UD1.2 Evolución de los lenguajes de marcas

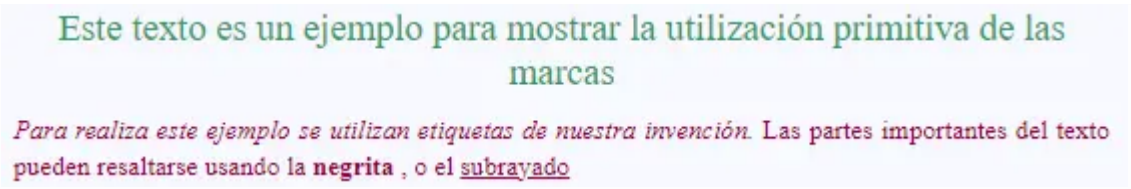
Introducción

En los años 70 surgieron unos lenguajes informáticos (distintos de los lenguajes de programación) orientados a la **gestión de información**. Con el desarrollo de los editores y procesadores de texto:

- Aparecieron los primeros lenguajes informáticos especializados en tareas de **descripción y estructuración** de información: los **lenguajes de marcas**.
- Paralelamente, también emergieron otros lenguajes informáticos orientados a la **representación, almacenamiento y consulta** eficiente de grandes cantidades de datos: **lenguajes y sistemas de bases de datos**.

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

> ⓘ **EJEMPLO**
>
> Código de marcas anterior a GML (etiquetas de invención propia):
>
> `<times 14><color verde><centrado> Este texto es un ejemplo para mostrar la utilización primitiva de las marcas</centrado></color> </times 14>`
>
> `<color granate><times 10><cursiva>Para realiza este ejemplo se utilizan etiquetas de nuestra invención. </cursiva>`
>
> `Las partes importantes del texto pueden resaltarse usando la`
>
> `<negrita>negrita</negrita>, o el <subrayar>subrayado</subrayar> </times 10></color>`



Posteriormente, se añadieron este tipo de características, como medio de presentación a la pantalla. Los códigos de estilo de visualización anteriores ya no aparecían, y se empleaban otros medios para marcados, distintos de la inclusión a mano de cadenas formateadoras. Ese proceso se automatizó y bastaba pulsar una combinación de teclas (o un botón) para lograr los resultados requeridos. Aunque esto era sólo una abstracción, para su uso interno, las aplicaciones seguían utilizando marcas para delimitar aquellas partes del texto que tenían un formato especial.

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

GML

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

Para resolver este problema, en los años 60, IBM encargó a Charles F. Goldfab la construcción de un sistema de edición, almacenamiento y búsqueda de documentos legales. Tras analizar el funcionamiento de la empresa, llegaron a la conclusión de que, para realizar un buen procesado informático de los documentos, había que establecer un formato estándar para todos los documentos que se manejaban en la empresa. Con ello, se lograba gestionar cualquier documento en cualquier departamento y con cualquier aplicación, sin tener en cuenta dónde ni con qué se generó el documento. Dicho formato tenía que ser válido para los distintos tipos de documentos legales que utilizaba la empresa, por tanto, debía ser flexible para que se pudiera ajustar a las distintas situaciones.

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

EJEMPLO

```
:h1.Capítulo 1: Introducción
:p.GML soporta herencia, como
:ol.
:li.Listas ordenadas (como esta),
```

`:li.Listas no ordenadas, y`
`:li.Listas de definición`
`:eol.`
`como también estructuras simples.`
`:p.Simplificación del marcado (más tarde generalizado y formalizado como SGML),`
`permite omitir las etiquetas de cierre para elementos "h1" y "p".`

SGML

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

EJEMPLO DE DOCUMENTO SGML

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

HTML

Entre los años 1989 y 1990, Tim Berners-Lee creó el WWW (*World Wide Web*) y se encontró con la necesidad de organizar, enlazar y compatibilizar gran cantidad de información procedente de diversos sistemas. Para resolverlo, creó un **lenguaje de descripción de documentos** llamado HTML, que, en realidad, era una combinación de dos estándares ya existentes:

- **ASCII** (*American Standard Code for Information Interchang*): es el formato que cualquier procesador de textos sencillo puede reconocer y almacenar. Por tanto, es un formato que permite la trasferencia de datos entre diferentes ordenadores.
- **SGML**: lenguaje que permite dar estructura al texto, resaltando los títulos o aplicando diversos formatos al texto.

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

HTML se convirtió en un **estándar general** para la **creación de páginas web**. Además, desde su creación, tanto las herramientas de software como los navegadores (que permiten visualizar páginas HTML) no han parado de mejorar.

