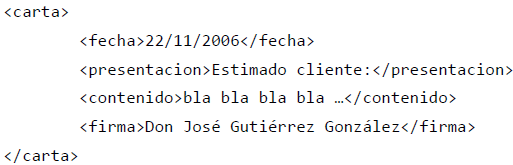
Tema 1: Reconocimiento de las características de lenguajes de marcas.

# 1.- Lenguajes de marcas.

Un lenguaje de marcas es **un modo de codificar un documento** donde, junto con el texto, **se incorporan etiquetas**, marcas o anotaciones **con información adicional** relativa a la estructura del texto o su formato de presentación.

Todos los lenguajes de marcas están definidos en un documento denominado DTD (Document Type Definition), en el que se establecen cuales son esas marcas, los elementos utilizados por dicho lenguaje y sus correspondientes etiquetas y atributos.

**Ejemplo de un documento realizado con un lenguaje de marcas**:



Aunque en la práctica, en un mismo documento pueden combinarse varios tipos diferentes de lenguajes de marca, **los lenguajes de marcas se clasifican de la siguiente forma**:

|  |  |
| --- | --- |
| **Tipo** | **Para qué sirven** |
| **De presentación** | Formatear textos. |
| **De procedimientos** | Orientado también a la presentación, pero el programa interpreta el código en el mismo orden en que aparece. |
| **Descriptivo o semántico** | Se estructura el documento, pero sin especificar cómo deben representarse. |

Ejemplos de lenguajes de marcado agrupados por su ámbito de utilización:

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

# 2.- Evolución de los lenguajes de marcas.

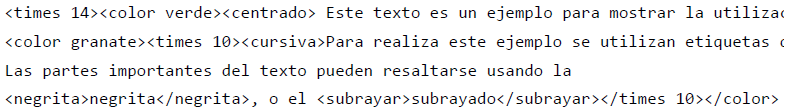
En los años 70, **surgen lenguajes orientados a la gestión de información** junto con el desarrollo de los editores y procesadores de texto: **los lenguajes de marcas**.

**Paralelamente**, también **surgen** otros **lenguajes informáticos orientados a** la representación, **almacenamiento y consulta eficiente de grandes cantidades de datos**: **lenguajes y sistemas de bases de datos**.

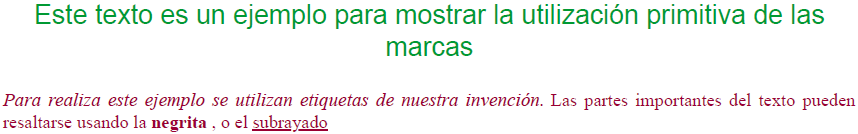
Los lenguajes de marcas **surgieron como lenguajes** **formados por el conjunto de códigos de formato que los procesadores de texto introducen en los documentos para dirigir el proceso de presentación (impresión) mediante una impresora**.

Inicialmente estos códigos de formato estaban ligados a las características de una máquina, programa o procesador de textos concreto.

**Ejemplo**: código de marcas anterior a GML, las etiquetas son de invención propia.



Al imprimirlo se obtendría:



**Posteriormente, se añadió nuevo medio de presentación la pantalla**. Hoy en día, podemos crear documentos de una forma mucho más automatizada, basta pulsar una combinación de teclas o pulsar un botón para lograr los resultados requeridos. Sin embargo, **internamente, las aplicaciones siguen utilizando marcas para delimitar aquellas partes del texto que tienen un formato especial**.

## 2.1.- GML (Generalized Markup Language).

**Uno de los problemas desde hace décadas** en la informática **es la falta de estandarización en los formatos de información**.

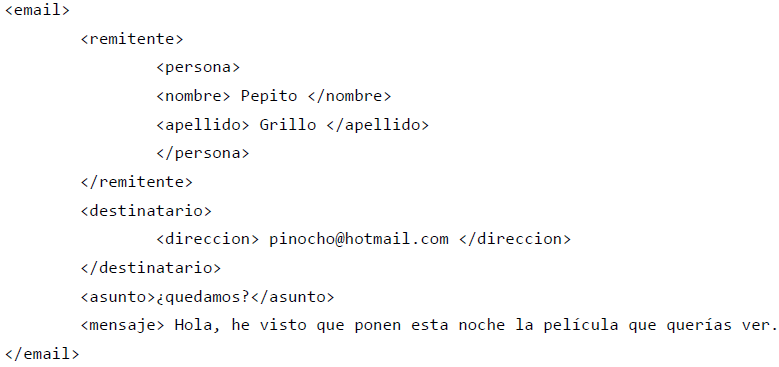
**Para resolver este problema, en los años sesenta** IBM encargó a Charles F. Goldfab la **construcción de un sistema de edición, almacenamiento y búsqueda de documentos legales**. **Con un formato estándar para todos los documentos** que se manejaban en la empresa, se lograba gestionar cualquier documento en cualquier departamento y con cualquier aplicación.

