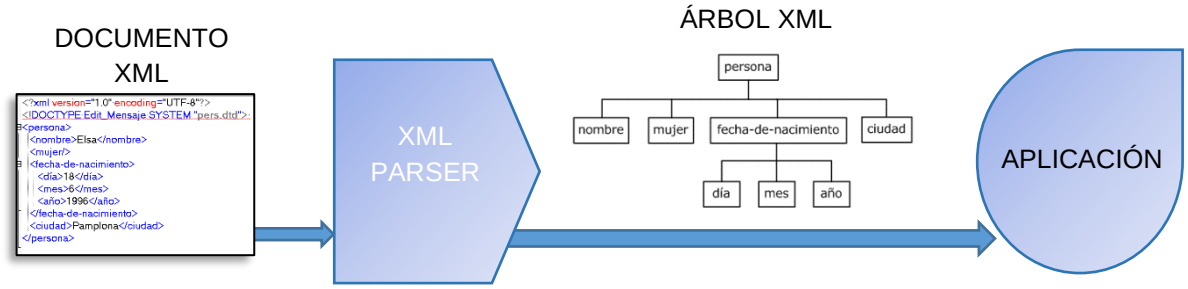
- **Editor XML**

  Una característica de los lenguajes de marcas es que se basan en la utilización de ficheros de **texto plano** por lo que se puede utilizar cualquier **procesador de texto** para construir un documento XML.

  Sin embargo, existen editores especializados que contienen funcionalidades adicionales para este tipo de documentos. Por ejemplo: Amaya (W3C) y xml-copy-editor

- **Procesador XML (*XML parser*)**

  Cuando una aplicación necesita leer un documento XML, la aplicación recurre a un procesador XML. El procesador XML (o analizador XML, en inglés *XML parser*) es el que lee el documento, analiza el contenido y le pasa la información en un formato estructurado a la aplicación. La recomendación XML especifica lo que debe hacer el procesador XML, pero no entra en lo que hace después la aplicación con esa información.



2.5 Gramáticas y vocabularios

Un vocabulario XML es el conjunto de reglas que debe cumplir para ser **válido** (qué etiquetas y atributos pueden usarse, cómo debe ser el contenido de una etiqueta, las jerarquías válidas entre etiquetas…)

Para definir un vocabulario XML se emplea una gramática. Las gramáticas para documentos XML son:

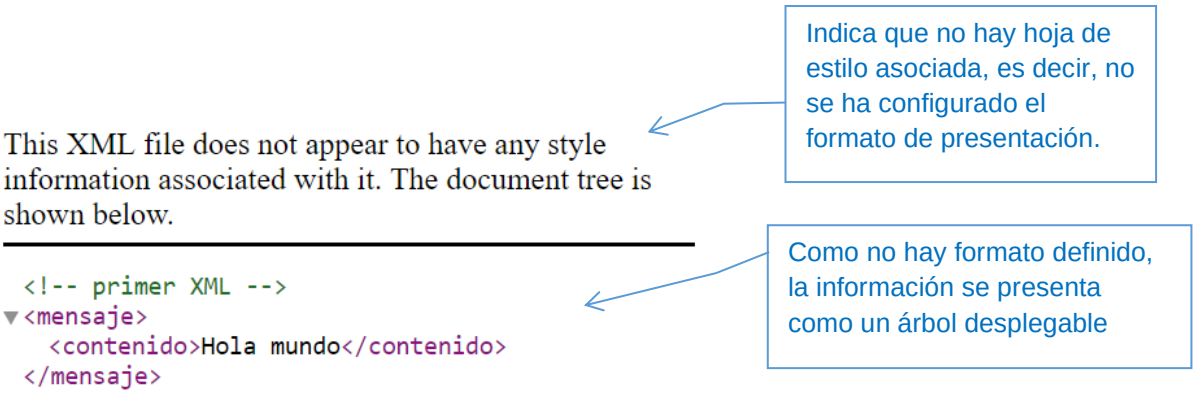
DTD (Document Type Definition). Se utiliza una definición de tipo de documento (<!DOCTYPE…>) que enlaza el fichero XML con una serie de reglas que definen el conjunto de etiquetas disponibles y la forma de utilizarlas. Es el modelo más antiguo, heredado del SGML.