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

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

EJEMPLO DE DOCUMENTO HTML

```
<html>
    <head>
        <title> Ejemplo de código HTML</title>
    </head>

    <body>
        <p></p>
        <p>
            <b>23 de julio de 2020</b>
        </p>
        <p><b> Bienvenido al modulo de “Lenguajes de Marcas y
Sistemas de Gestión de Información” </b></p>
        <p> En este curso aprender&aacute;s, entre otras cosas:
<br/>
        <ul>
            <li>Las ventajas que ofrece XML </li>
            <li>La creaci&oacute;n de documentos bien formados
</li>
            <li>La creaci&oacute;n de DTD</li>
        </ul>
        </p>
    </body>
</html>
```

En un navegador web se visualizaría así:

XML

Para resolver los problemas de HTML, el W3C (*World Wide Web Consortium*) establece, en 1998, el estándar internacional XML (*eXtensible Markup Language*), un **lenguaje de marcas puramente estructural** que no incluye ninguna información relativa al diseño. A diferencia de HTML, las etiquetas indican el significado de los datos en lugar del formato con el que se van a visualizar los datos.

XML es un estándar para el **intercambio de datos** en la **web** y otras aplicaciones, aunque está siendo substituído paulatinamente por JSON (*JavaScript Object Notation*).

Características

XML es un metalenguaje caracterizado por:

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

Estándares relacionados

En realidad, XML es un conjunto de estándares relacionados entre sí. Algunos de ellos son:

- **XSL** (*eXtensible Style Language*): permite definir hojas de estilo para los documentos XML.
- **XSLT** (*XSL Transformations*): permite transformar un documento XML a otro tipo.
- **XLink** (*XML Linking Language*): determina aspectos sobre los enlaces entre documentos XML.
- **XPath** (*XML Path*): permite realizar búsquedas dentro de un documento XML utilizando expresiones.
- **XML Namespaces**: proveen un contexto al que se aplican las marcas de un documento de XML y que sirve para diferenciarlas de otras con idéntico nombre válidas en otros contextos.
- **XML Schemas**: esquemas que permiten definir restricciones que se aplicarán a un documento XML. Actualmente los más usados son los DTD.

> **EJEMPLO DE XML**

```xml
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE biblioteca>
<biblioteca>
    <ejemplar tipo_ejem="libro" titulo="XML practico"
editorial="Ediciones Eni">
       <tipo> <libro isbn="978-2-7460-4958-1" edicion="1"
paginas="347"></libro> </tipo>
       <autor nombre="Sebastien Lecomte"></autor>
       <autor nombre="Thierry Boulanger"></autor>
       <autor nombre="Ángel Belinchon Calleja"
funcion="traductor"></autor>
       <prestado lector="Pepito Grillo">
           <fecha_pres dia="13" mes="mar" año="2009">
</fecha_pres>
           <fecha_devol dia="21" mes="jun" año="2009">
</fecha_devol>
       </prestado>
   </ejemplar>
   <ejemplar tipo_ejem="revista" titulo="Todo Linux 101.
Virtualización en GNU/Linux" editorial="Studio Press">
       <tipo>
           <revista>
               <fecha_publicacion mes="abr" año="2009">
</fecha_publicacion>
           </revista>
       </tipo>
       <autor nombre="Varios"></autor>
       <prestado lector="Pedro Picapiedra">
           <fecha_pres dia="12" mes="ene" año="2010">
</fecha_pres>
       </prestado>
   </ejemplar>
</biblioteca>
```

Otros lenguajes relacionados: JSON, YAML, TOML

Aunque no son lenguajes de marcado como XML, **JSON**, **YAML** y **TOML** son formatos de serialización de datos que se utilizan comúnmente en aplicaciones modernas, y que están estrechamente relacionados con los lenguajes de marcado.

Qué es un formato de serialización de datos

Un formato de serialización de datos es una manera de estructurar y codificar datos para que puedan ser fácilmente almacenados o transmitidos y luego reconstruidos en su forma original. Estos formatos son especialmente útiles en la comunicación entre diferentes sistemas o componentes de software.

Se diferencian de los lenguajes de marcado en que no se centran en la presentación o el formato del contenido, sino en la representación estructurada de los datos.

JSON

JSON (*JavaScript Object Notation*) es un formato ligero de intercambio de datos, fácil de leer y escribir para los humanos y fácil de analizar y generar para las máquinas. Se basa en un subconjunto del lenguaje de programación JavaScript, pero es independiente del lenguaje y se utiliza en muchos lenguajes de programación.