**Dicho formato, por tanto, tenía que ser flexible** para que se pudiera ajustar a las distintas situaciones. Este formato de documentos fue GML.

## 2.2.- SGML (Standard Generalized Markup Lenguage).

El formato GML evolucionó hasta que en 1986 dio lugar al **estándar ISO 8879, denominado SGML**. Era un **lenguaje muy complejo** y requería de unas herramientas de **software caras**.

Ejemplo:



## 2.3.- HTML (HyperText Markup Language).

En 1989/90, **Tim Berners-Lee creó el World Wide Web y** un lenguaje de descripción de documentos llamado **HTML**, el cual es una combinación de dos estándares ya existentes entonces:

* **ASCII**: es el formato que cualquier procesador de textos sencillo puede reconocer y almacenar. Por tanto, es un formato que permite la transferencia de datos entre diferentes ordenadores.
* **SGML**: lenguaje que permite dar estructura al texto, resaltando los títulos o aplicando diversos formatos al texto.

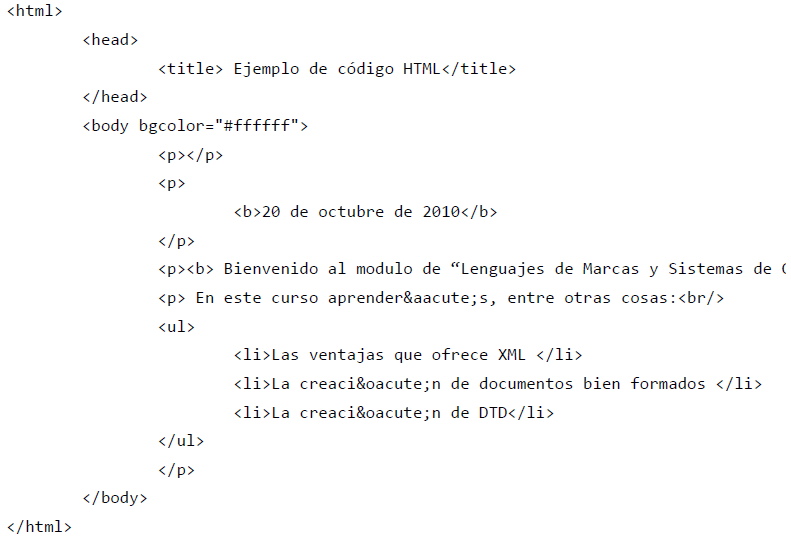
**HTML** **es una versión simplificada de SGML**. Solo utilizaba las **instrucciones absolutamente imprescindibles**. Se aceptó muy rápidamente ya que era muy fácil de comprender.

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

**Desventajas del lenguaje**:

* 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:



## 2.4.- XML (eXtensible Markup Language).

XML (estándar internacional), establecido en 1998 por W3C, es un lenguaje de marcas puramente estructural que **no incluye ninguna información relativa al diseño**. Se está convirtiendo rápidamente en estándar para el intercambio de datos en la Web.

**Las etiquetas indican el significado de los datos** **en lugar del formato** con el que se van a visualizar los datos.

**XML se caracteriza 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.

**XML es un conjunto de estándares relacionados entre sí**:

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

Ejemplo:



## 2.5.- Comparación de XML con 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** | Aplican un **conjunto limitado de etiquetas** sobre un único tipo de documento |
| 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 | El problema de la ‘no compatibilidad’ y las diferencias entre navegadores ha alcanzado un punto en el que la solución es dificil. |

**Ejemplo**:

|  |  |
| --- | --- |
|  |  |

## 2.6.- Comparación de XML con SGML.

|  |  |
| --- | --- |
| **XML** | **SGML** |
| **Su uso es sencillo** | Su uso es muy **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 más específicos** |
| **Compatibilidad** e integración **con HTML** | **No hay una compatibilidad con HTML** definida. |
| **Formateo y estilos fáciles de aplicar** | **Formateo y estilos relativamente complejos**. |
| **No usa etiquetas opcionales**. |  |

# 3.- Etiquetas.

Las etiquetas **se escriben encerradas entre ángulos < y >**. Normalmente se utilizan dos etiquetas: una de inicio y otra de fin para indicar que ha terminado el efecto que queríamos presentar.

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

**Ejemplo**: <u>Esto está subrayado</u>

Las últimas especificaciones emitidas por el W3C indican la **necesidad** **de que** **vayan** **escritas siempre en minúsculas**.