XSD (XML Schema Document) Se emplea un archivo XSD basado en texto que define las reglas de validación para un archivo XML.

2.6 Presentación de un archivo XML

Los documentos XML no tienen información acerca de cómo mostrar los datos.

Cuando se abre un documento XML en un navegador, es posible que aparezca un mensaje similar a este:



Para formatear XML se puede emplear PHP, Javascript, o XSLT, entre otros. No es recomendable hacerlo en CSS.

XSLT son las siglas de eXtensible Stylesheet Language Transformations, y es el leguaje de hojas de estilo de XML.

2.7 Incorporar hoja de estilo CSS

Es posible incorporar una hoja de estilo CSS a un documento XML (es más recomendable emplear XSLT). La utilidad es dar formato a la información organizada. Para ello habría que incorporar la siguiente línea (debajo de la línea <?xml ?>):

```
<?xml-stylesheet href="ejemplo.css"?>
```

En el fichero “ejemplo.css” estarían las reglas css que se aplicarían a las etiquetas del fichero XML. Los selectores y las propiedades CSS funcionan de igual forma en CSS para HTML.

Recordatorio: en HTML se incluía una hoja de estilos mediante la siguiente línea, que debía estar dentro de la cabecera <head>

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

Ejemplo: ventas.xml

Actividad 3: CSS

3.1 Incorporar CSS a XML

3. Estructura XML

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE mensaje [
<!ELEMENT mensaje (contenido)>
<!ELEMENT contenido (#PCDATA)>
]>
<mensaje>
<contenido>Hola mundo</contenido>
</mensaje>
```

Declaración XML

Definición del DTD

Datos XML

- **Declaración o prólogo XML (*XML declaration*)**

  La declaración o prólogo XML es una etiqueta que comienza por <?xml y termina por ?> y que proporciona información sobre el propio documento XML. Aunque no es obligatoria es conveniente que aparezca, y **debe aparecer siempre al principio del documento**.

  No es una instrucción de procesamiento, pero tiene la misma sintaxis (empieza por <? y acaba por ?>).

  La declaración xml indica el juego de caracteres (tipo de codificación) del documento:

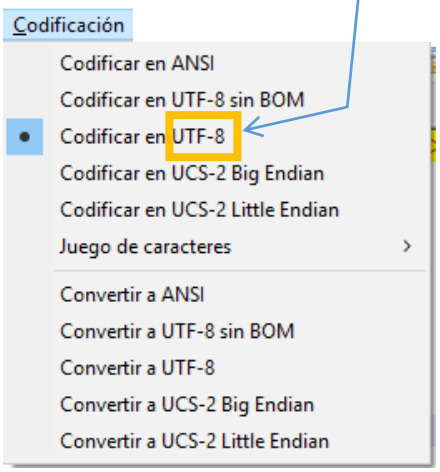
```
<?xml version="1.0" encoding="UTF-8"?>
```

Versión XML utilizada

En XML, el nombre del juego de caracteres se debe escribir en mayúsculas

Si la codificación en la declaración XML no coincide con la codificación del **documento o fichero, a la hora de guardarlo,** el procesador XML puede tener problemas leyendo el documento.

Codificación del fichero (aparece en alguna opción, dependiendo del editor utilizado)



Codificación de la declaración XML

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

Algunos editores, como XML Copy Editor, cambian automáticamente el valor del atributo encoding para que corresponda con el juego de caracteres del fichero.

- **Instrucciones de procesamiento (PI, *processing instruction*)**

  Una instrucción de procesamiento en una etiqueta que empieza por "<?" y acaba por "?>" y que contiene instrucciones dirigidas a las aplicaciones que leen el documento. Pueden aparecer en cualquier lugar del documento.

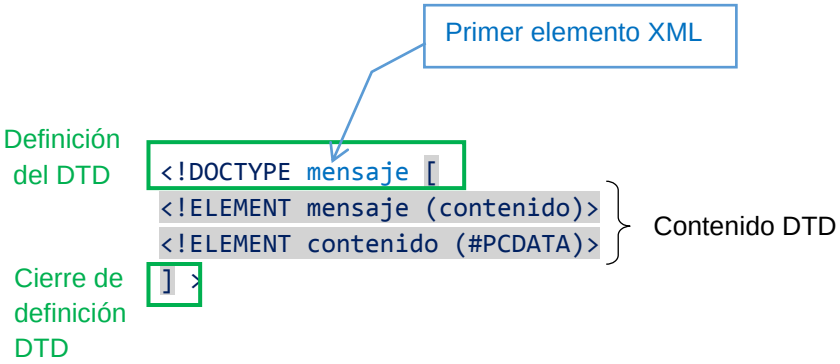
  Por ejemplo, destinada a una aplicación de tipo navegador web:

```
<?xml-stylesheet type="text/css" href="estilo.css" ?>
```

- **Definición de tipo de documento (DTD, *Document Type Definition*)**

  DTD es una **gramática** que define la estructura de un documento XML: los elementos, atributos, entidades, notaciones, etc, que pueden aparecer, el orden y el número de veces que pueden aparecer, cuáles pueden ser hijos de cuáles, etc.

  El procesador XML la utiliza para verificar si un documento es **válido**, es decir, si el documento cumple las reglas del DTD.



Lo veremos en detalle más adelante.

- **Comentarios**

  Un comentario es una etiqueta que comienza por <!-- y acaba por -->. Los comentarios no pueden estar dentro de otras marcas y no pueden contener los caracteres "--".

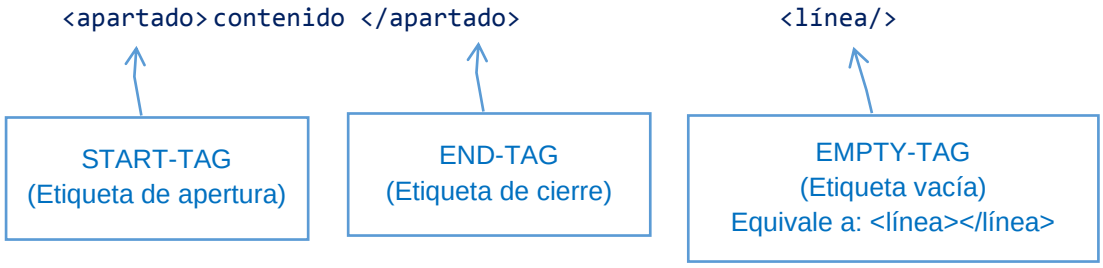
  Dentro de un comentario las entidades de carácter no se reconocen, es decir, sólo se pueden utilizar los caracteres del juego de caracteres del documento. Por ejemplo:

```
<!-- Esto es un comentario -->
```

4. Etiquetas, elementos y atributos

- **Etiquetas**

  Una etiqueta es un identificador que empieza por el carácter < y termina por >. Existen tres tipos de etiquetas:



- **Elementos**

  Un elemento es un componente lógico de un documento que cumple una de estas dos opciones:

  - Comienza por una etiqueta de apertura y termina por la etiqueta de cierre correspondiente.
  - Consiste en una única etiqueta vacía.

  El contenido de un elemento es todo lo que se encuentra entre las etiquetas de apertura y cierre, incluso si estos son también elementos en cuyo caso se llaman elementos hijos.

  Ejemplos:

```
<calle> Calle Laguna, 45 </calle>
```

Contenido del elemento “calle”

```
<dirección>
      <calle>Calle Laguna, 45</calle>
      <población>Madrid</población>
</dirección>
```

Contenido del elemento “dirección”: dos elementos hijos

```
<dirección>
     <calle>Calle Laguna, 45</calle>
     en la ciudad de
    <población>Madrid</población>
</dirección>
```

Contenido del elemento “dirección”: texto y dos elementos hijos (contenido mixto)

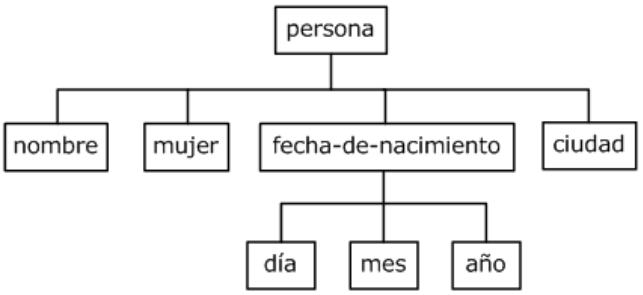
- **Jerarquía de elementos**

  Todo documento XML tiene que tener un único elemento raíz (padre) del que desciendan todos los demás.

  De esta forma, la estructura de cualquier documento XML se puede representar como un árbol invertido de elementos.

  Ejemplo:

```
<persona>
   <nombre>Elsa</nombre>
   <mujer/>
   <fecha-de-nacimiento>
      <día>18</día>
      <mes>6</mes>
      <año>1996</año>
   </fecha-de-nacimiento>
   <ciudad>Pamplona</ciudad>
</persona>
```



- **Atributos**

  Un atributo es un componente de las etiquetas que consiste en una pareja nombre (*name*) / valor (*value*) y **da información sobre el elemento que lo contiene**. Se puede encontrar en las etiquetas de apertura o en las etiquetas vacías, pero no en las de cierre. En una etiqueta no puede haber dos atributos con el mismo nombre.

  La sintaxis es siempre nombreAtributo="valorAtributo". Por ejemplo:

```
<prenda tipo="infantil">Vestido</prenda>

<profesor nombre="Bartolomé" apellidos="Pérez Bell" />
```

  Existen múltiples formas de organizar la información empleando etiquetas y atributos, que pueden ser equivalentes. Ejemplo:

Forma 1	`<animal>`
	`<tipo>Gato</tipo>`
	`<nombre>Canelo</nombre>`
	`</animal>`

Forma 2	`<animal tipo="Gato">Canelo</animal>`
Forma 3	`<animal tipo="Gato" nombre="Canelo" />`

5. Validación

5.1 Documentos bien formados

Un documento XML debe estar bien formado, es decir debe cumplir las reglas de sintaxis de la recomendación XML. Para que un documento esté bien formado, al menos debe cumplir los siguientes puntos:

- El documento contiene únicamente caracteres Unicode válidos.
- Se diferencia entre mayúsculas y minúsculas.
- Hay un elemento raíz que contiene al resto de elementos.
- Los nombres de los elementos y de sus atributos no contienen espacios.
- El primer carácter de un nombre de elemento o de atributo puede ser una letra, dos puntos (:) o guion bajo (_).
- El resto de caracteres pueden ser también números, guiones (-), guiones bajos (_) o puntos (.).
- Los caracteres "<" y "&" sólo se utilizan como comienzo de marcas.
- Las letras no inglesas (á, Á, ñ, Ñ...) están permitidas. Sin embargo, es recomendable no utilizarlas para reducir posibles incompatibilidades con programas que puedan no reconocerlas.
- Las etiquetas de apertura, de cierre y vacías están correctamente anidadas (no se solapan) y no falta ni sobra ninguna etiqueta de apertura o cierre.
- Las etiquetas de cierre coinciden con las de apertura (incluso en el uso de mayúsculas y minúsculas).
- Las etiquetas de cierre no contienen atributos.
- Ninguna etiqueta tiene dos atributos con el mismo nombre.
- Todos los atributos tienen algún valor.
- Los valores de los atributos están entre comillas (simples o dobles).
- No existen referencias en los valores de los atributos.
- Puede haber espacio o salto de línea al final de una etiqueta, pero no al principio.

  Correcto: `<elemento > </elemento >`

  Incorrecto: `< elemento> </ elemento>`

Los procesadores XML deben rechazar cualquier documento que contenga errores.

Detecta errores de sintaxis:

Elemento XML	Correcto/incorrecto Si es incorrecto indica la causa
`<persona>Eva</persona edad= "34">`	
`<letra  >R</letra>`	
`<variables x="2" y="5" x="6" />`	
`<nota-musica>la</nota-musica>`	
`<Ciudad>Pamplona</ciudad>`	
`<2colores>Rojo y azul</2colores>`	
`<color></color>`	
`<lugar="Brasil" año="1992"/>`	
`<_rojo>`	
`<día>18</dia>`	
`<persona edad=34>Eva</persona>`	
`<color principal>rojo` `</color principal>`	
`<_ciudad>Madrid</_ciudad>`	
`<mes>6<mes/>`	
`<mes>enero< /mes>`	

5.2 Documentos válidos

Para ser válido, un documento XML debe:

- **Ser un documento bien formado**
- incluir una referencia a una gramática
- incluir únicamente elementos y atributos definidos en la gramática
- cumplir las reglas gramaticales definidas en la gramática.

Existen distintos validadores XML.

5.3 Diferencia entre documento XML bien formado y documento válido

- Documento XML bien formado: está bien escrito sintácticamente, es decir, no hay etiquetas sin su correspondiente cierre, los nombres de elementos no contienen espacios, las comillas están bien utilizadas…
- Documento XML válido: está bien formado y además cumple todas las reglas especificadas en su método de validación: su **gramática**. Existen distintas herramientas que proporcionan el servicio de validación XML.

Ejemplo en lenguaje:

la casa tiene una gotera	➡	Es una frase que no está bien formada ya que tiene errores ortográficos (no empieza por mayúscula ni tiene punto final). Como consecuencia no es válida
La casam tiene una gotera.	➡	Es una frase que está bien formada pero no es válida ya que *Casam* no existe en el idioma español.
La casa tiene una gotera.	➡	Es una frase que está bien formada y es válida.

Entorno de trabajo: uso de XML Copy Editor

Actividad 4. Documentos bien formados

4.1 Corregir documentos XML
4.2 Crear un documento XML bien formado

Actividad 5: Documentos bien formados y válidos

5.1 Fichero XML no bien formado
5.2 Fichero XML no válido

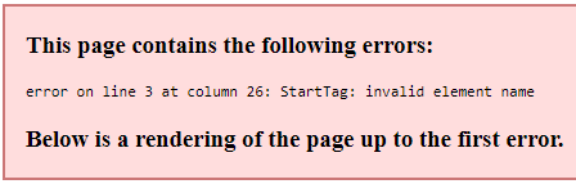
6. Incluir texto en XML

6.1 Entidades internas

En XML hay algunos caracteres que pueden dar problemas al incorporarse dentro del contenido, y como resultado el documento puede no estar bien formado.

Ejemplo con error:

```
<?xml version="1.0" encoding="UTF-8"?>
<prueba>
  <texto>Los caracteres < y & no pueden escribirse
  si no es como comienzo de marcas</texto>
</prueba>
```



Para solucionarlo se pueden emplear **referencias a entidades internas**, cuya codificación coincide con la de HTML. El formato es: `&caracteres;`

Referencia a entidad	Carácter
&lt;	<
&gt;	>
&amp;	&
&apos;	'
&quot;	"

Ejemplo correcto:

```
<?xml version="1.0" encoding="UTF-8"?>
<prueba>
  <texto>Los caracteres &lt; y &amp; no pueden escribirse
  si no es como comienzo de marcas</texto>
</prueba>
```

El contenido del elemento “texto” es:

> Los caracteres < y & no pueden escribirse si no es como comienzo de marcas

Pueden emplearse en el contenido de un elemento o en el valor de un atributo:

```
<elemento>John &amp; Ben</elemento>
<elemento nombre="John &amp; Ben" />
```

Nota: existe la posibilidad de definir entidades internas propias a través de DTD.

6.2 Secciones CDATA (CDATA *section*)

Una sección CDATA es una etiqueta que comienza por <![CDATA[y termina por]]> y cuyo contenido el procesador XML no interpreta como marcas sino como texto. **Se emplea en el contenido de los elementos**.

Es decir, que si aparecen los caracteres especiales (< & " ') en una sección CDATA, el procesador XML no interpreta que empieza una marca, sino que lo considera un carácter más.

Se suele utilizar en documentos en los que aparecen muchas veces esos caracteres especiales para no tener que estar utilizando las referencias a entidades (< y &) que dificultan la lectura del documento.

Ejemplo:

```
<?xml version="1.0" encoding="UTF-8"?>
<prueba>
  <texto><![CDATA["Los caracteres < y & no pueden escribirse
  si no es como comienzo de marcas"]]></texto>
</prueba>
```

El contenido del elemento “texto” es:

> Los caracteres < y & no pueden escribirse si no es como comienzo de marcas

Actividad 6. Referencias a entidades internas y CDATA

6.1 Crear un documento bien formado