> ⓘ **EJEMPLO DE DOCUMENTO JSON**

```
{
   "biblioteca": {
       "ejemplares": [
           {
               "tipo_ejem": "libro",
               "titulo": "XML practico",
               "editorial": "Ediciones Eni",
               "tipo": {
                   "libro": {
                       "isbn": "978-2-7460-4958-1",
                       "edicion": "1",
                       "paginas": 347
                   }
               },
               "autores": [
                   "Sebastien Lecomte",
                   "Thierry Boulanger",
                   {
                       "nombre": "Ángel Belinchon Calleja",
                       "funcion": "traductor"
                   }
               ],
               "prestado": {
                   "lector": "Pepito Grillo",
                   "fechas": {
                       "pres": "2009-03-13",
                       "devol": "2009-06-21"
                   }
               }
           },
           {
               "tipo_ejem": "revista",
               "titulo": "Todo Linux 101. Virtualización en
GNU/Linux",
               "editorial": "Studio Press",
               "tipo": {
                   "revista": {
                       "fecha_publicacion": "2009-04"
                   }
               },
               "autor": "Varios",
               "prestado": {
                   "lector": "Pedro Picapiedra",
                   "fecha_pres": "2010-01-12"
               }
           }
       ]
```

```
    }
}
```

YAML (*YAML Ain't Markup Language*) es un formato de serialización de datos legible por humanos, que se utiliza comúnmente para archivos de configuración y en aplicaciones donde los datos se almacenan o transmiten. Es un superconjunto de JSON, lo que significa que cualquier documento JSON válido también es un documento YAML válido.

ⓘ EJEMPLO DE DOCUMENTO YAML

```yaml
biblioteca:
  ejemplares:
    - tipo_ejem: libro
      titulo: XML practico
      editorial: Ediciones Eni
      tipo:
        libro:
          isbn: 978-2-7460-4958-1
          edicion: "1"
          paginas: 347
      autores:
        - Sebastien Lecomte
        - Thierry Boulanger
        - nombre: Ángel Belinchon Calleja
          funcion: traductor
      prestado:
        lector: Pepito Grillo
        fechas:
          pres: 2009-03-13
          devol: 2009-06-21
    - tipo_ejem: revista
      titulo: Todo Linux 101. Virtualización en GNU/Linux
      editorial: Studio Press
      tipo:
        revista:
          fecha_publicacion: 2009-04
      autor: Varios
      prestado:
        lector: Pedro Picapiedra
        fecha_pres: 2010-01-12
```

YAML es muy utilizado en la configuración de aplicaciones, en la definición de flujos de trabajo y en la serialización de datos. Por ejemplo, las automatizaciones en

herramientas de CI/CD, como Github, a menudo utilizan YAML para definir sus pipelines.

TOML

Por su parte, **TOML** (Tom's Obvious, Minimal Language) es otro formato de serialización de datos que se centra en la simplicidad y la legibilidad. Se utiliza comúnmente para archivos de configuración y es conocido por su sintaxis clara y concisa.

```
[libro]
titulo = "XML practico"
autor = ["Sebastien Lecomte", "Thierry Boulanger"]
editorial = "Ediciones Eni"
isbn = "978-2-7460-4958-1"
edicion = "1"
paginas = 347

[prestado]
lector = "Pepito Grillo"
fechas = { pres = "2009-03-13", devol = "2009-06-21" }

[[revistas]]
titulo = "Todo Linux 101. Virtualización en GNU/Linux"
editorial = "Studio Press"
fecha_publicacion = "2009-04"
autor = "Varios"
prestado = { lector = "Pedro Picapiedra", fecha_pres = "2010-01-
12" }
```

XML vs HTML

A continuación, se muestra una comparativa entre los lenguajes XML y HTML:

XML	HTML
Es un perfil de SGML.	Es una aplicación de SGML.
Especifica cómo deben definirse conjuntos de etiquetas aplicables a un tipo de documento.	Aplica un conjunto limitado de etiquetas sobre un único tipo de documento.

XML	HTML
Modelo de hiperenlaces complejo.	Modelo de hiperenlaces simple.
El navegador es una plataforma para el desarrollo de aplicaciones.	El navegador es un visor de páginas.
Fin de la guerra de los navegadores y etiquetas propietarias.	Problema de la falta de compatibilidad y diferencias entre navegadores web.

EJEMPLO XML VS. HTML

Ejemplo de un documento **XML**:

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

Visualización del documento **XML** en un navegador web:

Ejemplo de un documento **HTML**:

```
<!DOCTYPE html>
<html>
   <head>
       <title>Libro</title>
   </head>
   <body>
       <h3>XML practico</h3><br>
       <p>autores: Sebastien Lecomte, Thierry Boulanger</p>
       <ul>
           <li>editorial: Ediciones Eni</li>
           <li>isbn:978-2-7460-4958-1</li>
           <li>edicion: 1 </li>
           <li>paginas: 347</li>
       </ul>
   </body>
</html>
```

XML vs SGML

A continuación, se muestra una comparativa entre los lenguajes XML y SGML:

XML	SGML
Uso sencillo.	Uso complejo.
Trabaja con documentos bien formados. No exige que estén validados.	Solo trabaja con documentos válidos.
Facilita el desarrollo de aplicaciones de bajo coste.	Su complejidad hace que las aplicaciones informáticas para procesar SGML sean muy costosas.
Es muy utilizado en informática y en más áreas de aplicación.	Solo se utiliza en sectores muy específicos.
Compatibilidad e integración con HTML.	No hay una compatibilidad con HTML definida.