# 4.- Herramientas de edición.

Para trabajar en XML es necesario **editar** los documentos y luego **procesarlos**.

|  |  |
| --- | --- |
| **Editores XML** | Una característica de los lenguajes de marcas es que **se basan en la utilización de ficheros de texto plano**, por lo que **basta utilizar un procesador de texto normal y corriente** para construir un documento XML. |
| **Para crear documentos XML complejos**, es conveniente usar algún **editor** XML, para que nos ayude a crear estructuras y etiquetas de los elementos utilizados en los documentos. |
| El W3C ha desarrollado un editor JTML, XHTML, CSS y XML gratuito llamado **Amaya.** |
| **Procesadores XML** | Para interpretar el código XML se puede utilizar cualquier navegador. |
| Los procesadores de XML permiten leer los documentos XML y acceder a su contenido y estructura. |
| Es un conjunto de módulos de software entre los que se encuentra un parser (analizador) de XML que comprueba que el documento cumple con las normas establecidas para que pueda abrirse. |
| Dependiendo de las normas que verifique:   * Las necesarias para trabajar solo con documentos de tipo válido → **validadores**. * Solo exige que el documento esté bien formado → **no validadores**. |
| **Para publicar** un documento XML **en Internet** se utilizan los **procesadores XSLT**, que **permiten generar archivos HTML a partir de documentos XML**. |
| **XML se puede utilizar para el intercambio de datos entre aplicaciones**, por tanto, **existen motores independientes que se ejecutan sin que nos demos cuenta**. Entre ellos, destacan “XML para Java” de IBM, JAXP de Sun, etc. |

# 5.- XML: estructura y sintaxis.

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

**El XML ahorra tiempos de desarrollo y proporciona ventajas**, dotando a webs y aplicaciones de una **forma realmente potente de guardar la información**. Se ha convertido en un **formato universal**.

**Un documento XML** **es** un documento de texto con extensión “.xml” **compuesto de parejas de etiquetas**, **estructuradas en árbol**, que **describen** una función en **la organización del documento**. **Características básicas de XML**:

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

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

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

**El marcado se añade a un texto para estructurar el contenido del documento**. Esto le sirve a los ordenadores para “interpretar” los textos. Pero **los datos carácter son los que forman la verdadera información del documento XML**.

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

**Los documentos XML pueden tener comentarios**, mediante las cadenas **“<!-“ y “-->”**, que no serán interpretados por el intérprete XML. **Pueden ir en cualquier posición salvo**:

* Antes del prólogo.
* Dentro de una etiqueta.

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

## 5.1.- El prólogo.

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

|  |  |
| --- | --- |
| **Declaración XML** | En caso de incluirse, **ha de ser en la primera línea del documento**, si no, saltará un error que impide ser procesado el documento. |
| Al ser opcional, permite el procesamiento de documentos HTML y GML como si fueran XML. |
| Tiene **TRES FUNCIONES**:   * **Declaración de la versión de XML**:   <?xml versión= “1.0” ?>   * **Declaración de la codificación**: determina el conjunto de caracteres que se utiliza en el documento.   <?xml versión= “1.0” encoding=”iso-8859-1” ?>  Este iso (Latin-1) permite el uso de acentos o caracteres como la ñ. Para ver más, ver Anexo I.   * **Declaración de la autonomía del documento**: informa de si el documento necesita de otro para su interpretación. Para declararlo hay que definir el **prólogo completo**.   <?xml versión= “1.0” encoding=”iso-8859-1” standalone=”no” ?>  **Standalone=”no”**: documento independiente. |
| **Declaración del tipo de documento** | Define qué tipo de documento estamos creando para ser procesado correctamente.  **Sintaxis**: <!DOCTYPE Nombre\_tipo ...> |

### Anexo I. Códigos de los conjuntos de caracteres más importantes.

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

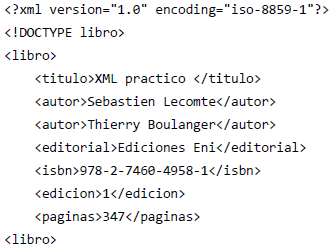
## 5.2.- El ejemplar. Los elementos.

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

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

**Para concretar más**: **el ejemplar es un solo y único elemento raíz** del que parten el resto de los elementos, a los cuales pertenecen los datos que almacena el documento XML.