XML	SGML
Formato y estilos fáciles de aplicar.	Formateo y estilos relativamente complejos.
No usa etiquetas opcionales.	

XML, concepto de documento bien formado y documento válido

En el contexto de XML, es importante entender la diferencia entre un documento **bien formado** y un documento **válido**.

Documento bien formado

Un documento **bien formado** es aquel que **verifica las reglas** establecidas por la recomendación del W3C.

Documento válido
Un documento **válido** es aquel, que además de estar bien formado, **verifica las restricciones de otro elemento del que depende para su interpretación**.

En la sección propia de XML, se dará más información sobre estos conceptos y se **profundizará en su importancia**.

Etiquetas

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

Por ejemplo, un *texto sin formato* (también llamado *en texto plano*) sería el siguiente:

`Lenguaje de marcas`

El mismo texto, pero con formato, puede tener el siguiente aspecto:

```
<modulo>Lenguaje de marcas</modulo>
```

La interpretación del ejemplo anterior es que *Lenguaje de marcas* es un módulo, algo que con el texto sin formato no es posible interpretar.

Sintaxis

Las etiquetas se escriben encerradas entre signos "*menor que*" (`<`) y "*mayor que*" (`>`). Por ejemplo:

`<etiqueta>`

Normalmente, se utilizan dos etiquetas:

- Etiqueta de **apertura** o inicio.
- Etiqueta de **cierre** o fin.

La etiqueta de apertura indica que el efecto que queremos aplicar empieza en ese momento y la de cierre que el ha terminado. La única diferencia entre ambas es que la de cierre lleva una barra inclinada (`/`) antes del nombre.

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

EJEMPLO

Etiqueta HTML de subrayado (*underline*):

```
<u>Esto está subrayado</u>
```

En el navegador web, el texto se verá como se muestra a continuación:

MINÚSCULAS

Las últimas especificaciones emitidas por el W3C indican la necesidad de que las **etiquetas** se escriban siempre en **minúsculas** para considerar que el documento está correctamente creado.

Herramientas de edición

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

- **Editores XML**. Por ejemplo, Visual Studio Code.
- **Procesadores XML**. Por ejemplo, un navegador web.

Editores XML

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

Tipos de ficheros

Se puede diferenciar dos grandes grupos de ficheros:

- **De texto**: la información se guarda codificada en texto utilizando un sistema de codificación de caracteres (ASCII, UTF-8, etc.). Son menos propensos a corromperse.
- **Binarios**: la informacion se guarda codificada como una secuencia de bytes (en binario). Los ficheros binarios pueden guardar múltiples tipos de datos en un mismo archivo. Ejemplos de ficheros binarios son las imágenes o los vídeos.

Para crear documentos XML complejos e ir añadiendo datos es conveniente usar algún editor XML. Éstos nos ayudan a crear estructuras y etiquetas de los elementos usados en los documentos, además algunos incluyen ayuda para la creación de otros elementos como DTD, hojas de estilo CSS o XSL.

El W3C ha desarrollado un editor de HTML, XHTML, CSS y XML gratuito llamado Amaya, pero su desarrollo fue discontinuado en 2012.

En la actualidad existen multitud de editores de texto más modernos, entre los cuales podemos encontrar **Visual Studio Code**.

Procesadores XML

Los procesadores XML permiten leer los documentos XML y acceder a su contenido y estructura. Un procesador es un conjunto de módulos de software, entre los que se encuentra un *parser*, el cual comprueba que el documento cumple las normas establecidas para que pueda abrirse.

Parser

Un ***parser*** es un software que analiza un bloque de datos y extrae el significado de los diferentes elementos (**análisis sintático**) para poder procesarlos y realizar operaciones con ellos.

Los procesadores XML pueden ser:

- Validadores: obliga a trabajar solo con documentos válidos.
- No validadores: solo exigen que el documento esté bien formado.

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**, los cuales permiten generar archivos HTML a partir de documentos XML.

Para **interpretar** el código XML se puede utilizar cualquier **navegador**.

XML también se puede utilizar para el **intercambio de datos** entre aplicaciones. En este caso, hay que recurrir a motores independientes, que se ejecutan de forma transparente.