**Ejemplo**: En este caso, el ejemplar es el elemento <libro>



La formación de elementos ha de cumplir ciertas **normas** **para que** queden perfectamente definidos y **pueda ser interpretado en los procesadores XML** sin errores:

* Debe existir un elemento raíz, y solo 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.
    - **Ejemplo**: <elemento></elemento> pasa a ser <elemento/>
* Al anidar elementos hay que tener en cuenta que no puede cerrarse un elemento que contenga algún otro elemento 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 “:”, ni por “xml”.
* **El contenido de los elementos no puede contener la cadena “]]>” por compatibilidad con SGML**.
* Además, no se pueden utilizar directamente los caracteres >, <, &, “ y ‘.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Carácter** | **Cadena** | **Carácter** | **Cadena** | **Carácter** | **Cadena** |
| > | &gt | & | &amp | ‘ | &apos |
| < | &lt | “ | &quot |  |  |

* **Para utilizar caracteres especiales**, como £, ©, ®, hay que usar las expresiones &#D; o &#H, donde D es número decimal y H es el hexadecimal correspondiente al carácter que se quiere representar en el código UNICODE.
  + **Euro en decimal**: &#8364
  + **Euro en hexadecimal**: &#x20AC.

**Debes conocer**: esta página para consultar la tabla con los caracteres ASCII, el nombre HTML y el número HTML de cada uno de ellos. [Códigos HTML - Tabla de caracteres y símbolos (ascii.cl)](https://ascii.cl/es/codigos-html.htm)

### 5.2.1.- Atributos.

**Permiten añadir propiedades a los elementos de un documento**. No pueden organizarse en ninguna jerarquía, **no pueden contener ningún otro elemento o atributo**.

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

Captura de pantalla de un celular con texto

Descripción generada automáticamente

Los atributos se definen y dan valor dentro de una etiqueta de inicio o de elemento vacío, **a continuación del nombre del elemento o de la definición de otro atributo siempre separado de ellos por un espacio**.

Los nombres de los atributos **han de cumplir las mismas reglas que los de los elementos**.

**Autoevaluación**: ¿Cuáles son los errores del siguiente documento XML?

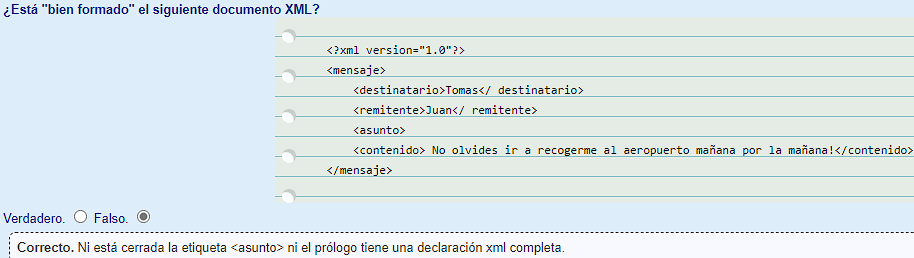
Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente con confianza media

# 6.- Documentos XML bien formados.

Todos los documentos XML bien formados deben verificar las **reglas sintácticas que define la recomendación del W3C**:

* El documento ha de tener **definido un prólogo** con la **declaración XML completa**.
* Existe un **único elemento raíz para cada documento**.
* Hay que **cumplir** las **reglas sintácticas del lenguaje XML para definir los distintos elementos y atributos** del documento.



# 7.- Utilización de espacios de nombres (namespaces) en XML.

Permiten definir la pertenencia de los elementos y los atributos de un documento XML al contexto de un vocabulario XML. **Así, 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**.

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** (documento) al que pertenece la etiqueta, seguido de dos puntos, :. Esto es:

<prefijo:nombre\_etiqueta></prefijo:nombre\_etiqueta>

Esta etiqueta se denomina **nombre cualificado**. En este nombre, no se pueden utilizar espacios ni caracteres especiales y no puede comenzar por un dígito.

Para utilizar este nombre cualificado, es necesario **definir 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.

**La sintaxis del URI es la siguiente**: <conexion>://<direccionservidor>/<apartado1>/<apartado2>/...

**Ejemplo**:

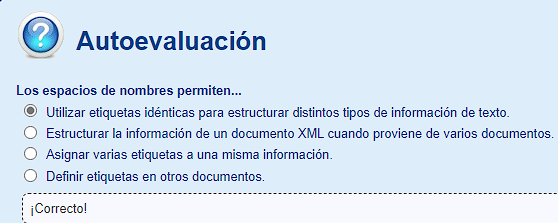
|  |  |
| --- | --- |
| **Alumno** |  |
| **Profesor** |  |
| **Documento de los miembros del curso DAW** | Aquí se puede ver el atributo xmlns, que sirve de nexo. |

**Ejemplo 2** del [vídeo de explicación](https://www.youtube.com/watch?v=BdVCve55Afw) de los espacios de nombres:

Interfaz de usuario gráfica, Texto

Descripción generada automáticamente

Dudas



**Para saber más**: los espacios de nombres tienen una recomendación en XML.

[Namespaces in XML 1.0 (Third Edition) (w3.org)](https://www.w3.org/TR/REC-xml-names/)